COLING 2016

The 26th International Conference on Computational Linguistics

Proceedings of COLING 2016: Technical Papers

December 11-16, 2016
Osaka, Japan

ISBN978-4-87974-702-0

Preface: General Chair

Welcome to COLING 2016 – the 26th International Conference on Computational Linguistics — held in Osaka, Japan! It is the third COLING in Japan after Tokyo (1980) and Kyoto (1994). It is a special pleasure for me to be General Chair (10 years after chairing the joint COLING-ACL 2006 in Sydney) of a COLING held in Japan, a country I love.

COLING is organised under the auspices of the International Committee on Computational Linguistics (ICCL, http://nlp.shef.ac.uk/iccl/index.html). ICCL is a very special committee, with no fixed rules and no funding, whose only function is to make sure that a COLING appears every two years and that it is a good and friendly conference.

I have participated to many COLINGs, since the one in Pisa in 1973. It was a COLING without email! I still remember when Antonio Zampolli (Local chair) received by Hans Karlgren (Program chair) a sketch of the program written by hand, almost unreadable, and asked me (very young at the time) to interpret it. I have seen COLINGs where submissions arrived on paper and many packages were sent around the world to area chairs, to be sent to reviewers, and all the results back again by normal mail. It seems impossible now.

COLING has changed over the years, together with the changes in our field. But it has always been important for ICCL to maintain the COLING "spirit": we always wanted COLING to be an inclusive and broad conference. We also want to underline that in our field "language" is important and we therefore pay special attention to having papers and workshops focusing on understanding language properties and complexities. Moreover, for us the social part of the conference is as important as the scientific one.

An outstanding competent and dedicated team has worked for the organisation of COLING 2016. I wish to warmly thank, also on behalf of ICCL, all the various Chairs, too many to mention them all here, for the wonderful work they have done. It has been a pleasure and a privilege for me to work together with all of them: they made my work as General chair very easy. But I owe a special thanks to Yuji Matsumoto and Rashmi Prasad, Program chairs, for their hard work in managing so smoothly an impressive number of submissions, many more than we expected. And I wish to express my deepest gratitude to the Local chairs – Eiichiro Sumita, Takenobu Tokunaga and Sadao Kurohashi – who have done a fantastic work with great dedication in all the various phases of the conference organisation, always keeping everything under control. Not an easy task, as I know too well!

I also want to thank the generosity of all the sponsors for their great support to COLING.

Last but not least, I thank the colleagues (so many) who submitted their work to COLING, the organisers of Workshops and Tutorials, the participants (more than 900 at the time of writing) and the many students among them. It is important that many young researchers can attend COLINGs. They show the great interest of our community in COLING.

I hope that you benefit not only from the scientific programme but also from the social parts of COLING. I hope you get from this COLING both new exciting ideas and also new friends.

Enjoy COLING 2016 in Japan!

Nicoletta Calzolari (ICCL, ILC-CNR and ELRA)

Preface: Program Chairs

It is with great pleasure that we welcome you to the 26th International Conference on Computational Linguistics (COLING 2016) in Osaka, Japan! COLING covers a broad spectrum of technical areas related to natural language and computation. This year, we received 1,039 valid submissions (from a total of 1127 submissions), of which we accepted 337 papers (32.4% acceptance rate). 134 papers were selected for oral presentation and 203 papers for poster presentation. No distinction is made in these proceedings between papers presented orally or as a poster, as they were not distinguished qualitatively but rather by judging the best mode for delivering the paper content.

To effectively cover the broad spectrum of topics included in the conference, we have 18 thematic areas, each chaired by two or more area chairs. We are extremely grateful to the area chairs, who led and monitored the reviewing and reviewer discussions, and sent us detailed recommendation reports resulting from the reviewing process, including best paper recommendations. We cannot thank enough the over 800 reviewers who have put in the requisite time and effort to carefully assess the very large number of submissions we received this year. Their dedication and commitment, and willingness to work with us even when there were tight time constraints, made the entire task proceed much more smoothly than we had hoped! Almost all papers were reviewed by at least three reviewers and we are very happy with the highly strong set of papers accepted for presentation. We thank all authors for their submissions describing their very commendable research, and hope that authors of papers we could not accept have nevertheless benefited from the feedback they received from reviewers.

We have structured the accepted submissions into ten sessions, with multiple thematic areas included in parallel, either for oral presentation or poster presentation. Only one session – the first session – does not have a parallel poster session. We are delighted to have four invited speakers to the conference: Joakim Nivre from Uppsala University: "Universal Dependencies – Dubious Linguistics and Crappy Parsing?"; Reiko Mazuka from RIKEN Brain Science Institute & Duke University: "Getting the Input Right: Refining our Understanding of What Children Hear"; Dina Demner-Fushman from the U.S. National Library of Medicine: "NLP to support clinical tasks and decisions", and Simone Teufel from University of Cambridge: "A Look at Computational Argumentation and Summarisation from a Text-Understanding Perspective".

We are extremely grateful to the members of the best paper committee, Tim Baldwin, Vincent Ng, and Hinrich Schütze, who agreed to put in extra time to select the two best papers at the conference. Best paper nominations were collected in a bottom-up fashion, with reviewers first providing their recommendation for each paper, and area chairs then collecting the positive recommendations, and upon their own assessment of the corresponding reviews and papers, selecting some or all to be forwarded to the PC chairs. PC chairs then invited the three experts to form a committee (chaired by the PC chairs) to select the two best papers from this set of nominated papers.

We would like to thank the many members of the organizing committee who have helped us in crucial ways at various stages of organizing the technical program – the General Chair, Nicoletta Calzolari; the Local Chairs, Eiichiro Sumita, Takenobu Tokunaga and Sadao Kurohashi; the Publication Chairs, Hitoshi Isahara and Masao Utiyama; the Publicity Chairs, Srinivas Bangalore, Dekai Wu and Antonio Branco; and the Web Master Akifumi Yoshimoto. Our special thanks go to Swapna Somasundaran for her voluntary help to recruit additional reviewers to handle the much larger than expected submissions to the conference. Last but not the least, we are grateful to the softconf manager, Rich Gerber, for his continuous help with our various questions and needs.

We hope that you enjoy the conference!

Yuji Matsumoto, Nara Institute of Science and Technology, Japan
Rashmi Prasad, University of Wisconsin-Milwaukee, U.S.A.

Preface: Local Chairs

Welcome to the COLING 2016!

It is a pleasure to welcome you to COLING 2016 organized by the Japanese Association of Natural Language Processing (ANLP) in Osaka. It has been 22 years since Japan last held the conference. While we are meeting here to discuss NLP, there is no substitute for personal contact. Therefore, we have arranged breaks, a reception, an excursion and a delightful banquet to facilitate discussion, collaboration and making connections. We hope and the modern conference venue together with the ambience of western Japan including Osaka, Nara and Kyoto (famous for their nature, culture, history, and food), help to make this an enjoyable experience for all. We hope the conference will result in accelerated growth of NLP.

Organizing a conference always takes a lot of work, and fortunately, we have experienced people from all around the world in attendance at the COLING 2016 site. It is both an honor and a great pleasure to work with them, and we thank them gratefully.

Since the proposal to host COLING was accepted by ICCL in 2014, our world has experienced some drastic changes. Under unfavorable economic conditions in Japan and considering the distance from Europe and America, we had to make a very conservative financial plan for the conference. The sponsorship chairs worked very hard and collected 33 sponsors, which is considerably more than in previous COLINGs.

This year's conference has attracted a huge number of submissions and has a high level of participation, reflecting the ongoing dynamism in artificial intelligence around the globe. We were both overwhelmed by the numbers of visa applications we had to handle, and at the same time delighted and excited by the tremendous response.

We'd like to end by reporting two special features of COLING 2016: (1) COLING will assist student participants with registration subsidies. Successful applicants for the Student Support Program will receive all-inclusive free registration; (2) the collocation of the first international symposium for young researchers working on Natural Language Processing (YRSNLP) as an official satellite event at COLING 2016.

Welcome, and enjoy the conference!

Eiichiro SUMITA, Takenobu TOKUNAGA, and Sadao KUROHASHI

COLING 2016 Local Chairs

Organisers

General Chair
Nicoletta Calzolari

Programme Chairs
Yuji Matsumoto
Rashmi Prasad

Local Chairs
Eiichiro Sumita
Takenobu Tokunaga
Sadao Kurohashi

Workshops Chairs
Key-Sun Choi
Monica Monachini
Yusuke Miyao

Demo Chair
Hideo Watanabe

Tutorial Chairs
Marcello Federico
Akiko Aizawa

Publication Chairs
Hitoshi Isahara
Masao Utiyama

Sponsorship Chairs
Satoshi Sekine
Su Jian
Patrick Pantel
Chengqing Zong
Ralf Steinberger

Publicity Chairs
Srinivas Bangalore
Dekai Wu
Antonio Branco

Advisor to the Local Committee
John Judge

Web Master
Akifumi Yoshimoto

Student Volunteer Coordinator
Yugo Murawaki

Program Booklet Coordinator
Akihiro Tamura

Programme Committee

Programme Chairs

Yuji Matsumoto

Rashmi Prasad

Linguistic Issues in NLP

Tracy Holloway King

Annie Zaenen

Machine Learning for NLP

Trevor Cohn

Sujith Ravi

Computational Psycholinguistics

Vera Demberg

Shravan Vasishth

Morphology, Segmentation, Tagging, Chunking

Daichi Mochihashi

Hinrich Schutze

Syntactic and Semantic Parsing, Grammar Induction

Daisuke Kawahara

Joakim Nivre

Lexical Semantics, Ontologies

Diana Inkpen

Roberto Navigli

Semantic Processing, Distributional Semantics, Compositionality

Chris Biemann

Roser Morante

Stephen Wu

Discourse Relations, Coreference, Pragmatics

Vincent Ng

Bonnie Webber

Natural Language Generation, Summarization

Donia Scott

Hiroya Takamura

Michael White

Paraphrasing, Textual Entailment

Roy Bar-Haim

Kentaro Inui

Sentiment Analysis, Computational Argumentation

Iryna Gurevych

Swapna Somasundaran

Information Retrieval, Information Extraction, Question Answering

Mausam

Marie-Francine Moens

Applications

Tim Baldwin

Maria Liakata

Dialog Processing and Dialog Systems, Multimodal Interfaces

Nina Dethlefs

Simon Keizer

Giuseppe Riccardi

Speech Recognition, Text-To-Speech, Spoken Language Understanding

Florian Metze

Chung-Hsien Wu

Machine Translation

Graham Neubig

Taro Watanabe

Min Zhang

Resources, Software and Tools

Christian Chiarcos

Nancy Ide

James Pustejovsky

Under-resourced Languages

Alexis Palmer

Richard Sproat

Reviewers

A Ramanathan, Aasish Pappu, Abhijit Mishra, Adam Przepiórkowski, Adam Grycner, Adam Meyers, Adam Tsakalidis, Adeline Nazarenko, Adrià de Gispert, Aida Nematzadeh,

AiTi Aw, Akihiro Tamura, Alan Akbik, Alan Ritter, Albert Gatt, Aldo Gangemi, Ales Horak, Alessandro Lenci, Alessandro Mazzei, Alessandro Moschitti,

Alessio Palmero Aprosio, Alex Rudnick, Alexander Clark, Alexander Panchenko, Alexandros Potamianos, Alexis Nasr, Alexis Palmer, Aleš Tamchyna, Alisa Zhila, Amal Zouaq,

Amanda Stent, Amba Kulkarni, Amin Mantrach, Amit Goyal, Amita Misra, Amjad Abu-Jbara, Amy Isard, Amália Mendes, Anastasia Krithara, Anders Björkelund,

Anders Søgaard, Andrea Horbach, Andreas Peldszus, Andreas Scherbakov, Andreas Søeborg Kirkedal, Andreas Vlachos, Andrei Popescu-Belis, Andrew Finch, Anette Frank, Animesh Mukherjee,

Anita Ramm, Anja Belz, Ann Bies, Ann Clifton, Anna Rumshisky, Anne Cocos, Anne-Lyse Minard, Annemarie Friedrich, Anoop Kunchukuttan, Antonio Pareja Lora,

Antonio Toral, Antske Fokkens, Antti Arppe, António Branco, Aoife Cahill, Arantza Diaz de Ilarraza, Ari Rappoport, Arkaitz Zubiaga, Arul Menezes, Arulmozi Selvaraj,

Ashish Vaswani, Ashutosh Modi, Atefeh Farzindar, Atsushi Fujii, Atsushi Fujita, Aurelie Neveol, Austin Matthews, Awais Athar, Axel-Cyrille Ngonga Ngomo,

Barbara Di Eugenio, Barbara Plank, Barry Haddow, Baskaran Sankaran, Beatrice Daille, Benjamin Marie, Benjamin Roth, Benoit Crabbé, Berlin Chen, Bernardo Magnini,

Bernd Bohnet, Bhavana Dalvi, Biao Zhang, Bilal Khaliq, Bill Dolan, Bin Chen, Bin Zou, Bing Zhao, Bishan Yang, Björn Gambäck,

Bob Coecke, Bolette Pedersen, Bonan Min, Bonaventura Coppola, Boxing Chen, Brian Murphy, Brian Riordan, Bryan Perozzi, Bryan Rink, Burcu Can,

Canasai Kruengkrai, Carlo Strapparava, Carlos Alzate, Carlos Gómez-Rodríguez, Carlos Ramisch, Carmen Heger, Carolina Gallardo, Carolina Scarton, Caroline Brun, Caroline Hagege,

Casey Kennington, Chandra Bhagavatula, Chen Chen, Chen Li, Chenchen Ding, Chengqing Zong, Chenhui Chu, Chi-kiu Lo, Chia-Ping Chen, Chloé Braud,

Chris Brockett, Chris Quirk, Christian Hardmeier, Christian Scheible, Christian Stab, Christiane Fellbaum, Christina Sauper, Christina Unger, Christophe Servan, Christopher Potts,

Christos Christodoulopoulos, Chu-Ren Huang, Chung-Hsien Wu, Chunyang Xiao, Cicero dos Santos, Ciprian Chelba, Claire Bonial, Claire Cardie, Claire Gardent, Clare Llewellyn,

Claude Barras, Claudia Borg, Colin Cherry, Cong Duy Vu Hoang, Congle Zhang, Constantin Orasan, Corentin Ribeyre, Cornelia Caragea, Costanza Navarretta, Courtney Napoles, Cristina Barros, Cristina Bosco, Cyril Goutte,

Daichi Mochihashi, Damiano Spina, Dan Flickinger, Dan Garrette, Dan Goldwasser, Dana Movshovitz-Attias, Dani Yogatama, Daniel Beck, Daniel Duma, Daniel Khashabi,

Daniel Marcu, Daniel Preoţiuc-Pietro, Danilo Croce, Danish Contractor, Danqi Chen, Danushka Bollegala, Dave Kleinschmidt, David Chiang, David Hall, David Jurgens,

David Kauchak, David McClosky, David R. Mortensen, David Schlangen, Delia Rusu, Deng Cai, Desislava Zhekova, Desmond Elliott, Deyi Xiong, Deyu ZHOU,

Diana Inkpen, Diana McCarthy, Diarmuid Ó Séaghdha, Didier Schwab, Diego Marcheggiani, Dimitri Kartsaklis, Dimitris Galanis, Dingcheng Li, Dipti Sharma, Do Kook Choe,

Doina Caragea, Dongdong Zhang, Donghong Ji, Donia Scott, Dorothee Beermann, Doug Downey, Doug Jones, Douglas Roland,

Eduardo Blanco, Edwin Simpson, Ehud Reiter, Ekaterina Vylomova, Elena Cabrio, Elena Lloret, Elias Iosif, Ella Rabinovich, Ellen Riloff, Elnaz Davoodi,

Els Lefever, emajd haddes, Emiel Krahmer, Emily M. Bender, Emmanuel Morin, Eneko Agirre, Eric De La Clergerie, Estevam Hruschka, Ethan Selfridge, Eugenio Martínez-Cámara,

Eva D'hondt, Eva Hajicova, Eva Hasler, Evelyne Viegas, Evgeny Stepanov, Eyal Shnarch,

Fabien Cromieres, Fabienne Cap, Fabio Massimo Zanzotto, Fabrice Lefevre, Fangtao Li, Farah Benamara, Fatiha Sadat, Feifei Zhai, Feiyu Xu, Felice Dell'Orletta,

Felix Engelmann, Fethi Bougares, Florian Boudin, Florian Metze, Francesca Bonin, Francesco Barbieri, Francis Bond, Francisco Casacuberta, Franco M. Luque, François Lareau, François Portet, François Yvon, Frederic Bechet, Frédéric Blain,

Gabriel Skantze, Gabriel Stanovsky, Gareth Jones, Geli Fei, Genichiro Kikui, George Foster, George Gkotsis, Georgeta Bordea, Geraldine Damnati, Gerasimos Lampouras,

German Rigau, Gerold Hintz, Gholamreza Haffari, Gianluca Lebani, Girish Jha, Giulia Venturi, Giuseppe Di Fabbrizio, Giuseppe Riccardi, Graeme Hirst, Graham Neubig,

Grazyna Demenko, Greg Durrett, Grzegorz Chrupała, Guenter Neumann, Guillaume Jacquet, Guillaume Wisniewski, Guillermo Garrido, Guo-Wei Bian, Guoguo Chen, Guohong Fu, Guy De Pauw, Guy Lapalme,

Hadi Amiri, Hai Zhao, Haitao Mi, Haithem Afli, Hajime Morita, Haris Papageorgiou, Helen Aristar-Dry, Helen Meng, Helmut Horacek, Helmut Schmid,

Hen-Hsen Huang, Heriberto Cuayahuitl, Hideto Kazawa, Hieu Hoang, Hinrich Schütze, Hiram Calvo, Hiroshi Kanayama, Hiroshi Noji, Hiroya Takamura, Hiroyuki Shindo,

Hitoshi Nishikawa, Hoang Cuong, Holger Schwenk, Hongwei Ding, Horacio Saggion, Hsin-Hsi Chen, Hsin-Min Wang, Hua Wu, Hui Lin, Hwee Tou Ng, Héctor Martínez Alonso,

Ielka van der Sluis, Iftekhar Naim, Ignacio Iacobacci, Ina Roesiger, Inderjeet Mani, Ines Rehbein, Ioannis Konstas, Irene Russo, Iria del Río Gayo, Irina Galinskaya,

Irina Temnikova, Iris Hendrickx, Isabel Moreno, Isabelle Augenstein, Ivan Habernal, Ivan Sanchez, Ivan Vulić,

Jan Hajic, Jan Niehues, Jan Šnajder, Jason D Williams, Jason Naradowsky, Jason Riesa, Jay Pujara, Jayant Krishnamurthy, Jeff Good, Jennifer Chu-Carroll,

Jennifer Williams, Jervis Pinto, Jesus Cardeńosa, Jesús González-Rubio, Jey Han Lau, Jiajun Zhang, Jill Burstein, Jim Blevins, Jim Breen, Jin Wang,

Jin-Dong Kim, Jing Jiang, Jing-Shin Chang, Jingbo Zhu, Jinho D. Choi, Joachim Bingel, Jodi Schneider, Joe Cheri, Joel Dunham, Joern Wuebker,

Johannes Bjerva, Johannes Daxenberger, John Carroll, John Chen, John Nerbonne, John Philip McCrae, John Richardson, John Wieting, Jon Dehdari, Jonathan Brennan,

Joonsuk Park, Jordi Atserias Batalla, Jorge Gracia, Jose Angel Olivas, Jose Camacho-Collados, Jose G. Moreno, Josef Ruppenhofer, Joseph Le Roux, José G. C. de Souza, Juan Carlos Gomez,

Judith Degen, Judith Eckle-Kohler, Juergen Trouvain, Julian Brooke, Julie Weeds, Julien Ah-Pine, Julien Kloetzer, Julien PEREZ, Julien Velcin, Jun'ichi Tsujii, Junhui Li, Junyi Jessy Li, Justine Kao, Jyoti Pawar, Jörg Tiedemann,

Kai Yu, Kalina Bontcheva, Kallirroi Georgila, Kam-Fai Wong, Kapil Thadani, Karel Pala, Karl Pichotta, Karl Stratos, Katharina Kann, Katsuhiko Hayashi,

Katsuhito Sudoh, Kay Berkling, Kay Mueller, Keelan Evanini, Keh-Yih Su, Keikichi Hirose, Keisuke Sakaguchi, Keith Hall, Keith Vander Linden, Kellie Webster,

Kemal Oflazer, Kenji Sagae, Kenneth Heafield, Kentaro Torisawa, Kevin Cohen, Kevin Duh, Kevin Gimpel, Kevin Knight, Kevin Scannell, Kirk Roberts,

Kishore Prahallad, Klaus Macherey, Koji Mineshima, Kostadin Cholakov, Krasimir Angelov, Krister Lindén, Kristina Striegnitz, Kuan-Yu Chen, Kuang-hua Chen, Kumiko Tanaka-Ishii, Kuntal Dey, Kyle Gorman,

L. Alfonso Urena Lopez, Lan Du, Laure Soulier, Laurent Besacier, Laurette Pretorius, Lea Frermann, Lei Li, Lemao Liu, Leo Wanner, Leon Derczynski,

Leonardo Badino, Leonardo Campillos Llanos, Liane Guillou, Liang-Chih Yu, Libby Barak, Likun Qiu, Lili Mou, Liling Tan, Lilja Øvrelid, Lina M. Rojas Barahona,

Lingpeng Kong, Lisa Beinborn, Lluís Màrquez, Long Duong, Lonneke van der Plas, Lori Levin, Louise McNally, Loïc Barrault, Lu Qin, Lu Wang,

Lucie Flekova, Luis Espinosa Anke, Lun-Wei Ku, Lung-Hao Lee, Luís Morgado da Costa, Lyle Ungar,

Maciej Ogrodniczuk, Maciej Piasecki, Maja Popović, Majid Laali, Makoto Kanazawa, Malvina Nissim, Mamoru Komachi, Manabu Okumura, Manabu Sassano,

Mandar Mitra, Manfred Stede, Mans Hulden, Marc Dymetman, Marc Elsässer, Marco Silvio Giuseppe Senaldi, Marco Turchi, Marcus Rohrbach, Marelie Davel, Maria Eskevich,

Maria J. Martin-Bautista, Maria Liakata, Marianna Apidianaki, Marie Candito, Marie-Francine Moens, Marie-Jean Meurs, Marina Sokolova, Marine Carpuat, Marion Weller-Di Marco, Mark Dras,

Mark Hasegawa-Johnson, Mark Sammons, Mark Steedman, Mark Stevenson, Mark-Jan Nederhof, Markus Dreyer, Markus Egg, Markus Saers, Marta Tatu, Marta Vicente,

Marten van Schijndel, Martin Benjamin, Martin Forst, Martin Reynaert, Martin Riedl, Marzieh Saeidi, Masaaki Nagata, Masato Hagiwara, Masayuki Asahara, Masoud Rouhizadeh,

Massimo Poesio, Mathieu Roche, Matje van de Camp, Matt Gardner, Matteo Negri, Matthew Purver, Matthew Stone, Matthias Gallé, Matthias Hartung, Matthieu Constant,

Maud Ehrmann, Mauro Cettolo, Mausam, Maxim Ionov, Maya Ramanath, Mehdi Samadi, Meishan Zhang, Melvin Johnson Premkumar, Menno van Zaanen, Miao Fan,

Michael Bloodgood, Michael Denkowski, Michael Elhadad, Michael Flor, Michael Maxwell, Michael Picheny, Michael Wiegand, Michael Zock, Michaela Regneri, Michal Konkol,

Miguel Ballesteros, Mike Kestemont, Mike Thelwall, Mikel Forcada, Mikhail Kozhevnikov, Milan Dojchinovski, Milica Gasic, Miloslav Konopik, Ming Huang, Mingbo Ma,

Minlie Huang, Mitesh M. Khapra, Mitja Trampus, Mo Shen, Mo Yu, Mohamed Yahya, Mohammad Salameh, Mohammad Taher Pilehvar, Mona Diab, Monica Monachini,

Monojit Choudhury, Moreno Coco, Mu Qiao, Muhammad Abdul-Mageed, Muhammad Humayoun, Muhua Zhu, Mária Gósy,

Nadir Durrani, Nancy Chen, Nancy Green, Naoaki Okazaki, Naoki Yoshinaga, Naoya Inoue, Natalia Grabar, Natalie Schluter, Ndapandula Nakashole, Ni Lao,

Nianwen Xue, Nick Campbell, Nick Thieberger, Nick Webb, Nicoletta Calzolari, Nigel Collier, Niket Tandon, Niko Schenk, Nikolaos Aletras, Nikolaos Pappas,

Niladri Chatterjee, Nils Reiter, Nina Dethlefs, Niranjan Balasubramanian, Nobuhiro Kaji, Nut Limsopatham,

Oier Lopez de Lacalle, Oliver Hellwig, Oliver Niebuhr, Olivier Ferret, Olivier Pietquin, Ondřej Bojar, Or Biran, Oscar Täckström,

Paavo Alku, Paolo Rosso, Paramita Mirza, Parinaz Sobhani, Parisa Kordjamshidi, Patrick Ehlen, Patrick Paroubek, Paul Buitelaar, Paul Crook, Pavel Pecina,

Pavel Straňák, Pawan Goyal, Peter Ljunglöf, Petra Barancikova, Petya Osenova, Philipp Cimiano, Philipp Koehn, Philippe Muller, Phillippe Langlais, Phong Le,

Pidong Wang, Piek Vossen, Pierre Lison, Pierre Zweigenbaum, Ping Jian, Piroska Lendvai, Pontus Stenetorp, Pradipto Das, Prasad Tadepalli, Prasanth Kolachina, Preethi Raghavan, Preslav Nakov, Prokopis Prokopidis, Pushpak Bhattacharyya,

Qi Li, Qun Liu,

Rabih Zbib, Rada Mihalcea, Radu Florian, Rafael E. Banchs, Rahul Gupta, Rajen Chatterjee, Ralph Grishman, Rami Al-Rfou, Raquel Amaro, Raquel Fernandez,

Raveesh Meena, Reut Tsarfaty, Richard Eckart de Castilho, Richard Johansson, Richard Sproat, Richard Tzong-Han Tsai, Richárd Farkas, Rico Sennrich, Rob Voigt, Robert Östling,

Roberto Basili, Robin Cooper, Roger Evans, Roland Kuhn, Roman Klinger, Ron Kaplan, Roy Bar-Haim, Rudra Murthy, Rui Xia, Ruihong Huang, Ryan Cotterell, Ryohei Sasano,

Ryu Iida, Ryuichiro Higashinaka,

Sa-kwang Song, Sabine Bergler, Sabino Miranda-Jiménez, Sadao Kurohashi, Saif Mohammad, Salah Ait-Mokhtar, Sam Wiseman, Sameer Singh, Sampo Pyysalo, Sanda Harabagiu,

Sanghoun Song, Sanjeev Kumar Karn, Sara Rosenthal, Sara Tonelli, Sascha Griffiths, Sascha Rothe, Satoshi Sekine, Saurav Sahay, Sean Chester, Sebastian Ebert,

Sebastian Hellmann, Sebastian Krause, Sebastian Martschat, Sebastian Nordhoff, Sebastian Padó, Sebastian Riedel, Seong-Bae Park, Shachar Mirkin, Sharid Loáiciga, Shashi Narayan,

Shibamouli Lahiri, Shih-Hung Wu, Shinsuke Mori, Shravan Vasishth, Shuly Wintner, Siddharth Patwardhan, Simon Keizer, Simon Mille, Simone Paolo Ponzetto, Simonetta Montemagni,

Sina Zarrieß, Sinno Jialin Pan, Smaranda Muresan, Sophia Ananiadou, Stefan Bott, Stefan L. Frank, Stefan Ultes, Stefano Faralli, Steffen Eger, Stephan Busemann,

Stephan Oepen, Stephen Wan, Stergos Afantenos, Steve DeNeefe, Steven Bethard, Steven Moran, Steven Wilson, Su-Youn Yoon, Sujay Kumar Jauhar, Sumithra Velupillai,

Sunghwan Mac Kim, Suresh Manandhar, Susanne Burger, Suzan Verberne, Svetla Koeva, Svetlana Stoyanchev, Svitlana Vakulenko, Svitlana Volkova,

Tadashi Nomoto, Takenobu Tokunaga, Takuya Matsuzaki, Tal Linzen, Tamás Váradi, Tanel Alumäe, Tara McIntosh, Tatsuya Kawahara, Tejaswini Deoskar, Tengfei Ma,

Tetsuji Nakagawa, Thiago Castro Ferreira, Thien Huu Nguyen, Thierry Hamon, Thomas Meyer, Tim Rocktäschel, Tim Van de Cruys, Timo Baumann, Ting Han, Ting Liu,

Titus von der Malsburg, Tommaso Caselli, tommaso pasini, Tomohide Shibata, Tomoya Iwakura, Tomáš Brychcín, Tomáš Hercig, Tong Wang, Torsten Zesch, Tracy Holloway King, Travis Goodwin, Tristan Miller, Tsutomu Hirao, Tsvetana Dimitrova,

Uladzimir Sidarenka, Ulrike Pado,

Valentina Presutti, Valia Kordoni, Vanni Zavarella, Vasile Rus, Vasileios Lampos, Vassilina Nikoulina, Verena Rieser, Verónica Pérez-Rosas, Veselin Stoyanov, Victoria Rosén,

Victoria Yaneva, Viet-An Nguyen, Vincent Ng, Virach Sornlertlamvanich, Vivek Kulkarni, Vivek Kumar Rangarajan Sridhar, Vivek Srikumar, Vladimir Eidelman, Véronique MORICEAU,

Walter Daelemans, Wanxiang Che, Wei Xu, Wei Xu, Wei Zhang, Wen-Lian Hsu, Wenliang Chen, Wenpeng Yin, Wilker Aziz, William Bryce, William Lewis, William Schuler, William Yang Wang, Wolfgang Macherey, Wolfgang Maier,

Xavier Tannier, Xiangyu Duan, Xiao Ling, Xiaochang Peng, Xiaojun Wan, Xinchi Chen, Xingyi Song, Xu Sun, Xuancong Wang, Xuanjing Huang,

Yadollah Yaghoobzadeh, Yael Netzer, Yajie Miao, Yang Li, Yang Liu, Yannick Versley, Yannis Korkontzelos, Yanshan Wang, Yi-Ting Huang, Yingtao Tian,

Yoan Gutiérrez, Yoav Goldberg, Yogarshi Vyas, Yonatan Bisk, Yoong Keok Lee, Yuan Cao, Yufang Hou, Yuichiroh Matsubayashi, Yulan He, Yulia Tsvetkov, Yun-Nung Chen, Yuta Tsuboi, Yvette Graham,

Zdenka Uresova, Zhenghua Li, Zhiguo Wang, Zhiyuan Chen, Zhongqiang Huang, Zhongyu Wei, Zico Pratama Putra, Ákos Kádár, Özlem Çetinoğlu, Željko Agić

Secondary Reviewers

Abdalghani Abujabal, Rodrigo Agerri, Milan Aggarwal, Domagoj Alagic, Alessio Aprosio Palmero, Nikolay Arefiev, Kartik Asooja, Hanna Bechara, Irshad Bhat, Filip Boltuzic,

Alexey Borisov, Julia Bosque-Gil, Lars Bungum, Alberto Castro-Hernandez, Andrew Cattle, Yong Cheng, Maximu Coavoux, Łukasz Dębowski, Milan Dojchinovski, Shichao Dong,

Erick Fonseca, Stella Frank, Miguel Ángel García Cumbreras, Matt Gardner, Simon-Pierre Genot, Maite Giménez, Filip Ilievski, Sharmistha Jat, Salud María Jiménez-Zafra, Charles Jochim,

Jenna Kanerva, Mladen Karan, Yasen Kiprov, Bettina Klimek, Julien Kloetzer, Arne Köhn, Alexandros Komninos, Canasai Kruengkrai, Wu Kui, Gorka Labaka,

Po-Ting Lai, Egoitz Laparra, Joseph Lark, Jing Li, Minglei Li, Zhuo Li, Ming Liao, Chingsheng Lin, Yunfei Long, Jing Ma,

Shuming Ma, Simone Magnolini, Márton Makrai, Shervin Malmasi, Fernando Martinez Santiago, John McCrae, Zoran Medic, Todor Mihaylov, Tsvetomila Mihaylova, Benjamin Milde,

Pruthwik Mishra, Véronique Moriceau, Alessandro Moschitti, Diego Moussallem, Kay Müller, Jason Narad, Jumana Nassour, Ciro Baron Neto, Minh Le Ngoc, Jong-Hoon Oh,

Tomoko Ohta, Yuya J. Ong, Sebastián Peña Saldarriaga, Haoruo Peng, Reshmi Gopalakrishna Pillai, Denys Proux, Gustavo Publio, Xinyin Qiu, Roi Reichart, Alexey Romanov,

Salvatore Romeo, Sophie Rosset, Dominic Rout, Andreas Rücklé, Nick Ruiz, Hassan Sajjad, Himanshu Sharma, Shiqi Shen, George Spithourakis, Pei-Hao Su,

Nina Tahmasebi, Partha Talukdar, Liling Tan, Serra Sinem Tekiroglu, Sara Tonelli, Adam Tsakalidis, Kateryna Tymoshenko, Dmitry Ustalov, Antonio Uva, Emiel van Miltenburg,

Chantal van Son, David Vilar Torres, Tim vor der Brück, Bo Wang, Pidong Wang, Shuohang Wang, Yu-Chun Wang, Johannes Welbl, Tsung-Hsien Wen, Michael White,

Chun-Kai Wu, Shuangzhi Wu, Jiacheng Xu, Wang Xuancong, Jian Zhang, Meishan Zhang, Meng Zhang, Nina Zhou, Patrick Ziering

Invited talk 1

Universal Dependencies – Dubious Linguistics and Crappy Parsing?

Joakim Nivre (Uppsala University)

Universal Dependencies is a framework for cross-linguistically consistent treebank annotation that has so far been applied to over 50 languages. It was developed primarily to support multilingual parsing research, but the resources have proven useful for a wide range of studies that were not foreseen originally, including research on language typology. A basic design principle in Universal Dependencies is to give priority to grammatical relations between content words, which are more likely to be parallel across languages, and to treat function words essentially as features of content words. This principle has been criticized both for being incompatible with theoretical linguistics, which tend to treat function words as syntactic heads, and for being suboptimal as a representation for dependency parsing, where higher accuracy is often observed with function words as heads. I will argue that both of these criticisms rest on a misinterpretation of the syntactic representations, and I will show that an alternative interpretation is compatible with both sound linguistics and improved parsing technology.

Invited talk 2

Getting the Input Right: Refining our Understanding of What Children Hear

Reiko Mazuka (RIKEN Brain Science Institute & Duke University)

As models for language learning become increasingly sophisticated, it is essential to pay close attention to the purported input received by learners. This talk presents two examples of phonological input for which a failure to account for relevant factors has led to misleading conclusions. A fully annotated dataset of infant-directed speech is now allowing a more refined analysis of what children actually hear. The first example concerns vowel-duration contrasts (long vs. short) in Japanese. One previous study, working under the assumption that long and short vowels occurred with equal frequency, concluded that the distinction could be learned by a simple distributional model. Our dataset, however, reveals that (a) in reality over 90% of vowels in Japanese are short, and (b) the distribution of long vowel duration is entirely encompassed within that of short vowels. The second example concerns the widely accepted claim that when adults speak to infants (infant-directed speech, IDS), they speak with a slower speech rate than when speaking to adults (adult-directed speech, ADS). Studies supporting this conclusion, however, have consistently failed to account for the fact that IDS utterances are shorter than those of ADS. Our dataset differentiates between utterance-internal speech rate and utterance-final lengthening, and finds taken separately, these values almost identical between IDS and ADS. As it turns out, IDS appeared to have a slower overall rate only because of the greater frequency of utterance-final segments.

Invited talk 3

NLP to support clinical tasks and decisions

Dina Demner-Fushman (U.S. National Library of Medicine)

Clinical decision support (CDS) provides clinicians and patients with information needed to enhance health and health care. Clinical NLP – natural language processing methods to support healthcare by operationalizing clinical information contained in clinical narrative – is an integral part of CDS. Clinical NLP has started in the early 1960s, with several successful applications now integrated in daily care. I will first discuss the successful applications that are already positively impacting clinical practice, as well as publicly available resources, including those developed by our group. Consumer language understanding is an equally important and rapidly evolving part of CDS. In the second part of the talk, I will present our work in understanding consumer health questions. I will conclude with the challenges and opportunities to contribute to these fascinating research areas that have practical implications for our health.

Invited talk 4

A Look at Computational Argumentation and Summarisation from a Text-Understanding Perspective

Simone Teufel (University of Cambridge)

In the past five years, computational argumentation has emerged as a new, active research field. This field studies all aspects of analysing and generating human argumentation, including argument mining, supportive debating technologies, logical representation of arguments, models of reasoning, and the connection of discourse processing and argumentation. As somebody who is mainly interested in the text-understanding challenges of computational argumentation, I think this new field has the potential to advance (and provide means of evaluating) the text-understanding capabilities of today's NLP systems.

When humans construct an argument in order to convince others, how do they order and structure the information they want to convey? I will argue that whatever principles are at work, they are almost identical to those needed when summarising a text. Amongst the relations of particular interest are entailment, causal and rhetorical relationships. I will give an overview of currently available (text understanding-based) analysis methods that can inform our understanding of these principles, and I will also say a few words about a proposition-based approach to summarisation we have developed at Cambridge University that has the potential to contribute insights to computational argumentation.

Table of Contents

Conference Program

Tuesday, December 13, 2016

09:00–09:30 Opening:

09:30–10:30 Invited talk 1: Joakim Nivre (Uppsala University)

11:00–12:30 Session 1-A: Syntactic and Semantic Parsing, Grammar Induction I

Boosting for Efficient Model Selection for Syntactic Parsing
Rachel Bawden and Benoît Crabbé

A Universal Framework for Inductive Transfer Parsing across Multi-typed Treebanks
Jiang Guo, Wanxiang Che, Haifeng Wang and Ting Liu

Grammar induction from (lots of) words alone
John K Pate and Mark Johnson

11:00–12:30 Session 1-B: Natural Language Generation, Summarization I

A Redundancy-Aware Sentence Regression Framework for Extractive Summarization
Pengjie Ren, Furu Wei, Zhumin CHEN, Jun MA and Ming Zhou

Generating Video Description using Sequence-to-sequence Model with Temporal Attention
Natsuda Laokulrat, Sang Phan, Noriki Nishida, Raphael Shu, Yo Ehara, Naoaki Okazaki, Yusuke Miyao and Hideki Nakayama

An Improved Phrase-based Approach to Annotating and Summarizing Student Course Responses
Wencan Luo, Fei Liu and Diane Litman

Tuesday, December 13, 2016 (continued)

11:00–12:30 Session 1-C: Applications I

CATENA: CAusal and TEmporal relation extraction from NAtural language texts
Paramita Mirza and Sara Tonelli

Forecasting Word Model: Twitter-based Influenza Surveillance and Prediction
Hayate ISO, Shoko WAKAMIYA and Eiji ARAMAKI

Task-Oriented Intrinsic Evaluation of Semantic Textual Similarity
Nils Reimers, Philip Beyer and Iryna Gurevych

11:00–12:30 Session 1-D: Resources, Software, Tools and Under-resourced languages I

Expanding wordnets to new languages with multilingual sense disambiguation
Mihael Arcan, John Philip McCrae and Paul Buitelaar

A Correlational Encoder Decoder Architecture for Pivot Based Sequence Generation
Amrita Saha, Mitesh M. Khapra, Sarath Chandar, Janarthanan Rajendran and Kyunghyun Cho

Zero-resource Dependency Parsing: Boosting Delexicalized Cross-lingual Transfer with Linguistic Knowledge
Lauriane Aufrant, Guillaume Wisniewski and François Yvon

12:30–14:00 ***Lunch break***

Tuesday, December 13, 2016 (continued)

14:00–16:00 Session 2-A: Machine Learning for NLP I

Improving historical spelling normalization with bi-directional LSTMs and multi-task learning
Marcel Bollmann and Anders Søgaard

Deceptive Opinion Spam Detection Using Neural Network
Yafeng Ren and Yue Zhang

Integrating Topic Modeling with Word Embeddings by Mixtures of vMFs
Ximing Li, Jinjin Chi, Changchun Li, Jihong Ouyang and Bo Fu

Bayesian Language Model based on Mixture of Segmental Contexts for Spontaneous Utterances with Unexpected Words
Ryu Takeda and Kazunori Komatani

14:00–16:00 Session 2-B: Morphology, Segmentation Tagging, Chunking I

Label Embedding for Zero-shot Fine-grained Named Entity Typing
Yukun Ma, Erik Cambria and SA GAO

The Role of Context in Neural Morphological Disambiguation
Qinlan Shen, Daniel Clothiaux, Emily Tagtow, Patrick Littell and Chris Dyer

Asynchronous Parallel Learning for Neural Networks and Structured Models with Dense Features
Xu Sun

An Empirical Exploration of Skip Connections for Sequential Tagging
Huijia Wu, Jiajun Zhang and Chengqing Zong

14:00–16:00 Session 2-C: Natural Language Generation, Summarization II

Exploring Text Links for Coherent Multi-Document Summarization
Xun Wang, Masaaki Nishino, Tsutomu Hirao, Katsuhito Sudoh and Masaaki Nagata

Syntactic realization with data-driven neural tree grammars
Brian McMahan and Matthew Stone

Abstractive News Summarization based on Event Semantic Link Network
Wei Li, Lei He and Hai Zhuge

A General Optimization Framework for Multi-Document Summarization Using Genetic Algorithms and Swarm Intelligence
Maxime Peyrard and Judith Eckle-Kohler

14:00–15:30 Session 2-D: Speech Recognition, Text-To-Speech, Spoken Language Understanding

Exploiting Sentence and Context Representations in Deep Neural Models for Spoken Language Understanding
Lina M. Rojas Barahona, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, Stefan Ultes, Tsung-Hsien Wen and Steve Young

Predictive Incremental Parsing Helps Language Modeling
Arne Köhn and Timo Baumann

A Neural Attention Model for Disfluency Detection
Shaolei Wang, Wanxiang Che and Ting Liu

Tuesday, December 13, 2016 (continued)

14:00–16:00 Session 2-P: Poster Session 1

Morphology, Segmentation, Tagging, Chunking

Detecting Sentence Boundaries in Sanskrit Texts
Oliver Hellwig

Consistent Word Segmentation, Part-of-Speech Tagging and Dependency Labelling Annotation for Chinese Language
Mo Shen, Wingmui Li, HyunJeong Choe, Chenhui Chu, Daisuke Kawahara and Sadao Kurohashi

Attending to Characters in Neural Sequence Labeling Models
Marek Rei, Gamal Crichton and Sampo Pyysalo

A Word Labeling Approach to Thai Sentence Boundary Detection and POS Tagging
Nina Zhou, AiTi Aw, Nattadaporn Lertcheva and Xuancong Wang

Assigning Fine-grained PoS Tags based on High-precision Coarse-grained Tagging
Tobias Horsmann and Torsten Zesch

Data-Driven Morphological Analysis and Disambiguation for Morphologically Rich Languages and Universal Dependencies
Amir More and Reut Tsarfaty

Automatic Syllabification for Manipuri language
Loitongbam Gyanendro Singh, Lenin Laitonjam and Sanasam Ranbir Singh

Speech Recognition, Text-To-Speech, Spoken Language Understanding

Learning to Distill: The Essence Vector Modeling Framework
Kuan-Yu Chen, Shih-Hung Liu, Berlin Chen and Hsin-Min Wang

Continuous Expressive Speaking Styles Synthesis based on CVSM and MR-HMM
Jaime Lorenzo-Trueba, Roberto Barra-Chicote, Ascension Gallardo-Antolin, Junichi Yamagishi and Juan M Montero

An Automatic Prosody Tagger for Spontaneous Speech
Monica Dominguez, Mireia Farrús and Leo Wanner

Frustratingly Easy Neural Domain Adaptation
Young-Bum Kim, Karl Stratos and Ruhi Sarikaya

Syntactic and Semantic Parsing, Grammar Induction

A House United: Bridging the Script and Lexical Barrier between Hindi and Urdu
Riyaz A. Bhat, Irshad A. Bhat, Naman Jain and Dipti Misra Sharma

Deeper syntax for better semantic parsing
Olivier Michalon, Corentin Ribeyre, Marie Candito and Alexis Nasr

Language Independent Dependency to Constituent Tree Conversion
Young-Suk Lee and Zhiguo Wang

Promoting multiword expressions in A TAG parsing*
Jakub Waszczuk, Agata Savary and Yannick Parmentier

Under-resourced Languages

Incrementally Learning a Dependency Parser to Support Language Documentation in Field Linguistics
Morgan Ulinski, Julia Hirschberg and Owen Rambow

Inducing Multilingual Text Analysis Tools Using Bidirectional Recurrent Neural Networks
Othman ZENNAKI, Nasredine Semmar and Laurent Besacier

Bitext Name Tagging for Cross-lingual Entity Annotation Projection
Dongxu Zhang, Boliang Zhang, Xiaoman Pan, Xiaocheng Feng, Heng Ji and Weiran XU

Determining the Multiword Expression Inventory of a Surprise Language
Bahar Salehi, Paul Cook and Timothy Baldwin

A Hybrid Deep Learning Architecture for Sentiment Analysis
Md Shad Akhtar, Ayush Kumar, Asif Ekbal and Pushpak Bhattacharyya

Word Segmentation in Sanskrit Using Path Constrained Random Walks
Amrith Krishna, Bishal Santra, Pavankumar Satuluri, Sasi Prasanth Bandaru, Bhumi Faldu, Yajuvendra Singh and Pawan Goyal

Mongolian Named Entity Recognition System with Rich Features
Weihua Wang, Feilong Bao and Guanglai Gao

Applications

Appraising UMLS Coverage for Summarizing Medical Evidence
Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong and Fang Chen

Large-scale Multi-class and Hierarchical Product Categorization for an E-commerce Giant
Ali Cevahir and Koji Murakami

Product Classification in E-Commerce using Distributional Semantics
Vivek Gupta, Harish Karnick, Ashendra Bansal and Pradhuman Jhala

16:00–16:30 ***coffee break***

Tuesday, December 13, 2016 (continued)

16:30–18:00 Session 3-A: Natural Language Generation, Summarization III

AttSum: Joint Learning of Focusing and Summarization with Neural Attention
Ziqiang Cao, Wenjie Li, Sujian Li, Furu Wei and Yanran Li

Using Relevant Public Posts to Enhance News Article Summarization
Chen Li, Zhongyu Wei, Yang Liu, Yang Jin and Fei Huang

A Proposition-Based Abstractive Summariser
Yimai Fang, Haoyue Zhu, Ewa Muszyńska, Alexander Kuhnle and Simone Teufel

16:30–18:00 Session 3-B: Syntactic and Semantic Parsing, Grammar Induction II

Cross-lingual Learning of an Open-domain Semantic Parser
Kilian Evang and Johan Bos

A subtree-based factorization of dependency parsing
Qiuye Zhao and Qun Liu

K-SRL: Instance-based Learning for Semantic Role Labeling
Alan Akbik and Yunyao Li

16:30–18:00 Session 3-C: Morphology, Segmentation Tagging, Chunking II

Keystroke dynamics as signal for shallow syntactic parsing
Barbara Plank

A Bayesian model for joint word alignment and part-of-speech transfer
Robert Östling

Splitting compounds with ngrams
Naomi Tachikawa Shapiro

16:30–18:00 Session 3-D: Applications II

GAKE: Graph Aware Knowledge Embedding
Jun Feng, Minlie Huang, Yang Yang and xiaoyan zhu

Ranking Responses Oriented to Conversational Relevance in Chat-bots
Bowen Wu, Baoxun Wang and Hui Xue

Probabilistic Prototype Model for Serendipitous Property Mining
Taesung Lee, Seung-won Hwang and Zhongyuan Wang

16:30–18:00 Session 3-P: Poster Session 2

Computational Psycholinguistics

Identifying Cross-Cultural Differences in Word Usage
Aparna Garimella, Rada Mihalcea and James Pennebaker

Reading-Time Annotations for "Balanced Corpus of Contemporary Written Japanese"
Masayuki Asahara, Hajime Ono and Edson T. Miyamoto

"How Bullying is this Message?": A Psychometric Thermometer for Bullying
Parma Nand, Rivindu Perera and Abhijeet Kasture

Learning grammatical categories using paradigmatic representations: Substitute words for language acquisition
Mehmet Ali Yatbaz, Volkan Cirik, Aylin Küntay and Deniz Yuret

Understanding the Lexical Simplification Needs of Non-Native Speakers of English
Gustavo Paetzold and Lucia Specia

How Interlocutors Coordinate with each other within Emotional Segments?
Firoj Alam, Shammur Absar Chowdhury, Morena Danieli and Giuseppe Riccardi

Linguistic Issues in NLP

Advancing Linguistic Features and Insights by Label-informed Feature Grouping: An Exploration in the Context of Native Language Identification
Serhiy Bykh and Detmar Meurers

Modeling Diachronic Change in Scientific Writing with Information Density
Raphael Rubino, Stefania Degaetano-Ortlieb, Elke Teich and Josef van Genabith

Different Contexts Lead to Different Word Embeddings
Wenpeng Hu, Jiajun Zhang and Nan Zheng

Machine Learning for Metrical Analysis of English Poetry
Manex Agirrezabal, Iñaki Alegria and Mans Hulden

Automated speech-unit delimitation in spoken learner English
Russell Moore, Andrew Caines, Calbert Graham and Paula Buttery

Learning to Identify Sentence Parallelism in Student Essays
Wei Song, Tong Liu, Ruiji Fu, Lizhen Liu, Hanshi Wang and Ting Liu

Evaluating anaphora and coreference resolution to improve automatic keyphrase extraction
Marco Basaldella, Giorgia Chiaradia and Carlo Tasso

Retrieving Occurrences of Grammatical Constructions
Anna Ehrlemark, Richard Johansson and Benjamin Lyngfelt

Automatic Extraction of Learner Errors in ESL Sentences Using Linguistically Enhanced Alignments
Mariano Felice, Christopher Bryant and Ted Briscoe

Contrasting Vertical and Horizontal Transmission of Typological Features
Kenji Yamauchi and Yugo Murawaki

How Regular is Japanese Loanword Adaptation? A Computational Study
Lingshuang Mao and Mans Hulden

Using Linguistic Data for English and Spanish Verb-Noun Combination Identification
Uxoa Iñurrieta, Arantza Diaz de Ilarraza, Gorka Labaka, Kepa Sarasola, Itziar Aduriz and John Carroll

Tuesday, December 13, 2016 (continued)

Applications

Analyzing Gender Bias in Student Evaluations
Andamlak Terkik, Emily Prud'hommeaux, Cecilia Ovesdotter Alm, Christopher Homan and Scott Franklin

Adverse Drug Reaction Classification With Deep Neural Networks
Trung Huynh, Yulan He, Alistair Willis and Stefan Rueger

Chinese Preposition Selection for Grammatical Error Diagnosis
Hen-Hsen Huang, Yen-Chi Shao and Hsin-Hsi Chen

Wednesday, December 14, 2016

09:00–10:00 **Invited talk 2: Reiko Mazuka (RIKEN Brain Science Institute & Duke University)**

10:00–10:30 ***coffee break***

10:30–12:00 **Session 4-A: Morphology, Segmentation, Tagging, Chunking III**

Extending the Use of Adaptor Grammars for Unsupervised Morphological Segmentation of Unseen Languages
Ramy Eskander, Owen Rambow and Tianchun Yang

CharNER: Character-Level Named Entity Recognition
Onur Kuru, Ozan Arkan Can and Deniz Yuret

A Neural Model for Part-of-Speech Tagging in Historical Texts
Christian Hardmeier

10:30–12:00 Session 4-B: Applications III

Extracting Discriminative Keyphrases with Learned Semantic Hierarchies
Yunli Wang, Yong Jin, Xiaodan Zhu and Cyril Goutte

Hashtag Recommendation Using End-To-End Memory Networks with Hierarchical Attention
Haoran Huang, Qi Zhang, Yeyun Gong and Xuanjing Huang

Automatic Labelling of Topics with Neural Embeddings
Shraey Bhatia, Jey Han Lau and Timothy Baldwin

10:30–12:00 Session 4-C: Computational Psycholinguistics and Linguistic Issues in NLP I

Memory-Bounded Left-Corner Unsupervised Grammar Induction on Child-Directed Input
Cory Shain, William Bryce, Lifeng Jin, Victoria Krakovna, Finale Doshi-Velez, Timothy Miller, William Schuler and Lane Schwartz

'Calling on the classical phone': a distributional model of adjective-noun errors in learners' English
Aurélie Herbelot and Ekaterina Kochmar

Are Cohesive Features Relevant for Text Readability Evaluation?
Amalia Todirascu, Thomas Francois, Delphine Bernhard, Nuria Gala and Anne-Laure Ligozat

10:30–12:00 Session 4-D: Resources, Software, Tools and Under-resourced languages II

Named Entity Recognition for Linguistic Rapid Response in Low-Resource Languages: Sorani Kurdish and Tajik
Patrick Littell, Kartik Goyal, David R. Mortensen, Alexa Little, Chris Dyer and Lori Levin

Multilingual Supervision of Semantic Annotation
Peter Exner, Marcus Klang and Pierre Nugues

Siamese Convolutional Networks for Cognate Identification
Taraka Rama

10:30–12:00 Session 4-P: Poster Session 3

Natural Language Generation, Summarization

Exploring Differential Topic Models for Comparative Summarization of Scientific Papers
Lei He, Wei Li and Hai Zhuge

Bridging the gap between extractive and abstractive summaries: Creation and evaluation of coherent extracts from heterogeneous sources
Darina Benikova, Margot Mieskes, Christian M. Meyer and Iryna Gurevych

Chinese Poetry Generation with Planning based Neural Network
Zhe Wang, Wei He, Hua Wu, Haiyang Wu, Wei Li, Haifeng Wang and Enhong Chen

Predicting sentential semantic compatibility for aggregation in text-to-text generation
Victor Chenal and Jackie Chi Kit Cheung

Sequential Clustering and Contextual Importance Measures for Incremental Update Summarization
Markus Zopf, Eneldo Loza Mencía and Johannes Fürnkranz

Natural Language Generation through Character-based RNNs with Finite-state Prior Knowledge
Raghav Goyal, Marc Dymetman and Eric Gaussier

A Hybrid Approach to Generation of Missing Abstracts in Biomedical Literature
Suchet Chachra, Asma Ben Abacha, Sonya Shooshan, Laritza Rodriguez and Dina Demner-Fushman

Imitation learning for language generation from unaligned data
Gerasimos Lampouras and Andreas Vlachos

Product Review Summarization by Exploiting Phrase Properties
Naitong Yu, Minlie Huang, Yuanyuan Shi and xiaoyan zhu

Generating Questions and Multiple-Choice Answers using Semantic Analysis of Texts
Jun Araki, Dheeraj Rajagopal, Sreecharan Sankaranarayanan, Susan Holm, Yukari Yamakawa and Teruko Mitamura

Resources, Software and Tools

Evaluation Strategies for Computational Construction Grammars
Tania Marques and Katrien Beuls

Building a Monolingual Parallel Corpus for Text Simplification Using Sentence Similarity Based on Alignment between Word Embeddings
Tomoyuki Kajiwara and Mamoru Komachi

Word2Vec vs DBnary: Augmenting METEOR using Vector Representations or Lexical Resources?
Christophe Servan, Alexandre Berard, zied elloumi, Hervé Blanchon and Laurent Besacier

Broad Twitter Corpus: A Diverse Named Entity Recognition Resource
Leon Derczynski, Kalina Bontcheva and Ian Roberts

Semantic overfitting: what 'world' do we consider when evaluating disambiguation of text?
Filip Ilievski, Marten Postma and Piek Vossen

Information Retrieval, Information Extraction, Question Answering

Extraction of Keywords of Novelties From Patent Claims
Shoko Suzuki and Hiromichi Takatsuka

Leveraging Multilingual Training for Limited Resource Event Extraction
Andrew Hsi, Yiming Yang, Jaime Carbonell and Ruochen Xu

LILI: A Simple Language Independent Approach for Language Identification
Mohamed Al-Badrashiny and Mona Diab

High Accuracy Rule-based Question Classification using Question Syntax and Semantics
Harish Tayyar Madabushi and Mark Lee

Incorporating Label Dependency for Answer Quality Tagging in Community Question Answering via CNN-LSTM-CRF
Yang Xiang, Xiaoqiang Zhou, Qingcai Chen, Zhihui Zheng, Buzhou Tang, Xiaolong Wang and Yang Qin

Semantically Motivated Hebrew Verb-Noun Multi-Word Expressions Identification
Chaya Liebeskind and Yaakov HaCohen-Kerner

Thursday, December 15, 2016

09:00–10:00 **Invited talk 3: Dina Demner-Fushman (U.S. National Library of Medicine)**

10:00–10:30 ***coffee break***

10:30–12:30 **Session 5-A: Semantic Processing, Distributional Semantics, Compositionality I**

Semantic Relation Classification via Hierarchical Recurrent Neural Network with Attention
Minguang Xiao and Cong Liu

A Unified Architecture for Semantic Role Labeling and Relation Classification
Jiang Guo, Wanxiang Che, Haifeng Wang, Ting Liu and Jun Xu

Facing the most difficult case of Semantic Role Labeling: A collaboration of word embeddings and co-training
Quynh Ngoc Thi Do, Steven Bethard and Marie-Francine Moens

Predictability of Distributional Semantics in Derivational Word Formation
Sebastian Padó, Aurélie Herbelot, Max Kisselew and Jan Šnajder

10:30–12:30 **Session 5-B: Computational Psycholinguistics and Linguistic Issues in NLP II**

Survey on the Use of Typological Information in Natural Language Processing
Helen O'Horan, Yevgeni Berzak, Ivan Vulic, Roi Reichart and Anna Korhonen

From phonemes to images: levels of representation in a recurrent neural model of visually-grounded language learning
Lieke Gelderloos and Grzegorz Chrupała

Linguistic features for Hindi light verb construction identification
Ashwini Vaidya, Sumeet Agarwal and Martha Palmer

Cross-lingual Transfer of Correlations between Parts of Speech and Gaze Features
Maria Barrett, Frank Keller and Anders Søgaard

10:30–12:30 Session 5-C: Lexical Semantics, Ontologies & Paraphrasing, Textual Entailment I

Sentence Similarity Learning by Lexical Decomposition and Composition
Zhiguo Wang, Haitao Mi and Abraham Ittycheriah

Chinese Hypernym-Hyponym Extraction from User Generated Categories
Chengyu Wang and Xiaofeng He

Dynamic Generative model for Diachronic Sense Emergence Detection
Martin Emms and Arun kumar Jayapal

Semi-supervised Word Sense Disambiguation with Neural Models
Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans and Eric Altendorf

10:30–12:30 Session 5-D: Machine Translation I

Fast Gated Neural Domain Adaptation: Language Model as a Case Study
Jian Zhang, Xiaofeng Wu, Andy Way and Qun Liu

Machine Translation Evaluation for Arabic using Morphologically-enriched Embeddings
Francisco Guzmán, Houda Bouamor, Ramy Baly and Nizar Habash

Ensemble Learning for Multi-Source Neural Machine Translation
Ekaterina Garmash and Christof Monz

Phrase-based Machine Translation using Multiple Preordering Candidates
Yusuke Oda, Taku Kudo, Tetsuji Nakagawa and Taro Watanabe

10:30–12:30 Session 5-P: Poster Session 4

Information Retrieval, Information Extraction, Question Answering

Hand in Glove: Deep Feature Fusion Network Architectures for Answer Quality Prediction in Community Question Answering
Sai Praneeth Suggu, Kushwanth Naga Goutham, Manoj K. Chinnakotla and Manish Shrivastava

Learning Event Expressions via Bilingual Structure Projection
Fangyuan Li, Ruihong Huang, Deyi Xiong and Min Zhang

Global Inference to Chinese Temporal Relation Extraction
Peifeng Li, Qiaoming Zhu, Guodong Zhou and Hongling Wang

Improved relation classification by deep recurrent neural networks with data augmentation
Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu and Zhi Jin

Relation Extraction with Multi-instance Multi-label Convolutional Neural Networks
Xiaotian Jiang, Quan Wang, Peng Li and Bin Wang

Named Entity Disambiguation for little known referents: a topic-based approach
Andrea Glaser and Jonas Kuhn

Natural Language Generation, Summarization

Building RDF Content for Data-to-Text Generation
Laura Perez-Beltrachini, Rania SAYED and Claire Gardent

Parallel Sentence Compression
Julia Ive and François Yvon

An Unsupervised Multi-Document Summarization Framework Based on Neural Document Model
Shulei Ma, Zhi-Hong Deng and Yunlun Yang

From OpenCCG to AI Planning: Detecting Infeasible Edges in Sentence Generation
Maximilian Schwenger, Alvaro Torralba, Joerg Hoffmann, David M. Howcroft and Vera Demberg

The Next Step for Multi-Document Summarization: A Heterogeneous Multi-Genre Corpus Built with a Novel Construction Approach
Markus Zopf, Maxime Peyrard and Judith Eckle-Kohler

Sentiment Analysis and Computational Argumentation

SentiHood: Targeted Aspect Based Sentiment Analysis Dataset for Urban Neighbourhoods
Marzieh Saeidi, Guillaume Bouchard, Maria Liakata and Sebastian Riedel

On the Impact of Seed Words on Sentiment Polarity Lexicon Induction
Dame Jovanoski, Veno Pachovski and Preslav Nakov

Evaluating Argumentative and Narrative Essays using Graphs
Swapna Somasundaran, Brian Riordan, Binod Gyawali and Su-Youn Yoon

Selective Co-occurrences for Word-Emotion Association
Ameeta Agrawal and Aijun An

Weighted Neural Bag-of-n-grams Model: New Baselines for Text Classification
Bofang Li, Zhe Zhao, Tao Liu, Puwei Wang and Xiaoyong Du

A Deeper Look into Sarcastic Tweets Using Deep Convolutional Neural Networks
Soujanya Poria, Erik Cambria, Devamanyu Hazarika and Prateek Vij

Exploring Distributional Representations and Machine Translation for Aspect-based Cross-lingual Sentiment Classification.
Jeremy Barnes, Patrik Lambert and Toni Badia

A Bilingual Attention Network for Code-switched Emotion Prediction
Zhongqing Wang, Yue Zhang, Sophia Lee, Shoushan Li and Guodong Zhou

UTCNN: a Deep Learning Model of Stance Classification on Social Media Text
Wei-Fan Chen and Lun-Wei Ku

Computational Psycholinguistics

The Role of Intrinsic Motivation in Artificial Language Emergence: a Case Study on Colour
Miquel Cornudella, Thierry Poibeau and Remi van Trijp

Predicting the Evocation Relation between Lexicalized Concepts
Yoshihiko Hayashi

Collecting and Exploring Everyday Language for Predicting Psycholinguistic Properties of Words
Gustavo Paetzold and Lucia Specia

Applications

Using Argument Mining to Assess the Argumentation Quality of Essays
Henning Wachsmuth, Khalid Al Khatib and Benno Stein

Grammatical Templates: Improving Text Difficulty Evaluation for Language Learners
Shuhan Wang and Erik Andersen

Still not there? Comparing Traditional Sequence-to-Sequence Models to Encoder-Decoder Neural Networks on Monotone String Translation Tasks
Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh and Iryna Gurevych

12:30–14:00 ***Lunch break***

14:00–16:00 Session 6-A: Information Retrieval, Information Extraction, Question Answering I

Towards Time-Aware Knowledge Graph Completion
Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li and Zhifang Sui

Learning to Weight Translations using Ordinal Linear Regression and Query-generated Training Data for Ad-hoc Retrieval with Long Queries
Javid Dadashkarimi, Masoud Jalili Sabet and Azadeh Shakery

Neural Attention for Learning to Rank Questions in Community Question Answering
Salvatore Romeo, Giovanni Da San Martino, Alberto Barrón-Cedeño, Alessandro Moschitti, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Mitra Mohtarami and James Glass

Simple Question Answering by Attentive Convolutional Neural Network
Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou and Hinrich Schütze

14:00–16:00 Session 6-B: Machine Learning for NLP II

Recurrent Dropout without Memory Loss
Stanislau Semeniuta, Aliaksei Severyn and Erhardt Barth

Modeling topic dependencies in semantically coherent text spans with copulas
Georgios Balikas, Hesam Amoualian, Marianne Clausel, Eric Gaussier and Massih R Amini

Consensus Attention-based Neural Networks for Chinese Reading Comprehension
Yiming Cui, Ting Liu, Zhipeng Chen, Shijin Wang and Guoping Hu

Semantic Annotation Aggregation with Conditional Crowdsourcing Models and Word Embeddings
Paul Felt, Eric Ringger and Kevin Seppi

14:00–16:00 Session 6-C: Machine Translation II

Interactive-Predictive Machine Translation based on Syntactic Constraints of Prefix
Na Ye, Guiping Zhang and Dongfeng Cai

Topic-Informed Neural Machine Translation
Jian Zhang, Liangyou Li, Andy Way and Qun Liu

A Distribution-based Model to Learn Bilingual Word Embeddings
Hailong Cao, Tiejun Zhao, Shu ZHANG and Yao Meng

Pre-Translation for Neural Machine Translation
Jan Niehues, Eunah Cho, Thanh-Le Ha and Alex Waibel

14:00–16:00 Session 6-D: Semantic Processing, Distributional Semantics, Compositionality II

Direct vs. indirect evaluation of distributional thesauri
Vincent Claveau and Ewa Kijak

D-GloVe: A Feasible Least Squares Model for Estimating Word Embedding Densities
Shoaib Jameel and Steven Schockaert

Predicting human similarity judgments with distributional models: The value of word associations.
Simon De Deyne, Amy Perfors and Daniel J Navarro

Distributional Hypernym Generation by Jointly Learning Clusters and Projections
Josuke Yamane, Tomoya Takatani, Hitoshi Yamada, Makoto Miwa and Yutaka Sasaki

14:00–16:00 Session 6-P: Poster Session 5

Discourse Relations, Coreference, Pragmatics

Incremental Fine-grained Information Status Classification Using Attention-based LSTMs
Yufang Hou

Detection, Disambiguation and Argument Identification of Discourse Connectives in Chinese Discourse Parsing
Yong-Siang Shih and Hsin-Hsi Chen

Multi-view and multi-task training of RST discourse parsers
Chloé Braud, Barbara Plank and Anders Søgaard

Implicit Discourse Relation Recognition with Context-aware Character-enhanced Embeddings
Lianhui Qin, Zhisong Zhang and Hai Zhao

Measuring Non-cooperation in Dialogue
Brian Plüss and Paul Piwek

Representation and Learning of Temporal Relations
Leon Derczynski

Revisiting the Evaluation for Cross Document Event Coreference
Shyam Upadhyay, Nitish Gupta, Christos Christodoulopoulos and Dan Roth

Modeling Discourse Segments in Lyrics Using Repeated Patterns
Kento Watanabe, Yuichiroh Matsubayashi, Naho Orita, Naoaki Okazaki, Kentaro Inui, Satoru Fukayama, Tomoyasu Nakano, Jordan Smith and Masataka Goto

Dialog Processing and Dialog Systems, Multimodal Interfaces

Multi-level Gated Recurrent Neural Network for dialog act classification
Wei Li and Yunfang Wu

Multimodal Mood Classification - A Case Study of Differences in Hindi and Western Songs
Braja Gopal Patra, Dipankar Das and Sivaji Bandyopadhyay

Detecting Context Dependent Messages in a Conversational Environment
Chaozhuo Li, Yu Wu, Wei Wu, Chen Xing, Zhoujun Li and Ming Zhou

Joint Inference for Mode Identification in Tutorial Dialogues
Deepak Venugopal and Vasile Rus

Dialogue Act Classification in Domain-Independent Conversations Using a Deep Recurrent Neural Network
Hamed Khanpour, Nishitha Guntakandla and Rodney Nielsen

Non-sentential Question Resolution using Sequence to Sequence Learning
Vineet Kumar and Sachindra Joshi

Context-aware Natural Language Generation for Spoken Dialogue Systems
Hao Zhou, Minlie Huang and xiaoyan zhu

Speech Recognition, Text-To-Speech, Spoken Language Understanding

Weakly-supervised text-to-speech alignment confidence measure
Guillaume Serrière, Christophe Cerisara, Dominique Fohr and Odile Mella

Domainless Adaptation by Constrained Decoding on a Schema Lattice
Young-Bum Kim, Karl Stratos and Ruhi Sarikaya

Sub-Word Similarity based Search for Embeddings: Inducing Rare-Word Embeddings for Word Similarity Tasks and Language Modelling
Mittul Singh, Clayton Greenberg, Youssef Oualil and Dietrich Klakow

Applications

Semi-automatic Detection of Cross-lingual Marketing Blunders based on Pragmatic Label Propagation in Wiktionary
Christian M. Meyer, Judith Eckle-Kohler and Iryna Gurevych

Ambient Search: A Document Retrieval System for Speech Streams
Benjamin Milde, Jonas Wacker, Stefan Radomski, Max Mühlhäuser and Chris Biemann

Semi-supervised Gender Classification with Joint Textual and Social Modeling
Shoushan Li, Bin Dai, Zhengxian Gong and Guodong Zhou

Predicting proficiency levels in learner writings by transferring a linguistic complexity model from expert-written coursebooks
Ildikó Pilán, Elena Volodina and Torsten Zesch

User Classification with Multiple Textual Perspectives
Dong Zhang, Shoushan Li, Hongling Wang and Guodong Zhou

Says Who…? Identification of Expert versus Layman Critics' Reviews of Documentary Films
Ming Jiang and Jana Diesner

Knowledge-Driven Event Embedding for Stock Prediction
Xiao Ding, Yue Zhang, Ting Liu and Junwen Duan

Distributed Representations for Building Profiles of Users and Items from Text Reviews
Wenliang Chen, Zhenjie Zhang, Zhenghua Li and Min Zhang

16:00–16:30 ***coffee break***

16:30–18:00 Session 7-A: Machine Translation III

Improving Statistical Machine Translation with Selectional Preferences
Haiqing Tang, Deyi Xiong, Min Zhang and Zhengxian Gong

Hierarchical Permutation Complexity for Word Order Evaluation
Miloš Stanojević and Khalil Sima'an

Interactive Attention for Neural Machine Translation
Fandong Meng, Zhengdong Lu, Hang Li and Qun Liu

16:30–18:00 Session 7-B: Applications IV

Get Semantic With Me! The Usefulness of Different Feature Types for Short-Answer Grading
Ulrike Pado

Automatically Processing Tweets from Gang-Involved Youth: Towards Detecting Loss and Aggression
Terra Blevins, Robert Kwiatkowski, Jamie MacBeth, Kathleen McKeown, Desmond Patton and Owen Rambow

Content-based Influence Modeling for Opinion Behavior Prediction
Chengyao Chen, Zhitao Wang, Yu Lei and Wenjie Li

16:30–18:00 Session 7-C: Computational Psycholinguistics and Linguistic Issues in NLP III

Data-driven learning of symbolic constraints for a log-linear model in a phonological setting
Gabriel Doyle and Roger Levy

Chinese Tense Labelling and Causal Analysis
Hen-Hsen Huang, Chang-Rui Yang and Hsin-Hsi Chen

Exploring Topic Discriminating Power of Words in Latent Dirichlet Allocation
Yang Kai, Cai Yi, Chen Zhenhong, Leung Ho-fung and LAU Raymond

16:30–18:00 Session 7-D: Lexical Semantics, Ontologies & Paraphrasing, Textual Entailment II

Textual Entailment with Structured Attentions and Composition
Kai Zhao, Liang Huang and Mingbo Ma

plWordNet 3.0 – a Comprehensive Lexical-Semantic Resource
Marek Maziarz, Maciej Piasecki, Ewa Rudnicka, Stan Szpakowicz and Paweł Kędzia

Time-Independent and Language-Independent Extraction of Multiword Expressions From Twitter
Nikhil Londhe, Rohini Srihari and Vishrawas Gopalakrishnan

16:30–18:00 Session 7-P: Poster Session 6

Information Retrieval, Information Extraction, Question Answering

Incremental Global Event Extraction
Alex Judea and Michael Strube

Hierarchical Memory Networks for Answer Selection on Unknown Words
jiaming xu, Jing Shi, Yiqun Yao, Suncong Zheng, Bo Xu and Bo Xu

Revisiting Taxonomy Induction over Wikipedia
Amit Gupta, Francesco Piccinno, Mikhail Kozhevnikov, Marius Pasca and Daniele Pighin

Joint Learning of Local and Global Features for Entity Linking via Neural Networks
Thien Huu Nguyen, Nicolas Fauceglia, Mariano Rodriguez Muro, Oktie Hassanzadeh, Alfio Massimiliano Gliozzo and Mohammad Sadoghi

Structured Aspect Extraction
Omer Gunes, Tim Furche and Giorgio Orsi

Robust Text Classification for Sparsely Labelled Data Using Multi-level Embeddings
Simon Baker, Douwe Kiela and Anna Korhonen

Mathematical Information Retrieval based on Type Embeddings and Query Expansion
Yiannos Stathopoulos and Simone Teufel

Text Retrieval by Term Co-occurrences in a Query-based Vector Space
Eriks Sneiders

Pairwise Relation Classification with Mirror Instances and a Combined Convolutional Neural Network
Jianfei Yu and Jing Jiang

FastHybrid: A Hybrid Model for Efficient Answer Selection
Lidan Wang, Ming Tan and Jiawei Han

Extracting Spatial Entities and Relations in Korean Text
Bogyum Kim and Jae Sung Lee

Hybrid Question Answering over Knowledge Base and Free Text
kun xu, Yansong Feng, Songfang Huang and Dongyan Zhao

Improved Word Embeddings with Implicit Structure Information
Jie Shen and Cong Liu

Sentiment Analysis, Computational Argumentation

Word Embeddings and Convolutional Neural Network for Arabic Sentiment Classification
Abdelghani Dahou, Shengwu Xiong, Junwei Zhou, Mohamed Houcine Haddoud and Pengfei Duan

Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts
Xingyou Wang, Weijie Jiang and Zhiyong Luo

Stance Classification in Rumours as a Sequential Task Exploiting the Tree Structure of Social Media Conversations
Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter and Michal Lukasik

Tweet Sarcasm Detection Using Deep Neural Network
Meishan Zhang, Yue Zhang and Guohong Fu

Agreement and Disagreement: Comparison of Points of View in the Political Domain
Stefano Menini and Sara Tonelli

Targeted Sentiment to Understand Student Comments
Charles Welch and Rada Mihalcea

Thursday, December 15, 2016 (continued)

Towards Sub-Word Level Compositions for Sentiment Analysis of Hindi-English Code Mixed Text
Aditya Joshi, Ameya Prabhu, Manish Shrivastava and Vasudeva Varma

Distance Metric Learning for Aspect Phrase Grouping
Shufeng Xiong, Yue Zhang, Donghong JI and Yinxia Lou

Friday, December 16, 2016

09:00–10:00 Invited talk 4: Simone Teufel (University of Cambridge)

10:00–10:30 ***coffee break***

10:30–12:30 Session 8-A: Information Retrieval, Information Extraction, Question Answering II

Constraint-Based Question Answering with Knowledge Graph
Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou and Tiejun Zhao

Selecting Sentences versus Selecting Tree Constituents for Automatic Question Ranking
Alberto Barrón-Cedeño, Giovanni Da San Martino, Salvatore Romeo and Alessandro Moschitti

Attention-Based Convolutional Neural Network for Semantic Relation Extraction
yatian shen and Xuanjing Huang

Table Filling Multi-Task Recurrent Neural Network for Joint Entity and Relation Extraction
Pankaj Gupta, Hinrich Schütze and Bernt Andrassy

10:30–12:30 Session 8-B: Machine Translation IV

Bilingual Autoencoders with Global Descriptors for Modeling Parallel Sentences
Biao Zhang, Deyi Xiong, jinsong su, Hong Duan and Min Zhang

Multi-Engine and Multi-Alignment Based Automatic Post-Editing and its Impact on Translation Productivity
Santanu Pal, Sudip Kumar Naskar and Josef van Genabith

Measuring the Effect of Conversational Aspects on Machine Translation Quality
Marlies van der Wees, Arianna Bisazza and Christof Monz

Enriching Phrase Tables for Statistical Machine Translation Using Mixed Embeddings
Peyman Passban, Qun Liu and Andy Way

10:30–12:30 Session 8-C: Discourse Relations, Coreference, Pragmatics

Anecdote Recognition and Recommendation
Wei Song, Ruiji Fu, Lizhen Liu, Hanshi Wang and Ting Liu

Training Data Enrichment for Infrequent Discourse Relations
Kailang Jiang, Giuseppe Carenini and Raymond Ng

Inferring Discourse Relations from PDTB-style Discourse Labels for Argumentative Revision Classification
Fan Zhang, Diane Litman and Katherine Forbes-Riley

Capturing Pragmatic Knowledge in Article Usage Prediction using LSTMs
Jad Kabbara, Yulan Feng and Jackie Chi Kit Cheung

10:30–12:30 Session 8-D: Sentiment Analysis, Computational Argumentation I

Aspect Based Sentiment Analysis using Sentiment Flow with Local and Non-local Neighbor Information
Shubham Pateria

Two-View Label Propagation to Semi-supervised Reader Emotion Classification
Shoushan Li, Jian Xu, Dong Zhang and Guodong Zhou

A Joint Sentiment-Target-Stance Model for Stance Classification in Tweets
Javid Ebrahimi, Dejing Dou and Daniel Lowd

SenticNet 4: A Semantic Resource for Sentiment Analysis Based on Conceptual Primitives
Erik Cambria, Soujanya Poria, Rajiv Bajpai and Bjoern Schuller

10:30–12:30 Session 8-P: Poster Session 7

Machine Learning for NLP

Joint Embedding of Hierarchical Categories and Entities for Concept Categorization and Dataless Classification
Yuezhang(Music) Li, Ronghuo Zheng, Tian Tian, Zhiting Hu, Rahul Iyer and Katia Sycara

Latent Topic Embedding
Di Jiang, Lei Shi, Rongzhong Lian and Hua Wu

Neural-based Noise Filtering from Word Embeddings
Kim Anh Nguyen, Sabine Schulte im Walde and Ngoc Thang Vu

Integrating Distributional and Lexical Information for Semantic Classification of Words using MRMF
Rosa Tsegaye Aga, Lucas Drumond, Christian Wartena and Lars Schmidt-Thieme

Semi Supervised Preposition-Sense Disambiguation using Multilingual Data
Hila Gonen and Yoav Goldberg

Monday mornings are my fave :) #not Exploring the Automatic Recognition of Irony in English tweets
Cynthia Van Hee, Els Lefever and Veronique Hoste

CNN- and LSTM-based Claim Classification in Online User Comments
Chinnappa Guggilla, Tristan Miller and Iryna Gurevych

Experiments in Idiom Recognition
JIng Peng and Anna Feldman

An Empirical Evaluation of various Deep Learning Architectures for Bi-Sequence Classification Tasks
Anirban Laha and Vikas Raykar

Learning Succinct Models: Pipelined Compression with L1-Regularization, Hashing, Elias-Fano Indices, and Quantization
Hajime Senuma and Akiko Aizawa

Semantic Processing, Distributional Semantics, Compositionality

Bad Company—Neighborhoods in Neural Embedding Spaces Considered Harmful
Johannes Hellrich and Udo Hahn

Implementing a Reverse Dictionary, based on word definitions, using a Node-Graph Architecture
Sushrut Thorat and Varad Choudhari

Is an Image Worth More than a Thousand Words? On the Fine-Grain Semantic Differences between Visual and Linguistic Representations
Guillem Collell and Marie-Francine Moens

On the contribution of word embeddings to temporal relation classification
Paramita Mirza and Sara Tonelli

Modeling Context-sensitive Selectional Preference with Distributed Representations
Naoya Inoue, Yuichiroh Matsubayashi, Masayuki Ono, Naoaki Okazaki and Kentaro Inui

Exploring the value space of attributes: Unsupervised bidirectional clustering of adjectives in German
Wiebke Petersen and Oliver Hellwig

Distributional Inclusion Hypothesis for Tensor-based Composition
Dimitri Kartsaklis and Mehrnoosh Sadrzadeh

Parameter estimation of Japanese predicate argument structure analysis model using eye gaze information
Ryosuke Maki, Hitoshi Nishikawa and Takenobu Tokunaga

Paraphrasing, Textual Entailment

Reading and Thinking: Re-read LSTM Unit for Textual Entailment Recognition
Lei Sha, Baobao Chang, Zhifang Sui and Sujian Li

A Paraphrase and Semantic Similarity Detection System for User Generated Short-Text Content on Microblogs
Kuntal Dey, Ritvik Shrivastava and Saroj Kaushik

Modeling Extractive Sentence Intersection via Subtree Entailment
Omer Levy, Ido Dagan, Gabriel Stanovsky, Judith Eckle-Kohler and Iryna Gurevych

Context-Sensitive Inference Rule Discovery: A Graph-Based Method
Xianpei Han and Le Sun

Modelling Sentence Pairs with Tree-structured Attentive Encoder
Yao Zhou, Cong Liu and Yan Pan

Neural Paraphrase Generation with Stacked Residual LSTM Networks
aaditya prakash, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu and Oladimeji Farri

12:30–14:00 ***Lunch break***

14:00–15:30 **Session 9-A: Information Retrieval, Information Extraction, Question Answering III**

English-Chinese Knowledge Base Translation with Neural Network
Xiaocheng Feng, Duyu Tang, Bing Qin and Ting Liu

Keyphrase Annotation with Graph Co-Ranking
Adrien Bougouin, Florian Boudin and Beatrice Daille

What's in an Explanation? Characterizing Knowledge and Inference Requirements for Elementary Science Exams
Peter Jansen, Niranjan Balasubramanian, Mihai Surdeanu and Peter Clark

Friday, December 16, 2016 (continued)

14:00–15:30 Session 9-B: Sentiment Analysis, Computational Argumentation II

"All I know about politics is what I read in Twitter": Weakly Supervised Models for Extracting Politicians' Stances From Twitter
Kristen Johnson and Dan Goldwasser

Leveraging Multiple Domains for Sentiment Classification
Fan Yang, Arjun Mukherjee and Yifan Zhang

Political News Sentiment Analysis for Under-resourced Languages
Patrik F. Bakken, Terje A. Bratlie, Cristina Marco and Jon Atle Gulla

14:00–15:30 Session 9-C: Applications V

Fast Inference for Interactive Models of Text
Jeffrey Lund, Paul Felt, Kevin Seppi and Eric Ringger

Combining Heterogeneous User Generated Data to Sense Well-being
Adam Tsakalidis, Maria Liakata, Theo Damoulas, Brigitte Jellinek, Weisi Guo and Alexandra Cristea

Hashtag Recommendation with Topical Attention-Based LSTM
Yang Li, Ting Liu, Jing Jiang and Liang Zhang

14:00–15:30 Session 9-D: Resources, Software, Tools & Under-resourced languages III

Better call Saul: Flexible Programming for Learning and Inference in NLP
Parisa Kordjamshidi, Daniel Khashabi, Christos Christodoulopoulos, Bhargav Mangipudi, Sameer Singh and Dan Roth

Crowdsourcing Complex Language Resources: Playing to Annotate Dependency Syntax
Bruno Guillaume, Karën Fort and Nicolas Lefebvre

Borrow a Little from your Rich Cousin: Using Embeddings and Polarities of English Words for Multilingual Sentiment Classification
Prerana Singhal and Pushpak Bhattacharyya

14:00–15:30 Session 9-P: Poster Session 8

Machine Translation

A Character-Aware Encoder for Neural Machine Translation
Zhen Yang, Wei Chen, Feng Wang and Bo Xu

Convolution-Enhanced Bilingual Recursive Neural Network for Bilingual Semantic Modeling
jinsong su, Biao Zhang, Deyi Xiong, Ruochen Li and Jianmin Yin

Improving Attention Modeling with Implicit Distortion and Fertility for Machine Translation
Shi Feng, Shujie Liu, Nan Yang, Mu Li, Ming Zhou and Kenny Q. Zhu

Neural Machine Translation with Supervised Attention
Lemao Liu, Masao Utiyama, Andrew Finch and Eiichiro Sumita

Lightly Supervised Quality Estimation
Matthias Sperber, Graham Neubig, Jan Niehues, Sebastian Stüker and Alex Waibel

Improving Translation Selection with Supersenses
Haiqing Tang, Deyi Xiong, Oier Lopez de Lacalle and Eneko Agirre

Is all that Glitters in Machine Translation Quality Estimation really Gold?
Yvette Graham, Timothy Baldwin, Meghan Dowling, Maria Eskevich, Teresa Lynn and Lamia Tounsi

Connecting Phrase based Statistical Machine Translation Adaptation
Rui Wang, Hai Zhao, Bao-Liang Lu, Masao Utiyama and Eiichiro Sumita

Fast Collocation-Based Bayesian HMM Word Alignment
Philip Schulz and Wilker Aziz

Learning to translate from graded and negative relevance information
Laura Jehl and Stefan Riezler

Universal Reordering via Linguistic Typology
Joachim Daiber, Miloš Stanojević and Khalil Sima'an

A Deep Fusion Model for Domain Adaptation in Phrase-based MT
Nadir Durrani, Hassan Sajjad, Shafiq Joty and Ahmed Abdelali

Inducing Bilingual Lexica From Non-Parallel Data With Earth Mover's Distance Regularization
Meng Zhang, Yang Liu, Huanbo Luan, Yiqun Liu and Maosong Sun

What Makes Word-level Neural Machine Translation Hard: A Case Study on English-German Translation
Fabian Hirschmann, Jinseok Nam and Johannes Fürnkranz

Improving Word Alignment of Rare Words with Word Embeddings
Masoud Jalili Sabet, Heshaam Faili and Gholamreza Haffari

Applications

Measuring the Information Content of Financial News
Ching-Yun Chang, Yue Zhang, Zhiyang Teng, Zahn Bozanic and Bin Ke

Automatic Generation and Classification of Minimal Meaningful Propositions in Educational Systems
Andreea Godea, Florin Bulgarov and Rodney Nielsen

First Story Detection using Entities and Relations
Nikolaos Panagiotou, Cem Akkaya, Kostas Tsioutsiouliklis, Vana Kalogeraki and Dimitrios Gunopulos

Textual complexity as a predictor of difficulty of listening items in language proficiency tests
Anastassia Loukina, Su-Youn Yoon, Jennifer Sakano, Youhua Wei and Kathy Sheehan

The Construction of a Chinese Collocational Knowledge Resource and Its Application for Second Language Acquisition
Renfen HU, Jiayong Chen and Kuang-hua Chen

15:30–16:00 ***coffee break***

16:00–17:00 Session 10-A: Information Retrieval, Information Extraction, Question Answering IV

Joint Inference for Event Coreference Resolution
Jing Lu, Deepak Venugopal, Vibhav Gogate and Vincent Ng

Event Detection with Burst Information Networks
Tao Ge, Lei Cui, Baobao Chang, Zhifang Sui and Ming Zhou

16:00–17:30 Session 10-B: Sentiment Analysis, Computational Argumentation III

Corpus Fusion for Emotion Classification
Suyang Zhu, Shoushan Li, Ying Chen and Guodong Zhou

Effective LSTMs for Target-Dependent Sentiment Classification
Duyu Tang, Bing Qin, Xiaocheng Feng and Ting Liu

Towards assessing depth of argumentation
Manfred Stede

16:00–17:30 Session 10-C: Applications VI

Video Event Detection by Exploiting Word Dependencies from Image Captions
Sang Phan, Yusuke Miyao, Duy-Dinh Le and Shin'ichi Satoh

Predicting Restaurant Consumption Level through Social Media Footprints
Yang Xiao, Yuan Wang, Hangyu Mao and Zhen Xiao

A Novel Fast Framework for Topic Labeling Based on Similarity-preserved Hashing
Xian-Ling Mao, Yi-Jing Hao, Qiang Zhou, Wen-Qing Yuan, Liner Yang and Heyan Huang

16:00–17:30 Session 10-D: Dialog Processing and Dialog Systems, Multimodal Interfaces

Sequence to Backward and Forward Sequences: A Content-Introducing Approach to Generative Short-Text Conversation
Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang and Zhi Jin

Disfluent but effective? A quantitative study of disfluencies and conversational moves in team discourse
Felix Gervits, Kathleen Eberhard and Matthias Scheutz

A Neural Network Approach for Knowledge-Driven Response Generation
Pavlos Vougiouklis, Jonathon Hare and Elena Simperl

16:00–17:30 Session 10-P: Poster Session 9

Resources, Software and Tools

PersoNER: Persian Named-Entity Recognition
Hanieh Poostchi, Ehsan Zare Borzeshi, Mohammad Abdous and Massimo Piccardi

OCR++: A Robust Framework For Information Extraction from Scholarly Articles
Mayank Singh, Barnopriyo Barua, Priyank Palod, Manvi Garg, Sidhartha Satapathy, Samuel Bushi, Kumar Ayush, Krishna Sai Rohith, Tulasi Gamidi, Pawan Goyal and Animesh Mukherjee

Efficient Data Selection for Bilingual Terminology Extraction from Comparable Corpora
Amir Hazem and Emmanuel Morin

TweetGeo - A Tool for Collecting, Processing and Analysing Geo-encoded Linguistic Data
Nikola Ljubešić, Tanja Samardzic and Curdin Derungs

Extending WordNet with Fine-Grained Collocational Information via Supervised Distributional Learning
Luis Espinosa Anke, Jose Camacho-Collados, Sara Rodríguez-Fernández, Horacio Saggion and Leo Wanner

A News Editorial Corpus for Mining Argumentation Strategies
Khalid Al Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen and Benno Stein

Universal Dependencies for Turkish
Umut Sulubacak, Memduh Gokirmak, Francis Tyers, Çağrı Çöltekin, Joakim Nivre and Gülşen Eryiğit

Creating Resources for Dialectal Arabic from a Single Annotation: A Case Study on Egyptian and Levantine
Ramy Eskander, Nizar Habash, Owen Rambow and Arfath Pasha

Multilingual Aliasing for Auto-Generating Proposition Banks
Alan Akbik, Xinyu Guan and Yunyao Li

PanPhon: A Resource for Mapping IPA Segments to Articulatory Feature Vectors
David R. Mortensen, Patrick Littell, Akash Bharadwaj, Kartik Goyal, Chris Dyer and Lori Levin

Semantic Processing, Distributional Semantics, Compositionality

Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling
Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao and Bo Xu

More is not always better: balancing sense distributions for all-words Word Sense Disambiguation
Marten Postma, Ruben Izquierdo Bevia and Piek Vossen

Language classification from bilingual word embedding graphs
Steffen Eger, Armin Hoenen and Alexander Mehler

Word Embeddings, Analogies, and Machine Learning: Beyond king - man + woman = queen
Aleksandr Drozd, Anna Gladkova and Satoshi Matsuoka

Semantic Tagging with Deep Residual Networks
Johannes Bjerva, Barbara Plank and Johan Bos

Lexical Semantics, Ontologies

A Supervised Approach for Enriching the Relational Structure of Frame Semantics in FrameNet
Shafqat Mumtaz Virk, Philippe Muller and Juliette Conrath

Reddit Temporal N-gram Corpus and its Applications on Paraphrase and Semantic Similarity in Social Media using a Topic-based Latent Semantic Analysis
Anh Dang, Abidalrahman Moh'd, Aminul Islam, Rosane Minghim, Michael Smit and Evangelos Milios

Dictionaries as Networks: Identifying the graph structure of Ogden's Basic English
Camilo Garrido and Claudio Gutierrez

Structured Generative Models of Continuous Features for Word Sense Induction
Alexandros Komninos and Suresh Manandhar

17:45–18:30 Closing

Boosting for Efficient Model Selection for Syntactic Parsing

Rachel Bawden[◇*] **Benoît Crabbé**[†‡]
[◇] LIMSI, CNRS, Univ. Paris-Sud, Universite Paris-Saclay
[†] Alpage, Univ. Paris-Diderot (Sorbonne Paris Cité) & Inria
[‡] Institut Universitaire de France
rachel.bawden@limsi.fr
benoit.crabbe@linguist.univ-paris-diderot.fr

Abstract

We present an efficient model selection method using boosting for transition-based constituency parsing. It is designed for exploring a high-dimensional search space, defined by a large set of feature templates, as for example is typically the case when parsing morphologically rich languages. Our method removes the need to manually define heuristic constraints, which are often imposed in current state-of-the-art selection methods. Our experiments for French show that the method is more efficient and is also capable of producing compact, state-of-the-art models.

1 Introduction

Model selection in feature-based parsing is crucial because features define a parsing model's capacity to predict syntactic structure. Choosing an optimal model is a trade-off between generalisation performance, compactness and parsing speed. Although too rarely mentioned, to this we should also add the speed of the selection method, which can determine how much of the search space can actually be explored. Parsing of languages other than English, and in particular morphologically rich languages, spurred on by initiatives such as the SPMRL (Statistical Parsing of Morphologically Rich Languages) shared tasks (Seddah et al., 2014), has received a heightened interest in recent years. For such languages, it is natural to want to exploit morphologically rich data to improve parsing performance. However the effect of this is an explosion in the number of possible models due to a huge number of potential features.

In this paper we introduce an efficient, language-independent model selection method for transition-based constituency parsing. It is designed for model selection when faced with a large number of possible feature templates, which is typically the case for morphologically rich languages, for which we want to exploit morphological information. The method we propose uses multi-class boosting (Zhu et al., 2006) for iterative selection in constant time, using virtually no *a priori* constraints on the search space. We do however introduce a pre-ranking step before selection in order to guide the selection process. We provide experiments on the adaptation of boosting for model selection in the parsing of high-dimensional data, using the transition-based lexicalised constituency parser presented in (Crabbé, 2015) and illustrating the feasibility of the method for our working language, French. Our results show that it is possible to produce high-performing, compact models much more efficiently than naive methods.

The structure of the paper is as follows. We begin by describing the transition-based parser used throughout the paper (Section 2). In Section 3 we review related work, both in model selection for parsing (Section 3.1) and on the boosting algorithm used (Section 3.2) in our proposal. In Section 4 we present our adaptation of the method for parsing and in Section 5 our experiments and results.

2 Discriminative constituency parsing

We base our experiments on the multilingual discriminative constituency parser described in (Crabbé, 2014; Crabbé, 2015) and inspired by transition-based parsing algorithms (Zhu et al., 2013). The parser

[*] This work was carried out while the first author was a Master's student at Alpage (Univ. Paris-Diderot & Inria).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1–11, Osaka, Japan, December 11-17 2016.

is an accurate and efficient transition-based lexicalised parser, capable of easily integrating lexical and morphological features. Following standard practice in transition-based parsing, the key structure for parsing is the *configuration* $C = \langle j, \mathbf{S} \rangle$ where j is the index of the parser in the queue of input tokens and $\mathbf{S}$ is a stack of partially constructed tree structures. A derivation $C_{0 \Rightarrow \tau}$ of length τ is represented by a sequence of configurations $C_0 \overset{a_0}{\Rightarrow} \ldots \overset{a_{\tau-1}}{\Rightarrow} C_\tau$ where each $a_i \in A$ is the parser *action* used to move from C_i to C_{i+1}. We use the same set A of actions as described in (Crabbé, 2014). Derivations are scored by a function of the form:

$$W(C_{0 \Rightarrow \tau}) = \sum_{i=0}^{\tau-1} \mathbf{w} \cdot \mathbf{\Phi}(a_i, C_i)$$

where $\mathbf{w} \in \mathbb{R}^D$ is a weight vector and where $\mathbf{\Phi}(a, C) \in \{0, 1\}^D$ denotes a function encoding a boolean-valued feature vector from a pair (a, C).

Let GEN_τ be the set of derivations of length τ. The best derivation in this set is defined as:

$$\hat{C}_{0 \Rightarrow \tau} = \underset{C_{0 \Rightarrow \tau} \in \text{GEN}_\tau}{\text{argmax}} \; W(C_{0 \Rightarrow \tau})$$

Weights for individual features are learnt using an averaged multi-class perceptron (Collins, 2002). They can either be optimised globally (over sequences of derivations) or locally (for each individual action). The first strategy is known to give better results for perceptron-based parsing (Zhang and Nivre, 2012).

2.1 Feature functions

The feature vector $\mathbf{\Phi}(a, C)$ is the result of a sequence of boolean feature functions $\phi_1(a, C) \ldots \phi_D(a, C)$, which have access to the action, to the top elements of the stack in C and to the beginning of the queue from index j. Each function is defined as per the following pattern:

$$\phi_l(a, C) = \begin{cases} 1 & \textbf{if} \quad \text{attr}_0 = a \\ & \textbf{and} \quad \text{attr}_1 = v_1 \\ & (\textbf{and} \quad \text{attr}_2 = v_2)? \\ & (\textbf{and} \quad \text{attr}_3 = v_3)? \\ 0 & \textbf{otherwise} \end{cases}$$

in which attr_0 is valued by the action $a \in A$ and the attributes $\text{attr}_{1 \leq i \leq 3}$ are extracted from configuration C. In practice, feature functions have access to the top three elements of the stack (see Figure 1) and the first four elements of the queue. They can address non-terminal categories, word-forms and morphological features (such as the part-of-speech (PoS), the gender, the number, the mood etc.) from the heads in the stack and from the words in the queue. Their values v_i are extracted from the configuration C, as illustrated in the following example:

$$\phi_l(a, C) = \begin{cases} 1 & \textbf{if} \quad \text{attr}_0 = a \\ & \textbf{and} \quad q_0.\text{word} = \text{“pot”} \\ & \textbf{and} \quad s_2.\text{cat} = \text{VP} \\ 0 & \textbf{otherwise} \end{cases}$$

As shown in the pattern above, a function can contain up to three attribute-value pairs, excluding the action. We refer to each pair as a condition, and refer to features as being uni-, bi- or tri-conditional depending on the number of conditions they contain.

2.2 Feature templates

It is common practice to use feature templates when defining hand-crafted models rather than to specify individual features. Feature templates are abstract feature representations in which only the attributes are specified, such that features with the same attribute types are grouped into sets. In the case of templates, which can also have up to three conditions, a condition refers simply to the attribute. For example, the bi-conditional template ‘q_0(word) & s_1(t, h, tag)’ represents all bi-conditional feature functions related to the word-form of the first queue element and to the PoS tag of the lexical head associated with the top of the second stack element.

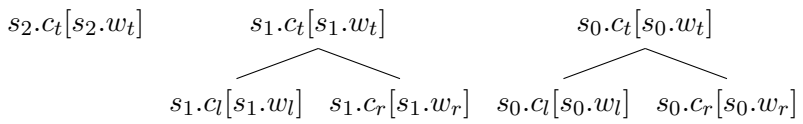


Figure 1: Stack elements for instantiation in feature functions. c_t, c_l and c_r represent non-terminal categories at three positions (top (t), left (l) and right (r)). Lexical information about the head is also available, indicated in squared brackets as $s_i.w_{\{t,l,r\}}$.

Since a feature vector is usually sparse, using templates is an advantage for computational reasons; the values of the feature functions can be dynamically and efficiently bound during parsing. They also enable a more compact and readable representation of models, making models easier to define manually.

When working with parsers that rely on templates, which is the case of most current implementations of existing feature-based parsers (Nilsson and Nugues, 2010; Ballesteros and Nivre, 2014; Crabbé, 2014, etc.), the template seems to be an acceptable level of granularity for specifying models. A selection process at the template-level rather than the feature-level has the advantage of being compatible with these parsers and also reduces the selection time by reducing the combinatorics of the selection process. We therefore focus our work on template selection.

3 Model selection for parsing

3.1 Previous work

For feature-based models, there are two main strategies for model selection. *Filter methods* remove features based on a static analysis, whereas *wrapper methods* iteratively refit the model by forward or backward selection. In the parsing literature, the wrapper method is the most prevalent. While some backward wrappers exist, provided that the template set is small (Attardi et al., 2007), most work focuses on forward wrappers, with a variety of constraints to reduce the search space and thus the time required. Nilsson and Nugues (2010) constrain the search space by imposing an order on stack and queue elements, under the assumption that more local elements are more useful than more distant ones. Ballesteros and Nivre (2014) and Ballesteros and Bohnet (2014) use a combination of forward and backward methods and fix heavy rule-based constraints on the order of templates selected. He et al. (2013) also implement template selection for discriminative parsing. Although applied to graph-based parsing, their work shares a likeness with our own, by their use of a pre-ranking wrapper to order templates prior to selection.

The reason for introducing such constraints is that wrapper methods are computationally intensive and can be known to take weeks to select a model, even with constraints and fully optimised, multi-processed implementations. Take for example the case of the forward wrapper. It starts with an empty model ($M \leftarrow \emptyset$). At each iteration, the template t from the pool of potential templates P that results in the highest overall accuracy gain is added to the model ($M \leftarrow M \cup t$). The process stops when the model's loss ceases to decrease. The most time-consuming part of the process is the training of the possible models, in order to select the template t that results in the highest accuracy gain. In principle, this requires fitting $|P|$ models at each iteration, and the size of the models requiring training grows at each iteration. Given that fitting a single parsing model can take hours (see Section 5.4), it is impractical to perform selection for parsing based on iteratively refitting a series of large and ever-growing models.

3.2 Model selection via boosting

We propose to overcome the limitations of the naive forward wrapper by using sequential additive fitting based on boosting (Freund and Schapire, 1999). Starting with an empty model ($M \leftarrow \emptyset$), additive fitting consists of evaluating the addition of a new template t by fitting t on its own before adding it ($M \leftarrow M \cup t$), without modifying the already fitted content M. At each iteration, as with the naive method, $|P|$ models need to be fit, but they are small and of constant size. As we will show in the

Algorithm 1 SAMME (Zhu et al., 2006)

▷ Data = $\{(x_1, y_1), \ldots, (x_N, y_N)\}$

1: Initialisation of data weights

$$w_i^{(1)} = \frac{1}{N}, \text{ i = 1, 2,..., N}$$

2: **for** iteration t=1 to T **do**

(i) Fit each weak learner $h_j(x)$ in the pool P to data using weights $w_i^{(t)}$

(ii) Calculate the weighted error of each weak learner $h_j(x)$

$$err_{h_j} = \frac{\sum_{i=1}^{N} w_i^{(t)} \mathbb{I}(y_i \neq h_j(x_i))}{\sum_{i=1}^{N} w_i^{(t)}}$$

(iii) Select the $h_j(x)$ with the lowest weighted error err_{h_j} provided that $err_{h_j} > \left(1 - \frac{1}{|Y|}\right)$.

Call the learner $g^{(t)}$ and its weighted error $err^{(t)}$

(iv) Calculate $\alpha^{(t)}$, where Y is the set of classes

$$\alpha^{(t)} = \log\left(\frac{1 - err^{(t)}}{err^{(t)}}\right) + \log\left(|Y| - 1\right)$$

(v) Update data weights using $\alpha^{(t)}$

$$w_i^{(t+1)} = w_i^{(t)} \cdot \exp\left(\alpha^{(t)} \mathbb{I}\left(y_i \neq g^{(t)}(x_i)\right)\right), \text{ i = 1,2,...,N}$$

(vi) Normalise weights such that $\sum_{i=1}^{N} w_i = 1$

end for

3: Prediction is the argmax of a weighted prediction of models $g^{(t)}$, t = 1, 2,..., T

$$f(\text{x}) = \text{argmax}_y \sum_{t=1}^{T} \alpha^{(t)} \mathbb{I}\left(g^{(t)}(x) = y\right)$$

remainder of the paper, this allows for a huge reduction in selection time, meaning that heavy constraints are not needed to reduce the search space.

We use the multi-class AdaBoost variant SAMME (Stagewise Additive Modelling using a Multi-class Exponential loss function) as described in (Zhu et al., 2006) and adapted here in Algorithm 1. The algorithm is designed for predictive modelling and provides the means of combining a set of *weak learners*[1] to produce a strong learner, by additively and iteratively selecting the best weak learner $g^{(t)}$ and calculating its coefficient $\alpha^{(t)}$ (its importance in the final model) until no more weak learners are available. An additive fit is achieved by encoding the exponential loss of already selected learners in a weight distribution over data examples, which is updated at each iteration. The algorithm comes with a theoretical guarantee that as long as the selected learner has a weighted accuracy above chance, the model's boosted accuracy will not decrease.

In the case of parsing, weak learners can be seen as weak parsers, trained each on a very small set of templates. It can be seen as iterative forward selection in that the selection of a weak parser constitutes the selection of the templates on which it is trained. In this paper, we use the term *weak parser* as an alias for the set of templates on which it is trained. Boosting has previously been used for feature selection for automatic face-detection in the domain of imagery (Viola and Jones, 2004) and for a variety of classification tasks by Das (2001). However to our knowledge, boosting methods have not yet been used in the context of template selection for parsing.

4 Adapting boosting for template selection for parsing

Although the algorithm has a theoretical guarantee, certain aspects must be reviewed to adapt it to template selection for parsing. Here we shall review four of these aspects, which prove essential, both in terms of providing a correctly functioning implementation of boosted selection and in terms of the time required for selection: (i) local training during selection, (ii) template grouping for weak parsers, (iii) a user-defined stopping criterion, and (iv) pre-ranking of weak parsers to reduce the pool size at each iteration.

[1] A weak learner (or *weak classifier*, *weak hypothesis*) is a classifier that performs better than random classification.

4.1 Local training of weak parsers

Ensemble methods such as boosting are notoriously problematic when it comes to structured prediction problems such as parsing that require globally optimised training (Cortes et al., 2014). Following Wang et al. (2007), who achieve good results on English and Chinese by boosting locally optimised parsers, we decide to train weak parsers locally during selection. However, unlike Wang et al. (2007), we perform boosting for the unique aim of selecting templates, rather than using the model fitted during boosting. In order to subsequently use the selected templates for parsing, we take the resulting template set, once selection is complete, and fit a globally optimised model. We make the assumption that locally boosted weak parsers provide a good approximation of a template set that can be used to produce a high-performing global model.[2]

4.2 Template grouping for weak parsers

In our approach, the smallest manipulable units for selecting parse models are templates. Instead of considering that each weak parser is trained on a single template, we choose to group templates in order to train larger and stronger weak parsers. The pool of weak parsers is therefore represented by the different template groups available for selection.

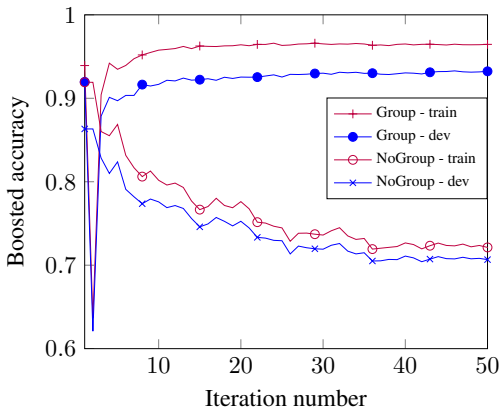


Figure 2: Boosted accuracy with and without grouping

Basis for grouping We use the conditions contained by templates (as defined in Section 2.2) as the basis for grouping. We group templates that share conditions, such that each group contains a single tri-conditional template as well as all the templates with a combination of the three conditions. Each group therefore contains seven templates. Note that uni- and bi-conditional templates necessarily belong to more than one group.

Why group? Grouping templates has several advantages: it allows for cases of co-prediction, it can accelerate model selection by reducing the number of weak parsers trained, and it strengthens the weak parsers. Although in theory handling individual templates would allow for a finer-grained selection, in practice, boosted accuracy does not increase as expected. It has previously been noted by Mukherjee and Schapire (2011) that the guarantee appears to be too weak when the learners perform only just above random. This can be seen in Figure 2, which shows the boosted accuracy for training and development sets with and without grouping. The scores of the resulting models are also significantly lower for individual rather than for grouped templates, and selection is more likely to terminate prematurely due to a lack of sufficiently strong weak parsers, effectively stunting the final model size.

4.3 Stopping criterion

The boosting algorithm has a last-resort stopping criterion, when there are no more weak parsers left. However this is not ideal for a parsing model; time efficiency at parse time is important and, depending on the number of weak parsers available, a model selected with this stopping criterion could be huge, and therefore slow at test time. We observe that the F-score of the retrained model continues to increase (albeit very gradually as the model size increases) as more and more templates are added (tested up to 200 templates), despite the fact that the weighted error during boosting appears to stabilise. We propose a practical stopping criterion, by which the maximum model size (the maximum number of templates) is defined in advance by the user, serving as a second stopping criterion for selection.

4.4 Limiting the pool size by pre-ranking

The size of the weak parser pool is a potential source of problems in terms of efficiency, given the possibility of a very large number of weak parsers. We therefore limit the pool size, by selecting the

[2] Although boosted accuracy is not in strictly perfect correlation with the accuracy of a retrained global model.

parsers from a pool of size k at each iteration, according to a certain criterion. The aim of model selection being to optimise the final performance of the parsing model, we introduce a pre-ranking step, in which all possible weak parsers are globally trained and ranked according to their accuracy. Although this is an approximation of the search space, pre-ranking should be able to guide the search. Weak parsers are not replaced in the pool once selected, which means that the pool is not limited to the original k best-ranked learners; each time a learner is selected, a new one becomes available.[3]

5 Experiments

The aim of our experiments is to investigate the properties of the boosting algorithm in terms of selection time and performance to test the feasibility of the selection method for high-dimensional search spaces. Our working language is French, for which we use a corpus with rich morphological annotations.[4]

5.1 Data

We conduct our experiments for French, using the French TreeBank (FTB) (Abeillé et al., 2003), from the 2014 SPMRL shared task (Seddah et al., 2014), but with automatic predictions of morphological tags[5] by MarMoT (Mueller et al., 2013), trained with ten-fold jackknifing. Templates can refer to the following morphological attributes: word-form, PoS tag, gender, number, mood and multi-word expression tags, as well as the syntactic categories of stack elements. They can access the top two stack elements and the next three queue elements, making a total of 36,050 possible templates and 34,220 possible weak parsers.[6] The corpus is divided into three sets: train, dev and test. Training data is used for pre-ranking and selection, and development data is used for tuning and comparing parameter values. Results on the test set are reported in Section 5.5. For pre-ranking, we use variants of the training set, which vary depending on the maximum length of the sentences they contain. We refer to these variants as FTB-10, FTB-20, FTB-50, where FTB-X contains all sentences from the original with a maximum sentence length of X. To avoid any confusion, we refer to the full set as FTB-all. Data characteristics can be found in Table 1.

Set	#Sents.	#Tokens	Avg. Len.	#(a, C)
FTB-20-train	4,903	63,239	12.90	184,811
FTB-50-train	13,006	330,711	25.43	979,127
FTB-all-train	14,759	443,113	30.02	1,314,580
FTB-all-dev	1235	38,820	31.43	115,225
FTB-all-test	2541	75,216	29.60	223,107

Table 1: Basic data characteristics

5.2 Setup

As described in Section 4, our weak parsers use template groups selected without replacement. During the pre-ranking step, different FTB-X-train variants are evaluated. Weak parsers are trained globally using a single perceptron iteration, a beam size of 4 and early updates (Collins and Roark, 2004). During selection, we also use a single perceptron iteration, but training is done locally. We evaluate the performance of selected models by retraining them globally using FTB-all data, with 30 perceptron iterations, a beam of size 8 and early updates. All experiments were run on an Intel(R) Xeon(R) CPU E5645 @ 2.40GHz. Training is multi-processed but all times reported are approximated for a single processor to enable a rough comparison.

[3] Replacement of learners is common in boosting, especially if they are decision stumps. However we find that with replacement, selection is slower and larger models are unobtainable.

[4] Although not amongst the most morphologically rich languages, the French data contains sufficiently rich morphological annotations to result in a huge number of possible templates, enabling us to test our selection method.

[5] In a realistic scenario where taggers are applied to raw texts before parsing, better parsing results are obtained when training is done on predicted rather than gold tags.

[6] The huge number of templates is due to the large number of morphological attributes available.

5.3 Parameter choices

We investigate the effect on performance and efficiency of selection of varying two parameters: (i) the data used to pre-rank the weak parsers and (ii) the size k of the pool during selection. We also compare against a non-boosted version in which weak learners are simply added in pre-ranking order, which is equivalent to greedy forward selection.

Data	# Sents.	Pre-ranking Time (hours)
FTB-20-train	4,903	63
FTB-50-train	13,006	496
FTB-all-train	14,759	743

Table 2: Pre-ranking times using different data types.

Selection method	Time (mins./iter.)
Pre-rank order	0
Boost, $k = 50$	18.53
Boost, $k = 100$	33.67

Table 3: Selection time per iteration.

Pre-ranking is the most time-consuming step in the process, and the speed is largely determined by training data size (see Table 2). The number of weak parsers is identical for all FTB data, yet there is a stark difference in pre-rank times; FTB-20-train is more than ten times faster than FTB-all-train. Average selection times per iteration, (see Table 3),[7] are directly correlated with the size of the pool k, but are fast compared to the pre-ranking step.

In Figure 3 we study the effect on model performance of varying pre-ranking data and the value of k for two types of model: a small model with a maximum size of 36 templates (for efficient parsing) and a larger model with a maximum size of 120 templates (for higher accuracy but slower parsing).[8]

For the 36-template model, all boosting runs outperform the selection method by pre-ranking only. The effect is greatest when the smaller datasets are used. The crossing pattern seen in the graph for $k = 50$ and $k = 100$ indicates that when the smaller dataset is used for pre-ranking, the larger value for k (of 100) is a better choice, and conversely, when the full dataset is used, the smaller value (of 50) is better. It is therefore possible to achieve almost comparable F-scores as when using full data for pre-ranking and a pool size of 50 by using a smaller dataset (and drastically reducing the time needed to pre-rank) and a larger, less constraining pool size of 100.

F-scores for the larger, 120-template models are higher, but for all methods we observed that the F-score starts to converge at such a large model size. Again, scores are superior when FTB-all-train data is used, but this time, the best model is reached using the simple pre-ranking order of template groups, as long as FTB-all-train is used to pre-rank. For these larger models, it appears that there is less need for the selection process, since a sufficiently large number of reasonably strong templates is all that is required. The disadvantage with this method is that all data is needed for pre-ranking, which, as we have shown above, is more time-consuming. However, as with the smaller models, boosting enables a high-performing model to be selected much faster, by using smaller pre-ranking data.

5.4 Topline model: Naive forward wrapper

We also implemented a naive forward wrapper (see Section 3.1), which we refer to as FWRAP. This method serves as a topline for parsing performance, as the method is a more accurate exploration of the model space, and, as it is very time-consuming, a baseline for selection time. As with the boosting method, templates are added by groups. We change the setup slightly to make the method feasible, using a smaller and less annotated dataset to reduce the number of weak parsers and the time taken to train them. Selection is performed using a variant of FTB-20-train, with the selection criterion being an increased accuracy on FTB-20-dev. We forbid templates from addressing morphological values other than the word-form, the PoS tag and a smoothed version of the word-form,[9] although we allow access to the third stack element and the fourth queue element. Importantly, the total number of weak parsers is far fewer than with the FTB data used above (9,880 vs. 34,220), which means that the initial setup is more

[7] We take the median selection time rather than the mean due to variation due to sporadic differences in machine usage.

[8] For all boosting methods for each parameter combination, an average is calculated over 5 runs to account for random variation due to the use of weighted resampling for training.

[9] Where words of fewer than two occurrences are replaced by a symbol $\$unknown\$$.

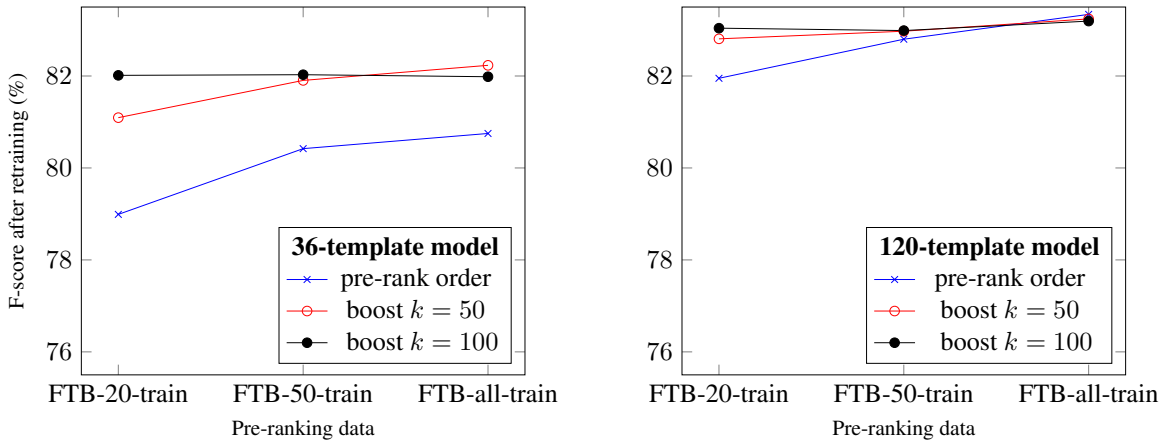


Figure 3: Average F-scores (over 5 separate runs) of selected models with a maximum size of 36 templates (left) and 120 templates (right) for different FTB pre-ranking data and different pool sizes (k).

favourable for FWRAP. Models are trained globally with a beam size of 4 and max-violation updates (Huang et al., 2012) for added speed. Since model sizes vary, the maximum number of perceptron iterations is 35, but we use a test for convergence, enabling training to stop early for smaller models.

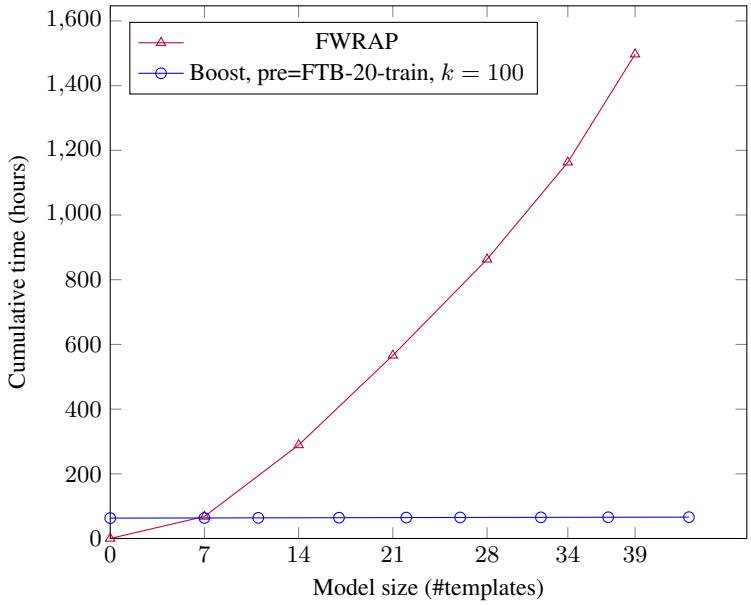


Figure 4: Comparison of cumulative selection time (including pre-ranking) in hours (approximated for a single processor) for FWRAP and a boosted method.

Figure 4 shows selection times for FWRAP and the boosting method (FTB-20-train, $k = 100$). Training an ever-growing model during selection for FWRAP results in an increasing selection time per iteration, whereas the boosting method's selection time is constant and considerably faster overall.

5.5 Results

In Table 4, we provide results for the final selected models on both the development and test sets, evaluated using `evalb`. We provide comparative scores of the Berkeley parser (Petrov et al., 2006), of the current highest performing single parser on French SPMRL data (Durrett and Klein, 2015), of the manually chosen model in (Crabbé, 2014) and of the model selected by FWRAP (Section 5.4). As before, we give our results for two model sizes, using the stopping criterion mentioned in Section 4.3 (a maximum size of 36 templates and of 120 templates).

Model	#tpls.	F-score (%) on FTB-all dev (P)	dev (NoP)	test (P)	test (NoP)	Select. time (hrs)	Parse speed (#Toks./Sec.)
Berkeley (Petrov et al., 2006)	-	79.74	-	80.38	80.73	-	-
Neural CRF (Durrett and Klein, 2015)	-	80.65	-	81.25	-	-	-
Manual (Crabbé, 2014)	68	-	81.79	-	81.43	-	-
FWRAP (Section 5.4)	34	81.80	83.53	**81.52**	83.43	1,163	3,646
FTB-all-train/Pre-rank order	35	79.13	80.75	78.50	80.36	743	3,607
FTB-all-train/$k = 50$	36	**80.73**	**82.42**	79.91	81.69	745	3,478
FTB-20-train/$k = 100$	36	80.58	82.11	**80.34**	**82.23**	**66**	3,426
FTB-all-train/Pre-rank order	117	81.63	83.34	81.14	82.97	743	1,347
FTB-all-train/$k = 100$	119	**81.84**	**83.39**	**81.26**	**83.08**	762	1,328
FTB-20-train/$k = 100$	119	81.59	83.29	80.98	82.71	**80**	1,295

Table 4: Comparison of models. Results are given for evaluation with punctuation (P) and without punctuation (NoP). Selection times include pre-ranking. All times are approximated for a single processor.

The best-performing model is selected using FWRAP, with a higher F-score than the current state-of-the-art single parser (Durrett and Klein, 2015).[10] However the selection time of 1,163 hours means that the method is not very tractable. Our final boosting-based results show the same general pattern as obtained on the development set in the previous section, showing that the approach is robust to a change in dataset, and also to a scaling-up of model training to full optimisation. As in Section 5, the best compromise is the boosted model with pre-ranking on FTB-20-train and $k = 100$. It results in one of the highest scores for the compact model, and the overall selection time is greatly reduced. Importantly, it also outperforms the state-of-the-art manual model (Crabbé, 2014), whilst being almost twice as compact. The results show that for larger models, selection via boosting is less useful, with comparable results between boosted and the "pre-rank only" model. If a large model is needed, simply sequentially adding individually trained weak parsers based on their accuracies can produce a high-performing model.

6 Conclusion

We have successfully performed efficient model selection by using stepwise additive fitting. Experiments on high-dimensional data for French show that compact, state-of-the-art parse models can be achieved, and confirm our hypothesis that locally trained parsers can provide a good approximation for globally trained models. Unlike current template selection methods for parsing (Nilsson and Nugues, 2010; Ballesteros and Nivre, 2014), we use no hand-written heuristic constraints to limit the search space, instead opting for pre-ranking of weak parsers to guide the search. Although pre-ranking is relatively time-consuming, the times are very low compared to standard selection methods. We provide a realistic selection scenario, which involves using only a portion of the training data for pre-ranking, capable of selecting a large model (120 templates) in a couple of hours (when multi-processed), as well as compact models that outperform state-of-the-art manually defined models. A release of all source code is available online at `https://bitbucket.org/rbawden/hyparse-boost-feature-selection`.

In future works, we will extend the approach to a variety of other languages, in particular morphologically rich languages (e.g. Arabic, Hungarian or Korean) and extend the possible templates used to take into account further features. Another interesting perspective would be to study the combination of our approach with constraining heuristics, although language-specific and manual rules would be required.

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a Treebank for French. In *Treebanks*, chapter 10, pages 165–188. Kluwer, Dordrecht.

[10]Note that using parser ensembles and reranking may still lead to higher scores at the expense of parse time efficiency (Björkelund et al., 2014).

Giuseppe Attardi, Felice Dell'Orletta, Maria Simi, Atanas Chanev, and Massimiliano Ciaramita. 2007. Multilingual Dependency Parsing and Domain Adaptation using DeSR. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP–CoNLL '07)*, pages 1112–1118, Prague, Czech Republic.

Miguel Ballesteros and Bernd Bohnet. 2014. Automatic Feature Selection for Agenda-Based Dependency Parsing. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING '14)*, pages 794–805, Dublin, Ireland.

Miguel Ballesteros and Joakim Nivre. 2014. MaltOptimizer: Fast and effective parser optimization. *Natural Language Engineering*, pages 1–27.

Anders Björkelund, Özlem Çetinoğlu, Agnieszka Faleńska, Richárd Farkas, Thomas Müller, Wolfgang Seeker, and Zsolt Szántó. 2014. The IMS-Wrocław-Szeged-CIS Entry at the SPMRL 2014 Shared Task: Reranking and Morphosyntax Meet Unlabeled Data. In *Proceedings of the 1st Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages (SPMRL–SANCL '14)*, pages 97–102, Dublin, Ireland.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL '04)*, pages 111–118, Barcelona, Spain.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '02)*, Philadelphia, Pennsylvania, USA.

Corinna Cortes, Mehryar Mohri, and Umar Syed. 2014. Deep Boosting. In *Proceedings of the 31st International Conference on Machine Learning (ICML '14)*, volume 32, pages 1179–1187, Beijing, China.

Benoît Crabbé. 2014. An LR-inspired generalized lexicalized phrase structure parser. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING '14)*, pages 541–552, Dublin, Ireland.

Benoît Crabbé. 2015. Multilingual discriminative lexicalized phrase structure parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP '15)*, pages 1847–1856, Lisbon, Portugal.

Sanmay Das. 2001. Filters, Wrappers and a Boosting-Based Hybrid for Feature Selection. *Proceedings of the 18th International Conference on Machine Learning (ICML '01)*, pages 74–81.

Greg Durrett and Dan Klein. 2015. Neural CRF Parsing. In *Proceedings of the Association for Computational Linguistics (ACL '15)*, Beijing, China.

Yoav Freund and Robert E. Schapire. 1999. A Brief Introduction to Boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI '99)*, volume 2, pages 1401–1406, Stockholm, Sweden.

He He, Hal Daumé III, and Jason Eisner. 2013. Dynamic Feature Selection for Dependency Parsing. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*, pages 1455–1464, Seattle, Washington, USA.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL '12)*, pages 142–151, Montreal, Canada.

Thomas Mueller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient Higher-Order CRFs for Morphological Tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*, pages 322–332, Seattle, Washington, USA.

Indraneel Mukherjee and Robert E. Schapire. 2011. A Theory of Multiclass Boosting. *Journal of Machine Learning Research*, 14(1):437–497.

Peter Nilsson and Pierre Nugues. 2010. Automatic Discovery of Feature Sets for Dependency Parsing. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING '10)*, pages 824–832, Beijing, China.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING–ACL '06)*, pages 433–440, Sydney, Australia.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 Shared Task on Parsing Morphologically-Rich Languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages (SPMRL–SANCL '14)*, pages 103–109, Dublin, Ireland.

Paul Viola and Michael J. Jones. 2004. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137–154.

Qin Iris Wang, Dekang Lin, and Dale Schuurmans. 2007. Simple training of dependency parsers via structured boosting. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '07)*, pages 1756–1762, Hyderabad, India.

Yue Zhang and Joakim Nivre. 2012. Analyzing the effect of global learning and beam-search on transition-based dependency parsing. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING '12)*, pages 1391–1400, Mumbai, India.

Ji Zhu, Ann Arbor, and Trevor Hastie. 2006. Multi-class AdaBoost. Technical report, Stanford University.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and Accurate Shift-Reduce Constituent Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL '13)*, pages 434–443, Sofia, Bulgaria.

A Universal Framework for Inductive Transfer Parsing across Multi-typed Treebanks

Jiang Guo[♪], **Wanxiang Che**[♪], **Haifeng Wang**[♩] and **Ting Liu**[♪]
[♪]Center for Social Computing and Information Retrieval, Harbin Institute of Technology, China
[♩]Baidu Inc., China
{jguo, car, tliu}@ir.hit.edu.cn
wanghaifeng@baidu.com

Abstract

Various treebanks have been released for dependency parsing. Despite that treebanks may belong to different languages or have different annotation schemes, they contain common syntactic knowledge that is potential to benefit each other. This paper presents a universal framework for transfer parsing across multi-typed treebanks with deep multi-task learning. We consider two kinds of treebanks as source: the *multilingual universal treebanks* and the *monolingual heterogeneous treebanks*. Knowledge across the source and target treebanks are effectively transferred through multi-level parameter sharing. Experiments on several benchmark datasets in various languages demonstrate that our approach can make effective use of arbitrary source treebanks to improve target parsing models.

1 Introduction

As a long-standing central problem in natural language processing (NLP), dependency parsing has been dominated by data-driven approaches for decades. The foundation of data-driven parsing is the availability and scale of annotated training data (i.e., *treebanks*). Numerous efforts have been made towards the construction of treebanks which established the benchmark research on dependency parsing, such as the widely-used Penn Treebank (Marcus et al., 1993). However, the heavy cost of treebanking typically limits the existing treebanks in both scale and coverage of languages.

To address the problem, a variety of authors have proposed to exploit existing heterogeneous treebanks with different annotation schemes via grammar conversion (Niu et al., 2009), quasi-synchronous grammar features (Li et al., 2012) or shared feature representations (Johansson, 2013) for the enhancement of parsing models. Despite their effectiveness in specific datasets, these methods typically lack the scalability of exploiting richer source treebanks, such as cross-lingual treebanks.

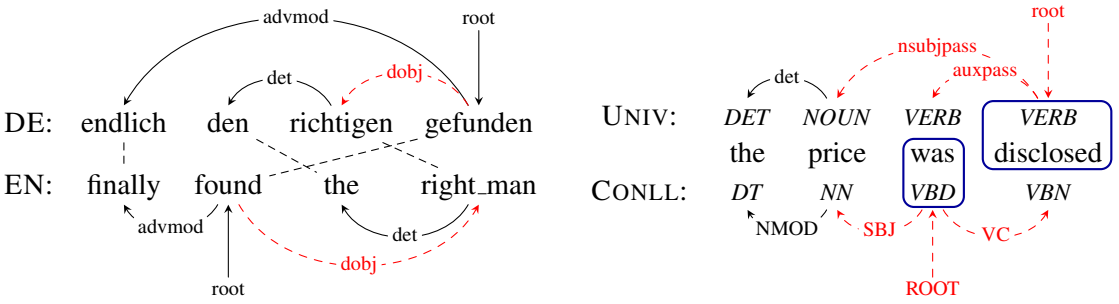
In this paper, we aim at developing a universal framework for transfer parsing that can exploit multi-typed source treebanks to improve parsing of a target treebank. Specifically, we will consider two kinds of source treebanks, that are *multilingual universal treebanks* and *monolingual heterogeneous treebanks*.

Cross-lingual supervision has proven highly beneficial for parsing low-resource languages (Hwa et al., 2005; McDonald et al., 2011), implying that different languages have a great deal of common ground in grammars. But unfortunately, linguistic inconsistencies also exist in both typologies and lexical representations across languages. Figure 1(a) illustrates two sentences in German and English with universal dependency annotations. The typological differences (*subject-verb-object* order) results in the opposite directions of the *dobj* arcs, while the rest arcs remain consistent.

Similar problems also come with monolingual heterogeneous treebanks. Figure 1(b) shows an English sentence annotated with respectively the universal dependencies which are *content-head* and the CoNLL dependencies which instead take the functional heads. Despite the structural divergences, these treebanks express the syntax of the same language, thereby sharing a large amount of common knowledge that can be effectively transferred.

† Corresponding author: Wanxiang Che

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 12–22, Osaka, Japan, December 11-17 2016.



(a) Multilingual universal dependencies.

(b) Monolingual heterogeneous dependencies.

Figure 1: Comparisons between multilingual universal dependencies (a) and monolingual heterogeneous dependencies (b).

The present paper proposes a simple yet effective framework that aims at making full use of the consistencies while avoiding suffering from the inconsistencies across treebanks. Our framework effectively ties together the deep neural parsing models with multi-task learning, using multi-level parameter sharing to control the information flow across tasks. More specifically, learning with each treebank is maintained as an individual task, and their interactions are achieved through parameter sharing in different abstraction levels on the deep neural network, thus referred to as *deep multi-task learning*. We find that different parameter sharing strategies should be applied for different typed source treebanks adaptively, due to the different types of consistencies and inconsistencies (Figure 1).

We investigate the effect of multilingual transfer parsing using the Universal Dependency Treebanks (UDT) (McDonald et al., 2013). We show that our approach improves significantly over strong supervised baseline systems in six languages. We further study the effect of monolingual heterogeneous transfer parsing using UDT and the CoNLL-X shared task dataset (Buchholz and Marsi, 2006). We consider using UDT and CoNLL-X as source treebanks respectively, to investigate their mutual benefits. Experiment results show significant improvements under both settings. Moreover, indirect comparisons on the Chinese Penn Treebank 5.1 (CTB5) using the Chinese Dependency Treebank (CDT)[1] as source treebank show the merits of our approach over previous work.[2]

2 Related Work

The present work is related to several strands of previous studies.

Monolingual resources for parsing Exploiting heterogeneous treebanks for parsing has been explored in various ways. Niu et al. (2009) automatically convert the dependency-structure CDT into the phrase-structure style of CTB5 using a trained constituency parser on CTB5, and then combine the converted treebanks for constituency parsing. Li et al. (2012) capture the annotation inconsistencies among different treebanks by designing several types of *transformation patterns*, based on which they introduce *quasi-synchronous grammar* features (Smith and Eisner, 2009) to augment the baseline parsing models. Johansson (2013) also adopts the idea of parameter sharing to incorporate multiple treebanks. They focuse on parameter sharing at feature-level with discrete representations, which limits its scalability to multilingual treebanks where feature surfaces might be totally different. On the contrary, our approach is capable of utilizing representation-level parameter sharing, making full use of the multi-level abstractive representations generated by deep neural network. This is the key that makes our framework scalable to multi-typed treebanks and thus more practically useful.

Aside from resource utilization, attempts have also been made to integrate different parsing models through stacking (Torres Martins et al., 2008; Nivre and McDonald, 2008) or joint inference (Zhang and Clark, 2008; Zhang et al., 2014).

[1] https://catalog.ldc.upenn.edu/LDC2012T05

[2] Our code is available at: https://github.com/jiangfeng1124/mtl-nndep.

Multilingual resources for parsing Cross-lingual transfer has proven to be a promising way of inducing parsers for low-resource languages, either through *data transfer* (Hwa et al., 2005; Tiedemann, 2014; Rasooli and Collins, 2015) or *model transfer* (McDonald et al., 2011; Täckström et al., 2012; Guo et al., 2015; Zhang and Barzilay, 2015; Guo et al., 2016).

Duong et al. (2015b) and Ammar et al. (2016) both adopt parameter sharing to exploit multilingual treebanks in parsing, but with a few important differences to our work. In both of their models, most of the neural network parameters are shared in two (or multiple) parsers except the feature embeddings,[3] which ignores the important *syntactical inconsistencies* of different languages and is also inapplicable for heterogeneous treebanks that have different transition actions. Besides, Duong et al. (2015b) focus on low resource parsing where the target language has a small treebank of ~3K tokens. Their models may sacrifice accuracy on target languages with a large treebank. Ammar et al. (2016) and Vilares et al. (2016) instead train a single parser on a multilingual set of rich-resource treebanks, which is a more similar setting to ours. We refer to their approach as *shallow multi-task learning* (SMTL) and will include as one of our baseline systems (Section 4.2). Note that SMTL is a special case of our approach in which all tasks use the same set of parameters.

Bilingual parallel data has also proven beneficial in various ways (Chen et al., 2010; Huang et al., 2009; Burkett and Klein, 2008), demonstrating the potential of cross-lingual transfer learning.

Multi-task learning for NLP There has been a line of research on joint modeling pipelined NLP tasks, such as word segmentation, POS tagging and parsing (Hatori et al., 2012; Li et al., 2011; Bohnet and Nivre, 2012). Most multi-task learning or joint training frameworks can be summarized as parameter sharing approaches proposed by Ando and Zhang (2005). In the context of neural models for NLP, the most notable work was proposed by Collobert and Weston (2008), which aims at solving multiple NLP tasks within one framework by sharing common word embeddings. Henderson et al. (2013) present a joint dependency parsing and semantic role labeling model with the Incremental Sigmoid Belief Networks (ISBN) (Henderson and Titov, 2010). More recently, the idea of neural multi-task learning was applied to sequence-to-sequence problems with recurrent neural networks. Dong et al. (2015) use multiple decoders in neural machine translation systems that allows translating one source language to many target languages. Luong et al. (2015) study the ensemble of a wide range of tasks (e.g., syntactic parsing, machine translation, image caption, etc.) with multi-task sequence-to-sequence models.

To the best of our knowledge, we present the first work that successfully integrate both monolingual and multilingual treebanks for parsing, with or without consistent annotation schemes.

3 Approach

This section describes the deep multi-task learning architecture, using a formalism that extends on the transition-based dependency parsing model with LSTM networks (Dyer et al., 2015) which is further enhanced by modeling characters (Ballesteros et al., 2015). We first revisit the parsing approach of Ballesteros et al. (2015), then present our framework for learning with multi-typed source treebanks.

3.1 Transition-based Neural Parsing

Neural models for parsing have gained a lot of interests in recent years, particularly boosted by Chen and Manning (2014). The heart of transition-based parsing is the challenge of representing the *state* (configuration) of a transition system, based on which the most likely transition action is determined. Typically, a state includes three primary components, a *stack*, a *buffer* and a set of *dependency arcs*. Traditional parsing models deal with features extracted from manually defined feature templates in a discrete feature space, which suffers from the problems of *Sparsity*, *Incompleteness* and *Expensive feature computation*. The neural network model proposed by Chen and Manning (2014) instead represents features as continuous, low-dimensional vectors and use a *cube* activation function for implicit feature composition. More recently, this architecture has been improved in several different ways (Dyer et al., 2015; Weiss et al., 2015; Zhou et al., 2015; Andor et al., 2016). Here, we employ the LSTM-based architecture enhanced with character bidirectional LSTMs (Ballesteros et al., 2015) for the following major reasons:

[3] Duong et al. (2015b) used $L2$ regularizers to tie the lexical embeddings with a bilingual dictionary.

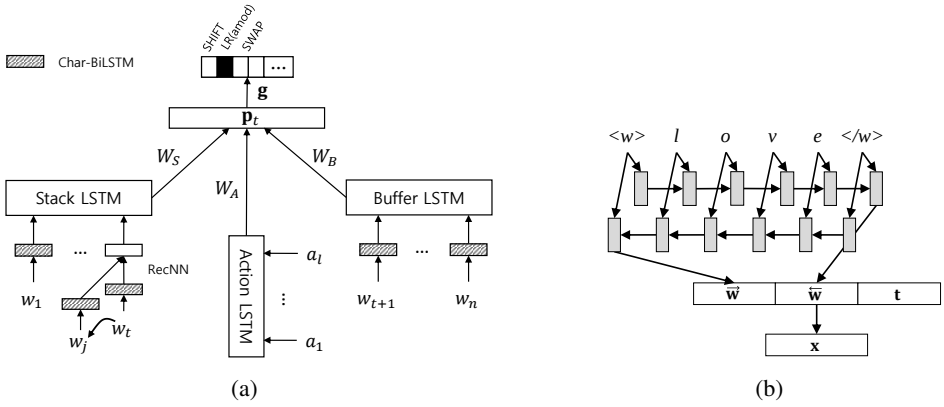


Figure 2: The LSTM-based neural parser (a) and the Char-BiLSTM for modeling words (b).

- Compared with Chen & Manning's architecture, it makes full use of the non-local features by modeling the full history information of a *state* with stack LSTMs.
- By modeling words, stack, buffer and action sequence separately which indicate hierarchical abstractions of representations, we can control the information flow across tasks via parameter sharing with more flexibility (Section 3.2).

Besides, we did not use the earlier ISBN parsing model (Titov and Henderson, 2007) due to its lack of scalability to large vocabulary. Figure 2(a) illustrates the transition-based parsing architecture using LSTMs. Bidirectional LSTMs are used for modeling the word representations (Figure 2(b)), which we refer to as Char-BiLSTMs henceforth. Char-BiLSTMs learn features for each word, and then the representation of each token can be calculated as:

$$\mathbf{x} = \texttt{ReLU}(\mathbf{V}[\overrightarrow{\mathbf{w}}; \overleftarrow{\mathbf{w}}; \mathbf{t}] + \mathbf{b}) \tag{1}$$

where $\mathbf{t}$ is the POS tag embedding. The token embeddings are then fed into subsequent LSTM layers to obtain representations of the *stack*, *buffer* and *action sequence* respectively referred to as $\mathbf{s}_t, \mathbf{b}_t$ and $\mathbf{a}_t$ (The subscript t represents the time step). Note that the subtrees within the stack and buffer are modeled with a *recursive neural network* (RecNN) as described in Dyer et al. (2015). Next, a linear mapping ($\mathbf{W}$) is applied to the concatenation of $\mathbf{s}_t, \mathbf{b}_t$ and $\mathbf{a}_t$, and passed through a component-wise ReLU:

$$\mathbf{p}_t = \texttt{ReLU}(\mathbf{W}[\mathbf{s}_t; \mathbf{b}_t; \mathbf{a}_t] + \mathbf{d}) \tag{2}$$

Finally, the probability of next action $z \in \mathcal{A}(S, B)$ is estimated using a softmax function:

$$p(z|\mathbf{p}_t) = \frac{\exp(\mathbf{g}_z^\top \mathbf{p}_t + \mathbf{q}_z)}{\Sigma_{z' \in \mathcal{A}(S,B)} \exp(\mathbf{g}_{z'}^\top \mathbf{p}_t + \mathbf{q}_{z'})} \tag{3}$$

where $\mathcal{A}(S, B)$ represents the set of valid actions given the current content in the *stack* and *buffer*.

We apply the non-projective transition system originally introduced by Nivre (2009) since most of the treebanks we consider in this study has a noticeable proportion of non-projective trees. In the SWAP-based system, both the *stack* and *buffer* may contain tree fragments, so RecNN is applied both in S and B to obtain representations of each position.

3.2 Deep Multi-task Learning

Multi-task learning (MTL) is the procedure of inductive transfer that improves learning for one task by using the information contained in the training signals of other related tasks. It does this by learning tasks in parallel while using a shared representation. A good overview, especially focusing on neural networks, can be found in Caruana (1997).

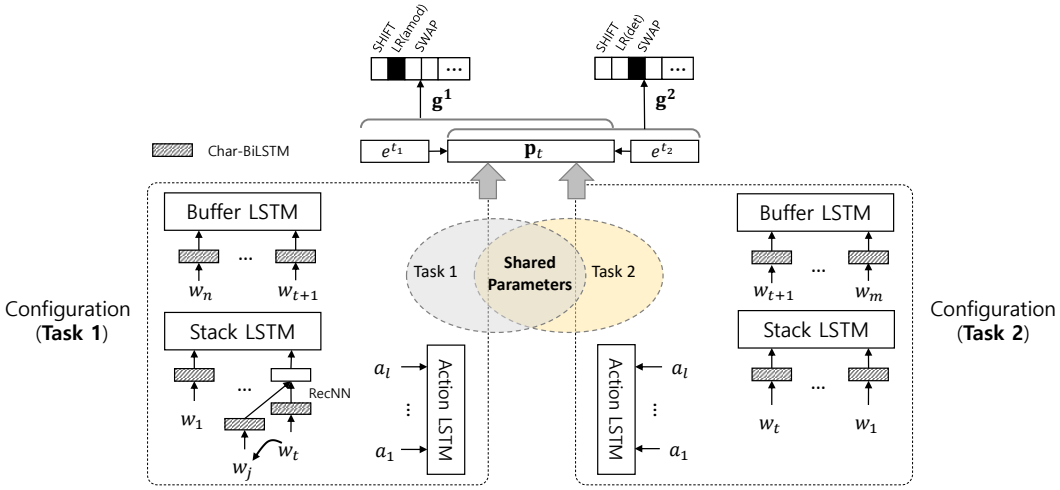


Figure 3: Illustration of the MTL framework.

We illustrate our multi-task learning architecture in Figure 3. As discussed in previous sections, multiple treebanks, either multilingual or monolingual heterogeneous, contain knowledge that can be mutually beneficial. We consider the target treebank processing as the *primary task*, and the source treebank as a *related task*. The two tasks are interacted through multi-level parameter sharing (Section 3.2.1). Inspired by Ammar et al. (2016), we introduce a task-specific vector e^t (*task embedding*) which is first combined with $\mathbf{s}_t, \mathbf{b}_t, \mathbf{a}_t$ to compute $\mathbf{p}_t$, and then further concatenated with $\mathbf{p}_t$ to compute the probability distribution of transition actions. Therefore, Eqn 2, 3 become:

$$\mathbf{p}_t = \texttt{ReLU}(\mathbf{W}[\mathbf{s}_t; \mathbf{b}_t; \mathbf{a}_t; e^t] + \mathbf{d}) \tag{4}$$

$$p(z|\mathbf{p}_t) = \texttt{softmax}(\mathbf{g}_z^\top[\mathbf{p}_t; e^t] + \mathbf{q}_z) \tag{5}$$

The joint cross-entropy is used as the objective function. The key of *multi-task* learning is *parameter sharing*, without which the correlation between tasks will not be exploited. In this work, we design sophisticated parameter sharing strategies according to the linguistic similarities and differences between the tasks.

3.2.1 Parameter Sharing

Deep neural networks automatically learn features for a specific task with hierarchical abstractions, which gives us the flexibility to control parameter sharing in different levels accordingly.

In this study, different parameter sharing strategies are applied according to the source and target treebanks being used. We consider two different scenarios: MTL with multilingual universal treebanks as source (**MULTI-UNIV**) and MTL with monolingual heterogeneous treebanks as source (**MONO-HETERO**). Table 1 presents our parameter sharing strategies for each setting.

MULTI-UNIV Multilingual universal treebanks are annotated with the same set of POS tags (Petrov et al., 2012), dependency relations, and share the same set of transition actions. However, the vocabularies (word, characters) are language-specific. Therefore, it makes sense to share the lookup tables (embeddings) of POS tags (E_{pos}), relations (E_{rel}) and actions (E_{act}), but separate the character embeddings (E_{char}) as well as the Char-BiLSTMs (BiLSTM(chars)). Additionally, linguistic typologies such as the order of *subject-verb-object* and *adjective-noun* (Figure 1(a)) varies across languages, which result in the divergence of inherent grammars of transition action sequences. So we set the action LSTM (LSTM(A)) as *task-specific*.

MONO-HETERO Monolingual heterogeneous treebanks instead share the same lexical representations, but have different POS tags, structures and relations due to the different annotation schemes. Hence the transition actions set varies across treebanks. For simplicity reasons, we convert the language-specific

	MULTI-UNIV	MONO-HETERO
Shared	$E_{pos}, E_{rel}, E_{act}$ LSTM(S), LSTM(B), RecNN W_A, W_S, W_B	E_{pos}, E_{char} LSTM(S), LSTM(B), BiLSTM(chars), RecNN W_A, W_S, W_B
Task-specific	E_{char}, e^t LSTM(A), BiLSTM(chars) $\mathbf{g}$	E_{rel}, E_{act}, e^t LSTM(A) $\mathbf{g}$

Table 1: Parameter sharing strategies for **MULTI-UNIV** and **MONO-HETERO**. LSTM(S) – *stack* LSTM; LSTM(B) – *buffer* LSTM; LSTM(A) – *action* LSTM; BiLSTM(chars) – Char-BiLSTM; RecNN – recursive NN modeling the subtrees; W_A, W_S, W_B – weights from A, S, B to the state ($\mathbf{p}_t$); $\mathbf{g}$ – weights from the state to output layer; E – embeddings.

POS tags of the heterogeneous treebanks into universal POS tags (Petrov et al., 2012). Consequently, E_{char} and BiLSTM(chars), E_{pos} are shared across tasks, but E_{rel}, E_{act}, LSTM(A) are *task-specific*.

Besides, the LSTM parameters for modeling the *stack* and *buffer* (LSTM(S), LSTM(B)), the RecNN for modeling tree compositions, and the weights from S, B, A to the state $\mathbf{p}_t$ (W_A, W_B, W_S) are shared for both MULTI-UNIV and MONO-HETERO. As standard in multi-task learning, the weights at the output layer ($\mathbf{g}$) are *task-specific* in both settings.

3.2.2 Learning

Training is achieved in a stochastic manner by looping over the tasks:

1. Randomly select a task.
2. Select a sentence from the task, and generate instances for classification.
3. Update the corresponding parameters by back-propagation w.r.t. the instances.
4. Go to 1.

We adopt the development data of the target treebank (primary task) for early-stopping.

4 Experiments

We first describe the data and settings in our experiments, then the results and analysis.

4.1 Data and Settings

We conduct experiments on UDT v2.0[4] and the CoNLL-X shared task data. For monolingual heterogeneous source, we also experiment on CTB5 using CDT as the source treebank, to compare with the previous work of Li et al. (2012). Statistics of the datasets are summarized in Table 2. We investigate the following experiment settings:

- MULTILINGUAL (UNIV→UNIV). In this setting, we study the integration of multilingual universal treebanks. Specifically, we consider the DE, ES, FR, PT, IT and SV universal treebanks as target treebanks, and the EN treebank as the common source treebank.
- MONOLINGUAL (CONLL↔UNIV). Here we study the integration of monolingual heterogeneous treebanks. The CoNLL-X corporas (DE, ES, PT, SV) and the UDT treebank of corresponding languages are used as source and target treebanks mutually.
- MONOLINGUAL (CDT→CTB5). We follow the same settings of Li et al. (2012), and consider two scenarios using automatic POS tags and gold-standard POS tags respectively.

[4] https://github.com/ryanmcd/uni-dep-tb

	Train	Dev	Test	Train	Dev	Test
	UDT			CoNLL-X		
EN	39,832	1,700	2,416	–	–	–
DE	14,118	800	1,000	35,295	3,921	357
ES	14,138	1,569	300	2,976	330	206
FR	14,511	1,611	300	–	–	–
PT	9,600	1,200	1,198	8,164	907	288
IT	6,389	400	400	–	–	–
SV	4,447	493	1,219	9,938	1,104	389
	CDT			CTB5		
ZH	55,500	1,500	3,000	16,091	803	1,910

Table 2: Statics of UDT v2.0 and CoNLL-X treebanks (with languages presented in UDT v2.0).

	MULTILINGUAL (UNIV → UNIV)							
	SUP		CAS$_{EN}$		SMTL$_{EN}$		MTL$_{EN}$	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
DE	84.24	78.40	84.24	78.65	84.37	79.07	**84.93**	**79.34**
ES	85.31	81.23	85.42	81.42	85.78	81.54	**86.78**	**82.92**
FR	85.55	81.13	84.57	80.14	86.13	81.77	**86.44**	**82.01**
PT	88.40	86.54	88.88	87.07	89.08	87.24	**89.24**	**87.50**
IT	86.53	83.72	86.58	83.67	86.53	83.64	**87.26**	**84.27**
SV	84.91	79.88	86.43	81.92	**86.79**	**82.31**	85.98	81.35
AVG	*85.82*	*81.82*	*86.02*	*82.15*	*86.45*	*82.60*	***86.77***	***82.90***

Table 3: Parsing accuracies of MULTILINGUAL (UNIV→UNIV). Significance tests with MaltEval yield p-values < 0.01 for (MTL *vs.* SUP) on all languages.

4.2 Baseline Systems

We compare our approach with the following baseline systems.

- Monolingual supervised training (SUP). Models are trained only on the target treebank, with the LSTM-based parser.
- Cascaded training (CAS). This system has two stages. First, models are trained using the source treebank. Then the parameters are used to initialize the neural network for training target parsers. Similar approach was studied in Duong et al. (2015a) and Guo et al. (2016) for low-resource parsing.

For MULTILINGUAL (UNIV→UNIV), we also compare with the *shallow multi-task learning* (SMTL) system, as described in Section 2, which is representative of the approach of Duong et al. (2015b) and Ammar et al. (2016). In SMTL all the parameters are shared except the character embeddings (E_{char}), and task embeddings (e^t) are not used. Unlike Duong et al. (2015b) and Ammar et al. (2016), we don't use external resources such as cross-lingual word clusters, embeddings and dictionaries which is beyond the scope of this work.

4.3 Results

In this section, we present empirical evaluations under different settings.

4.3.1 Multilingual Universal Source Treebanks

Table 3 shows the results under the MULTILINGUAL (UNIV→UNIV) setting. CAS yields slightly better performance than SUP, especially for SV (+1.52% UAS and +2.04% LAS), indicating that *pre-training* with EN training data indeed provides a better initialization of the parameters for cascaded training. SMTL in turn outperforms CAS overall (comparable for IT), which implies that training two treebanks jointly helps even with a unique model.

	DE	ES	FR
SUP	58.93	61.99	60.45
CAS	64.08	**70.45**	**68.72**
SMTL	63.57	69.01	65.04
+ *weighted sampling*	*63.50*	*70.17*	*68.52*
MTL	62.43	66.67	64.23
+ *weighted sampling*	***64.22***	*68.42*	*66.67*
Duong et al.	61.2	69.1	65.3
Duong et al. + Dict	61.8	70.5	67.2

Table 4: Low resource setup (3K tokens), evaluated with LAS.

Furthermore, with appropriate parameter sharing, our deep multi-task learning approach (MTL) outperforms SUP overall and achieves the best performances in five out of six languages. An exception is Swedish. As we can see, both CAS and SMTL outperforms MTL by a significant margin for SV. The underlying reasons we suggest are two-fold.

1. SV morphology is similar to EN with less inflections, encouraging the morphology-related parameters like BiLSTM(chars) to be shared.
2. SV has a much smaller treebank compared with EN (1:9). We suggest that SMTL and CAS work better than MTL in low resource setting. To our intuition, since knowledge contained in the low-resource target treebank is very limited, it is reasonable for us to put more emphasis on the source treebank through SMTL or CAS.

To verify the first issue, we conduct tests on SMTL without sharing Char-BiLSTMs, and observe significant degradation in performance (-0.73 in UAS and -0.81 in LAS). This observation also suggests that MTL has the potential to reach higher performances through *language-specific tuning* of parameter sharing strategies.

To verify the second issue, we consider a low resource setup following Duong et al. (2015b), where the target language has a small treebank (3K tokens). We train our models on identical sampled dataset shared by the authors on DE, ES and FR. As we can find in Table 4, while all the models outperform SUP, both CAS and SMTL work better than MTL, which confirms our assumption. Although not the primary focus of this work, we find that SMTL and MTL can be significantly improved in low resource setting through weighted sampling of tasks during training. Specifically, in the training procedure (Section 3.2.2), we sample from the source language (EN) which has a much richer treebank with larger probability of 0.9, while sample from the target language with probability of 0.1. In this way, the two tasks are encouraged to converge at a similar rate. As shown in Table 4, both SMTL and MTL benefit from weighted task sampling.

4.3.2 Monolingual Heterogeneous Source Treebanks

Among the four languages here, the SV universal treebank is mainly converted from the Talbanken part of the Swedish bank (Nivre and Megyesi, 2007), thus has a large overlap with the CONLL-X Swedish treebank. Therefore, we exclude the sentences in SV test set that appear in the source treebank for evaluation. Table 5 shows the results of MONOLINGUAL (CONLL↔UNIV). Overall MTL systems outperforms the supervised baselines by significant margins in both conditions, showing the mutual benefits of UDT and CONLL-X treebanks.[5]

To show the merit of our approach against previous approaches, we further conduct experiments on CTB5 using CDT as heterogeneous source treebank (Table 2). For CTB5, we follow (Li et al., 2012) and consider two scenarios which use automatic POS tags and gold-standard POS tags respectively. To compare with their results, we run SUP, CAS and MTL on CTB5. Table 6 presents the results.

[5] An exception is PT in MONOLINGUAL (UNIV→CONLL). This may be due to the low quality of the PT universal treebank caused by the automatic construction process. We discussed and verified this with the author of UDT v2.0.

	MONOLINGUAL (CONLL→UNIV)						MONOLINGUAL (UNIV→CONLL)					
	SUP		CAS		MTL		SUP		CAS		MTL	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
DE	84.24	78.40	85.02	80.05	**85.73**	**80.64**	89.06	86.48	89.64	86.66	**89.98**	**87.50**
ES	85.31	81.23	**85.90**	**81.73**	85.80	81.45	85.41	80.50	**86.46**	81.37	86.07	**81.41**
PT	88.40	86.54	89.12	87.32	**89.40**	**87.60**	**90.16**	**85.53**	89.50	85.03	89.98	85.23
SV	82.61	77.42	**85.39**	80.60	85.29	**81.22**	79.61	72.71	82.91	74.96	**84.86**	**77.36**
AVG	*85.14*	*80.90*	*86.35*	*82.43*	***86.56***	***82.73***	*86.06*	*81.31*	*87.13*	*82.01*	***87.72***	***82.88***

Table 5: MONOLINGUAL (CONLL↔UNIV) performance.

		Auto-POS			Gold-POS		
		SUP	CAS	MTL	SUP	CAS	MTL
OURS	UAS	79.34	80.25 (+0.91)	**81.13 (+1.79)**	85.25	86.29 (+1.04)	**86.69 (+1.44)**
	LAS	76.23	77.26 (+1.03)	**78.24 (+2.01)**	83.59	84.72 (+1.13)	**85.18 (+1.59)**
		SUP	with QG		SUP	with QG	
LI12-O2	UAS	79.67	81.04 (+1.37)		86.13	86.44 (+0.31)	
LI12-O2SIB		79.25	80.45 (+1.20)		85.63	86.17 (+0.54)	

Table 6: Parsing accuracy comparisons of MONOLINGUAL (CDT→CTB5). LI12-O2 use the O2 graph-based parser with both sibling and grandparent structures, while LI12-O2SIB only use the sibling parts.

The indirect comparison indicates that our approach can achieve larger improvement than their method in both scenarios. Beside the empirical comparison, our method has the additional advantages in its scalability to multi-typed source treebanks without the painful human efforts of feature design.

5 Conclusion

This paper propose a universal framework based on deep multi-task learning that can integrate arbitrary-typed source treebanks to enhance the parsing models on target treebanks. We study two scenarios, respectively using multilingual universal source treebanks and monolingual heterogeneous source treebanks, and design effective parameter sharing strategies for each scenario.

We conduct extensive experiments on benchmark treebanks in various languages. Results demonstrate that our approach significantly improves over baseline systems under various experiment settings.

Acknowledgments

We thank Ryan McDonald for helpful discussions, and thank Dr. Zhenghua Li for sharing the processed CTB and CDT dataset. We also thank the anonymous reviewers for their insightful comments and suggestions. This work was supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grant 61300113 and 61370164.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. One parser, many languages. *TACL*, 4:431–444.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853, December.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. of the 54th ACL*, pages 2442–2452, Berlin, Germany, August.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proc. of EMNLP*, pages 349–359, September.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proc. of EMNLP-CoNLL*, pages 1455–1465, July.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proc. of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, June.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proc. of the 2008 Conference on EMNLP*, pages 877–886, October.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of EMNLP*, pages 740–750, October.

Wenliang Chen, Jun'ichi Kazama, and Kentaro Torisawa. 2010. Bitext dependency parsing with bilingual subtree constraints. In *Proc. of the 48th ACL*, pages 21–29, July.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of the 25th ICML*, pages 160–167.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proc. of the 53rd ACL and the 7th IJCNLP (Volume 1: Long Papers)*, pages 1723–1732, July.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015a. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proc. of ACL-IJCNLP*, pages 845–850, July.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015b. A neural network model for low-resource universal dependency parsing. In *Proc. of the 2015 Conference on EMNLP*, pages 339–348, September.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL-IJCNLP*, pages 334–343, July.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proc. of ACL-IJCNLP*, pages 1234–1244, July.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *Proc. of AAAI*, February.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *Proc. of ACL*, pages 1045–1053, July.

James Henderson and Ivan Titov. 2010. Incremental sigmoid belief networks for grammar learning. *J. Mach. Learn. Res.*, 11:3541–3570, December.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics*, 39(4):949–998.

Liang Huang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proc. of the 2009 Conference on EMNLP*, pages 1222–1231, August.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(03):311–325.

Richard Johansson. 2013. Training parsers on incompatible treebanks. In *Proc. of NAACL*, pages 127–137, June.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for chinese pos tagging and dependency parsing. In *Proc. of EMNLP*, pages 1180–1191, July.

Zhenghua Li, Ting Liu, and Wanxiang Che. 2012. Exploiting multiple treebanks for parsing with quasi-synchronous grammars. In *Proc. of the 50th ACL (Volume 1: Long Papers)*, pages 675–684, July.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *CoRR*, abs/1511.06114.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proc. of the 2011 Conference on EMNLP*, pages 62–72, July.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proc. of ACL*, pages 92–97, August.

Zheng-Yu Niu, Haifeng Wang, and Hua Wu. 2009. Exploiting heterogeneous treebanks for parsing. In *Proc. of the Joint Conference of the 47th ACL and the 4th IJCNLP of the AFNLP*, pages 46–54, August.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proc. of ACL-08: HLT*, pages 950–958, June.

Joakim Nivre and Beata Megyesi. 2007. Bootstrapping a swedish treebank using cross-corpus harmonization and annotation projection. In *Proc. of the 6th International Workshop on Treebanks and Linguistic Theories*, pages 97–102.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proc. of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 351–359, August.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proc. of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 2089–2096, May.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proc. of EMNLP*, pages 328–338, September.

David A. Smith and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proc. of the 2009 Conference on EMNLP*, pages 822–831, August.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proc. of NAACL: HLT*, pages 477–487, June.

Jörg Tiedemann. 2014. Rediscovering annotation projection for cross-lingual parser induction. In *Proc. of COLING 2014*, pages 1854–1864, August.

Ivan Titov and James Henderson. 2007. Fast and robust multilingual dependency parsing with a generative latent variable model. In *Proc. of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 947–951, June.

André Filipe Torres Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proc. of the 2008 Conference on EMNLP*, pages 157–166, October.

David Vilares, Carlos Gómez-Rodríguez, and Miguel A. Alonso. 2016. One model, two languages: training bilingual parsers with harmonized treebanks. In *Proc. of the 54th ACL*, pages 425–431, August.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proc. of ACL-IJCNLP*, pages 323–333, July.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proc. of the 2015 Conference on EMNLP*, pages 1857–1867, September.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proc. of the 2008 Conference on EMNLP*, pages 562–571, October.

Meishan Zhang, Wanxiang Che, Yanqiu Shao, and Ting Liu. 2014. Jointly or separately: Which is better for parsing heterogeneous dependencies? In *Proc. of COLING 2014*, pages 530–540, August.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proc. of ACL-IJCNLP*, pages 1213–1222, July.

Grammar induction from (lots of) words alone

John K Pate
Department of Linguistics
The University at Buffalo
Buffalo, NY 14260, USA
johnpate@buffalo.edu

Mark Johnson
Department of Computing
Macquarie University
Sydney, NSW 2109, Australia
mark.johnson@mq.edu.au

Abstract

Grammar induction is the task of learning syntactic structure in a setting where that structure is hidden. Grammar induction from words alone is interesting because it is similiar to the problem that a child learning a language faces. Previous work has typically assumed richer but cognitively implausible input, such as POS tag annotated data, which makes that work less relevant to human language acquisition. We show that grammar induction from words alone is in fact feasible when the model is provided with sufficient training data, and present two new streaming or mini-batch algorithms for PCFG inference that can learn from millions of words of training data. We compare the performance of these algorithms to a batch algorithm that learns from less data. The minibatch algorithms outperform the batch algorithm, showing that cheap inference with more data is better than intensive inference with less data. Additionally, we show that the harmonic initialiser, which previous work identified as essential when learning from small POS-tag annotated corpora (Klein and Manning, 2004), is not superior to a uniform initialisation.

1 Introduction

How children acquire the syntax of the languages they ultimately speak is a deep scientific question of fundamental importance to linguistics and cognitive science (Chomsky, 1986). The natural language processing task of grammar induction in principle should provide models for how children do this. However, previous work on grammar induction has learned from small datasets, and has dealt with the resulting data sparsity by modifying the input and using careful search heuristics. While these techniques are useful from an engineering perspective, they make the models less relevant to human language acquisition.

In this paper, we use scalable algorithms for Probabilistic Context Free Grammar (PCFG) inference to perform grammar induction from millions of words of speech transcripts, and show that grammar induction from words alone is both feasible and insensitive to initialization. To ensure the robustness of our results, we use two algorithms for Variational Bayesian PCFG inference, and adapt two algorithms that have been proposed for Latent Dirichlet Allocation (LDA) topic models. Most importantly, we find that the three algorithms that scale to large datasets improve steadily over training to about the same predictive probability and parsing performance.

Moreover, while grammar induction from small datasets of POS-tagged newswire text fails without careful 'harmonic' initialization, we find that initialization is much less important when learning directly from larger datasets consisting of words alone. Of the algorithms in this paper, one does 2.5% better with harmonic initialization, another does 5% worse, and the other two are insensitive to initialization.

The rest of the paper is organized as follows. In Section 2, we discuss previous grammar induction research, in Section 3 we present the particular model grammar we will use, in Section 4 we describe the inference algorithms, and in Section 5 we present our experimental results.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 23–32, Osaka, Japan, December 11-17 2016.

2 Background

Previous grammar induction work has used datasets with at most $50,000$ sentences. Fully-lexicalized models would struggle with data sparsity on such small datasets, so previous work has assumed input in either the form of part-of-speech (POS) tags (Klein and Manning, 2004; Headden III et al., 2009) or word representations trained on a large external corpus (Spitkovsky et al., 2011; Le and Zuidema, 2015).

Some previous work has moved towards learning from word strings directly. Bisk and Hockenmaier (2013) used combinatory categorial grammar (CCG) to learn syntactic dependencies from word strings. However, they initialise their model by annotating nouns, verbs, and conjunctions in the training set with atomic CCG categories using a dictionary, and so do not learn from words alone. Pate and Goldwater (2013) learned syntactic dependencies from word strings alone, but used sentences from the Switchboard corpus of telephone speech that had been selected for prosodic annotation and so were unusually fluent.

Kim and Mooney (2010), Börschinger et al. (2011), and Kwiatkowski et al. (2012), learned from word strings together with logical form representations of sentence meanings. While children have situational cues to sentence meaning, these cues are ambiguous, and it is difficult to represent these cues in a way that is not biased towards the actual sentences under consideration. We focus on the evidence for syntactic structure that can be obtained from word strings themselves.

Grammar induction directly from word strings is interesting for two reasons. First, this problem setting more closely matches the language acquisition task faced by an infant, who will not have access to POS tags or a corpus external to her experience. Second, this setting allows us to attribute behavior of grammar induction systems to the underlying model itself, rather than additional annotations made to the input. Approaches to grammar induction that involve replacing words with POS tags or other lexical or syntactic observed labels make the process significantly more difficult to understand or compare across genres or languages, as the results will depend on exactly how these labels are assigned. Models that only require words alone as input do not suffer from this weakness.

3 PCFGs and the Dependency Model with Valence

3.1 Probabilistic Context Free Grammars

A Probabilistic Context Free Grammar is a tuple $(\boldsymbol{W}, \boldsymbol{N}, S, \boldsymbol{R}, \boldsymbol{\theta})$, where $\boldsymbol{W}$ and $\boldsymbol{N}$ are sets of terminal and non-terminal symbols, $S \in \boldsymbol{N}$ is a distinguished start symbol, and $\boldsymbol{R}$ is a set of production rules. $\boldsymbol{\theta}$ is a vector of multinomial parameters of length $|\boldsymbol{R}|$ indexed by production rules $A \rightarrow \beta$, so $\boldsymbol{\theta}_{A\rightarrow\beta}$ is the probability of the production $A \rightarrow \beta$. We use $\boldsymbol{R}_A$ to denote all rules with left-hand side A, and use $\boldsymbol{\theta}_A$ to denote the subvector of $\boldsymbol{\theta}$ indexed by the rules in $\boldsymbol{R}_A$. We require for all rules, $\theta_{A\rightarrow\beta} \geq 0$, and for all $A \in \boldsymbol{N}$, $\sum_{A\rightarrow\beta\in\boldsymbol{R}_A} \theta_{A\rightarrow\beta} = 1$, and use Δ to denote the probability simplex satisfying these constraints. The *yield* $\mathrm{y}(t)$ of a tree t is the string of terminals of t, and the yield of a vector of trees $\boldsymbol{T} = (t_1, \ldots, t_{|\boldsymbol{T}|})$ is the vector of yields of each tree: $\mathrm{y}(\boldsymbol{T}) = (\mathrm{y}(t_1), \ldots, \mathrm{y}(t_{|\boldsymbol{T}|}))$. The probability of generating a tree t given parameters $\boldsymbol{\theta}$ is:

$$P_G(t|\boldsymbol{\theta}) = \prod_{A\rightarrow\beta\in\boldsymbol{R}} \theta_r^{f(t,A\rightarrow\beta)}$$

where $f(t, A \rightarrow \beta)$ is the number of times rule $A \rightarrow \beta$ is used in the derivation of t.

To model uncertainty in the parameters, we draw the parameters of each multinomial $\boldsymbol{\theta}_A$ from a prior distribution $P(\boldsymbol{\theta}_A|\boldsymbol{\alpha}_A)$, where the vector of hyperparameters $\boldsymbol{\alpha}_A$ defines the shape of this prior distribution (and $\boldsymbol{\alpha}$ is just the concatenation of each $\boldsymbol{\alpha}_A$). The joint probability of a vector of trees $\boldsymbol{T}$ with one tree t_i for each sentence s_i, and parameters $\boldsymbol{\theta}$ is then:

$$P(\boldsymbol{T}, \boldsymbol{\theta}|\boldsymbol{\alpha}) = P(\boldsymbol{T}|\boldsymbol{\theta})P(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \prod_{A\rightarrow\beta\in\boldsymbol{R}} \theta_{A\rightarrow\beta}^{f(\boldsymbol{T},A\rightarrow\beta)} \left[\prod_{A\in\boldsymbol{N}} P(\boldsymbol{\theta}_A|\boldsymbol{\alpha}_A)\right]$$

where $f(\boldsymbol{T}, r)$ is the number of times rule r is used in the derivation of the trees in $\boldsymbol{T}$.

Dirichlet priors for these multinomials are both standard and convenient:

$$P_D(\boldsymbol{\theta}_A|\boldsymbol{\alpha}_A) = \frac{\Gamma\left(\sum_{A\to\beta\in\boldsymbol{R}_A}\alpha_{A\to\beta}\right)}{\prod_{A\to\beta\in\boldsymbol{R}_A}\Gamma(\alpha_{A\to\beta})}\prod_{A\to\beta\in\boldsymbol{R}_A}\theta_{A\to\beta}^{\alpha_{A\to\beta}-1}$$

where the Gamma function Γ generalizes the factorial function from integers to real numbers. Dirichlet distributions are convenient priors because they are *conjugate* to multinomial distributions: the product of a Dirichlet distribution and a multinomial distribution is itself a Dirichlet distribution:

$$P(\boldsymbol{T},\boldsymbol{\theta}|\boldsymbol{\alpha}) = P_G(\boldsymbol{T}|\boldsymbol{\theta})P_D(\boldsymbol{\theta}|\boldsymbol{\alpha}) \propto \prod_{A\to\beta\in\boldsymbol{R}}\theta_{A\to\beta}^{f(\boldsymbol{T},A\to\beta)+\alpha_{A\to\beta}-1} \tag{1}$$

For grammar induction, we observe only the corpus of sentences $\boldsymbol{C}$, and modify Equation 1 to marginalize over trees and rule probabilities.

$$P(\boldsymbol{C}|\boldsymbol{\alpha}) = \sum_{\boldsymbol{T}:\mathrm{y}(\boldsymbol{T})=\boldsymbol{C}}\int_{\Delta}P(\boldsymbol{T},\boldsymbol{\theta}|\boldsymbol{\alpha})d\boldsymbol{\theta} \tag{2}$$

This sum over trees introduces dependencies that make exact inference intractable.

We assessed grammar induction from words alone using the Dependency Model with Valence (DMV) (Klein and Manning, 2004). In the original presentation, it first draws the root of the sentence from a P_{root} distribution over words, and then generates the dependents of head h in each direction $dir \in \{\leftarrow,\rightarrow\}$ in a recursive two-step process. First, it decides whether to stop generating (a Stop decision) according to $P_{stop}(\cdot|h, dir, v)$, where v indicates whether or not h has any dependents in the direction of dir. If it does not stop (a $\neg$Stop decision), it draws the dependent word d from $P_{choose}(d|h, dir)$. Generation ceases when all words stop in both directions.

Johnson (2007) and Headden III et al. (2009) reformulated this generative process as a *split-head bilexical PCFG* (Eisner and Satta, 2001) so that the rule probabilities are DMV parameters. Such a PCFG represents each token of the string with two 'directed' terminals that handle leftward and rightward decisions independently, and defines rules and non-terminal symbols schematically in terms of terminals. Minimally, we need rightward-looking R_w, leftward-looking L_w, and undirected Y_w non-terminal labels for each word w. The grammar has a rule for each dependent word d of a head word h from the left ($L_h \rightarrow Y_d\ \ L_h$) and from the right ($R_h \rightarrow R_h\ \ Y_d$), a rule for each a word w to be the sentence root ($S \rightarrow Y_w$), and a rule for each undirected symbol to split into directed symbols ($Y_w \rightarrow L_w\ \ R_w$).

To incorporate Stop decisions into the grammar, we distinguished non-terminals that dominate a Stop decision from those that dominate a Choose decision by decorating Choose non-terminals with $'$ (so a left attachment rule is $L'_h \rightarrow Y_d\ \ L_h$), and introduced unary rules that rewrite to terminals ($L_h \rightarrow h_l$) for Stop decisions, and to Choose non-terminals ($L_h \rightarrow L'_h$) for $\neg$Stop decisions. We implemented valence with a superscript decoration on each non-terminal label: L_h^0 indicates h has no dependents to the left, and L_h indicates that h has at least one dependent to the left. Figure 1 presents PCFG rule schemas with their DMV parameters, and dependency and split-head PCFG trees for "dogs bark." We use several inference algorithms to learn production weights for this PCFG, and study how the parsing accuracy varies with algorithm and computational effort.

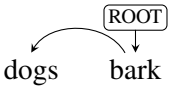
4 Inference[1]

One central challenge of learning from words alone is data sparsity. Data sparsity is most naturally addressed by learning from large amounts of data, which is easily available when learning from words alone, so we use algorithms that scale to large datasets. To ensure that our system reflects the underlying relationship between the model and the data, we explore three such algorithms. These algorithms are extensions of the batch algorithm for variational Bayesian (batch VB) inference of PCFGs due to Kurihara

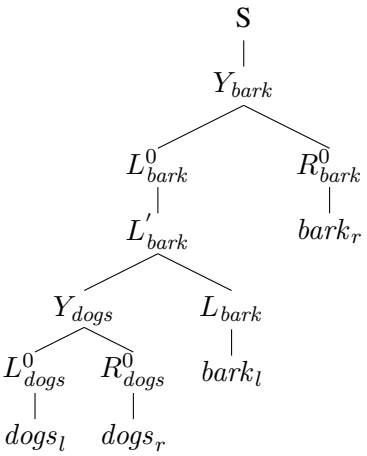
[1]Implementations and pre-processing software are available at `http://github.com/jkpate/streamingDMV`

PCFG Rule	DMV parameter
$S \rightarrow Y_h$	$P_{root}(h)$
$Y_h \rightarrow L_h^0 \ R_h^0$	1
$L_h^0 \rightarrow h_l$	$P_{stop}(\text{Stop}\|h, \leftarrow, \text{no\_dep})$
$L_h^0 \rightarrow L_h^{'}$	$P_{stop}(\neg\text{Stop}\|h, \leftarrow, \text{no\_dep})$
$L_h^{'} \rightarrow Y_d \ L_h$	$P_{choose}(d\|h, \leftarrow)$
$L_h \rightarrow h_l$	$P_{stop}(\text{Stop}\|h, \leftarrow, \text{one\_dep})$
$L_h \rightarrow L_h^{'}$	$P_{stop}(\neg\text{Stop}\|h, \leftarrow, \text{one\_dep})$

(a) Split-head rule schemas and corresponding probabilities for the DMV. The rules expanding L_h^0 and L_h symbols encode Stop decisions with no dependents and at least one dependent, respectively, and the the rules expanding L_h' symbols encode Choose decisions.



(c) Example dependency tree with one root and left arc.



(b) Tree for "dogs bark" using the grammar in Figure 1a.

Figure 1: The DMV as a PCFG, and dependency and split-head bilexical CFG trees for "dogs bark."

and Sato (2004), so we first review batch VB. Inspired by the reduction of LDA inference to PCFG inference presented in Johnson (2010), we then develop new streaming on-line PCFG inference algorithms by generalising the streaming VB (Broderick et al., 2013) and stochastic VB (Hoffman et al., 2010) algorithms for Latent Dirichlet Allocation (LDA) to PCFG inference. We finally review the collapsed VB algorithm due to Wang and Blunsom (2013) for PCFGs that we compare to the other algorithms.

Figure 2 summarizes the four algorithms for PCFG inference.

4.1 Batch VB

Kurihara and Sato's (2004) batch algorithm for variational Bayesian inference approximates the posterior $P(\boldsymbol{T}, \boldsymbol{\theta}|\boldsymbol{C}, \boldsymbol{\alpha})$ by maximizing a lower bound on the log marginal likelihood of the observations $\ln P(\boldsymbol{C}|\boldsymbol{\alpha})$. This lower bound $\mathcal{L}$ involves a variational distribution $Q(\boldsymbol{T}, \boldsymbol{\theta})$ over unobserved variables $\boldsymbol{T}$ and $\boldsymbol{\theta}$. By Jensen's inequality, for any distribution $Q(\boldsymbol{T}, \boldsymbol{\theta})$, we have:

$$\ln P(\boldsymbol{C}|\boldsymbol{\alpha}) = \ln \sum_{\boldsymbol{T}} \int Q(\boldsymbol{T}, \boldsymbol{\theta}) \frac{P(\boldsymbol{C}, \boldsymbol{T}, \boldsymbol{\theta}|\boldsymbol{\alpha})}{Q(\boldsymbol{T}, \boldsymbol{\theta})} d\boldsymbol{\theta} \geq \sum_{\boldsymbol{T}} \int Q(\boldsymbol{T}, \boldsymbol{\theta}) \ln \frac{P(\boldsymbol{C}, \boldsymbol{T}, \boldsymbol{\theta}|\boldsymbol{\alpha})}{Q(\boldsymbol{T}, \boldsymbol{\theta})} d\boldsymbol{\theta} = \mathcal{L}$$

$\ln P(\boldsymbol{C}|\boldsymbol{\alpha}) - \mathcal{L}$ is the Kullback-Leibler divergence $\text{KL}\left(Q(\boldsymbol{T}, \boldsymbol{\theta}) || P(\boldsymbol{T}, \boldsymbol{\theta}|\boldsymbol{C}, \boldsymbol{\alpha})\right)$. Variational inference adjusts the parameters of the variational distribution to maximize $\mathcal{L}$, which minimizes the KL divergence.

VB makes inference tractable by factorizing the variational posterior. The mean-field factorization assumes parameters and trees are independent: $Q(\boldsymbol{T}, \boldsymbol{\theta}) = Q_{\boldsymbol{\theta}}(\boldsymbol{\theta}) \prod_{i=1}^{|\boldsymbol{T}|} Q_{\boldsymbol{T}}(t_i)$. Kurihara and Sato showed that $Q_{\boldsymbol{\theta}}(\boldsymbol{\theta})$ is also a product of Dirichlet distributions, whose hyperparameters $\hat{\boldsymbol{\alpha}}_A$ are a sum of the prior hyperparameters $\alpha_{A \rightarrow \beta}$ and the expected count of $A \rightarrow \beta$ across the corpus under $Q_{\boldsymbol{T}}$:

$$\hat{\alpha}_{A \rightarrow \beta} = \alpha_{A \rightarrow \beta} + \sum_{i=1}^{|\boldsymbol{C}|} \hat{f}(s_i, A \rightarrow \beta)$$

$$\hat{f}(s, A \rightarrow \beta) = \mathbb{E}_{Q_{\boldsymbol{T}}}\left[f(t, A \rightarrow \beta)\right]$$

$f(t, A \rightarrow \beta)$ is the number of times rule $A \rightarrow \beta$ is used in the derivation of tree t. $\hat{f}(s_i, A \rightarrow \beta)$ is the expected number of times $A \rightarrow \beta$ is used in the derivation of sentence s_i, and can be computed using the Inside Outside algorithm (Lari and Young, 1990). Batch VB alternates between optimizing $Q_{\boldsymbol{\theta}}$, using

expected counts, and $Q_{\boldsymbol{T}}$, using the hyperparameters $\hat{\boldsymbol{\alpha}}$ of $Q_{\boldsymbol{\theta}}$ to compute probability-like ratios $\pi_{A\to\beta}$:

$$Q_{\boldsymbol{T}}(t) = \prod_{A\to\beta\in\boldsymbol{R}} \pi_{A\to\beta}^{f(t,A\to\beta)} \qquad \pi_{A\to\beta} = \frac{\exp\left(\Psi\left(\hat{\alpha}_{A\to\beta}\right)\right)}{\exp\left(\Psi\left(\sum_{A\to\beta'\in\boldsymbol{R}_A}\hat{\alpha}_{A\to\beta'}\right)\right)}$$

where the digamma function $\Psi(\cdot)$ is the derivative of the log Gamma function. Algorithm 1 presents the full algorithm.

4.2 Scalable VB algorithms

Batch VB requires a complete parse of the training data before parameter updates, which is computationally intensive. We explore three algorithms that divide the data into minibatches $\boldsymbol{C} = \left\{\boldsymbol{C}^{(1)}, \ldots, \boldsymbol{C}^{(n)}\right\}$ and update parameters after parsing each minibatch.

Streaming VB: Broderick et al. (2013) proposed a 'streaming VB' algorithm for LDA that approximates Bayesian Belief Updates (BBU) to make a single pass through the training data. A BBU uses the current posterior as a prior to compute the next posterior without reanalyzing previous minibatches:

$$P\left(\boldsymbol{\theta}|\boldsymbol{C}^{(1)}, \ldots, \boldsymbol{C}^{(n)}\right) \propto P\left(\boldsymbol{C}^{(n)}|\boldsymbol{\theta}\right) P\left(\boldsymbol{\theta}|\boldsymbol{C}^{(1)}, \ldots, \boldsymbol{C}^{(n-1)}\right)$$

However, the normalization constant involves an intractable marginalization. Broderick et al. suggested approximating each posterior with some algorithm $\mathcal{A}$ that computes an approximate posterior $Q^{(n)}$ given a minibatch $\boldsymbol{C}^{(n)}$ and the previous posterior $Q^{(n-1)}$:

$$P\left(\boldsymbol{\theta}|\boldsymbol{C}^{(1)}, \ldots, \boldsymbol{C}^{(n)}\right) \approx Q^{(n)}(\boldsymbol{\theta}) = \mathcal{A}\left(\boldsymbol{C}^{(n)}, Q^{(n-1)}(\boldsymbol{\theta})\right)$$

where $Q^{(0)}$ is the true prior. By using a mean-field VB algorithm for LDA inference for $\mathcal{A}$, they approximate each subsequent $Q^{(n)}$ as a product of Dirichlets, whose hyperparameters are a running sum of expected counts from previous minibatches and prior hyperparameters. We used the batch VB algorithm for $\mathcal{A}$ to generalise this algorithm to PCFG inference. Algorithm 2 presents the full algorithm.

Stochastic VB: Hoffman et al. (2010) proposed a 'stochastic VB' algorithm for LDA that uses each minibatch to compute the maximum of an estimate of the natural gradient of $\mathcal{L}$. This maximum is obtained by computing expected counts for $\boldsymbol{C}^{(i)}$, and scaling the counts as though they were gathered from the full dataset. The new hyperparameters are obtained by taking a step toward the maximum:

$$\alpha_{A\to\beta}^{(en+i)} = (1-\eta)\,\alpha_{A\to\beta}^{(en+i-1)} + \eta l_{A\to\beta}^{(i)} \hat{f}^{(en+i)}\left(A\to\beta\right)$$

where η is the step size, $\hat{f}^{(en+i)}\left(A\to\beta\right)$ is the expected count of rule $A\to\beta$ in minibatch i of epoch e, and $l_{A\to\beta}^{(i)}$ is the scaling term for rule $A\to\beta$. In their LDA inference procedure, each word has one topic, so the scaling term is the number of words in the full dataset divided by the number of words in the minibatch. For the DMV, a string s with $|s|$ terminals has one root, $|s|-1$ choose rules, and $2|s| + (|s|-1)$ stop decisions (two Stops and one ¬Stop rule for each arc). The scaling terms are then:

root rules	choose rules	stop rules
$l_{S\to Y_h}^{(i)} = \frac{\lvert\boldsymbol{C}\rvert}{\lvert\boldsymbol{C}^{(i)}\rvert}$	$l_{A\to\beta}^{(i)} = \frac{\sum_{j=1}^{\lvert\boldsymbol{C}\rvert}\lvert s_j\rvert-1}{\sum_{j'=1}^{\lvert\boldsymbol{C}^{(i)}\rvert}\lvert s_{j'}\rvert-1}$	$l_{A\to\beta}^{(i)} = \frac{\sum_{j=1}^{\lvert\boldsymbol{C}\rvert} 2\lvert s_j\rvert+\lvert s_j\rvert-1}{\sum_{j'=1}^{\lvert\boldsymbol{C}^{(i)}\rvert} 2\lvert s_{j'}\rvert+\lvert s_{j'}\rvert-1}$

Collapsed VB: Teh et al. (2007) proposed, and Asuncion et al. (2009) simplified, a 'collapsed VB' algorithm for LDA that integrates out model parameters and so achieves a tighter lower bound on the marginal likelihood. This algorithm cycled through the training set and optimized variational distributions over the topic assignment of each word given all the other words.

Wang and Blunsom (2013) generalized this algorithm to PCFGs. The variational distribution for each sentence is parameterized by expected rule counts for that sentence, and they optimize each sentence-specific distribution by cycling through the corpus and optimizing the distribution over trees for sentence s_i using counts from all the other sentences $\boldsymbol{C}^{(\neg i)}$. The exact optimization, marginalizing over rule probabilities, is intractable, so they instead use the posterior mean. Algorithm 4 presents the full algorithm.

Data: a corpus of strings $\boldsymbol{C}$
Initialization: prior hyperparameters $\boldsymbol{\alpha}$

$\hat{\boldsymbol{\alpha}}^{(0)} = \boldsymbol{\alpha}$
for $j = 1$ **to** m **do**
  $\pi_{A\to\beta} = \frac{\exp\left(\Psi\left(\hat{\alpha}_{A\to\beta}^{(j-1)}\right)\right)}{\exp\left(\Psi\left(\sum_{A\to\beta'\in\boldsymbol{R}_A}\hat{\alpha}_{A\to\beta'}^{(j-1)}\right)\right)}$
  for $i = 1$ **to** $|\boldsymbol{C}|$ **do**
    $\hat{f}^{(j)}(s_i, A\to\beta) = \mathbb{E}_{\boldsymbol{\pi}}\left[f(t, A\to\beta)\right]$
  end
  $\hat{\boldsymbol{\alpha}}^{(j)} = \boldsymbol{\alpha} + \hat{\boldsymbol{f}}^{(j)}$
end
output: $\hat{\boldsymbol{\alpha}}^{(m)}$

Algorithm 1: Batch VB. Here, $\hat{\alpha}^{(j)}$ are the posterior counts after iteration j, which define rule weights π for the next iteration.

Data: n minibatches $\left\{\boldsymbol{C}^{(1)}, \ldots, \boldsymbol{C}^{(n)}\right\}$
Initialization: prior hyperparameters $\boldsymbol{\alpha}$, step size schedule parameters τ, κ, epoch count E

$\hat{\boldsymbol{\alpha}}^{(0)} = \boldsymbol{\alpha}$
for $e = 0$ **to** $E-1$ **do**
  for $i = 1$ **to** n **do**
    $\pi_{A\to\beta} = \frac{\exp\left(\Psi\left(\hat{\alpha}_{A\to\beta}^{(en+i-1)}\right)\right)}{\exp\left(\Psi\left(\sum_{A\to\beta'\in\boldsymbol{R}_A}\hat{\alpha}_{A\to\beta'}^{(en+i-1)}\right)\right)}$
    $\hat{f}^{(en+i)}(A\to\beta) = \mathbb{E}_{\boldsymbol{\pi}}\left[f(t, A\to\beta)\right]$
    $\eta = (\tau+i)^{-\kappa}$
    for $A\to\beta \in \boldsymbol{R}$ **do**
      $\hat{\alpha}_{A\to\beta}^{(en+i)} = (1-\eta)\,\hat{\alpha}_{A\to\beta}^{(en+i-1)} + \eta l_{A\to\beta}^{(i)} \hat{f}^{(en+i)}(A\to\beta)$
    end
  end
end
output: $\hat{\boldsymbol{\alpha}}^{(n)}$

Algorithm 3: Stochastic VB. $\hat{f}^{(en+i)}(A\to\beta)$ is the expected count of rule $A\to\beta$ in the i^{th} minibatch in the e^{th} epoch, and $l_{A\to\beta}^{(i)}$ is the scaling parameter for rule $A\to\beta$ for the i^{th} minibatch, as described in the text.

Data: n minibatches $\left\{\boldsymbol{C}^{(1)}, \ldots, \boldsymbol{C}^{(n)}\right\}$
Initialization: prior hyperparameters $\boldsymbol{\alpha}$

$\hat{\boldsymbol{\alpha}}^{(0)} = \boldsymbol{\alpha}$
for $i = 1$ **to** n **do**
  $\forall A\to\beta \in \boldsymbol{R}\ \ \hat{f}^{(0)}(A\to\beta) = 0$
  for $j = 1$ **to** m **do**
    $\pi_{A\to\beta} = \frac{\exp\left(\Psi\left(\hat{f}\left(\boldsymbol{C}^{(i)}, A\to\beta\right)+\hat{\alpha}_{A\to\beta}\right)\right)}{\exp\left(\Psi\left(\sum_{A\to\beta'\in\boldsymbol{R}_A}\hat{f}\left(\boldsymbol{C}^{(i)}, A\to\beta'\right)+\hat{\alpha}_{A\to\beta'}\right)\right)}$
    $\hat{f}^{(i,j)}(A\to\beta) = \mathbb{E}_{\boldsymbol{\pi}}\left[f(t, A\to\beta)\right]$
  end
  $\hat{\boldsymbol{\alpha}}^{(i)} = \hat{\boldsymbol{f}}^{(i,m)} + \hat{\boldsymbol{\alpha}}^{(i-1)}$
end
output: $\hat{\boldsymbol{\alpha}}^{(n)}$

Algorithm 2: Streaming VB with m steps of VB per minibatch. $\hat{f}^{(j)}(A\to\beta)$ is the expected count of rule $A\to\beta$ in the i^{th} minibatch after j iterations.

Data: n single-string minibatches $\left\{\boldsymbol{C}^{(1)}, \ldots, \boldsymbol{C}^{(n)}\right\}$
Initialization: prior hyperparameters $\boldsymbol{\alpha}$, epoch count E, initial sentence-specific expected counts $\hat{\boldsymbol{f}}$

$\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha} + \sum_{i=1}^{n}\hat{\boldsymbol{f}}^{(i)}$
for $e = 0$ **to** $E-1$ **do**
  for $i = 1$ **to** n **do**
    $\hat{\boldsymbol{\alpha}} = \hat{\boldsymbol{\alpha}} - \hat{\boldsymbol{f}}^{(i)}$
    $\pi_{A\to\beta} = \frac{\hat{\alpha}_{A\to\beta}}{\sum_{A\to\beta'\in\boldsymbol{R}_A}\hat{\alpha}_{A\to\beta'}}$
    $\hat{f}^{(i)}(A\to\beta) = \mathbb{E}_{\boldsymbol{\pi}}\left[f(t, A\to\beta)\right]$
    $\hat{\boldsymbol{\alpha}} = \hat{\boldsymbol{\alpha}} + \hat{\boldsymbol{f}}^{(i)}$
  end
end
output: sentence-specific expected counts $\hat{\boldsymbol{f}}$, global hyperparameters $\hat{\boldsymbol{\alpha}}$

Algorithm 4: Collapsed VB. $\hat{f}^{(i)}(A\to\beta)$ is the expected count of rule $A\to\beta$ for the i^{th} sentence, and the global hyperparameters $\hat{\boldsymbol{\alpha}}$ are the sum of the expected counts for each sentence and prior hyperparameters.

Figure 2: The four variational Bayes algorithms for PCFG inference that are evaluated in this paper. Algorithm 1 is from Kurihara and Sato (2004), and Algorithm 4 is from Wang and Blunsom (2013). Algorithms 2 and 3 are novel PCFG inference algorithms developed here that generalise the LDA inference algorithms of Broderick et al. (2013) and Hoffman et al. (2010).

		train	dev	test
Swbd	Words	363,902	24,015	23,872
	Sentences	43,577	2,951	2,956
Fisher	Words	5,576,173	–	–
	Sentences	664,346	–	–

Table 1: Data set sizes. Fisher is only for training.

5 Experiments

We evaluate how millions of words of training data affects grammar induction from words alone by examining learning curves. We ran each algorithm 5 times, each with a different random shuffle of the training data on each run, and evaluated on the test set at logarithmically-spaced numbers of training sentences. Stochastic, collapsed, and batch VB used more than one pass over the training corpus, while streaming VB makes one pass over the training corpus. To obtain a consistent horizontal axis in our learning curves, we plot learning curves as a function of computational effort, which we measure by the number of sentences parsed, since almost all the computational effort of all the algorithms is in parsing.

Stochastic, collapsed, and streaming VB can learn from the full training corpus (although collapsed VB requires more RAM – 60GB rather than 6GB for us – as it stores expected rule counts for each sentence). Batch VB required about 50 iterations for convergence for large training sets, and so cannot be applied to the full training set due to long training times. We used batch VB with training sets of up to $100,000$ strings.

5.1 Hyperparameters and initialization

We use Dirichlet priors with symmetric hyperparameter $\alpha = 1$ for all algorithms (preliminary experiments showed that the algorithms are generally insensitive to hyper-parameter settings). We ran batch VB until the log probability of the training set changed less than 0.001%. For stochastic VB, we used $\kappa = 0.9$, $\tau = 1$, and minibatches of 10,000 sentences. To investigate convergence and overfitting, we ran stochastic and collapsed VB for 15 epochs of random orderings of the training corpus. For streaming VB, the first minibatch had $10,000$ sentences, the rest had 1, and we used one iteration of VB per minibatch.

Klein and Manning (2004) showed that initialization strongly influences the quality of the induced grammar when training from POS-tagged WSJ10 data, and they proposed a harmonic initialization procedure that puts more weight on rules that involve terminals that frequently appear close to each other in the training data. We present results both for a uniform initialization, where the only counts initially are the uninformative Dirichlet priors (plus, for collapsed VB, random sentence-specific counts), and a harmonic initialization. For streaming VB, harmonic counts are gathered from each minibatch, and for the others, harmonic counts are gathered from the entire training set.

5.2 Data

We present experiments on two corpora of words from spontaneous speech transcripts. Our first corpus is drawn from the Switchboard portion of the Penn Treebank (Calhoun et al., 2010; Marcus et al., 1993). We used the version produced by Honnibal and Johnson (2014), who used the Stanford dependency converter to convert the constituency annotations to dependency annotations (Catherine de Marneffe et al., 2006). We used Honnibal and Johnson's train/dev/test partition, and ignored their second dev partition. We discarded sentences shorter than four words from all partitions, as they tend to be formulaic backchannel responses (Bell et al., 2009), and sentences with more than 15 words (long sentences did not improve accuracy on the dev set).

We augmented the Switchboard training set with the Fisher corpus of telephone transcripts. We again used only sentences with 4 to 15 words. Unlike the words-only evaluation of Pate and Goldwater (2013), which used only the fluent sentences from Switchboard that had been prosodically annotated, both of these corpora contain disfluencies. These corpora have a vocabulary of $40,597$ word types. The grammars have one Root rule for each word type, four Stop rules (two directions x two Stop decisions) for

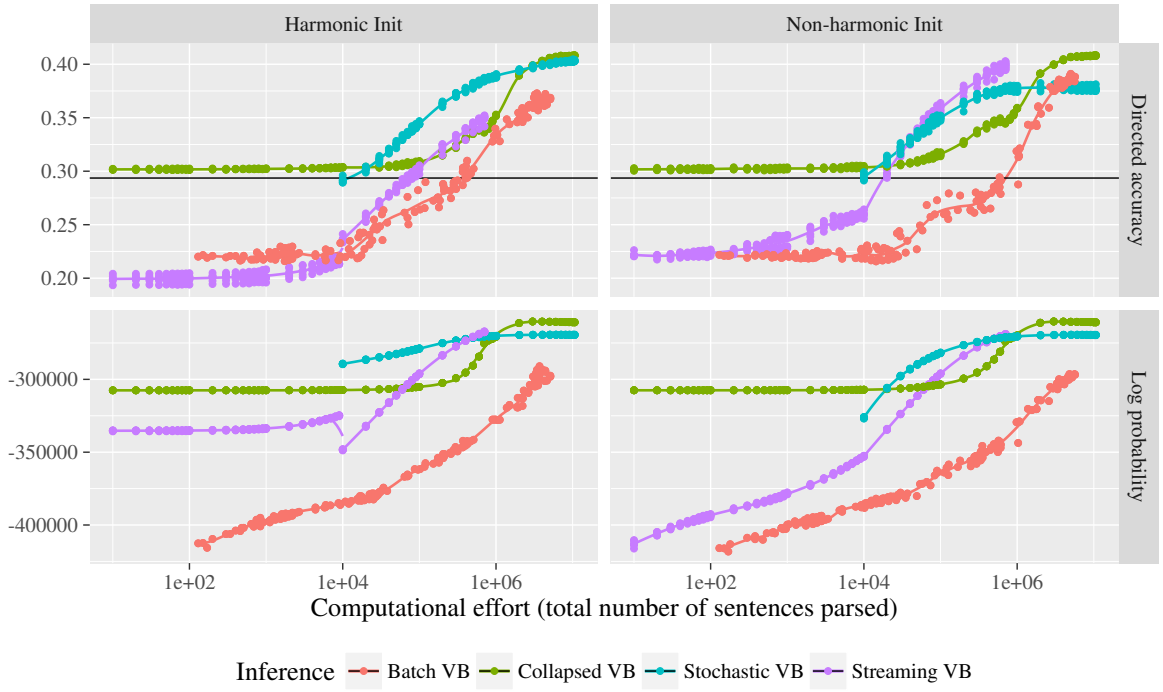


Figure 3: Directed accuracy (top) and predictive log probability (bottom) of test-set sentences from Switchboard with 4-15 words. The horizontal axis is the number of sentences parsed (all algorithms except streaming VB re-parse sentences multiple times). The left column presents inference with a harmonic initialization, and the right column presents inference with a uniform initialization (and, for collapsed VB, random sentence-specific counts). The black line is a uniform-branching baseline.

each word type, and $5,381,644$ Choose rules. Table 1 presents data set sizes.

5.3 Evaluation measures

We evaluated all algorithms in terms of predictive log probability and directed attachment accuracy. We computed the log probability of the evaluation set under posterior mean parameters, obtained by normalizing counts. Directed accuracy is the proportion of arcs in the Viterbi parse that match the gold standard, including the root. We also compared with a left-branching baseline, since it outperformed a right-branching baseline. A left-branching (right-branching) baseline sets the last (first) word of each sentence to be the root, and assigns each word to be the head of the word to its left (right). The left-branching baseline on this dataset is about 0.29, while on the traditional `wsj10` dataset of Klein and Manning (2004) it is 0.336, suggesting our dataset, with longer sentences, is somewhat more difficult.

5.4 Results

The bottom row of Figure 3 presents the predictive log probability of the test set under the posterior mean parameter vector as training proceeds. The figure contains one point per evaluation per run, and a loess-smoothed curve for each inference type. We see among all algorithms that the log probability of the test set constantly increases, regardless of initialization, with one exception. The sole exception is streaming VB with harmonic initialization, where predictive log probability drops after the initial minibatch of 10,000 sentences. Streaming VB with harmonic init parses each sentence of the initial minibatch using prior pseudocounts and harmonic counts. We will return to this drop when we discuss accuracy.

Batch VB learns more slowly (as a function of computational effort) than the online and minibatch algorithms, but the online and minibatch algorithms all ultimately obtain similar performance. Collapsed VB obtains the best predictive log probability, which, as it integrates out parameters and therefore has a tighter bound, is to be expected (Teh et al., 2007). There is no clear advantage to the harmonic initialization except early in training for streaming and stochastic VB, so it may be that earlier results showing

the importance of harmonic initialisation reflect the small training data sets used in those experiments.

The top row of Figure 3 presents directed accuracy on the test set as training proceeds. As in the predictive probability evaluation, there is no clear advantage to a harmonic initialization across algorithms. Batch VB and collapsed VB perform identically with both initializations, and streaming VB ultimately does 5% better while stochastic VB does 2.5% worse. While streaming VB showed a drop in predictive probability after the initial 10,000 sentence minibatch with harmonic initialization, it obtains a small but sharp improvement in parse accuracy at the same point. These two results suggest that the harmonic initialization, applied to words, captures regularities that are not syntactic but still explain data well.

The inference algorithms differ most obviously when they have parsed few sentences, indicating that each algorithm's bias is strongest in the face of small data. Streaming VB learns slowly initially because, throughout the large initial minibatch, it gathers counts using only the uninformative prior or only the uninformative prior plus harmonic counts. Collapsed VB, on the other hand, has sentence-specific counts for the entire training corpus even in the random case. These counts provide a rough estimate of how many opportunities there are for an arc to exist between each word in each direction at each valence, and therefore provide a stronger starting point that takes more time to overcome. Finally, the good performance of stochastic VB with small datasets, compared to streaming VB and batch VB, may reflect the conservatism of only taking a step in the direction of the gradient rather than always maximizing.

Regardless of the details of the different algorithms' performance, we see that they all steadily improve or stabilize as inference proceeds over a large dataset, and that initialization is not important when learning from large numbers of words.

6 Conclusion

Grammar induction from words alone has the potential to address important questions about how children learn and represent linguistic structure, but previous work has struggled to learn from words alone in a principled way. Our experiments show that grammar induction from words alone is feasible with a simple and well-known model if the dataset is large enough, and that heuristic initialization is not necessary (and may even interfere). Future computational work on child language acquisition should take advantage of this finding by applying richer models of syntax to large datasets, and learning distributed word representations jointly with syntactic structure.

References

Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. 2009. On smoothing and inference for topic models. In *Uncertainty in Artificial Intelligence*.

Alan Bell, Jason M Brenier, Michelle Gregory, Cynthia Girand, and Dan Jurafsky. 2009. Predictability effects on durations of content and function words in conversational English. *Journal of Memory and Language*, 60:92–111.

Yonatan Bisk and Julia Hockenmaier. 2013. An HDP model for inducing Combinatory Categorial Grammars. *Transactions of the Association for Computational Linguistics*, 1(Mar):75–88.

Benjamin Börschinger, Bevan K Jones, and Mark Johnson. 2011. Reducing grounded learning tasks to grammatical inference. In *Proceedings of the 2011 conference on Emprical Methods in Natural Language Processing (EMNLP)*, pages 1416–1425.

Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C Wilson, and Michale I Jordan. 2013. Streaming variational bayes. In *Neural Information Processing Systems*.

S Calhoun, J Carletta, J Brenier, N Mayo, D Jurafsky, M Steedman, and D Beaver. 2010. The NXT-format Switchboard corpus: A rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue. *Language Resources and Evaluation*, 44(4):387–419.

Marie Catherine de Marneffe, Bill MacCartney, and Christopher D Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*.

Noam Chomsky. 1986. Knowledge of language: Its nature, origin, and use.

Jason Eisner and Giorgio Satta. 2001. Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *ACL*.

Will Headden III, Mark Johnson, and David McClosky. 2009. Improved unsupervised dependency parsing with richer contexts and smoothing. In *NAACL-HLT*.

M Hoffman, D M Blei, and F Bach. 2010. Online learning for latent dirichlet allocation. In *Proceedings of NIPS*.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association for Computational Linguistics*, 2(April):131–142.

Mark Johnson. 2007. Transforming projective bilexical dependency grammars into efficiently-parseable CFGs with unfold-fold. In *ACL*.

Mark Johnson. 2010. PCFGs, topic models, adaptor grammars and learning topical collocations and the structure of proper names. In *Proceedings of the Association for Computational Linguistics*.

Joohyun Kim and Raymond J Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd international conference on computational linguistics (COLING 2010)*.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL*, pages 479–486.

Kenichi Kurihara and Taisuke Sato. 2004. An application of the Variational Bayesian approach to probabilistic context-free grammars. In *IJCNLP 2004 Workshop beyond Shallow Analyses*.

Tom Kwiatkowski, Sharon Goldwater, Luke Zettlemoyer, and Mark Steedman. 2012. A probabilistic model of syntactic and semantic acquisition from child-directed utterances and their meanings. In *Proceedings of the European Chapter of the Association for Computational Linguistics*.

K Lari and S J Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 5:237–257.

Phong Le and Willem Zuidema. 2015. Unsupervised dependency parsing: Let's use supervised parsers. In *Proceedings of ACL*.

Mitchell P Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

John K Pate and Sharon Goldwater. 2013. Unsupervised dependency parsing with acoustic cues. *Transactions of the ACL*.

Valentin I Spitkovsky, Hiyan Alshawi, Angel X Chang, and Daniel Jurafsky. 2011. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*.

Yee Whye Teh, David Newman, and Max Welling. 2007. A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. In *Neural Information Processing Systems*, volume 19, pages 705–729.

Pengyu Wang and Phil Blunsom. 2013. Collapsed variational bayesian inference for PCFGs. In *Proceedings the Seventeenth Conference on Computational Natural Language Learning*, pages 173–182.

A Redundancy-Aware Sentence Regression Framework for Extractive Summarization

Pengjie Ren[1*] **Furu Wei**[2] **Zhumin Chen**[1†] **Jun Ma**[1] **Ming Zhou**[2]
jay.ren@outlook.com, {chenzhumin,majun}@sdu.edu.cn, {fuwei,mingzhou}@microsoft.com
[1]Department of Computer Science and Technology, Shandong University / Jinan, China
[2]Microsoft Research Asia / Beijing, China

Abstract

Existing sentence regression methods for extractive summarization usually model sentence importance and redundancy in two separate processes. They first evaluate the importance $f(s)$ of each sentence s and then select sentences to generate a summary based on both the importance scores and redundancy among sentences. In this paper, we propose to model importance and redundancy simultaneously by directly evaluating the relative importance $f(s|S)$ of a sentence s given a set of selected sentences S. Specifically, we present a new framework to conduct regression with respect to the relative gain of s given S calculated by the ROUGE metric. Besides the single sentence features, additional features derived from the sentence relations are incorporated. Experiments on the DUC 2001, 2002 and 2004 multi-document summarization datasets show that the proposed method outperforms state-of-the-art extractive summarization approaches.

1 Introduction

Sentence regression is one of the branches of extractive summarization methods that achieves state-of-the-art performances (Cao et al., 2015b; Wan et al., 2015) and is commonly used in practical systems (Hu and Wan, 2013; Wan and Zhang, 2014; Hong and Nenkova, 2014). Existing sentence regression methods usually model sentence importance and sentence redundancy in two separate processes, namely sentence ranking and sentence selection. Specifically, in the sentence ranking process, they evaluate the importance $f(s)$ of each sentence s with a ranking model (Osborne, 2002; Conroy et al., 2004; Galley, 2006; Li et al., 2007) through either directly measuring the salience of sentences (Li et al., 2007; Ouyang et al., 2007) or firstly ranking words (or bi-grams) and then combining these scores to rank sentences (Lin and Hovy, 2000; Yih et al., 2007; Gillick and Favre, 2009; Li et al., 2013). Then, in the sentence selection process, they discard the redundant sentences that are similar to the already selected sentences.

In this paper, we propose a novel regression framework to directly model the relative importance $f(s|S)$ of a sentence s given the sentences S. Specifically, we evaluate the relative importance $f(s|S)$ with a regression model where additional features involving the sentence relations are incorporated. Then we generate the summary by greedily selecting the next sentence which maximizes $f(s|S)$ with respect to the current selected sentences S. Our method improves the existing regression framework from three aspects. First, our method is redundancy-aware by considering importance and redundancy simultaneously instead of two separate processes. Second, we treat the scores computed using the official evaluation tool as the groundtruth and find that our method has a higher upper bound. Third, there is no manually tuned parameters, which is more convenient in practice. We carry out experiments on three benchmark datasets from DUC 2001, 2002, and 2004 multi-document summarization tasks. Experimental results show that our method achieves the best performance in terms of ROUGE-2 recall metric and outperforms state-of-the-art extractive summarization approaches on all three datasets.

∗ This work was done during the internship of the first author at Microsoft Research Asia.
† Corresponding author.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 33–43, Osaka, Japan, December 11-17 2016.

2 Framework

2.1 Background

Formally, given a sentence set (from one or multiple documents) $D \in C$, extractive summarization tries to select a sentence set S^* as the summary that maximizes an utility function $f(S)$ with respective to the length limit l, Existing sentence regression methods usually model the importance of each sentence independently (Osborne, 2002; Galley, 2006; Li et al., 2007). Then, they use a threshold parameter to control the redundancy (Cao et al., 2015b; Galanis et al., 2012) when selecting sentences with the Greedy algorithm or Integer Linear Programming (ILP) algorithm (Cao et al., 2015a). The framework for these regression methods can be formulated as follows.

$$f(s|S) = \begin{cases} f(s) & 1 - SIM(s,S) \geq t \\ 0 & 1 - SIM(s,S) < t \end{cases} \tag{1}$$

where S is the set of already selected sentences, $f(s)$ models the importance of sentence s. $SIM(s,S)$ evaluates the similarity of sentence s with the current generated summary S. Usually, $SIM(s,S) = \frac{bi\text{-}gram\text{-}overlap(s,S)}{Len(s)}$, which is the bi-gram overlap ratio. $Len(s)$ is the length of s. t is a threshold parameter used to control the redundancy, which is usually set heuristically.

2.2 Our Framework

In this paper, we propose to directly model the relative importance $f(s|S)$ instead of the independent importance of each sentence $f(s)$. Specially, we model the importance of s given the sentences S as follows:

$$f(s|S) = \min_{s' \in S} f(s|s') \tag{2}$$

which considers the minimum relative importance of sentence s with respect to each sentence of S. $f(s|s')$ models the relative importance of sentence s given sentence s', which makes Equation 2 a redundancy-aware framework.

When generating summaries, we select the first sentence by treating $s' = \emptyset$ or using a $f(s)$ model. Then, we select the rest summary sentences with the following greedy algorithm:

$$s^* = \arg \max_{s \subset D \setminus S} \min_{s' \in S} f(s|s') \tag{3}$$

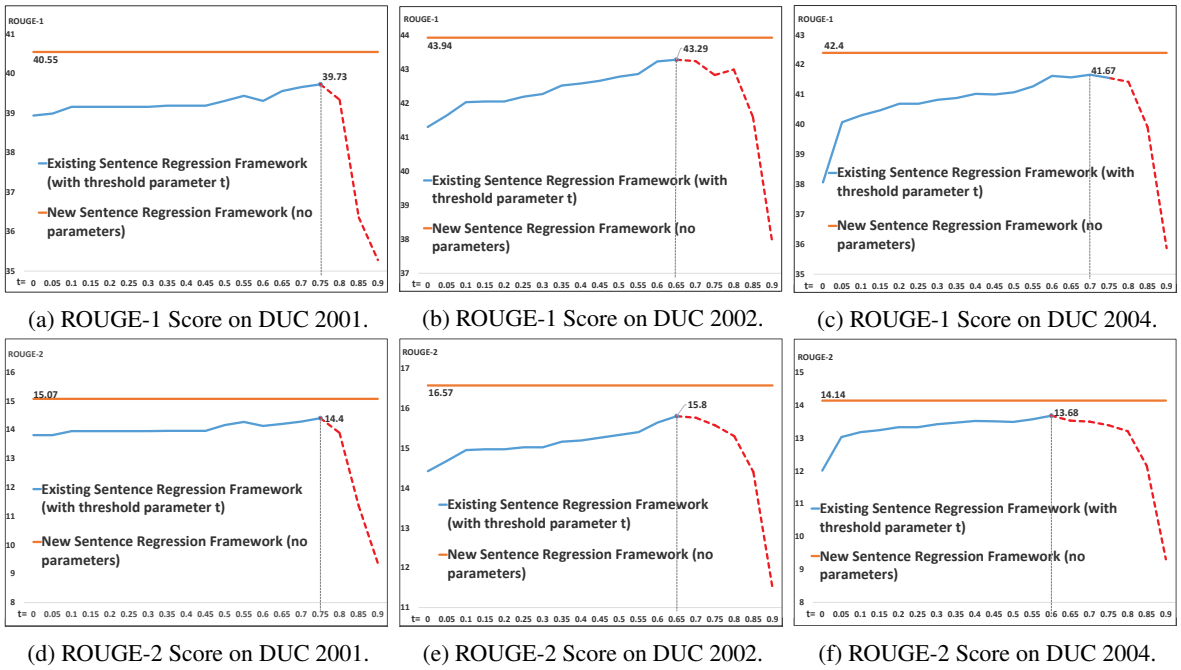
The algorithm starts with the first selected sentence. In each step, a new sentence is added to the summary that results in the maximum relative increase according to Equation 3. The algorithm terminates when the summary length constraint is reached.

Next we conduct experiments to analyze the upper bounds of the new framework compared with the existing framework (Equation 1). To this end, we compute $f(s)$ and $f(s|s')$ as follows:

$$\begin{aligned} &f(s) = ROUGE\text{-}2(s|S_{ref}) \\ &f(s|s') = f(\{s,s'\}) - f(s') = ROUGE\text{-}2(\{s,s'\}|S_{ref}) - ROUGE\text{-}2(s'|S_{ref}) \end{aligned} \tag{4}$$

where S_{ref} is one or several summaries written by people. The ROUGE-2 recall metric gives a score to a set of sentences with respective to the human written summaries. We compute $f(s|s')$ as the total gain of s and s' ($f(\{s,s'\})$) subtracted by the individual gain of s' ($f(s')$). Equation 4 can be seen as the groundtruth computation of $f(s)$ and $f(s|s')$.

The experimental upper bounds of different frameworks are shown in Figure 1. Similar results of ROUGE-1 and ROUGE-2 are achieved on all three benchmark datasets from DUC 2001, 2002 and 2004. The *advantages* of the new framework (Equation 2) are three-fold compared with the framework of Equation 1. First, there is no parameter to be tuned manually. By comparison, Equation 1 has a threshold parameter t, which is very sensitive around the best performance, as shown in the red dashed line parts of Figure 1. Second, the new framework has a higher upper bound, which means there is a bigger potential



(a) ROUGE-1 Score on DUC 2001. (b) ROUGE-1 Score on DUC 2002. (c) ROUGE-1 Score on DUC 2004.

(d) ROUGE-2 Score on DUC 2001. (e) ROUGE-2 Score on DUC 2002. (f) ROUGE-2 Score on DUC 2004.

Figure 1: Experimental Upper bounds of our sentence regression framework and existing sentence regression framework.

for improvement. Finally, besides single sentence features, additional features involving the relations of two sentences can be extracted to improve the regression performance.

The new proposed framework also has some *challenges*. First, the groundtruth of $f(s|s')$ is usually unavailable for many tasks. Fortunately, in the text summarization task, the groundtruth of $f(s|s')$ can be computed according to Equation 4. Second, the number of training instances is $O(|C||D|^2)$ ($O(|C||D|)$ for Equation 1). We come up with two ways to speed up the training process in the next session.

3 Implementation

3.1 Objective Function

We implement $f(s|s')$ with MultiLayer Perceptron (MLP) (Ruck et al., 1990; Gardner and Dorling, 1998).

$$f(s|s') = MLP\left(\Phi(s|s')|\theta\right) \tag{5}$$

where $\Phi(s|s')$ is the set of features and θ is the parameters to be learned.

We use the standard Mean Square Error (MSE) as the loss function as follows:

$$\begin{aligned} L(\theta) &= \frac{1}{|C||D|(|D|-1)} \sum_{D \in C} \sum_{s \in D} \sum_{\substack{s' \in D; \\ s' \neq s}} Err(s|s') \\ Err(s|s') &= \left(MLP\left(\Phi(s|s')|\theta\right) - ROUGE(s|s', S_{ref})\right)^2 \\ ROUGE(s|s', S_{ref}) &= ROUGE\text{-}2(\{s, s'\}|S_{ref}) - ROUGE\text{-}2(s'|S_{ref}) \end{aligned} \tag{6}$$

We use ROUGE-2 recall as the groundtruth score due to its high capability of evaluating automatic summarization systems (Owczarzak et al., 2012).

The s' in $f(s|s')$ should mainly refer to the sentences that have a big potential to be selected into the summary. To this end, we do not have to treat each sentence in D as s' during training. We can accelerate the training process by generating a set of sentences S' from D. We come up with two ways as shown in Algorithm 1. The first way is using the greedy strategy (Line 4 of Algorithm 1). Specifically, for each training episode of sentence s, we use the current model to generate the summary with greedy algorithm

as a part of the S'. We refer to this part as S'_1. The advantage is that S'_1 is adaptively generated with respective to the training status of the model. The second way is randomly sampling a small set of s' with respect to its groundtruth ROUGE score (Line 6 of Algorithm 1). Specifically, for each training episode of sentence s, we sample a small set S'_2 according to the following rule:

$$\begin{cases} NotSelected & rnd(0,1) > 0.05 * ROUGE\text{-}2(s) + 0.05 \\ Selected & Otherwise \end{cases} \tag{7}$$

where $rnd(0,1)$ generates random number from a uniform distribution within the range $[0,1]$. $ROUGE\text{-}2(s)$ is normalized to $[0,1]$. Each sentence is selected with at least 5% probability and sentences with higher ROUGE scores have higher probabilities. Different probabilities will influence the speed of the training process but will not make much difference in the final results according to our experiments. We use the randomly sampled S'_2 to avoid the premature convergence caused by S'_1. Finally, $S' = S'_1 \bigcup S'_2$. In this way, the number of training instances is $O(|C||D||S'|)$ while originally it is $O(|C||D|^2)$, where C is the set of all D in the training corpus. Note that $|S'|$ is a very small number compared to $|D|$.

Algorithm 1 The adaptive & randomized training.

Input:
Training corpus, C;
Max iterations, N;
Output:
Model parameters, θ;
Randomly initialize the parameters θ;
for $i = 1; i < N; i$++ **do**
  for each D such that $D \in C$ **do**
    Generate S'_1 greedily according to Equation 3;
    for each sentence s such that $s \in D$ **do**
      Generate S'_2 randomly according to Equation 7;
      for each s' such that $s' \in S'_1 \bigcup S'_2, s' \neq s$ **do**
        Make forward and backward propagation w.r.t the loss $L(\theta)$ (Equation 6);
        Update the model parameters θ;
      end for
    end for
  end for
  if θ converges **then**
    break;
  end if
end for
return θ;

3.2 Feature

We employ two groups of features in terms of sentence importance and redundancy, namely Sentence Importance Features and Sentence Relation Features. The former are widely studied by existing methods (Gupta et al., 2007; Li et al., 2007; Aker et al., 2010; Ouyang et al., 2011; Galanis et al., 2012; Hong et al., 2015). However, to our knowledge, the latter are firstly incorporated into a regression model in this paper. Details of used features are listed in Table 1. We use Sentence Importance Features to model the independent sentence importance of s. $Len(s)$, $Position(s)$, $Stop(s)$, $TF(s)$ and $DF(s)$ are commonly used features. Embedding feature $Emb(s)$ is an effective feature that encodes the sentence content which can be seen as summary prior nature of the sentence (Cao et al., 2015b). We use Sentence Relation Features to evaluate the content overlap between s and s'. $Match\text{-}P(s \cap s')$ and $Match\text{-}R(s \cap s')$ evaluate the ratio of the overlap words, while $TF(s \cap s')$, $DF(s \cap s')$ and $Stop(s \cap s')$ evaluate the importance of the overlap words. $Cos(s, s')$ evaluates the exact match similarity while $Emb\text{-}Cos(s, s')$ evaluates the meaning match similarity. All features in Table 1 are basic features commonly used in summarization.

Features	Formulations	Descriptions
Sentence Importance Features	$Len(s)$	Length of s
	$Position(s)$	Position of s in its document
	$Stop(s) = \frac{stop\text{-}count(s)}{Len(s)}$	Stop words ratio of s
	$TF(s) = \frac{\sum_{w \in s} GTF(w)}{Len(s)}$	Average Term Frequency $GTF(w)$ is the Global Term Frequency
	$DF(s) = \frac{\sum_{w \in s} DF(w)}{Len(s)}$	Average Document Frequency
	$Emb(s) \frac{\sum_{w \in s} Emb(w)}{Len(s)}$	Average Word Embedding $Emb(w)$ is the word embedding
Sentence Relation Features	$Match\text{-}P(s, s') = \frac{Match(s,s')}{Len(s)}$	Term match precision $Match\text{-}P(s, s') = 0$ if $s \cap s' = \emptyset$
	$Match\text{-}R(s, s') = \frac{Match(s,s')}{Len(s')}$	Term match recall $Match\text{-}R(s, s') = 0$ if $s \cap s' = \emptyset$
	$TF(s, s') = \frac{Len(s \cap s')}{\sum_{w \in s \cap s'} GTF(w)}$	Average Global Term Frequency of overlap $s \cap s'$ $TF(s, s') = 0$ if $s \cap s' = \emptyset$
	$DF(s, s') = \frac{Len(s \cap s')}{\sum_{w \in s \cap s'} DF(w)}$	Average Document Frequency of overlap $s \cap s'$ $DF(s, s') = 0$ if $s \cap s' = \emptyset$
	$Stop(s, s') = 1 - \frac{Stop\text{-}Count(s \cap s')}{Len(s \cap s')}$	Stop words ratio of overlap $s \cap s'$ $Stop(s, s') = 0$ if $s \cap s' = \emptyset$
	$Cos(s, s')$ $= Cosine(GTF(s), GTF(s'))$	Cosine of Global Term Frequency vector
	$Emb\text{-}Cos(s, s')$ $= Cosine(Emb(s), Emb(s'))$	Cosine of average embedding vector

Table 1: Summary of features

4 Experiment

4.1 Experimental Setup

Datasets. The benchmark evaluation corpora for summarization are the ones published by the Document Understanding Conferences (DUC[1]). We focus on the generic multi-document summarization task, so we carried out all our experiments on DUC 2001, 2002 and 2004 datasets. The documents are all from the news domain and are grouped into various thematic clusters. For each document set, we concatenated all the articles and split them into sentences using the tool provided with the DUC 2003 dataset. We train the model on two years' data and test it on the other year.

Evaluation Metric. ROUGE metrics are the official metrics of the DUC extractive summarization tasks (Rankel et al., 2013). We use the official ROUGE tool[2] to evaluate the performance of the baselines as well as our approach (Lin, 2004). The parameter of length constraint is "-l 100" for DUC 2001/2002, and "-b 665" for DUC 2004. We take ROUGE-2 recall as the main metric for comparison because Owczarzak et al. prove its high capability of evaluating automatic summarization systems (Owczarzak et al., 2012).

Comparison Methods. The comparison methods used in the experiments are listed as follows.

- LexRank: State-of-the-art summarization model (Erkan and Radev, 2004).
- ClusterHITS: State-of-the-art results on DUC 2001 (Wan and Yang, 2008).
- ClusterCMRW: State-of-the-art results on DUC 2002 (Wan and Yang, 2008).
- REGSUM[3]: State-of-the-art results on DUC 2004 (Hong and Nenkova, 2014).
- R2N2_GA/R2N2_ILP: State-of-the-art results on DUC 2001/2002/2004 (Cao et al., 2015a) with a neural network regression model.
- PriorSum: To our knowledge, the best results on DUC 2001, 2002 and 2004 using regression approaches (Cao et al., 2015b).
- SR (Sentence Regression): Evaluate sentence importance with MLP and the Sentence Importance Features in Table 1 and select the top ranks as the summary (without handling redundancy).

[1] http://duc.nist.gov/

[2] ROUGE-1.5.5 with options: -n 2 -m -u -c 95 -x -r 1000 -f A -p 0.5 -t 0

[3] REGSUM truncates a summary to 100 words.

	DUC 2001		DUC 2002		DUC 2004	
	ROUGE-1	ROUGE-2	ROUGE-1	ROUGE-2	ROUGE-1	ROUGE-2
BestSentence	37.32	10.44	39.75	11.60	40.36	11.68
Strategy 1	36.31	8.49	37.80	9.61	39.60	10.57
Strategy 2	36.32	8.52	37.82	9.26	38.75	10.19

Table 2: First sentence selection strategies

- t-SR (threshold t based Sentence Regression): Evaluate sentence importance with MLP and the Sentence Importance Features in Table 1 and generate the summary with greedy by directly discarding the redundant sentence according to Equation 1.
- RASR (**R**edundancy-**A**ware **S**entence **R**egression): The proposed method in this paper.

Note that for the methods with the parameter t, we tried all values of ranging from 0 to 1 with a step size of 0.05. The final value of t on each dataset is decided by 3-fold cross validation on the training datasets.

Model Configuration. The word embedding used in this paper is trained on the English Wikipedia Corpus[4] with Google's Word2Vec tool[5]. The dimension is 300. We use 4 hidden layers MLP with tanh activation function and the sizes of the layers are $[300, 200, 100, 1]$. To update the weights of MLP, we apply the diagonal variant of AdaGrad with mini-batches. We set the mini-batch size to 20.

4.2 Results and Analysis

First Sentence Selection. Remember that when generating a summary, our method first selects the first sentence then greedily selects the rest sentences with respective to $f(s|S)$. We tried two strategies to select the first sentence with RASR. Strategy 1: treating RASR as an united model by setting the Sentence Relation Features to zero when fitting $f(s)$ during training period or selecting the first sentence during test period. Strategy 2: treating RASR as two models that fit $f(s)$ and $f(s|S)$ respectively. The former is used to select the first sentence and the latter is used to select the rest sentences. We also use the sentence that gets the highest ROUGE-2 score as the first sentence as a comparison, namely BestSentence. The results are shown in Table 2. As expected, BestSentence is much better than Strategy 1 and Strategy 2, which means selecting a better first sentence will greatly improve the performance of RASR. It does not make too much difference whether using Strategy 1 or Strategy 2. We report the results of Strategy 1 to compare with the baseline methods in Table 3.

Performance Analysis. As shown in Table 3, the bold face indicates the best performance. Generally, our method RASR achieves the best performance in terms of ROUGE-2 metric on all three datasets. The improvement of ROUGE-2 on DUC 2001 is significant with p-value < 0.05 compared with LexRank, SR and t-SR. Although ClusterHITS and ClusterCMRW get higher ROUGE-1 scores, their ROUGE-2 scores are much lower. In contrast, our method works quite stably.

The improvements of our method come from two aspects. First, it is effective to model sentence importance and redundancy simultaneously with multiple nonlinear function transformations. This can be reflected by the following comparison experiments. SR does not handle redundancy at all, so it achieves bad performance especially on the DUC 2004 corpus. The other methods in Table 3 model sentence importance and redundancy in two separate processes by first ranking the sentences and then discarding the redundant ones whose bi-gram overlap ratio is larger than a threshold parameter. Although we tune the threshold parameter carefully, RASR still outperforms them. Second, effective features involving the sentence relations (i.e., Sentence Relation Features) are considered which cannot be incorporated by the baseline methods.

[4]https://en.wikipedia.org/wiki/Wikipedia:Database_download
[5]https://code.google.com/archive/p/word2vec/

	System	ROUGE-1	ROUGE-2
DUC 2001	Peer T	33.03	7.86
	ClusterHITS*	**37.42**	6.81
	LexRank	33.43	6.09
	R2N2_GA*	35.88	7.64
	R2N2_ILP*	36.91	7.87
	PriorSum*	35.98	7.89
	SR	35.34	7.67
	t-SR	35.41	7.76
	RASR	36.31	**8.49**
DUC 2002	Peer 26	35.15	7.64
	ClusterCMRW*	**38.55**	8.65
	LexRank	35.29	7.54
	R2N2_GA*	36.84	8.52
	R2N2_ILP*	37.96	8.88
	PriorSum*	36.63	8.97
	SR	36.70	8.59
	t-SR	37.49	8.95
	RASR	37.80	**9.61**
DUC 2004	Peer 65	37.88	9.18
	REGSUM*	38.57	9.75
	LexRank	37.87	8.88
	R2N2_GA*	38.16	9.52
	R2N2_ILP*	38.78	9.86
	PriorSum*	38.91	10.07
	SR	35.76	8.73
	t-SR	38.36	9.98
	RASR	**39.60**	**10.57**

Peer T/Peer 26/Peer 65 are the original results on DUC 2001/2002/2004 respectively. We cite the scores of some systems from their papers, indicated with the sign "*".

Table 3: Comparison results (%) on DUC datasets

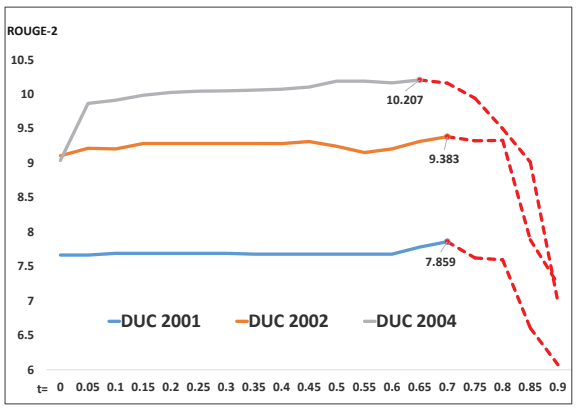
Parameter Sensitiveness. We present the ROUGE-2 performance of t-SR with the threshold parameter t ranging from 0 to 0.9 with a step size of 0.05 shown in Figure 1 and 2a. The best achieved performances of the groundtruth implementation are around 0.75, 0.65, 0.6 (Figure 1) while the best achieved performances in practice are around 0.7, 0.7, 0.65 (Figure 2a). t is still very sensitive around the best performance, as shown in the red dashed line in both Figure 1 and 2a.

Training Convergence. In order to speed up the training process of RASR, we randomly sample some pairwise training instances with Equation 7 for training of RASR. We want to know whether this will influence the convergence of RASR, so we present the decrease of loss with respect to training iterations in Figure 2b. We find that the random sampling has little influence on the convergence of RASR with t-SR as a comparison.

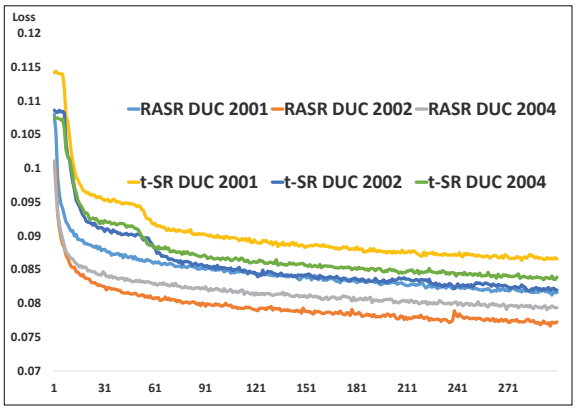
5 Related Work

Existing work on extractive summarization can be divided into two categories: unsupervised and supervised.

Two most famous unsupervised frameworks are Centroid based and Maximum Marginal Relevance based. Centroid-based methods evaluate the sentence centrality as its importance (Mihalcea, 2004). Radev et al. first propose to model cluster centroids in their summarization system, MEAD (Radev et al., 2000; Radev et al., 2004). Then LexRank (or TextRank) is proposed to compute sentence importance



(a) Sensitiveness of the parameter t of t-SR.



(b) Loss decrease with respect to training iterations.

based on the concept of eigenvector centrality in a graph of sentence similarities (Erkan and Radev, 2004; Mihalcea and Tarau, 2004). Due to its expansibility and flexibility, centroid-based methods have a lot of extensions. Wan et al. propose several centroid-based approaches for different summarization tasks, e.g., cross-language summarization, etc (Wan, 2008; Wan and Xiao, 2009; Wan, 2011). Maximum Marginal Relevance (MMR) based methods consider the linear trade-off between relevance and redundancy (Carbonell and Goldstein, 1998). Goldstein et al. first extend MMR to support extractive summarization by incorporating additional information (Goldstein et al., 2000). McDonald achieves good results by reformulating this as a knapsack packing problem and solving it using ILP (McDonald, 2007). Later Lin and Bilmes propose a variant of MMR framework which maximizes an objective function that considers the linear trade-off between coverage and redundancy terms (Lin and Bilmes, 2010; Lin and Bilmes, 2011).

Supervised methods model the extractive summarization task from various perspectives. Kupiec et al. train a naive-Bayes classifier to decide whether to include a particular sentence in the summary or not. (Kupiec et al., 1995). Li et al. evaluate the sentence importance with support vector regression, then a simple rule-based method is applied for removing redundant phrases (Li et al., 2007). Gillick and Favre evaluate bi-grams importance and then use these scores to evaluate sentence importance and redundancy with a linear combination (Gillick and Favre, 2009). Sipos et al. propose a structural SVM learning approach to learn the weights of feature combination using the MMR-like submodularity function proposed by Lin and Bilmes (Lin and Bilmes, 2010). Cao et al. evaluate the sentence importance with a neural regression model, then they remove the redundant sentence larger than a threshold parameter during greedy algorithm (Cao et al., 2015b). In another paper, they remove the redundant sentence by adding a redundancy constraint to the ILP objective which restricts the bi-gram redundancy of the selected sentences smaller than a threshold (Cao et al., 2015a).

In all above extractive summarization methods, redundancy is mainly considered in two ways. The first way is measuring the importance of each sentence then explicitly removing the redundant sentence larger than a threshold parameter during the sentence selection process. Another way is linearly substracting the sentence redundancy score or scoring the redundant parts with low weights. To the best of our knowledge, none of them studies the summarization task and models redundancy from the perspective of this paper.

6 Conclusion and Future Work

This paper presents a novel sentence regression framework to conduct regression with respect to the relative importance $f(s|S)$ of sentence s given a set of sentences S. Additional features involving the sentence relations are incorporated. We conduct experiments on three DUC benchmark datasets. Generally, our approach achieves the best performance in terms of ROUGE metrics compared with state-of-the-art approaches.

We believe our work can be advanced and extended from many different perspectives. First, more features can be designed especially those involving the relations of two sentences. Second, the results

can be further improved by exploring better strategies to select the first sentence. Third, the framework can be extended to other tasks, e.g., query-focused summarization, which can be achieved by introducing query-related features.

Acknowledgement

This work is supported by the Natural Science Foundation of China (61672322, 61672324, 61272240), Microsoft Fund (FY14-RES-THEME-025), the Natural Science Foundation of Shandong Province (ZR2012FM037), the Excellent Middle-Aged and Youth Scientists of Shandong Province (BS2012DX017) and the Fundamental Research Funds of Shandong University.

References

Ahmet Aker, Trevor Cohn, and Robert Gaizauskas. 2010. Multi-document summarization using a* search and discriminative training. In *Proceedings of EMNLP*, pages 482–491. Association for Computational Linguistics.

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015a. Ranking with recursive neural networks and its application to multi-document summarization. In *Proceedings of AAAI*, pages 2153–2159. AAAI Press.

Ziqiang Cao, Furu Wei, Sujian Li, Wenjie Li, Ming Zhou, and Houfeng Wang. 2015b. Learning summary prior representation for extractive summarization. *Proceedings of ACL*, pages 829–833.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of SIGIR*, pages 335–336. ACM.

John M Conroy, Judith D Schlesinger, Jade Goldstein, and Dianne P Oleary. 2004. Left-brain/right-brain multi-document summarization. In *Proceedings of the DUC*.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, pages 457–479.

Dimitrios Galanis, Gerasimos Lampouras, and Ion Androutsopoulos. 2012. Extractive multi-document summarization with integer linear programming and support vector regression. In *Proceedings of COLING*, pages 911–926. Citeseer.

Michel Galley. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proceedings of EMNLP*, pages 364–372. Association for Computational Linguistics.

Matt W Gardner and SR Dorling. 1998. Artificial neural networks (the multilayer perceptron)a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14):2627–2636.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 10–18. Association for Computational Linguistics.

Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. 2000. Multi-document summarization by sentence extraction. In *Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic summarization*, pages 40–48. Association for Computational Linguistics.

Surabhi Gupta, Ani Nenkova, and Dan Jurafsky. 2007. Measuring importance and query relevance in topic-focused multi-document summarization. In *Proceedings of ACL*, pages 193–196. Association for Computational Linguistics.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *Proceedings of EACL*, pages 712–721.

Kai Hong, Mitchell Marcus, and Ani Nenkova. 2015. System combination for multi-document summarization. In *Proceedings of EMNLP*, pages 107–117. The Association for Computational Linguistics.

Yue Hu and Xiaojun Wan. 2013. Ppsgen: Learning to generate presentation slides for academic papers. In *Proceedings of IJCAI*. AAAI Press.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of SIGIR*, pages 68–73. ACM.

Sujian Li, You Ouyang, Wei Wang, and Bin Sun. 2007. Multi-document summarization using support vector regression. In *Proceedings of DUC*. Citeseer.

Chen Li, Xian Qian, and Yang Liu. 2013. Using supervised bigram-based ilp for extractive summarization. In *Proceedings of ACL*, pages 1004–1013. Citeseer.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 912–920. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 510–520. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *Proceedings of COLING*, pages 495–501. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. *Advances in Information Retrieval*, pages 557–564.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of EMNLP*, pages 404–411, Barcelona, Spain, July. Association for Computational Linguistics.

Rada Mihalcea. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of ACL*, page 20. Association for Computational Linguistics.

Miles Osborne. 2002. Using maximum entropy for sentence extraction. In *Proceedings of the ACL-02 Workshop on Automatic Summarization*, pages 1–8. Association for Computational Linguistics.

You Ouyang, Sujian Li, and Wenjie Li. 2007. Developing learning strategies for topic-based summarization. In *Proceedings of CIKM*, pages 79–86. ACM.

You Ouyang, Wenjie Li, Sujian Li, and Qin Lu. 2011. Applying regression models to query-focused multi-document summarization. *Information Processing & Management*, 47(2):227–237.

Karolina Owczarzak, John M Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9. Association for Computational Linguistics.

Dragomir R Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of NAACL-ANLP*, pages 21–30. Association for Computational Linguistics.

Dragomir R Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing & Management*, 40(6):919–938.

Peter A Rankel, John M Conroy, Hoa Trang Dang, and Ani Nenkova. 2013. A decade of automatic content evaluation of news summaries: Reassessing the state of the art. In *Proceedings of ACL*, pages 131–136. Association for Computational Linguistics.

Dennis W Ruck, Steven K Rogers, Matthew Kabrisky, Mark E Oxley, and Bruce W Suter. 1990. The multilayer perceptron as an approximation to a bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1(4):296–298.

Xiaojun Wan and Jianguo Xiao. 2009. Graph-based multi-modality learning for topic-focused multi-document summarization. In *Proceedings of IJCAI*, pages 1586–1591. AAAI Press.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. In *Proceedings of SIGIR*, pages 299–306. ACM.

Xiaojun Wan and Jianmin Zhang. 2014. Ctsum: extracting more certain summaries for news articles. In *Proceedings of SIGIR*, pages 787–796. ACM.

Xiaojun Wan, Ziqiang Cao, Furu Wei, Sujian Li, and Ming Zhou. 2015. Multi-document summarization via discriminative summary reranking. *arXiv preprint arXiv:1507.02062*.

Xiaojun Wan. 2008. An exploration of document impact on graph-based multi-document summarization. In *Proceedings of EMNLP*, pages 755–762. Association for Computational Linguistics.

Xiaojun Wan. 2011. Using bilingual information for cross-language document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1546–1555. Association for Computational Linguistics.

Wen-tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. 2007. Multi-document summarization by maximizing informative content-words. In *Proceedings of IJCAI*, volume 7, pages 1776–1782. AAAI Press.

Generating Video Description using Sequence-to-sequence Model with Temporal Attention

Natsuda Laokulrat[1], Sang Phan[2], Noriki Nishida[3], Raphael Shu[3], Yo Ehara[1], Naoaki Okazaki[4,1], Yusuke Miyao[2,1] and Hideki Nakayama[3,1]

[1]Artificial Intelligence Research Center (AIRC), National Institute of Advanced Industrial Science and Technology (AIST), Japan
[2]National Institute of Informatics (NII), Japan
[3]The University of Tokyo, Japan
[4]Tohoku University, Japan

Abstract

Automatic video description generation has recently been getting attention after rapid advancement in image caption generation. Automatically generating description for a video is more challenging than for an image due to its temporal dynamics of frames. Most of the work relied on Recurrent Neural Network (RNN) and recently attentional mechanisms have also been applied to make the model learn to focus on some frames of the video while generating each word in a describing sentence. In this paper, we focus on a sequence-to-sequence approach with temporal attention mechanism. We analyze and compare the results from different attention model configuration. By applying the temporal attention mechanism to the system, we can achieve a METEOR score of 0.310 on Microsoft Video Description dataset, which outperformed the state-of-the-art system so far.

1 Introduction

Since the recent breakthrough in machine learning, generating description for static images has been intensively researched and high-quality image description can be achieved in the past few years by many research groups (Vinyals et al., 2014; Karpathy and Fei-Fei, 2015; Fang et al., 2015; Xu et al., 2015; You et al., 2016). However, video description generation is a much more challenging task, which requires understanding temporal relationship between video frames. Automatically generating video description can be useful in many aspects. It can help visually-impaired people to understand the content of videos. More importantly, it will enable computers to understand videos, rather than just working at pixel levels, because the generated descriptions contain objects appearing the videos along with their attributes, locations, actions, and relations with other objects. Though considered very challenging, being able to understand videos can have great impact and will be useful to many other applications, such as human-robot interaction, video indexing and query, and video classification.

This paper focuses on generating video description using a encoder-decoder sequence-to-sequence model with temporal attention mechanism. We perform a set of experiments using different configurations of attention mechanisms and also can achieve state-of-the-art results. Our main contributions are as follows: First, we apply temporal attention mechanism to the encoder-decoder sequence-to-sequence model and are able to outperform the state-of-the-art system on Microsoft Video Description dataset (MSVD) (Chen and Dolan, 2011) in terms of the METEOR (Denkowski and Lavie, 2014), CIDEr (Vedantam et al., 2014), and ROUGE-L (Lin, 2004) scores. Second, we analyze and compare the results from different model configurations, and show how temporal attention works in generating sentences. We also performed the experiments on a large movie dataset, Montreal Movie Annotation Dataset (M-VAD) collected by Torabi et al. (2015).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 44–52, Osaka, Japan, December 11-17 2016.

2 Related Work

There exist many works in the image captioning task inspired by recent advances in machine translation using encoder-decoder Recurrent Neural Network (RNN) (Bahdanau et al., 2014). Vinyals et al. (2014) replaced the RNN encoder with a Convolutional Neural Network (CNN) which was pre-trained for an image classification task. Then, the last hidden layer of the pre-trained CNN can be used to produce a meaningful representation of an image, which they used as an input to the RNN decoder that generates sentences.

Karpathy and Fei-Fei (2015) used a combination of a CNN and a bidirectional RNN to generate natural language sentences from an image as well as their corresponding regions. Fang et al. (2015) first trained visual detectors from a dataset of image-caption pairs, and then used the output words from the visual detectors as input to the language model for generating image description. Xu et al. (2015) also used a CNN-RNN encoder-decoder scheme and applied a spatial attention mechanism over an input image, so that the model can attend to a specific region of an image while generating each word of a caption sentence. The model of You et al. (2016) learned to selectively attend to semantic concepts (similar to visual concepts in (Fang et al., 2015)) of an image and input them into the RNN decoder at each time step of generating image description.

After great success in image captioning, researchers are currently moving forward into working on generating sentences that describe videos. Venugopalan et al. (2015b) proposed the first end-to-end system to translate a video into natural language by extending the CNN-RNN encoder-decoder framework for image captioning proposed by Vinyals et al. (2014) to generate description for videos. They performed a mean pooling over CNN feature vectors of frames to generate a single vector representation for a video, and then use the vector as input to the RNN decoder to generate a sentence. Yao et al. (2015) has incorporated an attentional mechanism to video caption generation. They took into account both local and global temporal structures of videos by incorporating a spatial temporal 3D CNN.

A sequence-to-sequence model for generating description of videos has been first proposed by Venugopalan et al. (2015a). They used 2 layers of RNN for both encoding the videos and decoding into sentences, so their model is able to learn both a temporal structure of a sequence of video frames and a sequence model for generating sentences.

3 Sequence-to-sequence Model

This section describes the concept and structure of the sequence-to-sequence model, including Long Short-Term Memory (LSTM), the two-layer encoder-decoder LSTM, and the temporal attention mechanism that we used in this work.

3.1 Long Short-Term Memory

An LSTM network, proposed by Hochreiter and Schmidhuber (1997), is a type of RNN that is commonly used in sequence-to-sequence models. It has been intensively used in machine translation, speech recognition as well as in image/video description generation.

LSTM can help to avoid exploding and vanishing gradient problems by using forget gates to reset memory block when they are out of date. Given an input x_t, at time step t, one unit of an LSTM can be formulated as

$$\begin{aligned} i_t &= sigmoid(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\ f_t &= sigmoid(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\ o_t &= sigmoid(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\ g_t &= tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g) \\ c_t &= f_t * c_{t-1} + i_t * g_t \\ h_t &= o_t * tanh(c_t) \end{aligned} \tag{1}$$

where i_t, f_t and o_t are input gates, forget gates, and output gates. The symbol $*$ represents the element-wise multiplication. W_{xi}, W_{hi}, W_{xf}, W_{hf}, W_{xo}, W_{ho}, W_{xg}, W_{hg} and b_i, b_f, b_o, b_g are the parameters

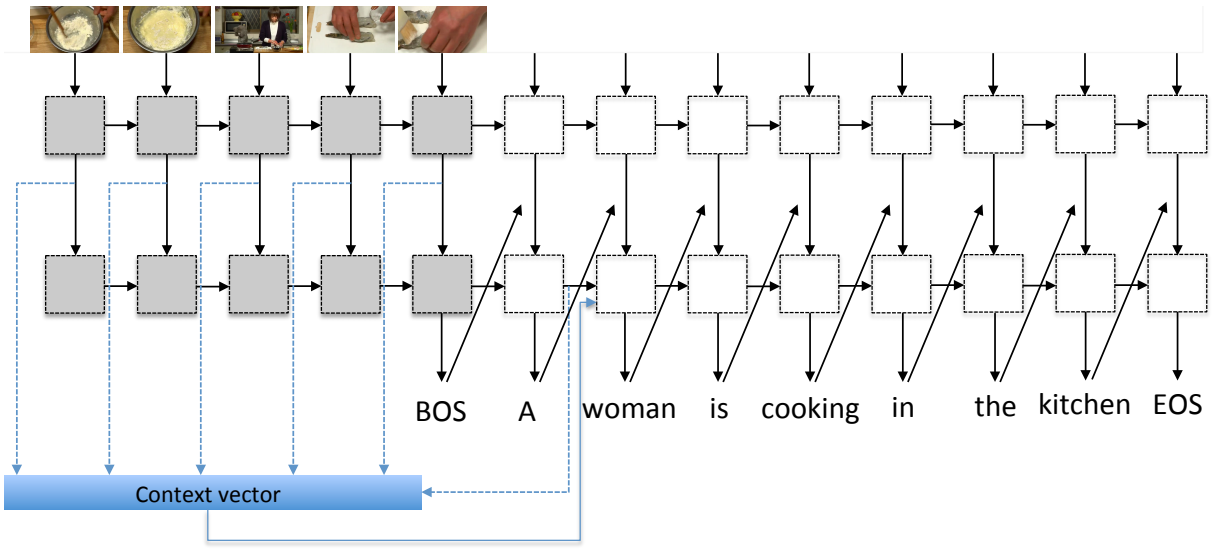


Figure 1: System architecture of the sequence-to-sequence model with temporal attention. In the figure, we omit the image embedding layer, the word embedding layer, and the softmax layer, due to the space constraint.

to be learned during training. h_t is the hidden state at time step t which will be an input to the next time step's LSTM unit.

3.2 Two-layer LSTM for Sequence-to-sequence Model

Figure 1 depicts our two-layer LSTM model for generating sentences from a video, which is based on the sequence-to-sequence model proposed by Venugopalan et al. (2015a). Given a video as a sequence of frames $V = \{v_1, v_2, ..., v_n\}$, where the video V has n frames and v_i is the i^{th} frame of the video, we can formulate our system as

$$h_t^{(1)} = LSTM^{(1)}(x_t, h_{t-1}^{(1)}) \tag{2}$$

where $h_t^{(1)}$ is the hidden state of the first (upper) LSTM layer, defined as $LSTM^{(1)}$, at time step t. In the encoding stage, the input $x_t = v_t$ and, in the decoding stage, $x_t = \vec{0}$.

The input to the second (lower) LSTM layer is the concatenation of the word (represented as word embedding) generated on the previous time step $t-1$ and the hidden state of the first LSTM layer.

$$h_t^{(2)} = LSTM^{(2)}([w_{t-1}; h_t^{(1)}], h_{t-1}^{(2)}) \tag{3}$$

where $h_t^{(2)}$ is the hidden state of the second LSTM layer, defined as $LSTM^{(2)}$, at time step t. In the encoding stage, we fix the word $w_{t-1} = \vec{0}$, since there is no word being generated. Lastly, the distribution over all the words at time step t can be computed by

$$p(w_t|w_1, ..., w_{t-1}, V) = softmax(W_s h_t^{(2)} + b_s) \tag{4}$$

3.3 Temporal Attention Mechanism

Our approach incorporates the previously-proposed sequence-to-sequence model with a temporal attention mechanism. The second-layer LSTM at decoding stage can be formulated as

$$h_t^{(2)} = LSTM^{(2)}([w_{t-1}; h_t^{(1)}], h_{t-1}^{(2)}, c_t) \tag{5}$$

where the context vector c_t, at the time step t in the decoding stage, is the weighted sum of encoder's hidden states $h_i^{(1)}$.

$$c_t = \sum_{i=1}^{n} \alpha_i^{(t)} h_i^{(1)} \tag{6}$$

	videos	sentences	tokens	vocab size	avg. length	captions/video	source
MSVD	1,970	80,827	567,874	12,594	10.2s	≈40	crowd
M-VAD	46,589	55,904	502,926	17,609	6.2s	1	professional

Table 1: Video description dataset statistics. Refer to Venugopalan et al. (2015a) and Torabi et al. (2015) for more details.

The weight $\alpha_i^{(t)}$ is computed at every time step t and can be computed by

$$\alpha_i^{(t)} = \frac{e^{a(h_i^{(1)}, h_{t-1}^{(2)})}}{\sum_{j=1}^{n} e^{a(h_j^{(1)}, h_{t-1}^{(2)})}} \tag{7}$$

where $a(h_i^{(1)}, h_{t-1}^{(2)})$ is the alignment function used to calculate relevance scores between every hidden state $h_i^{(1)}$ in the encoding stage the hidden state $h_{t-1}^{(2)}$ at the previous time step $t-1$.

3.4 Alignment model

To see which alignment functions are suitable for the model, we have used four different alignment functions in this work. Three of them were used in Luong et al. (2015), and we also use summation of hidden states as an alignment function.

$$a(h_i^{(1)}, h_{t-1}^{(2)}) = \begin{cases} {h_i^{(1)}}^\top h_{t-1}^{(2)} & \text{dot} \\ {h_i^{(1)}}^\top W_a h_{t-1}^{(2)} & \text{bilinear} \\ W_a[h_i^{(1)}; h_{t-1}^{(2)}] & \text{concat} \\ W_a h_i^{(1)} + W_b h_{t-1}^{(2)} & \text{sum} \end{cases} \tag{8}$$

The parameters W_a and W_b of the alignment model are jointly learned at training time with all other parameters in the network.

4 Dataset and Experiment Setting

This section describes the video description datasets, the pre-processing steps, the experiment setting, and the evaluation metrics that we used.

4.1 Dataset and pre-processing

In this paper, we trained the models and generated descriptions for two publicly-available video datasets as follows. The summary of the datasets is shown in Table 1.

Microsoft Research Video Description Corpus (MSVD) collected by Chen and Dolan (2011). It is a set of video clips aggregated from Youtube, containing 1,970 short clips with ≈40 captions/per clip. The videos were collected and annotated by crowdsourcing on Amazon Mechanical Turk. The clips mostly contain a single activity and can be described using only one sentence. For fair comparison with other previous work, we split the dataset into train/validation/test sets following Venugopalan et al. (2015b) and Yao et al. (2015). The size of the train, validation, and test sets is 1200, 100, and 670, respectively. We also use the pre-processed sentences and vocabularies from Venugopalan et al. (2015b). The pre-processing includes tokenizing, converting to lower case, and removing punctuations.

Montreal Video Annotation Dataset (M-VAD) M-VAD is a large collection of movie clips provided by Torabi et al. (2015). It was collected from 92 movies, and spitted into 46,589 short clips. Each clip is associated with a description, which can be more than one sentence. The dataset provides an official training/validation/test split, consisting of 36,921, 4,717 and 4,951 video clips respectively. We used all words in the training data as our vocabulary set and only pre-processed the data by tokenizing the sentences.

Model	BLEU	METEOR	CIDEr	ROUGE-L
Results reported by Venugopalan et al. (2015a)				
Mean pooling (VGG16)	-	0.277	-	-
Sequence to sequence (VGG16)	-	0.292	-	-
Sequence to sequence (VGG16) + Flow (AlexNet)	-	**0.298**	-	-
Results reported by Yao et al. (2015)				
Enc-Dec (Basic)	0.387	0.287	0.448	-
+ Local (3-D CNN)	0.388	0.283	0.509	-
+ Global (temporal attention)	0.403	0.290	0.480	-
+ Local + Global	**0.419**	**0.296**	**0.517**	-
Our system (VGG16) - *non-attention*	0.381	0.300	0.562	0.654
Our system (VGG16) - *dot*	**0.411**	0.307	0.574	0.664
Our system (VGG16) - *bilinear*	0.407	**0.310**	**0.615**	**0.676**
Our system (VGG16) - *concat*	0.390	**0.310**	0.595	0.667
Our system (VGG16) - *sum*	0.385	0.306	0.584	0.664
Our system (ResNet) - *non-attention*	0.427	0.318	0.706	0.675
Our system (ResNet) - *dot*	0.406	***0.326**	***0.750**	0.680
Our system (ResNet) - *bilinear*	0.425	0.318	0.733	0.675
Our system (ResNet) - *concat*	0.417	0.325	0.723	***0.681**
Our system (ResNet) - *sum*	***0.437**	0.319	0.718	0.676

Table 2: Scores of video description generation results on the MSVD dataset. * marks the top scores of each column.

4.2 Experiment Setting

We down-sample all the video clips by selecting every 8^{th} frame from the original videos and resize them to 224x224. We extract features for each frame using the pre-trained image classification models provided publicly in *Caffe Model Zoo* (Jia et al., 2014). In this work, we performed the experiments using the features extracted from the 4096-dimensional fc7 layer of the 16-layer VGG model (VGG16), proposed by Simonyan and Zisserman (2014), and the 2048-dimensional output from Deep Residual Networks (ResNet), recently proposed by He et al. (2015), who is the winner in the ILSVRC 2015 classification task. We embed input frame features into 512-dimensional embeddings.

For text input, after pre-processing, the word tokens are represented by one-hot vectors. We use the word BOS to mark the beginning of a sentence and EOS to represent the end of the sentence. We also embed the word vectors into 512-dimensional embeddings. The parameters for both image and word embedding layers are jointly learned with other parameters at training time.

We fix the number of encoding and decoding time steps in order to enable batch training. For the MSVD dataset, we constrain the number of encoding and decoding time steps to be 60 and 20, respectively. The M-VAD corpus has longer sentence length, so we set the number time steps to 50 and 30 for encoding and decoding, respectively, to allow the language model to generate longer sentences.

In every experiment, the LSTM hidden layer size is set to 1,000. We use the Adam optimizer (Kingma and Ba, 2014) with the learning rate of 0.0001 and the mini-batch size of 40. We also apply the dropout strategy (Srivastava et al., 2014) with the ratio of 0.3 at the video input layer to avoid overfitting. We implemented our system using *Chainer*, which is a powerful framework for developing neural networks developed by Tokui et al. (2015).

5 Experimental Results and Discussion

This section shows the experimental results in both qualitative and quantitative aspects. We performed a quantitative analysis of results based on four evaluation metrics, including

Model	BLEU	METEOR	CIDEr	ROUGE-L
Results reported by Venugopalan et al. (2015a)				
Mean pooling (VGG16)	-	0.061	-	-
Sequence to sequence (VGG16)	-	**0.067**	-	-
Results reported by Yao et al. (2015)				
Enc-Dec (Basic)	0.003	0.044	0.044	-
+ Local (3-D CNN)	0.004	0.051	0.050	-
+ Global (temporal attention)	0.003	0.040	0.047	-
+ Local + Global	**0.007**	**0.057**	**0.061**	-
Our system (VGG16) - *non-attention*	***0.008**	***0.072**	0.087	***0.159**
Our system (VGG16) - *dot*	***0.008**	0.062	***0.088**	0.140
Our system (VGG16) - *concat*	0.006	0.067	0.082	0.143
Our system (VGG16) - *sum*	0.007	0.070	0.074	0.155

Table 3: Scores of video description generation results on the M-VAD dataset. * marks the top scores of each column.

- **BLEU** (Papineni et al., 2002), an evaluation metric widely used in machine translation. BLEU calculates a score based on modified n-gram precision of the generated sentence against a set of human-annotated reference sentences.

- **METEOR** (Denkowski and Lavie, 2014), an automatic metric for machine translation evaluation. It is based on explicit word-to-word matching between the generated sentence and one or more reference sentences. METEOR supports matching between words with simple morphological variants and synonyms.

- **CIDEr** (Vedantam et al., 2014), an automatic consensus metric of image description quality. Consensus-based Image Description Evaluation (CIDEr) measures the similarity of a computer-generated sentence against a set of human-annotated sentences. It gives a higher score to the sentence that is more similar to the majority of human written descriptions.

- **ROUGE-L** (Lin, 2004), a recall-oriented evaluation metric popularly used in summarization community. It measures the number of in-sequence unigram matches between the generated sentence and sentences created by annotators.

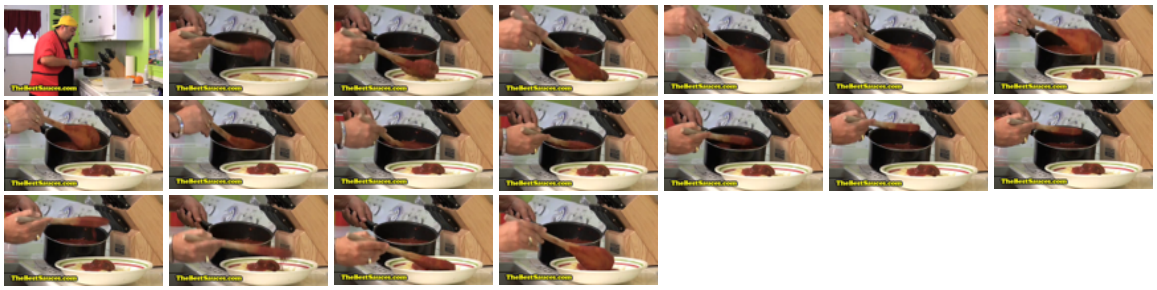
We use the caption evaluation package provided by the Microsoft COCO Image Captioning Challenge (Chen et al., 2015).

We compare our results to the results reported by the mean-pooling and sequence-to-sequence approaches reported in (Venugopalan et al., 2015a), and results from the CNN-RNN encoder-decoder model with a temporal attention mechanism reported in (Yao et al., 2015). However, the comparison to (Yao et al., 2015) is probably not a fair comparison, since we used different CNNs for feature extraction, e.g. they used GoogleNet and 3D-CNN but we used VGG16 and ResNet.

5.1 Results on the MSVD dataset

The results on the MSVD dataset are presented in Table 2 and the sample sentences are shown in Figure 2. The attention model plays an important role for both VGG16 and ResNet experiments. We can achieve the BLEU scores of 0.411 with VGG16 features and 0.437 with ResNet features. Our METEOR scores reached 0.310 when using VGG16 and 0.326 when using ResNet. However, in the experiment using VGG16 features, the *bilinear* alignment function seems to work best, while the *dot* alignment function gives the highest performance for the ResNet feature set.

As we can clearly see from the table, ResNet features are very powerful and can achieve the highest scores in all evaluation metrics.



VGG16
(non-attention) a man is adding sauce to a bowl
(dot) a man is adding sauce to a bowl
(bilinear) a man is pouring sauce into a bowl of chili
(concat) a man is pouring some sauce into a bowl
(sum) a man is pouring some sauce into a bowl

ResNet
(non-attention) a man is pouring sauce into a pot
(dot) a man is stirring a pot of food
(bilinear) a man is pouring sauce into a pot
(concat) a person is adding water to a pot
(sum) a man is stirring a pot

(ground truth)
(1) the man is pouring sauce over the pasta
(2) a man is putting food from pan to a plate
(3) a man is adding sauce to his spaghetti

Figure 2: Generated descriptions from MSVD dataset.

5.2 Results on the M-VAD dataset

The results on the M-VAD dataset are presented in Table 3. For this dataset, we did not perform the experiments using all the model configurations and feature types, due to the time constraints.

Even though the previous experiment on MSVD dataset showed that the result with ResNet were better than those with VGG16, we decided to use image features extracted from VGG16 to make a fair comparison with the previous work. From the table, our non-attention model can outperform the previous work in all evaluation metrics, but the attention model does not work well in this dataset. The reason probably comes from the characteristic of the M-VAD dataset that the videos contain a very high diversity of scenes and descriptions, so our attention models cannot be learned properly.

Some samples of generated sentences are shown in Figure 3.

6 Conclusion

In this paper, we have proposed a framework to automatically generate descriptions for video clips. We have applied the temporal attention mechanism to the sequence-to-sequence LSTM model. The results have proved that our model can generate high-quality short descriptions for videos, and can outperform the previous work. With the temporal attention mechanism, the model can learn to selectively focus on different parts of a video while generating each describing word.

For future work, we would like to use audio features or include a text-to-speech system in our framework since we think that audio is a very important piece of information for video understanding.

Acknowledgements

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO). We also would like to thank the anonymous reviewers for their insightful comments and suggestions, which were helpful in improving the quality of the paper.

VGG16
(non-attention) SOMEONE grabs a gun and the man steps out of the room .
(dot) SOMEONE and SOMEONE watch the men in their hands and the others .
(concat) He turns and walks off . SOMEONE follows SOMEONE to the floor , his eyes closed .
(sum) SOMEONE and SOMEONE step out of the room . SOMEONE and SOMEONE walk through the crowd .

(ground truth) SOMEONE appears and shoots SOMEONE in the leg . The mobster slips away . SOMEONE grabs SOMEONE .

Figure 3: Generated descriptions from M-VAD dataset.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*.

David L. Chen and William B. Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *ACL 2011*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *arXiv:1504.00325*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K. Srivastava, Li Deng, Piotr Dollar, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. 2015. From captions to visual concepts and back. In *CVPR 2015*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv:1512.03385*.

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

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR 2015*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980v8*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP 2015*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *ACL 2002*.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*.

Atousa Torabi, Christopher J. Pal, Hugo Larochelle, and Aaron C. Courville. 2015. Using descriptive video services to create a large data source for video annotation research. *arXiv:1503.01070*.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2014. Cider: Consensus-based image description evaluation. *arXiv:1411.5726v2*.

Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015a. Sequence to sequence – video to text. In *ICCV 2015*.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. 2015b. Translating videos to natural language using deep recurrent neural networks. In *NAACL-HLT 2015*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2014. Show and tell: A neural image caption generator. *arXiv:1411.4555*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *arXiv:1502.03044v3*.

Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Describing videos by exploiting temporal structure. In *ICCV 2015*.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. *arXiv:1603.03925*.

An Improved Phrase-based Approach to Annotating and Summarizing Student Course Responses

Wencan Luo[†] **Fei Liu**[‡] **Diane Litman**[†]
[†]University of Pittsburgh, Pittsburgh, PA 15260
[‡]University of Central Florida, Orlando, FL 32816
{wencan, litman}@cs.pitt.edu feiliu@cs.ucf.edu

Abstract

Teaching large classes remains a great challenge, primarily because it is difficult to attend to all the student needs in a timely manner. Automatic text summarization systems can be leveraged to summarize the student feedback, submitted immediately after each lecture, but it is left to be discovered what makes a good summary for student responses. In this work we explore a new methodology that effectively extracts summary phrases from the student responses. Each phrase is tagged with the number of students who raise the issue. The phrases are evaluated along two dimensions: with respect to text content, they should be informative and well-formed, measured by the ROUGE metric; additionally, they shall attend to the most pressing student needs, measured by a newly proposed metric. This work is enabled by a phrase-based annotation and highlighting scheme, which is new to the summarization task. The phrase-based framework allows us to summarize the student responses into a set of bullet points and present to the instructor promptly.

1 Introduction

Effective teachers use student feedback to adjust their teaching strategies. Nowadays, in large classes, there is far too much feedback for a single teacher to manage and attend to. If different perspectives in the student feedback could be summarized and pressing issues identified, it would greatly enhance the teachers' ability to make informed choices. In this work we seek to automatically summarize the student course feedback into a set of bullet points. Each bullet point corresponds to a phrase, tagged with the number of students who raise the issue. Our emphasis is on the textual feedback submitted by students after each lecture in response to two reflective prompts (Boud et al., 2013): 1) "Describe what you found most interesting in today's class" and 2) "Describe what was confusing or needed more detail." Education researchers have demonstrated that asking students to respond to reflection prompts can improve both teaching and learning (Van den Boom et al., 2004; Menekse et al., 2011). However, summarizing these responses for large classes (e.g., introductory STEM, MOOCs) remains costly, time-consuming, and an onerous task for humans (Mosteller, 1989).

In our prior work, **L**uo and **L**itman (2015) (henceforth **L&L**) introduced the task of automatic summarization of student responses. The challenges of this task include 1) high lexical variety, because students tend to use different word expressions to communicate the same or similar meanings (e.g., "bike elements" vs. "bicycle parts"), and 2) high length variety, as the student responses range from a single word to multiple sentences. To tackle the challenges, L&L proposed a *phrase summarization framework* consisting of three stages: phrase extraction, phrase clustering, and phrase ranking. The approach extracts noun phrases from student responses, groups the phrases using a greedy clustering algorithm, and finally selects representative phrases from the clusters using LexRank (Erkan and Radev, 2004).

There are three limitations in the phrase summarization framework. First, noun phrases do not suffice. Other types of phrases such as "how confidence intervals linked with previous topics" are useful and should be allowed. Second, clustering is based on similarity, but similarity of phrases that do not appear in a background corpus (i.e., the corpus used to learn the similarities) cannot be captured in the previous

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 53–63, Osaka, Japan, December 11-17 2016.

Reflective Prompt
Describe what was confusing or needed more detail.

Student Responses

S1: In the age of distributions example, application of qq plot [g] was confusing
S2: Last problem about normalization [m]
S3: central limit teorem [y] and A And B events example formulas were different. I did not understand that part well
S4: Sampling distribution [r] was a little bit abstract
S5: Q-q plot [g]
S6: Central Limit Thm [y]
S7: CLT [y]
S8: Normal approximation to binomial [b]
S9: bernaulli random variables
S10: The central limit [y] and normal approximations [b]
...

Human Summary 1

- central limit theorem [y] [12]
- q-q plot [g] [9]
- sampling distribution [r] [6]
- normal approximation [b] [5]
- normalization (last example) [m] [3]

Human Summary 2

- central limit theorem [13]
- q-q plots [9]
- general more explanations/details, better handwriting, move slower [9]
- sampling distributions [6]
- nothing [6]

Table 1: Example prompt, student responses, and two human summaries. 'S1'–'S10' are student IDs. The summary phrases are each tagged with the number of students who raise the issue (i.e., student supporters). The summary and phrase highlights are manually created by annotators. Phrases that bear the same color belong to the same issue. Each annotator is free to choose his/her color palette. We have only demonstrated the highlights of **Human Summary 1** to avoid overlaying of two sets of colors on student responses. The superscripts of the phrase highlights are imposed by the authors of this paper to differentiate colors when printed in grayscale (y: yellow, g: green, r: red, b: blue, and m: magenta).

setting. Lastly, a greedy clustering algorithm K-medoids (Kaufman and Rousseeuw, 1987) was previously used to group candidate phrases. It ignores global information and may suffer from a "collapsing" effect, which leads to the generation of a large cluster with unrelated items (Basu et al., 2013).

The goal of this work is to explore a phrase-based highlighting scheme, which is new to the summarization task. We aim to improve the phrase summarization framework by exploiting new capabilities that are enabled by the highlighting scheme. In the new scheme, human annotators are instructed to 1) create summary phrases from the student responses, 2) associate a number with each summary phrase which indicates the number of students who raise the issue (henceforth **student supporters**), and 3) highlight the corresponding phrases in both the human summary and student responses. Table 1 illustrates the highlighting scheme and more details are presented in §3. The new highlighting scheme makes it possible to develop a supervised candidate phrase extraction model (§4.1) and estimate pairwise phrase similarity with supervision (§4.2). To solve the third limitation, we explore a community detection algorithm OSLOM (Lancichinetti et al., 2011) that optimizes the statistical significance of clusters with respect to a global null model (§4.3). Experimental results show that the newly developed phrase extraction model is better than noun phrases only, in terms of both intrinsic and extrinsic measures; phrase similarity learning appears to produce marginal improvement; and the community detection approach yields better phrase summaries with more accurate estimation of the number of student supporters.

In summary, the contribution of this work is threefold.

- We introduce a new phrase-based highlighting scheme for automatic summarization, a departure from prior work. It highlights the phrases in the human summary and also the semantically similar phrases in student responses. We create a new dataset annotated with this highlighting scheme[1].
- We push the boundary of a phrase-based summarization framework by using our highlighting scheme to enable identification of candidate phrases as well as estimation of phrase similarities with supervision, and by using community detection to group phrases into clusters.
- We conduct comprehensive evaluations in terms of both summary text quality, measured by ROUGE (Lin, 2004), and how well phrase summaries capture the most pressing student needs, measured by a new evaluation metric based on color matching.

[1]This data set is publicly available at `http://www.coursemirror.com/download/dataset2`

2 Related Work

Work on automatic text summarization involves multiple granularities, ranging from keywords, phrases, to sentences. Traditional approaches have largely focused on sentence extraction (Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011; Li et al., 2013) and document abstraction (Liu et al., 2015; Rush et al., 2015; Durrett et al., 2016; Nallapati et al., 2016). In both cases, the produced summary is expected to be cohesive and coherent. We deviate from this path and seek to directly generate a set of bullet points as a summary. Phrases are easy to search and browse like words but more meaningful, and fit better on the small screen of a mobile device compared to sentences (Ueda et al., 2000; Luo et al., 2015).

Our task setting differs from those of keyphrase extraction (Wu et al., 2005; Liu et al., 2009; Medelyan et al., 2009; Hasan and Ng, 2014; Kan, 2015). Of key importance is that each summary phrase is associated with a numerical value, indicating the number of students who raise the issue. This information is critical to course instructors for making informed choices. Intuitively our task setting bears similarity to word/phrase cloud (Yatani et al., 2011; Brooks et al., 2014), where the cloud gives greater prominence to words or phrases that appear frequently in the source text. The downside is that they do not take lexical variety into account or considering semantically-equivalent words/phrases.

A summarization system is expected to produce high quality summary phrases and accurate estimates of the number of student supporters for each phrase. Luo and Litman (2015) focus on extracting noun phrases from student responses, however there lacks a comprehensive evaluation of the results, taking the number of student supporters into account. Other related work on student responses includes collecting student responses using a mobile application named CourseMIRROR (Luo et al., 2015; Fan et al., 2015), determining the quality of a student reflective response and providing feedback (Luo and Litman, 2016), and extracting informative sentences from the student feedback (Luo et al., 2016).

Traditional approaches to summary annotation have been based on either sentence extracts or document abstracts (Loza et al., 2014; Xiong and Litman, 2014; Wang and Ling, 2016). An effective linkage between the document content and human summary on the micro level have been largely absent. Barker et al.(2016) partially address this challenge by linking a summary back to a group of sentences that support the summary. However, this linkage is weak since it tells only that there is one sentence or more supporting the summary within the group, without explicitly telling which one(s). Approaches such as Pyramid (Nenkova and Passonneau, 2004) have exploited creating Summary Content Units (SCUs) to establish such links and alleviate the challenge. The new highlighting scheme described in this work holds promise for establishing direct links between the phrases in student responses and those in the human summary, allowing us to develop a new evaluation metric based on color matching.

3 New Data and Annotation

When reviewing the student feedback, we observe that not all issues are equally important. Some teaching problems are more prominent than others. Summary phrases should naturally reflect the number of students who raise the issue. But until now a reasonable sized dataset has been missing for this type of summarization setting. In this work we create a new dataset for this purpose. This allows us to develop a class of summarization approaches that learn to extract summary phrases from the student responses and estimate the number of student supporters for each summary phrase.

Our dataset consists of two statistics courses offered in a research university for industrial engineers. After each lecture, the students were asked to respond to two carefully designed reflection prompts using a mobile application named CourseMIRROR[2]: 1) "Describe what you found most interesting in today's class," and 2) "Describe what was confusing or needed more detail." For each course, two independent human annotators (native English speakers) with a statistics/mathematics background were recruited to create summaries for each lecture and prompt. The instructions we provide to the annotators include "*create a summary using 5 phrases and mark how many students semantically mentioned each phrase*." We limit the number of summary phrases to 5 per lecture and prompt in order to provide a concise summary to the instructor. Note that the summary phrases are not limited to extracts; while abstracts

[2] https://play.google.com/store/apps/details?id=edu.pitt.cs.mips.coursemirror

and fusion of phrases are also possible, they are rare. We further ask the annotators to "*highlight the corresponding phrases in the student responses which are semantically the same to the summary phrases using the same highlight colors*." The number of highlights in student responses should match the number of students who semantically mentioned the phrase. An example is illustrated in Table 1.

Note that L&L attempt to annotate the number of student supporters for summary phrases on a small dataset but without the highlighting scheme. We argue that the new highlighting scheme can provide many unique benefits. First, it allows us to track the "source phrases" that humans use to create the summary phrase. For example, the first summary phrase in Human Summary 1 of Table 1 is "central limit theorem." It is created from a collection of phrases in the student responses, including "The central limit", "central limit teorem" (a typo by the student), "CLT" (its abbreviation), and "Central Limit Thm" (another abbreviation). Naturally the highlighted source phrases lend themselves to a supervised approach to candidate phrase extraction. Second, the highlights inform us about the similarity and dissimilarity of phrases. For example, the source phrases that bear the same color are semantically similar to each other, whereas those with different colors are semantically dissimilar. In a similar vein, we develop a supervised approach that learns to predict the phrase similarity using highlights as guidance. Third, we are now able to accurately match the phrases in a system summary to those in a human summary, allowing the development of a novel summarization evaluation metric. For instance, assuming the system summary contains the phrase "Last problem about normalization" from S2 (Table 1), using the color highlights, we know that this phrase matches the human summary phrase "normalization (last example)." Such semantic matching between system and human summaries remains an elusive challenge for traditional summarization evaluation, but highlights make it an easy decision. Finally, the highlights on source texts indicate to what extent the information has been retained in the human summary. Specific to our task, we are interested to know the percentage of students whose responses are covered by the human summary. We define a student coverage score where a student is covered if and only if part of his/her response is highlighted. For example, in Table 1, S9 is considered not covered by Human Summary 1.

Basic statistics of the dataset are presented in Table 2.[3] The student coverage scores (75.9% for Course A and 82.4% for Course B) highlight the effectiveness of the current annotation scheme, with a majority of students covered by the human summaries.

Course	# Students	# Lectures	Averaged by Lecture/Prompt				
			# Responses	# Words	Words Per Res.	# Highlights	Student Coverage
A	66	11	34.1	156.5	4.5	27.8	75.9%
B	74	24	41.9	161.8	3.7	37.2	82.4%

Table 2: Basic statistics of the dataset. Because the student responses and human summaries are created for each lecture and prompt, we take the average of the corresponding statistics.

4 Improved Phrase Summarization

So far we have motivated the need for a new dataset with a highlighting scheme for phrase-based summarization. We proceed by describing three improvements to the phrase-based summmarization framework. Our first improvement involves a supervised approach to candidate phrase extraction (§4.1). Next, we learn to predict the pairwise phrase similarity (§4.2). Further, we explore a community detection algorithm to group the phrases into clusters (§4.3). We use the cluster size as an approximation to the number of student supporters for all the phrases within the cluster. L&L adopt LexRank (Erkan and Radev, 2004) to finally choose one representative phrase from each cluster. We follow the convention in this study. Note that our focus of this paper is not on developing new algorithms but to explore new capabilities that are enabled by the highlighting scheme. We thus perform direct comparisons with approaches described in L&L and leave comparisons to other approaches to future work. We present an intrinsic evaluation of each improvement in this section, followed by a comprehensive extrinsic evaluation in §5.

[3]While there are 22 lectures in total for Course A, unfortunately, only 11 of them have phrase highlighting.

4.1 Candidate Phrase Extraction

The phrase-based highlighting scheme lends itself to a supervised phrase extraction approach. In contrast, L&L used heuristics to extract noun phrases (NPs) only. This limitation has meant that informative non-NP phrases such as "how confidence intervals linked with previous topics" will be excluded from the summary, whereas uninformative NP phrases such as "the most interesting point" may be included.

We attempt to resolve this issue by formulating candidate phrase extraction as a word-level sequence labeling task. Concretely, we aim to assign a label to each word in the student responses. We choose to use the 'BIO' labeling scheme, where 'B' stands for the beginning of a phrase, 'I' for continuation of a phrase, 'O' for outside of a phrase. For example, "The (B) central (I) limit (I) and (O) normal (B) approximations (I)" illustrates the tagging of individual words, where the "The central limit" and "normal approximations" are two phrases highlighted by our annotators.

Local Features	• Word trigram within a 5-word window • Part-of-Speech tag trigram within a 5-word window • Chunk tag trigram within a 5-word window • Whether the word is in the prompt • Whether the word is a stopword • Label bigrams.
Global Features	• Total number of word occurrences (stemmed) • Rank of the word's term frequency

Table 3: Local and global features for supervised phrase extraction. Local features are extracted within one student's response. Global features are extracted using all student responses to a prompt in one lecture.

We choose to use the Conditional Random Fields (CRF) (Lafferty et al., 2001) as our sequence labeler[4] and develop a number of features (Table 3) based on sentence syntactic structure and word importance to signal the likelihood of a word being included in the candidate phrase. During training, we merge the phrase highlights produced by two annotators in order to form a large pool of training instances. When two highlights overlap completely, e.g., "normal approximations" are marked by both annotators using different colors, we keep only one instance of the phrase, resulting in 1,115 and 2,682 instances for Course A and Course B respectively. When the highlights partially overlap, we use each phrase highlight as a separate training instance. In this and all the following experiments, we perform leave-one-lecture-out cross validation on all the lectures and report results averaged across folds. Table 4 presents the intrinsic evaluation results on the phrase extraction task. We calculate Precision (P), Recall (R) and F-measure (F) scores based on the exact match of system phrases to gold-standard phrases. While the sequence labeling approach and the features presented here are straightforward, they do produce a collection of candidate phrases with higher precision. It removes noun phrases that are commonly used by students but uninformative (e.g., "a little bit abstract", "a problem with today's topic") as they were not highlighted by annotators. Phrase well-formedness is highly important to the summary quality, as evaluated in §5.

	Course A			Course B		
Candidate Phrase Extraction	P	R	F	P	R	F
L&L (NPs only)	0.426	**0.633**	0.503	0.538	0.714	0.609
Sequence Labeling with Highlights	**0.692***	0.569*	**0.618***	**0.771***	**0.743**	**0.753***

Table 4: Results of phrase extraction, intrinsically evaluated by comparing the system phrases to gold-standard phrases using exact match. The highest score in each column is shown in bold. $*$ means the difference is significant with $p < 0.05$.

4.2 Similarity Learning

Accurately estimating pairwise phrase similarity plays an essential role in phrase-based summarization. Better similarity learning helps produce better phrase clusters, which in turn leads to more accurate estimation of the number of student supporters for each summary phrase. While a human annotator

[4]We use the implementation of Wapiti (Lavergne et al., 2010) with default parameters.

could distinguish the semantic similarity or dissimilarity of the phrase highlights, it remains unclear if a single similarity metric could fulfill this goal or if we may need an ensemble of different metrics.

L&L calculate the pairwise phrase similarity using SEMILAR (Rus et al., 2013) with the latent semantic analysis (LSA) trained on the Touchstone corpus (Ştefănescu et al., 2014). One drawback of this approach is that the similarity of phrases that do not appear in a background corpus cannot be captured. In this work we develop an ensemble of similarity metrics by feeding them into a supervised classification framework. We use the phrase highlights as supervision, where phrases of the same color are positive examples and those of different colors are negative examples. We experiment with a range of metrics for measuring lexical similarity, including lexical overlap (Rus et al., 2013), cosine similarity, LIN similarity (Miller, 1995), BLEU (Papineni et al., 2002), SimSum (Lin, 2004), Word Embedding (Goldberg and Levy, 2014), and LSA (Deerwester et al., 1990). LIN similarity is based on WordNet definitions. Lexical overlap, cosine similarity, BLEU, and SimSum are related to how many words the two phrases have in common, while Word Embedding and LSA both capture the phrase similarity in a low dimensional semantic space. Therefore, we use an ensemble of the above similarity metrics by feeding them as features in a SVM classification model, assuming it will be better suited for this task than the LSA alone. Table 5 presents the intrinsic evaluation results. LSA has a poor degree of coverage (low recall) with many phrase similarities not being picked up by the metric.

	Course A			Course B		
Pairwise Phrase Similarity	P	R	F	P	R	F
L&L (LSA)	**0.904**	0.665	0.730	0.878	0.506	0.584
Similarity Learning with Highlights	0.895	**0.801***	**0.833***	**0.943***	**0.768***	**0.836***

Table 5: Results of predicting pairwise phrase similarity, measured using classification P/R/F.

4.3 Phrase Clustering

L&L use K-medoids for phrase clustering. It is a greedy iterative clustering algorithm (Kaufman and Rousseeuw, 1987), which may suffer from local minimal. We instead treat phrase clustering as a community detection problem. We define a **community** as a set of phrases that are semantically similar to each other, as compared to the rest of the phrases in student responses (Malliaros and Vazirgiannis, 2013). In our formulation, we consider each candidate phrase as a node in the network graph. We create an edge between two nodes if the two phrases are considered semantically similar to each other using the above similarity learning approach. Our goal is to identify tightly connected phrase communities in the network structure. The community size is used as a proxy for the number of students who semantically mention the phrase. Community detection has seen considerable success in tasks such as word sense disambiguation (Jurgens, 2011), medical query analysis (Campbell et al., 2014), and automatic summarization (Qazvinian and Radev, 2011; Mehdad et al., 2013).

Phrase Clustering	Course A	Course B
L&L (K-medoids)	82.2%	84.0%
Community Detection with OSLOM	**85.2%***	**88.8%***

Table 6: Results of phrase clustering measured by purity: ratio of number of phrases agreeing with the majority color in clusters.

We use OSLOM (Order Statistics Local Optimization Method, Lancichinetti et al., 2011) in this work. It is a widely used community detection algorithm that detects community structures (i.e., clusters of vertices) from a weighted, directed network. It optimizes locally the statistical significance of clusters with respect to a global null model during community expansion. We use an undirected version of OSLOM and set the p-value as 1.0 to encourage more communities to be identified[5] since the number of vertices in the constructed graph is relatively small compared to large complex networks. The key feature of OSLOM is that it supports finding overlapped community structures and orphaned vertices, offering more flexibility in the clustering process than K-medoids. We want to investigate if the unique characteristics of OSLOM allow it to produce better phrase clusters, hence more accurate estimation of

[5]L&L set the number of clusters is to be the square root of the number of extracted phrases.

the number of student supporters. We conduct an intrinsic evaluation using purity, corresponding to the percentage of phrases in the cluster that agree with the majority color. Results are presented in Table 6. While this metric by itself is not thorough enough, it does highlight the strength of the community detection approach in generating cohesive clusters. One advantage of OSLOM we found is that it will treat a phrase different from any other phrase as a singleton, while this phrase must be assigned to one of the clusters in K-medoids, resulting in a noisy cluster.

5 Summary Evaluation

The previous section described three improvements to the phrase summarization framework. Next, we evaluate them on the end task of summarizing student course responses. The phrase summaries are evaluated along two dimensions: we expect ROUGE (Lin, 2004) to measure the informativeness of the summary text content (§5.1); we further propose a new metric to quantify to what extent the most pressing student needs have been captured in the summary (§5.2).

5.1 ROUGE

ROUGE measures the n-gram overlap between system and human summaries. In this work, we report R-1, R-2, and R-SU4 scores, which respectively measure the overlap of unigrams, bigrams, and unigrams plus skip bigrams with a maximum distance of 4 words. These are metrics commonly used in the DUC and TAC competitions (Dang and Owczarzak, 2008). We implement the phrase summarization framework described in (Luo and Litman, 2015), named as **PhraseSum**. Further, we include **LexRank** (Erkan and Radev, 2004) as a competitive baseline. LexRank is a graph-based summarization approach based on eigenvector centrality. It has demonstrated highly competitive performance against the **PhraseSum** on a prior dataset (Luo and Litman, 2015). The summary is limited to 5 phrases or less in all experiments. Note that, the summary length is set independently of the number of clusters. If the number of clusters produced in §4.3 is less than 5, the phrase number is equal to the cluster number.

Course	System	R-1			R-2			R-SU4		
		P	R	F	P	R	F	P	R	F
A	LexRank	.276*	**.511**	.348*	.118*	**.245**	.154	.077*	**.260**	.106*
	PhraseSum	.402	.466	.415	.170	.208	.178	.162	.222	.160
	SequenceSum	.600*	.448	.493*	.307*	.231	.249*	.368*	.225	.244*
	SimSum	.597*	.460	**.504***	.302*	.241	.260*	.355*	.227	.249*
	CDSum	**.634***	.435	.499*	**.335***	.229	**.262***	**.404***	.210	**.250***
B	LexRank	.357*	**.560**	.429*	.187*	**.304***	.227	.129*	**.290**	.168*
	PhraseSum	.492	.545	.508	.231	.258	.239	.234	.283	.241
	SequenceSum	.618*	.485*	.531	.347*	.267	.294*	.385*	.238*	.274
	SimSum	.618*	.500*	.543	.353*	.284	.309*	.379*	.250	.285*
	CDSum	**.702***†	.480*	**.550***	**.433***†	.279	**.324***	**.500***†	.240*	**.293***

Table 7: Summarization Performance. **SequenceSum** means replacing the syntax phrase extraction in the PhraseSum baseline with the supervised sequence labeling phrase extraction. **SimSum** means replacing not only the phrase extraction but also the similarity scores using the supervised models. **CDSum** means using all three proposed techniques including the community detection. * indicates that the difference is statistically significant compared to PhraseSum with $p < 0.05$. † means that the improvement over SequenceSum is statistically significant with $p < 0.05$.

The summarization performance is shown in Table 7 (the caption explains the system names). The PhraseSum baseline, compared to LexRank, gets better P and F scores for all three ROUGE metrics for both courses, and the improvement of P is significant. This is the same as the findings in (Luo and Litman, 2015), and verifies our implementation of their model. For our enhancements of PhraseSum, the proposed supervised phrase extraction (SequenceSum) significantly improves P and thus improves (mostly significantly) F as well. SimSum is slightly better than SequenceSum for R and F, however, it is not significant using a two-tailed paired t-test. It suggests that a supervised method is not necessarily better than an unsupervised model in terms of the end-task performance, and its improvement over the

PhraseSum baseline is mainly due to the supervised phrase extraction step. In fact, the predicted similarity scores using the similarity learning model and the LSA model are highly correlated to each other ($r = 0.852$, $p < 0.01$) although it has a better classification performance (Table 5). Although CDSum is not significantly different from SequenceSum for the Course A, it does improve P significantly for all three ROUGE metrics for Course B. One possible explanation is that the latter course has a larger number of student responses, and thus benefits more from the community detection as the graph is larger.

5.2 A New Metric based on Color Matching

Our goal is to create a comprehensive evaluation metric that takes into account the following two factors.

- **Phrase matching.** While ROUGE is a classic summarization evaluation metric, it trivially compares the system vs. human summaries based on surface text form. In contrast, the phrase highlights allow us to accurately match the phrases in the system summary to those in the human summary based on color matching. This is due to two facts: first, our methods are extractive-based and all candidate phrases are extracted from the student responses; second, in the new highlighting scheme, the annotators are asked to highlight both the human summary phrase and any phrases in the student responses that are semantically the same with the summary phrase using the same color. It thus becomes easy to track the colors of the extracted phrases and verify if they match any of those in the human summary.
- **Student supporters.** Each summary phrase is tagged with the number of students who raise the issue. For human summary, this number is created by human annotators. For system summary, we approximate this number using the size of the cluster, from which the summary phrase is extracted.

Our proposed new metric resembles precision, recall, and F-measure. We define the true positive (TP) as the number of *shared colors* between system and human summaries. Each color is weighted by the number of student supporters, taken as the smaller value between system and human estimates. The *precision* is defined as TP over the total number of colors in the *system* summary, each weighted by system estimates; while *recall* is defined as TP over the total number of colors in the *human* summary, each weighted by human estimates. For example, assuming the phrases in the human summary are colored and tagged with estimates on student support: yellow/12, green/9, red/6, blue/5, magenta/3; similarly the phrases in the system summary are colored and tagged: yellow/11+3, green/17, red/7, blue/7. There are two phrases in the system summary that bear the same color, we thus add up the system estimates into yellow/11+3 (see Human Summary 1 in Table 1 and SequenceSum in Table 9). There are 4 shared colors between system and human summaries. The true positive is calculated as: $12+9+6+5 = 32$. The precision is $32/((11+3)+17+7+7) = 0.711$, and recall is $32/(12+9+6+5+3) = 0.914$. The F-measure is calculated as the harmonic mean of precision and recall scores.

The performance is shown in Table 8. Similar to the ROUGE evaluation, SequenceSum improves the P and F significantly. Now, CDSum not only significantly improves P, but also F for Course B.

	Course A			Course B		
	P	R	F	P	R	F
PhraseSum	.349	.615	.437	.485	.747	.576
SequenceSum	.626*	**.642**	**.614***	.698*	.757	.717*
SimSum	.602*	.636	.595*	.711*	.753	.723*
CDSum	**.643***	.634	.613*	**.777***†	**.762**	**.759***†

Table 8: Evaluation based on the new metric of color matching. P, R, and F are averaged by the annotators.

5.3 Example Summaries

The automatic summaries generated by different systems for the same example in Table 1 are shown in Table 9. The PhraseSum baseline extracts unnecessary content, which could be eliminated by the supervised phrase extraction model. For example, including "the example after" before "central limit theorem" makes it too specific. The "collapse" effect with a large cluster with unrelated items (Basu et al., 2013) can also be illustrated (e.g., the quantitative numbers for the phrase "i" in PhraseSum and

"q-q plot" in "SequenceSum" are much larger than the gold standard). This is solved by the community detection algorithm where such bigger clusters will not be considered as a single community.

PhraseSum	**SequenceSum**	**CDSum**
- i [40]	- q-q plot ^{g} [17]	- central limit theorem ^{y} [11]
- the example after central limit theorem ^{y} [12]	- central limit theorem ^{y} [11]	- q-q plot ^{g} [10]
- q q plot ^{g} [9]	- normal approximation to binomial ^{b} [7]	- sampling distributions ^{r} [7]
- the fact that we can sample as many as we want [9]	- sampling distributions ^{r} [7]	- normal approximation to binomial ^{b} [5]
- last problem about normalization ^{m} [6]	- clt ^{y} [3]	- nothing [4]

Table 9: Example system summaries for the example in Table 1. Note, the highlights in these summaries are NOT annotated by human after they are generated. Instead, they are automatically extracted from the dataset (§5.2).

6 Conclusion and Future Work

In this work, we introduce a new phrase-based highlighting scheme for automatic summarization. It highlights the phrases in the human summary and also the corresponding phrases in student responses. Enabled by the highlighting scheme, we improved the phrase-based summarization framework proposed by Luo and Litman (2015) by developing a supervised candidate phrase extraction, learning to estimate the phrase similarities, and experimenting with different clustering algorithms to group phrases into clusters. We further introduced a new metric that offers a promising direction for making progress on developing automatic summarization evaluation metrics. Experimental results show that our proposed methods not only yield better summarization performance evaluated using ROUGE, but also produce summaries that capture the pressing student needs. Future work includes thorough comparison with other approaches and extending the current research to multiple courses and other summary lengths in order to test the generalizability. We also plan to supplement our ROUGE scores with human evaluations of system summaries.

Acknowledgements

This research is supported by an internal grant from the Learning Research and Development Center at the University of Pittsburgh. We thank Jingtao Wang and Xiangmin Fan for developing the CourseMIR-ROR mobile system. We thank Fan Zhang and Huy Nguyen for valuable suggestions about the proposed summarization algorithm. We also thank anonymous reviewers for insightful comments and suggestions.

References

Emma Barker, Monica Lestari Paramita, Ahmet Aker, Emina Kurtic, Mark Hepple, and Robert Gaizauskas. 2016. The sensei annotated corpus: Human summaries of reader comment conversations in on-line news. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 42–52, Los Angeles, September. Association for Computational Linguistics.

Sumit Basu, Chuck Jacobs, and Lucy Vanderwende. 2013. Powergrading: a clustering approach to amplify human effort for short answer grading. *Transactions of the Association for Computational Linguistics*, 1:391–402.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of ACL*, pages 481–490, Portland, Oregon, USA.

David Boud, Rosemary Keogh, David Walker, et al. 2013. *Reflection: Turning experience into learning*. Routledge.

Bill J Brooks, Debra M Gilbuena, Stephen Krause, and Milo D Koretsky. 2014. Using word clouds for fast, formative assessment of students' short written responses. *Chemical Engineering Education*, 48(4):190–198.

William Campbell, Elisabeth Baseman, and Kara Greenfield. 2014. Content+context=classification: Examining the roles of social interactions and linguist content in Twitter user classification. In *Proceedings of the Second Workshop on Natural Language Processing for Social Media*, pages 59–65, Dublin, Ireland.

Dan Ştefănescu, Rajendra Banjade, and Vasile Rus. 2014. Latent semantic analysis models on wikipedia and tasa. In *Proceedings of LREC*, pages 26–31, Reykjavik, Iceland.

Hoa Trang Dang and Karolina Owczarzak. 2008. Overview of the TAC 2008 update summarization task. In *Proceedings of TAC*, pages 1–16.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of ACL*, pages 1998–2008, Berlin, Germany.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22(1):457–479.

Xiangmin Fan, Wencan Luo, Muhsin Menekse, Diane Litman, and Jingtao Wang. 2015. CourseMIRROR: Enhancing large classroom instructor-student interactions via mobile interfaces and natural language processing. In *Works-In-Progress of ACM Conference on Human Factors in Computing Systems*. ACM.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: Deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of ACL*, pages 1262–1273, Baltimore, Maryland.

David Jurgens. 2011. Word sense induction by community detection. In *Proceedings of TextGraphs-6 Workshop*, pages 24–28, Portland, Oregon.

Min-Yen Kan. 2015. Keywords, phrases, clauses and sentences: topicality, indicativeness and informativeness at scales. In *Proceedings of the ACL 2015 Workshop on Novel Computational Approaches to Keyphrase Extraction*, page 1, Beijing, China.

Leonard Kaufman and Peter Rousseeuw. 1987. Clustering by means of medoids. *Statistical Data Analysis Based on the $L1$-Norm and Related Method*, pages 405–416.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289, San Francisco, CA, USA.

Andrea Lancichinetti, Filippo Radicchi, José J Ramasco, and Santo Fortunato. 2011. Finding statistically significant communities in networks. *PloS one*, 6(4):e18961.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 504–513. Association for Computational Linguistics, July.

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document summarization via guided sentence compression. In *Proceedings of EMNLP*, pages 490–500, Seattle, Washington, USA.

Chin-Yew Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out*, volume 8. Barcelona, Spain.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of EMNLP*, pages 257–266, Stroudsburg, PA, USA.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of NAACL*, pages 1077–1086, Denver, Colorado.

Vanessa Loza, Shibamouli Lahiri, Rada Mihalcea, and Po-Hsiang Lai. 2014. Building a dataset for summarization and keyword extraction from emails. In *Proceedings of LREC*, pages 2441–2446, Reykjavik, Iceland.

Wencan Luo and Diane Litman. 2015. Summarizing student responses to reflection prompts. In *Proceedings of EMNLP*, pages 1955–1960, Lisbon, Portugal.

Wencan Luo and Diane Litman. 2016. Determining the quality of a student reflective response. In *Proceedings 29th International FLAIRS Conference*, Key Largo, FL.

Wencan Luo, Xiangmin Fan, Muhsin Menekse, Jingtao Wang, and Diane Litman. 2015. Enhancing instructor-student and student-student interactions with mobile interfaces and summarization. In *Proceedings of NAACL: Demonstrations*, pages 16–20, Denver, Colorado.

Wencan Luo, Fei Liu, Zitao Liu, and Diane Litman. 2016. Automatic summarization of student course feedback. In *Proceedings of NAACL*, pages 80–85, San Diego, California.

Fragkiskos D. Malliaros and Michalis Vazirgiannis. 2013. Clustering and community detection in directed networks: A survey. *CoRR*, abs/1308.0971.

Andre Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for NLP*, pages 1–9, Boulder, Colorado.

Olena Medelyan, Eibe Frank, and Ian H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of EMNLP*, pages 1318–1327, Stroudsburg, PA, USA.

Yashar Mehdad, Giuseppe Carenini, Frank Tompa, and Raymond T. NG. 2013. Abstractive meeting summarization with entailment and fusion. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 136–146, Sofia, Bulgaria.

Muhsin Menekse, Glenda Stump, Stephen J. Krause, and Michelene T.H. Chi. 2011. The effectiveness of students daily reflections on learning in engineering context. In *Proceedings of the American Society for Engineering Education Annual Conference*, Vancouver, Canada.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Frederick Mosteller. 1989. The 'muddiest point in the lecture' as a feedback device. *On Teaching and Learning: The Journal of the Harvard-Danforth Center*, 3:10–21.

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016. Sequence-to-sequence rnns for text summarization. *CoRR*, abs/1602.06023.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *Proceedings of NAACL*, pages 145–152, Boston, Massachusetts, USA.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BlEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadelphia, Pennsylvania, USA.

Vahed Qazvinian and Dragomir R. Radev. 2011. Learning from collective human behavior to introduce diversity in lexical choice. In *Proceedings of ACL*, pages 1098–1108, Portland, Oregon, USA.

Vasile Rus, Mihai Lintean, Rajendra Banjade, Nobal Niraula, and Dan Stefanescu. 2013. Semilar: The semantic similarity toolkit. In *Proceedings of ACL: System Demonstrations*, pages 163–168, Sofia, Bulgaria.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*, pages 379–389, Lisbon, Portugal.

Yoshihiro Ueda, Mamiko Oka, Takahiro Koyama, and Tadanobu Miyauchi. 2000. Toward the "at-a-glance" summary: Phrase-representation summarization method. In *Proceedings of COLING*, pages 878–884, Stroudsburg, PA, USA.

Gerard Van den Boom, Fred Paas, Jeroen JG Van Merrienboer, and Tamara Van Gog. 2004. Reflection prompts and tutor feedback in a web-based learning environment: effects on students' self-regulated learning competence. *Computers in Human Behavior*, 20(4):551 – 567.

Lu Wang and Wang Ling. 2016. Neural network-based abstract generation for opinions and arguments. In *Proceedings of NAACL*, pages 47–57, San Diego, California.

Yi-fang Brook Wu, Quanzhi Li, Razvan Stefan Bot, and Xin Chen. 2005. Domain-specific keyphrase extraction. In *Proceedings of CIKM*, pages 283–284, New York, NY, USA.

Wenting Xiong and Diane Litman. 2014. Empirical analysis of exploiting review helpfulness for extractive summarization of online reviews. In *Proceedings of COLING*, pages 1985–1995, Dublin, Ireland.

Koji Yatani, Michael Novati, Andrew Trusty, and Khai N. Truong. 2011. Review Spotlight: A user interface for summarizing user-generated reviews using adjective-noun word pairs. In *Proceedings of CHI*, pages 1541–1550, New York, NY, USA.

CATENA: CAusal and TEmporal relation extraction from NAtural language texts

Paramita Mirza
Max Planck Institute for Informatics
Saarland Informatics Campus, Germany
paramita@mpi-inf.mpg.de

Sara Tonelli
Fondazione Bruno Kessler
Trento, Italy
satonelli@fbk.eu

Abstract

We present CATENA, a sieve-based system to perform temporal and causal relation extraction and classification from English texts, exploiting the interaction between the temporal and the causal model. We evaluate the performance of each sieve, showing that the rule-based, the machine-learned and the reasoning components all contribute to achieving state-of-the-art performance on TempEval-3 and TimeBank-Dense data. Although causal relations are much sparser than temporal ones, the architecture and the selected features are mostly suitable to serve both tasks. The effects of the interaction between the temporal and the causal components, although limited, yield promising results and confirm the tight connection between the temporal and the causal dimension of texts.

1 Introduction

When the Greek government missed its 1.6 billion euro payment to the IMF as its bailout expired on 30 June 2015, people started to look for information, such as *What is going on? Why did it happen and what will happen next?* A compact summary that represents the development of a story over time, highlighting not only the temporal connections between events but also cause-effect chains, would be very beneficial for providing information that the readers need. Besides, this kind of knowledge, derived from structured information about events and their temporal-causal relations, could be used in a number of applications, from tools for automated generation of timelines to question answering and decision support systems.

While temporal relation classification is a well-studied task with a number of systems participating in the TempEval campaigns (Verhagen et al., 2010; UzZaman et al., 2013; Llorens et al., 2015), less attention has been devoted by the NLP community to the detection of causal links between events. Although recent attempts have tried to settle an annotation standard for causality inspired by TimeML (Mirza et al., 2014), the interactions between the temporal and the causal dimension of texts have been scarcely explored, especially from an empirical point of view. In this work, we face this challenge by presenting CATENA (CAusal and TEmporal relation extraction from NAtural language texts),[1] a multi-sieve architecture for the extraction and classification of both relation types from English documents, which are pre-annotated with *temporal entities*, namely *events* and *time expressions*.

2 Related Work

Our proposed approach for relation extraction is inspired by recent works on hybrid approaches for temporal relation extraction (D'Souza and Ng, 2013; Chambers et al., 2014). D'Souza and Ng (2013) introduce 437 hand-coded rules along with supervised classification models using lexical relation, semantic and discourse features. CAEVO, a CAscading EVent Ordering architecture by Chambers et al. (2014), combines rule-based and data-driven classifiers in a *sieve-based architecture* for temporal ordering. The classifiers (sieves) are ordered by their individual precision, and transitive closure is applied after each sieve to ensure consistent temporal graph.

[1] The system is made available at https://github.com/paramitamirza/CATENA.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 64–75, Osaka, Japan, December 11-17 2016.

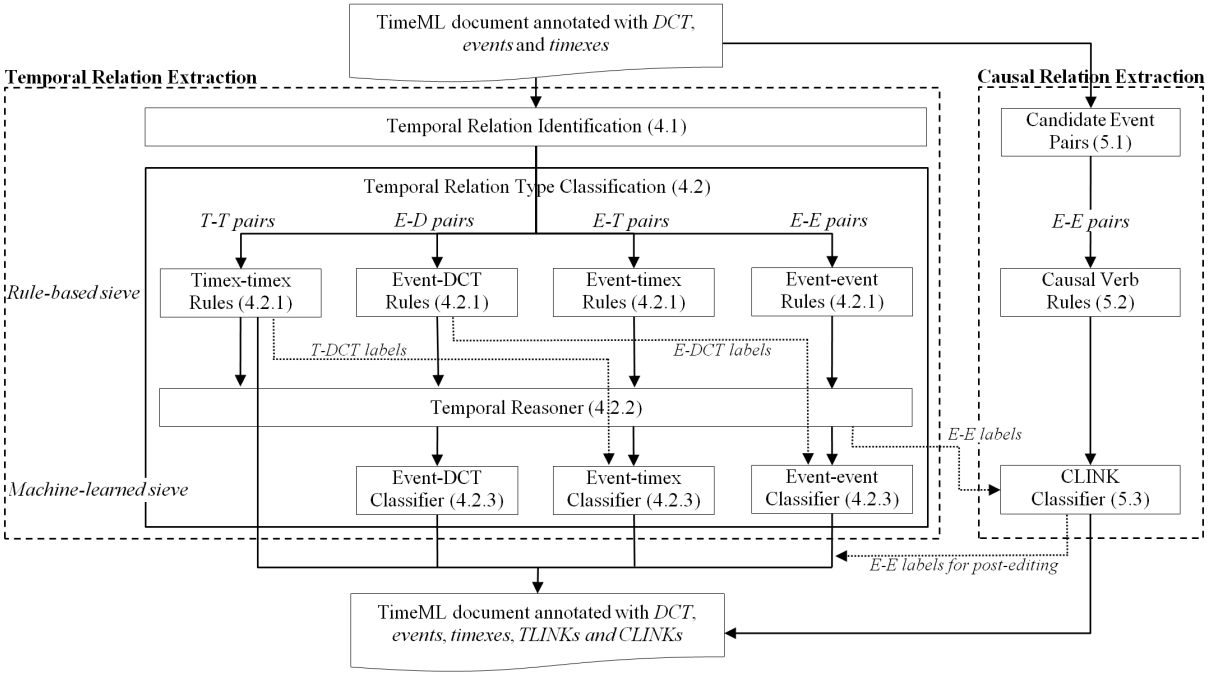


Figure 1: System architecture of CATENA

The problem of detecting causality between events is as challenging as recognizing their temporal order, but less analyzed from an NLP perspective. Besides, previous works have mostly focused on specific types of event pairs and causal expressions in text (Bethard et al., 2008; Do et al., 2011; Riaz and Girju, 2013). Several works, relying on corpus of parallel temporal and causal relations developed with specific connectives in mind (Bethard et al., 2008), have presented analyses on the interaction between temporal and causal relations (Bethard and Martin, 2008; Rink et al., 2010). Exploiting gold temporal labels as features for the causal relation classifier is shown to be beneficial. Mirza et al. (2014) presented some annotation guidelines to capture explicit causality between event pairs, inspired by TimeML. The resulting corpus, Causal-TimeBank, is then used to build supervised classification models for extracting causal relations (Mirza and Tonelli, 2014a). None of the above systems presents a hybrid approach in a sieve-based architecture to deal with this task. CATENA is at present the first integrated system available performing temporal and causal relation extraction.

3 System architecture

The CATENA system includes two main classification modules, one for temporal and the other for causal relations between events. As shown in Figure 1, they both take as input a document annotated with the so-called temporal entities according to TimeML guidelines (Pustejovsky et al., 2003), including the document creation time (DCT), events and time expressions (timexes). The output is the same document with temporal links (TLINKs) set between pairs of temporal entities, each assigned to one of the TimeML temporal relation types, such as BEFORE, INCLUDES or SIMULTANEOUS, which denotes the temporal ordering. The document is also annotated with causal relations (CLINKs) between event pairs.

The modules for temporal and causal relation classification rely both on a sieve-based architecture, in which the remaining unlabelled pairs – after running a rule-based component and/or a transitive reasoner – are fed into a supervised classifier. Although some steps can be run in parallel, the two modules interact, based on the assumption that the notion of causality is tightly connected with the temporal dimension and that information from one module can be used to improve or check the consistency of the other. In particular, (i) TLINK labels for event-event (E-E) pairs, resulting from the rule-based sieve + temporal reasoner modules, are used as features for the CLINK classifier; and (ii) CLINK labels (i.e. CLINK and CLINK-R) are used as a post-editing method for correcting the wrong labelled event pairs by the TLINK

classifier. This step relies on a set of rules based on the temporal constraint of causality, i.e. (i) CLINK(e_1, e_2) → BEFORE(e_1, e_2) and (ii) CLINK-R(e_1, e_2) → AFTER(e_1, e_2). The modules for temporal and causal relation extraction are detailed in Section 4 and 5 respectively.

4 Temporal Relation Extraction System

The module for the extraction of temporal relations contains two main components, one for (i) *temporal relation identification*, which is based on a set of rules, and the other for (ii) *temporal relation type classification*, which is a combination of rule-based and supervised classification modules, with a temporal reasoning component in between. The three steps for temporal relation type classification are ordered based on their individual precisions. This mechanism allows the system to first label few links with high precision using rules, then to infer new links through the reasoner, and finally to increase recall through supervised classification, based on the output of the previous steps.

4.1 Temporal Relation Identification

All pairs of temporal entities satisfying one of the following rules, inspired by the TempEval-3 task description, are considered as having temporal links (TLINKs): (i) two main events of consecutive sentences, (ii) two events in the same sentence, (iii) an event and a timex in the same sentence, (iv) an event and a document creation time and (v) pairs of all possible timexes (including document creation time) linked with each other.[2] These pairs are then grouped together into four different groups: *timex-timex* (T-T), *event-DCT* (E-D), *event-timex* (E-T) and *event-event* (E-E).

4.2 Temporal Relation Type Classification

Our *sieve-based architecture* is inspired by CAEVO (Chambers et al., 2014), although we significantly reduce the system complexity as follows:

- We merge all rule-based classifiers into one sieve component (rule-based sieve), and all Support Vector Machine (SVM) classifiers in the machine-learned sieve.
- Instead of running transitive inference after each classifier, we run our *temporal reasoner* module on the output of the rule-based sieve, only once.

Furthermore, we use the output of the rule-based sieve (Section 4.2.1) as features for the machine-learned sieve (Section 4.2.3), specifically: (i) the timex-DCT link label proposed by the *timex-timex rules* are used as a feature in the *event-timex SVM*, and (ii) the event-DCT link label proposed by the *event-DCT rules* are used as a feature in the *event-event SVM*.

4.2.1 Temporal Rule-Based Sieve

The temporal rule-based sieve relies on specific hand-crafted rules designed for each type of temporal entity pairs, and takes as input the entity pairs identified in the previous step.

Timex-timex Rules For timex-timex relations, we take into account temporal expressions of types DATE and TIME, and determine the relation types based on their *normalized values*. For example, *"7 PM tonight"* (2015-12-12T19:00) IS_INCLUDED in *"today"* (2015-12-12).

Event-DCT Rules The rules for labelling E-D pairs are based on the *tense* and/or *aspect* of the event word. For example, for the event mention *"(had) fallen"*, which is in the past tense with perfective aspect, its relation with the DCT is labelled as BEFORE.

Event-timex Rules As for E-T pairs, we build a set of rules based on the temporal senses of some prepositions (Litkowski and Hargraves, 2006; Litkowski, 2014).[3] In particular we assign a label whenever a temporal preposition establishes a dependency path between an event (E) and a timex (T), in which T acts as the *temporal modifier* of E. For example, if T is introduced by a temporal prepositions expressing a STARTTIME sense such as *from* or *since*, the relation is labelled as BEGUN_BY.

[2]Note that this is not included in the enumerated possible TLINKs in the TempEval-3 task description.

[3]We took the list of temporal prepositions from http://www.clres.com/db/classes/ClassTemporal.php.

In the absence of a temporal preposition, T might simply be a temporal modifier of E, as exemplified in *"Police [confirmed]* $_{\text{E}}$ *[Friday]* $_{\text{T}}$ *that the body was found..."*. In this case, we assume that the E-T label is IS_INCLUDED. Moreover, sometimes events are modified by temporal expressions marking the starting time and ending time in a *duration pattern* such as '*between* TBEGIN *and* TEND' or '*from* TBEGIN *to/until* TEND'. We define additional rules as follows: (i) If T matches TBEGIN then E-T label is BEGUN_BY, and (ii) if T matches TEND then E-T label is ENDED_BY.

Event-event Rules E-E pairs are finally labelled following two sets of rules. The first set is based on the dependency path possibly existing between the first (e_1) and the second event (e_2), and the verb information encoded in e_1. For example, if e_2 is the *logical subject* of e_1 as in *"...the chain reaction [touched]* $_{e_1}$ *off by the [collapse]* $_{e_2}$ *of Lehman Brothers"*, e_1 and e_2 are connected by an AFTER relation.

The other set of rules is taken from CAEVO, including: (i) rules for linking a reporting event and another event syntactically dominated by the first, based on *tense* and *aspect*; and (ii) rules based on the role played by various tenses of English verbs in conveying temporal discourse (Reichenbach, 1947).

Further details on the implemented rules for the temporal rule-based sieve can be found in Appendix A.

4.2.2 Temporal Reasoner

Based on the output of the previous sieve, we run a transitive reasoner layer, similar to CAEVO, in order to infer new temporal links among candidate pairs. This alleviates the issue of high precision and low recall, typical of the rule-based sieve.

An annotated TimeML document can be mapped into a constraint problem according to how TLINKs are mapped into Allen relations (Allen, 1983). We apply the following mapping:

- $<$ and $>$ for BEFORE and AFTER
- o and o^{-1} for DURING and DURING_INV
- d and d^{-1} for IS_INCLUDED and INCLUDES
- s and s^{-1} for BEGINS and BEGUN_BY
- f and f^{-1} for ENDS and ENDED_BY

Once the documents are mapped into constraint problems, they are then processed by an automated temporal reasoner for computing their deductive closure, globally reasoning on them. We rely on the *Generic Qualitative Reasoner* (GQR) (Westphal et al., 2010), a fast solver for generic qualitative constraint problems, such as Allen constraint problems. The rationale of preferring GQR to other solutions, such as fast *Boolean Satisfiability Problem* (SAT) solvers, is due to its scalability, simplicity of use and efficient performances (Westphal and Wölfl, 2009).

4.2.3 Temporal Supervised Classifiers

We build three supervised classification models, one for event-DCT (E-D), one for event-timex (E-T) and one for event-event (E-E) pairs. We use LIBLINEAR (Fan et al., 2008) L2-loss linear SVM (default parameters), and one-vs-rest strategy for multi-class classification.

Tools and Resources Several external tools and resources are used to extract features from each temporal entity pair, including:

- *MorphoPro* (Pianta et al., 2008), to get PoS tags and phrase chunk for each token.
- *Mate tools* (Bjorkelund et al., 2010) to extract the dependency path between words.
- *WordNet similarity module*[4] to compute semantic similarity (Lin, 1998) between words.
- *Temporal signal lists* from Mirza and Tonelli (2014b), further expanded using the Paraphrase Database (Ganitkevitch et al., 2013), and manually clustered e.g. {*before*, *prior to*, *in advance of*}.

Feature Set We implemented a set of features, listed in Table 1, largely inspired by the best performing systems in TempEval-2 (Verhagen et al., 2010) and TempEval-3 (UzZaman et al., 2013) campaigns. We simplified the possible values of some features as follows:

[4] `http://ws4jdemo.appspot.com/`

Feature	TLINK E-D	TLINK E-T	TLINK E-E	CLINK E-E	Rep.	Description
Morphosyntactic information						
PoS	x	x	x	x	one-hot	Part-of-speech tags of e_1 and e_2.
phraseChunk	x	x	x	x	one-hot	Shallow phrase chunk of e_1 and e_2.
samePoS		x	x	x	binary	Whether e_1 and e_2 have the same PoS.
Textual context						
entityOrder		x			binary	Appearance order of e_1 and e_2 in the text.[5]
sentenceDistance		x	x	x	binary	`0` if e_1 and e_2 are in the same sentence, `1` otherwise.
entityDistance		x	x	x	binary	`0` if e_1 and e_2 are adjacent, `1` otherwise.
EVENT attributes						
class	x	x	x	x	one-hot	EVENT attributes as specified in TimeML.
tense	x	x	x	x	one-hot	
aspect	x	x	x	x	one-hot	
polarity	x	x	x	x	one-hot	
sameClass			x	x	binary	Whether e_1 and e_2 have the same EVENT attributes.
sameTenseAspect			x	x	binary	
samePolarity			x	x	binary	
TIMEX3 attributes						
type	x	x			one-hot	TIMEX3 attributes as specified in TimeML.
Dependency information						
dependencyPath			x	x	one-hot	Dependency path between e_1 and e_2.
isMainVerb	x	x	x	x	binary	Whether e_1/e_2 is the main verb of the sentence.
Temporal signals						
tempSignalTokens		x	x	x	one-hot	Tokens (cluster) of temporal signal around e_1 and e_2.
tempSignalPosition		x	x	x	one-hot	Temporal signal position w.r.t e_1/e_2 (BETWEEN, BEFORE, BEGIN, etc.)
tempSignalDependency		x	x	x	one-hot	Temporal signal dependency path between signal tokens and e_1/e_2.
Causal signals						
causSignalTokens				x	one-hot	Tokens (cluster) of causal signal around e_1 and e_2.
causSignalPosition				x	one-hot	Causal signal position w.r.t e_1/e_2 (BETWEEN, BEFORE, BEGIN, etc.)
causSignalDependency			x	x	one-hot	Causal signal dependency path between signal tokens and e_1/e_2.
Lexical semantic information						
wnSim			x	x	one-hot	WordNet similarity computed between the lemmas of e_1 and e_2.
TLINK labels from the rule-based sieve						
timex-DCT label		x			one-hot	The TLINK type of the e_2 (timex) and DCT pair (if any).
event-DCT label			x		one-hot	The TLINK types of the e_1/e_2 and DCT pairs (if any).

Table 1: Feature sets for TLINK classification of event-DCT (E-D), event-timex (E-T) and event-event (E-E) pairs, and for CLINK classifier (E-E pairs), with corresponding feature representation (Rep).

- *dependencyPath* We only consider a dependency path between an event pair if it describes coordination, subordination, subject or object relation.
- *signalTokens* Given a temporal signal, we do not include in the feature set the token but the *clusterID* of the cluster containing synonymous signals, e.g. {*before*, *prior to*, *in advance of*}.
- *wnSim* The value of WordNet similarity measure is discretized as follows: $sim \leq 0.0$, $0.0 < sim \leq 0.5$, $0.5 < sim \leq 1.0$ and $sim > 1.0$.

We exclude lexical features such as *token/lemma* of temporal entities from the feature set in order to increase the classifiers' robustness in dealing with completely new texts with different vocabularies. Instead, we include *WordNet similarity* in the feature set to capture the semantic relations between event words.

Label Simplification For training the classification models, we only consider 10 out of the 14 relation types defined in TimeML by collapsing some types, i.e., IBEFORE into BEFORE, IAFTER into AFTER, DURING and DURING_INV into SIMULTANEOUS, due to the sparse annotation of such labels in the datasets.

5 Causal Relation Extraction System

We propose the same hybrid approach combining rule-based and supervised classifiers for the identification of causal relations. However, while temporal order has a clear formalization in the NLP community, capturing causal relationships in natural language text is more challenging, for they can be expressed by different syntactic and semantic features and involve both situation-specific information and world knowledge. We adopt the notion of causality proposed in the annotation guidelines of the Causal-TimeBank (Mirza et al., 2014; Mirza and Tonelli, 2014a), which accounts for CAUSE, ENABLE and

[5]The order of e_1 and e_2 in E-E pairs is always according to the appearance order in the text, while in E-T pairs, e_2 is always a timex regardless of the appearance order.

PREVENT phenomena (Wolff, 2007; Wolff and Song, 2003) that are overtly expressed in text. In particular, we aim at assigning a causal link to pairs of events when: (i) the causal relation is expressed by *affect*, *link* and *causative verbs* (CAUSE-, ENABLE- and PREVENT-type verbs), hereinafter simply addressed as *causal verbs*; or (ii) the causal relation is marked by a *causal signal* (see e.g. footnote 6).

The two cases require different algorithms: while causal constructions containing causal verbs are quite straightforward to identify, causal signals are very ambiguous and can appear in different syntactic constructions.[6] Therefore, we tackle the first through a rule-based approach, while the second is best covered via supervision, taking advantage of the freely available Causal-TimeBank.

5.1 Causal Relation Identification

Similar to the temporal processing module, the first step towards causal relation classification is the identification of candidate event pairs. Given a document already annotated with events, we take into account every possible combination of events in a sentence in a forward manner as *candidate event pairs*. For example, if we have a sentence "e_1, triggered by e_2, cause them to e_3," the candidate event pairs are (e_1,e_2), (e_1,e_3) and (e_2,e_3). We also include as candidate event pairs the combination of each event in a sentence with events in the following one, to account for inter-sentential causality, under the simplifying assumption that causality may be expressed also between events in two consecutive sentences.

5.2 Causal Rule-Based Sieve

In the rule-based sieve, we classify causal constructions containing causal verbs. These show strong regularities: given a causal verb v, the first event e_1 is usually the *subject* of v and the second event e_2 is either the *object* or the *predicative complement* of v. Such relations between events and causal verbs are usually syntactically expressed, therefore our rules aim at identifying pairs of events being related to a causal verb in a causal construction by looking at their dependency paths.

We take the list of 56 affect, link and causative verbs presented in Mirza et al. (2014) as the causal verb list. We further expand the list using the Paraphrase Database (Ganitkevitch et al., 2013) and original verbs as seeds, resulting in a total of 97 verbs. We then manually cluster the causal verbs sharing the same syntactic behaviour in groups and define a set of rules for each verb group, taking into account the possible existing dependency paths between v and e_1/e_2, as well as the *causal direction sense*[7] conveyed in v. Further details on the implemented rules for the causal rule-based sieve can be found in Appendix B.

5.3 Causal Supervised Classifier

In order to recognize and determine the causal direction of CLINKs that are signalled by a causal signal, we adopt a supervised approach. We build a classification model using LIBLINEAR (Fan et al., 2008) L2-loss linear SVM (default parameters), and one-vs-rest strategy for multi-class classification. The classifier has to label an event pair (e_1, e_2) with CLINK, CLINK-R or O for others.

We take as candidate event pairs only those in which the causal signal is connected via dependency path to either e_1 or e_2, or both. Besides, we exclude event pairs where the two events are directly connected through relations such as subject, object, coordinating or locative adverbial, because a causal relation usually does not hold in these cases.

Tools and Resources The same external tools and resources mentioned in Section 4.2.3 for building the temporal classifiers are used to extract features from each event pair. Additionally, we take the list of causal signals from the annotation guidelines presented in Mirza et al. (2014) as the *causal signal list*. Again we expand the list using the Paraphrase Database (Ganitkevitch et al., 2013), resulting in a total of 200 signals. We also manually cluster some signals together, e.g. {*therefore*, *thereby*, *hence*, *consequently*}, as we did for temporal signals.

[6] *"The building [collapsed]* $_{\text{T}}$ ***because of*** *the [earthquake]* $_{\text{S}}$*"* vs *"**Because of** the [earthquake]* $_{\text{S}}$ *the building [collapsed]* $_{\text{T}}$*"*. S and T denote the *source* (cause) and *target* (effect) of the causal relation.

[7] For example, *result in* and *result from* have different senses affecting the causal direction, i.e. the causing event is the subject of *result in* and the object of *result from*.

Feature Set The implemented features are listed in Table 1. As shown in Figure 1, the event-event labels added by the rule-based sieve and the reasoner in the temporal relation extraction module are also used as features for the CLINK classifier.

6 Evaluation

The purpose of the evaluation is two-fold: (i) to evaluate the quality of extracted temporal and causal links separately; and (ii) to investigate the interaction between temporal and causal relation extraction systems in the integrated architecture.

6.1 Temporal and Causal Relation evaluation

We perform two evaluations, one following *TempEval-3* and the other *TimeBank-Dense* evaluation methodology.

Dataset For the evaluation of the temporal relation extraction module following TempEval-3, we use the same training and test data released for the shared task,[8] i.e. *TBAQ-cleaned* (cleaned and improved version of the TimeBank 1.2 and the AQUAINT corpora) and *TempEval-3-platinum*, respectively. The TimeBank 1.2 corpus contains 183 documents coming from a variety of news report, specifically from the ACE program and PropBank, while the AQUAINT corpus contains 73 news report documents and often referred to as the *Opinion corpus*. The TempEval-3-platinum corpus, containing 20 news articles, was annotated/reviewed by the TempEval-3 organizers.

The *TimeBank-Dense* corpus (Chambers et al., 2014) is created to address the sparsity issue in the existing TimeML corpora. The resulting corpus contains 12,715 temporal relations over 36 documents taken from TimeBank 1.2. For the TimeBank-Dense evaluation, we follow the experimental setup in Chambers et al. (2014), in which the TimeBank-Dense corpus is split into a 22 document training set, a 5 document development set and a 9 document test set.[9]

To evaluate the causal relation extraction module, we use the Causal-TimeBank corpus[10] (Mirza and Tonelli, 2014a) for training. For TimeBank-Dense evaluation, the test set is a subset of TimeBank, so we exclude the 9 test documents from Causal-TimeBank during training. For TempEval-3 evaluation, we manually annotated 20 TempEval-3-platinum documents with causal links following the annotation guidelines of the Causal-TimeBank.[11] Causal relations are much sparser than temporal ones, and we found only 26 CLINKs.

Label Adjustment Since the set of TLINK types used in the TimeBank-Dense corpus is slightly different from the one used in TempEval-3,[12] we map the relation types of TLINKs labelled by the rule-based sieve of CATENA (Section 4.2.1) as follows: (i) BEGINS, ENDED_BY → BEFORE, (ii) BEGUN_BY, ENDS → AFTER, and (iii) DURING, IDENTITY → SIMULTANEOUS. The set of labels for the TLINK classifiers (Section 4.2.3) is also adjusted accordingly following the labels in the TimeBank-Dense training data.

Evaluation Results In Table 2, we compare the performance of CATENA with the two best-performing systems participating in the *Task C* of TempEval-3 (relation annotation given gold entities) and *Task C 'relation type only'* (relation annotation given gold entities and related pairs). We also compare the results on the second task with the results of Laokulrat et al. (2015), who recently presented a state-of-the-art system for relation classification based on timegraphs and stacked learning. In CATENA, Task C 'relation type only' is performed by disabling the module for identifying temporal links described in Section 4.1.

The evaluation shows that CATENA is the best performing system in both tasks, even if in Task C best precision and best recall are yielded by Bethard (2013) and Laokulrat et al. (2013), respectively. The recall drop (from .613 to .595) in Task C is because we remove the timex-timex pairs from the final

[8] Available at `https://www.cs.york.ac.uk/semeval-2013/task1/index.php\%3Fid=data.html`.

[9] Available at `http://www.usna.edu/Users/cs/nchamber/caevo/`.

[10] Available at `http://hlt-nlp.fbk.eu/technologies/causal-timebank`.

[11] Available at `https://github.com/paramitamirza/CATENA/data/`.

[12] Some relation types are not used, and the VAGUE relation introduced in the first TempEval task (Verhagen et al., 2007) is adopted to cope with ambiguous temporal relations, or to indicate pairs for which no clear temporal relation exists. The final set of TLINK types in TimeBank-Dense includes: BEFORE, AFTER, INCLUDES, IS_INCLUDED, SIMULTANEOUS and VAGUE.

	TempEval-3						TimeBank-Dense					
	Task C			Task C rel. type only				T-T	E-D	E-T	E-E	Overall
System	**P**	**R**	**F1**	**P**	**R**	**F1**	**System**	**F1**				**F1**
CATENA	.303	.595	**.402**	**.626**	**.613**	**.619**	**CATENA**	**.780**	.518	**.556**	.487	**.511**
Bethard (2013)	**.373**	.353	.363	-	-	-	CAEVO	.712	**.553**	.494	**.494**	.507
Laokulrat et al. (2013)	.152	**.656**	.247	.556	.574	.565						
Laokulrat et al. (2015)	-	-	-	.576	.579	.578						

Table 2: CATENA evaluated on Tempeval-3 data, compared with the two best participating systems according to UzZaman et al. (2013) and the system by Laokulrat et al. (2015) (left). CATENA is also compared with CAEVO on the TimeBank-Dense test set (right).

	CATENA						CAEVO		
	TempEval-3			TimeBank-Dense			TimeBank-Dense		
Sieve	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Temporal Relation Identification									
	.530	.954	.682	-	-	-	-	-	-
Temporal Relation Type Classification									
RB	.908	.127	.223	.727	.049	.092	-	-	-
RB + TR	.921	.163	.278	.713	.076	.138	-	-	-
ML	.610	.575	.592	.484	.471	.478	.458	.202	.280
RB + ML	.616	.595	.605	.495	.493	.494	.486	.240	.321
RB + TR + ML	**.626**	**.613**	**.619**	**.512**	**.510**	**.511**	.505	.328	.398
RB + TR + ML + AllVague	-	-	-	-	-	-	*.508*	*.506*	*.507*
Causal Relation Extraction									
RB	.917	.423	.579	-	-	-	-	-	-
ML	.429	.115	.182	-	-	-	-	-	-
RB + ML	**.737**	**.538**	**.622**	-	-	-	-	-	-

Table 3: Analysis of classifier performance per sieve. RB: rule-based sieve, ML: machine-learned sieve and TR: temporal reasoner.

annotated documents in order to avoid a relevant decrease in precision, since only very few of such pairs are annotated in the gold standard. The significant drop in precision shows the difficulty in matching annotators' decision to set TLINKs between entity pairs, although CATENA implements the instructions they had to follow in the annotation guidelines.

We also report in Table 2 the performance of CATENA in the TimeBank-Dense evaluation and compare it with CAEVO. We report only F1-score, since all possible links are labelled, yielding the same P and R values. We achieve a small improvement in the overall F1-score, i.e., .511 vs .507. If we consider the different entity pairs, CATENA performs best on timex-timex and event-timex relations, while CAEVO still achieves the best results on event-DCT and event-event pairs. One of the possible reasons for that is the lack of rules in CATENA to classify VAGUE TLINKs between E-E pairs, a relation type present only in TimeBank-Dense.

In order to measure the contribution of each component to the overall performance of CATENA, we also evaluate the performance of each sieve both in the temporal and in the causal module. Results are reported in Table 3, evaluated on both TempEval-3 and TimeBank-Dense test data. As expected, running a transitive closure module after the temporal rule-based sieve (RB + TR) results in improving recall, but the overall performance is still lacking (less than .30 F1-score).

Combining rule-based and machine-learned sieves (RB + ML) yields a slight improvement compared with enabling only the machine-learned sieve in the system (ML). Introducing the temporal reasoner module between the two sieves (RB + TR + ML) proves to be even more beneficial. This is especially evident in the TimeBank-Dense evaluation. The same phenomena are also observed by CAEVO; Table 3 (right) shows the related numbers reported in Chambers et al. (2014). Note that in CAEVO, the machine-learned sieves are not the last sieves, instead, the *AllVague* sieve is finally activated to label all remaining unlabelled pairs as VAGUE.

For causal relation extraction, the combination of rule-based and machine-learned sieves (RB + ML) achieves .622 F1-score in TempEval-3 evaluation, with the ML component contributing to increase

E-E pair	Sentence	TE3-gold	TE-label	CA-label	Post-editing
(e_{32}, e_{44})	*The [incident]* $_{e_{32}}$ *provoked an international [outcry]* $_{e_{44}}$*...*	-	SIMULTANEOUS	CLINK	BEFORE
(e_{32}, e_{45})	*The [incident]* $_{e_{32}}$ *provoked an international outcry and led to a major [deterioration]* $_{e_{45}}$ *in relations...*	-	AFTER	CLINK	BEFORE
(e_{18}, e_{19})	*...the [inspections]* $_{e_{18}}$ *were directly linked to the new law on NGOs and the targeted groups' [compliance]* $_{e_{19}}$ *with it.*	-	IS_INCLUDED	CLINK-R	AFTER
(e_4, e_6)	*A haze akin to volcanic fumes [cloaked]* $_{e_4}$ *the capital, causing convulsive [coughing]* $_{e_6}$ *and...*	INCLUDES	AFTER	CLINK	BEFORE

Table 4: Examples of E-E pairs in the TempEval-3-platinum dataset with gold annotated labels (TE3-gold), labelled by the temporal module (TE-label) and causal module (CA-label) of CATENA. These examples illustrate how TLINK post-editing using CLINK could improve the labelling quality.

the recall of the highly precise RB component. The low precision of the ML module is mostly due to dependency parsing mistakes and issues in disambiguating signals such as *from*, as in *"...passenger cars in China was on track to hit [400 million]* $_{\text{T}}$ *by 2030, up* ***from*** *[90 million]* $_{\text{S}}$ *now."* Unfortunately, from the total of 5 gold CLINKs in the 20 documents of the TimeBank-Dense test set, none is identified by CATENA.

6.2 Interaction between Temporal and Causal Relations

As shown in Figure 1, E-E labels returned by the temporal reasoner are used by the CLINK classifier as features, whose causal relations are then used to post-edit TLINK labels. We evaluate the impact of the first step through an ablation test, by removing TLINK types from the features used by the CLINK classifier. We only analyse the results of TempEval-3 evaluation, since there are no causal links recognized in the TimeBank-Dense test corpus. Without TLINK types, the F1-score drops from .622 to .571, with a significant recall drop from .538 to .462. This shows that temporal information is beneficial to the classification of causal relations between events, especially in terms of recall.

As for the evaluation of TLINK post-editing using CLINKs, the system identifies 19 causal links in the test set, which are passed to the temporal module. While 15 of them are already consistent with BEFORE/AFTER labels, 3 would add new correct TLINKs that are currently not annotated in the evaluation corpus, and were wrongly labelled by the temporal module of CATENA, as shown in Table 4. The fourth would add a BEFORE relation between *cloaked* and *coughing* in *"A haze akin to volcanic fumes [cloaked]* $_{\text{S}}$ *the capital,* ***causing*** *convulsive [coughing]* $_{\text{T}}$ *..."*. This relation is labelled as INCLUDES in the gold standard, but we believe that BEFORE would be correct as well.

7 Conclusions

We presented CATENA, a hybrid system for the extraction and classification of temporal and causal relations in text, which we make freely available to the research community. We adopt a sieve-based architecture both for the temporal and the causal module, integrating rule-based and machine learning components. The two modules were evaluated separately, showing that they achieve state-of-the-art performance on different tasks. Furthermore, the interaction between temporal and causal components, especially the benefits of passing information from one module to the other, was also analysed.

The system relies on the notion of events as defined in the TimeML standard, making it possible to easily put temporal and causal information in relation. Although the interplay between causality and temporality may seem obvious from a theoretical point of view, CATENA allows a systematic study and a quantification of this phenomenon. The presented approach would probably have more impact if implicit causality was also considered, which we did not take into account because it is not annotated in the Causal-TimeBank corpus. However, we plan to investigate this issue in the near future.

Acknowledgments

The research leading to this paper was partially supported by the European Union's 7th Framework Programme via the NewsReader Project (ICT-316404).

References

James F. Allen. 1983. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, November.

Steven Bethard and James H. Martin. 2008. Learning semantic links from a corpus of parallel temporal and causal relations. In *Proceedings of ACL-08: HLT, Short Papers*, pages 177–180, Columbus, Ohio, June. Association for Computational Linguistics.

Steven Bethard, William Corvey, Sara Klingenstein, and James H. Martin. 2008. Building a corpus of temporal-causal structure. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

Steven Bethard. 2013. Cleartk-timeml: A minimalist approach to tempeval 2013. In *Proceedings of the Seventh International Workshop on Semantic Evaluation*, SemEval '13, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Anders Bjorkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstrations*, pages 33–36, Beijing, China, August. Coling 2010 Organizing Committee.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Quang Do, Yee Seng Chan, and Dan Roth. 2011. Minimally supervised event causality identification. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 294–303, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Jennifer D'Souza and Vincent Ng. 2013. Classifying temporal relations with rich linguistic knowledge. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 918–927, Atlanta, Georgia, June. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT 2013*, pages 758–764, Atlanta, Georgia, June. ACL.

Natsuda Laokulrat, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Uttime: Temporal relation classification using deep syntactic features. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 88–92, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Natsuda Laokulrat, Makoto Miwa, and Yoshimasa Tsuruoka. 2015. Stacking approach to temporal relation classification with temporal inference. *Journal of Natural Language Processing*, 22(3):171–196.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, ICML '98, pages 296–304, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kenneth C. Litkowski and Orin Hargraves. 2006. Coverage and inheritance in the preposition project. In *3rd ACL-SIGSEM Workshop on Prepositions*.

Ken Litkowski. 2014. Pattern dictionary of english prepositions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1274–1283, Baltimore, Maryland, June. Association for Computational Linguistics.

Hector Llorens, Nathanael Chambers, Naushad UzZaman, Nasrin Mostafazadeh, James Allen, and James Pustejovsky. 2015. Semeval-2015 task 5: Qa tempeval - evaluating temporal information understanding with question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 792–800, Denver, Colorado, June. Association for Computational Linguistics.

Paramita Mirza and Sara Tonelli. 2014a. An analysis of causality between events and its relation to temporal information. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2097–2106, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Paramita Mirza and Sara Tonelli. 2014b. Classifying temporal relations with simple features. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 308–317, Gothenburg, Sweden, April. Association for Computational Linguistics.

Paramita Mirza, Rachele Sprugnoli, Sara Tonelli, and Manuela Speranza. 2014. Annotating causality in the tempeval-3 corpus. In *Proceedings of the EACL 2014 Workshop on Computational Approaches to Causality in Language (CAtoCL)*, pages 10–19, Gothenburg, Sweden, April. Association for Computational Linguistics.

Emanuele Pianta, Christian Girardi, and Roberto Zanoli. 2008. The textpro tool suite. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

James Pustejovsky, José Castaño, Robert Ingria, Roser Saurí, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003. Timeml: Robust specification of event and temporal expressions in text. In *Proceedings of the Fifth International Workshop on Computational Semantics (IWCS-5)*.

H Reichenbach. 1947. *Elements of symbolic logic*. University of California Press, Berkeley, CA.

Mehwish Riaz and Roxana Girju. 2013. Toward a better understanding of causality between verbal events: Extraction and analysis of the causal power of verb-verb associations. In *Proceedings of the SIGDIAL 2013 Conference*, pages 21–30, Metz, France, August. Association for Computational Linguistics.

Bryan Rink, Cosmin Adrian Bejan, and Sanda M. Harabagiu. 2010. Learning textual graph patterns to detect causal event relations. In *FLAIRS Conference*.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 75–80, Prague, Czech Republic, June. Association for Computational Linguistics.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 57–62, Uppsala, Sweden, July. Association for Computational Linguistics.

Matthias Westphal and Stefan Wölfl. 2009. Qualitative CSP, finite CSP, and SAT: Comparing Methods for Qualitative Constraint-based Reasoning. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, IJCAI'09, pages 628–633, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Matthias Westphal, Stefan Wölfl, and Jason Jingshi Li. 2010. Restarts and nogood recording in qualitative constraint-based reasoning. In *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*, pages 1093–1094.

Phillip Wolff and Grace Song. 2003. Models of causation and the semantics of causal verbs. *Cognitive Psychology*, 47(3):276–332.

Phillip Wolff. 2007. Representing causation. *Journal of experimental psychology: General*, 136(1):82–111.

Appendix A Temporal Rule Set

tense	*aspect*	**E-D label**
PAST	PERFECTIVE	BEFORE
PRESENT	PROGRESSIVE	INCLUDES
PRESENT	PERFECTIVE_PROGRESSIVE	INCLUDES
FUTURE	*	AFTER

Table 5: E-D label rules based on *tense* and *aspect* of E.

tsense	**E-T label**
TIMEPOINT (e.g. *in*, *at*, *on*)	IS_INCLUDED
TIMEPRECEDING (e.g. *before*)	BEFORE
TIMEFOLLOWING (e.g. *after*)	AFTER
DURATION (e.g. *during*, *throughout*)	DURING
STARTTIME (e.g. *from*, *since*)	BEGUN_BY
ENDTIME (e.g. *until*)	ENDED_BY

Table 6: E-T label rules based on the sense of temporal preposition (*tsense*) introducing T.

dep	e_1 *verb info*	**E-E label**	**Example**
LGS-PMOD	*	AFTER	"...reaction *[touched]* $_{e_1}$ off by the *[collapse]* $_{e_2}$ of..."
LOC-PMOD	*	IS_INCLUDED	"...enormous *[surge]* $_{e_1}$ in coal *[consumption]* $_{e_2}$..."
OPRD-IM/OPRD	aspectual verb for *initiation*	BEGINS	"...situation *[began]* $_{e_1}$ to *[relax]* $_{e_2}$ in..."
	aspectual verb for *culmination/termination*	ENDS	"...we 'd *[stop]* $_{e_1}$ *[bidding]* $_{e_2}$."
	aspectual verb for *continuation*	INCLUDES	"...industry 's growth *[continues]* $_{e_1}$ to *[slow]* $_{e_2}$."
	general verb, *aspect*=PERFECTIVE_PROGRESSIVE	SIMULTANEOUS	"...have been *[working]* $_{e_1}$ to *[develop]* $_{e_2}$ quantum..."
	general verb	BEFORE	"...consortium *[attempted]* $_{e_1}$ to *[block]* $_{e_2}$..."

Table 7: E-E label rules based on dependency path (*dep*) and verb information of e_1 (e_1 *verb info*).

Appendix B Causal Rule Set

v	***dep***$_1$	***dep***$_2$	***dir***	**E-E label**
AFFECT	(*)	OBJ		CLINK
LINK	(*)	OBJ/ADV-PMOD/DIR-PMOD/AMOD-PMOD	CLINK	CLINK
			CLINK-R	CLINK-R
CAUSE/ENABLE/PREVENT	(*)	OBJ/OPRD/OPRD-IM/ADV-PMOD		CLINK
		LGS-PMOD		CLINK-R
CAUSE-/ENABLE-/PREVENT-AMBIGUOUS	(*)	OPRD/OPRD-IM/ADV-PMOD		CLINK

Table 8: Causal verb rules for E-E pairs based on causal verb (v) category, dependency paths between v and e_1/e_2, and causal direction sense (*dir*). (*) denotes all possible dependency paths listed in Table 9.

Relation	**Path**	**Example**
between v **and** e_1	***dep1***	
e_1 is subject of v	SBJ	*The Pope's [visit]* $_{e_1}$ ***persuades*** $_v$ *Cubans...*
v is predicative complement of e_1	PRD-IM	*The [roundup]* $_{e_1}$ *was to* ***prevent*** $_v$ *them...*
v is modifier of e_1 (nominal)	NMOD	*An [agreement]* $_{e_1}$ *that* ***permits*** $_v$ *the Russian...*
v is apposition of e_1	APPO	*..., with the [crisis]* $_{e_1}$ ***triggered*** $_v$ *by...*
v is general adverbial of e_1	ADV	*The number [increased]* $_{e_1}$, ***prompting*** $_v$*...*
v is adverbial of purpose/reason of e_1	PRP-IM	*The major [allocated]* $_{e_1}$ *funds to* ***help*** $_v$*...*
between v **and** e_2	***dep2***	
e_2 is object of v	OBJ	*...have* ***provoked*** $_v$ *widespread [violence]* $_{e_2}$.
e_2 is logical subject of v (passive verb)	LGS-PMOD	*...***triggered** $_v$ *by the [end]* $_{e_2}$ *of the...*
e_2 is predicative complement of v (raising/control verb)	OPRD	*...funds to* ***help*** $_v$ *[build]* $_{e_2}$ *a museum.*
	OPRD-IM	*...***persuades** $_v$ *Cubans to [break]* $_{e_2}$ *loose.*
e_2 is general adverbial of v	ADV-PMOD	*...***protect** $_v$ *them from unspecified [threats]* $_{e_2}$.
e_2 is adverbial of direction of v	DIR-PMOD	*...***lead to** $_v$ *a [surge]* $_{e_2}$ *of inexpensive imports.*
e_2 is modifier of v (adjective or adverbial)	AMOD-PMOD	*...***related to** $_v$ *[problems]* $_{e_2}$ *under a contract.*

Table 9: Dependency paths considered for setting a causal link between two events e_1 and e_2 when a causal verb v is present.

Forecasting Word Model: Twitter-based Influenza Surveillance and Prediction

Hayate ISO, Shoko WAKAMIYA, Eiji ARAMAKI
Nara Institute of Science and Technology
{iso.hayate.id3,wakamiya,aramaki}@is.naist.jp

Abstract

Because of the increasing popularity of social media, much information has been shared on the internet, enabling social media users to understand various real world events. Particularly, social media-based infectious disease surveillance has attracted increasing attention. In this work, we specifically examine influenza: a common topic of communication on social media. The fundamental theory of this work is that several words, such as symptom words (*fever*, *headache*, etc.), appear in advance of flu epidemic occurrence. Consequently, past word occurrence can contribute to estimation of the number of current patients. To employ such forecasting words, one can first estimate the optimal time lag for each word based on their cross correlation. Then one can build a linear model consisting of word frequencies at different time points for nowcasting and for forecasting influenza epidemics. Experimentally obtained results (using 7.7 million tweets of August 2012 – January 2016), the proposed model achieved the best nowcasting performance to date (correlation ratio 0.93) and practically sufficient forecasting performance (correlation ratio 0.91 in 1-week future prediction, and correlation ratio 0.77 in 3-weeks future prediction). This report reveals the effectiveness of the word time shift to predict of future epidemics using Twitter.

1 Introduction

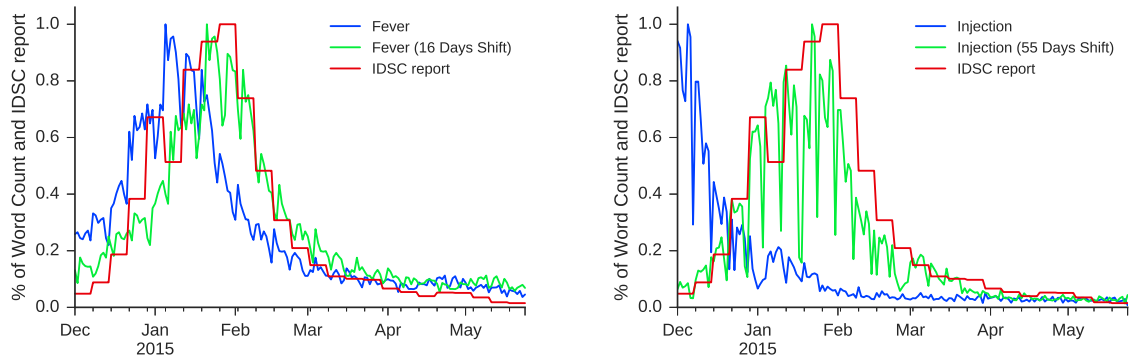
The increased use of social media platforms has led to wide sharing of personal information. Especially Twitter, a micro-blogging platform that enables users to communicate by updating their status using 140 or fewer characters, has attracted great attention of researchers and service developers because Twitter can be a valuable personal information resource. The feasibility of such approaches, known as social sensors, has been demonstrated in various event detection systems such as earthquakes (Sakaki et al., 2010), outbreaks of disease (Chew and Eysenbach, 2010), and stock market fluctuations (Bollen et al., 2011). Among the applications mentioned above, this study particularly examines detection of seasonal influenza epidemics because the influenza detection is a popular application of Twitter. To date, more than 30 Twitter-based influenza detection and prediction systems have been developed worldwide (Charles-Smith et al., 2015).

Although the detailed functions of these systems differ, they share the underlying assumption that the flu spreading in the real world is immediately reflected to the tweets. Therefore, most systems have simply aggregated counts of daily flu-related tweets to obtain the current patient status (Aramaki et al., 2011; Collier et al., 2011; Chew and Eysenbach, 2010; Lampos and Cristianini, 2010; Culotta, 2013; Paul et al., 2014). Their typical materials are presented as shown below.

- *I got a* **flu** 😢 *I can not go to school for the rest of the week*
- *I was diagnosed with a high* **fever**. *Maybe flu :(*

Although the former tweet is described by an actual influenza patient, the latter one merely expresses a suspicion of flu. From a practical (clinical) perspective, these differences have great importance because

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 76–86, Osaka, Japan, December 11-17 2016.



(a) "*Fever.*" (b) "*Injection.*"

Figure 1: Motivating examples: The time lag of the frequency of a word enables one to obtain a good approximation to the number of patients. The blue line shows the word frequency. The green line shows the word frequency shifted time lag days. The red line shows the number of patients.

the latter is noise that impedes precise influenza surveillance. Therefore, earlier studies (Aramaki et al., 2011; Kanouchi et al., 2015; SUN et al., 2014) have devoted great efforts to removal of such noise (suspicion, negation, news wired, and so on).

This study employs such noisy tweets. We assume that a word, "*fever*" presents a clue to an upcoming influenza outbreak. Inferring that people are frequently afflicted by symptoms such as "*fever*" and "*headache*" immediately before the onset and diagnosis of influenza, we designate such words as **forecasting words**.

More concrete examples of forecasting words are presented in Figure 1a. The figure reveals that an approximately 16-day time lag exists between the frequency of "*fever*" (blue line) and the number of patients (red line). If this time lag was known in advance, one could obtain a good approximation of the number of patients (red line) by a 16-day time shift operation (green line). Similarly, flu prevention words such as "*shot*" and "*injection*" have previously been used to describe outbreaks.

- *I took a flu* **shot** *today* 💉
- *I don't wanna get a flu* **injection** *cuz it hurts me*

In the latter case as shown in Figure 1b, we can find much longer time lag (55 days) between tweets (frequency of "*injection*") and the reality (number of patients).

Presuming that each word has its own time lag, then the problems to be solved are two-fold: (1) estimating the optimal time lag for each forecasting word and (2) incorporating these time lags into the model.

For the first problem, the suitable time lag for each word is measured by calculating the cross correlation between the word frequency and the patient number. For the second problem, we construct a word frequency matrix that consists of a shifted word frequency timeline (Sec. 3). Next, a linear model called **nowcasting model** is constructed from the modified word matrix, for which the parameters are estimated using several regularization models, Lasso and Elastic Net (Sec. 4).

Moreover, the nowcasting model can be extended easily to a predictive model called a **forecasting model**. In the forecasting model (Δf days future), only forecasting words that have more than n day time lag are used (Sec. 5).

Nowcasting models can dramatically boost the current patient number estimation capability (correlation ratio 0.93; $+0.10$ point). Forecasting models have demonstrated successful prediction performance (the correlation ratio 0.91 in 1-week future prediction, and the correlation ratio 0.77 in 3-weeks future prediction). This performance goes beyond the practical baseline (over 0.75 correlation).

Our contributions are summarized as presented below.

- We discover that **forecasting words** have a time lag between the virtual world (number of tweets in Twitter) and the real world (number of patients).

- We propose a method to build time-shifted features using **cross correlation** measures.
- We realize **nowcasting model** and its extended one, **forecasting model**, based on the time shift with parameter estimation. This report is the first of the relevant literature describing a successful model enabling the prediction of future epidemics over the practical baseline.

We make code and data publicly available. [1]

2 Dataset

2.1 Influenza Corpus

We collected 7.7 million influenza related tweets, starting from August 2012 to January 2016, via Twitter API[2]. Then, we filtered noises (removed retweets including the word, *RT*, and tweets linked to other web pages including the word, *http* from the collected tweet data). In the case of just counting influenza-related tweets, we should only consider unique users to avoid to count more than ones the tweets of the same patients. However, we didn't filter out the users which posted influenza-related tweets multiple times because we provide the different word for the different role even if these tweets were posted by the same patients. For example, the word, "fever" for nowcasting, and the word, "injection" for forecasting. To analyze a word, we applied a Japanese morphological parser (JUMAN[3]) and obtained the stem forms. As a result, 27,588 words were extracted. Then, we investigated the word frequency per day to build a word matrix ($days \times words$) as shown in Figure 2a.

2.2 IDSC report

In Japan, the Infectious Disease Surveillance Center (IDSC) announces the number of influenza patients once a week during an influenza epidemic season (typically during November–May in Japan). In fact, IDSC reports tend to delay around a week likewise the U.S. Centers for Disease Control and Prevention (CDC) (Paul et al., 2014), but even if we consider such time delay, twitter stream attains the peak faster than the real world.

To use the IDSC reports for evaluation, we divided the data into the following three periods: 2012/12/01–2013/05/31 (Season 1), 2013/12/01–2014/05/31 (Season 2), and 2014/12/01–2015/05/24 (Season 3). We prepared a buffer time (60 day maximum time shift) immediately preceding the experimental periods to secure the time shift width.

3 Method

To estimate the current influenza epidemics (nowcast) and forecast the future ones, the number of influenza patients was derived from the following linear model.

$$\hat{y}^{(t)} = x_1^{(t-\hat{\tau}_1)}\hat{\beta}_1 + x_2^{(t-\hat{\tau}_2)}\hat{\beta}_2 + \cdots + x_{|V|}^{(t-\hat{\tau}_{|V|})}\hat{\beta}_{|V|}$$

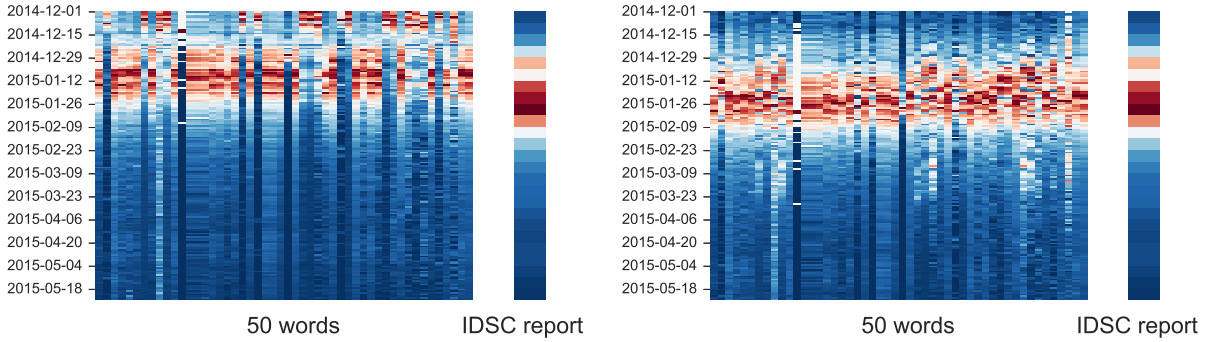
Therein, $\hat{y}^{(t)}$ shows the estimated number of influenza patients at time t, $x_v^{(t)}$ stands for the count of a word v at time t, and $\hat{\beta}$ represents a weight estimated in the training phase, $\hat{\tau}_v$ denotes a suitable time shift parameter for word v decided in the training phase, and $|V|$ denotes the size of vocabulary.

This section first provides methods to explore the most suitable time shift width $\hat{\tau}_v$ for each word v (Sec. 3.1). Then, the parameter estimation method is described (Sec. 3.2). Finally, the model of future prediction based on the original model is explained (Sec. 3.3).

[1] http://sociocom.jp/~iso/forecastword

[2] The tweet data dropout during June–October in 2013 and during June–October in 2014, because the Twitter API specifications were changed in those periods.

[3] http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN



(a) Ordinary Word Matrix.

(b) Time Shifted Word Matrix.

Figure 2: Word matrix transformation. The Y-axis shows a timeline. The X-axis shows words with the IDSC reports (right side).

3.1 Time Shift Estimation

The first problem to be solved is finding the optimal time shift width that achieves the best fit to the target influenza timeline. Given the IDSC reports and wider range of tweets, **Cross Correlation** is used to search for the most suitable time shift width for each word frequency as

$$r_{\mathbf{x}_v,\mathbf{y}}(\tau) = \frac{\sum_{t=1}^{T}(x_v^{(t-\tau)} - \bar{x}_v^{(t-\tau)})(y^{(t)} - \bar{y})}{\sqrt{\sum_{t=1}^{T}(x_v^{(t-\tau)} - \bar{x}_v^{(t-\tau)})^2 \sum_{t=1}^{T}(y^{(t)} - \bar{y})^2}},$$

where τ is a time shift parameter (time shift width)[4]. The cross correlation $r_{\mathbf{x}_v,\mathbf{y}}(\tau)$ measures the similarity between (τ days) time shift variable $\mathbf{x}_v$ and objective $\mathbf{y}$. In this study, $x_v^{(t-\tau)}$ is the count of word v with time shift width τ days earlier from t and $\mathbf{y} = [y^{(1)}, \ldots, y^{(T)}]^\top$ is the number of patients from the IDSC reports. It is formulated as $\hat{\tau}_v = \underset{\tau}{\operatorname{argmax}}\, r_{\mathbf{x}_v,\mathbf{y}}$.

Next, we construct a matrix, $\mathbf{X} \in \mathbb{N}^{T \times V}$, where T stands for the timeline and V represents the vocabulary, according to the Algorithm 1.

Algorithm 1: Time-shifted word matrix for nowcasting.

Set the maximum shift parameter $\tau_{\max}$
for $v \leftarrow 1$ **to** $|V|$ **do**
  for $\tau \leftarrow 0$ **to** $\tau_{\max}$ **do**
    Calculate Cross Correlation $r_{x_v,y}(\tau)$
  end
  $\hat{\tau}_v = \underset{\tau \in \{0,\ldots,\tau_{\max}\}}{\operatorname{argmax}} r_{x_v,y}(\tau)$
  Shift the word vector to maximize Cross Correlation $\hat{\mathbf{x}}_\mathbf{v} \leftarrow [x_v^{(1-\hat{\tau}_v)}, x_v^{(2-\hat{\tau}_v)}, \ldots, x_v^{(T-\hat{\tau}_v)}]$
end
return *Shifted Word Matrix* $\mathbf{X} = [\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_{|V|}]$

The algorithm decides the optimal time shift width ($\hat{\tau}_{\mathbf{x}_v,\mathbf{y}}$) based on the cross correlation for each word. After time shifts for all words, a shifted word matrix $\mathbf{X}$ is constructed.

Figure 2a presents the initial (original) word matrix ($\tau = 0$ for all words) of 50 words (randomly selected). This matrix includes several low-correlated words, making several vertically irregular lines. In contrast, the time shift operation arranges the irregular words to match the IDSC reports, producing a beautiful horizontal line, as shown in Figure 2b.

3.2 Nowcasting

To construct the linear model (called **nowcasting model**), the parameter $\boldsymbol{\beta}$ is estimated as minimizing the squared error. For this study, the vocabulary size $|V|$ is of much larger order than sample size T

[4]The cross correlation is exactly the same as the Pearson's correlation when $\tau = 0$.

so that the ordinary least squares estimator is not unique. It heavily overfits the data. According to the previous study's manner, parameters with a penalty are estimated as shown below.

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmax}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \mathbf{P}(\boldsymbol{\beta}, \boldsymbol{\lambda})$$

In that equation, $\mathbf{P}(\boldsymbol{\beta}, \boldsymbol{\lambda})$ is the penalty term.

In the case of $\mathbf{P}_{lasso}(\boldsymbol{\beta}, \boldsymbol{\lambda}) = \lambda\|\boldsymbol{\beta}\|_1$, the regularization method called the Least Absolute Shrinkage and Selection Operator (Lasso) is a well-known method for selecting and estimating the parameters simultaneously (Tibshirani, 1994). In earlier studies, Lasso was employed to model influenza epidemics by Lampos and Cristianini (2010). However, in the case of vocabulary size $|V|$, which is much larger order than sample size T, it has been observed empirically that the prediction performance of l_1-penalized regression, the Lasso is dominated by the l_2-penalized one.

Therefore, we employ the Elastic Net (Zou and Hastie, 2005), which combines the l_1-penalty and l_2-penalty $\mathbf{P}_{enet} = \lambda(\alpha\|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2^2)$, where α is called l_1 ratio. The Elastic Net was already employed for nowcasting influenza-like illness rates using search query log, not Twitter (Lampos et al., 2015). In the case of $\alpha = 1$, Elastic Net is exactly the same as Lasso and $\alpha = 0$, Ridge (l_2 regularization). Similarly to Lasso, the Elastic Net simultaneously does automatic variable selection and continuous shrinkage. It has a l-2 regularization advantage that selects groups of correlated variables. Elastic Net, as the generalized method of Lasso and Ridge, estimates with equal or better performance compared to both.

3.3 Forecasting

Our nowcasting model can be extended naturally to **forecasting model**. To predict the number of future patients Δf days after, we force to shift the word frequency at least Δf days. To do so, a setting of the nowcasting model in Algorithm 1 is just changed to $\tau_{\min} = \Delta f$, as shown in Algorithm 2. It enables forecasting of future epidemics, demonstrating a widely applicable methodology of the proposed approach.

Algorithm 2: Time-shifted word matrix for forecasting.

Set the maximum shift parameter $\underline{\boldsymbol{\tau}_{\mathbf{min}}}, \tau_{\max}$
for $v \leftarrow 1$ **to** $|V|$ **do**
  for $\tau \leftarrow \underline{\boldsymbol{\tau}_{\mathbf{min}}}$ **to** $\tau_{\max}$ **do**
    Calculate Cross Correlation $r_{x_v,y}(\tau)$
  end
  $\hat{\tau}_v = \underset{\tau \in \{\underline{\boldsymbol{\tau}_{\mathbf{min}}}, \ldots, \tau_{\max}\}}{\operatorname{argmax}} r_{x_v,y}(\tau)$
  Shift the word vector to maximize Cross Correlation $\hat{\mathbf{x}}_{\mathbf{v}} \leftarrow [x_v^{(1-\hat{\tau}_v)}, x_v^{(2-\hat{\tau}_v)}, \ldots, x_v^{(T-\hat{\tau}_v)}]$
end
return *Shifted Word Matrix* $\mathbf{X} = [\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_{|V|}]$

4 Experiment 1: Nowcasting

To assess the nowcasting performance, we used the actual influenza reports provided by the Japanese IDSC.

4.1 Comparable Methods

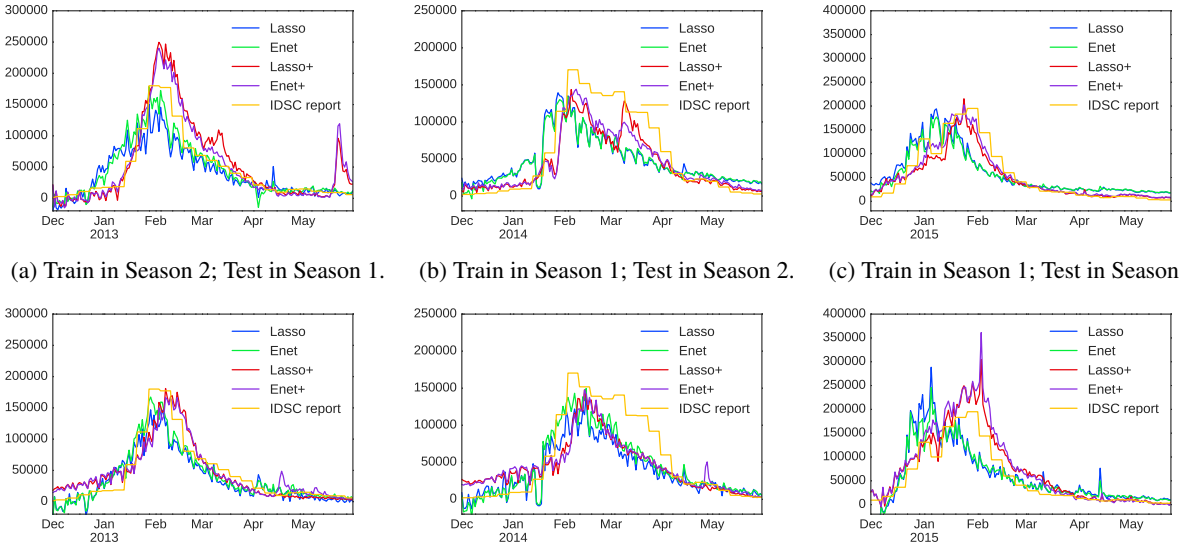
We compared four linear methods for nowcasting as shown below:

- `Lasso`: l_1-regularization method (Tibshirani, 1994; Lampos and Cristianini, 2010),
- `Lasso+`: Lasso and time shift combined method,
- `ENet`: Elastic-Net, which combines l_1-, l_2-regularization (Zou and Hastie, 2005),
- `ENet+`: Elastic-Net and time shift combined method.

All hyperparameters were tuned via five-fold cross validation in the training dataset.

Train	Season 2	Season 3	Season 1	Season 3	Season 1	Season 2	Avg.
Test	Season 1		Season 2		Season 3		
`Lasso`	0.854	0.916	0.768	0.894	0.770	0.753	0.826
`Enet`	0.900	**0.927**	0.809	**0.914**	0.792	0.805	0.858
`Lasso+`	**0.952**	0.907	0.951	0.888	0.955	**0.963**	**0.936**
`Enet+`	0.944	0.898	**0.960**	0.878	**0.967**	0.959	0.934

Table 1: Correlation between estimated values and the IDSC reports.



(a) Train in Season 2; Test in Season 1. (b) Train in Season 1; Test in Season 2. (c) Train in Season 1; Test in Season 3.

(d) Train in Season 3; Test in Season 1. (e) Train in Season 3; Test in Season 2. (f) Train in Season 2; Test in Season 3.

Figure 3: Timelines of estimated values obtained using the four methods for nowcasting.

4.2 Dataset and Evaluation Metric

The detailed dataset is described in Sec. 2. To construct the time-shifted word matrix, we set $\tau_{\max} = 60$. Our tweet corpus had a dropout period, so that we did not calculate the cross correlation with more than a 60-day shift. We employed each season's data as training data and others as test data.

The evaluation metric is based on correlation (Pearson correlation) between the estimated value and the value of the IDSC reports.

4.3 Result

Results of modeling accuracy are presented in Table 1. Correlations of our baselines, `Lasso` and `Enet`, were lower than those of previous studies. Results suggest that our dataset is more difficult than those used in earlier studies.

In contrast, time-shifted models (`Lasso+`, `Enet+`) demonstrated about 0.1 point improvement than their baseline models, indicating the contribution of time shift features.

It is noteworthy that `Lasso` type model and `Enet` type one did not differ so much. The whole trained model chose l_1 ratio parameter that is nearly equal to 1, so that the `Enet` type model became almost identical as `Lasso` type model.

Overestimation

Results showed that values in Figure 3a were overestimated in mid-May. One reason is that tweets related to news such as "**Scientists create hybrid flu that can go airborne**"[5] were popular in social media. Although tweets linked to web pages were removed during preprocessing, many tweets without links to web pages were posted by many people worried about the news. An example of such tweets is the following:

[5] `http://go.nature.com/29ATqc9`

- *What? In an attempt to make a vaccine for bird flu and swine flu had created a new strain of influenza virus? 😡 What are you doing? 😡😡😡*

In addition, the model trained in Season 2 included the word "*bird*" as one feature. This word's time shift was 15 days. Consequently, this peak occurred.

In most cases, these kinds of outlier words are not selected through model selection, but preprocessing will play an crucial role to prevent these kinds of outlier.

5 Experiment 2: Forecasting

We evaluate the forecasting performance described in Sec 3.3.

5.1 Comparable methods

`Lasso` and `Enet` have no features for predicting future values. Therefore, we use `Lasso+` and `Enet+` for forecasting. Additionally, we employ the following baseline model of `BaseLine`: $\hat{y}^{(t)}_{\text{test}} = y^{(t)}_{\text{train}}$ for comparison with our proposed models.

5.2 Dataset and Evaluation Metric

To evaluate the forecasting performance, we used the same dataset and evaluation metric as Experiment 1, except that we set the minimum time shift $\tau_{\min}$ from 1 day to 30 days.

5.3 Result

Results of forecasting accuracy are presented in Figure 4. In both models, the accuracy was superior to the baseline until around 3 weeks into the future. In addition, the accuracy for prediction one week into the future was almost identical to that in the case of $\tau_{\min} = 0$. That result might occur because the accuracy about one week future was nearly the same as that for the current state. In addition, there were many highly correlated features by shifting around 10 days into the future. Consequently, our model demonstrated equivalent performance up to 10 days into the future.

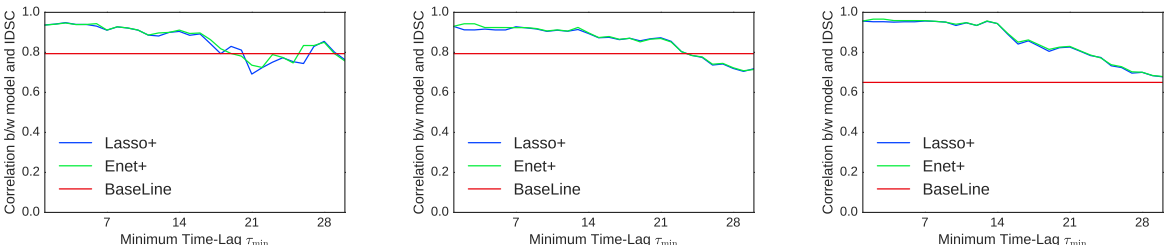
Furthermore, the forecasting performance decreased dramatically along with the increase of $\tau_{\min}$, as shown in Figure 4e. We discuss that point further in Sec. 6.

Figure 5 presents timeline plots of examples. From Figure 5a to Figure 5d are shown the values estimated by the forecasting models trained in Season 2 and tested in Season 1 for $\tau_{\min} \in \{7, 14, 21, 28\}$. The estimated values showed a consistently similar shape to that of the IDSC report. In Figure 5c, the same word, "*bird*", occurred as described in Sec. 4.3. In contrast, the weight for "*bird*" decreased in Figure 5d for that reason, the forecasting accuracy increased.

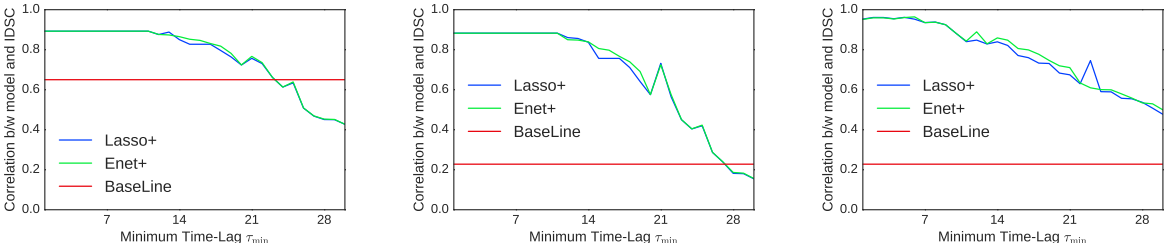
Then, from Figure 5e to Figure 5h show the values estimated by the forecasting models trained in Season 3 and tested in Season 2 for the same $\tau_{\min}$. Our models overestimated before outbreaks and underestimated after the peak of influenza epidemics. For $\tau_{\min} = 28$, this phenomenon was widely evident. We discuss that point further in Sec. 6.

6 Discussion

In general, the proposed approach (time shift operation) fitted the IDSC reports, demonstrating the basic feasibility. However, exceptions were apparent, as for the model trained in Season 3. One reason is that a gap exists in the suitable time shift widths between the train (Season 3) and the other (Seasons 1 and 2). `Lasso+` model trained in Season 3 selected the words, "*fever*" with $\hat{\tau}_{\textit{fever}} = 16$, "*vaccination*" with $\hat{\tau}_{\textit{vaccination}} = 55$, "*absent*" with $\hat{\tau}_{\textit{absent}} = 10$, and others as features. These words have high correlations only in Season 3, with poor correlation in other seasons. The most drastic example is "*vaccination*" with $\hat{\tau}_{\textit{vaccination}}$, (over 0.849 correlation in Season 3). This word is adversely affected by other seasons (0.313 correlation in Season 1 and 0.04 correlation in Season 2). The reason for the lost correlation was that $\hat{\tau}_{\textit{vaccination}}$ in Season 3 differed from that of other seasons. This phenomenon suggests that "*vaccination*" is just an annually cycling word. Neither the cycle of "*vaccination*" nor that of influenza is fixed, bringing us different time lags.

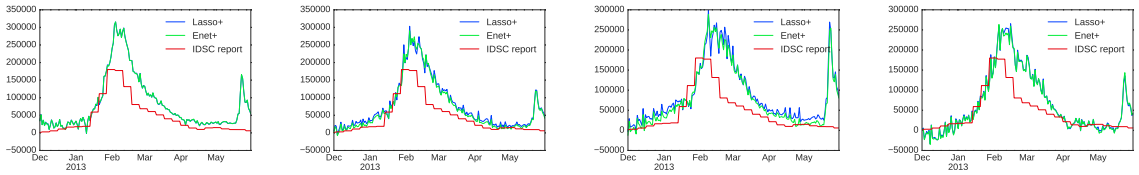


(a) Train in Season 2; Test in Season 1. (b) Train in Season 1; Test in Season 2. (c) Train in Season 1; Test in Season 3.

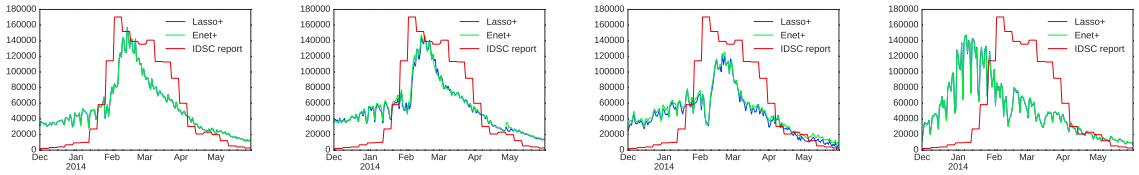


(d) Train in Season 3; Test in Season 1. (e) Train in Season 3; Test in Season 2. (f) Train in Season 2; Test in Season 3.

Figure 4: Correlation between estimated values using the two methods for forecasting.



(a) $\tau_{\min} = 7$ in Fig. 4a. (b) $\tau_{\min} = 14$ in Fig. 4a. (c) $\tau_{\min} = 21$ in Fig. 4a. (d) $\tau_{\min} = 28$ in Fig. 4a.



(e) $\tau_{\min} = 7$ in Fig. 4e. (f) $\tau_{\min} = 14$ in Fig. 4e. (g) $\tau_{\min} = 21$ in Fig. 4e. (h) $\tau_{\min} = 28$ in Fig. 4e.

Figure 5: Timelines of values estimated using the two methods for forecasting and the IDSC reports in each $\tau_{\min}$.

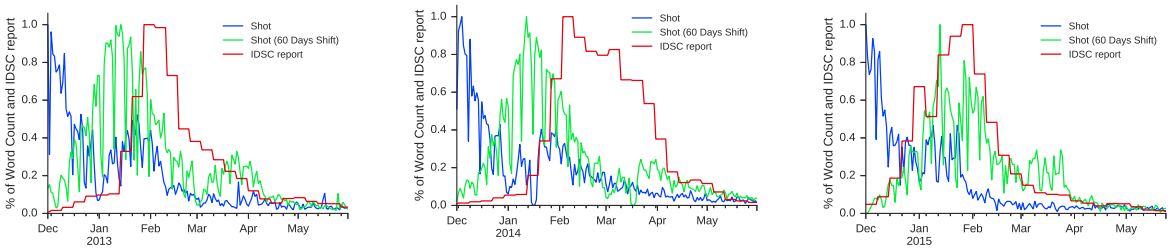
This inconsistency of time shifts also affected the forecasting performance directly. As shown in Figure 4e, the forecasting performance was decreased dramatically against the increase of $\tau_{\min}$. In spite of the word "*shot*" is the largest weighted feature in the case of $\tau_{\min} = 21$ and Train in Season 3, these word correlations were 0.310 in Season 1 and 0.03 in Season 2. Consequently, it caused a considerable decrease of the forecasting accuracy. In contrast, some words, such as "*fever*" and "*symptom*", showed consistently similar time shifts.

A technique to distinguish actual forecasting words such as "*fever*", and noises (simple year cycle words), "*vaccination*" is highly anticipated for use in the near future. If multiple-year training sets were available, one could filter out such noisy words.

Although some room for improvement remains, the basic feasibility of the proposed approach has been demonstrated. The time shift was effective for social media based surveillance. In addition, the model enables prediction.

7 Related Work

To date, numerous web based surveillance systems have been proposed, targeting the common cold (Kitagawa et al., 2015), drug side effects (Bian et al., 2012), cholera (Chunara et al., 2012), *E. Coli* (Diaz-Aviles et al., 2012), problem drinking (MA et al., 2012), smoking (Prier et al., 2011), campylobacteriosis (Chester et al., 2011), dengue fever (Gomide et al., 2011), and HIV/AIDS (Ku et al., 2010). Influenza has especially drawn much attention from earlier studies (Ginsberg et al., 2009; Polgreen et al., 2009;



(a) Season 1. (b) Season 2. (c) Season 3.

Figure 6: Frequencies of "*Shot*" in respective seasons.

Hulth et al., 2009; Corley et al., 2010) to current Twitter-based studies (Aramaki et al., 2011; Collier et al., 2011; Chew and Eysenbach, 2010; Lampos and Cristianini, 2010; Culotta, 2013).

Because of great variance in data resources and evaluation manner (region, year, only winter or all seasons), a precise comparison would be difficult and meaningless, Culotta (Culotta, 2013) and Ginsberg (Ginsberg et al., 2009) are apparently better than the others in US (correlation ratios $= 0.96$ and 0.94, respectively). Aramaki et al. (2011) achieved the best score for Japan (correlation ratio $= 0.89$). This study also examined Twitter data in Japan, and achieved competitive results for nowcasting. Another aspect of reviews of related studies is the manner of tweet counting. In earlier studies, a simple word counting, the direct number of tweets, is considered an index of the degree of disease epidemics. However, such a simple method is adversely affected by the huge numbers of noisy tweets. Currently, counting approaches of two types have been developed: (1) a classification approach (Kanouchi et al., 2015; SUN et al., 2014; Aramaki et al., 2011) aimed at extracting only tweets including patient information, and (2) a regression approach (Lamb et al., 2013; Culotta, 2010; Lampos and Cristianini, 2010; Paul and Dredze, 2011) that handles multiple words to build a precise regression model.

The proposed study fundamentally belongs among regression approaches, which explore optimal weight perimeters for each word. An important difference is that this study handles one more parameter for each word: time shift (days). To handle many parameters, we first ascertain the best time shift widths. Then we explore weight parameters using L1 or elastic net. It is noteworthy that this study does not employ any classification method, engaging a room to improve by incorporation with classification techniques.

8 Conclusions

This study proposed a novel social media based influenza surveillance system using forecasting words that appear in Twitter usage before main epidemics occur. First, for each word, the optimal time lag was explored, which maximized the cross correlation to influenza epidemics. Then, we shifted a matrix consisting of word frequencies at different time points by each optimal time lag. Using the time-shifted word matrix, this study produced and evaluated a nowcasting model and forecasting model designed to predict the number of influenza patients. In the experimentally obtained results, the proposed model achieved the best nowcasting performance to date (correlation ratio 0.93) and practically sufficient forecasting performance (correlation ratio 0.91 in the 1-week future prediction, and correlation ratio 0.77 in 3-week future prediction). This report is the first of the relevant literature describing a model that enables prediction of future epidemics. Furthermore, the model has much room for potential application to prediction of other events.

References

Eiji Aramaki, Sachiko Maskawa, and Mizuki Morita. 2011. Twitter catches the flu: Detecting influenza epidemics using twitter. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568–1576.

Jiang Bian, Umit Topaloglu, and Fan Yu. 2012. Towards large-scale twitter mining for drug-related adverse events. In *Proceedings of the International Workshop on Smart Health and Wellbeing (SHB)*, pages 25–32.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

Lauren E Charles-Smith, Tera L Reynolds, Mark A Cameron, Mike Conway, Eric HY Lau, Jennifer M Olsen, Julie A Pavlin, Mika Shigematsu, Laura C Streichert, Katie J Suda, et al. 2015. Using social media for actionable disease surveillance and outbreak management: A systematic literature review. *PloS one*, 10(10):e0139701.

Tammy L. Stuart Chester, Marsha Taylor, Jat Sandhu, Sara Forsting, Andrea Ellis, Rob Stirling, and Eleni Galanis. 2011. Use of a web forum and an online questionnaire in the detection and investigation of an outbreak. *Online J Public Health Inform*, 3(1):1–7.

C. Chew and G. Eysenbach. 2010. Pandemics in the age of twitter: Content analysis of tweets during the 2009 h1n1 outbreak. *PLoS ONE*, 5:e14118.

Rumi Chunara, Jason R. Andrews, and John S. Brownstein. 2012. Social and news media enable estimation of epidemiological patterns early in the 2010 haitian cholera outbreak. *Am J Trop Med Hyg.*, 86(1):39–45.

Nigel Collier, Nguyen Truong Son, and Ngoc Mai Nguyen. 2011. Omg u got flu? analysis of shared health messages for bio-surveillance. *J Biomed Semant*, 2.

Courtney D. Corley, Diane J. Cook, AR Mikler, and Karan P. Singh. 2010. Text and structural data mining of influenza mentions in web and social media. *International Journal of Environmental Research and Public Health*.

Aron Culotta. 2010. Detecting influenza outbreaks by analysing twitter messages. *CoRR*, abs/1007.4748.

Aron Culotta. 2013. Lightweight methods to estimate influenza rates and alcohol sales volume from twitter messages. *Lang. Resour. Eval.*, 47(1):217–238.

Ernesto Diaz-Aviles, Avaré Stewart, Edward Velasco, Kerstin Denecke, and Wolfgang Nejdl. 2012. Towards personalized learning to rank for epidemic intelligence based on social media streams. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 495–496.

Jeremy Ginsberg, Matthew H Mohebbi, Rajan S Patel, Lynnette Brammer, Mark S Smolinski, and Larry Brilliant. 2009. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–4.

Janaína Gomide, Adriano Veloso, Wagner Meira, Jr., Virgílio Almeida, Fabrício Benevenuto, Fernanda Ferraz, and Mauro Teixeira. 2011. Dengue surveillance based on a computational model of spatio-temporal locality of twitter. In *Proceedings of the International Web Science Conference (WebSci)*, pages 3:1–3:8.

Anette Hulth, Gustaf Rydevik, and Annika Linde. 2009. Web queries as a source for syndromic surveillance. *PLoS ONE*, 4(2):e4378, 02.

Shin Kanouchi, Mamoru Komachi, Naoaki Okazaki, Eiji Aramaki, and Hiroshi Ishikawa. 2015. Who caught a cold ? - identifying the subject of a symptom. In *Proceedings of the Association for Computer Linguistics (ACL)*, pages 1660–1670.

Yoshiaki Kitagawa, Mamoru Komachi, Eiji Aramaki, Naoaki Okazaki, and Hiroshi Ishikawa. 2015. Disease event detection based on deep modality analysis. In *Proceedings of the Association for Computer Linguistics*, pages 28–34.

Yungchang Ku, Chaochang Chiu, Yulei Zhang, Li Fan, and H. Chen. 2010. Global disease surveillance using social media: Hiv/aids content intervention in web forums. In *Intelligence and Security Informatics (ISI), 2010 IEEE International Conference on*, pages 170–170.

Alex Lamb, Michael J. Paul, and Mark Dredze. 2013. Separating fact from fear: Tracking flu infections on twitter. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Vasileios Lampos and Nello Cristianini. 2010. Tracking the flu pandemic by monitoring the social web. In *Proceedings of International Workshop on Cognitive Information Processing (CIP)*, pages 411–416.

Vasileios Lampos, Andrew C Miller, Steve Crossan, and Christian Stefansen. 2015. Advances in nowcasting influenza-like illness rates using search query logs. *Scientific reports*, 5.

Moreno MA, Christakis DA, Egan KG, Brockman LN, and Becker T. 2012. Associations between displayed alcohol references on facebook and problem drinking among college students. *Archives of Pediatrics and Adolescent Medicine*, 166(2):157–163.

M. J. Paul and M. Dredze. 2011. You are what you tweet: Analysing twitter for public health. In *Processing of the International AAAI Conference on Weblogs and Social Media (ICWSM)*.

Michael J Paul, Mark Dredze, and David Broniatowski. 2014. Twitter improves influenza forecasting. *PLOS Currents Outbreaks*.

Philip M. Polgreen, Yiling Chen, David M. Pennock, Forrest D. Nelson, and Robert A. Weinstein. 2009. Using internet searches for influenza surveillance. *Clinical Infectious Diseases*, 47(11):1443–1448.

Kyle W. Prier, Matthew S. Smith, Christophe Giraud-Carrier, and Carl L. Hanson. 2011. Identifying health-related topics on twitter: An exploration of tobacco-related tweets as a test topic. In *Proceedings of the International Conference on Social Computing, Behavioral-cultural Modeling and Prediction (SBP)*, pages 18–25.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: Real-time event detection by social sensors. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 851–860.

Xiao SUN, Jiaqi YE, and Fuji REN. 2014. Real time early-stage influenza detection with emotion factors from sina microblog. In *Proceedings of the Workshop on South and Southeast Asian NLP in the International Conference on Computational Linguistics (COLING)*, pages 80–84.

Robert Tibshirani. 1994. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288.

H. Zou and T. Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 67:301–320.

Task-Oriented Intrinsic Evaluation of Semantic Textual Similarity

Nils Reimers†, Philip Beyer‡, Iryna Gurevych†§
†Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
‡ zeb.rolfes.schierenbeck.associates GmbH
§ Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research
http://www.ukp.tu-darmstadt.de

Abstract

Semantic Textual Similarity (STS) is a foundational NLP task and can be used in a wide range of tasks. To determine the STS of two texts, hundreds of different STS systems exist, however, for an NLP system designer, it is hard to decide which system is the best one. To answer this question, an intrinsic evaluation of the STS systems is conducted by comparing the output of the system to human judgments on semantic similarity. The comparison is usually done using Pearson correlation. In this work, we show that relying on intrinsic evaluations with Pearson correlation can be misleading. In three common STS based tasks we could observe that the Pearson correlation was especially ill-suited to detect the best STS system for the task and other evaluation measures were much better suited. In this work we define how the validity of an intrinsic evaluation can be assessed and compare different intrinsic evaluation methods. Understanding of the properties of the targeted task is crucial and we propose a framework for conducting the intrinsic evaluation which takes the properties of the targeted task into account.

1 Introduction

Semantic Textual Similarity (STS) is the foundational NLP task of determining the degree of semantic similarity between two texts. Most STS systems compute the similarity score between two texts on a fixed scale, for example a scale between 0 and 5, with 0 indicating the semantics are completely independent and 5 indicating semantic equivalence. In recent years, the number and quality of systems that rate the STS between texts have increased, as has the number of tasks where such systems are used.

Textual similarity is an active research field and was part of several shared tasks. In 2012, the pilot *Semantic Textual Similarity (STS) Task* (Agirre et al., 2012) was established at the *Semantic Evaluation (SemEval)* workshop. Further shared tasks on text similarity were part of SemEval 2013 (Agirre et al., 2013), SemEval 2014 (Agirre et al., 2014), SemEval 2015 (Agirre et al., 2015), and SemEval 2016 (Agirre et al., 2016). For the latest shared task on semantic textual similarity at SemEval 2016, 43 teams were submitting 119 different systems, depicting the large interest in this field.

STS is a foundational NLP technique, however, STS systems are seldom used for the sole purpose of measuring the similarity of two texts. Often they are used in a larger context. Examples for such tasks can be found in the field of Automatic Essay Grading (Attali et al., 2006), Plagiarism Detection (Potthast et al., 2012), Automated Text Summarization (Barzilay and Elhadad, 1997), Question Answering (Lin and Pantel, 2001), or Link Discovery (He, 2009). In this paper we call a task that heavily depends on the output of an STS system an *STS based task*. These tasks are often strongly dependent on the quality of the STS system they use, but they might apply further steps as well.

Given this large number of different STS systems, it is hard for an NLP system designer to decide which STS system should be implemented and used for a specific task. As such tasks often strongly depend on the quality of the STS system, the NLP system designer likes to use the most suitable system. To support the NLP system designer in this decision, the quality of STS systems is most often compared in an *intrinsic evaluation*. In the SemEval shared tasks on STS, the participating systems were asked to

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 87–96, Osaka, Japan, December 11-17 2016.

return a continuously valued similarity score given two texts. Performance is assessed by computing the Pearson correlation between machine assigned semantic similarity scores and human judgments (Agirre et al., 2016). Systems with a high Pearson correlation coefficient are considered as "good" STS systems and would often be the first choice for the system designer of an STS based task.

Usage of the Pearson correlation is common practice despite the fact that Agirre et al. (2013) state in the discussion of the results of the SemEval 2013 task about STS: "Evaluation of STS is still an open issue" and that beside the Pearson correlation "... other alternatives need to be considered, depending on the requirements of the target application." Up to our knowledge, no one published so far results whether Pearson correlation is a good method to evaluate the performance of different STS systems.

There are two factors defining the quality of intrinsic evaluations: The used dataset with the human judgments on similarity and the used evaluation measure to compare system outputs with the judgments. In this work, we will concentrate on the used evaluation measure. We studied three STS based tasks with different properties and evaluated 14 different STS systems. As the first task, we selected a classification task on text reuse using the Wikipedia Rewrite Corpus (Clough and Stevenson, 2011), as the second task a binary classification task of news article pairs on their relatedness, and as the third task one on detecting a related news article in a large corpus.

In our three examined tasks, we noticed that the Pearson correlation was misleading and especially ill-suited to predict the best STS system for the task. The performance of the STS systems in the intrinsic evaluation using Pearson correlation had no resemblance to their performance in the three different STS based tasks, i.e. for an NLP system designer the results of the intrinsic evaluation could be discarded. Other evaluation measures were much better in predicting which STS systems will perform well. In our experiments we could not observe that a single evaluation measure consistently produced the best predictions. The requirements on the STS systems for different tasks are too distinct, that a single evaluation measure could cope with all those. We thus claim that understanding the properties of the task and mapping them to the desired properties of the evaluation measure is crucial when selecting a measure for an intrinsic evaluation. Therefore, we propose in section 4 a new framework on the intrinsic evaluation of STS systems by taking the requirements of the target task into account.

This publication is based on the thesis of Beyer (2015). Some details in this paper are ommited for brevity and can be found online.[1]

2 Limitations of the Pearson Correlation and Alternative Evaluation Measures

Figure 1 depicts the output of four hypothetical STS systems in comparison to the gold standard derived from human judgment. These four distributions, also known as Anscombe's quartet, all have the same Pearson correlation coefficient of 0.816. By comparing only the Pearson correlation, all systems would be judged as equally good.

Pearson correlation is especially sensitive to non-linear relations, for example as depicted in the upper-right scatter plot, and to outliers, as depicted in the bottom scatter plots. In the scatter plot in the down-left corner, a single outlier is sufficient to disturb an otherwise perfect correlation. In the down-right corner the opposite is the case, a single outlier is sufficient to produce a high correlation of 0.816 even though there is no relationship between all other outputs and the human judgments. It is obvious that a human would judge the quality of these four STS systems quite differently, even though all four systems achieve the same Pearson correlation coefficient of 0.816.

2.1 Different STS Based Tasks Require Different Evaluation Measures

The usage of the Pearson correlation for the evaluation of STS systems has been questioned before. Zesch (2010) lists the limitations that the Pearson correlation is sensitive to outliers, that it can only measure a linear relationship, and that the two variables need to be approximately normally distributed. To overcome these limitations, Zesch recommends to use Spearman's rank correlation coefficient. The Spearman's rank correlation does not use the actual values to compute a correlation, but the ranking of

[1] https://www.ukp.tu-darmstadt.de/publications/details/?no_cache=1&tx_bibtex_pi1[pub_id]=TUD-CS-2015-12076

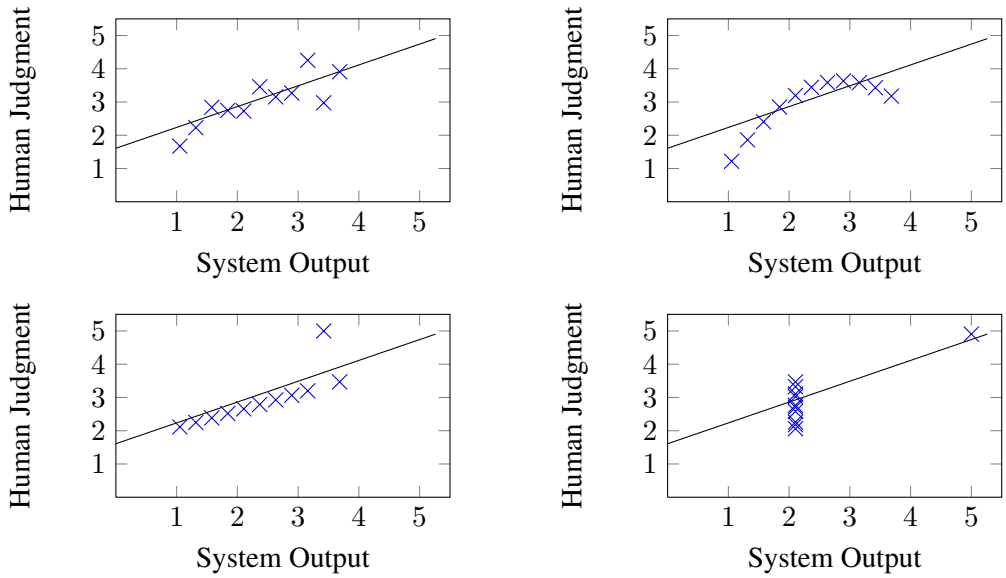


Figure 1: Anscombe's quartet with four different distributions. All distributions have a Pearson correlation coefficient of 0.816. All four STS systems would therefore be considered equally good.

the values. It is therefore not sensitive to outliers, non-linear relationships, or non-normally distributed data. However, most intrinsic evaluations of STS systems only report the Pearson correlation.

Depending on the STS based task, only some characteristics of the STS systems are important. For plagiarism detection, documents are often pre-filtered using an STS system and only documents with a score larger a certain threshold are passed for further inspection. For this task, only the decision whether the score is above the threshold is of importance, less the precise value. For the task of finding the top 10 most similar documents in a corpus for a search query, it is important that the STS system works well in distinguishing similar from dissimilar documents and is able to spot the most similar documents. Working perfectly on dissimilar documents and achieving the gold standard ranking for those is far less important than being able to find the few documents with high similarity. On the other hand, selecting semantically different sentences is important in automatic text summarization, therefore the STS system should work well to detect dissimilar text pairs.

It is unlikely that one evaluation measure can cope well with these different requirements. For Pearson and Spearman's rank correlation for example, all system outputs contribute equally, even though that for several tasks only some scores are relevant, often pairs that are especially similar or especially dissimilar. Using Pearson or Spearman's rank correlation therefore bears the risk that systems working especially well for the desired properties are missed. Hence, we study the following alternative evaluation measures for the intrinsic evaluation:

- The **normalized Cumulative Gain (nCG)** can be used to evaluate the ranking quality of STS scores (Järvelin and Kekäläinen, 2000). Let $\vec{m}$ be the vector with the gold STS values ranked by the STS system highest to lowest, i.e. at position 1 is the most similar pair according to the STS system and m_1 is the human judgment of this pair. The Cumulative Gain at k is defined as $CG_k = \sum_{i=1}^{k} m_i$. To normalize this value, it is divided by the so called ideal Cumulative Gain iCG_k which is the maximal value of CG_k for a perfect system. The normalized Cumulative Gain (nCG) is then defined as $nCG_k = \frac{CG_k}{iCG_k}$. Note that nCG is always equal 1 when k is equal to the number of text pairs.

- The **normalized Discounted Cumulative Gain (nDCG)** applies a discount factor to the normalized Cumulative Gain (Kekäläinen, 2005). It makes the assumption that similar text pairs are more important than less similar text pairs. An STS system which would score well with respect to the nDCG measure is well suited to find the most similar pairs in a corpus while it would be less suited to find the less similar, most distinct pairs. It is defined as $nDCG_k = \frac{DCG_k}{iDCG_k}$ with $DCG_k =$

$m_1 + \sum_{i=2}^{k} m_i / \log_2(i)$ and $iDCG_k$ the ideal Discounted Cumulative Gain. We can compute the $nDCG$ either for all text pairs or up to some position making an especially strong emphasis on the most similar pairs. Note: In case dissimilar pairs are more important for the targeted task, nCG and $nDCG$ can simply be modified by reversing the order of vector $\vec{m}$.

- **Accuracy** is a common evaluation measure for many tasks. However, as the STS scores are continuously valued, it is unclear how to compute it. One option is to define arbitrary bins and check whether the human judgments and the computed STS scores fall in the same bin. This requires that the minimal and maximal value of the STS systems is known and that there is a linear relationship between the STS scores and the human judgments. For the SemEval shared tasks, the systems were supposed to produce an output between 0 and 5 identical to the scale for the human judgments. In this work we set arbitrary borders at 1.5 and 3.5 with the intention to detect pairs with high similarity, e.g. for plagiarism detection, or pairs with low similarity, e.g. for detecting distinct sentences to be used in summarization. Accuracy^{low} describes the accuracy for scores below 1.5, and Accuracy^{high} describes the accuracy for scores higher than 3.5.

- Besides accuracy, the **$\mathbf{F_1}$-score** is a commonly used measure in several NLP tasks. Similar to accuracy, we have the challenge to define meaningful bins. As before, we set arbitrary borders at 1.5 and 3.5. F_1^{low} describes the F_1-score for low similarity pairs with a score below 1.5, and F_1^{high} describes the F_1-score for high similarity pairs with a score higher than 3.5.

Besides these evaluation measures, we define further measures by combining those using the harmonic mean $hmean(a, b) = 2ab/(a + b)$ and the unweighted macro average $macro\_avg(a, b) = (a + b)/2$. We also define the measure nDCG$_{AvgRank}$ which is the average of nDCG$_3$, nDCG$_5$, and nDCG$_{10}$ and nCG$_{AvgRank}$ which is the average of nCG$_3$, nCG$_5$, and nCG$_{10}$. These two measures only evaluate the top 3, 5, and 10 most similar text pairs according to the STS systems.

2.2 Impact of the Evaluation Measure on the Ranking

The evaluation measure of the intrinsic evaluation of STS systems can have a huge impact on the ranking of different STS systems. For the SemEval 2012 shared task on Semantic Textual Similarity (Agirre et al., 2012), 88 different STS systems were submitted by the participants. The official evaluation measure of this shared task used the Pearson correlation. We took the predicted STS scores and ranked the systems according to the alternative evaluation measures described in section 2. The results are depicted in Table 1. Using Spearman's rank correlation instead of Pearson correlation changed the positions of the STS systems on average by 6.6. The largest observed difference was 21 positions, i.e. a system that performed quite well according to the Pearson correlation coefficient achieved a mediocre result when the Spearman rank correlation coefficient is used. An even larger difference was observed when comparing Pearson correlation to the other presented evaluation measure like nDCG. The ranking of the STS systems according to the Pearson correlation was on average 19.0 positions different than the ranking according to nDCG.

The results show that the used evaluation measure plays an important role in defining the ranking of different STS systems in the intrinsic evaluation. A system which is assessed to be good using one evaluation measure, for example Pearson correlation, might perform badly according to another evaluation measure, for example Spearman's rank correlation.

3 Evaluation of the Predictiveness of Different Evaluation Measures

When the STS system is a crucial component of an STS based task, we expect that STS systems achieving good results in an intrinsic evaluation should in general lead to better results in the task and STS systems with weak results should in general lead to worse results for the STS based task. This allows us to define how *predictive* an intrinsic evaluation is:

> *Given the ranking of STS systems in an intrinsic evaluation as well as the ranking of the systems in an extrinsic evaluation, we say the intrinsic evaluation has **high predictiveness** when the*

Mean Absolute Difference	Spearman	nDCG$_{all}$	nDCG$_{AvgRank}$	Accuracy	F_1
Pearson	6.6	19.0	29.4	12.5	12.5
Spearman	-	19.6	29.1	12.0	15.2
nDCG$_{all}$		-	20.6	21.1	20.8
nDCG$_{AvgRank}$			-	31.8	26.3
Accuracy				-	14.3

Table 1: Mean Absolut Difference between ranks of submissions for the shared task on Semantic Textual Similarity at SemEval 2012 using different evaluation measures.

> *ranking in the intrinsic evaluation is similar to the ranking in the extrinsic evaluation. We say the **predictiveness** is **low**, when the ranking in the intrinsic evaluation is disconnected from the ranking of the systems in the extrinsic evaluation.*

In an ideal situation, the ranking of the STS systems in the intrinsic evaluation would be identical to the ranking in the extrinsic evaluation. As we cannot expect this, we need to define an objective measure on how similar these two rankings are. We can measure the resemblance of two rankings using the Mean Absolute Difference (MAD), the Mean Squared Difference (MSD), or the Spearman's rank correlation coefficient ρ. Given the rankings IR of the n STS systems in the intrinsic evaluation and the rankings ER in the extrinsic evaluation with IR_i corresponding to the rank of the i-th STS system in the intrinsic evaluation and ER_i corresponding to the rank in the extrinsic evaluation, the values are defined as:

$$\rho(IR, ER) = \frac{\text{cov}(IR, ER)}{\sigma_{IR}\sigma_{ER}}$$

$$\text{MSD}(IR, ER) = \frac{1}{n}\sum_{i=1}^{n}(IR_i - ER_i)^2$$

$$\text{MAD}(IR, ER) = \frac{1}{n}\sum_{i=1}^{n}|IR_i - ER_i|$$

where $\text{cov}(IR, ER)$ is the covariance of the rankings and σ_{IR}, σ_{ER} are the standard deviations of the rank variables. An intrinsic evaluation with high predictiveness would have MAD and MSD values close to 0 and a Spearman's correlation ρ close to 1. We would consider an intrinsic evaluation of STS system useful, when it scores well on MAD, MSD, and ρ values for a large range of STS based tasks.

3.1 Experiments

To assess the predictiveness of different evaluation measures presented in section 2, we chose three STS based tasks and evaluated 14 different STS systems. We used the implementation for these systems from the publically available framework DKPro Similarity[2]. For each STS system, we computed the score in an intrinsic evaluation. For the intrinsic evaluation, we used the datasets provided for the SemEval 2012 task on Semantic Textual Similarity (Agirre et al., 2012). The intrinsic evaluation was performed for 16 different evaluation measures described in section 2. We then compared the ranking of the different STS systems in the intrinsic evaluation with the ranking of those in the STS based task, which allows to compute the *predictiveness* of the (intrinsic) evaluation measure.

As our first STS based task, we chose the task of text reuse detection. Clough and Stevenson (2011) presented the Wikipedia Rewrite Corpus, a dataset with 95 documents, each containing an answer to one

[2] `https://dkpro.github.io/dkpro-similarity/`

of five questions about computer science. The answers employ different levels of reuse of a Wikipedia article. The degree of reuse was split in one of four categories: `near copy`, `light revision`, `heavy revision`, and `non plagiarized`. The performance for this task is evaluated by calculating the accuracy. To map the continuous output of the STS systems to the four categories, we used the One Rule (OneR) classifier (Holte, 1993) with optimized bucket sizes as well as a logistic regression classifier. The OneR classifier chooses a simple decision boundary for the different classes. Both have been evaluated using 10-fold cross-validation and the classifier with the better result was chosen.

The second and third STS based task uses a newly created corpus compiled from the German newspaper DIE ZEIT and ZEIT Online[3]. For most of the articles, the authors added two links to related articles that provide further information on the same news topic. The second task is a binary classification task with the goal to identify whether two articles are related or not. The ground truth is the original choice from the journalist. The OneR classifier was used to map the continuous STS score to the binary decision. Results were evaluated using 10-fold cross-validation.

The third STS based task tries to detect the two articles that are related to the target article in a set of articles from ZEIT Online. For the target article and each article in the set, we compute the STS score. The article in the set with the highest STS score was selected. We compared if this article is one of the related articles chosen by the author. Accuracy was computed for 100 randomly selected documents.

3.2 Results

We evaluated 14 different STS systems for the three presented tasks. For the first STS based task on text reuse detection, the best STS system achieved an accuracy of 70%, while the worst achieved an accuracy of 43%. For the second STS based task on deciding whether two news articles are related, the best STS system achieved an accuracy of 77%, while the worst achieved an accuracy of 48%. And for the third STS based task on finding the related article out of a set of articles, the best STS system achieved an accuracy of 67%, while the worst achieved an accuracy of 6%.

We compared the performance of the different STS systems in the three tasks with their performance in the intrinsic evaluation. We expect that the intrinsic evaluation allows us to distinguish between well performing STS systems and bad performing STS systems. Table 2 shows the Spearman ranking coefficient $\rho(IR, ER)$ between the performance of the STS measures in the intrinsic evaluation versus their performance on the STS based tasks. A coefficient close to 1 indicates that the ranking of the system in the intrinsic evaluation was similar to its performance in the STS based task. The values for the two other predictiveness indicators, *Mean Absolute Difference* and *Mean Squared Difference* can be found are nearly identical to the Spearman rank coefficient. Thus, we omit them for brevity.

3.3 Discussion

In the three studied STS based tasks, there was no correlation between the performance of STS systems on the intrinsic evaluation using Pearson or Spearman rank correlation and their performance in the STS based tasks. In two cases, the correlation $\rho(IR, ER)$ between the intrinsic ranking IR and extrinsic ranking ER was even negative, indicating that STS systems that performed well in the intrinsic evaluation performed especially poorly in STS based tasks. From an engineering perspective this raises serious doubts about the value of an intrinsic evaluation that uses Pearson correlation.

Using other measures than Pearson correlation for the intrinsic evaluation however enabled a much better prediction of the performance of the STS systems for the STS based task. A strong STS system in such an intrinsic evaluation was also able to perform well in the STS based task. It is interesting to note that other STS based tasks were especially good predictors, i.e. an STS system performing well in task 1 was also performing well in task 2 and task 3, even though the characteristics of these tasks were very distinct. In all three tasks, the same STS system achieved the best result. However, in none of the performed intrinsic evaluations achieved this system the best place and was placed on 2nd to 6th place depending on the used evaluation measure. The system that achieved the best place in various intrinsic

[3] `http://www.zeit.de`

Intrinsic Evaluation Measure	Task 1		Task 2		Task 3	
	ρ	Rank	ρ	Rank	ρ	Rank
nDCG$_{AvgRank}$	0.504	1	0.380	6	0.338	5
nCG$_{AvgRank}$	0.504	1	0.380	6	0.338	5
F_1^{low}	0.497	3	0.717	2	0.611	2
nDCG	0.431	4	0.238	10	0.238	8
hmean(F_1^{low}, F_1^{high})	0.427	5	0.722	1	0.614	1
hmean(Pearson,F_1)	0.264	6	0.686	3	0.536	3
hmean(Spearman,F_1)	0.163	7	0.594	4	0.439	4
macro_avg(F_1^{low}, F_1^{high})	-0.053	8	0.422	5	0.289	7
hmean(Pearson, nCG$_{AvgRank}$)	-0.136	9	0.339	8	0.130	9
hmean(Spearman, nCG$_{AvgRank}$)	-0.216	10	0.277	9	0.089	10
Accuracy^{low}	-0.277	11	0.057	14	-0.062	13
Pearson correlation	-0.326	12	0.198	11	-0.031	11
Spearman's rank correlation	-0.343	13	0.172	12	-0.040	12
hmean(Accuracy^{low},Accuracy^{high})	-0.370	14	0.031	15	-0.113	15
Accuracy^{high}	-0.378	15	0.062	13	-0.102	14
F_1^{high}	-0.524	16	-0.123	16	-0.283	16
Task 1: Text reuse classification			0.70		0.82	
Task 2: Binary classification of article pairs	0.70				0.91	
Task 3: Related article detection	0.82		0.91			

Table 2: The Spearman rank correlation $\rho(IR, ER)$ between the intrinsic ranking IR and the extrinsic ranking ER for the three evaluated STS based tasks: (1) Text reuse classification, (2) binary classification of article pairs, and (3) related article detection. A ρ-coefficient close to 1 means a large correlation between the performance in the intrinsic evaluation and the performance in the STS based task. The *Rank* depicts the ranking, highest to lowest, of the ρ-coefficients for each task.

evaluations performed quite poorly for the STS based tasks only achieving the 6th, 9th, and 12th place, respectively, out of 14 tested systems.

4 Proposal of an Evaluation Framework for Semantic Textual Similarity

On a well-designed and representative dataset, an STS system should show similar behavior in the intrinsic evaluation as it will show for real world data of STS based tasks. For example, in case the STS system is well suited to find the most similar text pairs in the intrinsic evaluation set, then it will likely also be suitable to find the most similar text pairs for other datasets. This STS system would then be useful for tasks where finding the most similar text pairs is essential.

However, different STS based tasks have different requirements on STS systems and, therefore, different properties of STS systems are important. After studying the most common STS based tasks, we propose the following three dimensions to classify the requirements of an STS based task:

- **Cardinality** describes how many texts are compared to how many others. It consists of two sub categories *1:1* and *1:n*. *1:1* in this context means that exactly one text is compared with exactly one other text and only the result of this single comparison is of interest. *1:n* means that one text will be compared with a whole set of other texts and the results of these comparisons will be used in some way. The third option, *m:n*, would theoretically be possible, but no example of this was found.

- **Set of Interest** describes which of the elements of the result set will be used. It has three sub categories: *All*, *k-best*, and *Threshold*. *All* in this context means that all results of all comparisons will be used in some form. *k-best* describes the case where only the "k" best results will be used in some way. And *Threshold* is used when only results over a certain threshold will be used.

- **Information** describes the type of information from the result set that is of interest. It has three sub categories: *Value*, *Rank*, and *Classification*. The case where the actual value of the result of a comparison is of interest falls in the category *Value*. *Rank* on the other hand is used if only the rank of each comparison is used in some way. *Classification* means that a simple classification, for example texts are similar or not, is used.

The first STS based task in section 3.1 on text reuse is an example for a task with cardinality *1:1*, as one text, the answer, is compared to only one other text, the Wikipedia article. The STS score is classified into one of four categories, hence, the *Information* of interest is *Classification*. Tasks of such type can only tolerate minimal variety in the STS scores, i.e. texts of similar similarity should be mapped to similar scores independent of other factors like text length etc. Otherwise, the classification into categories doesn't work well.

The third STS based task in section 3.1 on detecting the related articles in a set of articles is an example for a task with cardinality *1:n*, set of interest *k-best* and information *Rank*. For this task, one document is compared to a set of other documents and the user is interested in the most similar pair. STS systems for this task should be good at ranking text pairs according to their similarity.

With the three dimensions 18 different combinations are possible. However, some of these combinations can be disregarded because they are not plausible. For any combination that involves a *Cardinality* of *1:1* only a *Set of Interest* of *All* is useful, because the result set contains only one result. In addition, the *Information* can't be *Rank*. Overall, only nine combinations are plausible. All nine possible combinations with examples of STS based tasks are described in detail in (Beyer, 2015).

For these nine plausible combinations, we propose in Table 3 an evaluation measure for the intrinsic evaluation that should capture the requirements of the target task. The proposed evaluation measures take similar characteristics into account that are required for the task. An NLP system designer could use this framework to determine the requirements of his task. Instead of selecting the STS system with the best Pearson correlation, the system designer would use the selected evaluation measure to run his own ranking of the STS systems to spot potentially strong STS systems for his task. The reasoning for the individual choices is given in (Beyer, 2015).

Requirements	**Proposed Evaluation Measure for Intrinsic Evaluation**
(1:1, All, Classification)	harmonic mean of F_1-score for low and high similarity pairs
(1:1, All, Value)	Pearson correlation
(1:n, All, Rank)	nDCG or Spearman rank correlation
(1:n, All, Classification)	harmonic mean of F_1-score for low and high similarity pairs
(1:n, All, Value)	Pearson correlation
(1:n, k-best, Value)	harmonic mean of nCG_k and Pearson correlation
(1:n, k-best, Rank)	nDCG_k
(1:n, Threshold, Value)	harmonic mean of F_1-score for low and high similarity pairs and Pearson correlation
(1:n, Threshold, Rank)	harmonic mean of F_1-score for low and high similarity pairs and Spearman rank correlation

Table 3: This Semantic Textual Similarity Framework proposes an evaluation measure for intrinsic evaluation based on the requirements of the target task.

An extensive evaluation of this framework is topic of our future research. The focus of this paper was establishing the need of alternative evaluation measures besides Pearson correlation and how to assess the quality of intrinsic evaluations. We encourage future work by others on this topic to find an intrinsic evaluation that meets the diverse needs of STS based tasks.

5 Conclusion and Future Work

In this paper we demonstrated the challenges of the intrinsic evaluation of STS systems. We introduced the concept of *predictiveness*: An STS system performing well in an intrinsic evaluation should also perform well for STS based tasks. This notion of predictiveness allows us to compare different evaluation measures besides the commonly used Pearson correlation. For three studied tasks we could observe that the predictiveness of an intrinsic evaluation with Pearson correlation is fairly low or even negative. We presented other evaluation measures which had a much higher predictiveness, i.e. those methods could predict much better which STS systems perform well in the STS based tasks. Based on this, we proposed a framework how to evaluate STS scores that take the requirements of the target task into account.

For future intrinsic evaluations of STS systems we find it crucial that not only the Pearson correlation is published, but additionally the STS scores generated by the systems can be downloaded. This allows to compute other evaluation measures, for example Spearman's ranking correlation, nDCG, or F_1-score. It also allows NLP system designers to select an evaluation measure for the intrinsic evaluation that captures the important characteristics needed by their target task.

In our experiments we could observe that the predictiveness of other extrinsic evaluations is high. Systems performing well on the task of text reuse of English Wikipedia articles also did well for detecting related articles on German news articles despite the fact of a different language, a different text genre and a completely different task. An alternative evaluation method of STS systems could be to test those on a broad range of different STS based tasks. The design of these STS based tasks must be standardized and the impact of components or features besides the STS system should be reduced to a minimum.

Acknowledgements

This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1, by the German Institute for Educational Research (DIPF) and by the German Federal Ministry of Education and Research (BMBF) as a part of the Software Campus program under the promotional reference 01-S12054.

References

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 385–393, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic Textual Similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263, Denver, Colorado, June. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation. In *Proceedings of SemEval-2016*, pages 509–523, San Diego, California, June. Association for Computational Linguistics.

Yigal Attali, Jill Burstein, Yigal Attali, Jill Burstein, Michael Russell, Design Thomas Hoffmann, Yigal Attali, and Jill Burstein. 2006. Automated Essay Scoring with E-Rater V.2. *Journal of Technology, Learning, and Assessment*.

Regina Barzilay and Michael Elhadad. 1997. Using Lexical Chains for Text Summarization. In *In Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, pages 10–17.

Philip Beyer. 2015. Proposal for a STS Evaluation Framework for STS based Applications. Master's thesis, Technische Universität Darmstadt, March. Available online: `https://www.ukp.tu-darmstadt.de/publications/details/?no_cache=1&tx_bibtex_pi1[pub_id]=TUD-CS-2015-12076`.

Paul Clough and Mark Stevenson. 2011. Developing a corpus of plagiarised short answers. *Language Resources and Evaluation*, 45(1):5–24.

Jiyin He, 2009. *Link Detection with Wikipedia*, pages 366–373. Springer Berlin Heidelberg, Berlin, Heidelberg.

Robert C. Holte. 1993. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine Learning*, 11(1):63–90.

Kalervo Järvelin and Jaana Kekäläinen. 2000. IR Evaluation Methods for Retrieving Highly Relevant Documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, pages 41–48, New York, NY, USA. ACM.

Jaana Kekäläinen. 2005. Binary and Graded Relevance in IR evaluations-Comparison of the Effects on Ranking of IR Systems. *Inf. Process. Manage.*, 41(5):1019–1033, September.

Dekang Lin and Patrick Pantel. 2001. Discovery of Inference Rules for Question-answering. *Nat. Lang. Eng.*, 7(4):343–360, December.

Martin Potthast, Tim Gollub, Matthias Hagen, Johannes Kiesel, Maximilian Michel, Arnd Oberländer, Martin Tippmann, Alberto Barrón-Cedeño, Parth Gupta, Paolo Rosso, and Benno Stein. 2012. Overview of the 4th International Competition on Plagiarism Detection. In *CLEF 2012 Evaluation Labs and Workshop, Online Working Notes, Rome, Italy, September 17-20, 2012*.

Torsten Zesch. 2010. *Study of Semantic Relatedness of Words Using Collaboratively Constructed Semantic Resources*. Ph.D. thesis, Technische Universität, Darmstadt.

Expanding wordnets to new languages with multilingual sense disambiguation

Mihael Arcan **John P. McCrae** **Paul Buitelaar**

Insight Centre for Data Analytics, National University of Ireland, Galway
firstname.lastname@insight-centre.org

Abstract

Princeton WordNet is one of the most important resources for natural language processing, but is only available for English. While it has been translated using the *expand* approach to many other languages, this is an expensive manual process. Therefore it would be beneficial to have a high-quality automatic translation approach that would support NLP techniques, which rely on WordNet in new languages. The translation of wordnets is fundamentally complex because of the need to translate all senses of a word including low frequency senses, which is very challenging for current machine translation approaches. For this reason we leverage existing translations of WordNet in other languages to identify contextual information for wordnet senses from a large set of generic parallel corpora. We evaluate our approach using 10 translated wordnets for European languages. Our experiment shows a significant improvement over translation without any contextual information. Furthermore, we evaluate how the choice of pivot languages affects performance of multilingual word sense disambiguation.

1 Introduction

Princeton WordNet (Fellbaum, 1998) is a manually created resource that has been used in many different tasks and applications across linguistics and natural language processing. WordNet's hierarchical structure makes it a useful tool for many semantic applications and it also plays a vital role in modern deep learning based NLP systems (Rychalska et al., 2016). However, Princeton WordNet is only available for English and huge efforts have been made to extend WordNet with multilingual information in projects, such as EuroWordNet (Vossen, 1998), BalkaNet (Tufiş et al., 2004) and MultiWordNet (Pianta et al., 2002). However, most of the wordnet resources resulting from these efforts have fewer synsets than the Princeton WordNet and there are still many languages for which a wordnet does not exist or is not available to all potential users due to licensing restrictions, impacting applications in information retrieval, word sense disambiguation, sentiment analysis or knowledge management that rely on Princeton WordNet.

Most wordnets in languages other than English have followed an *extend* approach (Vossen, 2005), where the structure of Princeton WordNet is preserved and only the words in each synset are translated and new synsets are added for concepts, which are not lexicalized in English. Since manual multilingual translation and evaluation of wordnets using this approach is a very time consuming and expensive process, we apply statistical machine translation (SMT) to automatically translate WordNet entries. While an SMT system can only return the most frequent translation when given a term by itself, it has been observed that SMT provides strong word sense disambiguation when the word is given in the context of a sentence. As a motivating example, we consider the word *vessel*, which is a member of three synsets in Princeton WordNet, whereby the most frequent translation, e.g., as given by Google Translate, is *Schiff* in German and *nave* in Italian, corresponding to `i60833`[1] 'a craft designed for water transportation'. For the second sense, `i65336` 'a tube in which a body fluid circulates', we assume that we know the

[1] We use the CILI identifiers for synsets (Bond et al., 2016)

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 97–108, Osaka, Japan, December 11-17 2016.

German translation for this sense is *Gefäß*. In our approach we look for sentences in a parallel corpus, where the words *vessel* and *Gefäß* both occur and obtain a context such as 'blood vessel' that allows the SMT system to translate this sense correctly. This alone is not sufficient as *Gefäß* is also a translation of `i60834` 'an object used as a container', however in Italian these two senses are distinct (*vaso* and *recipiente* respectively), thus by using as many languages as possible we maximize our chances of finding a well disambiguated context.

In this work, we propose an approach to select the most relevant sentences from a parallel corpus based on the overlap with existing translations of WordNet in as many pivot languages as possible. The goal is to identify sentences that share the same semantic information in respect to the synset of the WordNet entry that we want to translate. This approach will allow us to provide a large multilingual WordNet in more than 20 different European languages, which we call Polylingual WordNet.[2] We present multiple evaluations of our approach and show that in general at least 4 languages should be used to assist in the selection of contexts and that languages closely related to the target language should be used in preference to more distant languages. We evaluated our approach on translating WordNet entries into Italian, Slovene, Spanish and Italian, showing improvements between 5 and more than 10 BLEU points compared to a generic translation approach. This approach has been used to expand wordnets for many European languages as well as generate the first wordnet for Maltese.

2 Related Work

Princeton WordNet inspired many researchers to create similarly structured wordnets for other languages. The EuroWordNet project (Vossen, 1998) linked wordnets in different languages through a socalled Inter-Lingual-Index (ILI) into a single multilingual lexical resource. Via this index, the languages are aligned between each other, which allows to go from a concept in one language to a concept with a similar meaning in any of the other languages. Further multilingual extensions were generated by the BalkaNet project (Tufiş et al., 2004), focusing on the Balkan languages and MultiWordNet (Pianta et al., 2002), aligning Italian concepts to English equivalents.

Due to the large interest in the multilingual extensions of the Princeton WordNet, several initiatives started with the aim to unifying and making these wordnets easily accessible. The KYOTO project (Fellbaum and Vossen, 2012) focused on the development of a language-independent module to which all existing wordnets can be connected, which would allow a better cross-lingual machine processing of lexical information. Recently this has been realized by a new Global WordNet Grid (Vossen et al., 2016) that takes advantage of the Collaborative Inter-Lingual Index (CILI) (Bond et al., 2016). Since most of the current non-English wordnets use the Princeton WordNet as a pivot resource, concepts, which are not in this English lexical resource cannot not be realized or aligned to it. Therefore the authors support the idea of a central platform of concepts, where new concepts may be added even if they are not represented (yet) in the Princeton WordNet or even lexicalized in English (e.g., many languages have distinct gendered role words, such as 'male teacher' and 'female teacher', but these meanings are not distinguished in English).

Previous studies of generating non-English wordnets combined Wiktionary knowledge with existing wordnets to extend them or to create new ones (de Melo and Weikum, 2009). Bond and Paik (2012) describe in their work the creation of the Open Multilingual Wordnet and its extension with other resources (Bond and Foster, 2013). A different approach to expand English WordNet synsets with lexicalizations in other languages was proposed in de Melo and Weikum (2012). The authors do not directly match concepts in the two different language resources, but demonstrate an approach that learns how to determine the best translation for English synsets by taking bilingual dictionaries, structural information of the English WordNet and corpus frequency information into account. With the growing amount of parallel data, Kazakov and Shahid (2009) show an approach to acquire a set of synsets from parallel corpora. The synsets are obtained by comparing aligned words in parallel corpora in several languages. Similarly, the sloWNet for Slovene (Fišer, 2007) and Wolf for French (Sagot and Fišer, 2008) are constructed using a multilingual corpus and word alignment techniques in combination with other existing lexical resources.

[2]The Polylingual WordNet is available at `http://polylingwn.linguistic-lod.org/`

Since all these approaches use word alignment information, they are not able to generate any translation equivalents for multi-word expressions (MWE). In contrast, our approach use an SMT system trained on a large amount of parallel sentences, which allows us to align possible MWEs, such as *commercial loan* or *take a breath*, between source and target language. Furthermore, we engage the idea of identifying relevant contextual information to support an SMT system translating short expressions, which showed better performance compared to approaches without a context. Arcan et al. (2015) built small domain-specific translation models for ontology translation from relevant sentence pairs that were identified in a parallel corpus based on the ontology labels to be translated. With this approach they improve the translation quality over the usage of large generic translation models. Since the generation of translation models can be computational expensive, Arcan et al. (2016) use large generic translation models to translate ontology labels, which were placed into a disambiguated context. With this approach the authors demonstrate translation quality improvement over commercial systems, like *Microsoft Translator*. Different from this approach, which uses the hierarchical structure of the ontology for disambiguation, we engage a large number of different languages to identify the relevant context.

Oliver and Climent (2012) present a method for WordNet construction and enlargement with the help of sense tagged parallel corpora. Since parallel sense tagged data are not always available, they use *Google Translate* to translate a manually sense tagged corpus. In addition they apply automatic sense tagging of a manually translated parallel corpus, whereby they report worse performance compared to the previous approach. We try to overcome this issue by engaging up to ten languages to improve the performance of the automatic sense tagging. Similarly, BabelNet (Navigli and Ponzetto, 2012) aligns the lexicographic knowledge from WordNet to the encyclopaedic knowledge of Wikipedia. This is done by assigning WordNet synsets to Wikipedia entries, and making these relations multilingual through the interlingual links. For languages, which do not have the corresponding Wikipedia entry, the authors use *Google Translate* to translate English sentences containing the synset in the sense annotated corpus. After that, the most frequent translation is included as a variant for the synset for the given language.

The use of parallel corpora has been previously exploited for word sense disambiguation, for example to construct sense-tagged corpora in another language (Ng et al., 2003) or by using translations as a method to discriminate senses (Ide et al., 2002). It has been shown that the combination of these techniques can improve supervised word sense disambiguation (Chan et al., 2007). A similar approach to the one proposed in this paper is that of Tufiş et al. (2004), where they show that using the interlingual index of WordNet with the help of parallel text can improve word sense disambiguation of a monolingual approach and we generalize this result to generate wordnets for new languages.

3 Methodology

Our approach takes the advantage of the increasing amount of parallel corpora in combination with wordnets in languages other than English for sense disambiguation, which will help us to improve automatic translations of English WordNet entries. We assume that we have a multilingual parallel corpus consisting of sentences, x_i^l in a language l, grouped into parallel translations:

$$\mathcal{X} = \{(x_i^{l_0}, \ldots, x_i^{l_T})\}$$

We also assume that we have a collection of wordnets consisting of a set of senses, w_{ij}^l, grouped into synsets, for each language:

$$\mathcal{S} = \{(\{w_{ij}^{l_0}\}, \ldots, \{w_{ij}^{l_T}\})\}$$

We say that a context $x_i^{l_0}$, in language l_0 (in our case this is always English), is *disambiguated in* n *languages* for a word $w_{jk}^{l_0}$ if:

$$\exists w_{jk_1}^{l_1}, \ldots, w_{jk_n}^{l_n} : w_{jk_1}^{l_1} \in x_i^{l_1} \wedge \ldots \wedge w_{jk_n}^{l_n} \in x_i^{l_n}$$

That is, a context is disambiguated in n languages for a word, if for each of its translations we have a context in the parallel corpus that contains one of the known synset translations. Furthermore, we assume

we have an SMT system that can translate any context in l_0 into our target language, l_T, and produces a phrase alignment such that we know which word in the output corresponds to the input word. We used the following methods to choose contexts for the SMT system:

None The SMT system is given only the word $w_{jk}^{l_0}$ as a single sentence as input, thus the most frequent translation is returned.

Random context A random $x_i \in \mathcal{X}$, such that $w_{jk}^{l_0} \in x_i^{l_0}$, is chosen.

Disambiguated context The contexts are ordered by the number of languages that they are disambiguated in, and the context that is disambiguated in the maximal number of languages is chosen. If there are multiple such languages, one context is chosen at random.

m **Disambiguated contexts** The contexts are ordered, as above, and the m top scoring contexts are used, with ties broken at random. Each of these contexts is given to the SMT system and the most frequent translation across these m contexts is used. The previous mode is the same as this when $m = 1$.

t**-best Translations** The SMT system is configured to return the t highest scoring translations, according to its model, and we select the translation as the most frequent translation of the context among this t-best list. In our experiments, we combined this with m disambiguations to give tm candidate translations from which the candidate is chosen.

Target Side Lookup (TSL) We can also utilize the translation of our context into the target language $x_i^{l_T}$ from the parallel corpus, however this cannot be applied directly as we do not know which word(s) in $x_i^{l_T}$ correspond to the input and previous work (Arcan et al., 2014) has shown that automatic inference of this alignment (e.g., with GIZA++) can seriously affect performance. Instead we filter contexts to those that generate a translation candidate, $w_k^{l_T}$, such that $w_k^{l_T} \in x_i^{l_T}$, i.e., the machine translation agrees with the gold-standard translation for this context.

4 Experimental Setting

This section gives an overview on the multilingual resources and the translation toolkit used in our experiment. Furthermore, we give insights into SMT evaluation techniques, considering the translation direction of the English WordNet entries into Italian, Slovene, Spanish and Croatian.

4.1 Wordnets for Sense Disambiguation in Parallel Corpora

Princeton WordNet is a large, publicly available lexical semantic database of English nouns, verbs, adjectives and adverbs, grouped into synsets ($\approx$ 117,000). We engage further wordnets in a variety of languages, provided by the Open Multilingual Wordnet web page.[3] The individual wordnets have been made by many projects and we use ten wordnets in different languages for our experiments, i.e, Croatian (Oliver et al., 2015), Dutch (Postma et al., 2016), Finnish (Lindén and Carlson., 2010), French (Sagot and Fišer, 2008), Italian (Toral et al., 2010), Polish (Maziarz et al., 2012), Portuguese (de Paiva and Rademaker, 2012), Romanian (Tufiş et al., 2008), Slovene (Fišer et al., 2012) and Spanish (Gonzalez-Agirre et al., 2012) WordNet. Table 1 illustrates the size of the wordnets and their coverage compared to the Princeton WordNet (last row).[4]

4.2 Statistical Machine Translation

Our approach is based on phrase-based SMT (Koehn et al., 2003), where we wish to find the best translation of a string, given by a log-linear model combining a set of features. The translation that maximizes the score of the log-linear model is obtained by searching all possible translations candidates.

[3] `http://compling.hss.ntu.edu.sg/omw/`

[4] Core refers to the percentage of synsets covered from the semi-automatically compiled list of 5000 "core" word senses in Princeton WordNet.

Language	Synsets	Words	Senses	Core	Language	Synsets	Words	Senses	Core
Croatian	23,120	29,008	47,900	100%	Polish	33,826	45,387	52,378	54%
Dutch	30,177	43,077	60,259	67%	Portuguese	43,895	54,071	74,012	84%
Finnish	116,763	129,839	189,227	100%	Romanian	56,026	49,987	84,638	94%
French	59,091	55,373	102,671	92%	Slovene	42,583	40,233	70,947	86%
Italian	35,001	41,855	63,133	83%	Spanish	38,512	36,681	57,764	76%

Table 1: Statistics on used wordnets for sense disambiguation on parallel corpora.

Parallel Corpus (language pair)	Source Words	Target Words	Parallel Sentences	Parallel Corpus (language pair)	Source Words	Target Words	Parallel Sentences
English–Croatian[1,2]	165M	133M	16M	English-Polish[2]	361M	296M	34M
English–Dutch[2]	426M	372M	37M	English–Portuguese[2]	391M	377M	33M
English–Finnish[2]	248M	165M	25M	English–Slovene[1,2]	166M	130M	13M
English–French[2]	730M	784M	52M	English–Spanish[1,2]	391M	378M	37M
English–Italian[1,2]	273M	270M	22M	English–Romanian[2]	317M	302M	43M

Table 2: Statistics on parallel data for translation model training and word-sense disambiguation. (parallel resources used for training the translation models[1] and/or word-sense disambiguation[2])

The decoder, which is a search procedure, provides the most probable translation based on a statistical translation model learned from the training data.

For our translation task, we use the statistical translation toolkit Moses (Koehn et al., 2007), where word alignments, necessary for generating translation models, were built with the GIZA++ toolkit (Och and Ney, 2003). The Kenlm toolkit (Heafield, 2011) was used to build a 5-gram language model.

4.3 Parallel Resources for SMT training and Word-Sense-Disambiguation

To ensure a broad lexical and domain coverage of our SMT system we merged the existing parallel corpora for each language pair from the OPUS web page[5] into one parallel data set, i.e., Europarl (Koehn, 2005), DGT - translation memories generated by the *Directorate-General for Translation* (Steinberger et al., 2014), MultiUN corpus (Eisele and Chen, 2010), EMEA, KDE4, OpenOffice (Tiedemann, 2009), OpenSubtitles2012 (Tiedemann, 2012). Similarly, we concatenate parallel corpora for identifying relevant sentences containing WordNet entries, which are then translated into the targeted languages. Table 2 shows the number of parallel sentences used for the ten language pairs.

4.4 Translation Evaluation Metrics

The automatic translation evaluation is based on the correspondence between the SMT output and reference translation (gold standard). For the automatic evaluation we used the BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014) and chrF (Popović, 2015) metrics. **BLEU** (Bilingual Evaluation Understudy) is calculated for individual translated segments (n-grams) by comparing them with a data set of reference translations.[6] The calculated scores, between 0 and 100 (perfect translation), are averaged over the whole *evaluation data set* to reach an estimate of the translation's overall quality. Considering the short length of the terms in WordNet, while we report scores based on the unigram overlap (BLEU-1), this is in most cases only precision, so in addition we also report other metrics. **METEOR** (Metric for Evaluation of Translation with Explicit ORdering) is based on the harmonic mean of precision and recall, whereby recall is weighted higher than precision. Along with exact word (or phrase) matching it has additional features, i.e. stemming, paraphrasing and synonymy matching. In contrast to

[5] `http://opus.lingfil.uu.se/index.php`

[6] Due to the possibility of including multiple references for evaluation within the BLUE metric, we use the set of target words within a synset as our gold standard.

	types		tokens	
	Num.	Perc.	Num.	Perc.
English-Italian	507	6.3	521	4.2
English-Spanish	396	4.9	406	3.3
English-Slovene	633	7.9	656	5.3
English-Croatian	600	7.5	621	5.0

Table 3: Number of Out-Of-Vocabulary words and their percentage between translation models and WordNet senses.

	Number	Percentage
English-Italian	4239	40.35
English-Spanish	4436	42.22
English-Slovene	4523	43.04
English-Croatian	3986	37.94

Table 4: Statistics (actual number and percentage) of identified context for the evaluated WordNet Senses.

BLEU, the metric produces good correlation with human judgement at the sentence or segment level. **chrF3** is a character n-gram metric, which has shown very good correlations with human judgements on the WMT2015 shared metric task (Stanojević et al., 2015), especially when translating from English into morphologically rich(er) languages. As there are multiple translations available for each sense in the target wordnet we use all translations as multiple references for BLEU, for the other two metrics we compare only to the most frequent member of the synset.

The approximate randomization approach in MultEval (Clark et al., 2011) is used to test whether differences among system performances are statistically significant with a p-value < 0.05.

5 Evaluation

In this section we present the evaluation of the translated English WordNet words into Italian, Slovene, Spanish and Croatian. We evaluate the quality of translations of the WordNet entries based on the provided contextual information as well as the impact on the number of languages and their effect on word-sense disambiguation.

5.1 Translation Quality Evaluation Based on Contextual Information

Our main evaluation focuses on the importance of identifying relevant contexts for translation into Spanish, Italian, Slovene and Croatian. For a comparable evaluation we translated only senses within synsets, which exist in all four targeted languages. Due to the large parallel corpora used to build the translation models, only a small percentage of the used senses (10,507) could not be translated (Table 3). For this evaluation, we required contexts to be disambiguated by at least five out of nine[7] other languages. For around 40% of these senses we could identify relevant context, which was used to guide the SMT to translate the WordNet senses in the right domain (Table 4).

Table 5 illustrates the contribution of the provided contextual information, which supports the SMT system in translating the WordNet entries into the correct sense. We observed that translating a WordNet entry without any contextual information, which we consider as our baseline, provides better translations than translating them within a random context, as the most frequent translation is more likely to be correct than a random disambiguation. Once we identify one unambiguous sentence with a WordNet entry to be translated, the translation quality significantly improves in terms of the BLEU metric for all four targeted languages. Due to the large amount of parallel resources (between ≈15 and ≈50 Million sentences) we provide further a set of ten disambiguated sentences to the SMT system and select the most frequent translation of the targeted English WordNet entry. We observed, that the usage of most frequent translation helps us to improve the translation quality for 1.1 (for Slovene) and 0.7 (for Croatian) BLEU score points. In our last setting we provide the most frequent translation out of the set of t-best possible translations provided by the SMT system, however this does not seem to increase the quality of translation. Finally, in all settings we applied the target side lookup (TSL) procedure and found that it improves the quality of translation in nearly all settings.

[7]The target language is not used to help for sense disambiguation.

		English→Spanish			English→Slovene		
Context	TSL	BLEU-1	METEOR	chrF	BLEU-1	METEOR	chrF
None (baseline)	/	65.8	33.0	64.0	49.4	21.2	56.3
Random	no	54.4	27.2	61.3	36.9	15.7	52.8
Random	yes	53.0	26.6	59.3	36.4	15.9	52.4
Disambiguated	no	66.2	32.4	65.7	52.9	22.8	57.5
Disambiguated	yes	67.8	33.5	64.6	56.0	24.7	58.1
10 disambiguated	no	65.9	32.2	65.5	54.0	23.5	57.9
10 disambiguated	yes	**70.8**	**35.0**	**66.6**	**57.9**	25.4	59.0
5-best 10 disambiguated	no	66.8	32.7	65.9	55.0	23.5	57.1
5-best 10 disambiguated	yes	68.8	33.8	64.7	57.1	**28.4**	**59.6**
		English→Italian			English→Croatian		
Context	TSL	BLEU-1	METEOR	chrF	BLEU-1	METEOR	chrF
None (baseline)	/	62.5	28.4	59.6	51.1	23.8	60.7
Random	no	46.4	20.6	56.1	40.3	18.4	56.9
Random	yes	49.4	21.3	56.4	39.5	17.9	54.9
Disambiguated	no	61.5	26.6	61.7	55.1	24.7	60.0
Disambiguated	yes	66.1	27.8	61.6	57.8	26.5	60.8
10 disambiguated	no	61.0	26.2	61.3	55.8	25.6	61.1
10 disambiguated	yes	**68.0**	**28.6**	**62.7**	**61.4**	**28.3**	**63.2**
5-best 10 disambiguated	no	63.1	27.2	61.8	56.7	25.2	60.7
5-best 10 disambiguated	yes	67.1	28.2	61.6	58.7	27.1	61.5

Table 5: Evaluation of WordNet translations into Spanish, Slovene, Italian and Croatian with context-aware techniques (TSL = Target Sentence Lookup; number of languages used for disambiguation = 5)

Error Analysis In order to investigate to what extent the automatically generated translations differ from the existing entries in the target wordnets we manually inspected the WordNet translations. We compare results where contextual information was used with the approach where WordNet entries were translated in isolation, hence without context. For Slovene, the contextual information provided a correct translation of the WordNet entry *space* (outer space/location outside the Earth's atmosphere, i81724) as *vesolje*, where the context-less translation approach produced the word *prostor*, in the meaning of place, room or property. Similarly, translating *medicine* (medical science, i38643) without contextual information provided a wrong translation as *zdravilo* (medication, drug, i56119), instead of the Slovene equivalent *medicina*. For Italian, an evident mistake was observed when translating the word *tip* (gratuity, i106560), where the translation of the word in isolation wrongly produced *punta*, meaning "the top or extreme point of something" (i82274). A correct translation in Italian supported by the contextual information was provided as *mancia*. Further, *union*, in the meaning of trade union or brotherhood (i80384), *sindacato* in Italian, was wrongly translated into the most dominant meaning *unione*, with its meaning of combination or cohesion. In Croatian, the word *weed* (i105476) as "any plant that crowds out cultivated plants", was wrongly translated into *trava* (drug street name, i57595), if translated in isolation. The correct translation as *korov* was generated with the disambiguated contextual information. For Spanish, *town* (i82504) was mistranslated into *ciudad* (city or large town), whereby the preferred sense of the translation *pueblo* (small town) was generated by using the contextual information.

5.2 Impact of the Number of Languages for Sense Disambiguation

Even with a very large parallel corpus, as we increase the number of languages, in which we disambiguate the sense, we find that for many senses we cannot find a context that is disambiguated in all languages. Thus, we evaluate the impact of changing the number of languages used to disambiguate an English

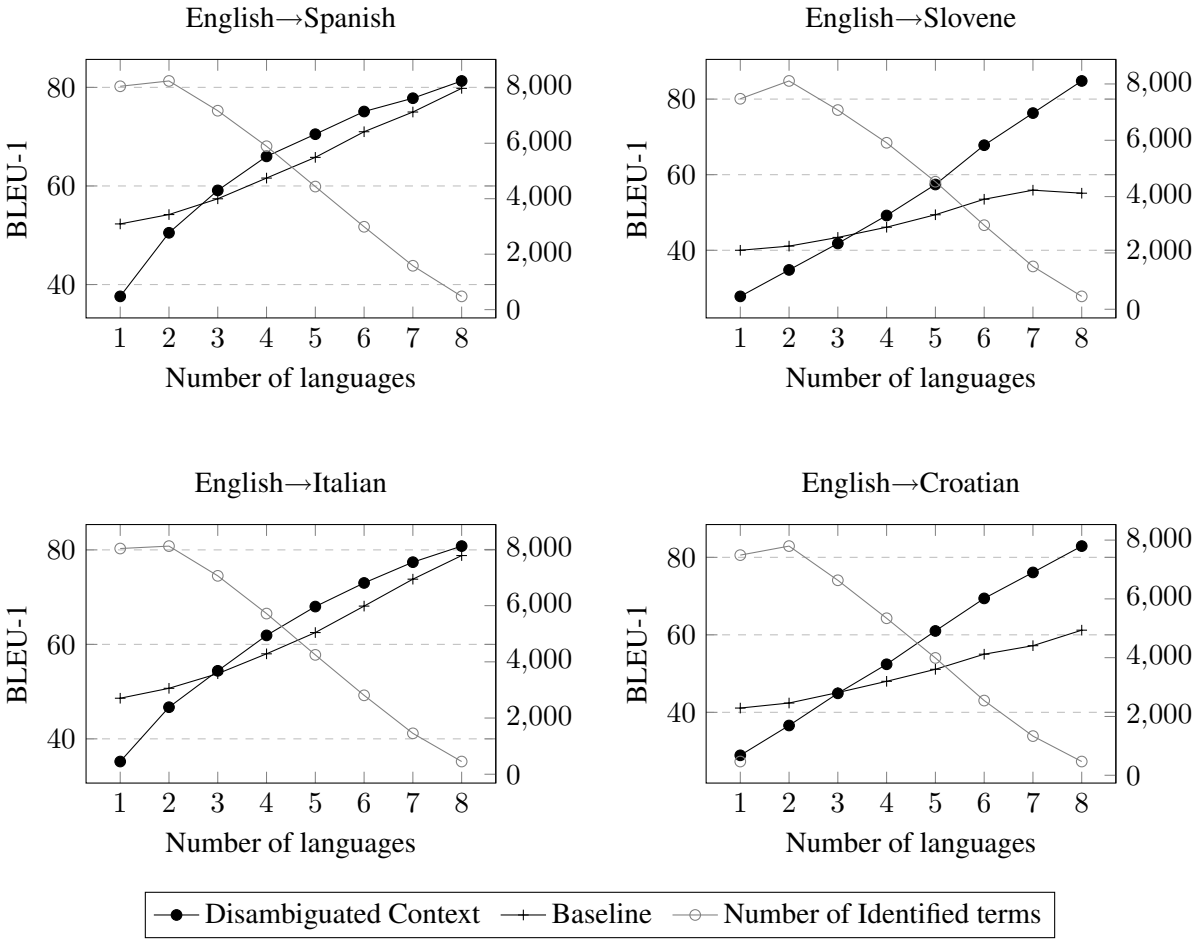


Figure 1: Impact of languages used for disambiguation and translation quality in terms of BLEU.

sentence. For this experiment we report the BLEU scores obtained by the best approach identified in Section 5.1, i.e. *10 disambiguated contexts*. For this evaluation we steadily increase the number of languages that we require a sense to be disambiguated in. We compare these results to the baseline setting, where WordNet entries are translated without any context. As the total number of senses that can be translated decreases, the BLEU score for the baseline does not stay constant and in fact increases, as the senses that our method can disambiguate in many languages are those that are more frequent and less ambiguous. Nevertheless, the disambiguation outperforms the baseline if the context is disambiguated in more than three languages (Figure 1).

For the Romance languages (Italian and Spanish), we outperform the baseline between 3 and 6 BLEU points. The improvement is more evident for the Slavic languages (Slovene and Croatian), where the differences can reach more than 10 BLEU points, if five or more languages are used. For all targeted languages, the observed improvements are statistically significant ($p<0.005$).

5.3 Impact of Language Family for Sense Disambiguation

In addition to the evaluation based on disambiguated contextual information and number of different languages, we were interested in how the similarity of languages affects the disambiguation. Firstly, we focus on the translations of English, a Germanic language, into Slovene, which is a member of the Slavic language family. We considered the cases, where the context is disambiguated in four languages, but looked at two different sets of four languages. Firstly, a group where four languages are of the same family, but different to the source and target language, using four Romance languages: French, Spanish, Romanian and Portuguese. Secondly, we evaluate the sense disambiguation approach using two Romance languages, French and Spanish, and two Slavic languages, Croatian and Polish. As illustrated

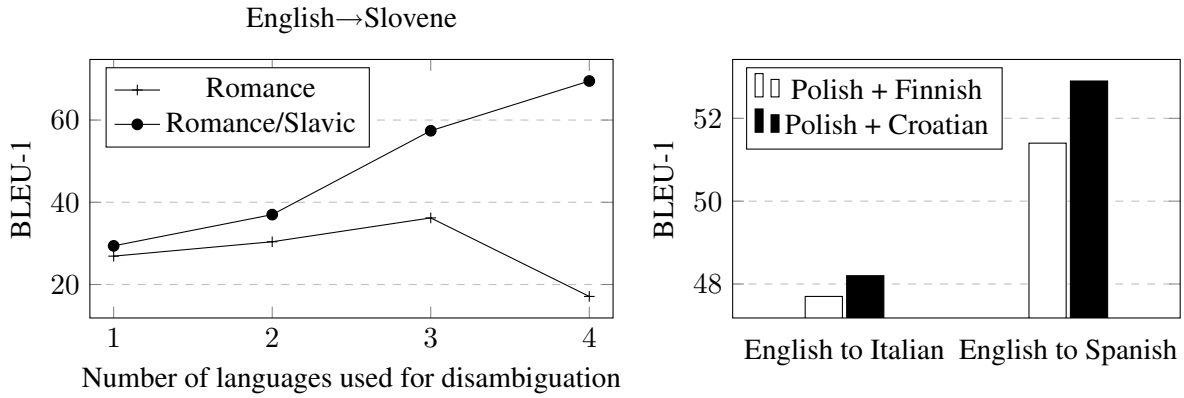


Figure 2: Evaluation on impact of closely-related languages on sense disambiguation for translation quality.

in Figure 2 (left part) the contextual disambiguation approach work significantly better if languages, closely related to the target language – in our case Slovene – are used. In our scenario including the Slavic languages to disambiguate the context yields to better translation quality compared to the usage of only Romance languages.

Secondly, we evaluated our approach if a very distant language is used in the disambiguation, namely Finnish, which is not part of the Indo-European family, the super-family of Romance, Germanic and Slavic languages. We perform disambiguation using Polish and Finnish and compare the results when Finnish is replaced with the Croatian language. The results in Figure 2 (right part) show that Finnish has less disambiguation power than Croatian even though Croatian is similar to Polish. This is because Croatian, even though it is not close to Spanish or Italian is still much closer than Finnish is.

This experiment showed that closely related languages contribute in the disambiguation approach, which yields in our scenario to better translation quality. They also show that using a diverse selection of highly distinct languages does not seem to be advantageous in disambiguating senses.

6 Conclusion and Future Work

We showed an automatic approach to increase the coverage of WordNet into different languages with high-quality translations. By identifying disambiguated context, we demonstrate statistical significant translation improvement for Spanish, Italian, Slovene and Croatian. We demonstrate the importance on closely related languages used for the sense disambiguation approach, which will help us in our ongoing work on generating translations of wordnets beyond the four targeted languages used in this work. This method allows us to release high quality extensions of Princeton WordNet, expanding the coverage for many languages, as well as creating wordnets for languages, where no wordnet has been created or the wordnet is not available to all potential users due to licensing issues.

Acknowledgements

This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289 (Insight) and the European Union supported project MixedEmotions (H2020-644632).

References

Mihael Arcan, Marco Turchi, Sara Tonelli, and Paul Buitelaar. 2014. Enhancing statistical machine translation with bilingual terminology in a CAT environment. In *Proceedings of the 11th Conference of the Association for Machine Translation in the Americas*, Vancouver, Canada.

Mihael Arcan, Marco Turchi, and Paul Buitelaar. 2015. Knowledge portability with semantic expansion of ontology labels. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, Beijing, China, July.

Mihael Arcan, Mauro Dragoni, and Paul Buitelaar. 2016. Translating ontologies in real-world settings. In *Proceedings of the 15th International Semantic Web Conference (ISWC-2016)*, Osaka, Japan.

Francis Bond and Ryan Foster. 2013. Linking and extending an open multilingual wordnet. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1352–1362, Sofia, Bulgaria, August. Association for Computational Linguistics.

Francis Bond and Kyonghee Paik. 2012. A survey of wordnets and their licenses. In *Proceedings of the 6th Global WordNet Conference (GWC 2012)*, Matsue, Japan. 64–71.

Francis Bond, Piek Vossen, John P. McCrae, and Christiane Fellbaum. 2016. CILI: the Collaborative Interlingual Index. In *Proceedings of the Global WordNet Conference 2016*, Bucharest, Romania.

Yee Seng Chan, Hwee Tou Ng, and Zhi Zhong. 2007. NUS-PT: exploiting parallel texts for word sense disambiguation in the english all-words tasks. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 253–256. Association for Computational Linguistics.

Jonathan Clark, Chris Dyer, Alon Lavie, and Noah Smith. 2011. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability . In *Proceedings of the Association for Computational Lingustics*.

Gerard de Melo and Gerhard Weikum. 2009. Towards a universal wordnet by learning from combined evidence. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009)*, pages 513–522, New York, NY, USA. ACM.

Gerard de Melo and Gerhard Weikum. 2012. Constructing and utilizing wordnets using statistical methods. *Language Resources and Evaluation*, 46(2):287–311.

Valeria de Paiva and Alexandre Rademaker. 2012. Revisiting a Brazilian wordnet. In *Proceedings of the 6th Global WordNet Conference (GWC 2012)*, Matsue, Japan.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Andreas Eisele and Yu Chen. 2010. MultiUN: A multilingual corpus from United Nation documents. In Daniel Tapias, Mike Rosner, Stelios Piperidis, Jan Odjik, Joseph Mariani, Bente Maegaard, Khalid Choukri, and Nicoletta Calzolari (Conference Chair), editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation*, pages 2868–2872. European Language Resources Association (ELRA), 5.

Christiane Fellbaum and Piek Vossen. 2012. Challenges for a multilingual wordnet. *Lang. Resour. Eval.*, 46(2):313–326, June.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Darja Fišer. 2007. Leveraging parallel corpora and existing wordnets for automatic construction of the Slovene wordnet. In *Language and Technology Conference*, pages 359–368. Springer Berlin Heidelberg.

Darja Fišer, Jernej Novak, and Tomaž Erjavec. 2012. sloWNet 3.0: development, extension and cleaning. In *Proceedings of 6th International Global Wordnet Conference (GWC 2012)*, pages 113–117, Matsue, Japan. The Global WordNet Association.

Aitor Gonzalez-Agirre, Egoitz Laparra, and German Rigau. 2012. Multilingual Central Repository version 3.0: upgrading a very large lexical knowledge base. In *Proceedings of the 6th Global WordNet Conference (GWC 2012)*, Matsue, Japan.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom.

Nancy Ide, Tomaz Erjavec, and Dan Tufiş. 2002. Sense discrimination with parallel corpora. In *Proceedings of the ACL-02 workshop on Word sense disambiguation: recent successes and future directions-Volume 8*, pages 61–66. Association for Computational Linguistics.

Dimitar Kazakov and Ahmad R. Shahid. 2009. Unsupervised construction of a multilingual wordnet from parallel corpora. In *Proceedings of the Workshop on Natural Language Processing Methods and Corpora in Translation, Lexicography, and Language Learning*, MCTLLL '09, pages 9–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, Stroudsburg, PA, USA.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*. AAMT.

Krister Lindén and Lauri Carlson. 2010. Finnwordnet — wordnet påfinska via översättning. *LexicoNordica — Nordic Journal of Lexicography*, 17:119–140. In Swedish with an English abstract.

Marek Maziarz, Maciej Piasecki, and Stanisław Szpakowicz. 2012. Approaching plWordNet 2.0. In *Proceedings of the 6th Global Wordnet Conference*, Matsue, Japan.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193:217–250, December.

Hwee Tou Ng, Bin Wang, and Yee Seng Chan. 2003. Exploiting parallel texts for word sense disambiguation: An empirical study. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 455–462. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29.

Antoni Oliver and Salvador Climent. 2012. Parallel corpora for wordnet construction: Machine translation vs. automatic sense tagging. In *Computational Linguistics and Intelligent Text Processing - 13th International Conference, CICLing 2012, New Delhi, India, March 11-17, 2012, Proceedings, Part II*, pages 110–121.

Antoni Oliver, Krešimir Šojat, and Matea Srebačić. 2015. Automatic expansion of Croatian Wordnet. In *In Proceedings of the 29th CALS international conference "Applied Linguistic Research and Methodology"*, Zadar (Croatia).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318.

Emanuele Pianta, Luisa Bentivogli, and Christian Girardi. 2002. MultiWordNet: developing an aligned multilingual database. In *Proceedings of the First International Conference on Global WordNet*, January.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, September. Association for Computational Linguistics.

Marten Postma, Emiel van Miltenburg, Roxane Segers, Anneleen Schoen, and Piek Vossen. 2016. Open Dutch WordNet. In *Proceedings of the Eight Global Wordnet Conference*, Bucharest, Romania.

Barbara Rychalska, Katarzyna Pakulska, Krystyna Chodorowska, Wojciech Walczak, and Piotr Andruszkiewicz. 2016. Samsung Poland NLP team at SemEval-2016 task 1: Necessity for methods to measure semantic similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, pages 614–620.

Benoît Sagot and Darja Fišer. 2008. Building a free French wordnet from multilingual resources. In European Language Resources Association (ELRA), editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Miloš Stanojević, Amir Kamran, Philipp Koehn, and Ondřej Bojar. 2015. Results of the WMT15 Metrics Shared Task. In *Proceedings of the 10th Workshop on Statistical Machine Translation (WMT-15)*, pages 256–273, Lisbon, Portugal, September.

Ralf Steinberger, Mohamed Ebrahim, Alexandros Poulis, Manuel Carrasco-Benitez, Patrick Schlüter, Marek Przybyszewski, and Signe Gilbro. 2014. An overview of the European Union's highly multilingual parallel corpora. *Language Resources and Evaluation*, 48(4):679–707.

Jörg Tiedemann. 2009. News from OPUS – A Collection of Multilingual Parallel Corpora with Tools and Interfaces. In *Advances in Natural Language Processing*, volume V, chapter V, pages 237–248. Borovets, Bulgaria.

Jörg Tiedemann. 2012. Character-based pivot translations for under-resourced languages and domains. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 141–151, Avignon, France, April.

Antonio Toral, Stefania Bracale, Monica Monachini, and Claudia Soria. 2010. Rejuvenating the italian wordnet: upgrading, standardising, extending. In *Proceedings of the 5th International Conference of the Global WordNet Association (GWC-2010)*, Mumbai, India.

Dan Tufiş, Dan Cristea, and Sofia Stamou. 2004. Balkanet: Aims, methods, results and perspectives. a general overview. *Romanian Journal of Information science and technology*, 7(1-2):9–43.

Dan Tufiş, Radu Ion, Luigi Bozianu, Alexandru Ceauşu, and Dan Ştefănescu. 2008. Romanian WordNet: Current state, new applications and prospects. In *Proceedings of the 4th Global WordNet Association Conference*, pages 441–452, Szeged.

Dan Tufiş, Radu Ion, and Nancy Ide. 2004. Fine-grained word sense disambiguation based on parallel corpora, word alignment, word clustering and aligned wordnets. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1312. Association for Computational Linguistics.

Piek Vossen, Francis Bond, and John P. McCrae. 2016. Toward a truly multilingual Global Wordnet Grid. In *Proceedings of the Global WordNet Conference (GWC2016)*, Bucharest, Romania.

Piek Vossen, editor. 1998. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, Norwell, MA, USA.

Piek Vossen. 2005. Building wordnets. `http://www.globalwordnet.org/gwa/BuildingWordnets.ppt`.

A Correlational Encoder Decoder Architecture for Pivot Based Sequence Generation

Amrita Saha
IBM Research India
amrsaha4@in.ibm.com

Mitesh M. Khapra
I.I.T. Madras, India
khapra.mitesh@gmail.com

Sarath Chandar
University of Montreal
anbilpas@iro.umontreal.ca

Janarthanan Rajendran
I.I.T. Madras, India
rsdjjana@gmail.com

Kyunghyun Cho
New York University
cho.k.hyun@gmail.com

Abstract

Interlingua based Machine Translation (MT) aims to encode multiple languages into a common linguistic representation and then decode sentences in multiple target languages from this representation. In this work we explore this idea in the context of neural encoder decoder architectures, albeit on a smaller scale and without MT as the end goal. Specifically, we consider the case of three languages or modalities X, Z and Y wherein we are interested in generating sequences in Y starting from information available in X. However, there is no parallel training data available between X and Y but, training data is available between X & Z and Z & Y (as is often the case in many real world applications). Z thus acts as a pivot/bridge. An obvious solution, which is perhaps less elegant but works very well in practice is to train a two stage model which first converts from X to Z and then from Z to Y. Instead we explore an interlingua inspired solution which **jointly learns** to do the following (i) encode X and Z to a common representation and (ii) decode Y from this common representation. We evaluate our model on two tasks: (i) bridge transliteration and (ii) bridge captioning. We report promising results in both these applications and believe that this is a right step towards truly interlingua inspired encoder decoder architectures.

1 Introduction

Interlingua based MT (Nirenburg, 1994; Dorr et al., 2010) relies on the principle that every language in the world can be mapped to a common linguistic representation. Further, given this representation, it should be possible to decode a target sentence in any language. This implies that given n languages we just need n decoders and n encoders to translate between these ${}^nC_2$ language pairs. Note that even though we take inspiration from interlingua based MT, the focus of this work is not on MT. We believe that this idea is not just limited to translation but could be applicable to any kind of conversion involving multiple source and target languages and/or modalities (for example, transliteration, multilingual image captioning, multilingual image Question Answering, *etc.*). Even though this idea has had limited success, it is still fascinating and considered by many as the holy grail of multilingual multimodal processing.

It is interesting to consider the implications of this idea when viewed in the statistical context. For example, current state of the art statistical systems for MT (Koehn et al., 2003; Chiang, 2005; Luong et al., 2015b), transliteration (Finch et al., 2015; Shao et al., 2015; Nicolai et al., 2015), image captioning (Vinyals et al., 2015b; Xu et al., 2015), *etc.* require parallel data between the source and target views (where a view could be a language or some other modality like image). Thus, given n views, we require ${}^nC_2$ parallel datasets to build systems to convert from any source view to any target view. Obviously, this does not scale well in practice because it is hard to find parallel data between all ${}^nC_2$ views. For example, publicly available parallel datasets for transliteration (Zhang et al., 2012) cater to < 20 languages. Similarly, publicly available image caption datasets are available only for English[1] and German[2].

[1] http://mscoco.org/dataset/#download

[2] http://www.statmt.org/wmt16/multimodal-task.html

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 109–118, Osaka, Japan, December 11-17 2016.

This problem of resource scarcity could be alleviated if we could learn only n statistical encoders and n statistical decoders wherein (i) the encoded representation is common across languages and (ii) the decoders can decode from this common representation (akin to interlingua based conversion). As a small step in this direction, we consider a scaled down version of this generic ${}^nC_2$ conversion problem. Specifically, we consider the case where we have three views X, Z, Y but parallel data is available only between XZ and ZY (instead of all ${}^3C_2$ parallel datasets). At test time, we are interested in generating natural language sequences in Y starting from information available in X. We refer to this as the bridge setup as the language Z here can be considered to be a bridge/pivot between X and Y.

An obvious solution to the above problem is to train a two-stage system which first converts from X to Z and then from Z to Y. While this solution may work very well in practice (as our experiments indeed suggest) it is perhaps less elegant and becomes tedious as the number of views increases. For example, consider the case of converting from X to Z to R to Y. Instead, we suggest a neural network based model which simultaneously learns the following (i) a common representation for X and Z and (ii) decoding Y from this common representation. In other words, instead of training two independent models using the datasets between XZ and ZY, the model jointly learns from the two datasets. The resulting common representation learned for X and Z can be viewed as a vectorial analogue of the linguistic representation sought by interlingua based approaches. Of course, by no means do we suggest that this vectorial representation is a substitute for the rich linguistic representation but its easier to learn from parallel data (as opposed to a linguistic representation which requires hand crafted resources).

Note that our work should not be confused with the recent work of (Firat et al., 2016), (Zoph and Knight, 2016) and (Elliott et al., 2015). The last two works in fact require 3-way parallel data between X, Z and Y and learn to decode sequences in Y given both X and Z. For example, at test time, (Elliott et al., 2015) generate captions in German, given both (i) the image and (ii) its corresponding English caption. This is indeed very different from the problem addressed in this paper. Similarly, even though (Firat et al., 2016) learn a single encoder per language and a single decoder per language they do not learn shared representations for multiple languages (only the attention mechanism is shared). Further, in all their experiments they require parallel data between the two languages of interest. Specifically, they do not consider the case of generating sentences in Y given a sentence in X when no parallel data is available between X and Y.

We present an empirical comparison of jointly trained models which explicitly aim for shared encoder representations with two-stage architectures. We consider two downstream applications (i) bridge transliteration and (ii) bridge caption generation. We use the standard NEWS 2012 dataset (Zhang et al., 2012) for transliteration. We consider transliteration between 12 languages pairs (XY) using English as the bridge (Z). Bridge caption generation is a new task introduced in this paper where the aim is to generate French captions for an image when no Image-French(XY) parallel data is available for training. Instead training data is available between Image-English (XZ) and English-French (ZY). In both these tasks we report promising results. In fact, in our multilingual transliteration experiments we are able to beat the strong two-stage baseline in many cases. These results show potential for further research in interlingua inspired neural network architectures. We do acknowledge that a successful interlingua based statistical solution requiring only n encoders and n decoders is a much harder task whereas our work is only a small step in that direction.

2 Related Work

Encoder decoder based architectures for sequence to sequence generation were initially proposed in (Cho et al., 2014; Sutskever et al., 2014) in the context of Machine Translation (MT) and have also been successfully used for generating captions for images (Vinyals et al., 2015b). However, such sequence to sequence models are often difficult to train as they aim to encode the entire source sequence using a fixed encoder representation. Bahdanau et al. (2014) introduced attention based models wherein a different representation is fed to the decoder at each time step by focusing the attention on different parts of the input sequence. Such attention based models have been more successful than vanilla encoder-decoder models and have been used successfully for MT (Bahdanau et al., 2014), parsing (Vinyals et

al., 2015a), speech recognition (Chorowski et al., 2015), image captioning (Xu et al., 2015) among other applications. All the above mentioned works focus only on the case when there is one source and one target. The source can be image, text, or speech signal but the target is always a text sequence.

Encoder decoder models in a multi-source, single target setting have been explored by (Elliott et al., 2015) and (Zoph and Knight, 2016). Specifically, Elliott et al. (2015) try to generate a German caption from an image and its corresponding English caption. Similarly, Zoph and Knight (2016) focus on the problem of generating English translations given the same sentence in both French and German. We would like to highlight that both these models require three-way parallel data while we are focusing on situations where such data is not available. Single source, multi-target and multi-source, single target settings have been considered in (Luong et al., 2015a). Recent work by Firat et al. (2016) explores multi-source to multi-target encoder decoder models in the context of MT. However, Firat et al. (2016) focus on multi-task learning with a shared attention mechanism and the goal is to improve the MT performance for a pair of languages for which parallel data is available. This is clearly different from the goal of this paper which is to design encoder decoder models for a pair of languages where no parallel data is available but data is available only between each of these languages and a bridge language.

Of course, in general the idea of pivot/bridge/interlingua based conversion is not new and has been used previously in several non-neural network settings. For example (Khapra et al., 2010) use a bridge language or pivot language to do machine transliteration. Similarly, (Wu and Wang, 2007; Zhu et al., 2014) do pivot based machine translation. Lastly, we would also like to mention the work in interlingua based Machine Translation (Nirenburg, 1994; Dorr et al., 2010) which is clearly the inspiration for this work even though the focus of this work is not on MT.

The main theme explored in this paper is to learn a common representation for two views with the end goal of generating a target sequence in a third view. The idea of learning common representations for multiple views has been explored well in the past (Klementiev et al., 2012; Chandar et al., 2014; Hermann and Blunsom, 2014; Chandar et al., 2016; Rajendran et al., 2015). For example, Andrew et al. (2013) propose Deep CCA for learning a common representation for two views. (Chandar et al., 2014; Chandar et al., 2016) propose correlational neural networks for common representation learning and Rajendran et al. (2015) propose bridge correlational networks for multilingual multimodal representation learning. From the point of view of representation learning, the work of Rajendran et al. (2015) is very similar to our work except that it focuses only on representation learning and does not consider the end goal of generating sequences in a target language.

3 Models

As mentioned earlier, one of the aims of this work is to compare a jointly trained model with a two stage model. We first briefly describe such a two stage encoder decoder architecture and then describe our model which is a correlation based jointly trained encoder decoder model.

3.1 A two stage encoder-decoder model

A two stage encoder-decoder is a straight-forward extension of sequence to sequence models (Cho et al., 2014; Sutskever et al., 2014) to the bridge setup. Given parallel data between XZ and ZY, a two stage model will learn a generative model for each of the pairs independently. For the purpose of this work, the source can be an image or text but the target is always a natural language text. For encoding an image, we simply take its feature representation obtained from one of the fully connected layers of a convolutional neural network and pass it through a feed-forward layer. On the other hand, for encoding the source text sequence, we use a recurrent neural network. The decoder is always a recurrent neural network which generates the text sequence, one token at a time.

Let the two training sets be $\mathcal{D}_1 = \{x_i, z_i\}_{i=1}^{N_1}$ and $\mathcal{D}_2 = \{z_i, y_i\}_{i=1}^{N_2}$ where $x_i \in X$, $y_i \in Y$ and $z_i \in Z$. Given $\mathcal{D}_1$, the first encoder learns to encode x_i and decode the corresponding z_i from this encoded representation. The second encoder is trained independently of the first encoder and uses $\mathcal{D}_2$ to encode $z_i \in Z$ and decode the corresponding $y_i \in Y$ from this encoded representation. These independent training processes are indicated by the dotted arrows in Figure 1. At test time, the two

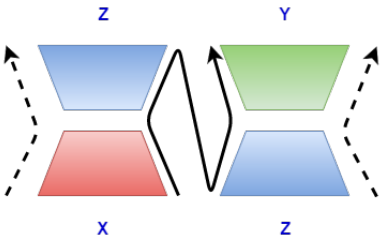


Figure 1: Two stage encoder-decoder model. Dashed lines denote how the model is used during training time and solid line denotes the test time usage. We can see that two encoder-decoders are trained independently but used jointly during testing.

stages are run sequentially. In other words, given x_j, we first encode it and decode z_j from it using the first encoder-decoder model. This decoded z_j is then fed to the second encoder-decoder model to generate y_j. This sequential test process is indicated by solid arrows in Figure 1.

While this two stage encoder-decoder model is a trivial extension of a single encoder-decoder model, it serves as a very strong baseline as we will see later in the experiments section.

3.2 A correlation based joint encoder-decoder model

While the above model works well in practice, it becomes cumbersome when more views are involved (for example, when converting from U to X to Y to Z). We desire a more elegant solution which could scale even when more views are involved (although for the purpose of this work, we restrict ourselves to 3 views only). To this end, we propose a joint model which uses the parallel data $\mathcal{D}_1$ (as defined above) to learn one encoder each for X and Z such that the representations of x_i and z_i are correlated. In addition and simultaneously the model uses $\mathcal{D}_2$ and learns to decode y_j from z_j. Note that this joint training has the advantage that the encoder for Z benefits from instances in $\mathcal{D}_1$ and $\mathcal{D}_2$.

Having given an intuitive explanation of the model, we now formally define the objective functions used during training. Given $\mathcal{D}_1 = \{x_i, z_i\}_{i=1}^{N_1}$, the model tries to maximize the correlation between the encoded representations of x_i and z_i defined as

$$\mathcal{J}_{\text{corr}}(\theta) = -\lambda \, \text{corr}(s(h_X(X)), s(h_Z(Z))) \tag{1}$$

where h_X is the representation computed by the encoder for X and h_Z is the representation computed by the encoder for Z. As mentioned earlier, these encoders could be RNN encoders (in the case of text) and simple feedforward encoders (in the case of images). $s()$ is a standardization function which adjusts the hidden representations h_X and h_Y so that they have zero-mean and unit-variance. Further, λ is a scaling hyperparameter and corr is the correlation function defined as

$$\sum_{i=1}^{N} s(h_X(x_i)) s(h_Z(z_i))^T \tag{2}$$

We would like to emphasize that $s()$ ensures that the representations already have zero mean and unit variance and hence no separate standardization is required while computing the correlation. In addition to the above loss function, given $\mathcal{D}_2 = \{z_i, y_i\}_{i=1}^{N_2}$, the model minimizes the cross entropy loss

$$\mathcal{J}_{\text{ce}}(\theta) = \frac{1}{N_2} \sum_{k=1}^{N_2} P(y_k|z_k); \qquad P(y_k|z_k) = \prod_{i=1}^{L} P(y_{k_i}|y_{k_{<i}}, z_k) \tag{3}$$

where L is the number of tokens in y_k.

The dotted lines in Figure 2 show the joint training process where the model simultaneously learns to compute correlated representations for x_i and z_i and decode y_i from z_i. The testing process is shown by the solid lines wherein the model computes a hidden representation for x_i and then decodes y_i from it directly without transitioning through z_i.

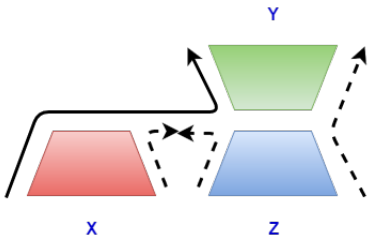


Figure 2: Correlated encoder-decoder model. Dashed lines denote how the model is used during training time and solid line denotes the test time usage. We can see that during training, both the encoders are trained to produce correlated representations and the decoder for Y is trained based on encoder Z. During test time only encoder for X and decoder for Y are used.

While training, we alternately pick mini-batches from $\mathcal{D}_1$ and $\mathcal{D}_2$ and use the corresponding objective function. Means and variances for the representations computed by the two encoders are updated at the end of every epoch based on the hidden representations of all instances in the training data. During the first epoch we assume the mean and variance to be 0 and 1. Note that λ rescales the value of the correlation loss term so that it is in the same range as the value of the cross-entropy loss term.

4 Experiment 1: Bridge Transliteration

We consider the task of transliteration between two languages X and Y when no direct data is available between them but parallel data is available between X & Z and Z & Y. In the following subsections we describe the datasets used for our task, the hyperparameters considered for our experiments and results.

4.1 Datasets

We consider transliteration between 4 languages, *viz.*, Hindi, Kannada, Tamil and Marathi resulting in ${}^4C_2 = 12$ language pairs. However, we do not use any direct parallel data between any of these languages. Instead we use the standard datasets available between English and each of these languages which were released as part of the NEWS 2012 shared task (Zhang et al., 2012). Just to be explicit, for the task of transliterating from Hindi to Kannada, we construct $\mathcal{D}_1$ from the English-Hindi dataset and $\mathcal{D}_2$ from the English-Kannada dataset. The size of the training set (in words) for the four language pairs English-Hindi, English-Kannada, English-Marathi and English-Tamil is 19918, 16556, 8500 and 16857 respectively and the validation and test set sizes (in words) are 500 and 1000 respectively. Fortunately, the English portion of the test set was common across all these four language pairs, thus allowing us to easily create test sets for all the 12 language pairs. For example, if h_i is the transliteration of the English word e_i in the English-Hindi test set and k_i is the transliteration of the same English word e_i in the English-Kannada test set then we add (h_i, k_i) as a transliteration pair in our Hindi-Kannada test set. In this way, we created test sets containing 1000 words for all the 12 language pairs.

4.2 Hyperparameters

For the two stage encoder decoder model, we considered the following hyperparameters: embedding size $\in \{1024, 2048\}$ for characters, rnn hidden unit size $\in \{1024, 2048\}$, initial learning rate $\in \{0.01, 0.001\}$ and batch size $\in \{32, 64\}$. The numbers in bracket indicate the distinct values that we considered for each hyperparameter. Note that the embedding size and rnn size are always kept equal. All these parameters were tuned independently for the two stages using their respective validation sets. For the correlated encoder decoder model, in addition to the above hyperparameters we also had $\lambda \in [0.1, 1.0]$ as a hyperparameter. Here, we tuned the hyperparameters based on the performance on the validation set available between ZY (since the correlated encoder decoder can also decode $y_i \in Y$ from $z_i \in Z$). Note that we do not use any parallel data between XY for tuning the hyperparameters because the general assumption is that no parallel data is available between XY. We used Adam (Kingma and Ba, 2014) as the optimizer for all our experiments.

Two Stage PBSMT

src \ tgt	Hi	Ka	Ta	Ma
Hi		36.3	33.2	33.6
Ka	41.3		32.1	26.2
Ta	30.5	25.8		19.2
Ma	46.9	33.7	30.9	

Two Stage Encoder Decoder

src \ tgt	Hi	Ka	Ta	Ma
Hi		42.1	**43.4**	34.8
Ka	46.2		**42.9**	**30.7**
Ta	**37.5**	**34.8**		**23.8**
Ma	45.8	34.9	31.7	

Correlational Encoder Decoder

src \ tgt	Hi	Ka	Ta	Ma
Hi		**43.1**	40.6	**40.9**
Ka	**47.5**		40.2	27.9
Ta	33.6	27.7		17.0
Ma	**59.0**	**37.1**	**34.5**	

Table 1: Transliteration Accuracy on the 12 language pairs involving (**Hi**ndi, **Ka**nnada, **Ta**mil, **Ma**rathi) for the three comparative methods (Two Stage PBSMT, Two Stage Encoder Decoder and the proposed Correlational Encoder Decoder model. An underlined number in this table signifies that for that specific language pair the corresponding system is performing better than the Two Stage PBSMT model and the best performing system for any of the language-pairs is represented in **bold** font

System	Accuracy (%) of Source-Target Pair							
	En-Hi	En-Ka	En-Ta	En-Ma	Hi-En	Ka-En	Ta-En	Ma-En
PBSMT	51.7	45.3	50.0	30.2	51.1	47.9	41.4	**35.0**
Encoder-Decoder	**61.6**	**53.7**	**57.7**	**38.0**	**57.3**	**54.5**	**46.2**	31.1

Table 2: Transliteration accuracy of the PBSMT system and the Encoder-Decoder model on the 4 Indian languages (**Hi**ndi, **Ka**nnada, **Ta**mil, **Ma**rathi) when transliterated from English and to English

4.3 Results

We compare our model with the following systems:
1. Two Stage PBSMT: Here, we train two PBSMT (Koehn et al., 2003) based transliteration systems using $\mathcal{D}_1$ and $\mathcal{D}_2$. This is an additional baseline to see how well an encoder decoder architecture compares to a conventional PBSMT based system. We used Moses (Koehn et al., 2007) for building our PBSMT systems. The decoder parameters were tuned using the validation sets. Language model was trained on the target portion of the parallel corpus.
2. Two Stage Encoder Decoder: Here, we train two encoder decoder based transliteration systems using $\mathcal{D}_1$ and $\mathcal{D}_2$ as described in Section 3.1.

Table 1 summarizes the accuracy (% of correct transliterations) of the three systems in the bridge setup. We observe that in 6 out of the 12 language pairs our correlated model does better than the 2 stage encoder decoder model. Further, it does better than the two-stage PBSMT baseline in 11 out of the 12 language pairs. This is very encouraging especially because such 2-stage approaches are considered to be very strong baselines for these tasks (Khapra et al., 2010). In general, the encoder decoder based approaches do better than PBSMT based systems. This is indeed the case even when we compare the performance of the PBSMT based system and the Encoder Decoder based system independently on the two stages (Table 2).

5 Experiment 2: Bridge Captioning

We now introduce the task of bridge caption generation. The purpose of introducing this task is two-fold. Firstly, we feel that it is important to put things in perspective and demonstrate that while interlingua inspired encoder decoder architectures are a step in the right direction, much more work is needed when dealing with different modalities in a bridge setup. Secondly, we think that this is an important task which has not received any attention in the past. We would like to formally define and report some initial baselines to motivate further research in this area. The formal task definition is as follows: Generate captions for images in language L_1 (say, French) when no parallel data is available between images and L_1 but parallel data is available between Image-L_2 ($\mathcal{D}_1$) and between L_1-L_2 ($\mathcal{D}_2$) where L_2 is another language (say, English). In the following subsection we describe the datasets used for this task, the hyperparameters considered for our experiments and the results.

5.1 Datasets

Even though we do not have direct training data between Image-French, we need some test data to evaluate our model. For this, we use the Image-French test set recently released by (Rajendran et al.,

Systems	BLEU-4	BLEU-3	BLEU-2	BLEU-1	ROUGE-L	CIDEr
Pseudo Im-Fr	15.5	24.2	37.4	56.5	38.3	41.2
Two Stage	**16.6**	**25.7**	**39.0**	**58.3**	**39.5**	**49.1**
Correlational Encoder Decoder	12.6	19.3	31.1	50.5	34.3	29.8

Table 3: Image Captioning performance in generating French caption for a given image for the three methods: Pseudo Im-Fr, Two Stage and our Correlational Encoder Decoder based model.

2015). To create this data, they first merged the 80K images from the standard train split and 40K images from the standard valid split of MSCOCO data. They then randomly split the merged 120K images into train(118K), validation (1K) and test set (1K). They then collect French translations for all the 5 captions for each image in the test set using crowdsourcing. CrowdFlower (`https://make.crowdflower.com`) was used as the crowdsourcing platform and they solicited one French and one German translation for each of the 5000 captions using native speakers. Note that (Rajendran et al., 2015) report results for cross modal search and do not address the problem of crosslingual image captioning.

In our model, for D_1 we use the same train(118K), validation (1K) and test sets (1K) as defined in (Rajendran et al., 2015) and explained above. Choosing $\mathcal{D}_2$ was a bit more tricky. Initially we considered the corpus released as part of WMT'12 (Callison-Burch et al., 2012) which contains roughly 44M English-French parallel sentences from various sources including News, parliamentary proceedings, etc. However, our initial small scale experiments showed that this does not work well because there is a clear mismatch between the vocabulary of this corpus and the vocabulary that we need for generating captions. Also the vocabulary is much larger (at least an order higher than what we need for image captioning) and it thus hampers training. Further, the average length and structure of these sentences is also very different from captions. Domain shift in MT is itself a challenging problem (not to mention the added complexity in a multimodal bridge setup). It was unrealistic to expect our model to work in the presence of these orthogonal complexities.

To isolate these issues and evaluate our model in a controlled environment, we needed a parallel corpus which had very similar characteristics to that observed in captions. Since we did not have such a corpus at our disposal we decided to follow (Rajendran et al., 2015) and use a pseudo parallel corpus between English-French. Specifically, we take the English captions from the MSCOCO data and translate them to French using the publicly available translation system provided by IBM (`http://www.ibm.com/smarterplanet/us/en/ibmwatson`). Note that our model still does not see direct parallel data between Image and French during training. We acknowledge that this is not the ideal thing to do but it is good enough to do a proof-of-concept evaluation of our model and understand its potential. We, of course, account for the liberty taken here by comparing with equally strong baselines as discussed later in the results section.

5.2 Hyperparameters

Our model has the following hyperparameters: embedding size, batch size, hidden representation size, λ and learning rate. Based on experiments involving direct Image-to-English caption generation we observed that the following parameters work well : embedding size = 512, batch size = 80, rnn hidden unit size = 512, and learning rate = 4e-4 with Adam (Kingma and Ba, 2014) as the optimizer. We just retained these hyperparameters and did not tune them again for the bridge setup. We tuned the value of λ by evaluating the correlation loss on the Image-English validation set. Again, we do not use any Image-French data for tuning any hyperparameters.

5.3 Results

We now present the results of our experiments where we compare with the following strong baselines.
1. Two Stage : Here we use a Show & Tell model (Vinyals et al., 2015b) trained using D_1 to generate an English caption for the image. We then translate this caption into French using IBM's translation system as described above.
2. Pseudo Im-Fr : Here we train an Image-to-French Show & Tell model (Vinyals et al., 2015b) by pairing the images in the MSCOCO dataset with their pseudo French captions generated by translating

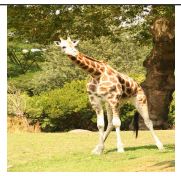
MSCOCO Images						
Correlational Encoder Decoder	Un homme est surfer sur une vague dans l'océan	Un skateur est en train de décoller sur un skateboard	Une plaque avec un sandwich et un verre de bière	Une girafe est debout dans la poussière près d'une arborescence	Un bus de transport en commun dans une rue de la ville	Un salon avec un canapé , une table et un téléviseur
Two-Stage	Un homme circonscription une vague sur une planche de surf	Un homme circonscription un skateboard sur une rampe en bois	Une plaque de nourriture sur une table avec un verre de vin	Une girafe debout dans un champ avec des arbres en arrièreplan	Un bus double sandwich au volant dune rue	Un salon avec un canapé fauteuil et une télévision
Pseudo Im-Fr	Un internaute dans une combinaison isothermique est circonscription une vague	Un jeune garçon circonscription un skateboard dans un parc	Une plaque de nourriture sur une table en bois	Une girafe debout à côté d'un autre girafe dans une zone	Un bus ville faire baisser une rue de la ville	Un salon avec un canapé , une table et un canapé

Table 4: Example captions generated by the three methods on a sample set of MSCOCO test images

the English captions into French (using IBM's translation system).

We observe that our model is unable to beat the two strong baselines described above but still comes close to their performance. We believe this reinforces our belief in this line of research and hopefully more powerful models (perhaps attention based) could eventually surpass these two baselines.

As a qualitative evaluation of our model, Table 4 shows the captions generated by our model. It is exciting that even in a complex multimodal bridge setup the model is able to capture correlations between Images and English sentences and further decode relevant French captions from a given image.

The code and datasets used for Experiment 1 and 2 would be made available on request.

6 Conclusion

In this paper, we considered the problem of pivot based sequence generation. Specifically, we are interested in generating sequences in a target language starting from information in a source view. However, no direct training data is available between the source and target views but data is available between each of these views and a pivot view. To this end, we take inspiration from interlingua based MT and propose a neural network based model which explicitly maximizes the correlation between the source and pivot view and simultaneously learns to decode target sequences from this correlated representation. We evaluate our model on the task of bridge transliteration and show that it outperforms a strong two-stage baseline for many language pairs. Finally, we introduce a novel bridge caption generation task and report promising initial results. We hope this new task will fuel further research in this area.

As future work, we would like to go beyond simple encoder decoder based correlational models. For example, we would like to apply the idea of correlation to attention based encoder decoder models. The ideas expressed here can also be applied to other tasks such as bridge translation, bridge Image QA, etc. However, for these tasks, additional issues such as larger vocabulary sizes, complex sentence structures, non-monotonic alignments between source and target language pairs need to be addressed. The model proposed here is just a beginning and much more work is needed to cater to these complex tasks.

References

Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. *ICML*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Seventh Workshop on Statistical Machine Translation*, WMT, pages 10–51, Montréal, Canada.

Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M Khapra, Balaraman Ravindran, Vikas Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Proceedings of NIPS*.

Sarath Chandar, Mitesh M. Khapra, Hugo Larochelle, and Balaraman Ravindran. 2016. Correlational neural networks. *Neural Computation*, 28(2):257 – 285.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 577–585.

Bonnie J. Dorr, Rebecca J. Passonneau, David Farwell, Rebecca Green, Nizar Habash, Stephen Helmreich, Eduard H. Hovy, Lori S. Levin, Keith J. Miller, Teruko Mitamura, Owen Rambow, and Advaith Siddharthan. 2010. Interlingual annotation of parallel text corpora: a new framework for annotation and evaluation. *Natural Language Engineering*, 16(3):197–243.

Desmond Elliott, Stella Frank, and Eva Hasler. 2015. Multi-language image description with neural sequence models. *CoRR*, abs/1510.04709.

Andrew Finch, Lemao Liu, Xiaolin Wang, and Eiichiro Sumita. 2015. Neural network transduction models in transliteration generation. *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 61.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 866–875.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 58–68.

Mitesh M. Khapra, A. Kumaran, and Pushpak Bhattacharyya. 2010. Everybody loves a rich cousin: An empirical study of transliteration through bridge languages. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*, pages 420–428.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing Crosslingual Distributed Representations of Words. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *HLT-NAACL*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015a. Multi-task sequence to sequence learning. *CoRR*, abs/1511.06114.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015b. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.

Garrett Nicolai, Bradley Hauer, Mohammad Salameh, Adam St Arnaud, Ying Xu, Lei Yao, and Grzegorz Kondrak. 2015. Multiple system combination for transliteration. *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 72.

Sergei Nirenburg. 1994. Pangloss: A machine translation project. In *Human Language Technology, Proceedings of a Workshop held at Plainsboro, New Jerey, USA, March 8-11, 1994*.

Janarthanan Rajendran, Mitesh M Khapra, Sarath Chandar, and Balaraman Ravindran. 2015. Bridge correlational neural networks for multilingual multimodal representation learning. *arXiv preprint arXiv:1510.03519*.

Yan Shao, Jörg Tiedemann, and Joakim Nivre. 2015. Boosting english-chinese machine transliteration via high quality alignment and multilingual resources. *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 56.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015a. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2773–2781.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015b. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3156–3164.

Hua Wu and Haifeng Wang. 2007. Pivot language approach for phrase-based statistical machine translation. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2048–2057.

Min Zhang, Haizhou Li, A. Kumaran, and Ming Liu. 2012. Report of NEWS 2012 machine transliteration shared task. In *Proceedings of the 4th Named Entity Workshop*, NEWS '12, pages 10–20, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiaoning Zhu, Zhongjun He, Hua Wu, Conghui Zhu, Haifeng Wang, and Tiejun Zhao. 2014. Improving pivot-based statistical machine translation by pivoting the co-occurrence count of phrase pairs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1665–1675.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *CoRR*, abs/1601.00710.

Zero-resource Dependency Parsing: Boosting Delexicalized Cross-lingual Transfer with Linguistic Knowledge

Lauriane Aufrant[1,2] and **Guillaume Wisniewski**[1] and **François Yvon**[1]
[1]LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, 91 405 Orsay, France
[2]DGA, 60 boulevard du Général Martial Valin, 75 509 Paris, France
{lauriane.aufrant,guillaume.wisniewski,francois.yvon}@limsi.fr

Abstract

This paper studies cross-lingual transfer for dependency parsing, focusing on very low-resource settings where delexicalized transfer is the only fully automatic option. We show how to boost parsing performance by rewriting the source sentences so as to better match the linguistic regularities of the target language. We contrast a data-driven approach with an approach relying on linguistically motivated rules automatically extracted from the World Atlas of Language Structures. Our findings are backed up by experiments involving 40 languages. They show that both approaches greatly outperform the baseline, the knowledge-driven method yielding the best accuracies, with average improvements of +2.9 UAS, and up to +90 UAS (absolute) on some frequent PoS configurations.

1 Introduction

The need to automatically process an increasing number of languages has made obvious the extreme dependency of standard development pipelines on in-domain, annotated resources that are required to train efficient statistical models. However, for most languages, annotated corpora only exist for a restricted number of domains, when they exist at all. In response to such low-resource scenario, four main strategies have been considered. The first is to hire experts and handcraft these resources, possibly with the help of active learning techniques: Garrette and Baldridge (2013) show that this strategy can be effective and probably cheaper than expected. An alternative is to use models learned on some resource-rich *source language(s)* to process a low-resource *target language*; note that this is only possible once the source and target data have been mapped into a shared representation space (Zeman and Resnik, 2008). When source-target parallel corpora are available, a third approach projects annotations across languages via alignment links (Yarowsky and Ngai, 2001; Hwa et al., 2005; Lacroix et al., 2016). A variant using *artificial* parallel corpora, obtained via Machine Translation, is suggested and discussed by Tiedemann et al. (2014).

In this work, we focus on the problem of learning dependency parsers for an under-resourced language and consider the *delexicalized transfer* approach of Zeman and Resnik (2008), in which the shared source-target representation is obtained by replacing all tokens by their PoS (assuming a common tagset). Thanks to this language-independent representation, a model trained with annotated sentences in a source language can be readily applied to parse sentences in any other language. Delexicalized techniques are especially useful in very low-resource settings, in which existing parallel corpora are likely to be too small or even non-existing. The development of cross-linguistically homogeneous and consistent schemes for PoS labels (Petrov et al., 2012) and, more recently, for dependency trees (McDonald et al., 2013) has been of great help to improve the applicability and effectiveness of delexicalized transfer methods. We contend, however, that having a universal PoS inventory is only a first step towards making the source and target languages more alike. In particular, these shared representations may hide fundamental differences in word order between source and target languages. As explained in § 2, these

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 119–130, Osaka, Japan, December 11-17 2016.

divergences introduce systematic biases in parsers: since many features rely on word linear sequence, their distribution across languages varies in great proportions, preventing useful generalizations to be effectively transferred cross-linguistically.

In the remaining sections, we study ways to improve the performance of delexicalized techniques by making the source word sequence more similar to target sentences, prior to transferring information. Two extensions are contrasted: a data-driven approach and a knowledge-driven approach (§ 3). The former uses PoS-based statistical language models estimated on target data while the latter relies only on the World Atlas of Language Structures (WALS) (Dryer and Haspelmath, 2013), which contains a series of linguistic typological features documenting 2,679 languages. Experiments on 40 languages exhibiting very different characteristics and covering several language families show that both methods outperform standard delexicalized transfer by a wide margin (§ 4), with the knowledge-based approach having the additional benefit to even dispense with the need of unlabeled target data and consequently to be readily usable for more than thousand languages. Incidentally, our experiments thoroughly re-evaluates previous proposals for improving baseline delexicalized transfer.

2 Motivations

2.1 Principles of Transition-Based Dependency Parsing

Transition-based dependency parsers (Nivre, 2008) are among the most popular methods for computing a syntactic structure. For clarity, we illustrate our work on greedy ARCEAGER parsers which have achieved state-of-the-art performance for many languages. However, our motivations hold regardless of the chosen parsing system, and exploratory experiments with our methods have shown similar improvements with other parsers (including graph-based parsers).

In an ARCEAGER parser, the parse tree is built incrementally while traversing the sentence from left to right, by executing elementary actions that either move words in a buffer and a stack (via SHIFT and REDUCE actions) or create dependency relationships between the word on top of the stack and the leftmost word in the buffer (using the LEFT or RIGHT actions depending whether the head is in the buffer or on the stack).

The actions performed during parsing are predicted by a feature-based classifier, a common choice being the averaged perceptron of Collins and Roark (2004). It is custom to base the classifier decisions on a limited window centered on the two tokens which could be moved or attached; the following features[1] are typically extracted from these neighborhoods and combined together to yield feature tuples: top of the stack (generally denoted s_0) and deeper stack elements (s_1, s_2) to its left, head of the buffer (n_0) and additional tokens (n_1, n_2) on its right.

Transition-based parsers heavily rely on word order: for instance, as shown in Figure 1, in an ARCEAGER system, the dependency between two words will be predicted by two different actions depending whether the head occurs before or after its dependent. More importantly, most features used in a dependency parser (no matter the transition system) are sensitive to the word order, as they encode the position of the word in the stack or in the buffer which, in turn, depends on the position of the word in the sentence.[2]

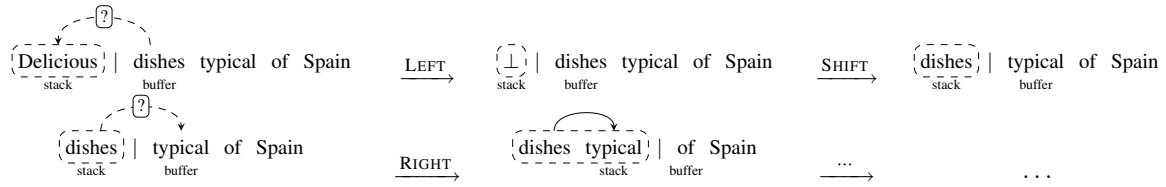


Figure 1: An order-sensitive sequence of transitions computing a dependency tree.

[1] For lexicalized parsers: the word forms and the PoS, for delexicalized: only the PoS.

[2] Graph-based parsing with standard feature templates is slightly less order-dependent, since the classification task and the features of the candidate dependent and head are already abstracted from the linear sequence. However, many features, related for instance to words located *between* these two tokens, remain sensitive to word order and our statement still holds.

2.2 A Waste of Cross-Lingual Knowledge

Delexicalized transfer has proven to be an effective method to transfer parsers between languages (Zeman and Resnik, 2008; McDonald et al., 2013). However, while delexicalized transfer extracts useful language-independent knowledge from training instances in the source language, we claim that this knowledge is often not encoded in the right form to be effectively used to process target sentences, due to divergences in word ordering.[3]

We illustrate this on delexicalized transfer from English to French. Let us assume that we have a delexicalized English parser that is able to perfectly predict the dependency structure of the noun phrase *the following question* and we use it to annotate the corresponding French phrase *la question suivante* (literally, *the question following*[4]). Thanks to recent efforts in defining universal annotation schemes for syntactic information, notably the Universal Dependencies (UD) project (Nivre et al., 2016), these phrases can be represented in a unified manner by mapping word forms into the corresponding PoS, yielding respectively DET ADJ NOUN and DET NOUN ADJ. As the English parser has learned that 'DETs depend on NOUNs' and that 'ADJs depend on NOUNs', the appropriate parse for the French phrase should be obvious, as these rules apply cross-linguistically. PoS sequences thus seem to provide an appropriate level of abstraction for cross-lingual transfer.

However, contrary to what this intuition suggests, the transfer of the ADJ-NOUN dependency often fails in practice. This is because the features underlying the high-level rules stated above are in fact order-dependent. Indeed, when parsing the French phrase, the parser configuration will be mainly described by the feature pair 's_0=NOUN $\wedge$ n_0=ADJ' (as *question* appears before *suivante*, it will be put on the stack first) while for the English phrase the relevant parser configuration would look like 's_0=ADJ $\wedge$ n_0=NOUN'. For lack of connecting these two situations, the parser has no way to predict the correct attachment in French using only English training instances.

Experimentally,[5] and denoting $\mathrm{UAS}\left[\begin{smallmatrix}C_2\\C_1\end{smallmatrix}\right]$ the percentage of C_1 tokens depending on a C_2 token that are correctly attached by the parser, while the English delexicalized model has a $\mathrm{UAS}\left[\begin{smallmatrix}\text{NOUN}\\\text{ADJ}\end{smallmatrix}\right]$ of 91.1% on English, it drops down to 50.8% for French. This decrease results directly from the word order difference between French and English, as English adjectives are almost always preposed[6] while their position in French is less deterministic: in the French UD, 28% of the adjectives occur before their head noun and 72% after it. As a result, the $\mathrm{UAS}\left[\begin{smallmatrix}\text{NOUN}\\\text{ADJ}\end{smallmatrix}\right]$ score on French actually decomposes as 96.8% for $\mathrm{UAS}\left[\begin{smallmatrix}\text{NOUN}\\\text{preposed ADJ}\end{smallmatrix}\right]$ and 34.5% for $\mathrm{UAS}\left[\begin{smallmatrix}\text{NOUN}\\\text{postposed ADJ}\end{smallmatrix}\right]$.[7] These observations highlight the impact of word order on delexicalized transfer: attachment patterns are not robust to variations in word ordering. Note that transfer from French to English is much more successful, with a $\mathrm{UAS}\left[\begin{smallmatrix}\text{NOUN}\\\text{ADJ}\end{smallmatrix}\right]$ of 80.5%. This is because the source language (here French) contains a sufficiently large number of preposed adjectives, which makes it possible to learn the patterns that are useful for English.

The discrepancies in word order can have an even more dramatic effect when transferring parsers between languages in which adjectives have a fixed position. This, for instance, happens when the source is Bulgarian (almost only preposed adjectives) and the target is Hebrew (only postposed): the resulting $\mathrm{UAS}\left[\begin{smallmatrix}\text{NOUN}\\\text{ADJ}\end{smallmatrix}\right]$ is as low as 28.7% (compared to an overall UAS of 60.1%). In the reverse direction, it drops down to 2.8% ($\mathrm{UAS}\left[\begin{smallmatrix}\text{NOUN}\\\text{preposed ADJ}\end{smallmatrix}\right]$ of 0.7%, $\mathrm{UAS}\left[\begin{smallmatrix}\text{NOUN}\\\text{postposed ADJ}\end{smallmatrix}\right]$ of 54.5%, with an overall UAS of 50.6%).[8]

The impact of differences in word order on cross-lingual transfer is not limited to the attachment of adjectives. Consider, for instance, the English phrase *the neighbor's car* (DET NOUN PART NOUN) and

[3]The issues described in this section are at least partially solved by transfer with annotation projection but these techniques require parallel data that are not always available.

[4]Keeping the original order (*la suivante question*) would be wrong in French.

[5]Our experimental data and protocols are presented in Section 4.

[6]In the English UD corpus, 93% of the adjectives come before the noun they depend on.

[7]This observation is consistent with the English monolingual scores (93.2% for the $\mathrm{UAS}\left[\begin{smallmatrix}\text{NOUN}\\\text{preposed ADJ}\end{smallmatrix}\right]$ majority case, and 55.0% for the much rarer $\mathrm{UAS}\left[\begin{smallmatrix}\text{NOUN}\\\text{postposed ADJ}\end{smallmatrix}\right]$ case).

[8]Source data quality cannot be the only cause of such poor results: when delexicalized models apply monolingually, $\mathrm{UAS}\left[\begin{smallmatrix}\text{NOUN}\\\text{ADJ}\end{smallmatrix}\right]$ is 97.4% in Bulgarian and 88.4% in Hebrew.

its French translation *la voiture du voisin* (DET NOUN ADP NOUN). After attaching function words, all that remains for the parser to process is the bigram NOUN NOUN: while the English parser has been trained to predict a left dependency (*car* being the head of *neighbor*), for French it must predict a right dependency (*voiture* being the head of *voisin*). Here the discrepancy of genitives' position across languages does not involve unseen features, but still leads the model to predict a wrong dependency with high confidence.

Our work aims at addressing such scenarios in which knowledge transfer is impeded by the word order of the source language. While current state-of-the-art models learn that 'an ADJ followed by a NOUN depends on that NOUN' and 'the first NOUN depends on the second NOUN', we would like them to transfer more abstract patterns such as 'ADJs depend on NOUNs' and 'genitives depend on NOUNs', leaving it up to the target side to decide which of both NOUNs plays the role of genitive.

3 Boosting Delexicalized Transfer

3.1 Reshaping Training Instances

In this work, we propose to preprocess the source data before they are used to train a delexicalized parser, that will then be directly applied on target sentences. This preprocessing aims at making the source word sequences more similar to target sentences, with the goal to make the cross-lingual knowledge more accessible after transfer. The available information is the same before and after preprocessing (no dependency is ever added), but is presented at training time in a form that should make it more useful at test time.

In the following, we introduce two ways of generating such transformations, by removing or permuting tokens. The first approach uses a language model estimated on target PoS sequences to find the most similar word order between the source and target languages in a lattice containing local reorderings of the source sentence. The second strategy relies on a data bank of linguistic typological features, the WALS (Dryer and Haspelmath, 2013), to generate a series of heuristic transformation operations.

The problem of finding good reorderings of a source sentence is closely related to the problem of word (p)reordering in Statistical Machine Translation (SMT) (Bisazza and Federico, 2016). However, where preordering aims to find an optimal (for SMT) permutation of source words *for each source sentence*, our objective is less ambitious, as we only intend to 'fix' a sufficiently large number of divergent patterns between the source and target languages, so as to increase the effectiveness of transfer *at the model level*.

3.2 Optimally Reordering the Training Corpus with a Language Model

Our first resource-light approach consists of two steps. We first generate a small subset of possible token permutations, compactly encoded in a finite-state graph. In our experiments, we consider all the permutations licensed by the MJ-2 reordering scheme (Kumar and Byrne, 2005), which generates all possible local permutations within a window of three words. Machine Translation experiments have shown that the MJ-2 constraints capture lots of plausible reorderings (Dreyer et al., 2007). In the context of cross-lingual transfer, its local nature allows to correct several important divergences in word order (e.g. the adjective-noun divergence described in § 2.2), while keeping the size of the reordering lattice polynomial with respect to the sentence length (Lopez, 2009).

The permutation lattice is then searched for a reordering that (a) corresponds to a high probability target PoS sequence and (b) preserves the projectivity constraint. In practice, we first generate the lattice of MJ-2 reorderings, score it with a language model estimated on the target PoS sequences, and extract the 1,000-best sequences. After filtering non-projective trees, we retain the one-best sequence (if one projective tree exists), or the original sequence otherwise. We expect the word order of this transformed source to be very close to the word order of a typical target sentence. We then transform the gold dependency tree according to this permutation and use it to train a target-adapted model.

This approach can be viewed as an extension of the data selection technique of Søgaard (2011) in which the delexicalized model is trained only on the source examples that are the most *relevant* for the target at hand. The similarity between the source and target languages is based on the similarity between

their PoS sequences: experimentally, the author retains the 90% sentences with lowest perplexity according to a target PoS language model (PoSLM). We add here an extra degree of freedom by allowing changes in the word order of the source PoS sequence, rather than simply discarding sentences.

3.3 Adapting the Training Corpus with Rewrite Rules

Our second proposal takes advantage of the linguistic knowledge that is now available for many languages. We use here the WALS, which contains a series of linguistic features documenting 2,679 languages. Some of these features are of prime interest for our study, and express general properties related to word order. In this work we focus on the following seven features that describe whether some PoS classes exist in a language and their relative position (preposed or postposed to the noun, or no dominant order): 37A (definite articles), 38A (indefinite articles), 85A (order of adposition and noun), 86A (order of genitive and noun), 87A (order of adjective and noun), 88A (order of demonstrative and noun) and 89A (order of numeral and noun).[9]

We first extract the relevant features for each language considered in our study, quantify their value and automatically transform them to relate to the raw PoS sequences found in our corpora. We extrapolate the order of DET and NOUN from feature 88A and identify the genitives mentioned by feature 86A as NOUNs or PROPNs depending on a NOUN. With an otherwise straightforward mapping, this results in the following set of properties: no definite DET, no indefinite DET (including the affix cases), and a precedence rate (denoted PR) of 0% (postposed), 50% (no dominant order) or 100% (preposed) for ADPs (resp. genitives, ADJs, DETs, NUMs) depending on a NOUN.[10]

The 'No dominant order' feature value of WALS provides very useful quantitative information: contrary to the PoSLM-based approach, which puts hard constraints on each phenomenon by choosing a reordering even when several choices would be almost equally likely, WALS features indicate when and how to balance our transformed treebanks.

By comparing two languages based on their feature values, it is then possible to define actionable transformation rules that remove or permute tokens and their associated subtrees. Table 1 lists the transformation rules derived from each pair of features. We preferred smooth transformations (with mean PR objectives and error margins) to prevent a full transformation of the corpus and a risk of information losses if the child position is less deterministic than expected. For instance, in the case of transfer from English (ADP-NOUN) to Japanese (NOUN-ADP) and according to the fourth transformation rule, we target a precedence rate of ADPs to NOUNs between 45% and 55%. This means that during source treebank traversal, while the precedence rate in previous sentences is above 55% (resp. below 45%), any encountered ADP-NOUN (resp. NOUN-ADP) bigram holding a dependency is switched, along with their dependents to preserve projectivity. According to first rule, for transfer to Czech (no definite article) from any source, all definite articles are systematically removed from source data.

Source feature	Target feature	Transformation rule
any	no DEF-DET	remove all definite DETs
any	no IND-DET	remove all indefinite DETs
$\mathrm{PR} = 0\%$	$\mathrm{PR} \geq 50\%$	switch subtrees to reach $\mathrm{PR} = 50\%$ (with 5% error margin)
$\mathrm{PR} = 100\%$	$\mathrm{PR} \leq 50\%$	switch subtrees to reach $\mathrm{PR} = 50\%$ (with 5% error margin)
$\mathrm{PR} = 50\%$	$\mathrm{PR} = 100\%$	switch subtrees to reach $\mathrm{PR} = 75\%$ (with 5% error margin)
$\mathrm{PR} = 50\%$	$\mathrm{PR} = 0\%$	switch subtrees to reach $\mathrm{PR} = 25\%$ (with 5% error margin)

Table 1: Transformation rules extracted from the comparison of the feature values of a language pair. All other feature pairs result in a no-op.

For each sentence, we apply each rule on the whole sequence (and then iterate 3 times to capture recur-

[9] We do not consider here features (81A, 82A, etc.) describing the relative position of a head VERB and its dependents. Their use would require us to condition our preprocessing patterns on labeled dependency relationship in the source, a task we leave for future work.

[10] For ADPs, and for resilience to annotation inconsistencies, we also include ADPs that are heads of NOUNs.

sive phenomena). Such heuristic rule application strategy is undoubtedly sensitive to the rule ordering, but we have not yet investigated this aspect and simply apply rules according to the lexicographic order of the child tag, breaking ties using word position.

In comparison to the PoSLM-based approach, the WALS-based approach suffers from a lack of exhaustivity regarding word order; by design, less phenomena will be captured. However, since the objective is not the best possible reordering but only more compatible PoS sequences, exhaustivity is probably not a big issue. Besides, working with a discrete and reduced set of transformation operations gives us a better control on the rewriting of dependencies. It also allows us to use extra operations such as word deletion, a transformation that may be difficult to control in the approach described in § 3.2.

Altogether, this linguistically rich method presents a notable upside: since all the required information is available in WALS, it is readily usable for more than thousand languages. Provided that PoS tags can be generated for the target data to parse, no extra resource is required, while estimating a PoSLM requires a sufficiently large corpus of reliably PoS-tagged target data.

4 Experiments

4.1 Experimental Setup

We evaluate our proposals on the Universal Dependencies corpus[11] (Nivre et al., 2016) and compare them with three baselines: (a) standard delexicalized transfer, (b) the data point selection method of Søgaard (2011) and (c) the weighted multi-source combination of Rosa and Zabokrtsky (2015), that weights and combines the hypotheses of several delexicalized source models using KL_{cpos^3} (Kullback-Leibler divergence of coarse PoS trigram distributions) as a syntactic similarity metric between languages. We also include the UAS of KL_{cpos^3} multi-source combination built on top of our knowledge-based model.

In all our experiments, we consider 3-gram PoS language models estimated on the training sets of UD. The KL_{cpos^3} metric is estimated on the same PoS sequences. From WALS, we extract and use the 37A, 38A and 85A to 89A features. For some languages, this information was incomplete. We completed missing features with a majority vote of the languages of same genus if available in the database; otherwise (i.e. for ancient languages, all absent from WALS) we assumed that there were separate article tokens and that there was no dominant order for word order features.

For each component of the algorithms, we use the universal PoS tagset and gold PoS tags. While this scenario is probably unrealistic, it allows us to get a clearer picture of the net effect of a better syntactic knowledge transfer, since possible sources of discrepancies between languages (e.g. more or less noisy tag labels) have been removed. The parser is a greedy ARCEAGER transition-based parser trained with a dynamic oracle (Goldberg and Nivre, 2012), an averaged perceptron classifier (Collins and Roark, 2004) and Zhang and Nivre (2011)'s feature templates (assuming fully delexicalized representations and unlabeled arcs). We use the PanParser implementation (Aufrant and Wisniewski, 2016) and all the code used in this work is available at `https://perso.limsi.fr/aufrant/`.

4.2 Results

Table 2 presents UAS results for the various transfer methods considered. As these experiments amount to 6,320 evaluated parsers, we provide the results in a compacted form as follows. For each target language, for mono-source transfer, we report the scores of the worst, median, best sources and the average score (or average gain) over all sources.

Overall, both preprocessing techniques outperform the direct transfer method of Zeman and Resnik (2008) as well as the selection strategy of Søgaard (2011). The WALS-based rewriting approach yields higher improvements (+2.9 UAS on average) than the PoSLM-based reordering strategy (+2.3 UAS on average). Thanks to the variety and the large number of sources, the multi-source methods have here much higher accuracies, often better than the best source; even in this setting, using WALS provides us with a slight improvement over the baseline multi-source parser.

[11] We consider the version 1.3 of the UD treebank. In order to present only fair sources, for languages where several treebanks are available, we retain only the *main* treebank. This is the case for the following languages: cs, en, es, fi, grc, la, nl, pt, ru, sl and sv.

Target	[Mono-source																]	[Multi-source]	
	Delexicalized				PoSLM selection				PoSLM reordering				WALS rewrite rules					Delex.	WALS
	min	med	max	avg	min	med	max	Δavg	min	med	max	Δavg	min	med	max	Δavg			
ar	5.1	43.2	56.9	36.1	4.8	43.0	57.2	-0.2	18.9	45.1	57.2	+5.6	12.2	47.6	56.9	**+5.7**		57.3	57.8
bg	26.4	67.5	78.9	59.6	26.3	67.5	78.9	-0.1	35.5	65.1	74.4	+1.5	27.2	67.6	78.9	**+1.8**		79.6	79.0
ca	28.4	62.3	78.5	57.8	27.9	62.0	78.7	-0.1	33.5	60.8	75.5	-0.2	30.4	66.2	78.6	**+1.9**		79.2	79.1
cs	29.7	58.5	74.0	54.3	29.2	58.6	74.0	-0.0	37.1	58.4	68.8	**+1.4**	30.8	59.3	73.8	+1.4		74.0	73.8
cu	22.6	58.5	74.7	53.9	22.6	58.7	75.4	+0.0	38.2	60.2	70.0	**+3.8**	24.3	60.3	74.7	+1.6		74.7	74.7
da	28.0	64.5	75.3	58.2	27.5	63.8	75.3	-0.2	40.3	61.0	70.4	-0.0	28.6	64.6	74.5	**+1.4**		75.7	75.2
de	36.2	61.2	70.5	57.7	35.9	61.0	70.0	-0.2	45.4	61.1	69.3	+1.3	43.0	61.8	70.8	**+1.5**		68.7	68.7
el	29.0	51.0	67.8	49.5	28.9	50.5	68.2	-0.0	33.5	51.6	64.9	+0.4	29.8	51.6	67.6	**+1.7**		67.0	66.2
en	33.1	56.5	65.8	52.8	32.5	56.2	65.5	-0.2	38.4	57.0	63.8	**+1.2**	32.2	58.4	65.9	+1.2		65.7	66.0
es	30.0	63.9	78.5	58.9	29.3	64.4	78.5	-0.0	37.6	62.8	76.1	+0.3	31.5	66.6	78.4	**+1.8**		79.2	79.3
et	28.4	53.1	69.4	51.5	26.9	52.9	69.6	-0.2	36.3	57.0	67.9	+3.7	37.1	57.9	69.3	**+4.4**		69.4	69.3
eu	20.7	45.2	57.8	44.2	20.9	46.4	57.7	-0.1	24.3	54.6	64.1	**+8.1**	24.6	52.0	63.3	+5.7		55.7	60.9
fa	17.9	45.3	56.1	40.3	17.6	45.0	56.0	-0.1	26.7	46.3	56.8	+3.2	25.5	48.4	58.8	**+5.2**		61.4	63.3
fi	27.4	48.1	62.1	46.6	27.4	48.2	61.8	-0.0	32.4	50.2	58.8	+2.3	32.4	53.0	62.2	**+3.7**		62.1	62.2
fr	30.9	64.0	79.1	59.0	29.9	63.9	78.7	-0.2	35.0	61.4	76.8	+0.5	34.2	66.0	78.9	**+1.8**		79.8	79.5
ga	16.4	56.0	65.8	50.1	16.3	56.2	66.4	-0.1	26.6	56.2	64.6	+1.8	20.8	59.0	65.3	**+2.3**		67.4	67.2
gl	33.0	40.3	47.5	40.6	32.8	40.5	47.5	-0.1	32.1	43.0	48.2	+1.5	35.6	43.7	51.0	**+2.6**		46.7	46.6
got	26.4	58.0	72.7	54.3	26.4	57.5	73.4	-0.0	38.0	60.0	66.3	**+3.1**	28.2	58.9	73.9	+0.9		72.7	73.9
grc	32.5	53.9	57.3	50.7	29.8	53.9	57.8	-0.1	39.0	53.9	58.3	**+1.1**	32.4	53.9	57.8	+0.0		61.0	60.3
he	20.1	53.8	68.0	49.9	19.8	54.2	67.7	+0.1	30.2	54.1	63.6	+1.2	21.9	55.4	65.8	**+1.6**		71.2	68.7
hi	11.0	27.1	66.5	32.3	11.1	26.9	65.8	-0.1	22.0	37.5	61.6	**+6.7**	19.8	33.8	66.8	+5.4		37.1	44.2
hr	26.8	55.4	71.2	52.0	26.0	56.3	70.9	-0.2	35.7	56.3	66.9	**+1.7**	28.9	56.3	70.3	+1.5		73.9	73.0
hu	27.8	52.7	67.8	50.8	27.1	53.1	68.2	-0.0	40.4	56.3	65.4	**+4.4**	40.1	55.4	68.3	+3.4		63.0	64.4
id	17.4	49.2	70.1	48.8	18.2	49.0	70.3	+0.1	27.6	50.2	66.1	+1.5	23.1	53.8	69.6	**+3.7**		70.8	71.9
it	31.0	67.1	82.6	61.7	30.4	66.9	82.2	-0.1	38.1	67.0	80.7	+1.2	34.0	70.8	82.3	**+2.2**		83.2	82.9
ja	7.0	18.6	72.6	26.7	7.2	18.3	72.2	+0.1	15.7	32.9	70.6	+10.0	18.1	35.2	72.3	**+11.4**		63.3	63.5
kk	10.7	33.0	56.3	32.4	10.9	34.0	54.3	+0.2	17.9	35.2	52.6	+3.4	20.6	38.5	55.6	**+4.9**		53.9	54.6
la	14.4	49.9	64.1	47.1	14.3	50.0	63.5	+0.0	19.5	51.4	61.8	+1.9	21.1	53.3	63.3	**+2.1**		58.3	60.0
lv	22.6	40.3	55.7	40.6	22.5	40.1	55.8	-0.2	28.7	42.6	51.0	+1.8	35.0	47.1	55.4	**+5.5**		50.2	57.3
nl	27.5	51.9	61.7	48.9	27.9	52.2	62.2	+0.1	32.4	49.3	56.8	-2.4	28.4	52.4	60.4	**+0.5**		62.3	60.7
no	25.8	64.0	76.6	57.1	25.6	63.9	76.7	-0.1	37.2	58.5	69.2	-0.2	26.5	63.8	76.2	**+1.3**		76.6	76.5
pl	25.7	62.1	77.9	59.2	25.6	62.7	77.9	-0.1	36.0	65.6	76.2	**+3.4**	29.5	65.5	77.4	+2.4		78.0	77.6
pt	30.8	62.8	75.5	56.7	30.2	63.3	75.5	+0.0	35.7	60.0	73.5	-0.9	32.8	63.5	75.4	**+1.4**		75.5	75.7
ro	19.8	55.5	69.2	51.8	18.6	55.7	68.7	-0.1	31.7	58.5	67.7	+3.1	24.1	60.5	70.3	**+4.0**		71.8	72.0
ru	26.8	53.9	69.0	51.3	26.1	54.0	68.9	-0.0	34.6	55.1	67.9	+2.3	30.9	59.2	68.7	**+4.2**		71.0	70.4
sl	30.6	65.2	80.4	59.4	30.4	65.1	80.5	-0.0	41.8	64.8	77.4	**+2.7**	30.5	64.9	80.4	+1.4		80.4	80.4
sv	29.4	62.7	75.5	56.9	29.4	62.3	75.5	-0.2	39.6	60.1	70.6	+0.5	30.4	63.9	74.9	**+2.3**		73.1	73.5
ta	9.1	36.3	66.3	36.8	9.1	35.6	66.4	-0.0	18.9	43.7	64.5	**+6.2**	19.0	41.1	65.8	+4.7		66.3	65.8
tr	14.1	35.3	67.0	38.8	14.7	35.4	67.0	-0.1	19.5	39.5	64.5	+2.0	21.5	40.8	67.8	**+3.5**		58.6	58.5
zh	15.6	32.5	43.1	31.7	15.8	32.5	43.0	+0.1	18.9	35.5	41.8	+2.3	20.1	36.6	44.1	**+3.5**		40.2	42.2
Avg	23.7	52.0	68.2	49.2	23.3	52.0	68.1	-0.1	31.8	53.5	65.6	+2.3	27.9	55.2	68.3	+2.9		66.9	67.4

Table 2: UAS of the various mono-source and multi-source transfer methods, on each UD target language (using UD language codes).
The first line reads as follows: for delexicalized transfer to Arabic, the worst, median and best sources yield UAS scores of 5.1, 43.2 and 56.9, and the average score over all 39 sources is 36.1, which the WALS-based method improves by 5.7 points.

Our experiments also show that the selection baseline method does not perform as well on Universal Dependencies (Nivre et al., 2016) as it did on the CoNLL 2006 Shared Task. Those differences can be explained in two ways. First, we experiment with cleaner treebanks and benefit from the availability of unified tagsets and annotation schemes. This is in contrast with previous experiments, which were using a tagset mapping as a preprocessing step, making the net effect of data selection more difficult to single out and evaluate precisely. Second, the data selection method was primarily intended for distantly related languages, whereas the UD corpus now offers a wide language diversity and often a few good sources for which data size reduction is only detrimental.

In general, our methods do not improve the best source but have a large effect on bad and average sources, often turning them into competitive sources. This is particularly true with PoSLM reordering, which improves the worst sources by 8.1 points and degrades the best ones by 2.6 points. By contrast, the WALS-based method is more conservative and offers lower but more reliable improvements, which in average proves successful.

Table 3 reports the average over some language families[12] of the UAS of the baseline, reordered and WALS-based mono-source models. It shows that accuracies of related sources are only marginally mod-

[12]While the considered ancient languages belong to some of those families, we chose to gather them into a separate category, since they rely on the same WALS completion heuristic, instead of their actual typological features.

Source language	Target language: Romance	Germanic	Slavic	Finno-Ugric	Semitic	Ancient
Romance	67.1‖65.6‖67.2	60.4‖60.4‖61.7	63.1‖63.5‖63.0	46.4‖50.8‖52.5	54.1‖52.1‖52.9	56.7‖56.5‖54.9
Germanic	61.2‖63.5‖65.8	65.9‖63.1‖65.8	61.3‖62.2‖63.2	57.2‖58.6‖58.5	41.2‖48.2‖49.8	54.5‖57.1‖56.7
Slavic	63.5‖61.7‖66.0	63.8‖60.5‖64.3	72.6‖68.4‖71.8	53.2‖57.0‖58.4	54.7‖53.6‖56.8	59.0‖59.2‖60.1
Finno-Ugric	46.3‖51.9‖52.3	57.1‖56.2‖57.6	53.8‖58.6‖56.9	64.1‖63.0‖64.2	30.0‖43.6‖41.5	50.8‖55.7‖56.1
Semitic	54.1‖54.2‖54.1	40.6‖48.2‖51.1	42.5‖54.6‖56.1	30.8‖41.2‖44.1	55.4‖55.6‖54.8	53.7‖55.9‖54.4
Ancient	56.1‖49.2‖55.9	56.7‖51.5‖56.1	60.9‖57.5‖60.6	52.2‖54.9‖56.0	51.1‖47.0‖50.6	62.7‖60.0‖62.6

Table 3: Delexicalized, PoSLM-based reordered and WALS-based UAS aggregated over language family pairs.
The first column reads as follows: the average UAS over all pairs of two Romance languages is 67.1 for mono-source delexicalized transfer; it is slightly improved (67.2) by the WALS-based method. Over all pairs of a Germanic source and a Romance target, the average mono-source UAS raises from 61.2 (delexicalized baseline) to 63.5 (PoSLM-based reordering) and 65.8 (WALS-based rules).

ified when source sentences are transformed according to WALS, which could be expected as related languages share most of their typological features. On the contrary, large gains are obtained for distantly related languages. Such languages are typically poor sources in direct delexicalized transfer due to systematic labeling errors that mostly concern few frequent word classes (in correlation with their typological features). We have found that such errors can often be corrected by transforming the source sentences. With those errors handled, the now competitive sources can in turn contribute with valuable knowledge in multi-source settings.

4.3 A Fine-grained Analysis

We have also investigated the improvements made over the baseline by our best method, the WALS-based rewriting rules, by analyzing the gain in accuracy separately for various PoS. It appears that, in most cases, improvements mostly concern PoS classes covered by the WALS features. For instance, the issue mentioned in § 2.2 for the English-French pair is almost solved with source reordering: 90.4% of the postposed ADJs are correctly predicted by the WALS-based method (34.5% in the baseline), without any detrimental impact on the preposed ones. The same holds for the Hebrew-Bulgarian textbook case, where the UAS $\left[\substack{\text{NOUN}\\\text{ADJ}}\right]$ raises from 0.7% to 95.1%.

We observe similar behaviors across the board for all the classes targeted by transformations: transfer from Czech to Danish had UAS $\left[\substack{\text{NOUN}\\\text{preposed NOUN}}\right]$ and UAS $\left[\substack{\text{NOUN}\\\text{postposed NOUN}}\right]$ scores of 2.8% and 78.4%, with WALS-based preprocessing they are respectively 61.1% and 80.4%. In Finnish-Arabic, scores of 6.3% and 30.9% on ADJs and ADPs become 65.8% and 61.4%, etc. In whole, 21% of the considered language pairs present very large gains (over +50 points) for at least one frequent tag pair (over 30 dependency occurrences in test data).

Careful comparison of results for both PoSLM-based methods shows that reordering improves ADJs' attachment for instance, when data selection does not. This can be explained in two ways. First, if the source corpus contains a very limited number of preposed ADJs, even with perfect selection the ADJs in final data cannot be mostly preposed. Second, data selection mostly targets sequences very far from the target syntax: sentences that only disrespect a local preference of child position are less fluent, and consequently have a lower rank, than their hypothetical counterpart with switched positions; but they are not ungrammatical enough to be pushed into the 10% worst territory. On the other hand, the data transformation approach is not restricted to preexisting n-grams, and it directly confronts the given sequence with its counterpart to keep only the most fluent, thus acknowledging local preferences. These key differences are confirmed experimentally on English-French data: PoSLM-based reordering lowers the precedence rate of ADJs to NOUNs from 93% to 60%, while the rate varies by less than 1% in Søgaard (2011)'s approach, leaving the majority case adjectives still under-trained.

Finally, detailed analyses reveal that the PoSLM reordering approach has lower improvements than the WALS-based one on *easy reorderings* such as the nearly deterministic Hebrew-Bulgarian adjectives

(UAS $\left[\begin{smallmatrix}\text{NOUN}\\ \text{ADJ}\end{smallmatrix}\right]$ of 93.2% with WALS, versus 86.1% with the PoSLM). This suggests that the data-driven technique still wastes part of the available knowledge: indeed, the use of a probabilistic model to rank reorderings does not guarantee that any interesting reordering is in fact selected. Another advantage of the knowledge-rich approach is that the distribution of *local* word orderings is easier to control, since it explicitly regulates the balance between co-existing word orders. Indeed, when two structures are possible and fluent, the PoSLM-based method will always prefer the majority class. While the projectivity requirement generally softens this hard constraint by discarding many reordering candidates, the effect holds for instance on typologically close languages: during transfer from French ($\mathrm{PR} = 28\%$ for ADJ-NOUN) to Italian ($\mathrm{PR} = 32\%$), PoSLM-based reorderings harden the preference down to $\mathrm{PR} = 16\%$, and end up under-training the ADJ-NOUN minority class.

In spite of this, the PoSLM reordering is still competitive, since it covers more diverse phenomena. For instance, when transferring from English to Tamil, the $\mathrm{UAS}\left[\begin{smallmatrix}\text{VERB}\\ \text{AUX}\end{smallmatrix}\right]$ only raises from 31.4% to 35.0% with the WALS-based method, but achieves a nice 91.2% with the PoSLM. Such improvements are however less predictable and unexplained losses also occur, as for the $\mathrm{UAS}\left[\begin{smallmatrix}\text{VERB}\\ \text{AUX}\end{smallmatrix}\right]$ in Hungarian-Tamil (98.5% for the baseline and WALS, 66.4% with PoSLM reordering).

These results suggest that both approaches have their upsides and downsides, which remain to be combined.

5 Related Work

The observation that cross-lingual transfer works better with typologically close or related languages has been already made by many. Indeed, several works have already pointed out that unified annotation schemes cannot compensate for syntactic divergences between source and target languages and that reducing these divergences was likely to improve the performance of transfer.

When several sources are available, a natural approach is to give more weight to the instances observed in related languages, where relatedness can be measured either based on linguistic description (Berg-Kirkpatrick and Klein, 2010) or empirically (Cohen et al., 2011).

Søgaard (2011) follows similar intuitions but binarizes the weights to apply instance selection. Thus, the delexicalized model is trained only on the source examples that are the most *relevant* for the target at hand, using PoSLM perplexity as a relevance metric. Note that this strategy can be applied both in mono-source and multi-source settings.

In Rosa and Zabokrtsky (2015)'s work, the syntactic similarity between languages is also based on the similarity between their PoS sequences. They show how the KL_{cpos^3} measure can be used to improve cross-lingual transfer either by selecting the best source language, or by weighting the source contribution to the output in a multi-source setting.

Both multi-source combination and data selection follow the same intuition that any source sentence or part of it can provide useful information on the syntax of the target language, even when the divergence between the source and the target is large. Indeed, a language is subject to many influences throughout its evolution and can borrow a phenomenon from a very distant language. This is for instance the case of Romanian, which belongs to the Romance family but has also strong Slavic influences.

As a result, both works aim at extracting useful knowledge even from poor sources, and our proposal can be viewed as an extension that pushes further this intuition, to a finer grain: Rosa and Zabokrtsky (2015) reward good source languages, Søgaard (2011) rewards target-relevant sentences, and our method rewards relevant local patterns, by performing a local reordering of target-irrelevant parse subtrees rather than ignoring the whole sentence. This has the effect of using the knowledge embedded in these subtrees as well as in the rest of the sentence more effectively. To see this, consider the case of transferring an English parser to a language in which no verb is labeled as auxiliary.[13] In this case, the method of (Søgaard, 2011) is likely to discard all the English sentences containing auxiliaries and the parser will hardly see, in training, sentences involving passive constructions or past participles; by contrast, methods based on data transformation would not remove the full sentence, but just the auxiliary – all the other dependencies, e.g. the by-agent, can still contribute to learning.

[13] In the UD treebank this is, for instance, the case for Greek, Latvian or Galician.

Thus, in comparison to previous works favoring close word orders at the cost of discarding some training examples or reducing source contribution, our method differs by improving cross-lingual transfer without knowledge loss.

In another line of work, Naseem et al. (2012) also distinguish the knowledge 'ADJs depend on NOUNs' from the ordering of both tokens, and use WALS to predict the latter. However, where our methods compensate for word order divergences at the data level, their work aims at abstracting the dependency prediction from word order, by designing a new parsing algorithm from scratch: their parser decomposes as a dependent selection component, shared among languages, and an ordering component that is specific to the target language. Even though it does not provide full order abstraction, our approach has the double advantage of wrapping any state-of-the-art parsing system, and allowing an extra degree of flexibility by manipulating the data, e.g. to handle PoS classes existing in only one language.

6 Conclusion

The contribution of this work is twofold. First, we have updated earlier results on delexicalized cross-lingual model transfer by reproducing them on the recent Universal Dependencies treebank. This collection of treebanks contains more languages than were previously available. Furthermore, the consistency of annotation schemes makes the analysis of results more reliable and enables to draw firmer conclusions. Second, based on a thorough analysis of the weaknesses of delexicalized transfer, we have proposed two strategies that aim to compensate for word sequence biases when transferring models across languages: a data-driven method using PoSLMs for reordering and a knowledge-based method exploiting heuristic rewrite rules extracted from WALS. The latter method proved to be the most effective of the two, with the additional benefit of being entirely resource-free and thus readily usable for the over thousand languages whose word order is specified in WALS. For the frequent PoS classes targeted by this method, we were able to obtain huge improvements, often 30 and up to 90 points.

A first natural continuation of this work will be to complete our repertoire of preprocessing rules with article insertions, PoS substitutions and patterns involving verbs, which were not considered so far. Another direction we would like to investigate is to contrast our techniques with annotation projection, which is another way to compensate for word order biases in cross-lingual transfer: by analyzing the pros and cons of each method we might find ways to combine them so that we can also use parallel data when available. We finally also aim at generalizing our WALS approach to other order-dependent tasks. Indeed, from a higher-level point of view, the aforementioned issues are not specific to dependency parsing, but occur theoretically with all data-driven NLP methods: however general it is, the linguistic knowledge is always only available as instantiated on a given word sequence and through the proxy of a particular data.

Acknowledgements

This work has been partly funded by the French *Direction générale de l'armement* and by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 645452 (QT21). We thank the anonymous reviewers for their detailed and inspiring comments on the paper.

References

Lauriane Aufrant and Guillaume Wisniewski. 2016. PanParser: a Modular Implementation for Efficient Transition-Based Dependency Parsing. Technical report, LIMSI-CNRS, March.

Taylor Berg-Kirkpatrick and Dan Klein. 2010. Phylogenetic grammar induction. In *The 48th Annual Meeting of the Association for Computational Linguistics*, pages 1288–1297, Uppsala, Sweden.

Arianna Bisazza and Marcello Federico. 2016. A Survey of Word Reordering in Statistical Machine Translation: Computational Models and Language Phenomena. *Computational Linguistics*, 42(2):163–205.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised Structure Prediction with Non-Parallel Multilingual Guidance. In *Proceedings of EMNLP 2011, the Conference on Empirical Methods in Natural Language Processing*, pages 50–61, Edinburgh, Scotland, UK., July.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04.

Markus Dreyer, Keith Hall, and Sanjeev Khudanpur. 2007. Comparing reordering constraints for smt using efficient bleu oracle computation. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 103–110, Rochester, New York, April. Association for Computational Linguistics.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 138–147, Atlanta, Georgia.

Yoav Goldberg and Joakim Nivre. 2012. A Dynamic Oracle for Arc-Eager Dependency Parsing. In *Proceedings of COLING 2012, the International Conference on Computational Linguistics*, pages 959–976, Bombay, India.

Rebecca Hwa, Philip Resnik, A.Weinberg, C. Cabezas, and O. Kolak. 2005. Bootstrapping Parsers via Syntactic Projection accross Parallel Texts. *Natural language engineering*, 11:311–325.

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 161–168, Vancouver, British Columbia, Canada.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Frustratingly Easy Cross-Lingual Transfer for Transition-Based Dependency Parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1058–1063.

Adam Lopez. 2009. Translation as weighted deduction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 532–540, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal Dependency Annotation for Multilingual Parsing. In *Proceedings of ACL 2013, the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria, August.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 629–637.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA).

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Rudolf Rosa and Zdenek Zabokrtsky. 2015. KL_{cpos^3} - a Language Similarity Measure for Delexicalized Parser Transfer. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 243–249, Beijing, China. Association for Computational Linguistics.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *Proceedings of ACL 2011, the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 682–686, Portland, Oregon, USA, June.

Jörg Tiedemann, Željko Agić, and Joakim Nivre. 2014. Treebank translation for cross-lingual parser induction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning (CoNLL 2014)*, pages 130–140, Ann Arbor, Michigan.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, NAACL '01, pages 1–8, Stroudsburg, PA, USA.

Daniel Zeman and Philip Resnik. 2008. Cross-Language Parser Adaptation between Related Languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*, pages 35–42, Hyderabad, India, January. Asian Federation of Natural Language Processing.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA.

Improving historical spelling normalization with bi-directional LSTMs and multi-task learning

Marcel Bollmann
Department of Linguistics
Ruhr-Universität Bochum
Germany
bollmann@linguistics.rub.de

Anders Søgaard
Dpt. of Computer Science
University of Copenhagen
Denmark
soegaard@hum.ku.dk

Abstract

Natural-language processing of historical documents is complicated by the abundance of variant spellings and lack of annotated data. A common approach is to normalize the spelling of historical words to modern forms. We explore the suitability of a deep neural network architecture for this task, particularly a deep bi-LSTM network applied on a character level. Our model compares well to previously established normalization algorithms when evaluated on a diverse set of texts from Early New High German. We show that multi-task learning with additional normalization data can improve our model's performance further.

1 Introduction

Interest in computational processing of historical documents is on the rise, as evidenced by the growing field of digital humanities and the increasing number of digitally available resources of historical data. Spelling normalization, i.e. the mapping of historical spelling variants to standardized/modernized forms, is often employed as a pre-processing step to allow the utilization of existing tools for the respective modern target language (Piotrowski, 2012).

Training data for supervised learning of spelling normalization is typically scarce in the historical domain. Furthermore, dialectal influences and even individual preferences of an author can have a huge impact on the spelling characteristics in a particular text, meaning that even training data from other corpora of the same language and time period cannot always be reliably used.

Algorithms have often been developed with this fact in mind, e.g. by being based on some form of phonetic, graphematic, or semantic similarity measure (Jurish, 2010; Bollmann, 2012; Amoia and Martinez, 2013). On the other hand, neural networks – and particularly deep networks with several hidden layers – are assumed to work best when trained on large amounts of data. It is therefore not clear whether neural networks are a good choice for this particular domain.

We frame spelling normalization as a character-based sequence labeling task, and explore the suitability of a deep bi-directional long short-term memory model (bi-LSTM) in this setting. By basing our model on individual characters as input, along with performing some basic preprocessing (e.g., downcasing all characters), we keep the vocabulary size small, which in turn reduces the model's complexity and the amount of data required to train it effectively. We show that this model outperforms both the existing normalization tool Norma (Bollmann, 2012) and a CRF-based tagger when evaluated on a diverse dataset from Early New High German.

Furthermore, we experiment with a multi-task learning setup using auxiliary data that has similar, but not identical spelling characteristics to the target text. We show that using bi-LSTMs with this multi-task learning setup can improve normalization accuracy further, while Norma and CRF do not profit much from the additional data in a traditional setup.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 131–139, Osaka, Japan, December 11-17 2016.

2 Datasets

We use a total of 44 texts from the Anselm corpus (Dipper and Schultz-Balluff, 2013) of Early New High German.[1] The corpus is a collection of several manuscripts and prints of the same core text, a religious treatise. Although the texts are semi-parallel and share some vocabulary, they were written in different time periods (between the 14th and 16th century) as well as different dialectal regions, and show quite diverse spelling characteristics. For example, the modern German word *Frau* 'woman' can be spelled as *fraw/vraw* (Me), *frawe* (N2), *frauwe* (St), *fraüwe* (B2), *frow* (Stu), *vrowe* (Ka), *vorwe* (Sa), or *vrouwe* (B), among others.[2]

All texts in the Anselm corpus are manually annotated with gold-standard normalizations following guidelines described in Krasselt et al. (2015). For our experiments, we excluded texts from the corpus that are shorter than 4,000 tokens, as well as a few texts for which annotations were not yet available at the time of writing (mostly Low German and Dutch versions). Nonetheless, the remaining 44 texts are still quite short for machine-learning standards, ranging from about 4,200 to 13,200 tokens, with an average length of 7,353 tokens.

For all texts, we removed tokens that consisted solely of punctuation characters. We also lowercase all characters, since it helps keep the size of the vocabulary low, and uppercasing of words is usually not very consistent in historical texts.

2.1 Conversion to labeled character sequences

Normalization is annotated on a word level; to reframe the problem as a character-based sequence labeling task, we need to align the historical wordforms and their normalizations on a character level. Ideally, we would like these alignments to be linguistically plausible, i.e., characters that most likely correspond to each other (e.g., historical *j* and modern *i*, as in *jn – ihn* 'him') should be aligned whenever possible.

The Levenshtein algorithm (Levenshtein, 1966) can be used to produce alignments that preferably align identical characters, but is ambiguous when multiple alignments with the same Levenshtein distance exist. We therefore use iterated Levenshtein distance alignment (Wieling et al., 2009), which uses pointwise mutual information on aligned segments to estimate statistical dependence, and favors alignments of characters that tend to cooccur often within the dataset. Since different texts can use the same characters in different ways, we perform this iterated alignment separately for each text.

A difficulty of these alignments is that the two wordforms can be of different lengths. We introduce a special epsilon label (ϵ) whenever a historical character is not aligned to any character in the normalization. We cannot do that for the inverse case, since the historical characters are our units of annotation and therefore need to be fixed, so we choose to perform a leftward merging of normalized characters whenever they are not aligned to any character in the historical wordform. For the word-initial case, we introduce a special "start of word" symbol (_). This symbol is prepended to each word during both training and testing, and is assigned the epsilon label during training when there is no word-initial insertion.

Here is an example of the final character sequence representation for the word pair *vsfuret – ausführt* '(he) leads out':

(1) _ v s f u r e t
a u s f üh r ϵ t

A consequence of this approach is that our labels cannot only be characters, but also combinations of characters (such as *üh* in the example above); our label set is therefore potentially unbounded. However, we found that this is not much of a problem in practice, since these cases tend to be comparatively rare.

3 Model

Our model architecture consists of: (i) an embedding layer for the input characters; (ii) a stack of bi-directional long short-term memory units (bi-LSTMs); and (iii) a final dense layer with a softmax activation to

[1] https://www.linguistics.rub.de/anselm/

[2] Abbreviations in brackets refer to individual texts using the same internal IDs that are found in the Anselm corpus.

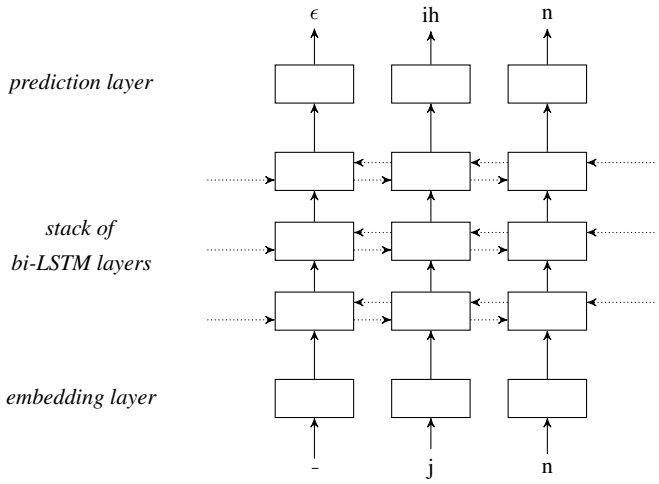


Figure 1: Flow diagram of the bi-LSTM character sequence labeling model, unrolled for time, for the word pair *jn – ihn* 'him'.

generate a probability distribution over the output classes at each timestep. An illustration of the model can be found in Figure 1.

The embedding layer maps one-hot input vectors (representing historical characters) to dense vectors. We did not use pre-trained embeddings; the embeddings are initialized randomly and learned as part of the regular network training process.

LSTMs (Hochreiter and Schmidhuber, 1997) are a form of recurrent neural network (RNN) designed to better learn long-term dependencies, and have proven advantageous to plain RNNs on many tasks. Bi-directional LSTMs read their input in both normal and reversed order, allowing the model to learn from both left and right context at each input timestep. A stack of bi-LSTMs, or a deep bi-LSTM, is a configuration of several bi-LSTM units so that the output of the ith unit is the input of the $(i+1)$th unit. In our model, we use a stack of three bi-LSTM layers.

The final dense layer is used to generate the output predictions, based on a linear transformation of the bi-LSTM outputs for each timestep followed by a softmax activation. We train the model by minimizing the cross-entropy loss across all output characters, and using backpropagation to update the weights in all layers (including the embedding layer). During prediction, we generate output labels in a greedy fashion, choosing the label with the highest probability for each timestep.

3.1 Multi-task learning setup

In multi-task learning (MTL), the performance of a model on a given task is improved by additionally training it on one or more auxiliary tasks (Caruana, 1993). For our bi-LSTM model, this means that all layers of the model are shared between the tasks apart from the final prediction layer, which is kept separate for the main and auxiliary tasks. This way, errors in an auxiliary task that are backpropagated through the network also affect the prediction of the main task, helping to regularize the network's weights and prevent overfitting.

Multi-task learning with (deep) neural network architectures was shown to be effective for a variety of NLP tasks, such as part-of-speech tagging, chunking, named entity recognition (Collobert et al., 2011); sentence compression (Klerke et al., 2016); or machine translation (Luong et al., 2016).

In our experiments, we regard spelling normalization within a target domain (i.e., a given historical text) as our main task, while using normalization within related domains (i.e., texts from a similar time period, but with distinct spelling characteristics) as our auxiliary task. During training, we alternate between training on a random instance from the main and the auxiliary tasks.

3.2 Hyperparameters

We set aside one of the texts (B) from the Anselm corpus for testing different hyperparameter configurations. On this text, we achieved the best results with a dimensionality of 128 for the embedding and bi-LSTM layers, using a dropout of 0.1, and training the model for 30 iterations. These settings were subsequently used for all further experiments.

3.3 Other models used for comparison

For comparison, we also train and evaluate with the Norma tool described by Bollmann (2012), since it was originally developed for the Anselm corpus and the implementation is publicly available.[3] However, Norma actually consists of a combination of three different normalization methods, one of which is a simple wordlist mapping of historical tokens to normalized forms. Since this wordlist mapping is conceptually very simple and could easily be added to our (or any other) normalization method, we exclude it for the comparison, and only use Norma's remaining two algorithms (which we denote Norma*).

Additionally, since we frame the problem as a sequence labeling task, we compare our results to a simple sequence labeling model using conditional random fields (CRF). The CRF model gets the same input/output sequences as our bi-LSTM model (cf. Sec. 2.1), and uses the two preceding and following characters from the historical wordform as additional features. Implementation was done with CRFsuite (Okazaki, 2007) using the averaged perceptron algorithm for training.

4 Evaluation

We evaluate our model separately for each text in our dataset. From each text, we use 1,000 tokens as our evaluation set, set aside another 1,000 tokens as a development set (which was not currently used), and train on the remaining tokens (between 2,000 and 11,000, depending on the text). Both CRF and our bi-LSTM model get their input as character sequences (as described in Sec. 2.1), while Norma requires full words as input.

For the multi-task learning setup, we randomly sample from all Anselm texts and regard each text as its own task. Effectively, we are learning a joint model over all Anselm texts with shared parameters but distinct prediction layers, while viewing the text we are currently evaluating on as our main task and the others as auxiliary tasks. The MTL setup is only applicable to our bi-LSTM model; however, since the auxiliary task consists of spelling normalization with data from the same corpus (although with a higher variety of different spelling characteristics compared to the target text), it is possible that the other methods could also profit from this additional training data. We therefore also evaluate Norma and CRF when the training sets have been augmented by 10,000 randomly sampled training examples from all texts.

4.1 Word accuracy

Evaluation results in terms of word-level accuracy are presented in Table 1.

Columns "S" show results for the traditional setup without multi-task learning. The basic bi-LSTM model performs better than Norma on 34 of the 44 texts. On average, there is an increase of 2.1 percentage points (pp), although the differences on individual texts vary wildly, from -2.9 pp (M5) to $+9.6$ pp (M), giving a standard deviation of 2.7 pp. The CRF model, on the other hand, is almost always worse than Norma, averaging a difference of -2.1 pp (± 2.0). This indicates that the reformulation of the task as character-based sequence labeling cannot alone be responsible for the bi-LSTM results, but the choice of a neural network architecture is crucial, too.

Columns "S+A" present the results when using the augmented training set. For bi-LSTM, this is the multi-task learning setup—using MTL improves the results by $+0.7$ pp (± 2.8) on average, but again there is a high variance within the individual scores. However, for the other methods, adding the 10,000 randomly selected samples to the training set actually decreases the average accuracy, by -0.4 pp for Norma and -2.0 pp for CRF. This is likely due to the fact that this additional training set introduces a variety of spelling characteristics that are not found in the target text. While Norma and

[3] https://github.com/comphist/norma

ID	Region	Tokens	Norma*		CRF		Bi-LSTM	
			S	S+A	S	S+A	S	S+A†
B	East Central	4,718	80.30%	77.80%	76.30%	72.80%	79.20%	**81.70%**
D3	East Central	5,704	80.50%	80.20%	77.20%	73.00%	80.10%	**81.20%**
H	East Central	8,427	82.70%	82.90%	78.60%	76.00%	**85.00%**	82.30%
B2	West Central	9,145	76.10%	77.60%	74.60%	71.70%	**82.00%**	79.60%
KÄ1492	West Central	7,332	77.50%	74.40%	74.80%	68.40%	**81.60%**	80.50%
KJ1499	West Central	7,330	77.00%	72.90%	73.50%	68.40%	**84.50%**	79.20%
N1500	West Central	7,272	76.70%	75.30%	72.70%	67.20%	79.00%	**79.20%**
N1509	West Central	7,418	78.10%	73.30%	74.30%	68.80%	**80.80%**	80.10%
N1514	West Central	7,412	78.30%	73.80%	72.20%	69.90%	79.00%	**80.10%**
St	West Central	7,407	72.60%	73.80%	70.30%	68.70%	**75.50%**	75.20%
D4	Upper/Central	5,806	75.60%	75.60%	72.40%	70.90%	76.50%	**76.60%**
N4	Upper	8,593	78.20%	78.10%	80.00%	78.40%	81.80%	**83.40%**
s1496/97	Upper	5,840	81.70%	83.40%	77.70%	76.90%	83.00%	**84.10%**
B3	East Upper	6,222	80.80%	80.60%	79.50%	79.10%	81.50%	**83.20%**
Hk	East Upper	8,690	77.80%	79.30%	78.20%	77.90%	80.90%	**82.20%**
M	East Upper	8,700	74.30%	74.40%	72.80%	68.40%	**83.90%**	80.90%
M2	East Upper	8,729	75.80%	76.00%	75.10%	72.40%	76.70%	**80.20%**
M3	East Upper	7,929	79.00%	79.70%	77.30%	74.10%	**80.40%**	79.60%
M5	East Upper	4,705	80.60%	80.70%	76.40%	78.30%	77.70%	**82.90%**
M6	East Upper	4,632	75.90%	76.30%	73.70%	74.40%	75.20%	**79.30%**
M9	East Upper	4,739	82.20%	81.50%	79.00%	76.90%	80.40%	**83.60%**
M10	East Upper	4,379	77.00%	78.60%	76.00%	75.80%	75.10%	**81.30%**
Me	East Upper	4,560	79.70%	80.10%	76.90%	75.50%	80.30%	**83.70%**
Sb	East Upper	7,218	78.00%	76.60%	75.70%	74.80%	**80.00%**	78.50%
T	East Upper	8,678	76.70%	78.60%	73.40%	72.20%	75.80%	**79.00%**
W	East Upper	8,217	75.90%	78.30%	78.20%	77.00%	**81.40%**	80.80%
We	East Upper	6,661	**83.10%**	81.50%	78.60%	75.80%	81.50%	**83.10%**
Ba	North Upper	5,934	79.80%	81.20%	80.20%	78.70%	80.70%	**82.80%**
Ba2	North Upper	5,953	81.40%	80.00%	78.10%	77.90%	82.50%	**84.10%**
M4	North Upper	8,574	76.90%	76.70%	75.70%	75.00%	79.40%	**82.30%**
M7	North Upper	4,638	79.40%	79.80%	75.60%	74.20%	78.20%	**82.10%**
M8	North Upper	8,275	78.50%	77.00%	78.20%	78.40%	81.10%	**82.50%**
n	North Upper	9,191	79.60%	81.30%	81.90%	78.20%	84.40%	**84.70%**
N	North Upper	13,285	75.50%	76.30%	71.70%	68.90%	**79.00%**	76.90%
N2	North Upper	7,058	82.20%	81.90%	80.30%	81.60%	**84.30%**	83.40%
N3	North Upper	4,192	79.10%	80.80%	76.40%	77.50%	77.60%	**84.20%**
Be	West Upper	8,203	75.50%	76.40%	75.30%	73.40%	**78.80%**	78.00%
Ka	West Upper	12,641	73.80%	74.10%	75.40%	72.80%	80.10%	**80.30%**
SG	West Upper	7,838	80.10%	79.90%	78.00%	76.80%	**81.70%**	80.90%
Sa	West Upper	8,668	72.60%	73.50%	71.90%	71.40%	76.10%	**76.50%**
St2	West Upper	8,834	73.20%	73.40%	73.20%	73.00%	78.20%	**79.90%**
Stu	West Upper	8,686	77.70%	77.10%	76.50%	72.10%	**79.40%**	77.00%
Sa2	West Upper	8,011	77.50%	77.90%	73.50%	73.30%	79.50%	**79.70%**
Le	Dutch	7,087	69.50%	60.30%	65.00%	55.80%	**75.60%**	67.50%
Average		7,353	77.83%	77.48%	75.73%	73.70%	79.90%	**80.55%**

Table 1: Word accuracy on the Anselm dataset, evaluated on the first 1,000 tokens; S = training set from the same text, S+A = like S, but augmented with 10,000 tokens randomly sampled from the other texts; † = Bi-LSTM (S+A) is the multi-task learning setup. Best results shown in bold.

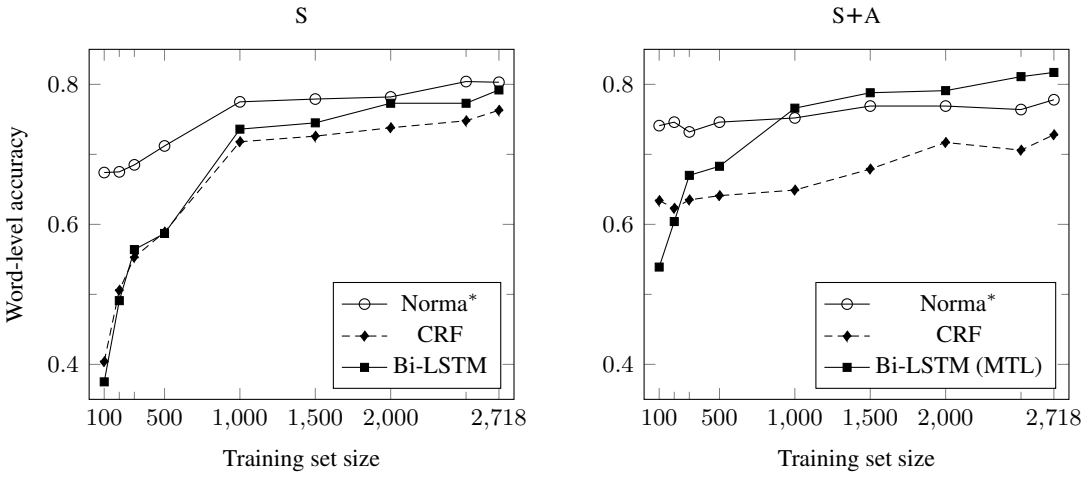


Figure 2: Word accuracy on the 'B' text for different sizes of the training set; left = train only on the training set from 'B' (S); right = use augmented training set/multi-task learning (S+A).

CRF cannot handle this out-of-domain training data well, the MTL setup can actually profit from it in many cases.

Table 1 also shows a rough classification of the dialectal regions from which the texts originate. There is a slight trend for multi-task learning to be advantageous on texts from the East and North Upper German regions, while for the Central and West Upper German texts, there are more instances of the standard bi-LSTM model (S) being better than the MTL model (S+A). This could either be due to linguistic properties of these dialectal regions, or due to the fact that East/North Upper German texts make up the majority of the dataset, thereby also featuring more prominently in the "S+A" settings.

The latter hypothesis is supported by the case of the 'Le' text, which is the only Dutch text in the sample (but which was nonetheless normalized to modern German in the corpus). Here, the "S+A" settings of the experiments all show a dramatic decrease in accuracy (up to -9.2 pp), suggesting that it is disadvantageous to augment the training set with samples that are too different from the target domain, even for the MTL setup.

In general, however, one of the bi-LSTM models is always best; there is only one text (We) for which Norma achieves an equal accuracy. This indicates that deep neural networks can be applied successfully to the spelling normalization task even with a comparatively small amount of training data. Also, we note that Norma always requires a lexical resource which it uses to filter results, while we do not.

4.2 Effect of training set size

In our evaluation, we use all but the first 2,000 tokens from a text for training (cf. the beginning of Sec. 4). Consequently, the training sets for each text are of different sizes. We calculate Spearman's rank correlation coefficient (ρ) between the size of the training sets and the normalization accuracy for each column in Table 1. We find no significant correlation for the CRF and bi-LSTM models ($|\rho| < 0.25$), although there seems to be a moderate *inverse* correlation for the Norma results ($\rho \approx -0.48$ on Norma "S"). The reasons for this are beyond the scope of this paper, though.

The question of how much training data is needed to effectively train a model is particularly relevant for historical spelling normalization, since training data can be very sparse in this domain. We therefore choose to evaluate each method in a scenario where we consider a single text, but vary the size of the training set, to estimate how well they perform with fewer data.

Figure 2 shows the results for different training set sizes on the 'B' text. Not surprisingly, when training on only 100 tokens, accuracy is bad ($< 41\%$) for CRF and bi-LSTM. Norma, on the other hand, already achieves 67.4% in this scenario. The biggest gains for all three methods can be seen for training set sizes between 100 and 1,000 tokens—for larger set sizes, the gains become less, and all three methods

are within close range of each other.

For the "S+A" scenario, all models have noticeably higher accuracy even with only 100 tokens from the 'B' text. However, the increases for Norma and CRF are not as high as in the "S" scenario; this is not surprising, since the total training set for these methods always contains at least 10,000 tokens (from the auxiliary set), and it is only the proportion of tokens coming from the 'B' text that increases. The bi-LSTM model with multi-task learning behaves differently, though: while it starts off as the weakest model (on 100 tokens), it is the best model when training on 1,000 tokens or more.

These learning curves illustrate that the MTL setup is fundamentally different from adding the auxiliary data to the training set normally, as is the case with CRF and the Norma tool. They also show that our bi-LSTM models can be better than or at least competitive with CRF/Norma for training set sizes as low as 1,000 tokens.

4.3 Multi-task learning with grapheme-to-phoneme mappings

It is conceivable to use different tasks than historical spelling normalization as the auxiliary task in a multi-task learning setup. In particular, we also experimented with grapheme-to-phoneme mapping as the auxiliary task, since it can be seen as a similar form of character-based sequence transduction.

For our dataset, we used the German part of the CELEX lexical database (Baayen et al., 1995), particularly the database of phonetic transcriptions of German wordforms. The database contains a total of 365,530 wordforms with transcriptions in DISC format, which assigns one character to each distinct phonological segment (including affricates and diphthongs). For example, the word *Jungfrau* 'virgin' is represented as `'jUN-frB`. We randomly sampled 4,000 tokens from this dataset for our experiment, and used the same algorithm as for the historical data to convert these mappings to a character-based sequence representation (cf. Sec. 2.1).

The evaluation, however, showed no real benefit of this MTL setup compared to the bi-LSTM model without MTL. While accuracy increased for some texts by up to 2.6 pp, it decreased slightly for the majority of texts, averaging to a -0.4 pp difference to the basic model.

5 Related Work

Various methods have been proposed to perform spelling normalization on historical texts; for an overview, see Piotrowski (2012). Many approaches use edit distance calculations or some form of character-level rewrite rules, but require either hand-crafting of the rules (Baron and Rayson, 2008) or a lexical resource to filter their output (Bollmann, 2012; Porta et al., 2013).

A newer approach is the application of character-based statistical machine translation (Pettersson et al., 2013; Sánchez-Martínez et al., 2013; Scherrer and Erjavec, 2013). In contrast to our sequence labeling approach, these methods do not require a fixed character alignment between wordforms, but it is not clear whether this is actually an advantage. To our knowledge, a comparative evaluation between these methods and other approaches has not yet been done.

Azawi et al. (2013) present the only other approach we are aware of that applies neural networks to normalization of historical data. They also use bi-directional LSTMs, but differ from our approach in the way they perform alignment between historical and modern wordforms. More importantly, they evaluate their model on a single dataset, the Luther bible, which has much more regular spelling than the texts in the Anselm corpus and is also significantly longer: they use about 200,000 tokens for their training set.

6 Conclusion and Future Work

We presented an approach to historical spelling normalization using bi-directional long short-term memory networks and showed that it outperforms a CRF baseline and the Norma tool by Bollmann (2012) for almost all of the texts in our dataset, a diverse corpus of Early New High German, despite using a relatively low amount of training data (about 2,000 to 11,000 tokens) and not making use of a lexical resource (like Norma does). We showed further that multi-task learning with additional normalization data can improve accuracy with bi-LSTMs, while adding the same data to the training set of Norma and CRF does not help on average, and can even be detrimental.

Many improvements to this approach are conceivable. Character-based statistical machine translation has been successfully applied to spelling normalization (cf. Sec. 5), but we are not aware of any experiments with neural machine translation (Cho et al., 2014) on this domain. Using an encoder–decoder architecture, e.g. similar to Sutskever et al. (2014), would remove the need for an explicit character alignment (cf. Sec. 2.1) but could also make the model more complex and potentially more difficult to train, so it is unclear whether this would be an improvement to our approach.

With regard to multi-task learning, our results seem to indicate that for the auxiliary task, it is preferable to use data with similar characteristics to the data in the main task. On the other hand, depending on the language variety to be annotated, such data might not always be readily available. We would therefore like to do further experiments with auxiliary data from different corpora or even different string transduction tasks, to see if and under which conditions they can have a beneficial effect on the spelling normalization task.

Acknowledgments

Marcel Bollmann was supported by Deutsche Forschungsgemeinschaft (DFG), Grant DI 1558/4.

References

Marilisa Amoia and Jose Manuel Martinez. 2013. Using comparable collections of historical texts for building a diachronic dictionary for spelling normalization. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 84–89, Sofia, Bulgaria.

Mayce Al Azawi, Muhammad Zeshan Afzal, and Thomas M. Breuel. 2013. Normalizing historical orthography for OCR historical documents using LSTM. In *Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing*, pages 80–85. ACM.

R. Harald Baayen, Richard Piepenbrock, and Léon Gulikers. 1995. The CELEX lexical database (Release 2) (CD-ROM). Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA.

Alistair Baron and Paul Rayson. 2008. VARD 2: A tool for dealing with spelling variation in historical corpora. In *Proceedings of the Postgraduate Conference in Corpus Linguistics*.

Marcel Bollmann. 2012. (Semi-)automatic normalization of historical texts using distance measures and the Norma tool. In *Proceedings of the Second Workshop on Annotation of Corpora for Research in the Humanities (ACRH-2)*, Lisbon, Portugal.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the 10th International Conference on Machine Learning (ICML)*, pages 41–48.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, pages 103–111, Doha, Qatar.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Stefanie Dipper and Simone Schultz-Balluff. 2013. The Anselm corpus: Methods and perspectives of a parallel aligned corpus. In *Proceedings of the NODALIDA Workshop on Computational Historical Linguistics*.

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

Bryan Jurish. 2010. More than words: using token context to improve canonicalization of historical German. *Journal for Language Technology and Computational Linguistics*, 25(1):23–39.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *Proceedings of NAACL-HLT 2016*, pages 1528–1533, San Diego, CA.

Julia Krasselt, Marcel Bollmann, Stefanie Dipper, and Florian Petran. 2015. Guidelines for normalizing historical German texts. *Bochumer Linguistische Arbeitsberichte*, 15.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. arXiv:1511.06114v4.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of conditional random fields (CRFs). http://www.chokkan.org/software/crfsuite/.

Eva Pettersson, Beáta Megyesi, and Jörg Tiedemann. 2013. An SMT approach to automatic annotation of historical text. In *Proceedings of the NODALIDA Workshop on Computational Historical Linguistics*, Oslo, Norway.

Michael Piotrowski. 2012. *Natural Language Processing for Historical Texts*. Number 17 in Synthesis Lectures on Human Language Technologies. Morgan & Claypool, San Rafael, CA.

Jordi Porta, José-Luis Sancho, and Javier Gómez. 2013. Edit transducers for spelling variation in Old Spanish. In *Proceedings of the NODALIDA Workshop on Computational Historical Linguistics*, Oslo, Norway.

Yves Scherrer and Tomaž Erjavec. 2013. Modernizing historical Slovene words with character-based SMT. In *Proceedings of the 4th Biennial Workshop on Balto-Slavic Natural Language Processing*, Sofia, Bulgaria.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS 2014)*, number 27, pages 3104–3112.

Felipe Sánchez-Martínez, Isabel Martínez-Sempere, Xavier Ivars-Ribes, and Rafael C. Carrasco. 2013. An open diachronic corpus of historical Spanish: annotation criteria and automatic modernisation of spelling. arXiv:1306.3692v1, 06.

Martijn Wieling, Jelena Prokić, and John Nerbonne. 2009. Evaluating the pairwise string alignment of pronunciations. In *Proceedings of the EACL 2009 Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education (LaTeCH – SHELT&R 2009)*, pages 26–34, Athens, Greece.

Deceptive Opinion Spam Detection Using Neural Network

Yafeng Ren and Yue Zhang
Singapore University of Technology and Design, Singapore
renyafeng@whu.edu.cn, yue_zhang@sutd.edu.sg

Abstract

Deceptive opinion spam detection has attracted significant attention from both business and research communities. Existing approaches are based on manual discrete features, which can capture linguistic and psychological cues. However, such features fail to encode the semantic meaning of a document from the discourse perspective, which limits the performance. In this paper, we empirically explore a neural network model to learn document-level representation for detecting deceptive opinion spam. In particular, given a document, the model learns sentence representations with a convolutional neural network, which are combined using a gated recurrent neural network with attention mechanism to model discourse information and yield a document vector. Finally, the document representation is used directly as features to identify deceptive opinion spam. Experimental results on three domains (*Hotel*, *Restaurant*, and *Doctor*) show that our proposed method outperforms state-of-the-art methods.

1 Introduction

Online reviews on products and services are extensively used by consumers and businesses for conducting decisive purchase, making product design and altering marketing strategies. As a result, deceptive opinion spam (e.g. deceptive reviews) arouses increasing attention (Streitfeld, 2012). Opinion spam is a type of review with fictitious opinions, deliberately written to sound authentic (Jindal and Liu, 2008; Ott et al., 2011). It can be difficult for human readers to distinguish them from truthful reviews. In a test by Ott et al. (2011), the average accuracy of three human judges is only 57.33%. It can be expensive to detect opinion spam manually over large user-generated texts. Hence, machine learning methods for automatically detecting deceptive opinion spam can be useful.

The objective of the task is to identify whether a given document is a spam or not. The majority of existing approaches follow the seminal work of Jindal and Liu (2008), employing classifiers with supervised learning. Most studies focus on designing effective features to enhance the classification performance. Typical features represent linguistic and psychological cues, but fail to effectively represent a document from the viewpoint of global discourse structures. For example, Ott et al. (2011) and Li et al. (2014) represent documents with Unigram, POS and LIWC (Linguistic Inquiry and Word Count) (Newman et al., 2003) features. Although such features give the strong performance, their sparsity makes it difficult to capture non-local semantic information over a sentence or discourse.

Recently, neural network models have been used to learn semantic representations for NLP tasks (Le and Mikolov, 2014; Tang et al., 2015), achieving highly competitive results. Potential advantages of using neural networks for spam detection are three-fold. First, neural models use dense hidden layers for automatic feature combinations, which can capture complex global semantic information that is difficult to express using traditional discrete manual features. This can be useful in addressing the limitation of discrete models mentioned above. Second, neural networks take distributed word embeddings as inputs, which can be trained from a large-scale raw text, thus alleviating the sparsity of annotated data to some

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 140–150, Osaka, Japan, December 11-17 2016.

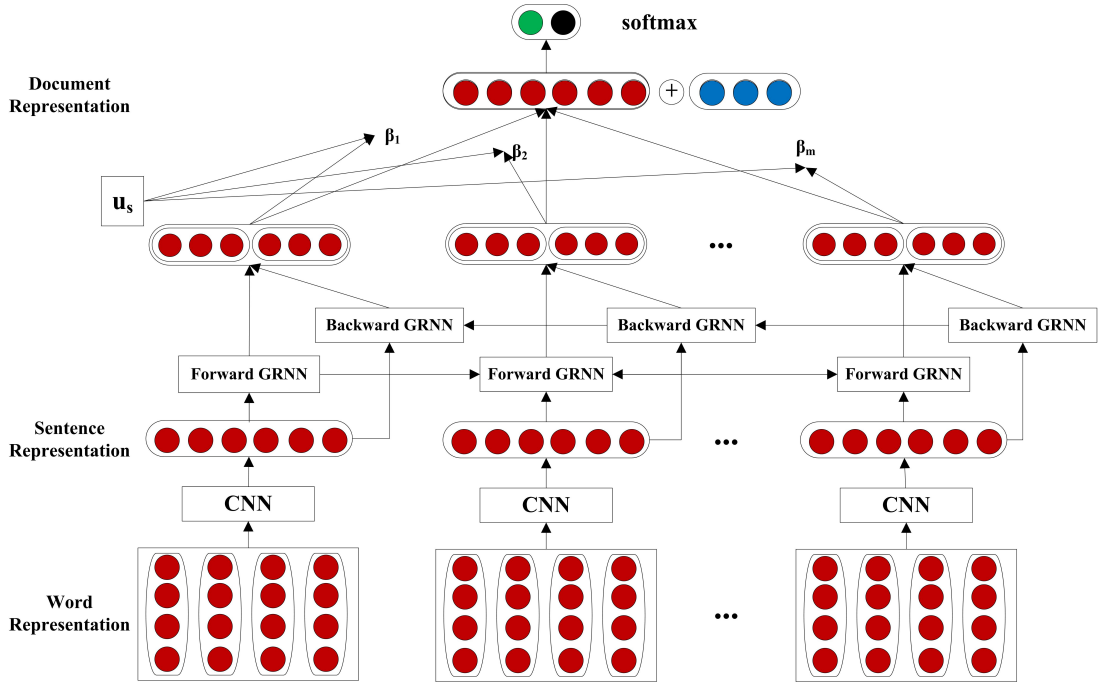


Figure 1: Neural network model structure for deceptive opinion spam detection, red nodes represent neural features, and blue nodes represent discrete features.

extent. Third, neural network models can be used to induce document representations from sentence representations, leveraging sentence and discourse information.

In this paper, we empirically investigate the effectiveness of learning dense document representations for opinion spam detection. In particular, we propose a three-stage neural network system, as shown in Figure 1. In the first stage, a convolutional neural network is used to produce sentence representations from word representations. Second, a bi-directional gated recurrent neural network with attention mechanism is used to construct a document representation from the sentence vectors. Finally, the document representation is used as features to identify deceptive opinion spam. Such automatically induced dense document representation is compared with traditional manually-designed features for the task.

We compare the proposed models on a standard benchmark (Li et al., 2014), which consists of data from three domains (*Hotel*, *Restaurant*, and *Doctor*). Results on in-domain and cross-domain experiments show that the dense neural features significantly outperforms the previous state-of-the-art methods, demonstrating the advantage of neural models in capturing semantic characteristics. In addition, automatic neural features and manual discrete features are complementary sources of information, and a combination leads to further improvements.

2 Related Work

2.1 Deceptive Opinion Spam Detection

Spam detection has been extensively investigated in the Web-page and E-mail domains (Gyöngyi et al., 2004; Ntoulas et al., 2006), while research has recently been extended to the customer review domain (Ott et al., 2011; Mukherjee et al., 2013; Li et al., 2014). Various types of indicator features have been investigated. For examples, Jindal and Liu (2008) trained models using features based on the review content, the reviewer, and the product itself. Yoo and Gretzel (2009) gathered 40 truthful and 42 deceptive hotel reviews and manually compared the linguistic differences between them.

Ott et al. (2011) created a benchmark dataset by employing *Turkers* to write fake reviews. Their data were adopted by a line of subsequent work (Ott et al., 2012; Feng et al., 2012; Feng and Hirst, 2013). For example, Feng et al. (2012) looked into syntactic features from Context Free Grammar (CFG) parse trees to improve the performance. Feng and Hirst (2013) built profiles of hotels from collections of

reviews, measuring the compatibility of customer reviews to the hotel profile, and using it as a feature for opinion spam detection. Recently, Li et al. (2014) created a wider-coverage benchmark, which comprises of data from three domains (*Hotel*, *Restaurant*, and *Doctor*), and explored generalized approaches for identifying online deceptive opinion spam. We adopt this dataset for our experiments due to its larger size and coverage.

Existing methods use traditional discrete features, which can be sparse and fail to effectively encode the semantic information from the overall discourse. In this paper, we propose to learn document-level neural representation for better detecting deceptive opinion spam. To our knowledge, we are the first to investigate deep learning for deceptive opinion spam detection.

There has been work that exploits features outside the review content itself. In addition to Jindal and Liu (2008), Mukherjee et al. (2013) explored the features from customer's behavior to identify deception. Based on some truthful reviews and a lot of unlabeled reviews, Ren et al. (2014) proposed a semi-supervised learning method, and built an accurate classifier to identify deceptive reviews. Kim et al. (2015) introduced a frame-based semantic feature based on FrameNet. Experimental results show that semantic frame features can improve the classification accuracy. We focus on the review content in this paper, but their features can be used to extend our model.

2.2 Neural Network Models for Representation Learning

Neural network models have been exploited to learn dense feature representation for a variety of NLP tasks (Collobert et al., 2011; Kalchbrenner et al., 2014; Ren et al., 2016b). Distributed word representations (Mikolov et al., 2013) have been used as the basic building block by most models for NLP. Numerous methods have been proposed to learn representations of phrases and larger text segments from distributed word representations. For example, Le and Mikolov (2014) introduced paragraph vector to learn document representations, extending to word embedding methods of Mikolov et al. (2013). Socher et al. (2013) introduced a family of recursive neural networks to represent sentence-level semantic composition. Follow-up research includes recursive neural network with global feed backward mechanisms (Paulus et al., 2014), deep recursive layers (Irsoy and Cardie, 2014), and adaptive composition functions (Dong et al., 2014).

Convolutional neural networks have been widely used for semantic composition (Kalchbrenner et al., 2014; Johnson and Zhang, 2014), automatically capturing n-gram information. Sequential models such as recurrent neural network or long short-term memory (LSTM) (Li et al., 2015a; Tang et al., 2015) have also been used for recurrent semantic composition. The attention mechanism was first proposed in machine translation (Bahdanau et al., 2014). Further uses of the attention mechanism include parsing (Vinyals et al., 2014), natural language question answering (Sukhbaatar et al., 2015; Kumar et al., 2015; Hermann et al., 2015), and image question answering (Yang et al., 2015). We explore CNN and recurrent neural networks with attention mechanism to learn document representation for detecting deceptive opinion spam, comparing their effect with bag-of-word and paragraph vector baselines.

3 Approach

The proposed neural network model learns real-valued dense vector representations for documents of variable lengths, which is used as the feature to classify each document. Shown in Figure 1, it consists of two main components, The first produces distributed vector sentence representations from word representations, and the second gives dense vector document representations from the sentence vectors.

Structurally, the composition of words in forming sentences is similar to the composition of sentences in forming documents, both tracking sequences of inputs with long range dependencies. Both CNN and RNN are typically used for representing sequences in NLP, giving state-of-the-art accuracies in various tasks. For example, for modeling sentences, CNN gives the best results for sentiment analysis (Johnson and Zhang, 2014; Ren et al., 2016a), while LSTM gives the best results for question answering (Wang and Nyberg, 2015). For modeling discourse structures, LSTM has been used more frequently (Li et al., 2015b; Tang et al., 2015). We experimented with both CNN and RNN for both sentence and document modeling, finding that the best development accuracies are obtained when CNN is used for sentence

modeling and RNN is used for document modeling. Therefore, we choose this structure in Figure 1. Note, however, that our main goal is to empirically study the effectiveness of neural features in contrast to manual discrete features, rather than find a most accurate neural model variation for this task.

3.1 Sentence Model

We represent words using embeddings (Bengio et al., 2003), which are low-dimensional dense real-valued vectors. For each word w, we use a look-up matrix E to obtain its embedding $e(w) \in R^D$, where $E \in R^{D \times V}$ is a model parameter, D is the word vector dimension size and V is the vocabulary size. E can be randomly initialized from a uniform distribution (Socher et al., 2013), or pre-trained from a large raw corpus (Mikolov et al., 2013).

As shown in the bottom of Figure 1, a convolutional neural network (CNN) (Kim, 2014; Kalchbrenner et al., 2014; Johnson and Zhang, 2014) is used to learn dense representations of a sentence. We use three convolutional filters to capture the local semantics of n-grams of various granularities. Formally, denote a sentence consisting of n words as $\{w_1, w_2, .., w_i, ..w_n\}$. Each word w_i is mapped to the embedding representation $e(w_i) \in R^D$. A convolutional filter is a list of linear layers with shared parameters. Let D_1, D_2, D_3 be the width of the three convolutional filters, respectively. We set $D_1 = 1$, $D_2 = 2$ and $D_3 = 3$ for representing unigrams, bigrams and trigrams, respectively. Taking D_2 for example, W_2 and b_2 are the shared parameters of linear layers for this filter. The input of a linear layer is the concatenation of word embeddings in a fixed-length window size D_2, which is denoted as $I_{2,i} = [e(w_i); e(w_{i+1}); ...; e(w_{i+D_2-1})] \in R^{D \times D_2}$. The output of a linear layer is calculated as

$$H_{2,i} = W_2 \cdot I_{2,i} + b_2, \tag{1}$$

where $W_2 \in R^{l_{oc} \times D \times D_2}$, l_{oc} is the output size of the linear layer. We use an average pooling layer to merge the varying number of outputs $\{H_{2,1}, H_{2,2}, .., H_{2,n}\}$ from the convolution layer into a vector with fixed dimensions.

$$H_2 = \frac{1}{n} \sum_{i=1}^{n} H_{2,i} \tag{2}$$

To incorporate nonlinearity, a activation function $tanh$ is used to obtain the output O_2 of this filter.

$$O_2 = tanh(H_2) \tag{3}$$

Similarly, we obtain the O_1 and O_3 for the other two convolutional filters with width 1 and 3, respectively. The outputs of three filters are lastly averaged to generate sentence representation.

3.2 Document Model

Given a document with m sentences, we use the sentence vectors $s_1, s_2, .., s_m$ obtained by the CNN model as inputs, and learn document composition with a gated recurrent neural network (GRNN). Standard recurrent neural networks (RNN) map sentence vectors of variable lengths to a fixed-length vector, by starting with an initial vector, and recurrently transforming the current sentence vector s_t together with the previous state vector h_{t-1} into a new state vector h_t. The transition function is typically a linear layer followed by a non-linear activation function such as *tanh*

$$h_t = tanh(W_r \cdot [h_{t-1}; s_t] + b_r), \tag{4}$$

where $W_r \in R^{l_h \times (l_h + l_{oc})}$, $b_r \in R^{l_h}$, l_h and l_{oc} are dimensions of state vectors and sentence vectors, respectively. Unfortunately, the standard RNN suffers the problem of vanishing gradients (Bengio et al., 1994; Hochreiter and Schmidhuber, 1997). This makes it difficult to model long-distance correlation in a sequence. We explore a gated recurrent neural network (GRNN) to address this, which is similar in spirit to LSTM (Cho et al., 2014; Chung et al., 2015), but empirically runs faster. Specifically, the transition function of the GRNN used in the work is calculated as follows

$$i_t = sigmoid(W_i \cdot [h_{t-1}; s_t] + b_i) \tag{5}$$

$$f_t = sigmoid(W_f \cdot [h_{t-1}; s_t] + b_f) \tag{6}$$

$$g_t = tanh(W_r \cdot [h_{t-1}; s_t] + b_r) \tag{7}$$

$$h_t = tanh(i_t \odot g_t + f_t \odot h_{t-1}) \tag{8}$$

where $\odot$ stands for element-wise multiplication, i_t and f_t represent the reset gate and update gate, respectively. W_i, W_f, b_i, b_f adaptively select and remove history state vectors and input vectors for semantic composition.

To better capture discourse relations, we apply the GRNN structure over sentence representation vectors in the left-to-right and right-to-left directions, respectively, resulting in a forward state sequence $h_1, h_2, .., h_n$ and a backward state sequence $h_n^{'}, h_{n-1}^{'}, .., h_1^{'}$, respectively. For each sentence vector node s_i, a combination of h_i and $h_i^{'}$ is used as its bi-directional state vector. Here, if all bi-directional state vectors are treated equally, the noisy or irrelevant part may degrade the classification performance. Meanwhile, Vrij et al. (2009) and Ott et al. (2011) find that different topics have different importance in deceptive opinion detection. For example, spatial information can usually be a strong indicator of non-spam for hotel reviews. So we introduce a simple attention mechanism to consider the importance of different state vectors. Specifically, for each sentence s_i in one document d, which contains the sentences vectors $s_1, s_2, .., s_m$, we integrate the weights into bi-directional state vector h_i and $h_i^{'}$. Specifically, we use the context vector to measure the importance of the sentences. This yields

$$u_i = tanh(W_s(h_i \oplus h_i^{'}) + b_s), \tag{9}$$

$$\beta_i = \frac{exp({u_i}^{\mathrm{T}} u_s)}{\sum_i exp({u_i}^{\mathrm{T}} u_s)} \tag{10}$$

The document vector d is represented as

$$d = \sum_i \beta_i (h_i \oplus h_i^{'}), \tag{11}$$

where $\sum_{i=1}^{m} \beta_i = 1$, and $\oplus$ is the vector concatenation function. The context vector u_s has been used in previous memory networks (Kumar et al., 2015; Sukhbaatar et al., 2015), and it can been randomly initialized and jointly learned during the training process.

3.3 The Classification Model

We use the document representation as features for identifying deceptive opinion spam. Specifically, a linear layer is added to transform the document vector into a real-valued vector, whose length is class number *C*. A *softmax* function is added to convert real vector to conditional probability for document classification.

Our training objective is to minimize the cross-entropy loss over a set of training examples $(x_i, y_i)|_{i=1}^{N}$, plus a l_2-regularization term,

$$L(\theta) = -\sum_{i=1}^{N} \log \frac{e^{\vec{o}(y_i)}}{e^{\vec{o}(0)} + e^{\vec{o}(1)}} + \frac{\lambda}{2} \parallel \theta \parallel^2, \tag{12}$$

where θ is the set of model parameters.

We use online AdaGrad to minimize the training objective. At step t, the parameters are updated by:

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\alpha}{\sqrt{\sum_{t'=1}^{t} g_{t',i}^2}} g_{t,i}, \tag{13}$$

where α is the initial learning rate, and $g_{t,i}$ is the gradient of the *i*th dimension at step *t*.

We initialize all the matrix and vector parameters with uniform samples in $(-\sqrt{6/(r+c)}, -\sqrt{6/(r+c)})$, where r and c are the numbers of rows and columns of the matrixes, respectively. We learn word embeddings of 100-dimensions using the CBOW model of Mikolov

Domain	Turker	Employee	Customer
Hotel	800	280	800
Restaurant	200	120	400
Doctor	200	32	200

Table 1: Statistics dataset.

Method	Accuracy (%)	Macro-F1 (%)
Average	73.0	73.9
CNN	75.9	77.4
RNN	63.2	64.8
GRNN	80.1	80.7
Bi-directional GRNN	83.6	83.4
Bi-directional GRNN (Attention)	**84.1**	**83.9**
Le and Mikolov (2014)	76.1	77.6

Table 2: Development results.

et al. (2013) from a large-scale Amazon reviews corpus [1]. During training, we use the average of all the pre-trained embeddings vectors to initialize unknown words. We set the output length of the convolutional filter as 50. The initial learning rate of Adagrad is set as 0.01.

4 Experiments

4.1 Experimental Setup

We use the dataset of Li et al. (2014), which consists of truthful and deceptive reviews in three domains, namely *Hotel*, *Restaurant* and *Doctor*. For each domain, a set of *Customer* reviews are collected as truthful reviews, and a set of deceptive reviews are collected from *Turkers* and *Employees*, respectively. We follow Li et al. (2014) in designing the evaluation metrics. For the *Hotel* domain, we perform both three-way (*Customer*/*Employee*/*Turker*) and two-way classification between *Customer* reviews and *Employee*/*Turker* reviews. This is because deceptive reviews from *Employee* and *Turker* can reflect different levels of domain knowledge. For the *Restaurant* and *Doctor* domains, we perform only two-way *Customer*/*Turker* classification because *Employee* reviews are relatively too few. Table 1 shows the statistics of the dataset. For each experiment, we measure both the per-instance accuracy and the macro-F1 score across different classes.

4.2 Development Experiments

To compare the effectiveness of various neural document models, we conduct a set of development experiments using the mixed dataset of all three domains. Only *Turker* and *Customer* reviews are used, and the total of 2600 reviews are split randomly into training/tuning/testing sets with a ratio of 80/10/10. The tuning set is used for optimizing the hyper-parameters for each neural network structure.

We compare a set of methods for document modeling, which include a single averaging method, tracking a document as a bag of sentences (Average), a CNN, a naive RNN, and our gated RNN in single- and bi-directional method. In addition, we compare the bi-directional RNN without attention and with attention being used.

Table 2 show the results. Without modeling discourse relations, the averaging method gives a baseline accuracy of 73.0%. CNN gives better results by capturing relationships between local sentences. Though modeling global sequential relations, RNN does not give better results compared with the averaging baseline, and the main reason is vanishing gradients in its training. By using gates, the results of GRNN is significantly better than both the baseline and the CNN document model. Both averaging and the bi-directional extension further increased the accuracies. By introducing the attention mechanism into the bi-directional GRNN, the best development result is 84.1%.

We also compare our methods with the paragraph vector model of Le and Mikolov (2014), which builds a document representation without considering sentence vectors. It gives results comparable to

[1] http://snap.stanford.edu/data/web-Amazon.html

Domain	Setting	Method	Accuracy (%)	Macro-F1 (%)
Hotel	Customer/Employee/Turker	Li et al.	66.4	67.3
		Neural/Logistic	**78.9**/66.5	**74.7**/67.6
		Integrated	**81.3**	**77.4**
	Customer/Turker	Li et al.	81.8	82.6
		Neural/Logistic	**84.1**/82.4	**84.2**/83.5
		Integrated	**86.1**	**86.0**
	Customer/Employee	Li et al.	79.9	80.9
		Neural/Logistic	**84.8**/79.4	**82.4**/80.6
		Integrated	**87.2**	**84.7**
	Employee/Turker	Li et al.	76.2	78.0
		Neural/Logistic	**91.1**/76.2	**87.9**/78.5
		Integrated	**92.8**	**90.4**
Restaurant	Customer/Turker	Li et al.	81.7	82.2
		Neural/Logistic	**84.8**/82.5	**85.0**/82.7
		Integrated	**87.1**	**87.0**
Doctor	Customer/Turker	Li et al.	74.5	73.5
		Neural/Logistic	75.3/74.4	73.4/72.9
		Integrated	**76.3**	**74.5**

Table 3: In-domain results.

the CNN model, but much lower compared with the GRNN models, which leverage non-local discourse structures.

4.3 In-Domain Results

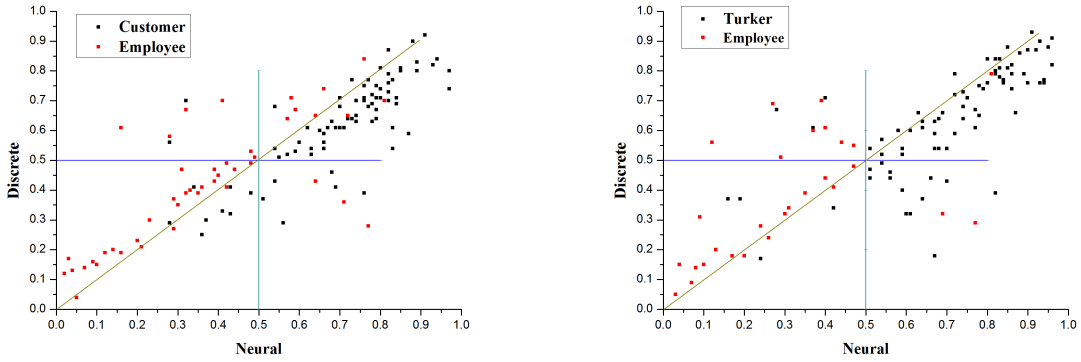
We choose the best neural model, namely the bi-directional GRNN (Attention), according to the development test results. A set of in-domain test are conducted according to Li et al. (2014)'s settings, in order to compare the neural model with the state-of-the-art discrete model with SVM. In particular, all results are reported by using ten-fold cross-validation. As mentioned in the introduction, Li et al. (2014) use hand-crafted features that contain the word, POS and other linguistic clues.

The results are shown in Table 3, in the Li et al. rows and the left items of the Neural/Logistic rows, respectively. For the *Hotel* domain, the neural model outperforms the discrete model of Li et al. (2014) on both three-way *Customer*/*Employee*/*Turker* classification and two-way classification tasks. While Li et al. (2014)'s method gives about around 80% accuracies on *Customer*/*Turker* and *Customer*/*Employee* classifications, which distinguish truthful and deceptive reviews. The accuracies drop to below 66.4% when all the three classes are involved. In contrast, our method gives an accuracy of 78.9% for the three-way task, demonstrating the power of the neural model in distinguishing deceptive reviews from different types of authors. The contrast on the two-way *Employee*/*Turker* classification task is consistent. This shows the power of the neural model in capturing subtle semantic features, which are difficult to express using manual indicator features.

The results on the *Restaurant* domain is similar to those on the *Hotel* domain, where the neural model significantly outperforms the discrete model. However, the neural model gives similar results compared with the discrete model on the *Doctor* domain. One possible reason is that number of reviews in this dataset is relatively lower, which leads to relatively lower accuracies by both models. The other reason is a relatively high OOV rate, and 7.02% of the test words in the *Doctor* domain are out of the embedding dictionary (in contrast to 3.25% in the *Hotel* domain and 3.43% in the *Restaurant* domain).

4.3.1 Analysis

In order to contrast the effect on discrete and neural features, we build a discrete model using logistic regression with the same discrete feature as Li et al. (2014). The main advantage of using this model is a direct comparison on features, because a logistic regression classifier is the same as the *softmax* output layer of our neural network model in mathematic form. The only difference is that the logistic regression method uses discrete features, while the neural model uses continuous features from the deep neural network. The results of the logistic regression model are shown in the right items of the Neural/Logistic rows in Table 3, which are slightly lower but comparable to Li et al. (2014)'s SVM results.



(a) Hotel (Customer/Employee)

(b) Hotel (Turker/Employee)

Figure 2: Output probability comparisons.

Figure 2 shows the output probabilities of the *Customer* and *Turker* classes by both the neural and the logistic discrete models, respectively. Results on the *Hotel* (*Customer*/*Employee*) and *Hotel* (*Turker*/*Employee*) datasets are shown in Figure 2(a) and 2(b), respectively. The x-axis shows the probability by the neural model and the y-axis shows the probability by the discrete model. Taking Figure 2(a) for example, true *Customer* reviews in the test set are shown in black, where false reviews by *Employee* are shown in red. As a result, black dots on the top of the figure and red dots on the bottom show cases which the discrete model predicted correctly, while black dots on the right and red dots on the left show cases which the neural model predicted correctly.

As shown in the figure, most black dots are on the top-right of the figure and most red dots are on the bottom-left, showing that both models are correct in most cases. However, the dots are relatively more disperse in the x-axis, showing that the neural model is more confident in scoring the inputs. This demonstrates the effectiveness of neural features. Observation in Figure 2(b) is similar. For the more challenging task, the neural model shows large advantages.

Figure 2 also shows that the errors by using neural and discrete features can be complementary, which suggests that integrating both types of features in a single model can further improve the results. We make a feature integration by directly concatenating the discrete feature vector (the blue nodes in Figure 1) to the neural features vector before the *softmax* layer. The results of the combined model are shown in the Integrated rows in Table 3. In all the test sets, the model gives significantly better results compared with both the neural and logistic models[2].

4.4 Cross-Domain Results

For the task of deceptive opinion spam detection, the sample numbers of the dataset are relatively small, and the collection of labeled data is time-consuming and expensive. We investigate two important questions. First, it is interesting to know whether the relatively more richly annotated *Hotel* domain dataset can be used to train effective deception detection models on the *Restaurant* or *Doctor* domain. Second, we study the generalization ability of our neural model. We frame the problems as a domain adaptation task, training a classifier on *Hotel* reviews, and evaluate the performance on the other domains. For simplicity, we focus on two-way *Customer*/*Turker* classification.

The results are shown in Table 4. First, the classifiers trained on *Hotel* reviews apply well to the *Restaurant* domain, which is reasonable due to the many shared properties among *Restaurant* and *Hotel*, such as the environment and location. However, the performance on the *Doctor* domain is much worse, largely due to the difference in vocabulary. Second, compared with the method of Li et al. (2014), our neural model gives better performance. For the *Doctor* domain, both models trained on the *Hotel* domain do not generalize well. Our neural model gives a higher F1 (66.3%) compared with the SVM classifier

[2]The p-value is below 10^{-3} using t-test

Domain	Method	Accuracy (%)	Macro-F1 (%)
Restaurant	Li et al.	78.5	77.8
	Neural	81.9	81.0
	Integrated	**83.7**	**82.6**
Doctor	Li et al.	55.0	61.7
	Neural	56.1	66.3
	Integrated	**57.3**	**67.6**

Table 4: Cross-domain results.

(61.7%), which shows some relative effectiveness of neural model. Similar to the in-domain results, the integrated model outperforms both the discrete and neural models.

5 Conclusion

We investigated a gated recurrent neural network model with attention mechanism for deceptive opinion spam detection. By capturing non-local discourse information over sentence vectors, the neural network model outperforms a state-of-the-art discrete baseline, and also simple neural document models such as paragraph vectors. Further experiments show that the accuracies can be improved by integrating discrete and neural features.

Acknowledgements

We thank all reviewers for all their detailed comments. This work supported by Singapore Ministry of Eduction (MOE) AcRF Tier 2 grant T2MOE201301. We also thank Jiayuan Deng for some experimental work in the early stage of this paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. *arXiv preprint arXiv:1502.02367*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2014. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Vanessa Wei Feng and Graeme Hirst. 2013. Detecting deceptive opinions with profile compatibility. In *Proceedings of the International Joint Conference on Natural Language Processing*.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. 2004. Combating web spam with trustrank. In *Proceedings of the International Conference on Very Large Data Bases*.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *arXiv preprint arXiv:1506.03340*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the International Conference on Web Search and Web Data Mining*.

Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Seongsoon Kim, Hyeokyoon Chang, Seongwoon Lee, Minhwan Yu, and Jaewoo Kang. 2015. Deep semantic frame-based deceptive opinion spam analysis. In *Proceedings of the ACM International Conference on Information and Knowledge Management*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Jiwei Li, Myle Ott, Claire Cardie, and Eduard Hovy. 2014. Towards a general rule for identifying deceptive opinion spam. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Jiwei Li, Dan Jurafsky, and Eudard Hovy. 2015a. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015b. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*.

Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Spotting opinion spammers using behavioral footprints. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Matthew L Newman, James W Pennebaker, Diane S Berry, and Jane M Richards. 2003. Lying words: Predicting deception from linguistic styles. *Personality and Social Psychology Bulletin*.

Alexandros Ntoulas, Marc Najork, Mark Manasse, and Dennis Fetterly. 2006. Detecting spam web pages through content analysis. In *Proceedings of the International World Wide Web Conference*.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Myle Ott, Claire Cardie, and Jeff Hancock. 2012. Estimating the prevalence of deception in online review communities. In *Proceedings of the International World Wide Web Conference*.

Romain Paulus, Richard Socher, and Christopher D Manning. 2014. Global belief recursive neural networks. In *Advances in Neural Information Processing Systems*.

Yafeng Ren, Donghong Ji, and Hongbin Zhang. 2014. Positive unlabeled learning for deceptive reviews detection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016a. Context-sensitive twitter sentiment classification using neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016b. Improving twitter sentiment classification using topic-enriched multi-prototype word embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

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

David Streitfeld. 2012. For $2 a star, an online retailer gets 5-star product reviews. *New York Times*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. Weakly supervised memory networks. *arXiv preprint arXiv:1503.08895*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2014. Grammar as a foreign language. *arXiv preprint arXiv:1412.7449*.

Aldert Vrij, Sharon Leal, and et al. Granhag. 2009. Outsmarting the liars: the benefit of asking unanticipated questions. *Law and human behavior*.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alexander J. Smola. 2015. Stacked attention networks for image question answering. *arXiv preprint arXiv:1511.02274*.

Kyung-Hyan Yoo and Ulrike Gretzel. 2009. Comparison of deceptive and truthful travel reviews. *Information and communication technologies in tourism 2009*.

Integrating Topic Modeling with Word Embeddings by Mixtures of vMFs

Ximing Li[1,2] **Jinjin Chi**[1,2] **Changchun Li**[1,2] **Jihong Ouyang**[1,2] **Bo Fu**[3]

1. College of Computer Science and Technology, Jilin University, China
2. Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, China
3. College of Computer and Information Technology, Liaoning Normal University, China
liximing86@gmail.com

Abstract

Gaussian LDA integrates topic modeling with word embeddings by replacing discrete topic distribution over word types with multivariate Gaussian distribution on the embedding space. This can take semantic information of words into account. However, the Euclidean similarity used in Gaussian topics is not an optimal semantic measure for word embeddings. Acknowledgedly, the cosine similarity better describes the semantic relatedness between word embeddings. To employ the cosine measure and capture complex topic structure, we use von Mises-Fisher (vMF) mixture models to represent topics, and then develop a novel mix-vMF topic model (MvTM). Using public pre-trained word embeddings, we evaluate MvTM on three real-world data sets. Experimental results show that our model can discover more coherent topics than the state-of-the-art baseline models, and achieve competitive classification performance.

1 Introduction

Topic models such as latent Dirichlet allocation (LDA) (Blei et al., 2003) are hierarchical probabilistic models of document collections. They can effectively uncover the main themes of corpora by using latent topics learnt from observed collections (Blei, 2012), however, they neglect semantic information of words. In topic modeling, a "topic" is a multinomial distribution over a fixed vocabulary, i.e., a word type proportion. Because words are represented by unordered indexes, with statistical inference algorithms, related words are grouped into topics mainly by using document-level word co-occurrence information (Wang and McCallum, 2006), rather than semantics of words. That is why LDA often outputs many low-quality topics, and views in (Das et al., 2015) even suggest that any such observation of semantically coherent topics in topic models is, in some sense, accidental.

To mix with semantics of words, a recent Gaussian LDA (G-LDA) (Das et al., 2015) model integrates topic modeling with word embeddings, which can effectively capture lexico-semantic regularities in language from a large unlabeled corpus (Mikolov et al., 2013). This hot technique transforms words into vectors (i.e., word vector). To model documents of word vectors, G-LDA replaces the discrete topic distributions over word types with multivariate Gaussian distributions on the word embedding space. Because words with similar semantic properties are closer to each other in the embedding space, semantic information of words can be taken into consideration by using Gaussian distributions to describe semantic centrality location of topics.

An issue of G-LDA is that the word weights in Gaussian topics are measured by the Euclidean similarity between word embeddings. However, the Euclidean similarity is not an optimal semantic measure, since most of word embedding algorithms use exponentiated cosine similarity as the link function (Li et al., 2016a). The cosine similarity may be a better choice to describe the semantic relatedness between word embeddings. Following this idea, in this paper we use von Mises-Fisher (vMF) distributions on the embedding space to represent topics, replacing Gaussian topics in G-LDA. The vMF distribution defines a probability density over vectors on a unit sphere, parameterized by mean μ and concentration

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 151–160, Osaka, Japan, December 11-17 2016.

parameter κ. Its density function for $x \in \mathcal{R}^M$, $\|x\| = 1$, $\|\mu\| = 1$, $\kappa \geq 0$ is given by:

$$p\left(x|\mu,\kappa\right) = c_p\left(\kappa\right)\exp\left(\kappa\mu^T x\right) \tag{1}$$

where $c_p\left(\kappa\right)$ is the normalization constant. Note that vMF concerns the cosine similarity defined by $\mu^T x$. It is a better way to represent topics of word embeddings.

Another issue we face is that topics often contain many words that are far away from each other in the embedding space. That is, the true distributions of topics often form two or more dominant clumps. However, a simple vMF distribution is unable to capture such structure. For example, the topic $\langle software, user, net, feedback, grade\rangle$ contains some "dissimilar" words, such as *net* and *grade*[1]. In this case, a simple vMF topic distribution can not simultaneously place high probabilities on these "dissimilar" words.

To address the problem mentioned above, we further use mixtures of vMFs to describe topics, rather than a single vMF. We then develop a novel mix-vMF topic model (MvTM). Mixtures of vMFs can help us capture complex topic structure that forms more dominant clumps. In MvTM, we consider two settings with respect to the topic, i.e., disjoint setting and overlapping setting. Naturally, in disjoint settings all mixtures of vMFs use disjoint vMF bases; and in overlapping setting some mixtures of vMFs share the same vMF bases. An advantages of the overlapping setting is that it can describe topic correlation in some degree. We have conducted a number of experiments on three real-world data sets. Experimental results show that our MvTM can discover more coherent topics than the state-of-the-art baseline topic models, and achieve competitive performance on the classification task.

2 Model

In this section, we simply review LDA and G-LDA.

2.1 LDA

LDA (Blei et al., 2003) is a representative probabilistic topic model of document collections. In LDA, the main themes of corpora are described by topics, where each topic is a multinomial distribution ϕ over a fixed vocabulary (i.e., a word type proportion). Each document is a multinomial distribution θ over topics (i.e., a topic proportion). For simplification, distributions ϕ and θ are designed to be sampled from the conjugate Dirichlet priors parameterized by β and α, respectively. Suppose that *D*, *K* and *V* denote the number of documents, topics and word types. The generative process of LDA is as follows:

1. For each topic $k \in \{1, 2, \cdots, K\}$
   (a) Sample a topic $\phi_k \sim Dir\left(\beta\right)$
2. For each document $d \in \{1, 2, \cdots, D\}$
   (a) Sample a topic proportion: $\theta_d \sim Dir\left(\alpha\right)$
   (b) For each of the N_d words embeddings
      i. Sample a topic indicator $z_{dn} \sim Multinomial\left(\theta_d\right)$
      ii. Sample a word $w_{dn} \sim Multinomial\left(\phi_{z_{dn}}\right)$

Reviewing the definition above, we note that a topic in LDA is a discrete distribution over observable word types (i.e., word indexes). In this sense, LDA neglects semantic information of words and precludes new word types to be added into topics.

2.2 G-LDA

G-LDA (Das et al., 2015) integrates topic modeling with word embeddings. This model replaces the discrete topic distributions over word types with multivariate Gaussian distributions on an *M*-dimensional

[1]This means that the cosine similarity between word embeddings of *net* and *grade* is small.

embedding space, and concurrently replaces the Dirichlet priors with the conjugate Normal-Inverse-Wishart (NIW) priors on Gaussian topics. Because word embeddings learnt from large unlabeled corpora effectively capture semantic information of words (Bengio et al., 2003), G-LDA can handle, in some sense, words' semantics and new word types. Let $\mathcal{N}(\mu_k, \Sigma_k)$ be the Gaussian topic *k* with mean μ_k and covariance matrix Σ_k. The generative process of G-LDA is as follows:

1. For each topic $k \in \{1, 2, \cdots, K\}$
   (a) Sample a Gaussian topic $\mathcal{N}(\mu_k, \Sigma_k) \sim NIW(\mu_0, \kappa_0, \Psi_0, \nu_0)$
2. For each document $d \in \{1, 2, \cdots, D\}$
   (a) Sample a topic proportion: $\theta_d \sim Dir(\alpha)$
   (b) For each of the N_d word embeddings
      i. Sample a Gaussian topic indicator $z_{dn} \sim Multinomial(\theta_d)$
      ii. Sample a word embedding $w_{dn} \sim \mathcal{N}(\mu_{z_{dn}}, \Sigma_{z_{dn}})$

3 MvTM

G-LDA defines Gaussian topics, which measure word weights in topics by the Euclidean similarity between word embeddings. However, the Euclidean similarity is not an optimal semantic measure of word embeddings. People often prefer the cosine similarity (Li et al., 2016a). To upgrade G-LDA, a novel mix-vMF topic model (MvTM) is proposed, where we replace the Gaussian topic in G-LDA with mixture of vMFs. In this work, we use mixture of vMFs with *C* mixture components (Banerjee et al., 2005) described by:

$$p(x|\pi_{1:C}, \mu_{1:C}, \kappa) = \sum_{c=1}^{C} \pi_c p_c(x|\mu_c, \kappa) \tag{2}$$

where $p_c(x|\mu_c, \kappa)$ is the mixture vMF component (i.e., base); π_c is the mixture weight and such that $\sum_{c=1}^{C} \pi_c = 1$. The design of MvTM has two advantages. First, the vMF distribution defines a probability density over normalized vectors on a unit sphere. Reviewing Eq.1, it can be seen that vMF concerns the cosine similarity. Second, using linear vMF mixture model can help us capture complex topic structure, which forms two or more dominant clumps.

Formally, MvTM models documents consisting of normalized word embeddings w in an *M*-dimensional space, i.e., $\|w\| = 1$ and $w \in R^M$. Suppose that there are *K* topics in total. We characterize each topic *k* as a mixture of vMFs with parameter $\Delta_k = \left\{\pi_{k|_{1:C}}, \mu_{k|1:C}, \kappa_k\right\}$. Besides the topic design, again suppose that each document is a topic proportion θ, drawn from a Dirichlet prior α. Let *D* and N_d be the number of documents and the number of words in document *d*, respectively. The generative process of MvTM is given by:

1. For each document $d \in \{1, 2, \cdots, D\}$
   (a) Sample a topic proportion: $\theta_d \sim Dir(\alpha)$
   (b) For each of the N_d word embeddings
      i. Sample a vMF mixture topic indicator $z_{dn} \sim Multinomial(\theta_d)$
      ii. Sample a word vector $w_{dn} \sim v\mathcal{MF}(\Delta_{z_{dn}})$

In MvTM, the vMF bases of different topics can be either disjoint or overlapping. For disjoint MvTM (abbr. MvTM$_d$), the vMF bases of different topics are disjoint. In MvTM$_d$, the total number of vMF bases is $C \times K$. For overlapping MvTM (abbr. MvTM$_o$), vMF bases are allowed to be shared by different topics. An advantage is that the overlapping setting can describe topic correlation in some degree. For example, if two topics share a same vMF base and their corresponding mixture weights are close to each other, they may be semantically correlated. In previous study, we have examined several overlapping patterns, e.g., all topics share a same set of vMF bases. However, an issue is that such patterns often

output many twinborn topics. In this work, we use the following overlapping scheme: suppose that there are *G* groups of K' topics. In a group, each topic consists of C' personal vMF bases, and all topics in this group share *P* public vMF bases, where $C' + P = C$. In this setting, the total number of vMF bases is $G \times (K' \times C' + P)$, and topics in a group $\mathcal{G}_g$ use a same κ_g, i.e., $\kappa_g = \kappa_k = \cdots = \kappa_{k'}$ if $k \cdots k' \in \mathcal{G}_g$. The intuition behind overlapping by topic groups is that only a small set of topics may be semantically correlated. Besides, the personal vMF base design can effectively avoid the outputs of twinborn topics.

3.1 Inference

For MvTM, the topic proportions $\{\theta_d\}_{d=1}^{d=D}$ and the topic assignments $\{z_{dn}\}_{d=1,n=1}^{d=D,n=N_d}$ are hidden variables; and the topics $\{v\mathcal{MF}\left(\Delta_k\right)\}_{k=1}^{k=K}$ are model parameters. Given an observable document collection *W* consisting of word embeddings, we wish to compute the posterior distribution over θ and *z*, and to estimate $v\mathcal{MF}\left(\Delta\right)$.

Because the exact posterior distribution $p(\theta, z|W, \alpha, \Delta)$ is intractable to be computed, we must resort approximation inference algorithms. Due to the multinomial-Dirichlet design, the topic proportion θ can be analytically integrated out. We then use hybrid variational-Gibbs (HVG) (Mimno et al., 2012) to approximate a posterior over the topic assignment *z*: $p(z|W, \alpha, \Delta)$. A variational distribution of the following form is used:

$$q(z) = \prod_{d=1}^{D} q(z_d) \tag{3}$$

where $q(z_d)$ is a single distribution over the K^{N_d} possible topic configurations, rather than a product of N_d distributions. By using this variational distribution, we obtain an Evidence Lower BOund (ELBO) $\mathcal{L}$ as follows :

$$\log p(z|W, \alpha, \Delta) \geq \mathcal{L}(z_d, \Delta) \triangleq \mathrm{E}_q\left[\log p(W, z|\alpha, \Delta)\right] - \mathrm{E}_q\left[\log q(z)\right] \tag{4}$$

We then develop an expectation maximization (EM) process to optimize this ELBO, where in the E-step we maximize $\mathcal{L}$ with respect to the variational distribution $q(z)$, and in the M-step we maximize $\mathcal{L}$ with respect to the model parameter Δ, holding $q(z)$ fixed. Optimizing $q(z)$ directly is expensive because for each document *d* it needs to enumerate all K^{N_d} possible topic configurations. We therefore apply Monte-Carlo approximation to this ELBO $\mathcal{L}$ in Eq.4 by:

$$\begin{aligned}\mathcal{L}(z_d, \Delta) &\triangleq \mathrm{E}_q\left[\log p(W, z|\alpha, \Delta)\right] - \mathrm{E}_q\left[\log q(z)\right] \\ &\approx \frac{1}{B}\sum_{b=1}^{B}\left(\log p(W, z^{(b)}|\alpha, \Delta) - \log q\left(z^{(b)}\right)\right)\end{aligned} \tag{5}$$

where $\left\{z^{(b)}\right\}_{b=1}^{b=B}$ are samples drawn from $q(z)$. Because the variational distributions $q(z_d)$ are independent from each other, reviewing Eq.3, each document *d* drives a personal sampling process with respect to $q(z_d)$.

In the **E-step**, for each document *d* we use Gibbs sampling to draw *B* samples from $q(z_d)$. This sequentially samples topic assignment to each word embedding from the posterior distribution conditioned on all other variables and the data. The sampling equation is given by:

$$p(z_{dn} = k|z_d^{-n}, \alpha, \Delta) \propto (N_{dk}^{-n} + \alpha) \times v\mathcal{MF}(w_{dn}|\Delta_k) \tag{6}$$

where N_{dk} is the number of word embeddings assigned to topic *k* in document *d*; the superscript "-n" is a quantity that excludes the word embedding w_{dn}. During per-document Gibbs sampling, we iteratively run the MCMC chain a fixed number of times and save the last *B* samples.

In the **M-step**, we optimize Δ given all samples of *z* obtained in E-step. This is achieved by maximizing the following approximate ELBO $\mathcal{L}'$:

$$\mathcal{L}' \triangleq \frac{1}{B}\sum_{b=1}^{B}\left(\log p(W, z^{(b)}|\alpha, \Delta) + const\right) \tag{7}$$

For the disjoint setting, i.e., MvTM$_d$, the optimization of $\mathcal{L}'$ is equivalent to independently estimate Δ_k for each topic k. Due to space limit, we omit the derivation details (Gopal and Yang, 2014). Extracting all N_k word embeddings assigned to topic k, for each word embedding w_i we compute its weights for all C vMF bases by:

$$weight_{ic} = \frac{\pi_{k|c} v\mathcal{MF}(w_i|\mu_{k|c}, \kappa_k)}{\sum_{j=1}^{C} \pi_{k|j} v\mathcal{MF}(w_i|\mu_{k|j}, \kappa_k)} \tag{8}$$

and then update Δ_k by:

$$\begin{aligned} R_{k|c} &= \sum_{i=1}^{N_k} weight_{ic} \times w_i, \quad r_k = \sum_{c=1}^{C} \frac{\left\|R_{k|c}\right\|}{N_k} \\ \mu_{k|c} &= \frac{R_{k|c}}{\left\|R_{k|c}\right\|}, \quad \pi_{k|c} = \sum_{i=1}^{N_k} \frac{weight_{ic}}{N_k}, \quad \kappa_k = \frac{r_k M - r_k^3}{1 - r_k^2} \end{aligned} \tag{9}$$

For the overlapping setting, i.e., MvTM$_o$, there are a few changes to the optimization of $\mathcal{L}'$. In each topic group $\mathcal{G}_g$, the updates of π and μ of personal vMF bases remain unchanged, whereas the mean μ of public vMF bases and κ of this group are updated by:

$$r_g = \sum_{c=1}^{C} \frac{\left\|\sum_{k\in\mathcal{G}_g} R_{k|c}\right\|}{\sum_{k\in\mathcal{G}_g} N_k}, \quad \mu_{k|p} = \frac{\sum_{k\in\mathcal{G}_g} R_{k|p}}{\left\|\sum_{k\in\mathcal{G}_g} R_{k|p}\right\|}, \quad \kappa_g = \frac{r_g M - r_g^3}{1 - r_g^2} \tag{10}$$

where $\mu_{k|p}$ is the mean of the pth public vMF base for topic k and note that $\mu_{k|p} = \mu_{k'|p}$ if $k, k' \in \mathcal{G}_g$.

For clarity, the overall EM inference algorithm for MvTM is outlined in *Algorithm 1*.

Algorithm 1 Inference for MvTM

Initialize parameters.
For $t = 1, 2, \cdots, Max\_iter$ **do**
  E-step
    For document d=1 to D **do**
      Gibbs sampling for B topic assignments $z_d^{(b)}$ using Eq.6
    End for
  M-step
    For MvTM$_d$, optimize Δ using Eq.8 and 9.
    For MvTM$_0$, optimize Δ using Eq.8, 9 and 10.
End for

3.2 Time Complexity

We first analyze the time complexities of E-step and M-step, and then present the overall time cost of MvTM.

In the **E-step**, the main time cost is the topic assignment sampling of each word embedding over K topics. Reviewing Eq.6, one sampling process needs to compute the probabilities of the current word embedding, i.e., $v\mathcal{MF}(w_{dn}|\Delta_k)$, in all K topics, which requires $O(KCM)$ time. Fortunately, the topics are fixed in the E-step, thus we only need to compute the value of $v\mathcal{MF}(w|\Delta_k)$ for each word embedding at the beginning of each EM sweep, and save them in the memory. This requires $O(VKCM)$ time, where V is the number of word embeddings. Consequently, the topic sampling process of MvTM is equivalent to the sampling of Gibbs sampling LDA, requiring $O(K)$ time. We present that the per-iteration time complexity of E-step is given by $O(VKCM + \zeta N_V K)$, where ζ is the iteration number in per-document Gibbs sampling and N_V is the total number of word embeddings occurred in a corpus. Recently, sparse sampling algorithms (Yao et al., 2009; Li et al., 2014) effectively accelerate the sampling

Table 1: Summarization of data sets used in our experiments. N_V is the total number of word tokens; N_V/D is the average document length; "label" denotes the number of pre-assigned classes.

Data set	V	D	N_V	N_V/D	label
NG	18,127	18,768	1,946,487	104	20
NIPS	4,805	1,740	2,097,746	1,206	–
Wiki	7,702	44,819	6,851,615	153	–

of topic models. Inspired by (Li et al., 2016b), we employ the Alias method (Walker, 1977; Marsaglia et al., 2004) to reduce the per-word sampling cost from $O(K)$ to $O(K_d)$, where K_d is the number of instantiated topics in document d and commonly $K_d \ll K$. The per-iteration time complexity of E-step now is $O(VKCM + \zeta N_V K_d)$.

In the **M-step**, the time cost of MvTM$_d$ and that of MvTM$_o$ are almost the same. We only present the time complexity of MvTM$_d$. Reviewing the M-step, we see that the most expensive updates include Eq.8, the first and the fourth equations in Eq.9. They require $O(VCM)$, $O(VCM)$ and $O(VC)$. Thus we present that the (per-iteration) time complexity of M-step is $O(VCM)$.

Overall, we see that in each EM sweep the E-step dominates the run-time, giving an approximate total per-iteration time complexity $O(VKCM + \zeta N_V K_d)$. Clearly, MvTM is much efficient than Gibbs sampling G-LDA (Das et al., 2015), because G-LDA needs to repeatedly compute the determinant and inverse of the covariance matrix in Gaussian topics. For each word occurring, this spends $O(M^2)$ time, even using Cholesky decomposition.

4 Experiment

In this section, we evaluate MvTM qualitatively and quantitatively.

4.1 Experimental Setting

Data set Three data sets were used in our experiments, including Newsgroup (NG), NIPS and Wikipedia (Wiki). The NG data set is a collection of newsgroup documents, consisting of 20 classes. We will use NG to examine the classification performance of MvTM in Section 4.3. The NIPS data set is a collection of papers in the NIPS conference. The processed versions of these two data sets were downloaded from the open source of G-LDA[2]. For the Wiki data set, we downloaded a number of documents from online English Wikipedia, and processed these documents using a standard vocabulary[3]. The statistics of the three data sets are listed in Table 1.

Baseline model: In the experiments, we used two baseline models, including LDA[4] and G-LDA[2]. For both baseline models, we use their open source codes publicly available on the net. A pre-trained 50-dimensional word embeddings[5] were used. Especially for MvTM, we normalized the word embeddings.

4.2 Evaluation on Topics

We use the PMI score (Newman et al., 2010) to evaluate the quality of topics learnt by topic models. This metric is based on the pointwise mutual information of a power-law reference corpus. For a topic k, given T most probable words the PMI score is computed by:

$$PMI\,(k) = \frac{1}{T\,(T-1)} \sum_{1 \le i \le j \le T} \log \frac{p\,(w_i, w_j)}{p\,(w_i)\,p\,(w_j)} \tag{11}$$

where $p\,(w_i)$ and $p\,(w_i, w_j)$ are the probabilities of occurring word w_i and co-occurring word pattern (w_i, w_j) estimated by the reference corpus, respectively. In the experiments, we use the Palmetto[6] tool

[2]https://github.com/rajarshd/Gaussian_LDA
[3]http://www.cs.princeton.edu/∼mdhoffma/
[4]http://gibbslda.sourceforge.net/
[5]GloVe word embeddings available at http://nlp.stanford.edu/projects/glove/
[6]http://aksw.org/Projects/Palmetto.html

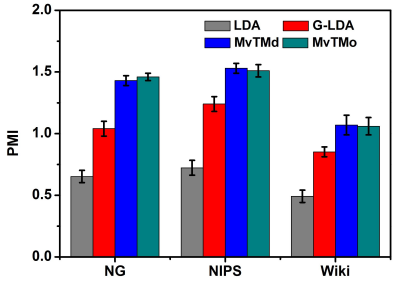


Figure 1: PMI performance of 15 top words on NG, NIPS and Wiki.

Table 2: Random selected examples of top words learnt by baseline models and our MvTM on NG.

LDA				G-LDA			
president	car	treating	space	government	car	disease	space
government	cars	writes	nasa	administration	university	food	nasa
fbi	engine	medical	gov	support	ohio	treatment	spacecraft
mr	good	cancer	orbit	state	cars	doctor	earth
clinton	oil	doctor	writes	military	carolina	medical	orbit
koresh	mr	doesn	don	leaders	virginia	eat	level
children	speed	treatment	moon	groups	harvard	patients	mars
people	drive	brain	mission	public	speed	cancer	put
batf	ford	patients	solar	policy	michigan	drink	asked
administration	article	drug	water	forces	missouri	course	shuttle
MvTM$_d$				**MvTM$_o$**			
country	car	disease	earth	country	wheel	patients	space
western	cars	treatment	orbit	government	door	treatments	earth
arab	driver	medical	mars	state	gear	therapy	orbit
muslim	bike	patients	light	president	car	treatment	mars
territory	drivers	infection	space	public	pulling	diabetes	spacecraft
government	truck	drugs	orbiting	policy	inside	diseases	light
war	vehicle	diseases	jupiter	leaders	wheels	hiv	surface
occupation	driving	brain	solar	administration	front	treating	orbiting
eastern	vehicles	tests	orbiter	war	stuck	disease	solar
occupied	wheel	treating	spacecraft	people	rolled	vaccine	orbiter

to compute PMI scores of the top 15 words.

We train baseline models and our MvTM with 50 topics, and evaluate the average PMI score of all topics. For MvTM$_d$, the number of vMF bases is set to 2, i.e., $C = 2$. For MvTM$_o$, topics are organized into ten groups, where each group consists of five topics; and the numbers of personal vMF bases and public vMF bases are set to 2 and 3, respectively[7].

The experimental PMI results on three data sets are shown in Figure 1. It is clearly seen that MvTM performs better than LDA and G-LDA. This implies that MvTM outputs more coherent topics. Some examples of top topic words are listed in Table 2. Overall, we see that the topics of MvTM seem more coherent than those of baseline models. The topics of LDA contain some noise words, e,g., "mr" and "don"; and G-LDA contains some less relevant words, e.g., the second topic of G-LDA is incoherent. In contrast, the topics of MvTM are more precise and clean. Besides, for MvTM$_o$ we measure topic correlation by computing the cosine between vMF weights of topics in the same group. Some topic pairs with high cosine similarity scores, such as $\langle patients, treatments, therapy, treatment, diabetes\rangle$ and $\langle blood, skin, heart, stomach, breathing\rangle$, seem semantically correlated.

[7]In previous experiments, we found that using mixtures of vMFs with 2 bases is able to better represent topics.

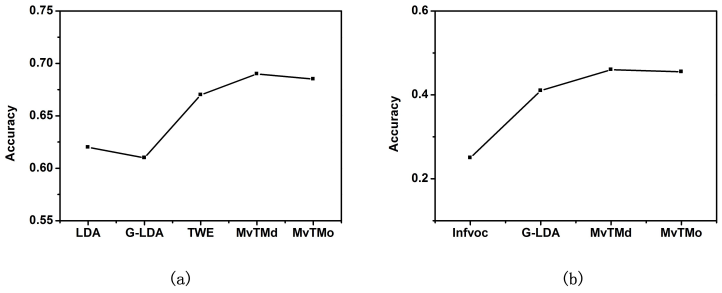


Figure 2: Classification performance on NG: (a) original test documents and (b) test documents with new words.

4.3 Evaluation on Classification

We compare the classification performance of MvTM with baseline topic models across NG. Two new baselines are used, i.e., topical word embedding (TWE) (Liu et al., 2015) and infvoc (Zhai and Boyd-Graber, 2013). For all models, we learn the topic proportions (K=50) as features of documents, and then use the SVM classifier implemented by LibSVM[8].

The results of original test documents are shown in Figure 2(a). Clearly, MvTM achieves better performance than LDA, G-LDA and TWE. MvTM can handle absent words in training data. To examine this ability, we compare MvTM with G-LDA and infvoc[9], where the two also can handle unseen words. We replace a number of words in test documents with synonyms by using WordNet as in (Das et al., 2015). The classification results are shown in Figure 2(b). It can be seen that MvTM outperforms G-LDA and infvoc. The results imply that MvTM works well even future documents containing new words. This may be insignificant in practice.

5 Related Work

Some early works have attempted to combine topic modeling with embeddings. (Hu et al., 2012) proposed a model to describe indexing representations for audio retrieval, which is similar with G-LDA. Another work (Wan et al., 2012) jointly estimates parameters of a topic model and a neural network to represent topics of images.

Recently, (Liu et al., 2015) proposed a straightforward TWE model. This model separately trains a topic model and word embeddings on the same corpus, and then uses the average of embeddings assigned to the same topic as the topic embedding. A limitation of TWE is that it lacks statistical foundations. Another modification latent feature topic modeling (LFTM) (Nguyen et al., 2015) extends LDA and Dirichlet multinomial mixture by incorporating word embeddings as latent features. However, LFTM may be infeasible for large-scale data sets, since it, i.e., the code provided by its authors, is time-consuming. A most recent nonparametric model (Batmanghelich et al., 2016) also uses vMF to describe the topic over word embeddings, where a topic is represented by a single vMF on the embedding space. By contrast, it may be less effective to capture complex topic structure.

6 Conclusion and Discussion

In this paper, we investigate how to improve topic modeling with word embeddings. A previous art G-LDA defines Gaussian topics over word embeddings, however, the word weights of topics are measured by the Euclidean similarity. To address this problem and further capture complex topic structure, we use mixtures of vMFs to model topics, and then propose a novel MvTM algorithm. The vMF bases of topics in MvTM can be either disjoint or overlapping, leading to two versions of MvTM. The overlapping MvTM can describe topic correlation in some degree. In empirical evaluations, we use the per-trained GloVe word embeddings, and then compare MvTM with LDA and G-LDA on three real-world data

[8]https://www.csie.ntu.edu.tw/∼cjlin/libsvm/

[9]For fair comparison, we train infvoc by a batch optimization procedure.

sets. The experimental results indicate that compared to the state-of-the-art baseline models MvTM can discover more coherent topics measured by PMI, and achieve competitive classification performance. In the future, we are interested in supervised versions of MvTM, directly applying to basic document tasks such as sentiment analysis.

Acknowledgements

This work was supported by National Natural Science Foundation of China (NSFC) under the Grant No. 61133011, and 61103091. We thank the reviewers for their useful comments.

References

Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, and Suvrit Sra. 2005. Clustering on the unit hypersphere using von Mises-Fisher distributions. *Journal of Machine Learning Research*, 6:1345–1382.

Kayhan Batmanghelich, Ardavan Saeedi, Karthik Narasimhan, and Sam Gershman. 2016. Nonparametric spherical topic modeling with word embeddings. *arXiv:1604.00126v1*.

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

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

David M. Blei. 2012. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for topic models with word embeddings. In *Annual Meeting of the Association for Computational Linguistics*, pages 795–804.

Siddharth Gopal and Yiming Yang. 2014. Von mises-fisher clustering models. In *International Conference on Machine Learning*, pages 154–162.

Pengfei Hu, Wenju Liu, Wei Jiang, and Zhanlei Yang. 2012. Latent topic model based on Gaussian-LDA for audio retrieval. In *Pattern Recognition, volume 321 of CCIS*, pages 556–563.

Aaron Q. Li, Amr Ahmed, Sujith Ravi, and Alexander J. Smola. 2014. Reducing the sampling complexity of topic models. In *International Conference on Knowledge Discovery and Data Mining*.

Shaohua Li, Tat-Seng Chua, Jun Zhu, and Chunyan Miao. 2016a. Generative topic embedding: a continuous representation of documents. In *Annual Meeting of the Association for Computational Linguistics*.

Ximing Li, Jihong Ouyang, and Xiaotang Zhou. 2016b. Sparse hybrid variational-gibbs algorithm for latent Dirichlet allocation. In *SIAM International Conference on Data Mining*.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and MaosongSun. 2015. Topical word embeddings. In *Association for the Advancement of Artificial Intelligence*, pages 2418–2424.

George Marsaglia, Wai Wan Tsang, and Jingbo Wang. 2004. Fast generation of discrete random variables. *Journal of Statistical Software*, 11:1–8.

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

David Mimno, Matthew D. Hoffman, and David M. Blei. 2012. Sparse stochastic inference for latent Dirichlet allocation. In *International Conference on Machine Learning*.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwi. 2010. Automatic evaluation of topic coherence. In *Annual Conference of the North American Chapter of the ACL*, pages 100–108.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics*, 3:299–313.

Alastair J. Walker. 1977. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software*, 3(3):253–256.

Li Wan, Leo Zhu, and Rob Fergus. 2012. A hybrid neural network-latent topic model. In *International Conference on Artificial Intelligence and Statistics*, pages 1287–1294.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: A non-Markov continuous-time model of topical trends. In *International Conference on Knowledge Discovery and Data Mining*, pages 424–433.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections. In *International Conference on Knowledge Discovery and Data Mining*.

Ke Zhai and Jordan L. Boyd-Graber. 2013. Online latent Dirichlet allocation with infinite vocabulary. In *International Conference on Machine Learning*, pages 561–569.

Bayesian Language Model based on Mixture of Segmental Contexts for Spontaneous Utterances with Unexpected Words

Ryu Takeda **Kazunori Komatani**
The Institute of Scientific and Industrial Research, Osaka University
8-1, Mihogaoka, Ibaraki, Osaka 567-0047, Japan
{rtakeda, komatani}@sanken.osaka-u.ac.jp

Abstract

This paper describes a Bayesian language model for predicting spontaneous utterances. People sometimes say unexpected words, such as fillers or hesitations, that cause the miss-prediction of words in normal N-gram models. Our proposed model considers mixtures of possible segmental contexts, that is, a kind of context-word selection. It can reduce negative effects caused by unexpected words because it represents conditional occurrence probabilities of a word as weighted mixtures of possible segmental contexts. The tuning of mixture weights is the key issue in this approach as the segment patterns becomes numerous, thus we resolve it by using Bayesian model. The generative process is achieved by combining the stick-breaking process and the process used in the variable order Pitman-Yor language model. Experimental evaluations revealed that our model outperformed contiguous N-gram models in terms of perplexity for noisy text including hesitations.

1 Introduction

1.1 Background

Language models (LMs) are widely used for text analysis, word segmentation and word prediction in automatic speech recognition (ASR). The basic LM is a conventional N-gram model that predicts a word depending on the patterns of the previous N words (*context*). The probability of a word is usually calculated by counting the words that match the context in text data as maximum likelihood estimation. Therefore, the model easily predicts frequent words or set expressions but not rare words or phrases.

Various N-gram language models have been proposed to prevent the incorrect probability assignment caused by the increase of the context length N. Since the number of combinations of N becomes $O(V^N)$ for vocabulary size V, there are a lot of patterns that do not appear in training data (data sparseness). Using an N-gram model based on a Bayesian framework is a promising approach for data sparseness. Because it is based on a Bayesian framework, an LM based on hierarchical Pitman-Yor process (HPYLM) has two main differences from previous language models (Teh, 2006), such as Witten-bell (WB) (Witten and Bell, 1991) and Kneser-ney (KN) smoothing (Kneser and Ney, 1995): 1) a Bayesian model expressing conventional smoothing methods and 2) automatic tuning of parameters from data. Since HPYLM is based on a Bayesian framework, we can integrate other probabilistic models theoretically for other problems and apply optimization methods in accordance with a Bayesian framework. In contrast, other smoothing methods has several parameters that need to be tuned manually.

Human utterances contain various fillers and hesitations (left in Fig. 1), and these cause the misprediction of words because they rarely appear in the training data, that is, another type of sparsity. This will affect 1) the word prediction accuracy in ASR and 2) the precision of word segmentation (Mochihashi et al., 2009) or lexicon acquisition from speech signal (Elsner et al., 2013; Kamper et al., 2016; Taniguchi et al., 2016), which are our main interest. Since such hesitations are usually not registered

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 161–170, Osaka, Japan, December 11-17 2016.

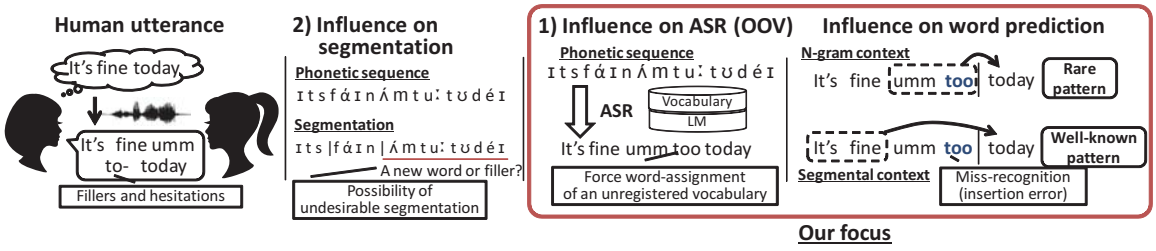


Figure 1: Problem caused by unexpected and inserted words

to an ASR vocabulary (out-of-vocabulary; OOV), they are recognized as the most similar and likely word in the vocabulary set in terms of pronunciation and context (middle-right in Fig. 1). Moreover, mis-recognized words may also affect the subsequent word prediction based on N-gram auto-regression. Such mis-recognition is a kind of insertion error caused by fillers, hesitations and other noise signals, such as coughs. For example, the hesitation "to-" is recognized as "too", and the filler "umm" and hesitation "too" are used for the prediction of the next word if we use normal N-gram model (right upper in Fig. 1). Note that hesitations are hard to eliminate by using only a filler-word list because their complete patterns cannot be prepared in advance. As for word segmentation and lexicon acquisition, the language model is trained from character/phoneme sequences or raw speech signal in an *unsupervised* manner. The Bayesian nonparametrics is often applied to this problem because it enables us to control the number of words/symbols dynamically according to the amount of data. Since the lexicon acquisition includes a kind of segmentation problem, fillers and hesitations may cause mis-segmentations. A nonparametric generative model that can deal with hesitations and fillers will help to recognize words sequence and segment words from phoneme sequence.

We propose using a Bayesian language model in which probability consists of a mixture of conditioned probabilities of segmental contexts for the word prediction problem. Since the lexicon acquisition from phonetic sequence or raw *conversational* speech signal is also our scope, Bayesian approach is necessary in terms of scalability. Our model removes (ignores) some words, such as fillers and hesitations in the ideal case, from the context in predicting words. For example, given the text "It's fine umm too today," the probability $p(\text{today}|\text{It's}, \text{fine}, \text{umm}, \text{too})$ is defined as a mixture of $\hat{p}(\text{today}|\text{It's}, \text{fine}, \text{umm}, \text{too})$, $\hat{p}(\text{today}|\text{fine}, \text{umm})$, $\hat{p}(\text{today}|\text{It's}, \text{fine})$ and so on (right lower Fig. 1). The risk of mis-prediction caused by the unknown context is reduced by other differently conditioned probabilities. Since the given term includes many patterns of segmental context, we constrain the pattern to one "contiguous" segment. That is, the probabilities of a discontiguous segment, such as $p(\text{today}|\text{It's}, \text{umm})$, are not included in the mixture. Since the generative process can be expressed by combining the stick-breaking process (Sethuraman, 1994) and the process used in the variable order Pitman-Yor language model (VPYLM) (Mochihashi and Sumita, 2007), the parameters can be estimated by Gibbs sampling (Christopher Michael Bishop, 2006) the same as they are for VPYLM.

1.2 Related Work on Mixture Models

The main differences between our work and previous studies are 1) assumed context patterns in the mixture and their purpose (text-level or utterance-level), and 2) whether the model is Bayesian or not. Our proposed model is one of various mixture language models and there are several language mixture models that consider word dependency. Again, we stochastically *ignore* some contiguous words in the context in accordance with the appearance of fillers, hesitations and noises (right in Fig. 1) at the utterance-level. Since other LM models correspond to the process for text generation in our framework, we can embed them in our process as mixture components if necessary. As shown in the right half of Fig. 2, our current model is based on the mixture of VPYLM which is based on the mixture of HPYLM. Note that VPYLM and HPYLM have no mechanism to select words in the context for prediction.

Previous studies used all combinations or syntactic structure of N words in context, and their methods are complex to deal with our filler/hesitation problems. The left half of Fig.2 shows a generalized lan-

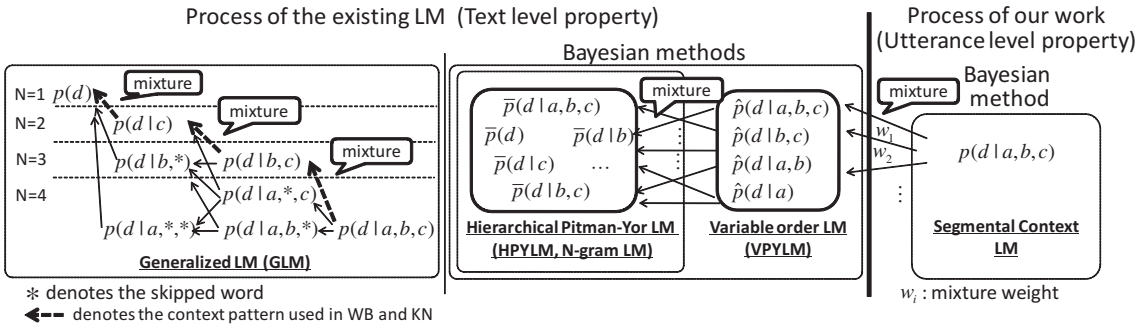


Figure 2: Language model structures: GLM (left), HPYLM/VPYLM (middle) and our model (right)

guage model (GLM) that mixes all probabilities of possible context patterns of N-grams hierarchically (Pickhardt et al., 2014). At each context depth, a word is *skipped* in the context (skip N-gram (Goodman, 2001; Guthrie et al., 2006)), and the probability is smoothed by shallow contexts. The relative position in the context remains, and the skipped word is denoted by the asterisk $*$. WB and KN use only the contiguous contexts for smoothing as shown in the Figure. Wu and Matsumoto (2015) proposed a hierarchical word sequence language model using directional information. The most frequently used word in the sentence is selected for splitting a sentence into two substrings, and a binary-tree is constructed by a recursive split. If a directional structure is assumed, the context patterns decrease in size and the processing time is shortened.

Running a language model on a recurrent neural network (RNN) (Mikolov et al., 2010) is, of course, a reasonable choice because of the good prediction performance for closed-vocabulary task. However, a neural network LM usually does not include a generative process, so it is difficult to apply to *unsupervised* training of a language model or lexicon acquisition from speech signals. In that sense, the LM based on generative model is still important. Of course, the combination method of Bayesian model and neural networks should be investigated for practical use.

Our work is the extension of VPYLM based on mixture of segmental contexts to deal with hesitations and fillers. And our mixture pattern is designed for hesitation and fillers, and it is simpler than that of others in terms of the number of context patterns.

2 Hierarchical Bayesian Language Model based on Pitman-Yor Process

This section explains the fundamental mechanism of a language model based on Bayesian nonparametrics. HPYLM should predict words more accurately than KN-smoothing because KN-smoothing is an approximation of this model.

2.1 Generative Model

The N-gram LM approximates the distribution over sentences $w_T, ..., w_1$ using the conditional distribution of each word w_t given a context $\boldsymbol{h}$ consisting of only the previous $N-1$ words $\boldsymbol{w}_{N-1}^{t-1} = \{w_{t-1}, ..., w_{t-N+1}\}$,

$$p(w_T, ..., w_1) = \prod_t^T p(w_t | \boldsymbol{w}_{N-1}^{t-1}). \tag{1}$$

The trigram model ($N = 3$) is typically used. Since the number of parameters increases exponentially as N becomes larger, the maximum-likelihood estimation severely overfits the training data. Therefore, smoothing methods are required if vocabulary V is large.

The probabilistic generative process of sentences based on HPY is explained by the Hierarchical Chinese restaurant process (CRP). In the CRP, there are tree-structured restaurants with tables and customers that are regarded as latent variables of words. When a customer enters the leaf restaurant $\boldsymbol{h}$, which corresponds to context, he/she sits down at an existing table or a new table depending on some probabilities.

If he/she selects a new table, an agent of the customer recursively enters the parent restaurant $\boldsymbol{h}'$ as a new customer. Here, we represent the depth of $\boldsymbol{h}$ as $|\boldsymbol{h}|$, and there is the relationship $|\boldsymbol{h}'| = |\boldsymbol{h}| - 1$. Given the seating arrangement of customers $\boldsymbol{s}$, the conditional probability of word w with the context $\boldsymbol{h}$ is defined as follows

$$p(w_t|\boldsymbol{s}, \boldsymbol{h}) = \frac{c_{\boldsymbol{h}w} - d_{|\boldsymbol{h}|}t_{\boldsymbol{h}w}}{c_{\boldsymbol{h}*} + \theta_{|\boldsymbol{h}|}} + \frac{\theta_{\boldsymbol{h}} + d_{|\boldsymbol{h}|}t_{\boldsymbol{h}*}}{c_{\boldsymbol{h}*} + \theta_{|\boldsymbol{h}|}} p(w_t|\boldsymbol{s}, \boldsymbol{h}'), \tag{2}$$

where $c_{\boldsymbol{h}w}$ is the count of word w at context $\boldsymbol{h}$, and $c_{\boldsymbol{h}*} = \sum_w c_{\boldsymbol{h}w}$ is its sum. $t_{\boldsymbol{h}w}$ is the number of table at context $\boldsymbol{h}$, and $t_{\boldsymbol{h}*}$ is also its sum. $\theta_{|\boldsymbol{h}|}$ and $d_{|\boldsymbol{h}|}$ are the common parameters of $\boldsymbol{h}$ with the same depth $|\boldsymbol{h}|$. The distribution over the current word given the empty context ϕ is assumed to be uniform over the vocabulary w of V words. The variable order PYLM integrates out the context length (depth) N, thus we need not determine the length in advance.

The predictive probability of word w is approximated by averaging Eq. (2) over sampled seating arrangement $\boldsymbol{s}_n (n = 1, ..., N)$.

$$p(w|\boldsymbol{h}) = \frac{1}{N} \sum_n p(w|\boldsymbol{s}_n, \boldsymbol{h}) \tag{3}$$

2.2 Inference of Parameters

The latent variable $\boldsymbol{s}$ and other parameters d and θ are obtained through simulations on the basis of Gibbs sampling given training text $\bar{w}_i (i = 1, ..., N_{train})$. The procedure for sampling a customer is as follows:

1. Add all customers to the restaurants
2. Select a certain customer $\bar{w}_i$
3. Remove the customer from the restaurant. If a table becomes null, also remove the agent from the parent restaurant recursively.
4. Add the customer to the leaf restaurant. He chooses a table with probabilities proportional to the number of customers at each table. If the table is null, also add an agent to the parent restaurant recursively. (Go back to Step 2).

The parameters are sampled using auxiliary variables from their posterior probability. Please see the work of Teh (Teh, 2006) for the detailed sampling algorithm.

2.3 Problem of Contiguous Context Model

The N-gram model is modeled as a series of words, and has an advantage in expressing common phrases. The Bayesian nonparametrics enables the N-gram model to tune the smoothing parameters automatically. This improves the accuracy of predicting rare words in a large context.

Unexpected words degrade the prediction accuracy of the N-gram model. The unexpected words include noises, fillers, and hesitations in actual utterances. For example, the probability of $p(\text{sing}|\text{he}, \text{will})$ is estimated reliably. However, the probability of $p(\text{sing}|\text{will}, \text{sh..})$, which includes a hesitation ("sh.."), is estimated unreliably because the hesitation does not appear in the corpus. The patterns of insertion location and bursty are also not determined in advance.

3 Bayesian Language Model based on Mixture of Segmental Contexts

This section explains the segmental context model for utterances. First, we explain the generative model and then its parameter inference. Note that the aim of this model is to improve the accuracy of word prediction under noisy context condition, not to detect fillers and hesitations.

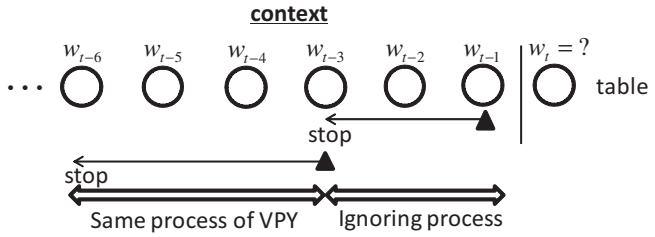


Figure 3: Process for segmental context model

3.1 Generative Model

We assume that the conditional distribution of each word w_t given a context is a mixture of the segmental N-gram context. The segmental N-gram is a part of context $w_{t-i}, .., w_{t-j}$, which begins at w_{t-i} and ends at w_{t-j}.

$$p(w|\boldsymbol{w}_{N-1}^{t-1}) = \sum_i \sum_{j>i} p(w|\boldsymbol{w}_{N-1}^{t-1}, i, j)p(i, j) = \sum_i \sum_{j>i} p(w|\boldsymbol{w}_j^{t-i})p(j|i)p(i) \tag{4}$$

If we consider the $N \to \infty$, the possible segmental patterns are also considered. Setting the start index i of N-gram appropriately can eliminate the influence of the sequential unexpected words for predicting the next word. The word probability term $p(w_t|\boldsymbol{w}_j^{t-i})$ is determined by HPYLM.

The stick-breaking process (SBP) represents the generative process of Eq. (4) as the same way of VPYLM (Mochihashi and Sumita, 2007). The process consists of two parts; 1) decide the start index i of N-gram and then 2) decide the end index j of N-gram. Each index is determined probabilistically using SBP (Fig. 3).

Step1 - Process for start index i: First, the customer walks along the tables (word) from the start, w_{t-1}. The customer stops at the i-th table with probability η_i, and passes it with probability $1 - \eta_i$. Therefore, the probability that the customer stops at the i-th table is given by

$$p(i|\boldsymbol{\eta}) = \eta_i \prod_{l=1}^{i-1}(1 - \eta_l). \tag{5}$$

This probability decreases exponentially. We assume that the prior of parameters $\boldsymbol{\eta}$ is Beta distribution Beta(α_1, β_1).

Step2 - Process for end index j: The end index j is also determined using the same process i. The customer walks along the tables from the i-th table, and stops at or passes the j-th table with probability ζ_j or $1 - \zeta_j$, respectively.

$$p(j|i, \boldsymbol{\zeta}) = \zeta_j \prod_{l=1}^{j-1}(1 - \zeta_l). \tag{6}$$

The prior of parameters $\boldsymbol{\zeta}$ is also assumed to be the Beta distribution Beta(α_2, β_2).

In fact, the whole process can be considered to be the combination of VPYLM and the start index determination process. We thus can describe the probability as

$$p(w_t|\boldsymbol{w}_\infty^{t-1}) = \sum_i P_{\text{vpy}}(w|\boldsymbol{w}_\infty^{t-i-1})p(i). \tag{7}$$

If we determine from which element, $P_{\text{vpy}}(w|\boldsymbol{w}_\infty^{t-i-1})$, the word comes in step 1, the latter process is the same as the VPYLM. In practice, we set a maximum length of context for parameter estimation.

Table 1: Parameters of experiment

	Artificial noisy data		Actual hesitation data
	English	Japanese	Japanese
Target text	War and Peace	CSJ	CSJ
Training	27876 sentences 479585 words	110566 sentences 2828499 words	114372 sentences 3084592 words
Test for clean	5128 sentences 88552 words	7134 sentences 199100 words	20440 sentences 184145 words
Test for noisy	5128 sentences 97373 words	7134 sentences 218945 words	2296 sentences 32342 words
Vocabulary size	10717	18357	19703
(α_1, β_1)	(9, 1)		(1, 1)
(α_2, β_2)	(1, 9)		(1, 8)

3.2 Inference of Start Index

We assume that all words in training data $\bar{\boldsymbol{w}}$ have the start index i_t as a latent variable, and are estimated stochastically by Gibbs sampling. The start index i_t of the word $\bar{w}_t$ is sampled given data $\bar{\boldsymbol{w}}$, seating arrangement $\boldsymbol{s}$, and start and end indexes of other words $\boldsymbol{i}_{-t}$ and $\boldsymbol{j}_{-t}$ as

$$i_t \sim p(i_t|\bar{\boldsymbol{w}}, \boldsymbol{s}_{-t}, \boldsymbol{j}_{-t}, \boldsymbol{i}_{-t}) \tag{8}$$

$$\propto p(w_t|\bar{\boldsymbol{w}}_{-t}, \boldsymbol{j}_{-t}, \boldsymbol{i}) p(i_t|\bar{\boldsymbol{w}}_{-t}, \boldsymbol{s}_{-t}, \boldsymbol{i}_{-t}, \boldsymbol{j}_{-t}) \tag{9}$$

where the notation $-t$ means that the t-th element corresponding to $\bar{w}_t$ is excluded. Here, the first term, $p(\bar{w}_t|\bar{\boldsymbol{w}}_{-t}, \boldsymbol{j}_{-t}, \boldsymbol{i})$, is calculated using VPYLM because the start index i_t is given. The second term is a prior probability to select the start index. It can be calculated in the same way used in the VPYLM:

$$p(i_t = l|\boldsymbol{w}_{-t}, \boldsymbol{s}_{-t}, \boldsymbol{i}_{-t}, \boldsymbol{j}_{-t}) = \frac{a_l + \alpha_1}{a_l + b_l + \alpha_1 + \beta_1} \prod_{k=1}^{l} \frac{b_k + \beta_1}{a_k + b_k + \alpha_1 + \beta_1}, \tag{10}$$

where α_1 and β_1 are hyper-parameters of the Beta distribution. The a_l and b_l are the count of customers who stopped at and those who passed table $\bar{w}_l$. This probability is assumed to depend only on $\bar{w}_l$, not whole context $\boldsymbol{h}$. Since the probability of the word corresponding to $\bar{w}_l$ is not important for the prediction is low, the effect of an unexpected word on this index is reduced.

Once the start index is set, we can also draw the end index j_t and the seating arrangement s_t through VPYLM process. The j_t is first drawn from its posterior distribution, and then seating s_t is also drawn from its posterior distribution. After sampling, the average word probability is used for prediction.

The computational cost of our model is proportional to $O(N)$ while the cost of the generalized language model is roughly proportional to $O(2^N)$. The enumeration of all combinations of words that should be used is computationally heavy for models based on Bayesian nonparametrics when N becomes larger and we optimize parameters of the model. Moreover, the context pattern of the generalized model is complex to deal with fillers and hesitations (insertion errors).

4 Experimental Evaluations

4.1 Experimental Setup

We used two kinds of text for evaluation: 1) artificial noisy text and 2) actual hesitation text (Japanese only). The former is for the validation of our method with model-matched data, and the latter is for the performance measurement with real utterances.

We used two languages English and Japanese text data for training and test dataset for the artificial noisy text. The English text was "War and Peace" from project Gutenberg[1], and the Japanese text was the Corpus of Spontaneous Japanese (CSJ[2]), consists of transcriptions of Japanese speech. For the English

[1] http://www.gutenberg.org/

[2] https://www.ninjal.ac.jp/english/products/csj/

text, we randomly selected 27,876 sentences from the entire of "War and Peace" for training data, and used the remaining 5,128 sentences for test data. For Japanese text, we used 110,566 sentences in the "non-core" set for training data and 7,134 sentences in the "core" set for test data. All hesitations and fillers were eliminated from the Japanese corpus to make it formal text data [3]. The utterances in the CSJ that have 0.5-second short-pauses were separated into sub-utterances, and each sub-utterance was treated as a sentence. The words that appeared more than once were selected for the vocabularies. The sizes of vocabularies were 10,717 words for English text and 18,357 words for Japanese text. To simulate the artificial noisy text, we added words randomly selected from vocabularies into the test data at a rate of 10 %. The OOVs in the test set were treated as a symbol, "<unk>".

The raw CSJ Japanese transcription text was used for the actual hesitation text. In this experiment, hesitations and fillers in the training set are *not* eliminated. The utterances that have 0.2-second short-pauses were separated into sub-utterance, and each sub-utterance was treated as a sentence. The 0.2-second is selected to make a rate of hesitation in noisy text about 8.0%. The test transcription data ("core" set) were divided into two categories: hesitation-included noisy text (2,296 sentences) and clean text (20,440 sentences). The number of hesitations in the test dataset was 2649 (about $2649/32342 = 8.1\%$). The hesitation-included noisy text included hesitations, so its vocabulary was 19,703. The out of vocabulary (OOV) words in the hesitation test data were replaced by words randomly selected from the vocabulary set that had a phoneme distance to the OOV word of less than 2. This is because *such OOV words including unknown hesitations are actually mis-recognized and assigned similar-sounding words in the ASR vocabulary*. Therefore, the vocabulary set was closed. Note that *frequent fillers and hesitations remained in both the test and training sets*. These settings are listed in Tab. 1.

We compared our model with other models: WB, KN, Modified KN (MKN) (Chen and Goodman, 1999), HPYLM, and VPYLM. The hyper-parameters, α_2 and β_2 of the Beta distribution used in VPYLM were set to 1 and 9 for the artificial data, and 1 and 8 for the actual hesitation data. Additionally, those of the start index process, α_1 and β_1, were set to 9 and 1 for the artificial data, and 1 and 1 for the actual hesitation data. These parameters were selected to perform best for each test set to evaluate the limitation of methods. For the English and Japanese text, N was set to 3, 4, 6, 10. For the Japanese transcription, it was set to 3, 6, 8, 10. The predictive probability was averaged over 30 seating arrangements after 90 iterations of Gibbs sampling. We also investigate the performance of RNN language model [4] as a reference. We tried several parameter set of RNN, such as the number of hidden layers and classes, and they are also tuned for each test set. Note that the main interest of our experiments is the performance comparison among Bayesian methods.

Perplexity (PP) was used as the evaluation criterion.

$$\text{PP} = 2^{P(\boldsymbol{w}_{\text{test}})}, \quad P(\boldsymbol{w}_{\text{test}}) = -\frac{1}{N_{\text{test}}} \sum_{s \in w_{\text{test}}} \log P(s), \tag{11}$$

where s is a sentence in the test data and N_{test} is the number of words in the test dataset. The PP was calculated under the assumption that each sentence was independent. Smaller PP values mean better word prediction accuracy. The prediction of OOVs, which are denoted by "<unk>", in the artificial test set is eliminated in calculating perplexity.

4.2 Results and Discussion

4.2.1 Artificial Noisy Data

The perplexity values for the two data sets and the four N-gram lengths can be seen in Tabs. 2 and 3 for English and Japanese text, respectively. The clean text denotes the raw formatted text, and the noisy text denotes the ones with randomly-added words. There is no noteworthy difference between the English and Japanese text other than the range of PP.

The differences among methods for clean text data with $N = 3$ are clear. Like in the results of previous studies, HPYLM and MKN had the lowest PP, followed by VPYLM and WB. Our model had worse PP

[3] All words were tagged by hand. The tags of fillers and hesitations were included.

[4] https://github.com/pyk/rnnlm-0.4b

Table 2: Perplexity for English text

Test dataset	Method	maximum context length N 3	4	6	10
Clean text	WB	167.9	163.8	163.2	163.2
	KN	152.7	150.9	157.9	157.8
	MKN	153.1	151.3	156.7	157.9
	HPYLM	155.0	151.6	151.5	151.6
	VPYLM	156.0	153.3	153.2	153.2
	Ours	161.7	154.6	152.7	152.9
	RNN	135.4			
Noisy text	WB	365.9	360.0	359.2	359.2
	KN	328.3	322.4	331.2	332.5
	MKN	321.4	316.5	328.2	331.8
	HPYLM	326.0	321.8	321.5	321.6
	VPYLM	327.4	324.0	324.1	324.2
	Ours	322.4	309.5	306.0	306.0
	RNN	312.0			

Table 3: Perplexity for Japanese text

Test dataset	Method	maximum context length N 3	4	6	10
Clean text	WB	56.6	55.8	56.6	56.9
	KN	53.1	50.7	50.4	51.4
	MKN	52.3	50.0	49.4	50.3
	HPYLM	52.1	50.0	49.5	49.5
	VPYLM	52.2	50.5	50.0	49.9
	Ours	53.1	51.0	50.4	50.5
	RNN	46.1			
Noisy text	WB	180.7	178.9	180.4	181.0
	KN	174.0	166.1	163.4	165.0
	MKN	164.4	158.3	156.3	159.3
	HPYLM	160.9	156.9	156.1	156.0
	VPYLM	158.8	155.0	153.3	152.4
	Ours	147.0	143.2	143.2	144.3
	RNN	166.1			

Table 4: Perplexity for Japanese Transcription

Test dataset	Method	maximum context length N 3	6	8	10
Clean text	WB	61.3	62.1	62.3	62.4
	KN	57.6	55.5	56.1	56.4
	MKN	53.9	56.8	54.8	54.0
	HPYLM	56.3	54.5	54.5	54.5
	VPYLM	56.4	54.7	54.7	54.7
	Ours	57.4	55.0	55.0	55.0
	RNN	46.0			

Test dataset	Method	maximum context length N 3	6	8	10
Noisy text (hesitations)	WB	102.4	104.4	104.8	104.9
	KN	95.6	92.2	93.2	93.7
	MKN	93.1	89.5	89.8	90.6
	HPYLM	91.3	89.0	89.1	89.0
	VPYLM	91.2	89.0	89.0	89.1
	Ours	91.8	88.2	88.1	88.2
	RNN	83.1			

than MKN, HPYLM and VPYLM. Since our model stochastically ignores some contiguous words in the context, the prediction accuracy for formatted text was worse than those of other methods. This can be reduced by using more text data or an improved model discussed in the next subsection. Using a longer context improved the PPs of HPYLM and our model. Therefore, a longer context is useful for word prediction. The perplexity of RNN was smallest, and RNN outperformed others by 15 and 4 points for English and Japanese text.

The ranking were different for the noisy text data. The relative performances of WB, KN, MKN, HPYLM, and VPYLM were almost the same as those for the clean text, but our model had the lowest PP. Its performance improved with the context length $N = 6$ or 10. The perplexity of RNN is also higher than that of our model. This indicates that the segmental context mixture works as intended, i.e. reducing the negative effect of unknown context. The improvement with a longer context means that Bayesian smoothing works well.

4.2.2 Actual Hesitation Data

The perplexity values for the four N-gram lengths can be seen in Tab. 4 for clean sentences and hesitation included sentences. The perplexity was much higher for all four models with the noisy text mainly due to hesitations and substitution errors caused by OOVs. Therefore, the word-prediction for actual utterance is more difficult than written text.

The relative performances were almost the same as those for artificial noisy data although the improvement of perplexity seems to be slight. That indicates that our model is effective for the actual transcription. The differences of perplexity among models are smaller than with artificial noisy data due to a) the difference in the hesitation-word ratio (about 8 %) , b) the appearance of patterns of fillers or hesitations in the training text, and c) the substitution of hesitations to pre-defined vocabularies (closed vocabulary set). The substitution suffers the estimation of true skip probability Eq. (6) and (9) of hesitations and true vocabulary. This means that we need to handle hesitation problem in raw-level symbol sequence, such as phoneme sequence. The reason the RNN outperformed our model might be due to the closed vocabulary set in this experiment. On the other hand, the context information in RNN might be

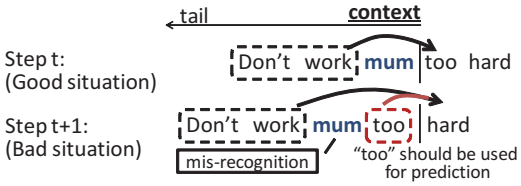


Figure 4: Weakness of segmental context model

suffered from the contiguous noisy words that were caused by the combination of the noise word and OOVs, "<unk>", in the artificial noisy data, and RNN degraded prediction accuracy for artificial noisy data. This indicated that RNN is unfamiliar to open-vocabulary tasks, such as lexicon acquisition.

The model validation with these text-level experiments provides us important knowledge and significant results for the next-level research step. Our method will be more effective for the word/phoneme segmentation problem because the substitution of hesitations to OOVs does not happen and we have to handle raw hesitation symbols. For example, the hesitation "to-" will be treated as itself "to-" or a phonetic expression "t u:", and the skip prior/posterior probability Eq. (6) and (9) of a hesitation symbol will be estimated properly. Our model will provide criteria for which words or symbols should be skipped. Therefore, the model integration of ours and the OOV-free model (Mochihashi et al., 2009) is required to process actual conversational utterances.

4.2.3 Remaining Problem on Model

The main problem of our model is clear from these results: it completely ignores neighbor context and does not use it for prediction, as illustrated in Figure 4. Since the neighbor words are usually useful for prediction, ignoring such words will degrade perplexity, especially that of clean text. The actual fillers/hesitations and mis-recognized words move from head to tail in the context in predicting words sequentially. Therefore, if the unknown segment is away from the context root, we can use the neighbor context without risk. For example, the probability $\hat{p}(\text{hard}|\text{work}, \text{mum}, \text{too})$ should be a mixture of $p(\text{hard}|\text{work}, \text{mum})$, $p(\text{hard}|\text{work}, \text{too})$, $p(\text{hard}|\text{too})$ and so on. The probability $p(\text{hard}|\text{work}, \text{too})$ is not considered in our current model. By modeling this property, our model will perform the same as HPYLM and VPYLM for clean text.

The future work also includes the fundamental modification of our model and the application to word/phoneme segmentation problem of actual utterances. Since hesitation is often a part of phoneme sequence of a word, it also depends on the currently or previously uttered word. A new generative process modeling above properties is required to deal with conversational utterances.

5 Conclusion

We proposed a segmental context mixture model to reduce the prediction error caused by noises, fillers, and hesitations in utterances, which rarely appear in the training text. Although hesitations or fillers will appear for speech transcriptions, they vary according to a speaker and topic. The model's probability consists of a mixture of conditioned probabilities of part of context words. The generative process can be expressed by combining the stick-breaking process and the process used in the variable order Pitman-Yor Language model (VPYLM). Experimental results revealed our model had better perplexity for noisy text than hierarchical PYLM, VPYLM, Witten-Bell and Kneser-ney smoothing.

The remaining challenges include building a more specific process for fillers and mis-recognitions for the language model and evaluation using text obtained by automatic speech recognition. For recognized text, we can use the re-scoring technique to apply our model. As mentioned in the discussion, our model can be improved by considering the movement property of filler and hesitations. Since our further interest is to acquire lexicons and meaning from conversational speech signals through spoken dialogue, the impact of our model on word segmentation should be evaluated.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 15K16051.

References

Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

Christopher Michael Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer-Verlag New York.

Micha Elsner, Sharon Goldwater, Naomi Feldman, and Frank Wood. 2013. A joint learning model of word segmentation, lexical acquisition, and phonetic variability. In *In proc. of the Conference on Empirical Methods on Natural Language Processing*, pages 42–54.

Joshua T. Goodman. 2001. A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434.

David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. 2006. A closer look at skip-gram modelling. In *Proc. of the 5th international Conference on Language Resources and Evaluation*, pages 1222–1225.

Herman Kamper, Aren Jansen, and Sharon Goldwater. 2016. Unsupervised word segmentation and lexicon discovery using acoustic word embeddings. *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, 24(4):669–679.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proc. of International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. of Interspeech*, pages 1045–1048.

Daichi Mochihashi and Eiichiro Sumita. 2007. The infinite markov model. In *Advances in Neural Information Processing Systems*, pages 1017–1024.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested pitman-yor language modeling. In *Proc. of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 100–108.

Rene Pickhardt, Thomas Gottron, Martin Körner, Steffen Staab, Paul Georg Wagner, and Till Speicher. 2014. A generalized language model as the combination of skipped n-grams and modified kneser ney smoothing. In *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1145–1154.

Jayaram Sethuraman. 1994. A constructive definition of dirichlet priors. *Statistica Sinica*, pages 639–650.

Tadahiro Taniguchi, Shogo Nagasaka, and Ryo Nakashima. 2016. Nonparametric bayesian double articulation analyzer for direct language acquisition from continuous speech signals. *IEEE Transactions on Cognitive and Developmental Systems*, 8(3):171–185.

Yee Whey Teh. 2006. A bayesian interpretation of interpolated kneser-ney. *Technical Report TRA2/06, School of Computing, NUS*.

Ian H Witten and Timothy C Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Trans. on Information Theory*, 37(4):1085–1094.

Xiaoyi Wu and Yuji Matsumoto. 2015. An improved hierarchical word sequence language model using directional information. In *Proc. of The 29th Pacific Asia Conference on Language, Information and Computation*, pages 449–454.

Label Embedding for Zero-shot Fine-grained Named Entity Typing

Yukun Ma[1,2], **Erik Cambria**[1,2], **Sa Gao**[1,2]
[1]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[2]Rolls-Royce@NTU Corporate Lab, Nanyang Technological University, Singapore
mayu0010@e.ntu.edu.sg, cambria@ntu.edu.sg, gaos0011@e.ntu.edu.sg

Abstract

Named entity typing is the task of detecting the types of a named entity in context. For instance, given "Eric is giving a presentation", our goal is to infer that 'Eric' is a speaker or a presenter and a person. Existing approaches to named entity typing cannot work with a growing type set and fails to recognize entity mentions of unseen types. In this paper, we present a label embedding method that incorporates prototypical and hierarchical information to learn pre-trained label embeddings. In addition, we adapt a zero-shot framework that can predict both seen and previously unseen entity types. We perform evaluation on three benchmark datasets with two settings: 1) few-shots recognition where all types are covered by the training set; and 2) zero-shot recognition where fine-grained types are assumed absent from training set. Results show that prior knowledge encoded using our label embedding methods can significantly boost the performance of classification for both cases.

1 Introduction

Named entity typing (NET) is the task of inferring types of named entity mentions in text. NET is a useful pre-processing step for many natural language processing (NLP) tasks, e.g., auto-categorization and sentiment analysis. Named entity linking, for instance, can use NET to refine entity candidates of a given mention (Ling and Weld, 2012). Besides, NET is capable of supporting applications based on a deeper understanding of natural language, e.g., knowledge completion (Dong et al., 2014) and question answering (Lin et al., 2012; Fader et al., 2014). Standard NET approaches or sometime known as named entity recognition (Chinchor and Robinson, 1997; Tjong Kim Sang and De Meulder, 2003; Doddington et al., 2004) are concerned with coarse-grained types (e.g, person, location, organization) that are flat in structure. In comparison, fine-grained named entity typing (FNET) (Ling and Weld, 2012), which has been studied as an extension of standard NET task, uses a tree-structured taxonomy including not only coarse-grained types but also fine-grained types of named entities. For instance, given "[*Intel*] *said that over the past decade*", standard NET only classifies *Intel* as *organization*, whereas FNET further classifies it as *organization/corporation*.

FNET is faced with two major challenges: growing type set and label noises. Since the type hierarchy of entities is typically built from knowledge bases such as DBpedia, which is regularly updated with new types (especially fine-grained types) and entities, it is natural to assume that the type hierarchy is growing rather than fixed over time. However, current FNET systems are impeded from handling a growing type set for that information learned from training set cannot be transferred to unseen types. Another problem with FNET is that the weakly supervised tagging process used for automatically generating labeled data inevitably introduces label noises. Current solutions rely on heuristic rules (Gillick et al., 2014) or embedding method (Ren et al., 2016) to remove noises prior to training the multi-label classifier. In order to address these two problems at the same time, we propose a simple yet effective method for learning prototype-driven label embeddings that works for both seen and unseen types and is robust to the label

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,
pages 171–180, Osaka, Japan, December 11-17 2016.

noises. Another contribution of this work is that we combine prototypical and hierarchical information for learning label embeddings.

The remainder of this paper is organized as follows: Section 2 proposes a survey of prior works related to FNET; Section 3 introduces the embedding-based FNET method and its zero-shot extension; Section 4 describes our label embedding method; Section 5 illustrates experiments and analysis for both few-shot and zero-shot settings; finally, Section 6 concludes the paper and discusses future work.

2 Related Work

There is little related work specifically on zero-shot FNET but several research lines are considered related to this work: fine-grained named entity recognition, prototype-driven learning, and multi-label classification models based on embeddings. As FNET works with a much larger type set as compared with standard NET, it becomes difficult to have a sufficient training set for every type when relying on manual annotation. Instead, training data can be automatically generated from semi-structural data such as Wikipedia pages (Ling and Weld, 2012). Consequently, a single supervised classifier (Ling and Weld, 2012; Yogatama et al., 2015) or a series of classifiers (Yosef et al., 2012) are trained on this auto-annotated training set. This auto-annotating practice has been followed by later works on FNET (Yosef et al., 2012; Yogatama et al., 2015; Ren et al., 2016). However, since the automated tagging process is not accurate all the time, a number of noisy labels are then propagated to supervised training and affect the performance negatively.

The starting point of this work is the embedding method, WSABIE (Weston et al., 2011), adapted by (Yogatama et al., 2015) to FNET. WSABIE maps input features and labels to a joint space, where information is shared among correlated labels. However, the joint embedding method still suffers from label noises which have negative impacts on the learning of joint embeddings. In addition, since the labeled training set is the only source used for learning label embeddings, WSABIE cannot learn label embeddings for unseen types. DeViSE (Frome et al., 2013) is proposed for annotating image with words or phrases. As in such case, labels are natural words, e.g., fruit, that can be found in textual data, Skip-gram word embeddings (Mikolov et al., 2013) learned from a large text corpus are directly used for representing labels. In addition to label itself, prior works have also tried to learn label embeddings from side information such as attributes (Akata et al., 2013), manually-written descriptions (Larochelle et al., 2008), taxonomy of types (Weinberger and Chapelle, 2009; Akata et al., 2013; Akata et al., 2015), and so on.

Another related line of research is prototype-driven learning. (Haghighi and Klein, 2006) presented a sequence labeling model using prototypes as features and has tested the model on NLP tasks such as part-of-speech (POS) tagging. Prototype-based features (Guo et al., 2014) are then adapted for coarse-grained named entity recognition task. Even though we select prototypes in the same way as (Guo et al., 2014), we use prototypes in a very different manner: we consider prototypes as the basis for representing labels, whereas prototypes are mainly used as additional features in prior works (Haghighi and Klein, 2006; Guo et al., 2014). In other words, prototypes are previously used on the input side, while we use them on the label side.

3 Embedding Methods for FNET

In this section, we introduce the embedding method for FNET proposed by (Yogatama et al., 2015) and its extension to zero-shot entity typing.

3.1 Joint Embedding Model

Each entity mention m is represented as a feature vector $x \in \mathbb{R}^V$; and each label $y \in Y$ is a one-hot vector, where Y is the set of true labels associated with x. $\bar{Y}$ denotes the set of false labels of the given entity mention. The bi-linear scoring function for a given pair of x and y is defined as follows:

$$f(x, y, W) = x'Wy,$$

where $W \in \mathbb{R}^{M \times N}$ matrix with M the dimension of feature vector and N the number of types.

Instead of using a single compatibility matrix, WSABIE (Weston et al., 2011; Yogatama et al., 2015) considers an alternate low-rank decomposition of W, i.e., $W = A^\top B$, in order to reduce the number of parameters. WSABIE rewrites the scoring function as

$$f(x, y, A, B) = \phi(x, A) \cdot \theta(y, B) = x' A^\top B y,$$

which maps feature vector x and label vector y to a joint space. Note that it actually defines feature embeddings and label embeddings as

$$\phi(x, A) : x \rightarrow Ax,$$
$$\theta(y, B) : y \rightarrow By,$$

where $A \in \mathbb{R}^{D \times M}$ and $B \in \mathbb{R}^{D \times N}$ are matrices corresponding to lookup tables of feature embeddings and label embeddings, respectively. The embedding matrices A and B are the only parameters to be learned from supervised training process. In (Weston et al., 2011), the learning is formulated as a learning-to-rank problem using weighted approximate-rank pairwise (WARP) loss,

$$\sum_{y \in Y} \sum_{y' \in \bar{Y}} L(rank(x, y)) \max(1 - f(x, y, A, B) + f(x, y', A, B), 0),$$

where the ranking function $rank(x, y) = \sum_{y' \in \bar{Y}} \mathbb{I}(1 + f(x, y', A, B) > f(x, y, A, B))$, and $L(k) = \sum_{i=1}^{k} \frac{1}{i}$ which maps the ranking to a floating-point weight.

3.2 Zero-shot FNET Extension

A zero-shot extension of above WSABIE method can be done by introducing pre-trained label embeddings into the framework. The pre-trained label embeddings are learned from additional resources, e.g., text corpora, to encode semantic relation and dependency between labels. Similar to (Akata et al., 2013), we use two different methods for incorporating pre-trained label embeddings. The first one is to fully trust pre-trained label embeddings. Namely, we fix B as the pre-trained $\tilde{B}$ and only learn A in an iterative process. The second method is to use pre-trained label embedding as prior knowledge while adjusting both A and B according to the labeled data, i.e., adding a regularizer to the WARP loss function,

$$\sum_{y \in Y} \sum_{y' \in \bar{Y}} L(rank(x, y)) \max(1 - f(x, y, A, B) + f(x, y', A, B), 0) + \lambda ||B - \tilde{B}||_F^2,$$

where $|| \cdot ||_F$ is the Frobenius norm, and λ is the trade-off parameter.

4 Methods

4.1 Prototype-driven Label Embedding

Joint embedding methods such as WSABIE learn label embeddings from the whole training set including noisy labeled instances resulting from weak supervision. It is inevitable that the resulting label embeddings are affected by noisy labels and fail to accurately capture the semantic correlation between types. Another issue is that zero-shot frameworks such as DeViSE are not directly applicable to FNET as conceptually complex types, e.g, *GPE* (Geo-political Entity) cannot be simply mapped to a single natural word or phrase.

To address this issue, we propose a simple yet effective solution which is referred to as prototype-driven label embedding (ProtoLE), and henceforth we use $\tilde{B}^P$ to denote the label embedding matrix learned by ProtoLE. The first step is to learn a set of prototypes for each type in the type set. ProtoLE does not fully rely on training data to generate label embeddings. Instead, it selects a subset of entity mentions as the prototypes of each type. These prototypes are less ambiguous and noisy compared to the rest of the full set.

Even though it is already far less labor-intensive to manually select prototypes than annotating entity mentions one by one, we consider an alternative automated process using Normalized Point-wise Mutual Information (NPMI) as the particular criterion for prototype selection. The NPMI between a label and an entity mention is computed as:

$$\texttt{NPMI}(y, m) = \frac{\texttt{PMI}(y, m)}{-\ln p(y, m)},$$

where $\texttt{NPMI}(\cdot, \cdot)$ is the point-wise mutual information computed as follows:

$$\texttt{PMI}(y, m) = \log \frac{p(y, m)}{p(y)p(m)},$$

where $p(y)$, $p(m)$ and $p(y, m)$ are the probability of entity mention m, label y and their joint probability. For each label, NPMI is computed for all the entity mentions and only a list of top k mentions are selected as prototypes. Note that NPMI is not applicable to unseen labels. In such case, it is necessary to combine manual selection and NPMI.

Word embeddings methods such as Skip-gram model (Mikolov et al., 2013) are shown capable of learning distributional semantics of words from unlabeled text corpora. To further avoid affected by label noises, we use pre-trained word embeddings as the source to compute prototype-driven label embeddings. For each label y_i, we compute its label embedding as the average of pre-trained word embeddings of the head words of prototypes, i.e.,

$$\tilde{B}_i^P = \frac{1}{k} \sum_{j=1}^{k} v_{m_{ik}},$$

where $v_{m_{ik}}$ denotes the word embedding of kth word in the prototype list of label y_i. In the case of using phrase embeddings, the full strings of multi-word prototypes could be used directly.

4.2 Hierarchical Label Embedding

Another side information that is available for generating label embeddings is the label hierarchy. We adapt the Hierarchical Label Embeddings (HLE) (Akata et al., 2013) to FNET task. Unlike (Akata et al., 2013), which uses the WordNet hierarchy, FNET systems typically have direct access to predefined tree hierarchy of type set. We denote the label embedding matrix resulting from label hierarchy as $\tilde{B}^H$. Each row in $\tilde{B}^H$ corresponds to a binary label embedding and has a dimension equal to the size of label set. For each label, the sets $\tilde{B}_{ij}^H$ to 1 when y_j is the parent of y_i or $i = j$, and 0 to the remainder,

$$\tilde{B}_{ij}^H = \begin{cases} 1 & \text{if } i = j \text{ or } y_j \in Parent(y_i) \\ 0 & \text{otherwise} \end{cases}.$$

HLE explicitly encodes the hierarchical dependency between labels by scoring a type y_i given m using not only y_i but also its parent type $Parent(y_i)$. The underlying intuition is that recognition of a child type should be also based on the recognition of its parent.

4.3 Prototype-driven Hierarchical Label Embedding

One shortcoming of HLE is that it is too sparse. A natural solution is combining HLE with ProtoLE, which is denoted as Proto-HLE. Since $\tilde{B}^H \in \mathbb{R}^{N \times N}$ and $\tilde{B}^P \in \mathbb{R}^{D \times N}$, the combined embedding matrix $\tilde{B}^{HP}$ can be obtained by simply multiplying $\tilde{B}^H$ by $\tilde{B}^P$, i.e.,

$$\tilde{B}^{HP} = \tilde{B}^P \tilde{B}^{H\top}.$$

Note that $\tilde{B}^{HP}$ has the same shape as $\tilde{B}^P$, and it is actually representing the child label as a linear combination of the ProtoLE vectors of its parent and itself.

4.4 Type Inference

Having computed the scoring function for each label given a feature vector of the mention, we conduct type inference to refine the top k type candidates. In the setting of few-shots FNET, k is typically set to the maximum depth of type hierarchy, while different values for k may be used for a better prediction of unseen labels in zero-shot typing. For top k type candidates, we greedily remove the labels that conflict with others. However, unlike (Yogatama et al., 2015), we use a relative threshold t to decide whether the selected type should remain in the final results, which is more consistent with the margin-infused objective function than a global threshold. Namely, a type candidate will be passed to type inference only if the difference of score from the 1-best is less than a threshold.

5 Experiments

5.1 Experiment Setup

Our method uses feature templates similar to what have been used by state-of-the-art FNET methods (Ling and Weld, 2012; Gillick et al., 2014; Yogatama et al., 2015; Xiang Ren, 2015). Table 1 illustrates the full set of feature templates used in this work. We evaluate the performance of our methods on three benchmark datasets that have been used for the FNET task: BBN dataset (Weischedel and Brunstein, 2005), OntoNotes dataset (Weischedel et al., 2011) and Wikipedia dataset (Ling and Weld, 2012). (Xiang Ren, 2015) has pre-processed the training sets of BBN and OntoNotes using DBpedia Spotlight[1]. Entity mentions in the training set are automatically linked to a named entity in Freebase and assigned with the Freebase types of induced named entity. As shown in Table 2, BBN dataset contains 2.3K news articles of Wall Street Journal, which includes 109K entity mentions belonging to 47 types. OntoNotes contains 13.1K news articles and 223.3K entity mentions belonging to 89 entity types. The size of Wikipedia dataset is much larger than the other two with 2.69M entity mentions of 113 types extracted from 780.5K Wikipedia articles. Each data set has a test set that is manually annotated for purpose of evaluation. To tune parameters such as the type inference threshold t and trade-off parameter λ, we randomly sample 10% instances from each testing set as the development sets and use the rest as evaluation sets.

Feature	Description	Example
Tokens	Unigram words in the mentions	"White", "House"
Head	Head word of the mention	"House"
Cluster	Brown Cluster IDs of the head word	"4_1111", .. ,"8_11111101"
POS Tag	POS tag of the mention	"NNP"
Character	Lower-cased character trigrams in the head word	"hou","ous","use"
Word Shape	The word shape of words in the mention	"Aa","Aa"
Context	Unigram/bigram words in context of the mention	"Bennett","the", "Bennett_the"
Dependency	Dependency relations involving the head word	"gov_nn_director"

Table 1: Features extracted for context "*William Bennet, the [White House] drug-policy director....*"

Dataset		Types	Documents	Sentences	Mentions
BBN	train	47	2.3K	48.8K	109K
	test		459	6.4K	13.8K
OntoNotes	train	89	13.1K	147.7K	223.3K
	test		76	1.3K	9.6K
Wikipedia	train	113	780.5K	1.15M	2.69M
	test		-	434	563

Table 2: Statistics of datasets

[1] http://github.com/dbpedia-spotlight/dbpedia-spotlight

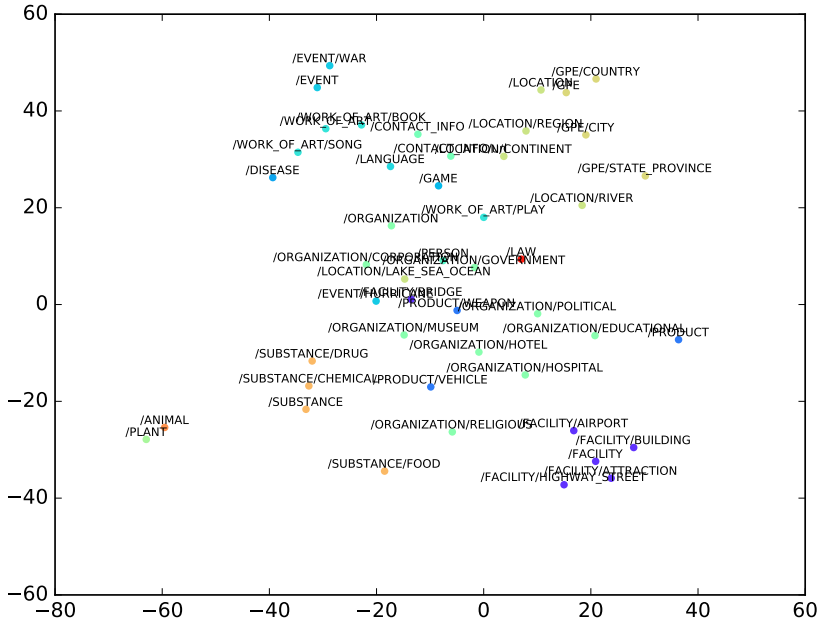


Figure 1: t-SNE visualization of the prototype-driven label embeddings for BBN dataset

Following prior works (Ling and Weld, 2012), we evaluate our methods and baseline systems using both loose and strict metrics, i.e., Macro-F1, Micro-F1, and strict Accuracy (Acc.). Given the evaluation set D, we denote Y_m as the ground truth types for entity mention $m \in D$ and $\widehat{Y}_m$ as the predicted labels. Strict accuracy (Acc) can be computed as:Acc= $\frac{1}{D}\sum_{m\in D}\sigma(Y_m = \widehat{Y}_m)$, where $\sigma(\cdot)$ is an indicator function. Macro-F1 is based on Macro-Precision (Ma-P) and Micro-Recall (Ma-R), where Ma-P = $\frac{1}{|D|}\sum_{m\in D}\frac{|Y_m\cap\widehat{Y}_m|}{Y_m}$, and Ma-R= $\frac{1}{|D|}\sum_{m\in D}\frac{|Y_m\cap\widehat{Y}_m|}{Y_m}$. And Micro-F1 is based on Micro-Precision (Mi-P) and Micro-Recall (Mi-R), where Mi-P= $\frac{\sum_{m\in D}|Y_m\cap\widehat{Y}_m|}{\sum_{m\in D}\widehat{Y}_m}$, and Mi-R= $\frac{\sum_{m\in D}|Y_m\cap\widehat{Y}_m|}{\sum_{m\in D}Y_m}$.

5.2 Generating ProtoLE

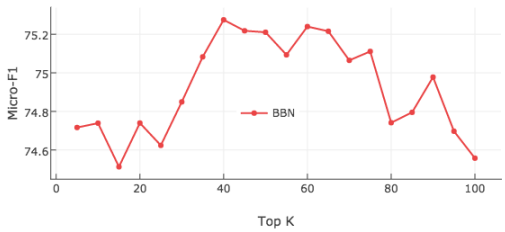
Our ProtoLE embeddings use Continuous-Bag-of-Words (CBOW) word embedding model (Mikolov et al., 2013) trained on Wikipedia dump using a window of 2 words to both directions. We use 300 dimensions for all embedding methods except HLE. Table 3 illustrates examples of prototypes learned for types in BBN dataset. It can be observed that most of the top ranked mentions are correctly linked to types, even though there are still some noises, e.g., north_american for `/LOCATION/CONTINENT`. It also shows that prototypes of related types such as `/LOCATION` and `/GPE` are also semantically related. Figure1 visualizes the prototype-driven label embeddings for BBN dataset using -Distributed Stochastic Neighbor Embedding (t-SNE)(Maaten and Hinton, 2008). It can be easily observed that semantic related types are close to each other in the new space, which proves that prototype-driven label embeddings can capture the semantic correlation between labels.

Figure 2 shows the Micro-F1 score of FNET with regard to the number of PMI prototypes used by ProtoLE. It shows that the Micro-F1 score does not change significantly on BBN and Wikipedia dataset, whereas using fewer prototypes per type (≤ 40) results in a drop of Micro-F1. Since the definitions of several types, especially the coarse-grained types, are actually very general, it may introduce bias into the label embeddings if using too few prototypes. We use $K = 60$ for all our experiments for that it achieves decent performance on all three datasets. Our pre-trained label embeddings and manually-selected prototypes (zero-shot typing) are available for download[2].

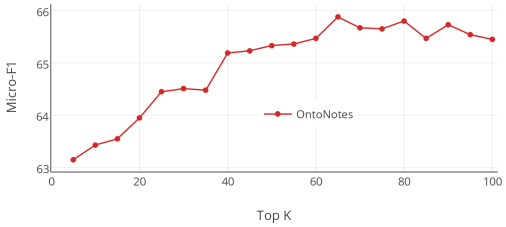
[2] `http://github.com/fnet-coling/ner-zero/tree/master/label_embedding`

Type	Prototypes
/LOCATION	areas connaught earth lane brooklyn
/LOCATION/CONTINENT	north_america europe africa north_american asia
/LOCATION/LAKE_SEA_OCEAN	big_bear lake_erie champ lake_geneva fujisawa
/LOCATION/RIVER	hudson thompson mississippi_river james_river tana
/GPE	soviet edisto canada china france
/GPE/STATE_PROVINCE	california texas ohio arizona jersey

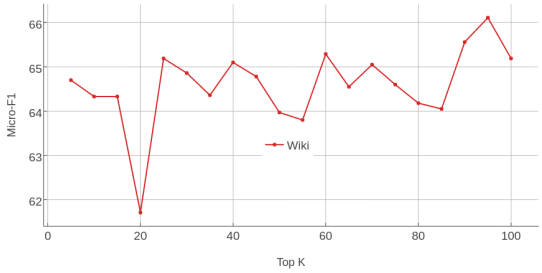
Table 3: Example prototypes learned by PMI for types in BBN dataset



(a) BNN Dataset



(b) OntoNotes Dataset



(c) OntoNotes Dataset

Figure 2: Performance changes on the development set with regard to the sizes of prototype list

5.3 Few-shots Fine-grained Entity Typing

In this section, we compare performances of FNET methods in the setting of few-shots FNET where the training set covers all types. Methods compared in this section are trained using the entire type set. We use evaluation metrics for our experiments: macro-F1, micro-F1 and accuracy. As in section 3.2, we train our label embeddings in two different ways: 1) non-adaptive training where label embeddings are fixed during training; and 2) adaptive training where label embeddings are also updated. Table 4 shows the comparison with state-of-the-art FNET methods: FIGER(Ling and Weld, 2012), HYENA(Yosef et al., 2012) and WSABIE (Yogatama et al., 2015). We make several findings from the results.

Firstly, embedding methods with WARP loss function consistently outperform non-embedding methods (i.e., FIGER and HYENA) on all three datasets. The performance gaps are huge for BBN and OntoNotes, where the best embedding method achieves 10%-20% absolute improvement over the best non-embedding method (FIGER). However, the gap is much smaller on Wikipedia dataset whose size is significantly larger than the other two.

Secondly, non-adaptive embedding methods always outperform their adaptive versions except HLE on Wikipedia dataset. Performance of adaptive label embeddings are all close to WSABIE, which suggests that adaptive label embeddings might suffer from same label noise problem as WSABIE does.

Thirdly, our ProtoLE and its combination with HLE consistently outperform both non-embedding and embedding baselines. Using the prototype information and non-adaptive framework results in absolute 3%-5% improvement with both loose and strict evaluation metrics. Non-adaptive HLE performs poorer than other embedding methods, which is most likely due to its sparsity in representing labels. However, Proto-HLE performs very close to ProtoLE on BBN and Wiki, while it improves all three measures by another absolute ≈2.5% on OntoNotes .

Method	Adapt	BBN			OntoNotes			Wiki		
		Ma-F1	Mi-F1	Acc.	Ma-F1	Mi-F1	Acc.	Ma-F1	Mi-F1	Acc.
FIGER	NA	67.28	60.70	46.92	58.77	52.37	38.01	**68.28**	64.71	47.37
HYENA	NA	51.38	52.85	45.01	47.65	43.97	26.56	45.51	43.80	30.67
WSABIE	NA	71.28	72.08	66.22	62.03	55.83	43.61	67.97	64.49	48.28
HLE	Y	70.84	71.61	65.74	61.54	49.16	43.25	67.09	65.65	47.01
	N	68.86	70.00	63.32	59.52	54.01	41.60	65.29	62.53	45.19
ProtoLE	Y	72.67	73.54	67.58	60.90	54.68	42.82	66.96	65.78	49.18
	N	**75.78**	**76.50**	**70.43**	65.91	59.08	46.94	68.06	**66.53**	**53.54**
Proto-HLE	Y	71.97	72.89	67.05	62.71	56.64	44.81	67.85	65.74	50.27
	N	74.54	74.38	69.46	**68.23**	**61.27**	**49.30**	66.61	65.29	50.45

Table 4: Performance of FNET in a few-shots learning on 3 benchmark datasets

5.4 Zero-shot Fine-grained Entity Typing

In this section, we evaluate our method's capability recognizing mentions of unseen fine-grained types. We assume that the training set contains only coarse-grained types (i.e., Level-1), and Level-2 types are unseen types to be removed from the training set. Table 5 shows the Micro-Precision for Level-1 and Level-2 types using top k type candidates for type inference. NPMI is computed for Level-1 types. We manually build prototype lists for unseen types by choosing from a randomly sampled list of entity mentions. Level-3 types are ignored for OntoNotes as Level-3 types never show in top-10 list produced by all methods. As the prediction for coarse-grained types are the same with regard to k, we only list the results using $k = 3$.

One interesting finding on all three datasets is that combining hierarchical and prototypical information results in better classification of coarse-grained types. It suggests that embeddings of unseen fine-grained types contains information complementary to the embeddings of coarse-grained types. Since HLE actually produces random prediction on Level-2 types due to its sparse representation, HLE perform poorly on Level-2 types.

Data Set	Method	Micro-Precision @k Level 1	Micro-Precision @k Level 2		
		3	3	5	10
BBN	ProtoLE	76.71	**42.95**	**36.61**	**42.34**
	HLE	70.44	13.08	13.16	12.82
	Proto-HLE	**76.89**	42.35	35.18	30.16
OntoNotes	ProtoLE	73.26	**21.01**	**13.72**	**12.22**
	HLE	66.96	7.13	6.14	6.23
	Proto-HLE	**76.33**	7.09	11.43	9.91
Wiki	ProtoLE	65.52	12.50	21.28	17.91
	HLE	65.13	0.00	8.82	8.99
	Proto-HLE	**67.41**	**20.01**	**31.25**	**24.24**

Table 5: Performance of zero-shot entity typing

ProtoLE outperforms HLE by 100%-300% in terms of Micro-Precision. However, again the combination of prototypes and hierarchy achieves similar or better results than ProtoLE on BBN and Wikipedia dataset. The drop of precision of Proto-HLE on OntoNotes is likely due to a different nature of annotation. It is more prevalent in test set of OntoNotes that one entity mention is annotated with multiple Level-1 types, and the presence of fine-grained types are less constrained by the label hierarchy. In such case, hierarchical constrains enforced by Proto-HLE might have negative impacts on type inference.

6 Conclusion

In this paper, we presented a prototype-driven label embedding method for fine-grained named entity typing (FNET). It shows that our method outperforms state-of-the-art embedding-based FNET methods in both few-shots and zero-shots settings. It also shows that combining prototype-driven label embeddings and type hierarchy can improve the prediction on coarse-grained types. In the near future, we plan to integrate our method with other types of side information such as definition sentences as well as label noise reduction framework (Ren et al., 2016) to further boost the robustness of FNET.

Acknowledgements

This work was conducted within the Rolls-Royce@NTU Corp Lab with support from the National Research Foundation Singapore under the Corp Lab@University Scheme.

References

Zeynep Akata, Florent Perronnin, Zad Harchaoui, and Cordelia Schmid. 2013. Label-embedding for attribute-based classification. In *Proceedings of CVPR*, pages 819–826.

Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. 2015. Evaluation of output embeddings for fine-grained image classification. In *Proceedings of CVPR*, pages 2927–2936.

Nancy Chinchor and Patricia Robinson. 1997. Muc-7 named entity task definition. In *Proceedings of the 7th MUC*, page 29.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*, page 1.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of SIGKDD*, pages 601–610.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of SIGKDD*, pages 1156–1165.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. 2013. Devise: A deep visual-semantic embedding model. In *NIPS*, pages 2121–2129.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *Proc. of EMNLP*, pages 110–120.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of HLT-NAACL*, pages 320–327. Association for Computational Linguistics.

Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. 2008. Zero-data learning of new tasks. In *AAAI*, page 3.

Thomas Lin, Oren Etzioni, et al. 2012. No noun phrase left behind: detecting and typing unlinkable entities. In *Proceedings of EMNLP*, pages 893–903.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *In Proc. of the 26th AAAI Conference on Artificial Intelligence*.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS'13*, pages 3111–3119.

Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016. Label noise reduction in entity typing by heterogeneous partial-label embedding. *arXiv preprint arXiv:1602.05307*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 142–147.

Kilian Q Weinberger and Olivier Chapelle. 2009. Large margin taxonomy embedding for document categorization. In *Advances in Neural Information Processing Systems*, pages 1737–1744.

Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. Technical report, Linguistic Data Consortium, Philadelphia.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2011. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of IJCAI'11*.

Chi Wang Xiang Ren, Ahmed El-Kishky. 2015. Clustype: Effective entity recognition and typing by relation phrase-based clustering. In *Proceedings of SIGKDD'15*.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of ACL'15*, pages 291–296.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: Hierarchical type classification for entity names. In *Proceedings of COLING 2012: Posters*, pages 1361–1370.

The Role of Context in Neural Morphological Disambiguation

Qinlan Shen, Daniel Clothiaux, Emily Tagtow, Patrick Littell, Chris Dyer
Carnegie Mellon University
{qinlans, dclothia, etagtow, plittell, cdyer}@cs.cmu.edu

Abstract

Languages with rich morphology often introduce sparsity in language processing tasks. While morphological analyzers can reduce this sparsity by providing morpheme-level analyses for words, they will often introduce ambiguity by returning multiple analyses for the same surface form. The problem of disambiguating between these morphological parses is further complicated by the fact that a correct parse for a word is not only dependent on the surface form but also on other words in its context. In this paper, we present a language-agnostic approach to morphological disambiguation. We address the problem of using context in morphological disambiguation by presenting several LSTM-based neural architectures that encode long-range surface-level and analysis-level contextual dependencies. We applied our approach to Turkish, Russian, and Arabic to compare effectiveness across languages, matching state-of-the-art in two of the three languages. Our results also demonstrate that while context plays a role in learning how to disambiguate, the type and amount of context needed varies between languages based on their morphological and syntactic properties.

1 Introduction

Morphologically rich languages introduce sparsity in language processing tasks, as different surface variants over the same root are often taken as independent entities. Using a morphological analyzer can decompose inflected words into known tags that encode syntactic and semantic information about the word. However, finding the correct morphological parse is a non-trivial task. Functionally different morphemes may have similar forms, and long strings of potentially ambiguous morphemes compound the problem of ambiguity. As a result, analyzers for morphologically complex languages often return several parses for the same surface word. Table 1, for example, shows the resulting candidate parses for the surface form "alın" returned by Oflazer's (1994) morphological analyzer for Turkish.

```
alın+Noun+A3sg+Pnon+Nom (forehead)
al+Adj^DB+Noun+Zero+A3sg+P2sg+Nom (your red)
al+Adj^DB+Noun+Zero+A3sg+Pnon+Gen (of red)
al+Verb+Pos+Imp+A2pl ((you) take)
al+Verb^DB+Verb+Pass+Pos+Imp+A2sg ((you) be taken)
alın+Verb+Pos+Imp+A2sg ((you) be offended)
```

Table 1: Possible morphological parses for surface form "alın"

The surrounding context of the word being disambiguated plays a major part in determining the role of a word in a sentence and thus its correct morphological parse. For example, the surface form "evi" in Turkish can be interpreted as either accusative or third-person singular possessive, as shown in Table 2 (Yildiz et al., 2016); this determination cannot reliably be made without further context. Moreover, the disambiguation of a word can in turn disambiguate other words; if we have determined that "evi" is

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 181–191, Osaka, Japan, December 11-17 2016.

accusative, we can infer that a transitive verb must follow and that other ambiguous forms in a sentence are not accusative.

Sentence and translation	Analysis of `evi`
Evi bulabildiniz mi? – Did you find the house?	`ev+Noun+3sg +Pnon+Acc`
Evi gerçekten güzelmiş. – His/Her house is really beautiful	`ev+Noun+3sg +P3sg+Nom`

Table 2: Possible interpretations for "`evi`" based on context

The problem of disambiguating over the candidate morphological parses generated by a morphological analyzer has been tackled in many languages and with different strategies. These systems primarily rely on methods for capturing the structure of a target word and its candidate tag sequences (Yuret and Türe, 2006; Habash and Rambow, 2005; Daybelge and Cicekli, 2007; Daoud, 2009) and/or the surrounding context of a target word (Hakkani-Tür et al., 2002; Smith et al., 2005; Sak et al., 2008; Lee et al., 2011) to choose the best candidate analysis. Despite the breadth of work on this problem, there is little work on disambiguation using neural network models. Based on previous work, it is not clear whether neural models are able to disambiguate only using surface forms or how context plays a role in a neural disambugation model. Models that disambiguate jointly over tokens incorporate more information about the surrounding context of a word, but models that make decisions at the word level are often simpler to train.

In this paper, we present a language-agnostic LSTM-based approach for morphological disambiguation that takes into account both the structure of a word, including its candidate tag sequences, and the surrounding context of a target word. We propose a neural architecture for generating vector embeddings of the candidate analyses of a target word using character-based LSTMs to generate representations for the stems and surface forms, as well as an LSTM over the tag sequences to embed analyses.

We then describe several model architectures that operate over these vector representations, differing according to the window of context used and whether the context tokens have themselves been disambiguated. Because our architecture relies on character- and tag-level embeddings, the models can be adapted into other languages and handle unknown words at test time. Experiments on Turkish, Russian, and Arabic show that different languages benefit from different types and windows of context.

2 Models

The problem of morphological disambiguation involves selecting among a list of possible parses returned by an analyzer. Given the output of a morphological analyzer for tokens in a sentence, we use several LSTM architectures to predict the correct analysis for a word based on its context. These architectures vary based on the amount and type of context taken into account when choosing the best analysis.

Our approach to disambiguation relies on two embeddings: vector representations for each possible analysis for a word (expressed as matrix $\mathbf{R}$), which encode the stem and all of the morpheme tags for each analysis, and a vector embedding $\mathbf{h}$ of the relevant context of the target word. Using these embeddings, we can define a compatibility function between the representation of each candidate analysis and the representation of the context by taking the product of $\mathbf{R}$ and $\mathbf{h}$. Taking the softmax of this compatibility function will give us a probability distribution over possible parses given the relevant context.

$$p(y_t = a|x) = \text{softmax}(\mathbf{R}_{x_t} \times \mathbf{h}_t) \quad (1)$$

We describe a general architecture for embedding the possible parses of a target word, as well as the different architectural and objective variants for different models of context.

2.1 Analysis Embeddings

Given a morphological analysis of the form

$$stem_i + tag_{i,1} + tag_{i,2} + ... + tag_{i,L}$$

where $stem_i = (stem_{i,1}, stem_{i,2}...stem_{i,K})$ is the K character long stem of the i-th parse and each $tag_{i,j}$ is the j-th tag in the i-th parse (containing L tags), we use a bidirectional character-based LSTM to embed the stem, and a separate bidirectional LSTM over tags to embed the morphemes. A bidirectional LSTM creates a representation g_x of an input sequence $x = (x_1, x_2, ..., x_T)$ by computing a forward sequence $\overrightarrow{g}$ and a reverse sequence $\overleftarrow{g}$ over the input sequence, concatenating the results of the two sequences, and applying a rectified linear unit (ReLU) activation

$$\overrightarrow{g}_t = f(x_t, \overrightarrow{g}_{t-1}) \tag{2}$$

$$\overleftarrow{g}_t = f(x_t, \overleftarrow{g}_{t+1}) \tag{3}$$

$$g = \text{ReLU}([\overrightarrow{g}_T, \overleftarrow{g}_0]) \tag{4}$$

where $f(x, y)$ is the output of an LSTM unit with inputs x and y.

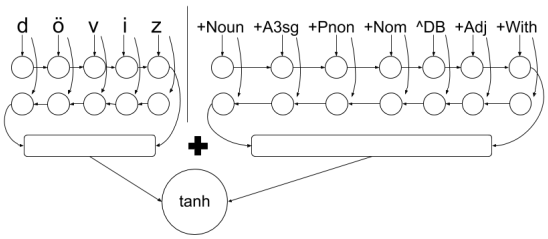


Figure 1: Neural architecture for analysis embedding

Thus, we create a representation of the stem by taking the characters of $(stem_{i,1}, stem_{i,2}, ..., stem_{i,K})$ as the input sequence for a "stem" LSTM and a representation of the tags by taking $(tag_{i,1}, tag_{i,2}, ..., tag_{i,L})$ as the input sequence for a separate "tag" LSTM. We then add these two representations and apply a $\tanh$ nonlinearity to create an embedding, r_i, for the i-th potential analysis a of the word (Figure 1).

$$r_i = \tanh(g_{stem_i} + g_{tag_i}) \tag{5}$$

These vectors of parse options r_i are concatenated to form matrix $\mathbf{R}$ for a word with N analyses, where each row in the matrix corresponds to a possible parse for a given word.

$$\mathbf{R} = [r_1; r_2; ...; r_N] \tag{6}$$

As a baseline model, we use matrix $\mathbf{R}$ without leveraging any of the surrounding context of the target word. To obtain the probability distribution of the possible parses from $\mathbf{R}$, we take the softmax of the product of $\mathbf{R}$ and a learned parameter vector $\mathbf{h}$. We then take the analysis a with the highest probability as the predicted analysis for the word.

$$p(y_t = a|x_t) = \text{softmax}(\mathbf{R}_{x_t} \times \mathbf{h}) \tag{7}$$

2.2 Surface Model

One method of integrating the context around a target word for morphological disambiguation is to leverage the surface forms of the words surrounding the target word. To capture the surface-level context of a target word, we first use another bidirectional character LSTM to embed the surface forms of each word x_i surrounding the current target word, creating a vector representation for each surrounding word.

We then use the embeddings for the relevant context words to the left of the target word as input to a left-to-right LSTM and the embeddings of the relevant context words to the right as input to a right-to-left LSTM over words to create vectors representing the left ($\overrightarrow{c}_t$) and right ($\overleftarrow{c}_t$) contexts of the target word at position t.

$$\overrightarrow{c}_t = f(x_t, \overrightarrow{c}_{t-1}) \tag{8}$$

$$\overleftarrow{c}_t = f(x_t, \overleftarrow{c}_{t+1}) \tag{9}$$

We add the left and right context vectors, then apply a tanh non-linearity to get $\mathbf{h}_t$ representing the surrounding surface context of the target word at position t.

$$\mathbf{h}_t = \tanh(\overrightarrow{c}_t + \overleftarrow{c}_t) \tag{10}$$

To combine this surface context representation with the possible parse embeddings matrix $\mathbf{R}$, we take a softmax over the product of $\mathbf{R}$ and $\mathbf{h}_t$ to return a probability distribution over possible parses given surface-level context.

$$p(y_t = a|x) = \text{softmax}(\mathbf{R}_{x_t} \times \mathbf{h}_t) \tag{11}$$

We compare two models that leverage the surface context of a word. The full context model uses all the words to the left of the target word and all the words to the right of the target word to build context vector $\mathbf{h}_t$ (Figure 2). The local context model uses a one word window to the left and right of its target word to build context vector $\mathbf{h}_t$.

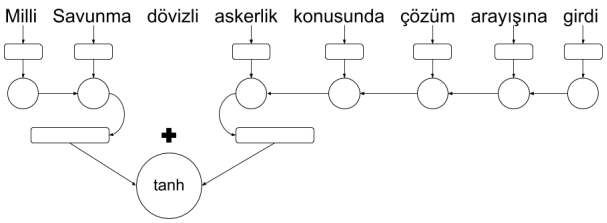


Figure 2: Neural architecture for full surface context embedding

2.3 Left-To-Right Analysis

A further question regarding the use of context in morphological disambiguation is whether the contextual tokens themselves need to be disambiguated, or whether their surface forms alone suffice to guide further disambiguation. For example, in a language like Russian with subject-verb agreement, having disambiguated a word as being third-person singular in the nominative case can help us predict that the verb should have third-person agreement.

To explore this question, we also consider models that take previously-disambiguated tokens as context. A simple way of leveraging information about the parses of surrounding words is to disambiguate the words in the sentence sequentially and use the previously selected parses to inform our decision at the current position. We use an LSTM over the selected parses of previously disambiguated words to create m_t, a representation of the decisions that were made up until position t. To build m_t, we take the representation of chosen parse $\tilde{r}^t$ from $\mathbf{R}_{x_t}$, and feed it into the LSTM encoding the sequence of previously chosen parses.

$$m_t = f(\tilde{r}_i^t, m_{t-1}) \tag{12}$$

Then, when choosing the next parse at position $t+1$, we also add m_t to the stem and morpheme representations $g_{stem_i}^{t+1}$ and $g_{tag_i}^{t+1}$ and apply a tanh nonlinearity when creating parse embeddings r_i^{t+1}.

$$r_i^{t+1} = \tanh(g_{stem_i}^{t+1} + g_{tag_i}^{t+1} + m_t) \tag{13}$$

$$\mathbf{R}_{x_{t+1}} = [r_1^{t+1}; r_2^{t+1}; ...; r_N^{t+1}] \tag{14}$$

Intuitively, the goal of this operation is to learn interactions between the previous parses and each candidate parse at the current position. We can then calculate a probability distribution over the candidates given both surface context and the previous parses in a manner similar to the process in Section 2.2.

$$p(y_t = a|x, y_1, y_2, ..., y_{t-1}) = \text{softmax}(\mathbf{R}_{x_t} \times \mathbf{h}_t) \tag{15}$$

When decoding, we use a greedy approach to select the parse to add to the previous parse LSTM. Thus, our objective is to predict the output sequence

$$\hat{y} = \text{argmax}_{\tilde{y} \in Y_X} \prod_{i=1}^{T} p(\tilde{y}_t|x, \tilde{y}_1, \tilde{y}_2, ..., \tilde{y}_{t-1}) \tag{16}$$

2.4 Conditional Random Field Joint Decoding

Locally normalized models suffer from the label bias problem (Andor et al., 2016), meaning that they have little to no ability to revise previous decisions. Conditional Random Fields (CRFs) have been shown to be effective at modeling sequences in tasks like part of speech tagging and named entity recognition (Lample et al., 2016). In this approach, we attempt to find the best sequence of parses that takes the entire sentence into account.

The CRF model is built on top of our full-context surface model, using the same process for embedding the parses and the surface context. Rather than taking the softmax over the combined representation of the surface context vector and the parse matrix, we use the product directly as a vector $\mathbf{u}_t$ of emission scores between each word x_t and its possible parses.

$$\mathbf{u}_t = \mathbf{R}_{x_t} \times \mathbf{h}_t \tag{17}$$

To model the transition scores between the j-th parse of word at position p and the i-th analysis of the previous word, we concatenate the embeddings of the two analyses and input the resulting vector to a feed-forward layer to give a transition score between the two parses, $\mathbf{v}(r_i^{t-1}, r_j^t)$.

$$\mathbf{v}(r_i^{t-1}, r_j^t) = \tanh(\mathbf{W}_{trans}[r_i^{t-1}, r_j^t]) \tag{18}$$

We can then produce a trellis of possible parses for each word and their transitions to use for picking the best parse sequence for a given sentence x of length N. We score sequences using the function

$$S(x, y) = \sum_{i=1}^{N} \mathbf{v}(y_{i-1}, y_i) + \mathbf{u}_i^{y_i} \tag{19}$$

To find the best parse sequence, we predict the output sequence

$$\hat{y} = \text{argmax}_{\tilde{y} \in Y_X} S(x, \tilde{y}) \tag{20}$$

We learned the parameters of the CRF as part of our neural architecture, then decode using the Viterbi algorithm to compute the best analysis sequence.

3 Experimental Setup

To demonstrate the language-agnostic nature of our disambiguation model, we applied our approach to Turkish, Russian, and Arabic, three morphologically-complex but typologically-distinct languages with well-established morphological analyzers (Oflazer, 1994; Korobov, 2015; Maamouri et al., 2010).

	Turkish		Russian		Arabic	
	Ambiguous	All	Ambiguous	All	Ambiguous	All
Training	332,457	783,209	830,055	1,815,414	253,058	318,821
Development	16,327	38,744	22,344	49,773	13,915	17,387
Annotated Test	379	946	16,340	50,083	14,231	18,021
Generated Test	18,022	42,000	-	-	-	-

Table 3: Token counts for data sets

3.1 Turkish

We used the same dataset for Turkish as in Sak et al. (2007) and Yildiz et al. (2016). The datasets were extracted from a corpus of approximately 1 million words of semi-automatically disambiguated Turkish (Yuret and Türe, 2006), which were split into training, development, and test sets. To limit the effect of noise from using semi-automatically disambiguated data in our training and evaluation, we also evaluated over the small test set of human disambiguated tokens in context that was provided with the data.

Preliminary analysis of the training set found that the average number of parses per word was 1.60 (std=1.304, max=24) for all tokens and 2.81 (std=1.208) for only ambiguous tokens. The average length of a Turkish sentence within our training set was 16 tokens (std=20.231).

3.2 Russian

Data for Russian was extracted from OpenCorpora, a freely available treebank for Russian (Bocharov et al., 2011). OpenCorpora provides a large corpus of approximately 1.7 million tokens, as well as a strict, manually disambiguated subset of approximately 50,000 tokens. Due to the relatively small amount of manually disambiguated data for training, we used the parse scores returned by the pymorphy2 morphological analyzer (Korobov, 2015) to semi-automatically disambiguate the full corpus, sampling the "gold" parse based on its parse score. One previous work in Russian morphology (Muzychka et al., 2014) used the SynTagRus dataset. However, this dataset used a different tagset than the pymorphy2 analyzer (Oflazer, 1994), which is based on OpenCorpora data.

The average number of parses per word in the training set was 3.10 (std=5.329, max=69) for all tokens and 5.81 (std=6.961) for only ambiguous tokens. The average length of a Russian sentence within our training set was 19.87 tokens (std=22.974).

We took the manually disambiguated data as our test set and randomly split the remaining data from the full corpus into a training set and development set.

3.3 Arabic

Data for Arabic was extracted from the Arabic Penn Treebank (ATB) part 3 version 3.2 (catalog number LDC2010T08) (Maamouri et al., 2004), a corpus containing approximately 370,000 annotated tokens. Analyses for the tokens were generated using the Buckwalter Morphological Analyzer Version 1.0 (Buckwalter, 2002), with human annotators selecting the correct parse. Previous approaches used different versions and subsets of the ATB. Here, we used a comparable subset to previous work, split into training, development, and test sets.

Each token had 9.11 possible parses (std=8.5932, max=86) on average, with each ambiguous token having 11.31 possible parses (std=8.301). The average length of a sentence was 26.13 (std=30.78) tokens.

3.4 Training

We considered each sentence to be a minibatch for training. The objective function used for training was the total cross-entropy loss between the selected parse and the correct parse for every token in the sentence. Stochastic gradient descent and backpropagation were used to adjust the parameters for our model. To prevent overfitting on the training set, we used validation-based early-stopping, saving the model parameters based on a periodic accuracy evaluation step over the development set. All LSTMs in

our models were trained with a single hidden layer. We used a hidden dimension size of 100 for the tag, stem, and surface form LSTMs and 200 for the context and previous parse LSTMs.

4 Discussion

4.1 Results

In all cases, using some contextual information improved accuracy over the no-context baseline, but there is noticeable variation between languages regarding what kinds of context were most valuable. We report accuracy over ambiguous tokens, as well as all tokens for sake of comparison with other systems.

	Ambiguous Tokens		All Tokens	
	Annotated Test	Generated Test	Annotated Test	Generated Test
No Context	88.65	90.72	95.45	96.08
Local Context	89.18	92.65	95.67	96.90
Full Context	**91.03**	**93.46**	**96.41**	**97.24**
Left-to-Right	90.50	93.42	96.19	97.23
CRF	90.24	93.06	96.09	97.07

Table 4: Disambiguation results for Turkish

Table 4 shows the performance of all models in Turkish on a hand-annotated test set, as well as the generated test set. We see that each of the models with some form of context is able to beat the no context baseline. The best-performing model on both the generated and annotated datasets is the full surface context model, followed closely by the left-to-right and CRF models. This shows that some form of long-range contextual information is useful for disambiguation for Turkish. Our full context model is comparable to the previous state of the art of 96.28% on annotated test and 96.80% on generated test established in Sak et al. (2007).

As we will see below, of the languages considered here, Turkish is the only one in which the full surface context model approached (and slightly exceeded) other models, which may stem from the typological character of Turkish. Turkish is a strongly head-final subject-object-verb (SOV) language, and so the best disambiguating evidence for a token often follows it. For example, in disambiguating "evi" (cf. Table 2), the best evidence for its case (nominative or accusative) will lie in whether a transitive or intransitive verb follows, and in a verb-final language the verb can follow at some distance from its arguments. Meanwhile, as seen in Section 3.1, Turkish is the least ambiguous of these languages (with a mean of 1.60 parses per word compared to 3.10 for Russian and 9.11 for Arabic), so surface context is not much less informative than disambiguated context. In other words, Turkish combines the greatest need for full context with higher relative informativeness of the surface context.

	Ambiguous Tokens	All Tokens
No Context	64.97	88.58
Local Context	71.56	90.72
Full Context	69.49	90.05
Left-to-Right	68.55	89.75
CRF	**72.78**	**91.13**

Table 5: Disambiguation results for Russian

The results for Russian are detailed in Table 5. Again, each of the contextual models perform much better than the no context baseline. However, we see an interesting pattern when comparing the four contextual models; unlike in Turkish, the CRF model performs the best. Many words in Russian are required to agree in gender, case, and number to their heads. We argue that a model that takes the parse information of neighboring words into account is more suited to capturing agreement than one that only uses the surface form information. The CRF model over parse sequences would be able to capture this

phenomenon, whereas the left-to-right decoding model, which also operates over neighboring parses, may fail due to making locally but not globally optimal decisions near the beginning of the sentence.

Muzychka et al. (2014) reported an accuracy of 91.06% in their CRF-based disambiguation model. In comparison, our neural CRF-based model achieves a similar accuracy of 91.13%.

	Ambiguous Tokens	All Tokens
No Context	72.22	78.06
Local Context	80.10	84.29
Full Context	86.45	88.95
Left-to-Right	**89.30**	**91.27**

Table 6: Disambiguation results for Arabic

We see a different pattern in Table 6 for Arabic. Like in Turkish and Russian, there is a large gap between the no context baseline and the local context model. The left-to-right model performs the best on Arabic. Habash et al. (2005) report an accuracy of 96.2% on all parses, while Smith et al. (2005) achieves an accuracy of 95.4% on different subsets of the ATB.

We can also note that surface-context models performed relatively poorly in Arabic. This is likely because of the greater ambiguity of Arabic (9.11 parses per word), which stems in part from the absence of vowels in Arabic writing. Manual inspection of the Arabic output suggested that the surface-context models were frequently making the same mistakes as the no-context model (Table 7), which did not tend to occur in the Turkish output. So, in contrast to the Turkish systems, in which having only surface context was as good as having disambiguated context, in the Arabic systems having only surface context was more similar to having no context at all, emphasizing the need for disambiguated context in more highly ambiguous languages.[1]

Input:	... dwl kvyrp HAlAt **SEbp** wnsbp Alnmw mtdnyp ...
Gold:	`SaEob_1+ADJ+NSUFF_FEM_SG+CASE_INDEF_ACC` ✓
No Context:	`SaEob_1+ADJ+NSUFF_FEM_SG+CASE_INDEF_GEN`
Low Context:	`SaEob_1+ADJ+NSUFF_FEM_SG+CASE_INDEF_NOM`
Full Context:	`SaEob_1+ADJ+NSUFF_FEM_SG+CASE_INDEF_GEN`
Left-to-Right:	`SaEob_1+ADJ+NSUFF_FEM_SG+CASE_INDEF_ACC` ✓

Input:	... bAlAntSAr kmA nHyy **AlAntfADp** fy flsTyn mlyn An ...
Gold:	{`inotifADap_1+DET+NOUN+NSUFF_FEM_SG+CASE_DEF_ACC` ✓
No Context:	{`inotifADap_1+DET+NOUN+NSUFF_FEM_SG+CASE_DEF_GEN`
Low Context:	{`inotifADap_2+DET+NOUN+NSUFF_FEM_SG+CASE_DEF_GEN`
Full Context:	{`inotifADap_1+DET+NOUN+NSUFF_FEM_SG+CASE_DEF_GEN`
Left-to-Right:	{`inotifADap_1+DET+NOUN+NSUFF_FEM_SG+CASE_DEF_ACC` ✓

Table 7: Example outputs for Arabic for targets "`SEbp`" and "`AlAntfADp`"

Within our experiments, context clearly does play a role in learning to disambiguate possible morphological analyses. The type and extent of the context needed, however, appears to vary based on features of the language, such as the word order or the degree of morphological ambiguity. This raises the possibility of a future system that can choose the best disambiguation context based on a language's typological properties.

[1]Following this hypothesis, we would expect the CRF architecture to perform well on Arabic. However, the much higher ratio of parses-per-word in Arabic made our neural CRF model computationally impractical.

4.2 Future Work

As mentioned in Section 2.4, a disadvantage of using greedy decoding with our left-to-right analysis model is the label-bias problem. Because decisions depend on the previously selected parses, an incorrect decision made early on can propagate through the sentence. Thus, a natural extension to the left-to-right model is to use beam search for decoding instead of our greedy approach. The advantage of using beam search is that it allows the model to explore multiple previous parse paths. This partially gives the model the ability to recover from making a decision that appears locally optimal in the short-term but leads to greater losses in the long-term.

Another model of context inspired by recent advances in machine translation (Bahdanau et al., 2015) and caption generation (Xu et al., 2015) is using an attentional mechanism to define the important context for a target word. For the full surface context model, rather than using separate LSTMs over the left and right contexts of the word, an attentional context model can learn to attend to different parts of the surrounding context of a target word based on its sentence position and candidate parse representations. This model will potentially allow us to capture longer-range surface dependencies without decay.

5 Related Work

A common early approach to morphological disambiguation is to rely purely on hand-crafted rules to select the correct parse out of a set of candidate analyses (Daybelge and Cicekli, 2007; Daoud, 2009). These rule-based methods primarily try to capture the relationship between a target word and its candidate analyses. Other approaches used a blend of rules and statistical methods. For example, Oflazer and Tür (1996) used a set of linguistically motivated rules and corpus-dependent statistics to either choose or delete possible parses in Turkish. The model also learned rules based on unambiguous words that appear in unambiguous contexts within the corpus. Similarly, Hajič et al. (2007) built upon the same type of deletion disambiguation, using the output from a choose-delete rule-based system to first reduce the number of possible parses before running a statistical part of speech tagger in Czech.

Yuret and Türe (2006) used the Greedy Prepend Algorithm to learn rules for Turkish disambiguation. For every tag in their dataset, a decision list of patterns was created to determine whether the tag is contained in the best analysis. A heuristic search that added new attributes to patterns already in the decision list was used to generate more candidate patterns for a tag.

Other statistical approaches to morphological disambiguation tried to directly model the context of a target word. Hakkani-Tür et al. (2002) proposed a statistical approach for Turkish disambiguation using a language model trained on disambiguated data. They trained trigram language models under different root and inflectional group independence assumptions and used the resulting language models to select the best candidate parses. Smith et al. (2005) used a conditional random field to learn to disambiguate over sentence by modeling local contexts. Sak et al. (2007) and (2008) used the perceptron algorithm on a set of 23 handcrafted features, including bigrams and trigrams at the word and inflectional group level.

There is relatively little work on designing neural models specifically for morphological disambiguation. Yildiz et al. (2016) proposed a convolutional architecture that creates a representation for the surface form of a word from a root and a set of morpheme features. They then trained their model to predict the correct analysis of a word given the ground truth annotations for the previous words within a window of the target. Their model was able to achieve an accuracy of 84% over ambiguous tokens in Turkish. In contrast, our proposed model uses long short-term memory (LSTM)-based architectures to capture longer range dependencies between a target word and its surrounding context. Additionally, we consider the context of a target word at both the surface-form and analysis level, providing additional information to our models.

6 Conclusion

In this paper, we present several LSTM-based neural network architecture for disambiguating morphological parses using varying amounts of surrounding context. We demonstrate using these architectures that the type and amount of context needed for disambiguation varies between languages based on the

linguistic features of a particular language. For a language like Turkish, where most of the morphological information is apparent based on the surrounding context, a model that uses the surface context can capture long-range information to make disambiguation decisions. Other languages, where the surface representation is less informative, such as Arabic, greatly benefit from using representations of the surrounding parse candidates in addition to the surface forms of the surrounding words. We also show that, while this system is language agnostic and can be applied to typologically different languages, the best architecture for a language depends on its morphological and syntactic properties.

Acknowledgements

This work was sponsored in part by the Defense Advanced Research Projects Agency Information Innovation Office (I2O) Program under the Low Resource Languages for Emergent Incidents (LORELEI) program issued by DARPA/I2O under Contract No. HR0011-15-C-0114. We would like to thank Graham Neubig, Yulia Tsvetkov, Michael Miller Yoder, and the anonymous reviewers for their helpful comments and feedback.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *ArXiv preprint*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceedings of the International Conference on Learning Representations*.

Victor Bocharov, Svetlana Bichineva, Dmitry Granovsky, Natalia Ostapuk, and Maria Stepanova. 2011. Quality assurance tools in the OpenCorpora project. In *Computational Linguistics and Intelligent Technology: Proceeding of the International Conference Dialog*, pages 10–17.

Tim Buckwalter. 2002. Buckwalter {Arabic} morphological analyzer version 1.0.

Daoud Daoud. 2009. Synchronized morphological and syntactic disambiguation for Arabic. *Advances in Computational Linguistics*, pages 73–86.

Turhan Daybelge and Ilyas Cicekli. 2007. A rule-based morphological disambiguator for Turkish. In *Proceedings of Recent Advances in Natural Language Processing*, pages 145–149.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 573–580.

Jan Hajič, Jan Votrubec, Pavel Krbec, Pavel Květoň, et al. 2007. The best of two worlds: Cooperation of statistical and rule-based taggers for czech. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing: Information Extraction and Enabling Technologies*, pages 67–74.

Dilek Z. Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2002. Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities*, (4):381–410.

Mikhail Korobov. 2015. Morphological Analyzer and Generator for Russian and Ukrainian Languages. pages 320–332.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*.

John Lee, Jason Naradowsky, and David A. Smith. 2011. A discriminative model for joint morphological disambiguation and dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 885–894.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR conference on Arabic language resources and tools*, pages 466–467.

Mohamed Maamouri, Dave Graff, Basma Bouziri, Sondos Krouna, Ann Bies, and Seth Kulick. 2010. Standard Arabic morphological analyzer (SAMA) version 3.1. *Linguistic Data Consortium, Catalog No.: LDC2010L01*.

S Muzychka, A Romanenko, and I Piontkovskaja. 2014. Conditional random field for morphological disambiguation in russian. In *Conference Dialog-2014, Bekasovo*.

Kemal Oflazer and Gokhan Tür. 1996. Combining hand-crafted rules and unsupervised learning in constraint-based morphological disambiguation. *Proceedings of the ACL-SIGDAT Conference on Empirical Methods in Natural Language Processing*.

Kemal Oflazer. 1994. Two-level description of Turkish morphology. *Literary and linguistic computing*, (2):137–148.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2007. Morphological disambiguation of Turkish text with perceptron algorithm. pages 107–118.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2008. Turkish language resources: Morphological parser, morphological disambiguator and web corpus. pages 417–427.

Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 475–482.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *International Conference on Machine Learning*.

Eray Yildiz, Caglar Tirkaz, H. Bahadir Sahin, Mustafa Tolga Eren, and Ozan Sonmez. 2016. A Morphology-Aware Network for Morphological Disambiguation. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Deniz Yuret and Ferhan Türe. 2006. Learning morphological disambiguation rules for Turkish. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 328–334.

Asynchronous Parallel Learning for Neural Networks and Structured Models with Dense Features

Xu Sun
∗MOE Key Laboratory of Computational Linguistics, Peking University
†School of Electronics Engineering and Computer Science, Peking University
xusun@pku.edu.cn

Abstract

Existing asynchronous parallel learning methods are only for the sparse feature models, and they face new challenges for the dense feature models like neural networks (e.g., LSTM, RNN). The problem for dense features is that asynchronous parallel learning brings gradient errors derived from overwrite actions. We show that gradient errors are very common and inevitable. Nevertheless, our theoretical analysis shows that the learning process with gradient errors can still be convergent towards the optimum of objective functions for many practical applications. Thus, we propose a simple method *AsynGrad* for asynchronous parallel learning with gradient error. Base on various dense feature models (LSTM, dense-CRF) and various NLP tasks, experiments show that *AsynGrad* achieves substantial improvement on training speed, and without any loss on accuracy.

1 Introduction

Stochastic learning methods can accelerate the training speed compared with traditional batch training methods. A widely used stochastic learning method is the stochastic gradient descent method (SGD) (Bertsekas, 1999; Bottou and Bousquet, 2008; Shalev-Shwartz and Srebro, 2008; Sun et al., 2012; Sun et al., 2014). For large-scale datasets, the SGD training methods can be much faster than batch training methods. For further improve the training speed over multi-core machines and clusters, a variety of asynchronous (lock-free) parallel learning methods has been developed based on stochastic learning (Niu et al., 2011; Mcmahan and Streeter, 2014). Those asynchronous methods have shown to be more efficient than the synchronous (locked) parallel learning versions (Langford et al., 2009; Gimpel et al., 2010). Other related work on parallel stochastic learning also includes (Zinkevich et al., 2010; Dekel et al., 2012; Recht and Re, 2013; Dean et al., 2012).

Existing asynchronous parallel learning methods are mainly for the sparse feature models, and feature sparseness is a major assumption for those parallel learning methods (Niu et al., 2011; Mcmahan and Streeter, 2014). For example, Niu et al. (2011) proposed an interesting asynchronous parallel learning method *HogWild* for strict sparse machine learning problems with sparse separable cost functions (e.g., sparse SVM, low-rank matrix completion). Niu et al. (2011) stated that the key idea that underlies their lock-free approach is that the targeted machine learning problems are sparse. More recently, Mcmahan and Streeter (2014) proposed a delay-tolerant asynchronous parallel learning method, which is an extension of the method of Niu et al. (2011). This asynchronous parallel learning method proposed by Mcmahan and Streeter (2014) also strictly requires the sparseness of the features.

The situation is different for the dense feature models like neural networks (e.g., LSTM, RNN). Since the existing asynchronous parallel learning methods are strictly for sparse feature models, it is no longer reasonable to apply those methods for the dense feature models like neural networks. To our knowledge, there is very limited study on developing asynchronous parallel learning methods for neural networks. In most cases only synchronous versions of parallel learning are applied to neural networks, such as

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 192–202, Osaka, Japan, December 11-17 2016.

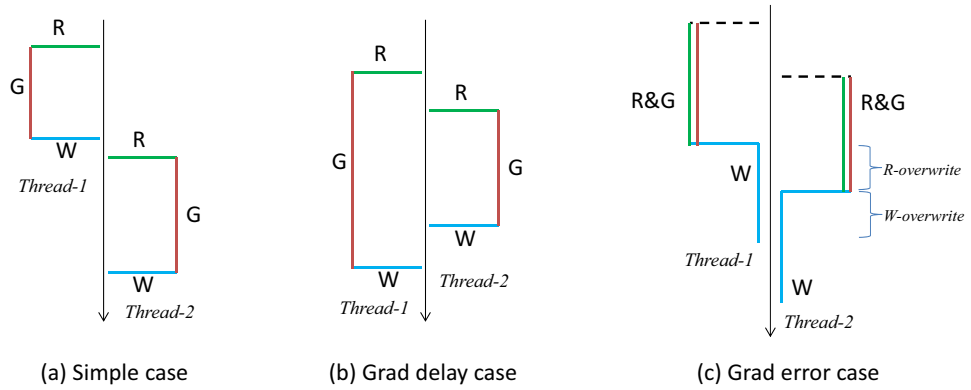


Figure 1: Illustrations of the *simple case* (left), the *gradient delay case* (middle), and the *gradient error case* (right) based on stochastic parallel learning. G means the gradient-computing action. R means the read action of shared memory. W means the write action.

GPU-based and mini-batch-based parallel learning methods. For GPU-based parallel learning, a matrix is computed in a synchronously parallelized way by using GPU units. For mini-batch-based parallel learning, the gradient of a mini-batch of samples are calculated in a synchronously parallelized way as well.

The major problem for applying asynchronous parallel learning to dense feature models is from the gradient errors. We show that gradient errors are very common and inevitable in applying asynchronous parallel learning to dense feature models. Suppose a dense feature model with 10 features/parameters, and we apply asynchronous (lock-free) parallel learning. When one thread is computing gradient, it needs to read the parameters from the shared memory. It is possible that 5 parameters are overwritten by another thread, which makes the computed gradient wrong. Not only there can be read-overwrite errors, but also there can be write-overwrite errors, as shown in Figure 1 (right).

Figure 1 illustrates the *simple case* (left), the *gradient delay case* (middle), and the *gradient error case* (right) for stochastic parallel learning. The simple case is normally from the synchronous parallel learning setting. The gradient delay case is considered for asynchronous parallel learning over sparse feature models (Niu et al., 2011; Mcmahan and Streeter, 2014). Essentially, the gradient delay problem is a simplification of the asynchronous parallel learning problem, and the simplification is from the feature sparseness. For the dense feature models like neural networks, it is unreasonable to use this simplification, and it goes to the gradient error case. As we can see from Figure 1 (right), there are both read-overwrite and write-overwrite problems between the two threads, and this created the gradient error. The gradient error problem is more complex than the gradient delay problem, and it requires new analysis and solutions. We will give more detailed analysis in Section 2.

Although the gradient error problem is more complex, it does not mean that the asynchronous parallel learning is doomed for the dense feature models. We give theoretical analysis to show that the learning process with the gradient error problem can still achieve the optimum given certain conditions, and those conditions are usually valid for the final convergence region of real-world applications.

Based on the analysis, we propose a simple asynchronous parallel learning method for dense feature models including neural networks and other structured models, and it works well in real-world NLP tasks in spite of gradient errors. Base on various dense feature models (LSTM, dense-CRF) and various NLP tasks, experiments show that our method achieves substantial improvement on training speed, and without any loss on accuracy.

2 *AsynGrad*: Asynchronous Parallel Learning with Gradient Error

As we can see from Figure 1, the gradient delay case is mostly considered for sparse feature models (Niu et al., 2011; Mcmahan and Streeter, 2014). Here the read and write of parameters for each sample is fast because the only a very small portion of the features are used for each sample. In this case, the read and write actions are considered being almost atomic actions, and the major concern goes to the "delay" of

Algorithm 1 *AsynGrad*: Asynchronous Parallel Learning with Gradient Error

Input: model weights $\boldsymbol{w}$, training set S of m samples
Run k threads in parallel with share memory, and procedure of each thread is as follows:
repeat
  Get a sample $\boldsymbol{z}$ uniformly at random from S
  Get the update term $\boldsymbol{s_z}(\boldsymbol{w})$, which is computed as $\nabla f_{\boldsymbol{z}}(\boldsymbol{w})$ but usually contains error
  Update $\boldsymbol{w}$ such that $\boldsymbol{w} \leftarrow \boldsymbol{w} - \gamma \boldsymbol{s_z}(\boldsymbol{w})$
until Convergence
return $\boldsymbol{w}$

the gradients (Niu et al., 2011; Mcmahan and Streeter, 2014). In Figure 1 (middle), thread-1's gradient is delayed by thread-2, because the former is expected to write to the memory before thread-2's write action. The *HogWild* method (Niu et al., 2011) and the delay-tolerant method (Mcmahan and Streeter, 2014) mainly cast/simplify the asynchronous parallel learning problem as the gradient delay problem, and they give solutions accordingly — they show that asynchronous parallel learning can achieve the optimum when dealing with the gradient delay problem.

The gradient delay problem is a simplification of the asynchronous parallel learning problem, and the simplification is from the feature sparseness. For the dense feature models like neural networks, it is no longer reasonable to use the simplification. The major problem for applying asynchronous parallel learning to dense feature models is the gradient errors.

As shown in Figure 1 (right), for dense feature models the read action is together with the gradient-computing action. When the feature is dense, it is not efficient to read all the parameters into a thread before computing the gradient. Also, when the feature is dense, the write action is no longer a fast action. As we can see from Figure 1 (right), there are both read-overwrite and write-overwrite problems, which create gradient errors. The gradient error problem is more complex than the gradient delay problem, and it requires new analysis and solutions.

2.1 *AsynGrad* Algorithm

Let $f(\boldsymbol{w})$ be the objective function of dense feature model and $\boldsymbol{w} \in \mathcal{W}$ is the weight vector. Recall that the SGD update with learning rate γ has a form like this:

$$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \gamma \nabla f_{\boldsymbol{z}_t}(\boldsymbol{w}_t) \tag{1}$$

where t represents a time stamp, and $\nabla f_{\boldsymbol{z}}(\boldsymbol{w}_t)$ is the stochastic estimation of the gradient based on $\boldsymbol{z}$, which is randomly drawn from the training set S.

Then, we assume a shared memory machine with multiple processors, and a vector of variables $\boldsymbol{w}$ in the shared memory is accessible to all processors. Each processor can read and update $\boldsymbol{w}$.

Based on the multicore computing machine, the proposed method creates k parallel threads, and $\boldsymbol{w}$ is shared among the k threads. For each thread, it gets a sample $\boldsymbol{z}$ uniformly at random from the training set S. Then, it tries to compute the update term $\boldsymbol{s_z}(\boldsymbol{w})$ as follows:

$$\boldsymbol{s_z}(\boldsymbol{w}) \leftarrow^{\approx} \nabla f_{\boldsymbol{z}}(\boldsymbol{w}) \tag{2}$$

where $\leftarrow^{\approx}$ means the $\boldsymbol{s_z}(\boldsymbol{w})$ is computed as the gradient $\nabla f_{\boldsymbol{z}}(\boldsymbol{w})$ in a single thread, but the shared weights $\boldsymbol{w}$ may be partially or completely overwritten by other threads when computing the gradient, which leads to gradient errors of $\nabla f_{\boldsymbol{z}}(\boldsymbol{w})$. Hence, $\boldsymbol{s_z}(\boldsymbol{w})$ is an approximation of the gradient $\nabla f_{\boldsymbol{z}}(\boldsymbol{w})$ with potential errors.

Then, the thread updates the shared weights $\boldsymbol{w}$ based on the update term $\boldsymbol{s_z}(\boldsymbol{w})$, such that

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \gamma \boldsymbol{s_z}(\boldsymbol{w}) \tag{3}$$

For each thread, it repeats this process until it reaches convergence — later we will show that this process can reach convergence by given certain conditions.

To summarize, the proposed parallel learning algorithm called *AsynGrad* is shown in Algorithm 1.

Although the implementation of *AsynGrad* is simple, the method *AsynGrad* itself is not that straightforward, because people may think it is wrong to use a method like *AsynGrad*. There is not much existing theoretical analysis and empirical evidence supporting the design of *AsynGrad*. In this paper, we show that it is actually reasonable and promising to use a method like *AsynGrad*. We give theoretical justification of the convergence of *AsynGrad* based on certain conditions, and we show experimental results that *AsynGrad* indeed works well in practice.

2.2 Theoretical Analysis of *AsynGrad*

Although *AsynGrad* does the weight update in parallel from different threads, from the viewpoint of $\boldsymbol{w}$ it simply receives a sequence of gradient updates (but with potential errors, which is the major difference from non-parallel SGD learning). In other words, the *AsynGrad* learning problem can be casted as a traditional SGD learning problem, and the only difference compared with traditional SGD is that the gradients are with errors. To state our convergence analysis results, we need several assumptions.

We assume f is strongly convex with modulus c, that is, $\forall \boldsymbol{w}, \boldsymbol{w}' \in \mathcal{W}$,

$$f(\boldsymbol{w}') \geq f(\boldsymbol{w}) + (\boldsymbol{w}' - \boldsymbol{w})^T \nabla f(\boldsymbol{w}) + \frac{c}{2}||\boldsymbol{w}' - \boldsymbol{w}||^2 \tag{4}$$

where $||\cdot||$ means 2-norm $||\cdot||_2$ by default in this work. For some dense feature models like dense-CRF, this assumption is suitable for the objective function. For some other dense feature models like neural networks, this assumption is a bit too strong because we know that the objective function of neural networks is non-convex. Nevertheless, even when the objective function is non-convex, the assumption usually holds within the final convergence region because the cost function is locally convex in many practical applications.

We also assume Lipschitz continuous differentiability of ∇f with the constant q, that is, $\forall \boldsymbol{w}, \boldsymbol{w}' \in \mathcal{W}$,

$$||\nabla f(\boldsymbol{w}') - \nabla f(\boldsymbol{w})|| \leq q||\boldsymbol{w}' - \boldsymbol{w}|| \tag{5}$$

Also, let the norm of $\boldsymbol{s_z}(\boldsymbol{w})$ is bounded by $\kappa \in \mathbb{R}^+$:

$$||\boldsymbol{s_z}(\boldsymbol{w})|| \leq \kappa \tag{6}$$

Moreover, it is reasonable to assume

$$\gamma c < 1 \tag{7}$$

because even the ordinary gradient descent methods will diverge if $\gamma c > 1$ (Niu et al., 2011).

Based on the conditions, we show that *AsynGrad* converges closely towards the minimum (optimum) $\boldsymbol{w}^*$ of $f(\boldsymbol{w})$ with a small distance expressed by ϵ, and the convergence rate is given as follows.

Theorem 1 (*AsynGrad* convergence and convergence rate)**.** *With the conditions (4), (5), (6), (7), let $\epsilon > 0$ be a target distance of convergence (i.e., the closeness of the convergence point to the real optimum). Let τ denote the bound to describe the severeness of gradient errors, such that*

$$[\nabla f(\boldsymbol{w}) - \boldsymbol{s}(\boldsymbol{w})]^T (\boldsymbol{w} - \boldsymbol{w}^*) \leq \tau \tag{8}$$

where $\boldsymbol{w}$ is the weight vector during AsynGrad *training, and $\boldsymbol{s}(\boldsymbol{w})$ is expected $\boldsymbol{s_z}(\boldsymbol{w})$ over $\boldsymbol{z}$ such that $\boldsymbol{s}(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{s_z}(\boldsymbol{w})]$. Let γ be a learning rate as*

$$\gamma = \frac{c\epsilon - 2\tau q}{\beta q \kappa^2} \tag{9}$$

where we can set β as any value as far as $\beta \geq 1$. Let t be the number of updates as follows

$$t \doteq \frac{\beta q \kappa^2 \log(q a_0/\epsilon)}{c(c\epsilon - 2\tau q)} \tag{10}$$

Table 1: Some major experimental results on dense-CRF. Time means time cost per iteration.

Method	POS-Tag		Chunking		MSR-WordSeg	
	Acc (%)	Time (s)	F-score (%)	Time (s)	F-score (%)	Time (s)
SGD	97.18	466.08	94.55	1.92	97.13	64.94
AsynGrad(10-thread)	97.18	108.38	94.59	0.25	97.15	8.11
AsynGrad(10-thread)+SR	97.35	25.99	94.60	0.21	97.22	6.59

where $\doteq$ means ceil-rounding of a real value to an integer, and a_0 is the initial distance such that $a_0 = ||\boldsymbol{w}_0 - \boldsymbol{w}^*||^2$. Then, after t updates of $\boldsymbol{w}$, AsynGrad *converges towards the optimum such that $\mathbb{E}[f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)] \leq \epsilon$, as far as the gradient errors are bounded such that*

$$\tau \leq \frac{c\epsilon}{2q} \tag{11}$$

The proof is in Section 4.

This theorem shows that *AsynGrad* is also convergent towards the optimum of the objective function, as far as the gradient errors are bounded such that Eq.(11) holds. For real-world applications, those assumptions and conditions usually hold at the final convergence region. In the final convergence region, $\boldsymbol{w}$ is not far away from the optimum $\boldsymbol{w}^*$, and both $||\nabla f(\boldsymbol{w})||$ and $||\boldsymbol{s}(\boldsymbol{w})||$ are supposed to be small. Thus, $[\nabla f(\boldsymbol{w}) - \boldsymbol{s}(\boldsymbol{w})]^T(\boldsymbol{w} - \boldsymbol{w}^*)$ is supposed to be small. This makes the bound τ to be small, which makes Eq.(11) valid in the final convergence region.

Moreover, the convergence rate is given in the theorem — *AsynGrad* is guaranteed to converge with t updates, and the value of t is given by Eq.(10).

The theoretical analysis can be concluded as follows. First, it shows that *AsynGrad* is convergent with gradient errors by given certain assumptions and conditions. Second, those assumptions and conditions are usually valid in the final convergence region of practical applications. Third, the convergence speed is given.

3 Experiments

We conduct experiments on natural language processing tasks as follows. Experiments are performed on a computer based on Intel(R) Xeon(R) 3.0GHz CPU of 12 cores, and with 128G memory.

Part-of-Speech Tagging (POS-Tag): We use the standard benchmark dataset in prior work (Collins, 2002), which is derived from PennTreeBank corpus and uses sections 0 to 18 of the Wall Street Journal (WSJ) for training (38,219 samples), and sections 22-24 for testing (5,462 samples). The evaluation metric is per-word accuracy.

Text Chunking (Chunking): The phrase chunking data is extracted from the data of the *CoNLL-2000 shallow-parsing shared task* (Sang and Buchholz, 2000; Sun et al., 2008). The training set consists of 8,936 sentences, and the test set consists of 2,012 sentences. The evaluation metric for this task is balanced F-score.

Word Segmentation (WordSeg). For the LSTM model, we use the social media word segmentation data (Weibo-WordSeg) provided by the NLPCC 2016 Shared Task.The training set contains around 60% of the data set, with 12,081 sentences. The test set contains around 40% of the data set, with 8,055 sentences. However, this data set is relatively new and we do not have good feature templates to implement the dense-CRF model. To deal with this problem, we use the MSR word segmentation data set (MSR-WordSeg) for the experiments on dense-CRF. The MSR-WordSeg data set is provided by *SIGHAN-2004 contest* (Gao et al., 2007). There are 86,918 training samples and 3,985 test samples. For this data set, we have standard feature templates, which are similar to (Sun et al., 2014). The evaluation metric is balanced F-score.

3.1 Experiments on Dense-CRF (Moderately Dense Case)

The dense-CRF is a moderately dense version of CRF by adding rich edge features, which usually has much better accuracy than traditional CRF for NLP tasks (Sun et al., 2012; Sun et al., 2014). For

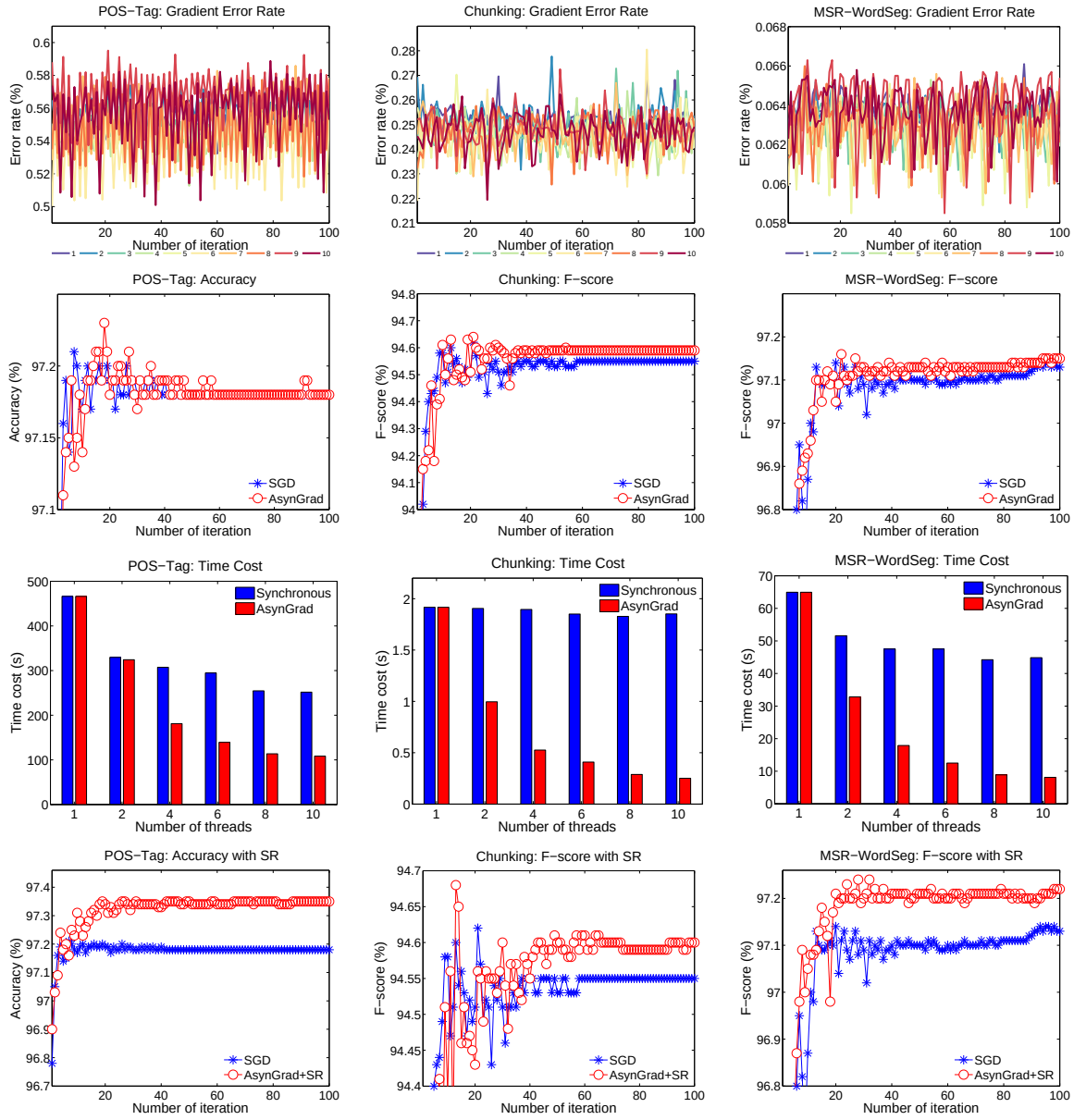


Figure 2: Experimental results on dense-CRF. For the 1st row, 1 to 10 denote the 10 threads of *AsynGrad*. For the 2nd row, *AsynGrad* denotes *AsynGrad* with 10 threads.

traditional implementation of CRF, usually the edges features contain only tag transitions with the form (y_{i-1}, y_i). In dense-CRF, we incorporate local tokens of $\boldsymbol{x}$ into the edge features, which is called rich edge features (Sun et al., 2012; Sun et al., 2014). For example, a rich edge feature can be (x_i, y_{i-1}, y_i) or even $(x_{i-1}, x_i, y_{i-1}, y_i)$. A simple way to automatically create massive rich edge features is to extend each node feature (x, y_i) to the rich edge feature (x, y_{i-1}, y_i), where x means the tokens of a local window. Usually a naive way to do this will cause feature explosion. However, it is easy to solve this problem — only extend high frequency node features (e.g., frequency larger than 10) to rich edge features. In many real-world tasks we find this type of dense-CRF works much better than traditional CRF, and without feature explosion problem even for tasks with many tags (e.g., the POS tagging task).

The standard SGD learning method (Bertsekas, 1999; Bottou and Bousquet, 2008; Shalev-Shwartz and Srebro, 2008) is chosen as the baseline. Based on tuning, the initial learning rate of SGD is set as 0.02, 0.05, 0.1 for the POS-Tag, Chunking, and MSR-WordSeg tasks, respectively. To control overfitting, we use the L_2 regularizer, i.e., a Gaussian prior. The L_2 regularization strength (i.e., the term λ) is set as 1, 0.5, 1 for the three tasks, respectively.

The experimental results are shown in Figure 2. The 1st row shows the percentage of gradient errors

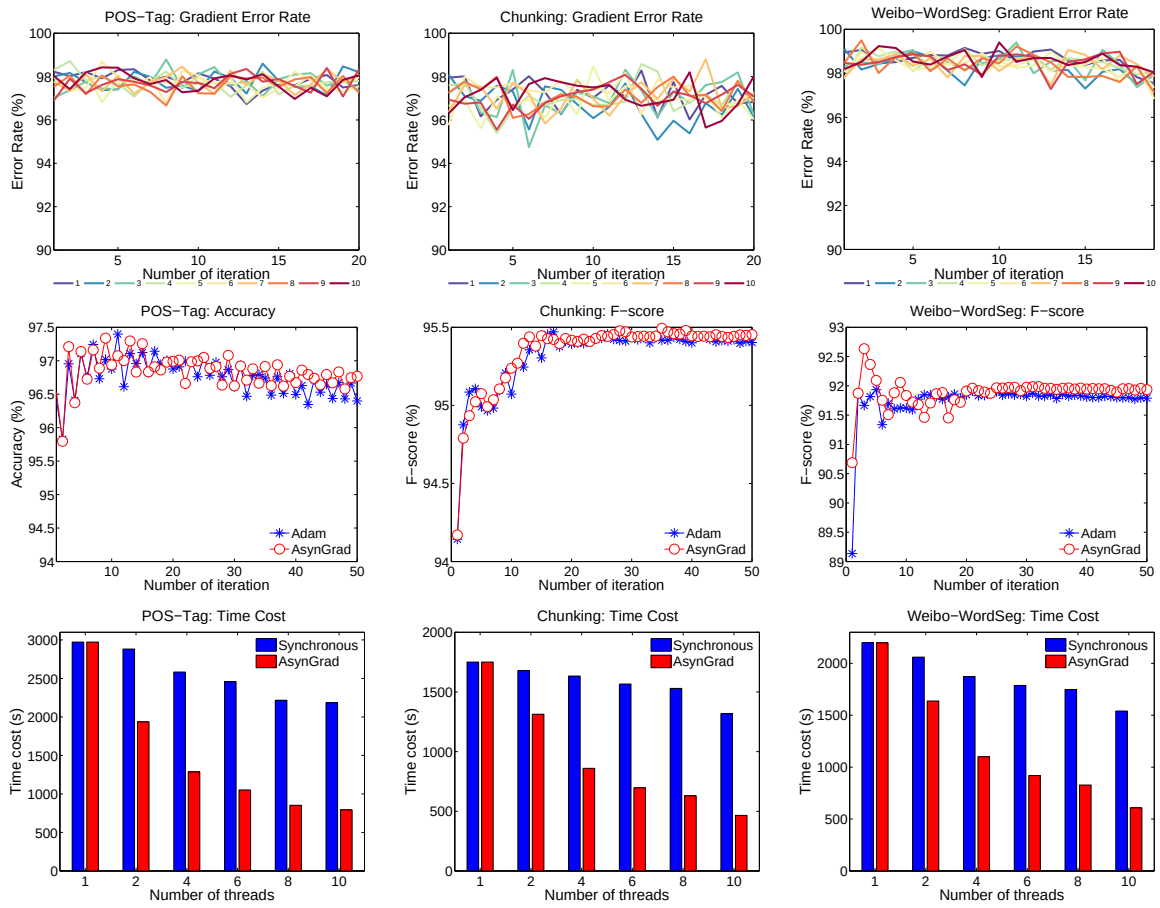


Figure 3: Experimental results on LSTM. For the 1st row, 1 to 10 denote the 10 threads of *AsynGrad*. For the 2nd row, *AsynGrad* denotes *AsynGrad* with 10 threads.

Table 2: Some major experimental results on LSTM. Time means time cost per iteration.

Method	POS-Tag		Chunking		Weibo-WordSeg	
	Acc (%)	Time (s)	F-score (%)	Time (s)	F-score (%)	Time (s)
Adam	96.40	2972.13	95.40	1750.68	91.79	2198.45
AsynGrad(10-thread)	96.77	793.40	95.45	465.95	91.93	607.22

among total gradients. For example, if a thread used 100 gradients and 5 gradients are with errors in this iteration, its gradient error rate is 5%. As we can see, the gradient errors are not rare in asynchronous parallel learning.

The 2nd row shows the accuracy/F-score curves. As we can see, when facing the gradient errors, the proposed asynchronous parallel learning method *AsynGrad* has no loss at all on the accuracy/F-score, compared with traditional SGD (single thread).

The 3rd row shows the speed acceleration of *AsynGrad*, compared with synchronous parallel learning method (Langford et al., 2009). As we can see, when adding more threads for parallel learning, *AsynGrad* achieves substantially better acceleration on the training speed than the synchronous parallel learning method.

The 4th row shows the accuracy/F-score curves by combining *AsynGrad* with structure regularization (SR) (Sun, 2014). SR is a simple method to prevent overfitting by splitting samples into mini-samples, thus it reduces the complexity and sparsity of structures (Sun, 2014). As we can see, *AsynGrad* can easily combine with SR to improve the accuracy/F-score on those tasks.

Some major experimental results are summarized in Table 1.

3.2 Experiments on LSTM (Completely Dense Case)

Recently, long short term memory networks (LSTM) has widely applied to many tasks (Chen et al., 2015; Hammerton, 2003; Cho et al., 2014). In this work, we use the bi-directional long short term memory network (Bi-LSTM) as the implementation of LSTM, which has better accuracy than single-directional LSTM in many practical tasks. The LSTM recurrent cell is controlled by three "gates", namely input gate, forget gate and output gate.

Adaptive learning versions of SGD are popular to train large neural networks, including the Adam learning method (Kingma and Ba, 2014), the RMSProp learning method (Tieleman and Hinton, 2012), and so on. The experiments on LSTM are based on the Adam learning method, with the hyper parameters as follows: $\beta_1 = \beta_2 = 0.95$, $\epsilon = 1 \times 10^{-4}$ (Kingma and Ba, 2014). The input/hidden dimension is 170, 280, and 220 for the POS-Tag, Chunking, and Weibo-WordSeg tasks, respectively. For the tasks with LSTM, we find there is almost no difference on the results by adding L_2 regularization or not. Hence, we do not add L_2 regularization for LSTM. All weight matrices, except for bias vectors and word embeddings, are diagonal matrices and randomly initialized by normal distribution.

The experimental results are shown in Figure 3. The 1st row shows the percentage of gradient errors among total gradients. As we can see, for neural networks like LSTM, the gradient errors are very common (the rate is over 90%) in asynchronous parallel learning. The gradient error rate is much higher than the dense-CRF case, this is because the LSTM is a much more dense model compared with dense-CRF. In fact, LSTM can be seen as a completely dense model because there is totally no sparse feature existing.

The 2nd row shows the accuracy/F-score curves. As we can see, when facing the intensive gradient errors which are over 90% on very dense models like LSTM, *AsynGrad* has no loss at all on accuracy/F-score.

The 3rd row shows the speed acceleration of *AsynGrad*, compared with synchronous parallel learning method (Langford et al., 2009). As we can see, when adding more threads for parallel learning, *AsynGrad* achieves substantially better acceleration on the training speed than the synchronous version.

The experiments shows that *AsynGrad* can substantially accelerate the training speed of LSTM, and without any loss on accuracy. Some major experimental results are summarized in Table 2.

4 Proof

Here we give the proof of Theorem 1. First, the recursion formula is derived. Then, the bounds are derived.

4.1 Recursion Formula

By subtracting $\boldsymbol{w}^*$ from both sides and taking norms for (1), we have

$$\begin{aligned} ||\boldsymbol{w}_{t+1} - \boldsymbol{w}^*||^2 &= ||\boldsymbol{w}_t - \gamma \boldsymbol{s}_{\boldsymbol{z}_t}(\boldsymbol{w}_t) - \boldsymbol{w}^*||^2 \\ &= ||\boldsymbol{w}_t - \boldsymbol{w}^*||^2 - 2\gamma(\boldsymbol{w}_t - \boldsymbol{w}^*)^T \boldsymbol{s}_{\boldsymbol{z}_t}(\boldsymbol{w}_t) + \gamma^2 ||\boldsymbol{s}_{\boldsymbol{z}_t}(\boldsymbol{w}_t)||^2 \end{aligned} \tag{12}$$

Taking expectations and let $a_t = \mathbb{E}||\boldsymbol{w}_t - \boldsymbol{w}^*||^2$, we have

$$\begin{aligned} a_{t+1} &= a_t - 2\gamma \mathbb{E}[(\boldsymbol{w}_t - \boldsymbol{w}^*)^T \boldsymbol{s}_{\boldsymbol{z}_t}(\boldsymbol{w}_t)] + \gamma^2 \mathbb{E}[||\boldsymbol{s}_{\boldsymbol{z}_t}(\boldsymbol{w}_t)||^2] \\ &\quad \text{(based on (6))} \\ &\leq a_t - 2\gamma \mathbb{E}[(\boldsymbol{w}_t - \boldsymbol{w}^*)^T \boldsymbol{s}_{\boldsymbol{z}_t}(\boldsymbol{w}_t)] + \gamma^2 \kappa^2 \\ &\quad \text{(since the random draw of } \boldsymbol{z}_t \text{ is independent of } \boldsymbol{w}_t) \\ &= a_t - 2\gamma \mathbb{E}[(\boldsymbol{w}_t - \boldsymbol{w}^*)^T \mathbb{E}_{\boldsymbol{z}_t}(\boldsymbol{s}_{\boldsymbol{z}_t}(\boldsymbol{w}_t))] + \gamma^2 \kappa^2 \\ &= a_t - 2\gamma \mathbb{E}[(\boldsymbol{w}_t - \boldsymbol{w}^*)^T \boldsymbol{s}(\boldsymbol{w}_t)] + \gamma^2 \kappa^2 \end{aligned} \tag{13}$$

We define

$$\delta(\boldsymbol{w}) = \nabla f(\boldsymbol{w}) - \boldsymbol{s}(\boldsymbol{w}) \tag{14}$$

and insert it into (4), it goes to

$$\begin{aligned} f(\boldsymbol{w}') &\geq f(\boldsymbol{w}) + (\boldsymbol{w}' - \boldsymbol{w})^T[\boldsymbol{s}(\boldsymbol{w}) + \delta(\boldsymbol{w})] + \frac{c}{2}||\boldsymbol{w}' - \boldsymbol{w}||^2 \\ &= f(\boldsymbol{w}) + (\boldsymbol{w}' - \boldsymbol{w})^T\boldsymbol{s}(\boldsymbol{w}) + \frac{c}{2}||\boldsymbol{w}' - \boldsymbol{w}||^2 + (\boldsymbol{w}' - \boldsymbol{w})^T\delta(\boldsymbol{w}) \end{aligned} \tag{15}$$

By setting $\boldsymbol{w}' = \boldsymbol{w}^*$, we further have

$$\begin{aligned} (\boldsymbol{w} - \boldsymbol{w}^*)^T\boldsymbol{s}(\boldsymbol{w}) &\geq f(\boldsymbol{w}) - f(\boldsymbol{w}^*) + \frac{c}{2}||\boldsymbol{w} - \boldsymbol{w}^*||^2 - (\boldsymbol{w} - \boldsymbol{w}^*)^T\delta(\boldsymbol{w}) \\ &\geq \frac{c}{2}||\boldsymbol{w} - \boldsymbol{w}^*||^2 - (\boldsymbol{w} - \boldsymbol{w}^*)^T\delta(\boldsymbol{w}) \end{aligned} \tag{16}$$

Combining (13) and (16), we have

$$\begin{aligned} a_{t+1} &\leq a_t - 2\gamma\mathbb{E}\Big[\frac{c}{2}||\boldsymbol{w}_t - \boldsymbol{w}^*||^2 - (\boldsymbol{w}_t - \boldsymbol{w}^*)^T\delta(\boldsymbol{w}_t)\Big] + \gamma^2\kappa^2 \\ &= (1 - c\gamma)a_t + 2\gamma\mathbb{E}[(\boldsymbol{w}_t - \boldsymbol{w}^*)^T\delta(\boldsymbol{w}_t)] + \gamma^2\kappa^2 \end{aligned} \tag{17}$$

Considering (8) and (14), it goes to

$$a_{t+1} \leq (1 - c\gamma)a_t + 2\gamma\tau + \gamma^2\kappa^2 \tag{18}$$

We can find a steady state a_∞ by the formula $a_\infty = (1 - c\gamma)a_\infty + 2\gamma\tau + \gamma^2\kappa^2$, which gives

$$a_\infty = \frac{2\tau + \gamma\kappa^2}{c} \tag{19}$$

Defining the function $A(x) = (1 - c\gamma)x + 2\gamma\tau + \gamma^2\kappa^2$, based on (18) we have

$$\begin{aligned} &a_{t+1} \leq A(a_t) \\ &\quad \text{(Taylor expansion of } A(\cdot) \text{ based on } a_\infty\text{, with } \nabla^2 A(\cdot) \text{ being 0)} \\ &\quad = A(a_\infty) + \nabla A(a_\infty)(a_t - a_\infty) \\ &\quad = A(a_\infty) + (1 - c\gamma)(a_t - a_\infty) \\ &\quad = a_\infty + (1 - c\gamma)(a_t - a_\infty) \end{aligned} \tag{20}$$

Thus, we have $a_{t+1} - a_\infty \leq (1 - c\gamma)(a_t - a_\infty)$, and unwrapping it goes to

$$a_t \leq (1 - c\gamma)^t(a_0 - a_\infty) + a_\infty \tag{21}$$

4.2 Bounds

Since $\nabla f(\boldsymbol{w})$ is Lipschitz according to (5), we have

$$f(\boldsymbol{w}) \leq f(\boldsymbol{w}') + \nabla f(\boldsymbol{w}')^T(\boldsymbol{w} - \boldsymbol{w}') + \frac{q}{2}||\boldsymbol{w} - \boldsymbol{w}'||^2$$

Setting $\boldsymbol{w}' = \boldsymbol{w}^*$, it goes to $f(\boldsymbol{w}) - f(\boldsymbol{w}^*) \leq \frac{q}{2}||\boldsymbol{w} - \boldsymbol{w}^*||^2$, such that

$$\mathbb{E}[f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)] \leq \frac{q}{2}\mathbb{E}||\boldsymbol{w}_t - \boldsymbol{w}^*||^2 = \frac{q}{2}a_t$$

In order to have $E[f(\boldsymbol{w}_t) - f(\boldsymbol{w}^*)] \leq \epsilon$, it is required that $\frac{q}{2}a_t \leq \epsilon$, that is

$$a_t \leq \frac{2\epsilon}{q} \tag{22}$$

Combining (21) and (22), it is required that

$$(1 - c\gamma)^t(a_0 - a_\infty) + a_\infty \leq \frac{2\epsilon}{q} \tag{23}$$

To meet this requirement, it is sufficient to set the learning rate γ such that both terms on the left side are less than $\frac{\epsilon}{q}$. For the requirement of the second term $a_\infty \leq \frac{\epsilon}{q}$, recalling (19), it goes to $\gamma \leq \frac{c\epsilon - 2\tau q}{q\kappa^2}$. Thus, introducing a real value $\beta \geq 1$, we can set γ as

$$\gamma = \frac{c\epsilon - 2\tau q}{\beta q \kappa^2} \tag{24}$$

Note that, to make this formula meaningful, it is required that $c\epsilon - 2\tau q \geq 0$. Thus, it is required that

$$\tau \leq \frac{c\epsilon}{2q}$$

which is solved by the condition of (11).

On the other hand, we analyze the requirement of the first term that $(1 - c\gamma)^t (a_0 - a_\infty) \leq \frac{\epsilon}{q}$. Since $a_0 - a_\infty \leq a_0$, it holds by requiring $(1 - c\gamma)^t a_0 \leq \frac{\epsilon}{q}$, which goes to

$$t \geq \frac{\log \frac{\epsilon}{q a_0}}{\log (1 - c\gamma)} \tag{25}$$

Since $\log (1 - c\gamma) \leq -c\gamma$ given (7), and that $\log \frac{\epsilon}{q a_0}$ is a negative term, we have

$$\frac{\log \frac{\epsilon}{q a_0}}{\log (1 - c\gamma)} \leq \frac{\log \frac{\epsilon}{q a_0}}{-c\gamma}$$

Thus, (25) holds by requiring

$$t \geq \frac{\log \frac{\epsilon}{q a_0}}{-c\gamma} = \frac{\log (q a_0 / \epsilon)}{c\gamma} \tag{26}$$

Combining (24) and (26), the problem can be addressed as far as

$$t \geq \frac{\beta q \kappa^2 \log (q a_0 / \epsilon)}{c(c\epsilon - 2\tau q)}$$

Thus, setting $t \doteq \frac{\beta q \kappa^2 \log (q a_0 / \epsilon)}{c(c\epsilon - 2\tau q)}$ can solve the problem. This completes the proof. □

5 Conclusions

In this paper we show that gradient errors are very common and inevitable in dense feature models. Nevertheless, we show that the learning process with the gradient error problem can still approach the optimum in many practical applications. We propose a simple method *AsynGrad* for asynchronous stochastic parallel learning with gradient error. We show that *AsynGrad* works well for dense feature models like neural networks and dense-CRF, in spite of gradient errors. Base on various NLP tasks, our experiments show that *AsynGrad* can substantially accelerate the training speed of LSTM and dense-CRF, and without any loss on accuracy.

Acknowledgements

Many thanks to Shuming Ma, Jingjing Xu, and Xuancheng Ren for the help on running experiments and drawing figures. This work was supported in part by National Natural Science Foundation of China (No. 61300063), and National High Technology Research and Development Program of China (863 Program, No. 2015AA015404).

References

D.P. Bertsekas. 1999. *Nonlinear Programming*. Athena Scientific.

Léon Bottou and Olivier Bousquet. 2008. The tradeoffs of large scale learning. In *NIPS'08*, volume 20, pages 161–168.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In Llu ís M àrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *EMNLP*, pages 1197–1206. The Association for Computational Linguistics.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP'02*, pages 1–8.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew W. Senior, Paul A. Tucker, Ke Yang, and Andrew Y. Ng. 2012. Large scale distributed deep networks. In *Proceedings of NIPS*, pages 1232–1240.

Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. 2012. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13:165–202.

Jianfeng Gao, Galen Andrew, Mark Johnson, and Kristina Toutanova. 2007. A comparative study of parameter estimation methods for statistical natural language processing. In *Proceedings of ACL'07*, pages 824–831.

Kevin Gimpel, Dipanjan Das, and Noah A. Smith. 2010. Distributed asynchronous online learning for natural language processing. In Mirella Lapata and Anoop Sarkar, editors, *Proceedings of CoNLL*, pages 213–222.

James Hammerton. 2003. Named entity recognition with long short-term memory. In Walter Daelemans and Miles Osborne, editors, *CoNLL*, pages 172–175. ACL.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

John Langford, Alex J. Smola, and Martin Zinkevich. 2009. Slow learners are fast. In *NIPS'09*, pages 2331–2339.

Brendan Mcmahan and Matthew Streeter. 2014. Delay-tolerant algorithms for asynchronous distributed online learning. In *Advances in Neural Information Processing Systems 27*, pages 2915–2923.

Feng Niu, Benjamin Recht, Christopher Re, and Stephen J. Wright. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS'11*, pages 693–701.

Benjamin Recht and Christopher Re. 2013. Parallel stochastic gradient algorithms for large-scale matrix completion. *Math. Program. Comput.*, 5(2):201–226.

Erik Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL'00*, pages 127–132.

Shai Shalev-Shwartz and Nathan Srebro. 2008. Svm optimization: inverse dependence on training set size. In *ICML'08*, pages 928–935. ACM.

Xu Sun, Louis-Philippe Morency, Daisuke Okanohara, and Jun'ichi Tsujii. 2008. Modeling latent-dynamic in shallow parsing: A latent conditional model with improved inference. In *Proceedings of COLING'08*, pages 841–848, Manchester, UK.

Xu Sun, Houfeng Wang, and Wenjie Li. 2012. Fast online training with frequency-adaptive learning rates for chinese word segmentation and new word detection. In *Proceedings of ACL'12*, pages 253–262.

Xu Sun, Wenjie Li, Houfeng Wang, and Qin Lu. 2014. Feature-frequency-adaptive on-line training for fast and accurate natural language processing. *Computational Linguistics*, 40(3):563–586.

Xu Sun. 2014. Structure regularization for structured prediction. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2402–2410.

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

Martin Zinkevich, Markus Weimer, Alexander J. Smola, and Lihong Li. 2010. Parallelized stochastic gradient descent. In *Proceedings of NIPS'10*, pages 2595–2603.

An Empirical Exploration of Skip Connections for Sequential Tagging

Huijia Wu[1,3], Jiajun Zhang[1,3], and Chengqing Zong[1,2,3]
[1]National Laboratory of Pattern Recognition, Institute of Automation, CAS
[2]CAS Center for Excellence in Brain Science and Intelligence Technology
[3]University of Chinese Academy of Sciences
{huijia.wu,jjzhang,cqzong}@nlpr.ia.ac.cn

Abstract

In this paper, we empirically explore the effects of various kinds of skip connections in stacked bidirectional LSTMs for sequential tagging. We investigate three kinds of skip connections connecting to LSTM cells: (a) skip connections to the gates, (b) skip connections to the internal states and (c) skip connections to the cell outputs. We present comprehensive experiments showing that skip connections to cell outputs outperform the remaining two. Furthermore, we observe that using gated identity functions as skip mappings works pretty well. Based on this novel skip connections, we successfully train deep stacked bidirectional LSTM models and obtain state-of-the-art results on CCG supertagging and comparable results on POS tagging.

1 Introduction

In natural language processing, sequential tagging mainly refers to the tasks of assigning discrete labels to each token in a sequence. Typical examples include part-of-speech (POS) tagging and combinatory category grammar (CCG) supertagging. A regular feature of sequential tagging is that the input tokens in a sequence cannot be assumed to be independent since the same token in different contexts can be assigned to different tags. Therefore, the classifier should have memories to remember the contexts to make a correct prediction.

Bidirectional LSTMs (Graves and Schmidhuber, 2005) become dominant in sequential tagging problems due to the superior performance (Wang et al., 2015; Vaswani et al., 2016; Lample et al., 2016). The horizontal hierarchy of LSTMs with bidirectional processing can remember the long-range dependencies without affecting the short-term storage. Although the models have a deep horizontal hierarchy (the depth is the same as the sequence length), the vertical hierarchy is often shallow, which may not be efficient at representing each token. Stacked LSTMs are deep in both directions, but become harder to train due to the feed-forward structure of stacked layers.

Skip connections (or shortcut connections) enable unimpeded information flow by adding direct connections across different layers (Raiko et al., 2012; Graves, 2013; Hermans and Schrauwen, 2013). However, there is a lack of exploration and analyzing various kinds of skip connections in stacked LSTMs. There are two issues to handle skip connections in stacked LSTMs: One is where to add the skip connections, the other is what kind of skip connections should be used to pass the information. To answer the first question, we empirically analyze three positions of LSTM blocks to receive the previous layer's output. For the second one, we present an identity mapping to receive the previous layer's outputs. Furthermore, following the gate design of LSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) and highway networks (Srivastava et al., 2015a; Srivastava et al., 2015b), we observe that adding a multiplicative gate to the identity function will help to improve performance.

In this paper, we present a neural architecture for sequential tagging. The input of the network are token representations. We concatenate word embeddings to character embeddings to represent the word and morphemes. A deep stacked bidirectional LSTM with well-designed skip connections is then used

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 203–212, Osaka, Japan, December 11-17 2016.

to extract the features needed for classification from the inputs. The output layer uses *softmax* function to output the tag distribution for each token.

Our main contribution is that we empirically evaluated the effects of various kinds of skip connections within stacked LSTMs. We present comprehensive experiments on the supertagging task showing that skip connections to the cell outputs using identity function multiplied with an exclusive gate can help to improve the network performance. Our model is evaluated on two sequential tagging tasks, obtaining state-of-the-art results on CCG supertagging and comparable results on POS tagging.

2 Related Work

Skip connections have been widely used for training deep neural networks. For recurrent neural networks, Schmidhuber (1992); El Hihi and Bengio (1995) introduced deep RNNs by stacking hidden layers on top of each other. Raiko et al. (2012); Graves (2013); Hermans and Schrauwen (2013) proposed the use of skip connections in stacked RNNs. However, the researchers have paid less attention to the analyzing of various kinds of skip connections, which is our focus in this paper.

The works closely related to ours are Srivastava et al. (2015b), He et al. (2015), Kalchbrenner et al. (2015), Yao et al. (2015), Zhang et al. (2016), and Zilly et al. (2016). These works are all based on the design of extra connections between different layers. Srivastava et al. (2015b) and He et al. (2015) mainly focus on feed-forward neural network, using well-designed skip connections across different layers to make the information pass more easily. The Grid LSTM proposed by Kalchbrenner et al. (2015) extends the one dimensional LSTMs to many dimensional LSTMs, which provides a more general framework to construct deep LSTMs.

Yao et al. (2015) and Zhang et al. (2016) propose highway LSTMs by introducing gated direct connections between internal states in adjacent layers and do not use skip connections, while we propose gated skip connections across cell outputs. Zilly et al. (2016) introduce recurrent highway networks (RHN) which use a single recurrent layer to make RNN deep in a vertical direction, in contrast to our stacked models.

3 Recurrent Neural Networks for Sequential Tagging

Consider a recurrent neural network applied to sequential tagging: Given a sequence $x = (x_1, \ldots, x_T)$, the RNN computes the hidden state $h = (h_1, \ldots, h_T)$ and the output $y = (y_1, \ldots, y_T)$ by iterating the following equations:

$$h_t = f(x_t, h_{t-1}; \theta_h) \tag{1}$$

$$y_t = g(h_t; \theta_o) \tag{2}$$

where $t \in \{1, \ldots, T\}$ represents the time. x_t represents the input at time t, h_{t-1} and h_t are the previous and the current hidden state, respectively. f and g are the transition function and the output function, respectively. θ_h and θ_o are network parameters.

We use a negative log-likelihood cost to evaluate the performance, which can be written as:

$$\mathcal{C} = -\frac{1}{N} \sum_{n=1}^{N} \log y_{t^n} \tag{3}$$

where $t^n \in \mathbb{N}$ is the true target for sample n, and y_{t^n} is the t-th output in the *softmax* layer given the inputs x^n.

The core idea of Long Short-Term Memory networks is to replace (1) with the following equation:

$$c_t = f(x_t, h_{t-1}) + c_{t-1} \tag{4}$$

where c_t is the internal state of the memory cell, which is designed to store the information for much longer time. Besides this, LSTM uses gates to avoid weight update conflicts.

Standard LSTMs process sequences in temporal order, which will ignore future context. Bidirectional LSTMs solve this problem by combining both the forward and the backward processing of the input sequences using two separate recurrent hidden layers:

$$\overrightarrow{h_t} = \text{LSTM}(\overrightarrow{x_t}, \overrightarrow{h_{t-1}}, \overrightarrow{c_{t-1}}) \tag{5}$$

$$\overleftarrow{h_t} = \text{LSTM}(\overleftarrow{x_t}, \overleftarrow{h_{t-1}}, \overleftarrow{c_{t-1}}) \tag{6}$$

$$y_t = g(\overrightarrow{h_t}, \overleftarrow{h_t}) \tag{7}$$

where LSTM$(\cdot)$ is the LSTM computation. $\overrightarrow{x_t}$ and $\overleftarrow{x_t}$ are the forward and the backward input sequence, respectively. The output of the two hidden layers $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ in a birectional LSTM are connected to the output layer.

Stacked RNN is one type of deep RNNs, which refers to the hidden layers are stacked on top of each other, each feeding up to the layer above:

$$h_t^l = f^l(h_t^{l-1}, h_{t-1}^l) \tag{8}$$

where h_t^l is the t-th hidden state of the l-th layer.

4 Various kinds of Skip Connections

Skip connections in simple RNNs are trivial since there is only one position to connect to the hidden units. But for stacked LSTMs, the skip connections need to be carefully treated to train the network successfully. In this section, we analyze and compare various types of skip connections. At first, we give a detailed definition of stacked LSTMs, which can help us to describe skip connections. Then we start our construction of skip connections in stacked LSTMs. At last, we formulate various kinds of skip connections.

Stacked LSTMs without skip connections can be defined as:

$$\begin{pmatrix} i_t^l \\ f_t^l \\ o_t^l \\ s_t^l \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \qquad \begin{aligned} c_t^l &= f_t^l \odot c_{t-1}^l + i_t^l \odot s_t^l \\ h_t^l &= o_t^l \odot \tanh(c_t^l) \end{aligned} \tag{9}$$

During forward pass, LSTM needs to calculate c_t^l and h_t^l, which is the cell's internal state and the cell outputs state, respectively. To get c_t^l, s_t^l needs to be computed to store the current input. Then this result is multiplied by the input gate i_t^l, which decides when to keep or override information in memory cell c_t^l. The cell is designed to store the previous information c_{t-1}^l, which can be reset by a forget gate f_t^l. The new cell state is then obtained by adding the result to the current input. The cell outputs h_t^l are computed by multiplying the activated cell state by the output gate o_t^l, which learns when to access memory cell and when to block it. "sigm" and "tanh" are the sigmoid and tanh activation function, respectively. $W^l \in \mathbb{R}^{4n \times 2n}$ is the weight matrix needs to be learned.

The hidden units in stacked LSTMs have two forms. One is the hidden units in the same layer $\{h_t^l, t \in 1, \ldots, T\}$, which are connected through an LSTM. The other is the hidden units at the same time step $\{h_t^l, l \in 1, \ldots, L\}$, which are connected through a feed-forward network. LSTM can keep the short-term memory for a long time, thus the error signals can be easily passed through $\{1, \ldots, T\}$. However, when the number of stacked layers is large, the feed-forward network will suffer the gradient vanishing/exploding problems, which make the gradients hard to pass through $\{1, \ldots, L\}$.

The core idea of LSTM is to use an identity function to make the constant error carrosel. He et al. (2015) also use an identity mapping to train a very deep convolution neural network with improved performance. All these inspired us to use an identity function for the skip connections. Rather, the gates of LSTM are essential parts to avoid weight update conflicts, which are also invoked by skip connections. Following highway gating, we use a gate multiplied with identity mapping to avoid the conflicts.

Skip connections are cross-layer connections, which means that the output of layer $l-2$ is not only connected to the layer $l-1$, but also connected to layer l. For stacked LSTMs, h_t^{l-2} can be connected to the gates, the internal states, and the cell outputs in layer l's LSTM blocks. We formalize these below:

Skip connections to the gates. We can connect h_t^{l-2} to the gates through an identity mapping:

$$\begin{pmatrix} i_t^l \\ f_t^l \\ o_t^l \\ s_t^l \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} \left(W^l \; I^l \right) \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \\ h_t^{l-2} \end{pmatrix} \tag{10}$$

where $I^l \in \mathbb{R}^{4n \times n}$ is the identity mapping.

Skip connections to the internal states. Another kind of skip connections is to connect h_t^{l-2} to the cell's internal state c_t^l:

$$c_t^l = f_t^l \odot c_{t-1}^l + i_t^l \odot s_t^l + h_t^{l-2} \tag{11}$$
$$h_t^l = o_t^l \odot \tanh(c_t^l) \tag{12}$$

Skip connections to the cell outputs. We can also connect h_t^{l-2} to cell outputs:

$$c_t^l = f_t^l \odot c_{t-1}^l + i_t^l \odot s_t^l \tag{13}$$
$$h_t^l = o_t^l \odot \tanh(c_t^l) + h_t^{l-2} \tag{14}$$

Skip connections using gates. Consider the case of skip connections to the cell outputs. The cell outputs grow linearly during the presentation of network depth, which makes the h_t^l's derivative vanish and hard to convergence. Inspired by the introduction of LSTM gates, we add a gate to control the skip connections through retrieving or blocking them:

$$\begin{pmatrix} i_t^l \\ f_t^l \\ o_t^l \\ g_t^l \\ s_t^l \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \qquad \begin{aligned} c_t^l &= f_t^l \odot c_{t-1}^l + i_t^l \odot s_t^l \\ h_t^l &= o_t^l \odot \tanh(c_t^l) + g_t^l \odot h_t^{l-2} \end{aligned} \tag{15}$$

where g_t^l is the gate which can be used to access the skipped output h_t^{l-2} or block it. When g_t^l equals 0, no skipped output can be passed through skip connections, which is equivalent to traditional stacked LSTMs. Otherwise, it behaves like a feed-forward LSTM using gated identity connections. Here we omit the case of adding gates to skip connections to the internal state, which is similar to the above case.

Skip connections in bidirectional LSTM. Using skip connections in bidirectional LSTM is similar to the one used in LSTM, with a bidirectional processing:

$$\begin{aligned} \overrightarrow{c_t^l} &= \overrightarrow{f} \odot \overrightarrow{c_{t-1}^l} + \overrightarrow{i} \odot \overrightarrow{s_t^l} \\ \overrightarrow{h_t^l} &= \overrightarrow{o} \odot \tanh(\overrightarrow{c_t^l}) + \overrightarrow{g} \odot \overrightarrow{h_t^{l-2}} \end{aligned} \qquad \begin{aligned} \overleftarrow{c_t^l} &= \overleftarrow{f} \odot \overleftarrow{c_{t-1}^l} + \overleftarrow{i} \odot \overleftarrow{s_t^l} \\ \overleftarrow{h_t^l} &= \overleftarrow{o} \odot \tanh(\overleftarrow{c_t^l}) + \overleftarrow{g} \odot \overleftarrow{h_t^{l-2}} \end{aligned} \tag{16}$$

5 Neural Architecture for Sequential Tagging

Sequential tagging can be formulated as $P(t|w;\theta)$, where $w = [w_1, \ldots, w_T]$ indicates the T words in a sentence, and $t = [t_1, \ldots, t_T]$ indicates the corresponding T tags. In this section we introduce an neural architecture for $P(\cdot)$, which includes an input layer, a stacked hidden layers and an output layer. Since the stacked hidden layers have already been introduced, we only introduce the input and the output layer here.

5.1 Network Inputs

Network inputs are the representation of each token in a sequence. There are many kinds of token representations, such as using a single word embedding, using a local window approach, or a combination of word and character-level representation. Here our inputs contain the concatenation of word representations, character representations, and capitalization representations.

Word representations. All words in the vocabulary share a common look-up table, which is initialized with random initializations or pre-trained embeddings. Each word in a sentence can be mapped to an embedding vector w_t. The whole sentence is then represented by a matrix with columns vector $[w_1, w_2, \ldots, w_T]$. We use a context window of size d surrounding with a word w_t to get its context information. Following Wu et al. (2016), we add logistic gates to each token in the context window. The word representation is computed as $w_t = [r_{t-\lfloor d/2 \rfloor} w_{t-\lfloor d/2 \rfloor}; \ldots; r_{t+\lfloor d/2 \rfloor} w_{t+\lfloor d/2 \rfloor}]$, where $r_t := [r_{t-\lfloor d/2 \rfloor}, \ldots, r_{t+\lfloor d/2 \rfloor}] \in \mathbb{R}^d$ is a logistic gate to filter the unnecessary contexts, $w_{t-\lfloor d/2 \rfloor}, \ldots, w_{t+\lfloor d/2 \rfloor}$ is the word embeddings in the local window.

Character representations. Prefix and suffix information about words are important features in sequential tagging. Inspired by Fonseca et al. (2015) et al, which uses a character prefix and suffix with length from 1 to 5 for part-of-speech tagging, we concatenate character embeddings in a word to get the character-level representation. Concretely, given a word w consisting of a sequence of characters $[c_1, c_2, \ldots, c_{l_w}]$, where l_w is the length of the word and $L(\cdot)$ is the look-up table for characters. We concatenate the leftmost most 5 character embeddings $L(c_1), \ldots, L(c_5)$ with its rightmost 5 character embeddings $L(c_{l_w-4}), \ldots, L(c_{l_w})$. When a word is less than five characters, we pad the remaining characters with the same special symbol.

Capitalization representations. We lowercase the words to decrease the size of word vocabulary to reduce sparsity, but we need an extra capitalization embeddings to store the capitalization features, which represent whether or not a word is capitalized.

5.2 Network Outputs

For sequential tagging, we use a *softmax* activation function $g(\cdot)$ in the output layer:

$$y_t = g(W^{hy}[\overrightarrow{h_t}; \overleftarrow{h_t}]) \tag{17}$$

where y_t is a probability distribution over all possible tags. $y_k(t) = \frac{\exp(h_k)}{\sum_{k'} \exp(h_{k'})}$ is the k-th dimension of y_t, which corresponds to the k-th tags in the tag set. W^{hy} is the hidden-to-output weight.

6 Experiments

6.1 Combinatory Category Grammar Supertagging

Combinatory Category Grammar (CCG) supertagging is a sequential tagging problem in natural language processing. The task is to assign supertags to each word in a sentence. In CCG the supertags stand for the lexical categories, which are composed of the basic categories such as N, NP and PP, and complex categories, which are the combination of the basic categories based on a set of rules. Detailed explanations of CCG refers to (Steedman, 2000; Steedman and Baldridge, 2011).

The training set of this task only contains 39604 sentences, which is too small to train a deep model, and may cause over-parametrization. But we choose it since it has been already proved that a bidirectional recurrent net fits the task by many authors (Lewis et al., 2016; Vaswani et al., 2016).

6.1.1 Dataset and Pre-processing

Our experiments are performed on CCGBank (Hockenmaier and Steedman, 2007), which is a translation from Penn Treebank (Marcus et al., 1993) to CCG with a coverage 99.4%. We follow the standard splits, using sections 02-21 for training, section 00 for development and section 23 for the test. We use a full category set containing 1285 tags. All digits are mapped into the same digit '9', and all words are lowercased.

6.1.2 Network Configuration

Initialization. There are two types of weights in our experiments: recurrent and non-recurrent weights. For non-recurrent weights, we initialize word embeddings with the pre-trained 200-dimensional GolVe vectors (Pennington et al., 2014). Other weights are initialized with the Gaussian distribution

Model	Dev	Test
Clark and Curran (2007)	91.5	92.0
Lewis et al. (2014)	91.3	91.6
Lewis et al. (2016)	94.1	94.3
Xu et al. (2015)	93.1	93.0
Xu et al. (2016)	93.49	93.52
Vaswani et al. (2016)	94.24	94.5
7-layers + skip output + gating	94.51	94.67
7-layers + skip output + gating (no char)	94.33	94.45
7-layers + skip output + gating (no dropout)	94.06	94.0
9-layers + skip output + gating	**94.55**	**94.69**

Table 1: 1-best supertagging accuracy on CCGbank. "skip output" refers to the skip connections to the cell output, "gating" refers to adding a gate to the identity function, "no char" refers to the models that do not use the character-level information, "no dropout" refers to models that do not use dropout.

$\mathcal{N}(0, \frac{1}{\sqrt{\text{fan-in}}})$ scaled by a factor of 0.1, where *fan-in* is the number of units in the input layer. For recurrent weight matrices, following Saxe et al. (2013) we initialize with random orthogonal matrices through SVD to avoid unstable gradients. Orthogonal initialization for recurrent weights is important in our experiments, which takes about 2% relative performance enhancement than other methods such as Xavier initialization (Glorot and Bengio, 2010).

Hyperparameters. For the word representations, we use a small window size of 3 for the convolutional layer. The dimension of the word representation after the convolutional operation is 600. The size of character embedding and capitalization embeddings are set to 5. The number of cells of the stacked bidirectional LSTM is set to 512. We also tried 400 cells or 600 cells and found this number did not impact performance so much. All stacked hidden layers have the same number of cells. The output layer has 1286 neurons, which equals to the number of tags in the training set with a RARE symbol.

Training. We train the networks using the back-propagation algorithm, using stochastic gradient descent (SGD) algorithm with an equal learning rate 0.02 for all layers. We also tried other optimization methods, such as momentum (Plaut and others, 1986), Adadelta (Zeiler, 2012), or Adam (Kingma and Ba, 2014), but none of them perform as well as SGD. Gradient clipping is not used. We use on-line learning in our experiments, which means the parameters will be updated on every training sequences, one at a time. We trained the 7-layer network for roughly 2 to 3 days on one NVIDIA TITAN X GPU using Theano [1] (Team et al., 2016).

Regularization. Dropout (Srivastava et al., 2014) is the only regularizer in our model to avoid overfitting. Other regularization methods such as weight decay and batch normalization do not work in our experiments. We add a binary dropout mask to the local context windows on the embedding layer, with a drop rate p of 0.25. We also apply dropout to the output of the first hidden layer and the last hidden layer, with a 0.5 drop rate. At test time, weights are scaled with a factor $1 - p$.

6.1.3 Results

Table 1 shows the comparisons with other models for supertagging. The comparisons do not include any externally labeled data and POS labels. We use stacked bidirectional LSTMs with gated skip connections for the comparisons, and report the highest 1-best supertagging accuracy on the development set for final testing. Our model presents state-of-the-art results compared to the existing systems. The character-level information (+ 3% relative accuracy) and dropout (+ 8% relative accuracy) are necessary to improve the performance.

[1] `http://deeplearning.net/software/theano/`

6.1.4 Experiments on Skip Connections

We experiment with a 7-layer model on CCGbank to compare different kinds of skip connections introduced in Section 4. Our analysis mainly focuses on the identity function and the gating mechanism. The comparisons (Table 2) are summarized as follows:

No skip connections. When the number of stacked layers is large, the performance will degrade without skip connections. The accuracy in a 7-layer stacked model without skip connections is 93.94% (Table 2), which is lower than the one using skip connections.

Various kinds of skip connections. We experiment with the gated identity connections between internal states introduced in Zhang et al.(2016), but the network performs not good (Table 2, 93.14%). We also implement the method proposed in Zilly et al. (2016), which we use a single bidirectional RNH layer with a recurrent depth of 3 with a slightly modification[2]. Skip connections to the cell outputs with identity function and multiplicative gating achieves the highest accuracy (Table 2, 94.51%) on the development set. We also observe that skip to the internal states without gate get a slightly better performance (Table 2, 94.33%) than the one with gate (94.24%) on the development set. Here we recommend to set the forget bias to 0 to get a better development accuracy.

Identity mapping. We use the sigmoid function to the previous outputs to break the identity link, in which we replace $g_t \odot h_t^{l-1}$ in Eq. (15) with $g_t \odot \sigma(h_t^{l-1})$, where $\sigma(x) = \frac{1}{1+e^{-x}}$. The result of the sigmoid function is 94.02% (Table 2), which is poor than the identity function. We can infer that the identity function is more suitable than other scaled functions such as sigmoid or tanh to transmit information.

Exclusive gating. Following the gating mechanism adopted in highway networks, we consider adding a gate g_t to make a flexible control to the skip connections. Our gating function is $g_t^l = \sigma(W_g^l h_{t-1}^l + U_g^l h_t^{l-2})$. Gated identity connections are essential to achieving state-of-the-art result on CCGbank.

Case	Variant	Dev	Test
H-LSTM, Zhang et al.(2016)	-	93.14	93.52
RHN, Zilly et al. (2016)	$L = 3$, with output gates	94.28	94.24
no skip connections	-	93.94	94.26
to the gates, Eq. (10)	-	93.9	94.22
to the internals	no gate, Eq. (11)	94.33	94.63
	with gate	94.24	94.52
to the outputs	no gate, Eq. (14)	93.89	93.98
	with gate, $b_f = 5$, Eq. (15)	94.23	94.81
	with gate, $b_f = 0$, Eq. (15)	94.51	94.67
	sigmoid mapping: $g_t \odot \sigma(h_t^{l-1})$	94.02	94.18

Table 2: Accuracy on CCGbank using 7-layer stacked bidirectional LSTMs, with different types of skip connections. b_f is the bias of the forget gate.

6.1.5 Experiments on Number of Layers

Table 3 compares the effect of the depth in the stacked models. We can observe that the performance is getting better with the increased number of layers. But when the number of layers exceeds 9, the performance will be hurt. In the experiments, we found that the number of stacked layers between 7 and 9 are the best choice using skip connections. Notice that we do not use layer-wise pretraining (Bengio et al., 2007; Simonyan and Zisserman, 2014), which is an important technique in training deep networks.

[2]Our original implementation of Zilly et a. (2016) with a recurrent depth of 3 fails to converge. The reason might be due to the explosion of s_L^t under addition. To avoid this, we replace s_L^t with $o_t * \tanh(s_L^t)$ in the last recurrent step.

Further improvements might be obtained with this method to build a deeper network with improved performance.

#Layers	Dev	Test
3	94.21	94.35
5	94.51	94.57
7	94.51	94.67
9	94.55	94.7
11	94.43	94.65

Table 3: Accuracy on CCGbank using gated identity connections to cell outputs, with different number of stacked layers.

6.2 Part-of-speech Tagging

Part-of-speech tagging is another sequential tagging task, which is to assign POS tags to each word in a sentence. It is very similar to the supertagging task. Therefore, these two tasks can be solved in a unified architecture. For POS tagging, we use the same network configurations as supertagging, except the word vocabulary size and the tag set size. We conduct experiments on the Wall Street Journal of the Penn Treebank dataset, adopting the standard splits (sections 0-18 for the train, sections 19-21 for validation and sections 22-24 for testing).

Model	Test
Søgaard (2011)	97.5
Ling et al. (2015)	97.36
Wang et al. (2015)	**97.78**
Vaswani et al. (2016)	97.4
7-layers + skip output + gating	97.45
9-layers + skip output + gating	97.45

Table 4: Accuracy for POS tagging on WSJ.

Although the POS tagging result presented in Table 4 is slightly below the state-of-the-art, we neither do any parameter tunings nor change the network architectures, just use the one getting the best development accuracy on the supertagging task. This proves the generalization of the model and avoids heavy work of model re-designing.

7 Conclusions

This paper investigates various kinds of skip connections in stacked bidirectional LSTM models. We present a deep stacked network (7 or 9 layers) that can be easily trained and get improved accuracy on CCG supertagging and POS tagging. Our experiments show that skip connections to the cell outputs with the gated identity function performs the best. Our explorations could easily be applied to other sequential processing problems, which can be modelled with RNN architectures.

Acknowledgements

The research work has been funded by the Natural Science Foundation of China under Grant No. 61333018 and supported by the Strategic Priority Research Program of the CAS under Grant No. XDB02070007. We thank the anonymous reviewers for their useful comments that greatly improved the manuscript.

References

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. 2007. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153.

Stephen Clark and James R Curran. 2007. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics*, 33(4):493–552.

Salah El Hihi and Yoshua Bengio. 1995. Hierarchical recurrent neural networks for long-term dependencies. In *NIPS*, volume 400, page 409. Citeseer.

Erick R Fonseca, João Luís G Rosa, and Sandra Maria Aluísio. 2015. Evaluating word embeddings and a revised corpus for part-of-speech tagging in portuguese. *Journal of the Brazilian Computer Society*, 21(1):1–14.

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

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256.

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

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 190–198.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Lstm can solve hard long time lag problems. *Advances in neural information processing systems*, pages 473–479.

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: a corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2015. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Mike Lewis and Mark Steedman. 2014. Improved CCG parsing with semi-supervised supertagging. *Transactions of the Association for Computational Linguistics*, 2:327–338.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. Lstm ccg parsing. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

David C Plaut et al. 1986. Experiments on learning by back propagation.

Tapani Raiko, Harri Valpola, and Yann LeCun. 2012. Deep learning made easier by linear transformations in perceptrons. In *AISTATS*, volume 22, pages 924–932.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Jürgen Schmidhuber. 1992. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Anders Søgaard. 2011. Semisupervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 48–52. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015a. Training very deep networks. In *Advances in neural information processing systems*, pages 2377–2385.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015b. Highway networks. *arXiv preprint arXiv:1505.00387*.

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. *Non-Transformational Syntax: Formal and Explicit Models of Grammar. Wiley-Blackwell*.

Mark Steedman. 2000. *The syntactic process*, volume 24. MIT Press.

Theano Development Team, Rami Alrfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frdric Bastien, Justin Bayer, and Anatoly Belikov. 2016. Theano: A python framework for fast computation of mathematical expressions.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging with lstms. In *Proceedings of the Human Language Technology Conference of the NAACL*.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *arXiv preprint arXiv:1510.06168*.

Huijia Wu, Jiajun Zhang, and Chengqing Zong. 2016. A dynamic window neural network for ccg supertagging. *arXiv preprint arXiv:1610.02749*.

Wenduan Xu, Michael Auli, and Stephen Clark. 2015. CCG supertagging with a recurrent neural network. *Volume 2: Short Papers*, page 250.

Wenduan Xu, Michael Auli, and Stephen Clark. 2016. Expected f-measure training for shift-reduce parsing with recurrent neural networks. In *Proceedings of NAACL-HLT*, pages 210–220.

Kaisheng Yao, Trevor Cohn, Katerina Vylomova, Kevin Duh, and Chris Dyer. 2015. Depth-gated lstm. In *Presented at Jelinek Summer Workshop on August*, volume 14, page 1.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yaco, Sanjeev Khudanpur, and James Glass. 2016. Highway long short-term memory rnns for distant speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5755–5759. IEEE.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2016. Recurrent highway networks. *arXiv preprint arXiv:1607.03474*.

Exploring Text Links for Coherent Multi-Document Summarization

Xun Wang, Masaaki Nishino, Tsutomu Hirao, Katsuhito Sudoh and Masaaki Nagata
NTT Communication Science Labortories
Kyoto, 619-0237, Japan

{wang.xun,nishino.masaaki,hirao.tsutomu
sudoh.katsuhito,nagata.masaaki@lab.ntt.co.jp}

Abstract

Summarization aims to represent source documents by a shortened passage. Existing methods focus on the extraction of key information, but often neglect coherence. Hence the generated summaries suffer from a lack of readability. To address this problem, we have developed a graph-based method by exploring the links between text to produce coherent summaries. Our approach involves finding a sequence of sentences that best represent the key information in a coherent way. In contrast to the previous methods that focus only on salience, the proposed method addresses both coherence and informativeness based on textual linkages. We conduct experiments on the DUC2004 summarization task data set. A performance comparison reveals that the summaries generated by the proposed system achieve comparable results in terms of the ROUGE metric, and show improvements in readability by human evaluation.

1 Introduction

Automatic summarization is extremely useful in this age of information overload. It provides readers with easier access to information without the labour of reading the source text. According to the number of documents dealt with, summarization falls into two categories: single document summarization and multi-document summarization. While they both aim to represent the source text using a shorten passage, the latter deals with a set of documents sharing the same topic. Based on the method adopted, existing approaches to summarization can be divided into two kinds: abstraction based or extraction based. The difference lies in the sentences they use to generate summaries: the former selects sentences (clauses, or other text units, hereafter we refer to all of them as sentences.) from source documents and the latter generates new sentences. Most existing summarization systems are extraction-based because abstraction-based methods require the use of natural language generation technology, which is still a growing field. This paper, without exception, also employs extraction-based methods and we focus on multi-documents summarization.

Currently the extraction-based methods face some major challenges. One is informativeness, which means we need to maintain the important information of source documents in summaries. This is the focus of almost all research on summarization. Another challenge is presentation, namely that the extracted text should be well presented, i.e., it should contain little redundancy and be coherent so as to be readily understandable. Previous work has addressed the problem of redundancy, and some successful solutions like Maximum Marginal Relevance (MMR) (Carbonell and Goldstein, 1998) have been proposed and widely adopted (e.g., (Li and Li, 2013)), but very few try to deal with coherence. Therefore the generated summaries generally suffer as regards readability and are very difficult to use for practical applications. In the report of the TAC 2011 summarization task (Owczarzak and Dang, 2011), it is stated that "in general, automatic summaries are better than baselines[1], except *Readability*." Such a statement suggests, as for summarization, coherence should be treated with the same as salience and redundancy.

[1]The baseline they used is the lead paragraph method and summaries are evaluated by human and ROUGE (Recall-Oriented Understudy for Gisting Evaluation (Lin, 2004)).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 213–223, Osaka, Japan, December 11-17 2016.

Existing work addresses coherence in summarization from different aspects. One kind of method employs reordering after selecting sentences, and the drawback is evident: coherence is considered after sentence selection. Another kind of widely adopted method takes discourse relations into consideration when selecting sentences, as discourse relations are believed to be essential for maintaining textual coherence. Hirao et al. (2013) formulated single document summarization as to extract a sub tree from the complete discourse tree and thus preserve the relations between extracted document units to form a readable text. Wang et al. (2015) extended it to multi-document summarization by regarding a document set as one document and developed a model which combined discourse parsing and summarization together. Christensen et al. (2013) proposed a graph-based model to bypass the tree constraints. They employed rich textual features to build a discourse relation graph for source documents with the aim of representing the relations between sentences (both inter and intra-document relations). Christensen et al. (2013) reported ROUGE scores lower than some baselines. This is because that, they claim, ROUGE is salience-focused and fails to notice the improvement in coherence. In a further human evaluation, they reported improvements in readability.

These discourse-based methods without exception have discourse analysis as a prerequisite. As we all know, discourse analysis is still under development thus preventing the expected improvement. Furthermore, languages other than English do not enjoy plenty of ready-to-use discourse analysis tools. This also limits the usage of these discourse-based methods.

Is it possible to consider coherence in summarization without discourse analysis? Before answering this question, we need to find out what is the key to coherence in text. According to the centering theory (Grosz et al., 1995; Walker et al., 1998), the coherence of text is to a large extent maintained by entities and the relations between them. This indicates that discourse analysis is not a must to preserve coherence; we can directly take advantages of entities and their relations to generate coherence text.

Based on this point, we design a novel graph-based model for multi-document summarization that eliminates the effort of conducting discourse relation analysis (inter or intra document) and generates informative and readable summaries. We formulate the document set as a graph whose nodes corresponds to sentences. These nodes are connected with each other according to the entities they contains and the relations between their containing entities. Each path in the graph represents a piece of text and is evaluated using a novel scoring function that considers informativeness and coherence. To extract a summary is to find a path in the graph with the highest score. This is a weighted longest path problem. We further present a variant of the proposed model based on local coherence and explore decoding algorithms for both of them.

Experiments are conducted on the Document Understanding Conference (DUC) 2004 multi-document summarization task data set. As ROUGE cannot fully capture our improvement in coherence which is one of the key contributions of this work, we also conduct a human evaluation. Results show that we obtain summaries comparative with state-of-the-art systems in terms of ROUGE metrics and get improvements in readability in human evaluations.

This work provides a method of generating high quality summaries without the effort of discourse analysis. The proposed method can be easily extended to other languages without much efforts. It also provides inspiration as regards other tasks that require computers to generate coherent text. The rest of the papers is organized as follows: Section 2 presents the centering theory and a coherence model based on entities. Section 3 presents our model. Section 4 describes the experiments and results. Section 5 presents some previous work and Section 6 concludes this paper.

2 Centering Theory and Coherence Modelling

The centering theory (Grosz et al., 1995) as a popular theory on discourse analysis, serves as the basis of some coherence evaluation methods (Barzilay and Lapata, 2008; Burstein et al., 2010; Li and Hovy, 2014; Li and Jurafsky, 2016) and enables us to measure the coherence score of any given text without discourse parsing solely based on the reappearance of entities. Entities here refer to noun/pronoun

word/phrases [2].

According to the centering theory, we have the following assumptions:

1. Text that contains successive mentions of the same entities would be more coherent.

2. The main entities that are focused on tend to play an important grammatical role, such as the subject or object of the sentences.

Therefore the key to the coherence of a text lies in what entities it contains and how their roles change. The coherence of a generated text can be evaluated accordingly.

Barzilay and Lapata (2008) presented such a model. The key is to represent text as an *entity grid*. Assume text T contains n sentences $\{S_1, S_2..., S_n\}$ and m entities. r_i^k represents the grammatical role of Entity e_k in Sentence S_i. Four kinds of roles are used, i.e., "subj", "obj", "others" and "absent". "Others" indicates that the entity is present, but is neither the subject nor the object. Then the grammatical roles of e_k in text T can be expressed as a sequence: $\{r_1^k, r_2^k, ..., r_n^k\}$. For each entity in T, such a chain showing how the entity's grammatical roles change in T is extracted. Thus text T can be represented as an $n * m$ matrix $M(T)$ where n is the number of sentences and m is the number of entities in T, and $M(T)_{ij}$ corresponds to the grammatical roles of Entity j in Sentence i. $M(T)$ is referred to as the *Entity Grid* of T (Barzilay and Lapata, 2008).

To calculate the coherence score of T, Barzilay and Lapata (2008) used $M(T)$ as a feature vector. They calculated the transition probability for $|\{s(subj), o(bj), x(others), -(absent)\}^2| = 16$ transition patterns from $M(T)$ without distinguishing between entities, to form a vector $f(T)$ for T, and a weight vector w was then learnt from training data so that $w * f(T)$ can be used as the coherence score for T.

This kind of method has been adopted in many studies (Filippova and Strube, 2007; Barzilay and Lapata, 2008; Burstein et al., 2010). In particular, Filatova and Hatzivassiloglou (2004) extends entity grids to model semantical relations between entities, which provides a possible further improvement for our models.

3 Modeling Summarization

The above model can only be used to measure coherence but summarization is much complex as it involves not only coherence bust also informativeness and redundancy. We design a much more sophisticated models leveraging entities.

Two models are presented below. Both of them are based on entities and consider coherence as well as informativeness. The first one is based on global coherence and the second one local coherence. The global coherence consider the full sequence when evaluating coherence and the local coherence is calculated based on relations between adjacent sentences. Intuitively, global coherence is better than local coherence, but considering the full sequence increases the time complexity. The model based on local coherence, on the other hand, reduces the time complexity and enables us to obtain an exact solution efficiently.

3.1 Problem Set-up

Assume we have K documents with n sentences in total. Note that we are dealing with multi-document summarization, and we do not distinguish between inter-document and intra-document relations. We construct a graph with n nodes, each of which corresponds to one sentence. Weighted directed edges are used to connect these nodes together. To each node, we assign a cost score, which is the number of words the corresponding sentence contains. To each path in the directed graph, we assign a gain score. The gain score is a comprehensive evaluation of the informativeness and coherence of the sequence of sentences represented by the path. The problem of extracting a good summary becomes the problem of extracting the best path. Note that it is an asymmetric graph. Gain scores for $A \rightarrow B \rightarrow C$ and $C \rightarrow B \rightarrow A$ are different. The direction determines the positions of corresponding sentences in the generated text.

[2] In some previous work on summarization (Takamura and Okumura, 2009; Hirao et al., 2013), concepts are used to measure informativeness. Concepts can be used to refer any non functional words, including adjectives, adverbs. All the entities can be regarded as concepts, but some concept words (non-nominal words) are not entities. Entity is a subset of Concept.

One more thing to consider is the redundancy. Instead of formulating redundancy explicitly, we remove edges connecting similar sentences to turn the complete graph into an incomplete graph. This ensures that similar sentences do not occupy adjacent positions in the generated summaries and thus reduce redundancy. The similarities of sentence pairs are based on word overlaps and we keep $d\%$ of all the edges.

Note that for temporal text removing edges can also help us maintain the temporal relations between sentences, though we do not explore this point here.

3.2 Summarization Considering Global [3] Coherence

To extract a summary is to find such a sequence of sentences Seq that maximizes $Score(Seq)$.

$$\begin{aligned} Score(Seq) &= \sum_{k=1}^{m} a_k F_k \\ F_k &= \prod_i p_{e_k}(r_i^k r_{i+1}^k), S_i, S_{i+1} \in Seq \\ s.t. &\sum_{S_i \in Seq} length(S_i) \leq threshold \end{aligned} \tag{1}$$

a_k is the weight of Entity e_k. r_i^k is the state of Entity e_k in Sentence S_i. Here we use four states: "s", "o", "x", "-", which represent "subj", "obj", "present" and "absent" respectively. It is also possible to use more or fewer states.

$p_{e_k}(**)$ is the transition probability between two states for e_k. For each document set, the transition probabilities for each entity is estimated using $p_{e_k}(ab) = \frac{\#e_k(a)e_k(b)}{n-K}$. $\#e_k(a)e_k(b)$ marks the times that Entity e_k presents as grammatical role a in the preceding sentence and as grammatical role b in the following one. $n-K$ denotes the total number of adjacent sentence pairs in a document set with K documents and n sentences. F_k is the coherence score contributed by e_k in the extracted sequence Seq. F_k is based on the transitions of e_k between adjacent sentences in Seq. We use $Score(Seq)$ which considers salience, coherence and redundancy as an index as to how suitable the extracted sentence sequence Seq is as a summary. This model is a weighted longest path problem with a fixed length.

This is an NP-hard problem. Due to the time cost, we adopt the simple randomized algorithm as shown in Algorithm 1 to obtain an approximated solution. Other decoding algorithms like greedy algorithms

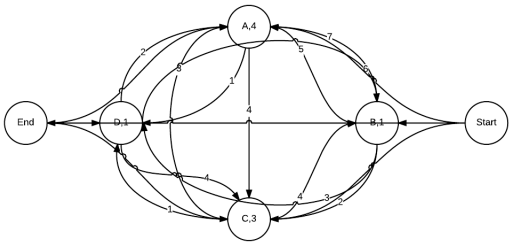
Algorithm 1 A randomized algorithm for the weighted longest path problem

```
Initialization:
Set U ⟵ all the sentences in the current doc set
Set S ⟵ EmptySet
Queue Q ⟵ EmptySet
repeat
  randomly select sentence s ∈ U & s ∉ Q;
  if length(s) + Σ_i length(s_i) <= threshold, s_i ∈ Q then
    push s to the rear of Q
  else
    push Q into S, Queue Q ⟵ EmptySet
  end if
until 10K times
return argmax_Q F(T), Q ∈ S
```

can also be employed. But none of them are capable of obtaining an exact solution. Below we present another model considering *local* coherence.

[3] "Global" means the model considers coherence according to the *whole* text.

Figure 1: A Complete Graph with Dummy Start and End Nodes



3.3 Summarization Considering Local Coherence

The above model considers global coherence which is calculated according to the whole text. The model presented below is directly based on local coherence and enables us to obtain an exact solution. We want to maximize $Score(Seq)$:

$$\begin{aligned} Score(Seq) &= \sum_{S_i \in Seq} (\alpha \sum_{e_k \in S_i} a_k + (1-\alpha) gain_{i,(i+1)}) \\ s.t. \quad & \sum_{S_i \in Seq} length(S_i) \leq threshold \end{aligned} \tag{2}$$

This formulation contains two parts. $\sum_{e_k \in S_i} a_k$ implies the weight of Sentence S_i, which is the sum of its containing entities' weights. $gain_{i,(i+1)}$ is the gain score for $Edge(S_i, S_{i+1})$. α manipulates the impacts of the two parts.

$$gain(S_i, S_{i+1}) = \sum_{e_k \in S_i \bigcup S_{i+1}} p_{e_k}(r_i^k r_{i+1}^k) \tag{3}$$

As is stated, r_i^k is the state of Entity e_k in Sentence S_i.

For the convenience of decoding, we turn the above model to an integer linear programming (ILP) problem. We add two dummy nodes, called $Start$ and End Node. All paths start from $Start$ and end with End. The costs of both $Start$ and End are 0. The gains of edges connected with $Start$ or End are 0. Note that although here we present a full connected graph for simplicity, in reality we deleted several edges to reduce redundancy. Following such a setting, an arbitrary path in the old graph (the one without dummy Start and End nodes) can be represented as a path from $Start$ to End. We write the $Start$ node as Node 0 and the End node as Node t. Then we formulate the problem of the weighted longest path as follows:

$$maximize \alpha \sum_i (\sum_{e_k \in S_i} a_k) x_i + (1-\alpha) \sum_{i,j} gain_{i,j} y_{ij}$$

subject to

$$\begin{cases} 1) \sum_i cost_i x_i \leq threshold \\ 2) \sum_i y_{0i} = 1 \\ 3) \sum_i y_{it} = 1 \\ 4) \sum_i y_{ij} + y_{0j} - (\sum_i y_{ji} + y_{jt}) = 0, \forall j \\ 5) \sum_i y_{ij} + y_{0j} - x_j = 0, \forall j \\ 6) x_i \in \{0,1\}, \forall i \\ 7) y_{ij} \in \{0,1\}, \forall i,j \end{cases} \tag{4}$$

Equations 2 and 3 are used to ensure we have only one start and one end node. Equation 4 ensures that the in degree equals the out degree for all nodes. Equation 5 ensures that the in degree is either 0 or 1 and equals x_a for all nodes. $x_i = 1$ indicates that S_i is selected for the summary. $x_i = 0$ means S_i is

not contained in the summary. $y_{ij} = 1$ means S_i and S_j are selected and placed as adjacent sentences in the summary. $cost_i$ is the number of words in S_i (length of S_i).

We resolve this ILP problem using the dual simplex method provided by IBM CPLEX optimizer [4] which is a powerful optimization software package. CPLEX provides both a primal simplex method and a dual simplex method for ILP problems. Here we adopt the latter.

4 Experiments & Analysis

4.1 Experiment

Experiments are conducted on the data set of the DUC2004 Summarization Task, which is a multi-document summarization task. 50 document clusters, each of which consists of 10 documents, are given. One summary is to be generated for each cluster. The target length is up to 100 words. Weights of entities are learnt by logistic regression as is adopted by Takamura and Okumura (2009) [5]. For entities that are not contained in DUC2003, we assign tf-based weights to them as Barzilay and Lapata (2008) did.

For the evaluation we firstly use the generally acknowledged metric for summarization: ROUGE metric. It essentially calculates n-gram overlaps between automatically generated summaries and human written (the gold standard) summaries. A high level of overlap indicates a high level of shared information between the two summaries. Among others, we focus on ROUGE-1 in the discussion of the result, because ROUGE-1 has proved to have a strong correlation with human annotation (Lin, 2004).

Some necessary preprocessing includes stemming, removing stop-words and simple simplification. In previous work, there is usually no co-reference resolution and different words are regarded as different entities. Here we use Stanford CoreNLP toolkit (Manning et al., 2014) to deal with the co-reference problem. The Stanford CoreNLP toolkit contains a ready-to-use entity identification tool and a co-reference resolution tool. The co-reference resolution is not a must, though preferred if reliable tools are available.

After the co-reference resolution, different forms of the same entities are replaced by their unified forms. For each document set, we need to estimate the transition probabilities for each entity according to the documents contained in the cluster as stated above.

Parameters are tuned using the DUC2003 dataset. d is the threshold of redundancy. We keep d percent of all edges and d varies from 10 to 100 with an interval of 10. We tune the parameter using the randomized algorithm and evaluate the results using ROUGE-1 Recall. In the following experiments, we set $d = 80$, which means we keep 80% of the sentences.

As for the model presented in Section 3.3, we need to tune α. Using the same data, we try α from 0 to 1 with an interval of 0.1 and eventually choose $\alpha = 0.4$.

4.2 Evaluation & Discussion

We compare our models with state-of-the-art multi-document summarization systems using ROUGE and human evaluation. The former aims to evaluate informativeness and the latter targets readability.

ROUGE Evaluation MCKP is the maximum coverage methods proposed by Takamura and Okumura (2009). Lin is a model that uses a class of submodular functions (Lin and Bilmes, 2011). Christ is a graph based model proposed by Christensen et al. (2013). DPP is the determinantal point processes model Borodin (2009) and ICSI is another model based on maximum coverage Gillick et al. (2008). The results of DPP and ICSI comes from the repository presented in Hong et al. (2014). M1 is our model described in Section 3.1. M2 is the model described in Section 3.3, which is resolved using an ILP method. MEAD Radev et al. (2004a) is a baseline that employs ranking algorithms to generate multi-document summaries.

The results are shown in Table 1. As we can see, our system (M1 and M2) produces comparable results to the state-of-the-art systems. With the MCKP method, all content words are used as concepts. But in

[4] http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/

[5] This method was first proposed by Yih et al. (2007) and then improved by Takamura and Okumura (2009). Here we follow the same steps with Takamura and Okumura (2009).

our systems, only nouns and pronouns are regarded as entities. There are fewer nouns and pronouns than content words. This has a negative impact on the evaluation of information coverage. But according to the experiment results, our approach still obtain satisfying results based on these entities. It proves that even with much simpler feature settings of just nouns and pronouns, the proposed model generates summaries with good coverage of the important information in source documents. We have addressed that ROUGE is merely an index of informativeness and cannot evaluate our improvements in readability as has been proved by *Christ*, another coherence-focused model (Christensen et al., 2013). So we also conduct a human evaluation.

Human Evaluation As some of the systems mentioned in Table 1 are not accessible, in this work we compare summaries produced by some typical systems: M2 (the best proposed system evaluated by ROUGE), MCKP (one of the state-of-the-art salience-focused methods) and humans (the gold standard).

We asked four professional annotators (who are not the authors of this paper and have rich experience in annotating various NLP tasks and are fluent in English) to assign a score to each summary regarding its readability. We randomly selected 48 summaries (16+16+16) from the three systems, and asked them to assign a readability score to each document without reading the source documents (summarization is useful because we do not need to read source documents). The score is an integer between 1 (very poor) and 5 (very good).

The average scores for the 3 systems are $Human = 4.3; M2 = 3.5; MCKP = 3.1$. Significance testing (significance level $\alpha = 0.05$) shows that the summaries generated by the proposed method show improvements in readability compared with previous salience-focused work.

Type	SysName	ROUGE-1(R)
Simple Ranking	MEAD	.339
Maximum Coverage	MCKP	.385
	ICSI	.384
Point Process	DPP	.398
Sub Modular	Lin	.394
Discourse-based	Christ	.373
	M1	.383
	M2	.390

Table 1: ROUGE Results on DUC2004

In our model, we assume the states of entities can be formulated as Markov chains. Although sophisticated models can be employed, such assumptions help simplify the model and they are proved to be of use. Also we can use more or fewer grammatical roles for entities. We tried using just two kinds of roles: presence and absence, and the performance we obtained was unsatisfying.

5 Related Work

A summary is much shorter than the original documents but still needs to provide readers with sufficient information. Hence the summarization systems need to identify important information and keep as much of it as possible. Most existing research follows such a guideline and takes salience as its sole focus.

Salience-focused systems cannot guarantee the readability of the generated text as they fail to take coherence into consideration. Sentence reordering, as a post processing task has began to develop. Apparently, it cannot make up for the flaws of salience-focused systems because it is simply a reorganization of sentences. Besides, it also faces problems when dealing with temporal text (Yan et al., 2011; Ge et al., 2015). A better solution is to consider coherence when selecting sentences. Such comprehensive models have been proposed. Most of them are discourse driven and sacrifice informativeness for coherence. In this sense, our model is novel in dealing with coherence without discourse analysis.

5.1 Salience-Focused Method

As stated, the summarization systems need to identify the important information and keep as much of it in the generated summaries as possible. One straightforward method is Maximum Marginal Relevance (Carbonell and Goldstein, 1998) (MMR). It is a greedy method, and is proposed to select sentences that are most relevant but not too similar to the already selected ones. It tries to keep a balance between relevance and redundancy. MMR is also widely employed to avoid redundancy in summarization systems. Among existing research, one popular kind is the ranking method (e.g., Textrank (Mihalcea and Tarau, 2004), Lexrank (Erkan and Radev, 2004) and its variants (Wan et al., 2007; Wang et al., 2012)), which construct a graph between text units and use ranking algorithms to select top sentences to build summaries. Another kind is the optimization method. Our work is one of this kind. It formulates summarization as finding a subset that optimizes certain objective functions without violating certain constraints. To find such an optimal subset is a combinatorial optimization problem, which is an NP hard problem and hence cannot be solved in linear time (McDonald, 2007).

Recently, maximum coverage methods have been proposed and yield good results (Gillick et al., 2009; Gillick and Favre, 2009; Takamura and Okumura, 2009). Maximum coverage methods formulate summarization as a maximum knapsack problem (MKMC). In MKMC methods, the meanings of sentences are believed to be made up by concepts, which usually refer to content words. And summarization involves extracting a subset of sentences that covers as many important concepts as possible without violating the length constraint. It is usually formulated as an integer linear problem. And some algorithms are proposed for obtaining approximated solutions (Takamura and Okumura, 2009; Gillick et al., 2009). Lin and Bilmes (2011) design a class of submodular functions for document summarization. The functions they use combine two parts, encouraging the summary to be representative of the corpus, and rewarding diversity separately. Other methods that have been applied to summarization include centroid-based methods (Radev et al., 2004b; Saggion and Gaizauskas, 2004), and minimum dominating set methods (Shen and Li, 2010). All these methods suffer in coherence.

5.2 Coherence-Focused Method

Sentence reordering methods are developed to correct the salience-focused models. Sentence reordering tries to generate a more coherent text by reordering its contents. Rich semantic and syntactic features are used to find a better permutation for input sentences (Barzilay et al., 2001; Bollegala et al., 2010; Okazaki et al., 2004).

The drawback to sentence reordering is obvious. The preceding sentence selection focuses solely on informativeness and totally neglects coherence. Thus it prevents the improvements expected from permutation. This is confirmed by the fact that the above methods all reports limited improvement. A consideration of coherence during sentence selection leads to new methods, and these are mainly discourse driven models. Some of the summarization methods encode discourse analysis results in feature presentations together with other frequency based features for sentence selection/compression. The problem is that these discourse based features usually play secondary roles, because the models all try to improve information coverage, which are evaluated by ROUGE. And ROUGE, as is commonly known, is not sensitive to coherence.

Some others work directly on discourse analysis results, and they usually try to derive a passage from a given parse tree. The problem of summarization is regarded as finding a text T so that $T = \arg\max F(T|Tr)$ for a given tree Tr. Here F is the objective function. Early representative work of this kind includes that of Marcu (1998) and that of Daumé III and Marcu (2002). Recently, Hirao et al. (2013) has viewed summarization as a knapsack problem on trees, and uses an integer linear problem (ILP) to formulate it. A sub tree that maximizes some objective function and obeys some given constraints is extracted from the original parse tree as the summary.

Discourse tree based methods cannot be extended to multi-document summarization. Christensen et al. (2013) propose a graph model that bypasses the tree constraints. They build a graph to represent discourse relations between sentences and then extract summaries accordingly.

Recently the neural network based discourse analysis (Li et al., 2014; Ji and Eisenstein, 2014) provides

us with an alternative way of conducting discourse analysis without traditional feature engineering. It can be used in our future work of modelling coherence using semantic relations.

6 Conclusion

Previous summarization methods have usually focused on salience and neglected coherence. This work proposed a novel summarization system that combines coherence with salience. By taking entities and links between them into consideration, our weighted longest path model successfully improves the quality of summaries. The proposed model does not require discourse analysis and hence can be applied to languages which do not enjoy plenty of ready-to-use discourse analysis tools.

In this paper only syntactic linkages are used for modelling coherence. In the future, we can take advantage of the semantic relations between entities to evaluate coherence and to further improve our system.

References

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Regina Barzilay, Noemie Elhadad, and Kathleen R McKeown. 2001. Sentence ordering in multidocument summarization. In *Proceedings of the 1st International Conference on Human Language Technology Research*, pages 1–7. Association for Computational Linguistics.

Danushka Bollegala, Naoaki Okazaki, and Mitsuru Ishizuka. 2010. A bottom-up approach to sentence ordering for multi-document summarization. *Information Processing & Management*, 46(1):89–109.

Alexei Borodin. 2009. Determinantal point processes. *arXiv preprint arXiv:0911.1153*.

Jill Burstein, Joel Tetreault, and Slava Andreyev. 2010. Using entity-based features to model coherence in student essays. In *Proceeding of the 2010 Conference of the North American Chapter of the Association for Computational Linguistics ? Human Language Technologies*, pages 681–684. Association for Computational Linguistics.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336. Association for Computing Machinery.

Janara Christensen, Stephen Soderland Mausam, and Oren Etzioni. 2013. Towards coherent multi-document summarization. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, pages 1163–1173.

Hal Daumé III and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proceedings of the 40th Annual Meeting of Computational Linguistics*, pages 449–456. Association for Computational Linguistics.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22(1):457–479.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. Event-based extractive summarization. In *Proceedings of the 42nd Annual Meeting of Computational Linguistics Workshop on Summarization*, volume 111.

Katja Filippova and Michael Strube. 2007. Extending the entity-grid coherence model to semantically related entities. In *Proceedings of the 11th European Workshop on Natural Language Generation*, pages 139–142. Association for Computational Linguistics.

Tao Ge, Wenzhe Pei, Heng Ji, Sujian Li, Baobao Chang, and Zhifang Sui. 2015. Bring you to the past: Automatic generation of topically relevant event chronicles. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL2015)*.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 10–18. Association for Computational Linguistics.

Dan Gillick, Benoit Favre, and Dilek Hakkani-Tur. 2008. The icsi summarization system at tac 2008. In *Proceedings of the Text Understanding Conference*.

Daniel Gillick, Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-Tur. 2009. A global optimization framework for meeting summarization. In *Acoustics, Speech and Signal Processing, 2009. IEEE International Conference on*, pages 4769–4772. IEEE.

Barbara J Grosz, Scott Weinstein, and Aravind K Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational linguistics*, 21(2):203–225.

Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-document summarization as a tree knapsack problem. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1515–1520. Association for Computational Linguistics.

Kai Hong, John M Conroy, Benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repository of state of the art and competitive baseline summaries for generic news summarization. In *LREC*, pages 1608–1616.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of Computational Linguistics*, pages 13–24.

Jiwei Li and Eduard H Hovy. 2014. A model of coherence based on distributed sentence representation. In *EMNLP*, pages 2039–2048.

Jiwei Li and Dan Jurafsky. 2016. Neural net models for open-domain discourse coherence. *arXiv preprint arXiv:1606.01545*.

Jiwei Li and Sujian Li. 2013. Evolutionary hierarchical dirichlet process for timeline summarization. In *ACL (2)*, pages 556–560. Citeseer.

Jiwei Li, Rumeng Li, and Eduard H Hovy. 2014. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 2061–2069.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics - Human Language Technologies*, pages 510–520. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL'04 Workshop*, pages 74–81.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Computational Linguistics: System Demonstrations*, pages 55–60.

Daniel Marcu. 1998. Improving summarization through rhetorical parsing tuning. In *The 6th Workshop on Very Large Corpora*, pages 206–215.

Ryan McDonald. 2007. *A study of global inference algorithms in multi-document summarization*. Springer.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, volume 4, page 275. Barcelona, Spain.

Naoaki Okazaki, Yutaka Matsuo, and Mitsuru Ishizuka. 2004. Improving chronological sentence ordering by precedence relation. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 750. Association for Computational Linguistics.

Karolina Owczarzak and Hoa Trang Dang. 2011. Overview of the tac 2011 summarization track: Guided task and aesop task. In *Proceedings of the 2011 Text Analysis Conference*.

Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Celebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, et al. 2004a. Mead-a platform for multidocument multilingual text summarization. In *Proceedings of the 4th Language Resources and Evaluation Conference*. Language Resources and Evaluation Conference.

Dragomir R Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004b. Centroid-based summarization of multiple documents. *Information Processing & Management*, 40(6):919–938.

Horacio Saggion and Robert Gaizauskas. 2004. Multi-document summarization by cluster/profile relevance and redundancy removal. In *Proceedings of the Document Understanding Conference*, pages 6–7.

Chao Shen and Tao Li. 2010. Multi-document summarization via the minimum dominating set. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 984–992. Association for Computational Linguistics.

Hiroya Takamura and Manabu Okumura. 2009. Text summarization model based on maximum coverage problem and its variant. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 781–789. Association for Computational Linguistics.

Marilyn A Walker, Aravind Krishna Joshi, and Ellen Friedman Prince. 1998. *Centering theory in discourse*. Oxford University Press.

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Annual Meeting-Association for Computational Linguistics*, volume 45, page 552.

Xun Wang, Lei Wang, Jiwei Li, and Sujian Li. 2012. Exploring simultaneous keyword and key sentence extraction: improve graph-based ranking using wikipedia. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 2619–2622. ACM.

Xun Wang, Yasuhisa Yoshida, Tsutomu Hirao, Katsuhito Sudoh, and Masaaki Nagata. 2015. Summarization based on task-oriented discourse parsing. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 23:1358–1367.

Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang. 2011. Evolutionary timeline summarization: a balanced optimization framework via iterative substitution. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 745–754. ACM.

Wen-tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. 2007. Multi-document summarization by maximizing informative content-words. In *IJCAI*, volume 7, pages 1776–1782.

Syntactic realization with data-driven neural tree grammars

Brian McMahan and **Matthew Stone**
Computer Science, Rutgers University
brian.mcmahan, matthew.stone@rutgers.edu

Abstract

A key component in surface realization in natural language generation is to choose concrete syntactic relationships to express a target meaning. We develop a new method for syntactic choice based on learning a stochastic tree grammar in a neural architecture. This framework can exploit state-of-the-art methods for modeling word sequences and generalizing across vocabulary. We also induce embeddings to generalize over elementary tree structures and exploit a tree recurrence over the input structure to model long-distance influences between NLG choices. We evaluate the models on the task of linearizing unannotated dependency trees, documenting the contribution of our modeling techniques to improvements in both accuracy and run time.

1 Introduction

Where natural language understanding systems face problems of ambiguity, natural language generation (NLG) systems face problems of choice. A wide coverage NLG system must be able to formulate messages using specialized linguistic elements in the exceptional circumstances where they are appropriate; however, it can only achieve fluency by expressing frequent meanings in routine ways. Empirical methods have thus long been recognized as crucial to NLG; see e.g. Langkilde and Knight (1998).

With traditional stochastic modeling techniques, NLG researchers have had to predict choices using factored models with handcrafted representations and strong independence assumptions, in order to avoid combinatorial explosions and address the sparsity of training data. By contrast, in this paper, we leverage recent advances in deep learning to develop new models for syntactic choice that free engineers from many of these decisions, but still generalize more effectively, match human choices more closely, and enable more efficient computations than traditional techniques.

We adopt the characterization of syntactic choice from Bangalore and Rambow (2000): the problem is to use a stochastic tree model and a language model to produce a linearized string from an unordered, unlabeled dependency graph. The first step to producing a linearized string is to assign each item an appropriate *supertag*—a fragment of a parse tree with a leaf left open for the lexical item. This process involves applying a learned model to make predictions for the syntax of each item and then searching over the predictions to find a consistent assignment for the entire sentence. The resulting assignments allow for many possible surface realization outputs because they can underdetermine the order and attachment of adjuncts. To finish the linearization, a language model is used to select the most likely surface form from among the alternatives. While improving the language model would improve the linearized string, we focus here on more accurately predicting the correct supertags from unlabeled dependency trees.

Our work exploits deep learning to improve the model of supertag assignment in two ways. First, we analyze the use of embedding techniques to generalize across supertags. Neural networks offer a number of architectures that can cluster tree fragments during training; such models learn to treat related structures similarly, and we show that they improve supertag assignments. Second, we analyze the use of tree recurrences to track hierarchical relationships within the generation process. Such networks can track more of the generation context than a simple feed-forward model; as a side effect, they can

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 224–235, Osaka, Japan, December 11-17 2016.

simplify the problem of computing consistent supertag assignments for an entire sentence. We evaluate our contributions in two ways: first, by varying the technique used to embed supertags, and then by comparing a feed-forward model against our recurrent tree model.

Our presentation begins in § 2 with an introduction to tree grammars and a deterministic methodology for inducing the elementary trees of the grammar. Next, § 3 presents the techniques we have developed to represent a tree grammar using a neural architecture. Then, in § 4, we describe the specific models we have implemented and the algorithms used to exploit the models in NLG. The experiments in § 5 demonstrate the improvement of the model over baseline results based on previous work on stochastic surface realization. We conclude with a brief discussion of the future potential for neural architectures to predict NLG choices.

2 Tree Grammars

Broadly, tree grammars are a family of tree rewriting formalisms that produce strings as a side effect of composing primitive hierarchical structures. The basic syntactic units are called elementary trees; elementary trees combine using tree-rewrite rules to form derived phrase structure trees describing complex sentences. Inducing a tree grammar involves fixing a formal inventory of structures and operations for elementary trees and then inferring instances of those structures to match corpus data.

2.1 Grammar Formalism

The canonical tree grammar is perhaps lexicalized tree-adjoining grammar (LTAG) (Joshi and Schabes, 1991). The elementary trees of LTAG consist of two disjoint sets with distinct operations: initial trees can perform substitution operations and auxiliary trees can perform adjunction operations. The substitution operation replaces a non-terminal leaf of a target tree with an identically-labeled root node of an initial tree. The adjunction operation modifies the internal structure of a target tree by expanding a node identically-labeled with the root and a distinguished foot note in the auxiliary tree. The lexicalization of the the grammar requires each elementary tree to have at least one lexical item as a leaf.

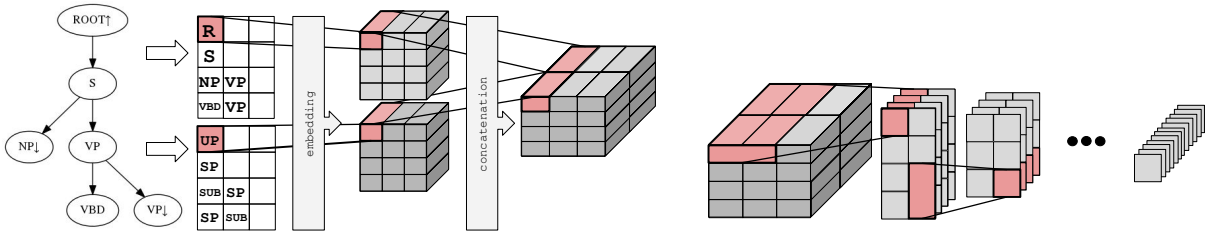
LTAG incurs computational costs because it is mildly context-sensitive in generative power. Several variants reduce the complexity of the formalism by limiting the range of adjunction operations. For example, the Tree Insertion Grammar allows for adjunction as long as it is either a left or right auxiliary tree (Schabes and Waters, 1995). Tree Substitution Grammars, meanwhile, allow for no adjunction and only substitutions (Cohn et al., 2009). We adopt one particular restriction on adjunction, called sister-adjunction or *insertion*, which allows trees to attach to an interior node and add itself as a first or last child (Chiang, 2000). Chiang's sister-adjunction allows for the flat structures in the Penn Treebank while limiting the formalism to context-free power.

2.2 Grammar Induction

In lexicalized tree grammars, the lexicon and the grammatical rules are one and the same. The set of possible grammatical moves which can be made are simultaneously the set of possible words which can be used next. This means that inducing a tree grammar from a data set is a matter of inferring the set of constructions in the data.

We follow previous work in using bracketed phrase structure corpora and deterministic rules to induce the grammar (Bangalore et al., 2001; Chiang, 2000). Broadly, the methodology is to split the observed trees into the constituents which make it up, according to the grammar formalism. We use head rules (Chiang, 2000; Collins, 1997; Magerman, 1995) to associate internal nodes in a bracketed tree with the lexical item that owns it. We use additional rules to classify some children as complements, corresponding to substitution sites and root notes of complement trees; and other children as adjuncts, corresponding to insertion trees that combine with the parent node, either to the right or to the left of the head. This allows us to segment the tree into units of substitution and insertion.[1]

[1] One particular downside of deterministically constructing the grammar this way is that it can produce an excess of superfluous elementary trees. We minimize this by collapsing repeated projections in the treebank. Other work has provided Bayesian models for reducing grammar complexity by forcing it to follow Dirichlet or Pitman-Yor processes (Cohn et al., 2010)—an interesting direction for future work.



(A) Tree Node Embedding (B) Convolutional Filter Encoding

Figure 1: Embedding supertags using convolutional neural networks. In (A), a tree is encoded by its features and then embedded. In (B), convolutional layers are used to encode the supertag into a vector.

3 Neural Representations

The grammar induction of § 2 allows us to construct an inventory of supertags to match a corpus. For NLG, we also need to predict the most likely supertag for any lexical item given the generation context. We approach this problem using neural networks. In particular, this work makes two contributions to improve stochastic tree modeling with neural networks. First, we represent supertags as vectors through embedding techniques that enable to model to generalize over complex, but related structures. Second, we address the hierarchical dependence between choices using a recurrent tree network that can capture long-distance influences as well as local ones. We now describe these representations in more detail.

3.1 Embedding Supertags

Different supertags for the same word can encode differences in the item's own combinatorial syntax, differences in argument structure, and differences in word order. Accordingly, words have many related supertags, with substantial overlaps in structure, and, presumably, corresponding similarities in their patterns of occurrence. A traditional machine learning approach to supertag prediction would treat individual supertags as atoms for classification; generalizing across supertags would require linking model parameters to handcrafted features or back-off categories.

By contrast, neural techniques work by embedding such tokens into a vector space. This process learns an abstract representation of tokens that clusters similar items together and makes further predictions as a function of those items' learned features. The resulting ability to generalize across sparse data seems to be one of the most important reasons for the success of deep learning in NLP.

The simplest way to embed supertags is to treat each structure as a distinct token that indexes a corresponding learned vector. This places no constraints on the learned similarity function, but it also ignores the hierarchical structure of the elementary trees themselves. Previous work on deep learning with graph structures suggests convolutional neural networks can exploit similarities in structure (Kalchbrenner et al., 2014; Niepert et al., 2016). Thus, we developed analogous techniques to encode supertags based on their underlying tree structure. In particular, to embed a supertag, we embed each node, group the resulting vectors to form a tensor, and then summarize the tensor into a single vector using a series of convolutional neural networks.

Note that each elementary tree is a complex structure with nodes labeled by category and assigned a role that enables further tree operations. The root node's role represents the overall action associated with that elementary tree—either substitution or insertion. The remaining nodes either have the substitution point role or the spine role—they are along the spine from root to the lexical attachment point, and thus provide targets for further insertion.

We first embed each node independently, then combined the vectors to form a tensor of embeddings. Specifically, symbols representing the syntactic category and node roles are treated as distinct vocabulary tokens, mapped to integers, and used to retrieve a vector representation that is learned during training. The vectors are grouped into a tensor by placing the root node into the first cell of the first row and left-aligning the descendants in the subsequent rows. The two tensors are concatenated along the embedding dimension. This embed-and-group method is shown in on the left in Figure 1.

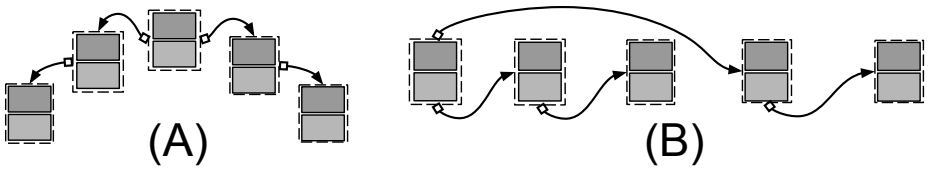


Figure 2: A recurrent tree network. (A) The dependency structure as a tree. (B) The dependency structure as a sequence.

Using a series of convolutional neural networks which learn their weights during training, the tensor of embeddings can be reduced to a single vector. To reduce the tensor to a vector, the convolutions are designed with increasingly larger filter sizes. Additionally, the dimensions are reduced alternatingly to also facilitate the capture of features. The entire process is summarized in Eq. 1 with Λ representing the supertags, G representing embedding matrices, and C representing the convolutional neural network layers. Specifically, G_s is the syntactic category embedding matrix and G_r is the node role embedding matrix. Each convolutional layer C is shown with its corresponding height and width as $C^{i,j}$. The encoding first constructs the tensor, T_Λ, through the embed-and-group method. Then, the embedding matrix G_Λ is summarized from T_Λ using the series of convolutional layers.

$$\begin{aligned} T_\Lambda &= [G_s(\Lambda_{syntactic\ category});\ G_r(\Lambda_{role})] \\ G_\Lambda &= C^{4,5}(C^{3,1}(C^{1,3}(C^{2,1}(C^{1,2}(T_\Lambda))))) \end{aligned} \tag{1}$$

The final product, a vector per supertag, is aggregated with the other vectors and turned into an embedding matrix. This is visualized in on the right in Figure 1. During training and test time, supertags are simply input as indices and their feature representations retrieved as an embedding.

3.2 Recurrent Tree Networks

Our models predict supertags as a function of the target word and its context. Neural networks make it possible to generalize over such contexts by learning to represent them with a hidden state vector that aggregates and clusters information from the relevant history. Our approach is to do this using a recurrent tree network. While recurrent neural networks normally use the previous hidden state in the sequential order of the inputs, recurrent tree networks use the hidden state from the parent. Utilizing the parent's hidden state rather than the sequentially previous hidden state, the recurrent connection can travel down the branches of a tree. An example of a recurrent tree network is shown in Figure 2.

In our recurrent tree network, child nodes gain access to a parent's hidden state through an internal *tree state*. During a tree recurrence, the nodes in the dependency graph are enumerated in a top-down traversal. At each step in the recurrence, the resulting recurrent state is stored in the tree state at the step index. Descendents access the recurrent state using a topological index that is passed in as data.

The formulation is summarized in Equation 2. The input to each time step in the current tree is the data, x_t, and a topological index, p_t. The recurrent tree uses p_t to retrieve the parent's hidden state, s_p, from the tree state, S_{tree}, and applies the recurrence function, $g(\dot{)}$. The resulting recurrent state is the hidden state for child node, s_c. The recurrent state s_c is stored in the tree state, S_{tree}, at index t.

$$\begin{aligned} s_c &= RTN(x_t, p_t) \\ &= g(x_t, S_{tree}[p_t]) \\ &= g(x_t, s_p) \\ S_{tree}[t] &= s_c \end{aligned} \tag{2}$$

The use of topological indices allows for many recurrent tree networks to be run in parallel on a GPU for efficiency. GPU implementations must be formulated homogeneously so that the same operations are applied across the entire data structure. Normally, tree operations involve conditional access to parent nodes, but using topological indices and a tree state accesses the parent in a homogeneous way.

4 Models

To analyze the representations we describe in § 2 and § 3, we developed two alternative architectures for predicting supertags in context. The first is a feed-forward neural network designed to solve a closely analogous task to the supertagging step of Bangalore and Rambow (2000)'s original FERGUS model. We call it Fergus-N (for Neuralized). The second uses a recurrent tree network to model the generation context. Because it has this richer context representation, it takes advantage of a slightly different characterization of the supertag prediction problem to streamline the problem solving involved in using the model. We call this Fergus-R (for Recurrent).

For both stochastic tree models, a recurrent neural network language model is used to complete the linearization task. The same language model is used to eliminate the confound of language model performance and measure performance differences in the stochastic tree modeling.

4.1 Model 1: Fergus-N

Fergus-N is a stochastic tree model which uses local parent-child information as inputs to a feed-forward network. Each parent-child pair is treated as independent of all others. The probability of the parent's supertag is predicted using an embedding of the pair's lexical material and an embedding of the child's supertag. (Our experiments compare the different embedding options surveyed in § 3.) Training maximizes the likelihood of the training data according to the model. Formally, our objective is to minimize the negative log probability of the observed parent supertags for each parent-child pair, as formally defined in Eq. 3.

$$min_\theta - [\sum_p \sum_{p \rightarrow c} log[P_\theta(tag_p | lex_p, lex_c, tag_c)] + \sum_c log[P_\theta(tag_c | lex_p, lex_c)]] \tag{3}$$

Here tag_p is the parent supertag, tag_c is the child supertag, lex_p is the parent's lexical material, and lex_c is the child's lexical material. Note that the probability of supertags for the leaves of the tree are computed with respect to their parent's lexical material.

The model is implemented as a feed-forward neural network. Equation 4 details the model formulation. The lexical material, lex_p and lex_c, are embedded using the word embedding matrix, G_w, concatenated, and mapped to a new vector, ω_{lex}, with a fully connected layer, FC_1. The child supertag, tag_c, is embedded using the target supertag embedding G_s and concatenated with the lexical vector, ω_{lex}, forming an intermediate vector representation of the node, ω_{node}. The node vector is repeated for each of the parent's possible supertags, $tagset_p$, and then concatenated with their embeddings to construct the set of treelet vectors, $\Omega_{treelet}$. The vector states for the leaf nodes are similarly constructed, but instead combine the lexical vector, ω_{lex} with the embeddings of the child's possible supertags, $tagset_c$. The final operation induces a probability distribution over the treelet and leaf vectors using a score computed by the vectorized function, $\Psi_{predict}$, as the scalar in a softmax distribution.

$$\begin{aligned}
&\omega_{lex} = FC_1([G_w(lex_p); G_w(lex_c)]) \\
&\omega_{node} = concat([G_s(tag_c);\ \omega_{lex}]) \\
&\Omega_{treelet} = concat([repeat(\omega_{node}),\ G_s(tagset_p)]) \\
&\Omega_{leaf} = concat([repeat(\omega_{lex}),\ G_s(tagset_c)]) \\
&P_\theta(tag_{p,i} | lex_p, lex_c, tag_c) = \frac{exp(\Psi_{predict}(\omega_{treelet_i}))}{\sum_{j \in |tagset_p|} exp(\Psi_{predict}(\omega_{treelet_j}))} \\
&P_\theta(tag_{c,i} | lex_p, lex_c) = \frac{exp(\Psi_{predict}(\omega_{leaf_i}))}{\sum_{j \in |tagset_c|} exp(\Psi_{predict}(\omega_{leaf_j}))}
\end{aligned} \tag{4}$$

At generation time, we are given a full dependency tree. A decoding step is necessary to compute a high probability assignment for all supertags simultaneously. In this process, tags for children must be chosen consistently with one another, and the resulting probabilistic information must be propagated upward to rerank tags elsewhere in the tree. We solve this problem with an A* algorithm. At each step,

the algorithm uses a priority queue to select subtrees based on their inside-outside scores. The inside score is computed as the sum of the log probabilities of the supertags in the subtree. The outside score is the sum of the best supertag for nodes outside the subtree, similar to Lewis and Steedman (2014). Once selected, the subtree is attached to the possible supertags of its parent that are both locally consistent and consistent among its already attached children. These resulting subtrees are placed into the priority queue and the algorithm iterates to progress the search. The search succeeds when the first complete tree has been found.[2]

4.2 Model 2: Fergus-R

Fergus-R is a stochastic tree model implemented in a top-down recurrent tree network and augmented with soft attention. For each node in the input dependency tree, soft attention—a method which learns a vectorized function to weight a group of vectors and sum into a single vector—is used to summarize its children. The soft attention vector and the node's embedded lexical material serve as the input to the recurrent tree. The output of the recurrent tree represents the vectorized state of each node and is combined with each node's possible supertags to form prediction states. Importantly, removing the conditional dependence on descendents' supertags results in the simplified objective function in Eq. 5 where lex_C is the children's lexical information, lex_p is the parent's lexical information, tag_p is the supertag for the parent node, and RTN is the recurrent tree network.

$$min_\theta - [\sum_{(p,C)} P_\theta(tag_p | RTN,\ lex_p,\ lex_C)] \tag{5}$$

The Fergus-R model uses only lexical information as input to calculate the probability distribution over each node's supertags. The specific formulation is detailed in Eq. 6. First, a parent node's children, lex_C, are embedded using the word embedding matrix, G_w, and then summarized with an attention function, Ψ_{attn}, to form the child context vector, ω_C. The child context is concatenated with the embedded lexical information of the parent node, lex_p, and mapped to a new vector space with a fully connected layer, FC_1, to form the lexical context vector, ω_{lex}. The context vector and a topological vector for indexing the internal tree state (see § 3.2) are passed to the recurrent tree network, RTN, to compute the full state vector for the parent node, ω_{node}. Similar to Fergus-N, the state vector is repeated and concatenated with the vectors of the parent node's possible supertags, $tagset_p$, and mapped to a new vector space with a fully connected layer, FC_2. A vector in this vector space is labeled $\omega_{elementary}$ because the combination of supertag and lexical item constitutes an elementary tree. The last step is to compute the probability of each supertag using the vectorized function, $\Psi_{predict}$.

$$\begin{aligned}
&\omega_C = \Psi_{attn}(G_w(lex_C)) \\
&\omega_{lex} = FC_1(concat(\omega_C,\ G_w(lex_p))) \\
&\omega_{node} = RTN(\omega_{lex},\ topology) \\
&\Omega_{elementary} = FC_2(concat(repeat(\omega_{node}),\ G_s(tagset_p))) \\
&P_\theta(tag_{p,i} \mid RTN,\ lex_p,\ lex_C) = \frac{exp(\Psi_{predict}(\omega_{elementary_i})))}{\sum_{j \in |\Omega|} exp(\Psi_{predict}(\omega_{elementary_j}))}
\end{aligned} \tag{6}$$

Although the same A* algorithm from Fergus-N is used, the decoding for Fergus-R is far simpler. As supertags are incrementally selected in the algorithm, the inside score of the subsequent subtree is computed. Where Fergus-N had to compute a incremental dynamic program to evaluate the inside score, Fergus-R decomposes into a sum of conditionally independent distributions. The resulting setup is a chart parsing problem where the inside score of combining two consistent (non-conflicting) edges is just the sum of their inside scores.

[2] Although, the data has some noise so that sometimes there is no complete tree that can possibly be formed.

4.3 Linearization

The final step to linearizing the output of Fergus-N and Fergus-R—a dependency tree annotated with supertags and partial attachment information—is a search over possible orderings with a language model. There are many possibilities, primarily due to ambiguities in insertion order. Following Bangalore and Rambow (2000), a language model is used to select between the alternate orderings. The language model used is a two-layer LSTM trained using the Keras library on the surface form of the Penn Treebank. The surface form was minimally cleaned[3] to simulate realistic scenarios.

The difficulty of selecting orderings with a language model is that the possible linearizations can grow exponentially. In particular, our implementations result in a large amount of insertion trees.[4] We approach this problem using a prefix tree which stores the possible linearizations as back-pointers to their last step and the word for the current step. The prefix tree is greedily searched with 32 beams.

5 Experiments

Using the representations of § 3, the models of § 4 can be instantiated in six different ways. We can use a feed-forward Fergus-N architecture or a recurrent Fergus-R architecture. Each architecture can embed supertags *minimally*, by learning a scalar corresponding to each supertag; *atomically*, by learning an embedding vector corresponding to each supertag; or *structurally*, by using convolutional coding over each supertag's tree structure to form a vector. In each case, the vector (a size-one vector in the minimal condition) is concatenated as described in § 4.

5.1 Training

We trained six such models using a common experimental platform. We started from the Wall Street Journal sections of the Penn Treebank, which have been previously used for evaluating statistical tree grammars (Chiang, 2000).[5] Our data pipeline breaks each sentence in the treebank into component elementary trees and then represents the sentence in terms of a derivation tree, specifying the tree-rewriting operations required to construct the actual treebank surface tree from the basic supertags. Removing supertags from the derivation tree leads to the unlabeled dependency trees our models assume as input.

From this input, we extracted the atomic supertag prediction instances and trained a network defined by each of the architectures of § 4 and each of the supertag representations of § 3. As always, we used Sections 02-21 for training, Section 22 for development, and Section 23 for testing. A complete description of network organization and training parameters is given in the appendix. The code and complete experimental setup are publicly available.[6]

5.2 Performance Metrics

We evaluate the performance of the models in several ways. First, we look at the accuracy of the supertag predictions directly output by each model. Second, we look at the accuracy of the final supertags obtained by decoding the model predictions to the best-ranked consistent global assignment. These metrics directly assess the ability of the models to successfully learn the target distributions.

Next, we evaluate the models on the full NLG task, including linearization. The linearization task allows more freedom in supertag classifications because supertags may differ in minor ways, such as the projections present along the spine, which will not affect generation output for a particular target input. The freedom means models may not be penalized based on decisions that don't matter—thus, at the same time, it also mutes the distinctions between classification decisions. We report a modified edit distance measure, Generation String Accuracy, following (Bangalore et al., 2000). Since linearization uses a beam search, we report statistics both for the top-ranked beam and for the empirically based beam among the candidates computed during search. The difference gives an indication of the effect of the language model in guiding the decisions that remain after supertagging.

[3] With respect to the surface form, the only cleaning operations were to merge proper noun phrases into single tokens. Punctuation and other common cleaning operations were not performed.

[4] Many of the validation examples had more than 2^{40} possible linearizations.

[5] A possible additional data source, the data from the 2011 Shared Task on Surface Realization, was not available.

[6] `https://github.com/braingineer/neural_tree_grammar`

		Accuracy		
Model	**Embedding**	**Raw Model**	**After Decoding**	**Running Time**
Fergus-N	Structural	58.17%	57.40%	1.97s
	Atomic	60.69%	55.56%	1.81s
	Minimal	52.09%	54.18%	2.02s
Fergus-R	Structural	67.62%	57.04%	0.30s
	Atomic	82.65%	62.73%	0.36s
	Minimal	10.13%	19.66%	0.54s

Table 1: For each supertag and embedding pair, the mean accuracy of supertag classification directly output by the model and in the consistent global assignment output by A* decoding. Also shown is the median running time—which includes model computation and A* search. The structural embeddings are computed with convolutional coding, the atomic embeddings as rows in a matrix, and the minimal embeddings as scalars in a vector.

Finally, we report statistics about the run time of different generation steps. This allows us to assess the complexity of the different decoding steps involved in generation, to reveal any tradeoffs among the models between speed and accuracy.

5.3 Results

Table 1 shows the results of supertag prediction. All differences between model are significant using a Paired-Sample t-test ($p < 10^{-5}$) The structural and atomic embedding methods consistently perform better, suggesting that the clustering capabilities of neural methods is a crucial part of their effectiveness. For post-decoding performance, Fergus-N utilizes the structural embeddings more than the atomic embeddings. This merits further investigation: it might be because Fergus-N predicts one supertag as a function of another, and so the compositional relationships among the two trees are more important—or because Fergus-R's contextualized decisions depend on similarities among supertags (involving argument structure or information structure) that are difficult for the convolutional coding to represent or learn. Additionally, the minimal embeddings suggests that Fergus-N's architecture might provide enough structure to reduce the difficulty of a large number of cases.

The overall best results come from Fergus-R, suggesting that it is worthwhile to take additional context into account in this task. At the same time, the median time taken to classify and decode a sentence with Fergus-R is just one sixth that of Fergus-N. We suspect that there is a general lesson in this speedup: because neural models can be more flexible about the information they take into account in decisions, it's especially advantageous in designing neural architectures to break a problem down into decisions that can be combined easily.

Finally, decoding the network generally leads to lower accuracy. It seems that our models are not doing a good job of using the predictions they make to triangulate to accurate and consistent supertags. This suggests that the models could be improved by taking more or better information into account in decoding. This is more pronounced in the atomic embeddings than the structural embeddings, which suggests that the lack of structure in the vector representation allows for the model to learn clustering relationships that don't correlate with the structural requirements.

Figure 2 shows the NLG evaluation results for the different models. All differences in model are significant using an Independent t-test ($p < 10^{-5}$). [7] For both models, the differences between structural embeddings (using convolutional coding) and atomic embeddings (using standard vector embedding techniques) were not significant, while the differences between the two embeddings and minimal embeddings were significant ($p < 10^{-5}$). The performance confirms our expectation that differences in supertag accuracy after decoding correlate with NLG accuracy overall, but that differences in NLG performance are attenuated. We note by comparison that Bangalore and Rambow report an accuracy of

[7] An Indepedent t-test was used instead of a Paired-Sample t-test because of intermittent failures during linearization that resulted in slightly different numbers of observations.

		Accuracy	
Model	**Embedding**	**Top Scoring**	**Best Performance**
Fergus-N	Structural	65.80%	72.58%
	Atomic	65.52%	71.82%
	Minimal	63.79%	71.09%
Fergus-R	Structural	68.22%	74.70%
	Atomic	69.29%	75.56%
	Minimal	58.23%	65.04%

Table 2: Shown above as accuracy is the percentage of tokens in the linearized strings that are in correct positions according to an edit distance measure.

74.9% in their best evaluation of FERGUS—on a data set of just 100 sentences with an average length of 16.7. Our evaluation, on 2400 sentences with an average length of 22.1, is more strenuous.

6 Related Work

There are several lines of related work which explore stochastic tree models from the standpoint of parsing and understanding. While using the same methods, NLG has different goals and we think the perspective is instructive. Where parsing infers the most probable underlying structure, generation infers the most likely way of expressing a semantic structure. This divergence of goals leads to different concerns, alternatives, and emphasis.

The works most similar to ours explicitly model tree structures, but focus on resolving the uncertainty involved with the latent structure of an observed sentence. For example, the top down tree structure of Zhang et al. (2016) expresses the generation of a dependency structure as the decisions of a set of long short-term memory networks. For each decision, the possible options are different tree structures which can produce the target linear form. In contrast, the generation problem is concerned with different linear forms that can result from the same tree structure. In more extensive tasks, the generation problem can include simulated interpretation to inform decisions; using the ease of structural inference from linear form quantifies the understandability of a sentence.

Although the methodology presented in this work is closely related to several recent neural networks models for long-distance relationships, it differs distinctly in its treatment of state and search. Specifically, forward-planning in a generation task produces a growing horizon of syntactic choices while shrinking the horizon of semantic goals. At each step, syntactic operations grow the number of available syntactic choices while limiting the number of semantic goals left to express. In contrast, parsing and understanding begin with the surface form and construct the organized semantic content, either for a downstream decision or just for the structure itself. The most notable works in this line of research are the recurrent neural network grammars (Dyer et al., 2016), a shift-reduce parser and interpreter (Bowman et al., 2016), and a dynamic network for composing other neural network modules (Andreas et al., 2016). Interestingly, there is a common theme of using indexable and dynamic data structures in neural architecture to make long-distance decisions.

7 Conclusion

This paper has explored issues in deep learning of probabilistic tree grammars from the standpoint of natural language generation. For NLG, we need models that predict high-probability structures to encode deep linguistic relationships—rather than to infer deep relationships from surface cues. This problem brings new challenges for learning, as it requires us to represent new kinds of linguistic elements and new kinds of structural context in order to capture the regularities involved. Despite these challenges, however, the problem continues to have the mix of data sparsity, rich primitives and combinatorial interactions that has made deep learning attractive for use in natural language parsing and understanding.

Of the range of models we surveyed here, the best combines a top down tree recurrence to cluster contexts with appropriate embedding methods to cluster syntactic and lexical elements. Our evaluations

suggest that the model is more accurate and faster than alternative techniques. However, it would still be good to analyze the performance of the model more deeply. Can we get better results in the key decoding step? How do human readers find the output of the system?

Looking forward, we see this research a step towards learned models that capture more of the NLG task. We plan to explore similar techniques in planning surface text from more properly semantic inputs or even from abstract communicative goals. Further, we plan to integrate learned methods with knowledge-based techniques to offer designers more control over system output in specific applications. Developing methods appropriate to such settings will require researchers to revisit the core problems of generalizing across linguistic structures and contexts—and, we hope, to build on and extend the provisional solutions we have explored here.

Acknowledgments

This research was supported in part by NSF IIS-1526723 and by a sabbatical leave from Rutgers to Stone.

A Appendix

All of our models were implemented in the Keras (Chollet, 2015) and Theano (Theano Development Team, 2016) libraries. The specific parameters that were used are shown in Table 3. The parameters were selected by measured performance on the development portion of the data set. In the accompanying code repository, the full experiment parameters—including programmatic parameters controlling the experimental design—are specified in configuration files.

In our experiments, the corpus was preprocessed using Stanford NLP tools (De Marneffe et al., 2006) to fix common issues and remove extraneous information. The resulting parse trees were then analyzed to mark the head words, the dependents, and the adjuncts. The marked-up trees were split at adjunction and substitution positions to form the grammar. Our models use an output distribution that's restricted to the set of supertags that have occurred with the lexical item, which requires indices to the supertag embedding matrix to be passed into the computation with the rest of the data. We implement the affinity matrix between the supertag embeddings and lexical state vectors, by concatenating the vectors, mapping them to a new space using a fully connected layer, and computing a score with a vectorized function. (The vectorized function operation is the same mechanism which calculates the probability distribution used in soft attention.)

Model Parameter	Value
Fergus-N Parameters	
Fully connected layer size	256
Batch size	128
Fergus-R Parameters	
Fully connected layer size	256
Hidden state size	128
Batch size	16
Embedding Parameters	
Convolution filter size	48
Syntactic category embedding size	32
Node role embedding size	32
Word embedding size (Pennington et al., 2014)	300

Model Parameter	Value
Language Model Parameters	
Hidden state size	368
Batch size	32
Optimization Parameters	
Optimization Algorithm	ADAM
Fergus-R and Fergus-N Learning Rate	1e-4
Language Model Learning Rate	0.01
Fully-Connected Dropout Rate	0.5
Recurrent Weight Dropout Rate	0.2
L_2 Weight Decay	1e-6
Max gradient norm	10.0
Gradient clip threshold	5.0

Table 3: The parameters for the Fergus-R, Fergus-N, and language models. The exact specifications in configuration files can be found in the code repository that accompanies this paper.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1545–1554. Association for Computational Linguistics.

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 42–48. Association for Computational Linguistics.

Srinivas Bangalore, Owen Rambow, and Steve Whittaker. 2000. Evaluation metrics for generation. In *Proceedings of the first international conference on Natural language generation-Volume 14*, pages 1–8. Association for Computational Linguistics.

Srinivas Bangalore, John Chen, and Owen Rambow. 2001. Impact of quality and quantity of corpora on stochastic generation. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Langauge Processing, Pittsburgh, PA*.

R. Samuel Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, D. Christopher Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1466–1477. Association for Computational Linguistics.

David Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 456–463. Association for Computational Linguistics.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 548–556. Association for Computational Linguistics.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2010. Inducing tree-substitution grammars. *The Journal of Machine Learning Research*, 11:3053–3096.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23. Association for Computational Linguistics.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and A. Noah Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209. Association for Computational Linguistics.

Aravind K Joshi and Yves Schabes. 1991. Tree-adjoining grammars and lexicalized grammars.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665. Association for Computational Linguistics.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 704–710. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. 2014. Improved CCG parsing with semi-supervised supertagging. *Transactions of the Association for Computational Linguistics*, 2:327–338.

David M Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 276–283. Association for Computational Linguistics.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. *arXiv preprint arXiv:1605.05273*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Yves Schabes and Richard C. Waters. 1995. Tree Insertion Grammar : A Cubic-Time , Parsable Formalism that Lexicalizes Context-Free Grammar without Changing the Trees Produced. *Computational Linguistics*, 21(4):479–513.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016. Top-down tree long short-term memory networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 310–320, San Diego, California, June. Association for Computational Linguistics.

Abstractive News Summarization based on Event Semantic Link Network

Wei Li[1,+], **Lei He**[2,*], **Hai Zhuge**[3,+,*]
[+]Key Lab of Intelligent Information Processing, Institute of Computing Technology,
University of Chinese Academy of Sciences, Beijing, China
[*]Aston University, Birmingham, UK
[1]weili.ict.kg@gmail.com, [2]hel2@aston.ac.uk, [3]zhuge@ict.ac.cn

Abstract

This paper studies the abstractive multi-document summarization for event-oriented news texts through event information extraction and abstract representation. Fine-grained event mentions and semantic relations between them are extracted to build a unified and connected event semantic link network, an abstract representation of source texts. A network reduction algorithm is proposed to summarize the most salient and coherent event information. New sentences with good linguistic quality are automatically generated and selected through sentences over-generation and greedy-selection processes. Experimental results on DUC 2006 and DUC 2007 datasets show that our system significantly outperforms the state-of-the-art extractive and abstractive baselines under both pyramid and ROUGE evaluation metrics.

1 Introduction

Automatic summarization on news documents enables readers more easily to get general information of interesting news. Most of existing summarization methods have neglected the important event-oriented characteristics of news texts although some popular tasks such as DUC (Document Understanding Conference) and TAC (Text Analysis Conference) target at summarizing news documents. The examples below show that the core information of news texts is the atomic event mentions as shown in bolded words and their related concepts as shown in italic phrases.

- Lawyer *Morris Dees*, who is **representing** *Victoria Keenan* after she was **attacked** by *two guards* in *July 1998*, **introduced** *depositions* to **contradict** *the men's testimony*.
- *Morris S. Dees Jr.*, who was the co-founder of the Southern Poverty Law Center, **defended** for *Keenan* after she was **assaulted** by *two security guards* near *the headquarters of the Aryan Nations*.

An event usually tells us "**who** did **what** to **whom when** and **where** ..." The most important components of an event include its **actor** (who, the agent of the event), **action** (what, the core meaning of the event) and **receiver** (whom, the target of the event action). Other arguments indicate other attributes of the event, such as **time** (when) and **location** (where). The event arguments are concepts related with the event action. For event-based news summarization, extracting the most salient events and related concepts are the core tasks.

One of the most similar related work (Glavaš and Šnajder, 2014) investigated constructing event graph for multi-document summarization. The nodes in event graph denote event mentions while edges denote temporal relations between event mentions. It ranks event mentions based on the temporal relations and then generates summary by extracting sentences that contain salient event mentions. However, the problems of information overlapping and lacking of coherence cannot be overcomed by extractive methods. This paper explores the issue of abstractive summarization for event-oriented news texts. The semantic relations between events like cause-effect relation are also extracted to help generate more coherent and informative summary in our system.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 236–246, Osaka, Japan, December 11-17 2016.

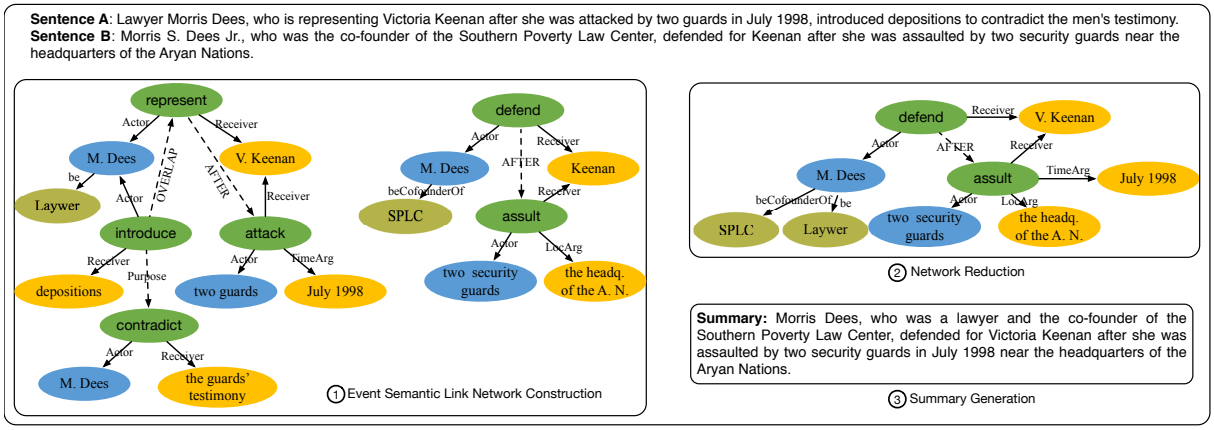


Figure 1: An example illustrating the framework of our summarization system.

Figure 1 illustrates the procedure of our system. Firstly, the semantic information of texts is represented by constructing event semantic link network (Zhuge, 2012). The semantic nodes of the network are events extracted from the source texts while semantic links are relations between events. Concept co-reference resolution and event co-reference resolution are both conducted within and cross documents to aggregate information from different places. Secondly, the event semantic link network is reduced to obtain connected and condensed summary network. A network reduction algorithm that makes use of the semantic links between event nodes is proposed to trade off among selecting salient information, maintaining coherence, and conveying correct and complete information. Finally, coherent and concise summary is automatically generated based on the summary network through sentences over-generation and greedy selection processes. The contributions of this work include:

- The abstractive summarization for event-oriented news texts is made by extracting fine-grained events and constructing event semantic link network as the abstract representation of source texts.
- An ILP-based network reduction algorithm using semantic links between events is proposed to obtain the most condensed, salient and coherent semantic information of source texts.
- Informative and concise summary is automatically generated based on the event semantic link network after reduction.

2 Event Semantic Link Network Construction

As shown in Figure 1, the first procedure of our system is to extract events and construct event semantic link network (ESLN). Within ESLN, semantic nodes are event mentions consisting of event actions and arguments. The action indicates the central meaning of an event, while the arguments render the attributes of an event (Ahn, 2006). In this work, each event is represented as a tree with the event action as its root node. The children of the root node are event arguments, including actor, receiver, time and location. The collapsed form of an event tree can be denoted as *e=Action (Actor, Receiver, TimeArg, LocArg)*. We use semantic relations between events as semantic links (subsection 2.3). ESLN provides an event-based abstract representation for news documents, which is a directed and connected graph.

The ESLN is constructed by: (1) extracting concepts from documents; (2) identifying event actions and extracting event arguments; (3) predicting the semantic links between event mentions.

2.1 Concept Extraction

All noun phrases extracted from documents are defined as concepts. To enrich the semantics of a concept, we model it as an object which consists of its core noun phrase and attributes. The attributes of a concept reflect the relationships between this concept and other concepts. A concept a implied by its core noun phrase is denoted by $a(\xrightarrow{r_1} c_1, \xrightarrow{r_2} c_2, \cdots, \xrightarrow{r_n} c_n)$ where c_i indicates another concept and r_i indicates a specific relation between concept a and concept c_i. Concept c_i is defined as an attribute of concept a.

Lexical features	word, lemma of the token and its surrounding tokens (five tokens to the left and right)
POS-tag features	part-of-speech tags of the token and its surrounding tokens (five to the left and right)
Syntactic features	the set of dependency relations of the token
Modifier features	modal modifiers, auxiliary verbs and negations.
Word vectors	100-dimensional GloVe word vector (Pennington et al., 2014)

Table 1: The features for the event identification model

Position features	the set of features that measure the distance between event actions (number of tokens) and their relative position (same sentence, adjacent sentences, adjacent event mentions)
Lexical features	word, lemma, stem, and pos-tag of both event actions as well as features indicating whether the word forms are the same, the semantic similarity between actions words, the word and lemma of each token between the action words
Syntactic features	syntactic path between the actions (dependency labels on the syntactic path between the actions), features indicating whether one action syntactically dominates the other, features indicating whether one is a predicate of an adverbial clause governed by the other event, and the set of dependency relations of both actions
Modifier features	the set of features that describe the modal, auxiliary, negation, and determination modifiers of both event actions
Word vectors	100-dimensional GloVe word vector of both event action words
Discourse features	the discourse relations between event mentions. We use the document-level discourse analysis method (Joty et al., 2013) to extract the discourse relations between event mentions.

Table 2: The features for the event relation prediction model

We extract concepts and their attributes based on dependency trees. Texts are preprocessed by Stanford CoreNLP pipeline (Manning et al., 2014). The dependency trees are transformed into semantic graph by pronoun resolution (Schuster et al., 2015). All named entities are identified as concepts. For other nouns, we expand on "`compound`", "`name`", "`amod`", "`neg`", "`nummod`" and "`dep`" dependency edges to build the basic noun-phrase concept. We also expand on "`appos`", "`acl`", "`acl:relcl`", "`nmod:of`" and "`nmod:poss`" edges for non-proper nouns, since these are relative clauses that convey important information.

To extract the attributes of a concept, we extract the relations between the concept and other related concepts. In order to differentiate with event actions, the valid syntactic patterns of relations between the head concept and its attributes is restricted as "`be`", "`be-NP-prep`" and "`be-AP-prep`" where NP indicates noun phrase and AP indicates adjective phrase, such as *"Morris Dees* **is** *a lawyer"* and *"Morris Dees* **is the co-founder of** *Southern Poverty Law Center"*. Several syntactic rules, which use the dependency labels (including "`nsubj`", "`appos`", "`nmod:of`" and "`nmod:poss`") between head tokens of concepts, are designed to detect those specific relations between concepts.

To aggregate information across documents, we need to recognize all concept co-references across documents. The co-reference resolution within single document has been conducted during the preprocessing stage by Stanford CoreNLP pipeline, so those resolution rules can be adopted. We formulate the co-references detection in a hierarchical agglomerative clustering framework similar to (Shen et al., 2013). A set of clusters are obtained and each cluster contains mentions refer to the same concept in the documents. For each cluster of co-referential concepts, we only reserve the most representative one and merge the attributes of all other mentions. For example, the concept *"Morris Dees"* in Figure 1.

2.2 Event Identification

The procedure of event identification consists of two steps: event action identification and event arguments extraction. The first step is formulated as a supervised classification task with features as shown in Table 1.

The arguments of an event are concepts related to the event action. Since we have extracted all concepts from documents, the argument extraction is to judge the argument type of each concept. We define in total fifteen dependency patterns using Semgrex expressions (Chambers et al., 2007). These patterns mainly capture the subject-predicate-object constructions, subject-predicate constructions, passive constructions, prepositional constructions and clausal constructions.

Since important events are usually mentioned many times in the documents. For example, in Figure 1 "*Victoria Keenan* was **attacked** by *two guards* in *July 1998*" and "*Keenan* was **assaulted** by *two security guards*" refer to the same event. To determine whether two event mentions are co-referential, both the event actions and event arguments are compared. We use WordNet-based similarity method (Pedersen et

al., 2004) to judge the semantic similarity between event actions. Two event mentions are identical only when the similarity between event actions is above a threshold (set as 0.8 after tuning) and corresponding event arguments are identical or co-referential. For all identical event mentions, we just reserve the most representative one and merge the relations and arguments of other mentions.

2.3 Event Relation Prediction

We leverage the sentence structures and discourse features in documents to infer the relations between events in order to construct an informative event semantic link network. Through analyzing large numbers of news texts, we find following types of semantic relations between events are very common:

- *Temporal link*. It indicates the temporal relations between two events, which consists of directed asymmetric links (BEFORE and AFTER) and symmetric links (OVERLAP). For symmetric links, we add two directed links in opposite directions between two event nodes;
- *Cause-effect link*, denoted by ce as in $e \xrightarrow{ce} e^{'}$, for which the predecessor event e is a cause of its successor event $e^{'}$ and the successor event $e^{'}$ is an effect of its predecessor event e.
- *Purpose link*, denoted by pur as in $e \xrightarrow{pur} e^{'}$, for which the successor event $e^{'}$ is the purpose of its predecessor event e. Event $e^{'}$ is to be realized through event e.
- *Means link*, denoted by mea as in $e \xrightarrow{mea} e^{'}$, for which the event $e^{'}$ is a method or instrument which tends to make realization of event e more likely.
- *Condition link*, denoted by con as in $e \xrightarrow{con} e^{'}$, for which the predecessor event e is a condition of its successor event $e^{'}$. Realization of $e^{'}$ depends on realization of event e.
- *Sequential link*, denoted by seq as in $e \xrightarrow{seq} e^{'}$, for which the event $e^{'}$ is a successor of event e. It usually describes a number of event actions with succession relationships.
- *Attribution link*, denoted by $attri$ as in $e \xrightarrow{attri} e^{'}$, for which event $e^{'}$ is an attribution of event e, indicating its specific contents.

For predicting the semantic links between each pair of event nodes, we use an L2-regularized maximum entropy classifier with features as shown in Table 2.

In order to make the event semantic link network denser and more informative, we add "*Common Argument*" links between event nodes that share the same concept as argument. For example, "*Morris Dees defended for Keenan*" and "*Morris Dees contradicted the men's testimony*" both use concept "*Morris Dees*" as actor argument. After expanding the semantic links between events, we get a unified, connected and informative ESLN to represent the abstract information of source texts.

3 Summarization

The constructed ESLN is an abstract representation of source documents. We summarize the documents by summarizing the network and generate summary based on the reduced network. For event-based summarization, the summary network must contain the most salient events and concepts information. We model the summarization of ESLN as a structured prediction problem (Collins, 2002) that trades off among selecting salient information, maintaining coherence, and conveying correct and complete information.

Let E and C denote all the event nodes and concepts in ESLN, where each node $e \in E$ represents a unique event and each concept $c \in C$ is an argument of an event. To obtain the most salient and condensed summary network, we seek to maximize the summation of saliency scores of the selected events and concepts. For summary network which contains event set $E^{'}$ and concept set $C^{'}$, its saliency score is:

$$\sum_{e \in E^{'}} \theta^T f(e) + \sum_{c \in C^{'}} \psi^T g(c) \quad (1)$$

where $f(e)$ and $g(c)$ represent the features of event e and concept c respectively (described in Table 3). θ and ψ are vectors of feature weights for events and concepts respectively.

The network reduction problem is decoded as an integer linear programming (ILP) by incorporating some priori knowledge as constraints (§3.1). Features weights are estimated by using structured pre-

Concept Features	Concept Type	binary feature indicates whether it's named entity and whether it appears in the topic description
	Concept Freq.	one binary feature for each frequency threshold t=0/1/2/5/10
	Concept Pos.	average and foremost position of sentences containing the concept (binarized by 5 thresholds)
	Concept Head	word, lemma, pos, depth in the dependency tree (binarized by 5 thresholds) and whether it appears in the topic description
	Concept Span	average and longest word span of concept (binarized by 5 thresholds)
Event Features	Action Word	word, lemma, pos, depth in the dependency tree (binarized by 5 thresholds) and whether it appears in the topic description
	Action Freq.	binary feature for each frequency threshold t=0/1/2/5/10, average and foremost position of sentences containing the concept
	Actor Arg.	all concept features of actor argument
	Receiver Arg.	all concept features of receiver argument. If don't contain receiver argument, all set as 0
	Time Argument	one binary feature indicates whether it contains time argument
	Location Arg.	one binary feature indicates whether it contains location argument
	Semantic Links	total number of links from and to the event node in event graph (binarized by 5 thresholds))

Table 3: Event and concept features (all binaries)

diction algorithm (§3.2). After obtaining the summary network, concise and coherent summary can be generated through sentences over-generation and greedy selection (§3.3).

3.1 Network Reduction

Let M and N be the total number of event nodes and concepts in source ESLN. We use e_i and c_j to represent the i-th event and j-th concept respectively. Let u_i and v_j be binary variables. u_i is set to 1 iff event e_i is selected and v_j is set to 1 iff concept c_j is selected.

The ILP maximization objective can be transformed into Equation 2, which contains two parts: the first part tends to select more important events; and the second part tends to select more concepts to increase information diversity and reduce redundancy in the summary.

$$\sum_{i=1}^{M} u_i \theta^T f(e_i) + \sum_{j=1}^{N} v_j \psi^T g(c_j) \tag{2}$$

To ensure the summary network could generate coherent summary and convey complete and correct information, the following groups of constraints are required:

Complete facts. To guarantee the selected event node convey complete fact, the following constraints are introduced:

$$\forall i, if c_j \in Arguments(e_i), v_j \geq u_i \tag{3}$$

$$\forall j, \sum_{i \in c_j.relatedEvents} u_i + \sum_{k \in c_j.attributes} v_k \geq v_j \tag{4}$$

$$\forall i, k, \text{if } e_i \xrightarrow{Attribution} e_k, u_i \leq u_k \tag{5}$$

Equation 3 ensures that if an event was selected, the arguments of the event should all be selected. Equation 4 guarantees that if a concept was selected, at least one event that it related to or an attribute that it has should be selected. These two constraints ensure the selected event or concept convey complete information. If event e_k is an attribution of event e_i, then e_k describes specific contents of event e_i. Equation 5 guarantees that if event e_i is selected, its attribution e_k must be selected.

Coherence. In order to generate coherent summary, the reduced summary network should be connected. Flow-based constraints have previously been used (Thadani and McKeown, 2013; Liu et al., 2015) to ensure the connectivity of subgraph. For each pair of event nodes e_i and e_k, the binary variable $l_{i,k}$ indicates the semantic link between them. Only if both e_i and e_k are selected and there is a link between them, $l_{i,k}$ can be set to 1, otherwise 0, which can be formulated as following:

$$\begin{aligned} &\forall i, k, l_{i,k} \leq e_i, l_{i,k} \leq e_k \\ &\text{if there is no link from } e_i \text{ to } e_k, l_{i,k} = 0 \end{aligned} \tag{6}$$

A set of single-commodity flow variables $f_{i,k}$ that each takes a non-negative integral value and represents the flow from event node e_i to e_k, were used to enforce the connectivity of summary network. We set a dummy "ROOT" node which is connected with only one selected event node in the ESLN (Equation 7), denoted as e_0 . The root node sends up to M units of flows to the selected event nodes (Equation 8). Each selected node consumes one unit of flow (Equation 9). Flow can only be sent over a link if and only if the link variable l is 1 (Equation 10).

CONCEPT DESCRIPTION RULES
For concept $a(\xrightarrow{r_1} c_1, \xrightarrow{r_2} c_2, \cdots, \xrightarrow{r_n} c_n)$, the description of concept a can be:
1.**Appositive modifier** "a, $c_1, c_2 \ldots$", e.g. *"Morris Dees, civil rights lawyer, co-founder of Souther Poverty Low Center, ..."*
2.**Attributive clause** *"a who/which/that $r_1 c_1 \ldots$"*, e.g. *"Morris Dees who was the co-founder of Southern Poverty Law Center and a civil rights lawyer ..."*
3.**Appositive modifier mixed with attributive clause**, e.g. *"Civil rights lawyer Morris Dees who was the co-founder of Southern Poverty Law Center ..."*
SENTENCE STRUCTURING RULES
if $e_1 \xrightarrow{after/before/overlap} e_2$, then generate "$e_1$ after/before/when e_2"; if $e_1 \xrightarrow{ce} e_2$,then generate "Because $e_1, e_2,$ " and "e_2 because e_1";
if $e_1 \xrightarrow{pur} e_2$,then generate "$e_1$ in order to e_2" and "e_1 so that e_2"; if $e_1 \xrightarrow{mea} e_2$, then generate "$e_1$ by e_2" and "e_1 by the way that e_2"
if $e_1 \xrightarrow{attri} e_2$, then generate "$e_1$ e_2", "e_1 about/on/in/with/at e_2" and "e_1 that e_2"; if $e_1 \xrightarrow{seq} e_2$, then generate "$e_1, e_2$", "$e_1$ and e_2";

Table 4: The set of concept description and sentence structure rules.

$$\forall i \geq 1, l_{0,i} \leq u_{i;}, \sum_{i=1}^{M} l_{0,i} = 1 \tag{7}$$

$$\sum_{i=1}^{M} f_{0,i} - \sum_{i=1}^{M} u_i = 0 \tag{8}$$

$$\forall k \geq 1, \sum_{i} f_{i,k} - \sum_{p} f_{k,p} - u_k = 0 \tag{9}$$

$$\forall i \geq 0, k \geq 1, M \cdot l_{i,k} - f_{i,k} \geq 0 \tag{10}$$

Length Constraint. To control the summary compression rate, the total number of selected events is limited less than L:

$$\sum_{i=1}^{M} u_i \leq L \tag{11}$$

where parameter L is set to control the graph size after reduction.

3.2 Feature Weights Estimation

We learn feature weights θ and ψ by training on a set of source ESLN paired with gold summary network. The source ESLN is constructed from source texts whereas the gold summary network is constructed from reference summaries and then mapped to the source ESLN by texts similarity method (Pilehvar et al., 2013). We formulate our estimation problem as follows:

$$-score(G^*) + max_G(score(G) + cost(G; G^*)) \tag{12}$$

where G^* denotes the gold summary network. $score()$ is defined in Equation 1. $cost(G; G^*)$ penalizes each event or concept in G but not in G^*, which can be easily incorporated into the linear objective in Equation 2. We optimize our objective using AdaGrad (Duchi et al., 2011) with l_2 regularization ($\lambda = 0.01$), with an initial step size 0.1. The ILP model is solved using Gurobi 6.5.2.

3.3 Summary Generation

Since each event node is structured as *e=Action (Actor, Receiver, TimeArg, LocArg)*, we can generate complete sentence efficiently for it using SimpleNLG (Gatt and Reiter, 2009). However, through experiments we find that low linguistic quality is the biggest problem with the generated sentences, which include syntax error, monotone sentence structure and repetition of the same noun phrases. To improve the linguistic quality of summary, we first over-generate large numbers of summary sentences and then use a greedy algorithm to select sentences with the best linguistic quality and no information overlapping.

Sentence over-generation. To generate a complete and informative sentence, both the description of concepts and the organization of sentence structures need to be settled in following ways:

- Each concept with several attributes can be described in different ways using concept description rules in Table 4.
- For each event node, we use SimpleNLG (Gatt and Reiter, 2009) tool to generate several different sentences, among which the description of concepts or the orders of arguments are different.

	Concepts	Events	Event Relations	After Expanding
avg#/topic	5206	1538	1089	10383

Table 5: The average number of concepts, events, event relations and relations after expanding of each topic in annotated DUC2007 (including 45 topics, each topic has 25 documents).

	precision	recall	F1-score
Concept Extraction	N/A	0.7928	N/A
Event Action identification	0.8532	0.8468	0.8499
Event Mention extraction	0.7272	0.7067	0.7168
Event Relations prediction	0.5894	0.6222	0.6054

Table 6: The performance of concepts extraction, events identification and event relations prediction.

- If two events share semantic links with each other, we merge them to generate one unified sentence by using corresponding sentence structuring rules in Table 4. Note that, when two events share the same actor concept, only one is reserved.
- For any two events that share the same actor, we merge them to generate one sentence using conjunction word "and" to connect event actions and arguments. Only one actor is kept as the subject.

Greedy selection. After the above step, we get large numbers of candidate summary sentences. Some of them would have information overlapping with each other if generated from the same event node. To improve the linguistic quality of summary, we iteratively select a sentence with the highest linguistic quality and delete sentences that have information overlapping with it from the candidate sentences set. The linguistic quality of sentence $s = \{w_1, w_2, \ldots, w_L\}$ is defined similarly as (Banerjee et al., 2015):

$$LQ(s) = 1 \Big/ \left(1 - \left(log_2 \prod_{t=1}^{L} P(w_t|w_{t-1}w_{t-2})\right) \Big/ L\right) \quad (13)$$

where L is the total number of words in sentence s; w_0 and w_{-1} both represent the beginning of sentence s. The 3-gram model $P(w_t|w_{t-1}, w_{t-2})$ is trained on the English Gigaword corpus (`http://www.keithv.com/software/giga/`).

The coherence constraints guarantee the selected summary network to be connected and have a flow from the ROOT node to selected nodes. The selected sentences are ordered based on the direction of flows to obtain a coherent summary.

4 Evaluation Results

4.1 Dataset and Experimental Settings

To evaluate the performance of our system, we use two datasets that have been widely used in multi-document summarization shared tasks: DUC 2006 and DUC 2007. Each task has a gold standard dataset consisting of document clusters and reference summaries. DUC 2007 was manually annotated by using annotation tool brat (`http://brat.nlplab.org`) to extract gold events and gold relations between events, which are used for training the event identification model and event relations prediction model. Table 5 shows the details of the annotated dataset.

The annotated dataset was split into training set (25 topics), development set (5 topics) and test set (15 topics). After training and tuning, the performance of our system is evaluated on the test set as shown in Table 6. An event mention is correctly extracted only if both the event action and event arguments are correct. Table 6 only shows the recall of concept extraction, because we extract all kinds of concepts, whereas only event arguments are annotated in the annotated dataset. The feature weights θ and ψ of event nodes and concepts are also estimated on the training set.

To evaluate the performance of our summarization model, we use both ROUGE (Lin and Hovy, 2003) and Pyramid (Nenkova and Passonneau, 2004) evaluation metrics.

4.2 Results with ROUGE Evaluation

ROUGE-1.5.5 toolkit was used to evaluate the quality of summary on DUC 2006 and DUC 2007 (test set) dataset. We differentiate the different components of our system by including and not including the coherence constraints in ILP-based network reduction algorithm and using the manually annotated gold ESLN in our system. Our systems are compared with several baselines: Centroid (Radev et al., 2000) and

	DUC2007(test set)				DUC2006			
	ROUGE-1	**ROUGE-2**	**ROUGE-SU4**	**Pyramid**	**ROUGE-1**	**ROUGE-2**	**ROUGE-SU4**	**Pyramid**
Baselines								
Centroid	0.36455	0.07032	0.12401	N/A	0.35211	0.06097	0.11570	N/A
LexRank	0.37501	0.07995	0.13528	N/A	0.36275	0.06830	0.12569	N/A
DUC NIST Baseline	0.33434	0.06479	0.11360	N/A	0.32082	0.05267	0.10408	N/A
AverageDUC	0.39684	0.09495	0.14671	N/A	0.37789	0.07483	0.12943	N/A
State-of-the-arts								
MultiMR	0.41967	0.10302	0.15385	N/A	0.39706	0.08508	0.13797	N/A
RA-MDS	0.403	0.092	0.146	N/A	0.391	0.081	0.136	N/A
ILPSumm (Abstractive)	0.41052	0.10060	0.15185	0.844	0.38564	0.07993	0.13279	0.811
PSM (Abstractive)	0.41917	0.10336	0.15608	0.851	0.39287	0.08173	0.13671	0.817
Our Systems								
ESLN with Coherence	**0.42423**	**0.10897**	**0.16137**	**0.865**	0.39487	**0.08756**	**0.14083**	**0.825**
ESLN w/o Coherence	0.41553	0.10291	0.15258	N/A	0.38586	0.08023	0.13618	N/A
Gold-ESLN with Coherence	0.44532	0.12229	0.17267	N/A	0.41162	0.09642	0.15348	N/A

Table 7: Comparison of ROUGE scores (F-score) and Pyramid scores on DUC 2006 and 2007(test set).

LexRank (Erkan and Radev, 2004). The performance of NIST baseline and the average ROUGE scores of all the participating systems (i.e. *AveDUC*) both for DUC 2006 and DUC 2007 main tasks are also listed. According to the results in Table 7, our systems significantly outperform (paired t-test with $p<0.05$) all the baselines, which demonstrates that extracting event information from texts and summarizing based on structured information is much more effective than summarizing on sentence level.

In addition, we also compare our system (ESLN with coherence) with several state-of-the-art summarization methods: graph-based extractive method MultiMR (Wan and Xiao, 2009), sparse-coding-based compressive method RA-MDS (Li et al., 2015), and two most recently developed abstractive methods ILPSumm (Banerjee et al., 2015) and PSM (Bing et al., 2015). The results show that our system significantly (paired t-test with $p<0.05$) outperforms all the other four systems.

The results also show that our system with coherence constraints achieves better performance than the counterpart without coherence constraints. So the coherence constraints are very helpful to select more salient and coherent information. Just as expected, the system using gold ESLN achieves the best performance. Incorrect dependency parsing and co-reference resolution will reduce the accuracy of extracting event information. On the other hand, it also verifies that the method that summarize texts based on accurate event information is effective.

4.3 Results with Pyramid Evaluation

Since ROUGE metric evaluates summaries by strict string matching, we also use the pyramid evaluation metric which can measure the summary quality beyond simply string matching. It involves semantic matching of summary content units (SCUs) so as to recognize alternate realizations of the same meaning, which provides a better metric for abstractive summary evaluation. We employ the automated version of pyramid scoring (set threshold value to 0.6) in (Passonneau et al., 2013). Table 7 shows the evaluation results of our system and two abstractive baselines on both DUC 2006 and DUC 2007(test set). The results show that our system significantly ($p<0.05$) outperform the two baselines on both datasets, which demonstrates that our system can generate more informative summary.

4.4 Discussion

Table 8 shows a comparison of summaries generated by our system and human on DUC 2007 dataset (D0701A). The results show that our summary behaves similarly to human summary in following aspects: (1) Aggregating information from different places. For example, the description of *"Morris Dees"* includes information from several different documents, which are extracted as attributes of concept *"Morris Dees"* in our system; (2) Organizing sentences coherently. The coherence constraints in ILP-based network reduction component ensure the selected event information to be coherent. (3) Clearly pronoun reference. The adjacent sentences with the same subject in the summary are post-edited by replacing subjects of successor sentences with appropriate pronouns. Even though we incorporate the sentences over-generation and greedy-selection components in our system, some sentences in the generated summaries also have few syntax errors. Most cases are because of non-accurate event extraction caused by incorrect dependency parsing or coreference resolution.

Summary by Our System: Morris Dees who was a crusader against intolerance, Keenans' attorney, the chief trial counsel, executive director and co-founder of the Southern Poverty Law Center used lawsuit to fight hate groups. He kept track of hate crime. He put East Peoria leader and won significant civil judgment against White Aryan Resistance and Ku Klux Klan in touch in Chicago with David Ostendorf. He formed a broad-based coalition and won a series of civil rights suit against other racist group in a campaign on race issue. He got an unwarranted slap in the Media Watch column and introduced photograph in the same issue. Southern Poverty Law Center, montgomery-based used civil suit and previously recorded a 20-percent increase in hate group. It battled racial bias and used civil law. It tracked hate group and won major legal fight against other white supremacist group and Ku Klux Klan. The 1973 federal lawsuit had the practical effect. The practical effect provided equal service to black...	**Human-written Summary:** The Southern Poverty Law Center is a nonprofit research group based in Montgomery, Alabama that battles racial bias. It tracks US hate crimes and the spread of racist organizations. It covers right-wing extremists in its magazine Intelligence Report. Through its Teaching Tolerance program, it provides materials to teachers to promote interracial and intercultural understanding. It freely distributes booklets on combating hate to schools, mayors, police chiefs, and other interested groups and citizens. It advises city leaders faced with hate crimes. Morris Dees co-founded the SPLC in 1971 and is its chief trial counsel and executive director, following Julian Bond. Dees and the SPLC seek to destroy hate groups through multi-million dollar civil suits that go after assets of groups and their leaders. In six lawsuits based on hate crimes or civil rights abuses, they have never lost. They successfully sued the Ku Klux Klan and the related Invisible Empire Klan, United Klan of America and ...

Table 8: Example summary of D0701A in DUC2007 dataset by our system and the gold human summary (Only several leading sentences are displayed).

5 Related Work

Abstractive Multi-document summarization. Previous researches have shown that human write summaries through sentence aggregation and fusion (Cheung and Penn, 2013). Abstraction-based approaches that gather information across sentences boundaries have become more and more popular in recent years. Different abstractive summarization methods can be summarized into four technique routes: (1) sentence fusion based methods (Barzilay and McKeown, 2005; Filippova and Strube, 2008; Banerjee et al., 2015) first cluster sentences into several themes and then generate a new sentence for each cluster by fusing the common information of all sentences in the cluster; (2) information extraction based methods (Genest and Lapalme, 2011; Li, 2015) extract information units, such as Information Items or Basic Semantic Unit, as components for generating sentences; (3) summary revision based methods (Nenkova, 2008; Siddharthan et al., 2011) try to improve quality of summary by noun phrases rewriting and co-reference resolution; (4) pattern-based sentence generation methods (Wang and Cardie, 2013; Pighin et al., 2014; Bing et al., 2015) generate new sentences based on a set of sentence generation patterns learned from corpus or designed templates.

Recently, some works studied the use of deep learning techniques for abstractive summarization tasks, which use sequence-to-sequence generation techniques on single document or sentence summarization (Rush et al., 2015; Chopra et al., 2016). A multi-dimensional summarization methodology was proposed to transform the paradigm of traditional summarization research through multi-disciplinary fundamental exploration on semantics, dimension, knowledge, computing and cyber-physical society (Zhuge, 2016).

Event extraction. Event extraction is a traditional task in Information Extraction, which aims to recognize event mentions and arguments of predefined types (such as the ACE tasks). The works on event extraction either divide the task into separate subtasks, such as event-trigger extraction and argument extraction (Liao and Grishman, 2010; Hong et al., 2011) or model it jointly (Li et al., 2013; Li and Ji, 2014). These works mainly focus on predefined event and argument types. However, we focus on open-domain and more fine-grained event information extraction for multi-document summarization.

Abstract representations. With the development of Abstract Meaning Representation (AMR) (Banarescu et al., 2012), representing semantic information with graphs has been studied in such tasks as summarization (Liu et al., 2015) and event detection (Kai and Grishman, 2015). Although several techniques on parsing sentences to AMR (Flanigan et al., 2014; Wang et al., 2015) have been developed, the performance of AMR parsing is very limited at the present.

6 Conclusions

The approach proposed in this paper generates summary based on event information extraction and abstract representation, which achieves good performance on both DUC 2006 and DUC 2007 datasets. It generates new sentences based on structured event information and organizes sentences coherently based on semantic links. The experiment results show that the summaries generated by our system are relatively informative, coherent and compact, which demonstrates that the semantic link network based abstract representation of source texts is effective in making abstractive summarization.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2012. Abstract meaning representation (amr) 1.0 specification. In *EMNLP*, pages 1533–1544.

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *IJCAI*, pages 1208–1214.

Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. *arXiv preprint arXiv:1506.01597*.

Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine De Marneffe, Daniel Ramage, Eric Yeh, and Christopher D Manning. 2007. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 165–170. Association for Computational Linguistics.

Jackie Chi Kit Cheung and Gerald Penn. 2013. Towards robust abstractive multi-document summarization: A caseframe analysis of centrality and domain. In *ACL (1)*, pages 1233–1242.

Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. 2016. Abstractive sentence summarization with attentive recurrent neural networks. *NAACL*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8. Association for Computational Linguistics.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, pages 457–479.

Katja Filippova and Michael Strube. 2008. Sentence fusion via dependency graph compression. In *EMNLP*, pages 177–185. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Jaime G Carbonell, Chris Dyer, and Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation.

Albert Gatt and Ehud Reiter. 2009. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 90–93. Association for Computational Linguistics.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 64–73. Association for Computational Linguistics.

Goran Glavaš and Jan Šnajder. 2014. Event graphs for information retrieval and multi-document summarization. *Expert systems with applications*, 41(15):6904–6916.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *ACL*, pages 1127–1136. Association for Computational Linguistics.

Shafiq R Joty, Giuseppe Carenini, Raymond T Ng, and Yashar Mehdad. 2013. Combining intra- and multi-sentential rhetorical parsing for document-level discourse analysis. In *ACL*, pages 486–496.

Xiang Li Thien Huu Nguyen Kai and Cao Ralph Grishman. 2015. Improving event detection with abstract meaning representation. *ACL-IJCNLP 2015*, page 11.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *ACL (1)*, pages 402–412.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL (1)*, pages 73–82.

Piji Li, Lidong Bing, Wai Lam, Hang Li, and Yi Liao. 2015. Reader-aware multi-document summarization via sparse coding. *arXiv preprint arXiv:1504.07324*.

Wei Li. 2015. Abstractive multi-document summarization with semantic information extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1908–1913.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *ACL*, pages 789–797. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *HLT-NAACL*, pages 71–78. Association for Computational Linguistics.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2015. Toward abstractive summarization using semantic representations.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method.

Ani Nenkova. 2008. Entity-driven rewrite for multi-document summarization.

Rebecca J Passonneau, Emily Chen, Weiwei Guo, and Dolores Perin. 2013. Automated pyramid scoring of summaries using distributional semantics. In *ACL (2)*, pages 143–147.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at HLT-NAACL 2004*, pages 38–41. Association for Computational Linguistics.

Daniele Pighin, Marco Cornolti, Enrique Alfonseca, and Katja Filippova. 2014. Modelling events through memory-based, open-ie patterns for abstractive summarization. In *ACL (1)*, pages 892–901.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *ACL (1)*, pages 1341–1351.

Dragomir R Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic summarization*, pages 21–30. Association for Computational Linguistics.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D Manning. 2015. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proceedings of the Fourth Workshop on Vision and Language*, pages 70–80.

Chao Shen, Fei Liu, Fuliang Weng, and Tao Li. 2013. A participant-based approach for event summarization using twitter streams. In *HLT-NAACL*, pages 1152–1162.

Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2011. Information status distinctions and referring expressions: An empirical study of references to people in news summaries. *Computational Linguistics*, 37(4):811–842.

Kapil Thadani and Kathleen McKeown. 2013. Sentence compression with joint structural inference. In *CoNLL*, pages 65–74.

Xiaojun Wan and Jianguo Xiao. 2009. Graph-based multi-modality learning for topic-focused multi-document summarization. In *IJCAI*, pages 1586–1591.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *ACL (1)*, pages 1395–1405.

Chuan Wang, Nianwen Xue, Sameer Pradhan, and Sameer Pradhan. 2015. A transition-based algorithm for amr parsing. In *HLT-NAACL*, pages 366–375.

Hai Zhuge. 2012. *The Knowledge Grid: Toward Cyber-Physical Society*. World Scientific.

Hai Zhuge. 2016. Multi-dimensional summarization in cyber-physical society. *Morgan Kaufmann*.

A General Optimization Framework for Multi-Document Summarization Using Genetic Algorithms and Swarm Intelligence

Maxime Peyrard and **Judith Eckle-Kohler**
Research Training Group AIPHES and UKP Lab
Computer Science Department, Technische Universität Darmstadt
www.aiphes.tu-darmstadt.de, www.ukp.tu-darmstadt.de

Abstract

Extracting summaries via integer linear programming and submodularity are popular and successful techniques in extractive multi-document summarization. However, many interesting optimization objectives are neither submodular nor factorizable into an integer linear program. We address this issue and present a general optimization framework where any function of input documents and a system summary can be plugged in. Our framework includes two kinds of summarizers – one based on genetic algorithms, the other using a swarm intelligence approach. In our experimental evaluation, we investigate the optimization of two information-theoretic summary evaluation metrics and find that our framework yields competitive results compared to several strong summarization baselines. Our comparative analysis of the genetic and swarm summarizers reveals interesting complementary properties.

1 Introduction

Extractive multi-document summarization (MDS) is often cast as a discrete optimization problem where the document collection is considered as a set of sentences and the task is to select an optimal subset of the sentences under a length constraint. Many successful approaches solve this problem using integer linear programming (ILP) or submodular function maximization. Both ILP and submodular function maximization require that the objective function to maximize has certain properties: it needs to be submodular or linearly factorizable (for ILP). The commonly used optimization objective that combines maximizing the coverage of relevant units while minimizing their redundancy has been shown to be submodular by Lin and Bilmes (2011). Lin and Bilmes (2011) also proved that the summary evaluation metric ROUGE itself is submodular, which has been leveraged by Sipos et al. (2012) to optimize ROUGE directly via submodular function maximization. Peyrard and Eckle-Kohler (2016) optimize an approximation of ROUGE via ILP. Using ROUGE as an optimization objective in MDS is reasonable, because ROUGE-1 and ROUGE-2 have been shown to correlate well with the results of human evaluation (e.g., Owczarzak et al. (2012)), and are therefore good proxies for system summary quality.

The results we present in this paper start from the observation that ROUGE is just one possible proxy for summary quality – there are other automatic metrics to evaluate system summaries, which also correlate well with human judgments (Louis and Nenkova, 2013). For example, the Jensen Shannon divergence (JS divergence), an information-theoretic measure not relying on human-written summaries, compares system summaries with source documents regarding their underlying probability distribution of n-grams (Lin et al., 2006). However, unlike ROUGE, JS divergence is neither submodular nor factorizable (see, e.g., Louis and Nenkova (2013)), and consequently can not be optimized via ILP or submodular function optimization. Similarly, we can imagine other interesting, arbitrarily complex summary evaluation metrics, which are neither submodular nor factorizable, but which we might want to optimize. For example, we might be interested in combining several metrics such that the combination correlates well with human evaluation scores.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 247–257, Osaka, Japan, December 11-17 2016.

In order to solve a discrete optimization problem (which is NP-hard) in the general case, where the optimization objective does not have specific properties, we have to rely on search algorithms. A particular promising class of search algorithms to tackle this problem are metaheuristics (Bianchi et al., 2009). In this paper, we consider two kinds of metaheuristics – genetic algorithms and a swarm intelligence approach called Artificial Bee Colony – and propose a general optimization framework for MDS where the function to be optimized is a parameter and can be exchanged.

Our contributions can be summarized as follows: we present an optimization framework for extractive MDS that is able to optimize an arbitrarily complex objective function of input documents and a summary, without making any assumptions on its properties. For our framework, we developed two summarizers, one based on genetic algorithms, the other based on swarm intelligence. In our experimental evaluation, we investigate the optimization of information-theoretic summary evaluation metrics and find that our framework yields competitive results compared to several strong baselines. We also comparatively analyze the behavior of our two summarizers, and observe that there are several system summaries with very high ROUGE score, which have almost no sentences in common.

2 Background

Some branches of research on optimization study the landscape of functions to maximize (also called fitness landscape). When the landscape has particular properties, methods can be derived to find optima efficiently. For example, ILP leverages the linear properties (Schrijver, 1986) of an objective function. Recently, submodular functions have been extensively studied and simple algorithms have been proved to yield nice solutions (Krause and Golovin, 2014; Schrijver, 2003).

However, we might not want to restrict ourselves to particular kinds of functions, because often, the landscape of the objective function does not have easy to exploit properties (Bianchi et al., 2009). To maximize such functions, the optimization field developed techniques that do not make any assumption about their landscape (Blum and Roli, 2003; Wright, 1932; Fogel et al., 1966).

To solve such complex optimization problems, search-based techniques are used (i.e., they do not find the exact solution like ILP), which employ various strategies to efficiently explore the search space. The majority of the successful search-based techniques are biologically inspired and take efficient and adaptive optimization strategies used in nature as a role model.

These algorithms can roughly be divided into two main categories according to their biological inspiration: **Genetic algorithms** which simulate the evolution process, and **swarm intelligence** algorithms which simulate the behavior of swarms and colonies of animals such as bees, ants or fireflies.

Genetic Algorithms In artificial intelligence, genetic algorithms (GA) use mechanisms inspired by biological evolution, such as reproduction, mutation and selection (Wright, 1932; Goldberg, 1989).

The candidate solutions of the optimization problem are represented as individuals in a population. The fitness function is the function to optimize and determines the quality of an individual. To apply a GA to an optimization problem, the parameters that define a candidate solution are considered as the genotype of the individual. The genotype is the footprint that uniquely identifies every possible solution.

Evolution of the population takes place after multiple iterations where biological operators are applied: Reproduction and mutation search the space of solutions by creating new candidate solutions, while the selection operator ensures that better candidate solutions survive to the next generation more often than worse ones.

Swarm Intelligence Swarm Intelligence (SI) refers to the collective behavior of decentralized, self-organized artificial systems. The agents in such systems follow very simple rules, and although there is no centralized control structure, local and random interactions between agents lead to the emergence of intelligent global behavior, unknown to the individual agents themselves (Beni and Wang, 1993; Bonabeau et al., 1999; Parsopoulos and Vrahatis, 2002).

While the population of the GAs consists of candidate solutions, the swarm population is made up of agents which search the solution space and interact locally with the environment. The candidate solutions are points in the space investigated by the agents. The agent interactions with the candidate space usually

consist of the evaluation of a given area. To apply a SI algorithm to an optimization problem, one must define a space where candidate solutions live. A point in the space of candidate solutions is analogous to the vectors of genes (parameters, i.e., genotype) of the GA. Simple communication channels allow agents to exchange information about promising areas. Examples of such natural systems are ant colonies, fireflies glowing, fish schooling or bird flocking.

One successful model we decided to follow in this work is the model of honey bees searching for nectar in a field, also known as Artificial Bee Colony (ABC) (Karaboga and Basturk, 2007; Karaboga et al., 2014). In ABC, there are three groups of bees: employed bees, onlookers and scouts.

There is only one employed bee per food source (the number of employed bees in the colony is equal to the number of food sources investigated in parallel). Employed bees collect food from their food source and dance in this area after evaluating the quantity of food in the direct neighborhood. The dance indicates the amount of food in the area identified by the employed bee. When the food source is abandoned, the employed bee becomes a scout and starts to search for a new food source elsewhere. Onlookers watch the dances of employed bees and choose food sources which are especially promising. The overall behavior allows the swarm to find areas which contain a lot of nectar (or food). The employed bees and onlookers use the nectar function to evaluate the quantity of nectar at a given location (i.e., candidate solution) which is the equivalent to the fitness function in the GA.

We will study the differences and similarities of these algorithms when we adapt them to extractive summarization in the following section.

3 Optimization Framework

We propose a general framework that extracts a summary with high score for any metric that is a function of the source documents and the summary only. The metric to maximize is a parameter of our framework that can easily be changed. First, we present the metrics we considered in our framework, and then the optimization techniques we used.

3.1 Metrics

In our framework, we consider classical similarity metrics for comparing the source documents and the summary content.

As it was shown in previous work, good summaries are often characterized by a low divergence between the probability distributions of n-grams in the source documents and the summary (Haghighi and Vanderwende, 2009; Louis and Nenkova, 2013). One common metric for determining this divergence is the Kullback Leibler (KL) divergence. The KL divergence between two probability distributions P and Q is given by:

$$KL(P\|Q) = \sum_g p_P(w) log_2 \frac{p_P(w)}{p_Q(w)} \tag{1}$$

In the case of MDS, the two probability distributions P and Q are computed from the source documents and the summary.

Jensen Shannon (JS) divergence is a symmetric version of the KL divergence, incorporating the idea that the distance between two distributions cannot be very different from the average of distances from their mean distribution. It is given by:

$$JS(P\|Q) = \frac{1}{2}(KL(P\|A) + KL(Q\|A)) \tag{2}$$

where $A = \frac{P+Q}{2}$ is the mean distribution of P and Q.

The metrics are defined on the basis of n-grams, and are not restricted to words only. In practice, we use both the unigram and bigram version of these metrics. It is important to point out that these metrics only serve as example metrics – any other metric function of the input documents and the summary can be plugged into the framework.

3.2 Biologically-Inspired Optimization Techniques

In this section, we will describe how to adapt GA and SI to the problem of extractive summarization.

Genetic Summarizer In order to produce a Genetic Summarizer, we use a simple analogy to the problem of extractive summarization.

- **Population** The individuals of the population are the candidate solutions which are valid extractive summaries. Valid means that the summary meets the length constraint. The size of the population is a hyper-parameter of the algorithm.

- **Genome and Genotype** The genome is the set of sentences in the source documents of the topic. It corresponds to the building blocks of each possible individual. Then, the genotype of a summary is simply a binary vector indicating which sentences it contains.

- **Fitness Function** The fitness function which evaluates the individuals (i.e., summaries) is the function we wish to maximize (or minimize). For example, it might be one of the metrics we described above. The population is scored and sorted according to the fitness function, a threshold indicates which summaries will survive to the next generation. The survival rate is another hyper-parameter.

- **Mutation** The mutation of a summary is done by randomly removing one of its sentences and adding a new one that does not violate the length constraint. Mutation affects individuals of a population randomly, and the mutation rate is a hyper-parameter.

- **Reproduction** The reproduction is done by randomly selecting parents among the survivors of the previous generation. Then, the union set of the sentences of the parents is considered. The child is a random valid summary extracted from these sentences. Similar as for the mutation, we define a reproduction rate which controls the number of children in each generation.

- **Initial Population** The initial population is simply created by randomly building valid summaries. We observe a convergence speed-up if we help the algorithm by including good summaries in the initial population (e.g., summaries produced by baseline algorithms).

We note that in nature, the fertilized egg cell undergoes a process known as embryogenesis before becoming a mature embryo. This is believed to make the genetic search more robust by reducing the probability of fatal mutation. At each step, we only consider valid summaries, which is the direct analogy to embryogenesis.

Swarm Summarizer

- **Food Location** The locations in the field are the candidate solutions which are the valid extractive summaries. The number of food locations is a hyper-parameter equivalent to the size of the population in the Genetic Summarizer.

- **Location Coordinates** The summaries are points in the space searched by the bees. The coordinates are given by the binary vector indicating which sentences the summary contains. The coordinates of a food location is the equivalent to the genotype of an individual.

- **Nectar Function** The nectar function which evaluates the food locations (i.e., summaries) is the function to maximize. It corresponds to the fitness function.

- **Employed Bees Local Search** At each iteration, employed bees evaluate the direct neighborhood of their assigned food location. To move to a neighbor, a sentence is randomly removed from the current summary (i.e., food location) and a new sentence that does not violate the length constraint is added. This is the analogy to a mutation.

- **Employed Bees Dance and Onlooker Bees** When employed bees have evaluated the summaries, all the summaries are scored and sorted. Onlooker bees observe this distribution of scores and randomly choose one location to join and help with the neighbor search. This random choice is based on the following probability:

$$P_i = \frac{score_i}{\sum_k score_k} \tag{3}$$

  As a result, onlookers choose high scoring locations (i.e., summaries) more often.

- **Scouting Bees** An employed bee stays in place if the neighbor it evaluates is not better than its current location. After several iterations at the same place, it abandons it and becomes a scouting bee. To move to another place, the scouting bee selects a random valid summary. This is equivalent to generating an individual from the initial population in the Genetic Summarizer. The number of iterations before becoming a scouting bee is the second hyper-parameter of the Swarm Summarizer.

Genetic vs. Swarm In the Genetic Summarizer, the reproduction produces significant changes in the summaries studied, because, on average, half of the genotype of the child is different from its parents. But at the same time, it stays in a reasonable distance range from its parents because it keeps half of the genotype from each parent (on average). In this sense, we say that the Genetic Summarizer has efficient mid-range search capabilities. The local search is much reduced because it is done via mutations happening randomly and rarely in the population. The long-range search is done via insertion of random individuals into the population whenever the population becomes too small. A new completely random individual is quite likely to have a low fitness score and to die in the next generation with few opportunities to reproduce or mutate.

The Swarm Summarizer has complementary strengths and weaknesses. The employed bees perform intensive local search around a specific location and the onlooker bees help them around the locations of interest. For long-range search, the scout bees regularly look for new locations and investigate each new area for at least t rounds (where t is the hyper-parameter controlling the number of retry before becoming a scout bee). However, in the Swarm Summarizer, the mid-range search is limited compared to the reproduction mechanism, because it is achieved only by either successfully applying several local movements, or by randomly scouting the mid-range areas, both of which are unlikely.

Even if we did not conduct an extensive hyper-parameters optimization, we observe that the Swarm Summarizer has much less hyper-parameters, which would make it simpler to optimize.

4 Experiments

4.1 Summarizer Performance

Baselines: We compare our algorithms to several classical baselines:

TF*IDF weighting This simple heuristic was introduced by Luhn (1958). Each sentence receives a score from the TF*IDF of its terms and the best sentences are greedily extracted until the length constraint is met.

LexRank (Erkan and Radev, 2004) is a popular graph-based approach. A similarity graph $G(V, E)$ is constructed where V is the set of sentences and an edge e_{ij} is drawn between sentences v_i and v_j if and only if the cosine similarity between them is above a given threshold. Sentences are scored according to their PageRank score in G. We use the implementation available in the sumy package.[1]

ICSI (Gillick and Favre, 2009) is a recent system that has been identified as one of the state-of-the-art systems by Hong et al. (2014). It is a global linear optimization framework that extracts a summary by solving a maximum coverage problem considering the most frequent bigrams in the source documents. Boudin et al. (2015) released a Python implementation (ICSI sume) that we use in our experiments.

KL-div Greedy Haghighi and Vanderwende (2009) presented a greedy algorithm to minimize the KL-divergence of extracted summaries. The approach iteratively selects a sentence s_i to be included in the summary S, which is done by picking the sentence that minimizes the KL divergence between the

[1] https://github.com/miso-belica/sumy

	DUC-02		DUC-03		DUC-02		DUC-03	
	JS-1	JS-2	JS-1	JS-2	R-1	R-2	R-1	R-2
TF*IDF	0.4075	0.5522	0.4714	0.6144	0.4072	0.1201	0.3222	0.0660
LexRank	0.3618	0.5454	0.4208	0.6010	0.4311	0.1388	0.3574	0.0793
KL-Greedy	0.3805	0.5531	0.4413	0.6036	0.3945	0.1125	0.3231	0.0715
JS-Greedy	0.3683	0.5475	0.4386	0.5961	0.4299	0.1455	0.3312	0.0640
ICSI	0.3537	0.5107	0.4045	0.5657	0.4434	0.1556	0.3763	0.0947
JS-Gen	0.3196	0.5125	0.3776	0.5727	0.4397	0.1416	0.3728	0.0855
JS-Gen-2	**0.3026**	**0.4937**	0.4059	0.5476	**0.4545**	**0.1629**	**0.3787**	**0.0959**
JS-Swarm	0.3244	0.5326	**0.3751**	0.5651	0.4433	0.1352	0.3761	0.0879
JS-Swarm-2	0.3089	0.5004	0.4027	**0.5402**	0.4321	0.1507	0.3615	0.0893
KL-Gen	0.3325	0.5314	0.3810	0.5724	0.4470	0.1505	0.3658	0.0873
KL-Gen-2	0.3811	0.4952	0.4171	0.5556	0.3903	0.1369	0.3593	0.0893
KL-Swarm	0.3455	0.5311	0.3859	0.5604	0.4451	0.1499	0.3703	0.0876
KL-Swarm-2	0.3572	0.5140	0.3934	0.5543	0.4449	0.1574	0.3698	0.0897

Table 1: Performance of biologically inspired algorithms to minimize JS and KL divergence.

word distributions of the original input documents and $S \cup s_i$. We also implement **JS-div Greedy** which is the same greedy algorithm, but minimizes JS divergence instead.

Experimental Setup: In our experiments, we use two datasets from the Document Understanding Conference (DUC) (Over et al., 2007), namely the datasets from 2002 and 2003 (DUC-02 and DUC-03). We compare algorithms with the JS divergence metric for both the unigrams (JS-1) and the bigrams (JS-2) variants. The results are shown in Table 1. For completeness, we also report the ROUGE scores identified by Owczarzak et al. (2012) as strongly correlating with human evaluation methods: ROUGE-1 (R-1) and ROUGE-2 (R-2) recall with stemming and stopwords not removed. Our algorithms are denoted by F-A-(2), where F is the function minimized, A is the algorithm used (Genetic or Swarm), and 2 is present if the bigram version of the function is minimized.

We use a standard implementation of JS divergence with stemming to get the JS divergence scores. ROUGE scores are obtained with the ROUGE-1.5.5 toolkit.[2]

Results Analysis: We observe that Genetic and Swarm summarizers have similar performances, there is no statistically significant difference when evaluated with JS and ROUGE evaluation metrics.

Both summarizers outperform the greedy versions by a large margin in all experiments. This shows that the Genetic and Swarms summarizers are strong search algorithms for extractive summarization.

We compare the behavior of Swarm and Genetic in more detail in section 4.2.

Our summarizers perform on par with the state-of-the-art algorithm ICSI in terms of ROUGE (no statistically significant difference). However, the Genetic and Swarm summarizers perform significantly better for the JS divergence evaluation metric. This suggests that there are several different high scoring summaries in terms of ROUGE, and our algorithms find summaries different from ICSI. We investigate this observation in more detail in section 4.3. ROUGE is known to have issues with distinguishing good summaries, and here we observe that JS is a complementary metric allowing us to distinguish high-scoring summaries.

The biologically inspired algorithms can even outperform ICSI and the other baselines for the ROUGE metric, thus confirming the correlation between JS divergence and ROUGE.

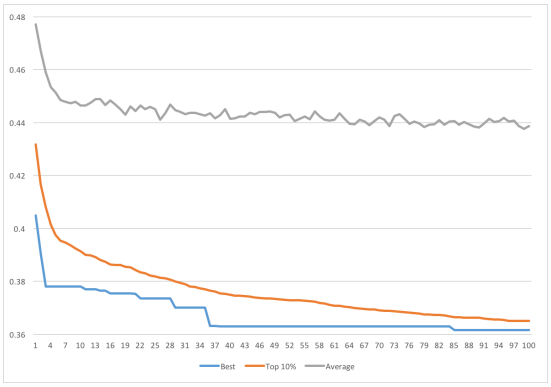
4.2 Convergence behavior

In this section, we study the behavior of our algorithms by investigating the convergence process of both the Genetic Summarizer and the Swarm Summarizer. For this experiment, we focus on the JS divergence minimization with the topic d30003t of DUC-2003.

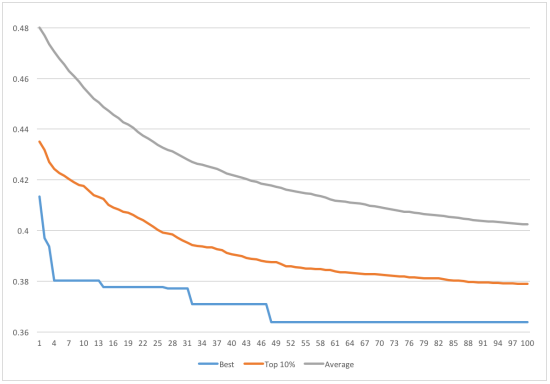
We plot in Figure 1a a graph displaying the evolution of three measures of the Genetic Summarizer across generations: the JS divergence of the best individual found, the average JS divergence of the top

[2]ROUGE-1.5.5 with the parameters: -n 2 -m -a -l 100 -x -c 95 -r 1000 -f A -p 0.5 -t 0. The length parameter becomes -l 200 for DUC-02.

10% of the population, and the average JS divergence of the population. The same analysis is reported for the Swarm Summarizer in Figure 1b.



(a) Genetic Summarizer convergence behavior (d30003t from DUC-2003).



(b) Swarm Summarizer convergence behavior (d30003t from DUC-2003).

Similarities: For both summarizers, the score of the best solution is quickly decreasing until some plateau is reached. Then, only punctual breakthroughs lead to significant improvements of the best solution. The top 10% of the population is regularly improving, following an exponentially decreasing curve, which makes it a good indicator that the algorithm has converged. Indeed, the best individual curve can plateau for several iterations and experience a large jump in one iteration. In Figure 1a, we observe a big breakthrough at generation 35 after a long period without improvement.

Differences: In the Genetic Summarizer, the average of the whole population quickly drops in the first few iterations and then stabilizes. In contrast, in the Swarm Summarizer, the curve of the overall average follows the curve of the top 10%.

The Genetic algorithm includes a lot of randomness at each generation. The majority of the population is randomly generated at each iteration (random restart and reproduction), and therefore the average of the population stays constant (there is a drop in the first iterations because the best members of the population are not random).

In the Swarm Summarizer, the randomness is controlled and the focus is put on local search. Employed bees and onlooker bees work on promising areas, and new random restarts (scouting bees) are introduced only when an area is not promising enough. By driving its workforce towards local search around promising areas, the Swarm Summarizer keeps working around better solutions.

For the Genetic summarizer, we also observe that the best elements of the population are closing the gap with the best individual, and the newly accumulated potential in the population allows for a new breakthrough via reproduction. The top 10% of the population are closer to the best solution than the top 10% of the Swarm summarizer.

With a good initialization (by introducing good summaries in the initial candidate solutions), the Swarm Summarizer converges much faster than the Genetic Summarizer. While the Genetic Summarizer keeps finding breakthroughs after 85 iterations, the Swarm Summarizer has converged already after 50 iterations.

In terms of runtime, both algorithms can find summaries close to the best ones after only several seconds. The Swarm Summarizer converges within about a minute for a given topic, while the Genetic Summarizer converges within 2-3 minutes on average [3].

4.3 Summaries Found

Our experiments in section 4.1 suggested that there are several different high scoring summaries. In this section, we study how different the summaries are from each other, and measure the diversity of the summaries discovered by our techniques.

[3] On a standard MacBook Pro with 16GB of RAM and i5 2.9GHz.

	JS-Gen	JS-Swarm
Jaccard to ICSI	0.0556	0.0610
Jaccard among top 10% (avg.)	0.4350	0.1786

Table 2: Diversity of summaries found by JS-Gen and JS-Swarm.

We know from section 4.1 that summaries produced by our algorithms and ICSI have many words and bigrams in common. However, since extractive MDS is a combinatorial problem, we can compare two summaries by comparing which sentences have been selected.

For this, we use the Jaccard distance between two summaries (each represented as set of sentences). It indicates the percentage of sentences two summaries have in common. We measure the Jaccard distance between the summaries of ICSI and the ones created by JS-Gen and JS-Swarm. To measure the diversity among the summaries found by our summarizers, we also measure the average Jaccard distance between two summaries belonging to the top 10% of solutions. The results are reported in Table 2. Surprisingly, we see that the summaries found by JS-Gen or JS-Swarm and those found by ICSI do use different sentences, as they have less than 10% of sentences in common – even though the JS-Gen and JS-Swarm summaries tend to use the same words and bigrams as ICSI. As they are not significantly different in terms of ROUGE scores, this indicates that there exist several extractive summaries with high ROUGE scores. In contrast, the JS divergence metric is able to distinguish between these summaries with high ROUGE scores.

We also observe that JS-Swarm works on more diverse summaries than JS-Gen, because the average Jaccard distance between summaries in the top 10% is much smaller compared to JS-Gen. JS-Swarm is capable of finding high scoring summaries in different areas and is therefore more likely to identify many of the different good summaries.

5 Related Work and Discussion

Previous applications of metaheuristics in the context of MDS used several different approaches to solve the MDS task. There are clustering-based approaches where metaheuristics are used to obtain a good sentence clustering (Aliguliyev, 2009; Song et al., 2011), and also ranking-based approaches where metaheuristics perform an optimization of an importance metric for sentence scoring (Litvak et al., 2010).

Most related to our framework is previous work where extractive MDS is also cast as a discrete optimization problem that is solved using metaheuristics. Several approaches applied genetic algorithms or variants to perform extractive MDS, but used different optimization objectives than us. Alguliev et al. (2013) employ differential evolution, an algorithm similar to GA where mutation is the main operation, and optimize a combination of content coverage and diversity. However, the runtime complexity of differential evolution is high and also depends on the runtime complexity of the fitness function used, which is high as well because their function includes a similarity comparison between sentences.

Nandhini and Balasundaram (2013) also use GA for extractive MDS, but consider the creation of assistive summaries rather than generic ones, and thus optimize a combination of readability, cohesion and topic-relatedness. He et al. (2006) use GA with an objective that combines the maximization of informativeness, the reduction of redundancy and also includes the length constraint. In contrast, our use of GA performs embryogenesis, i.e., non-valid individuals (summaries that do not have the required length) are not born.

While we considered KL and JS divergence as exemplary optimization objectives in our framework, any metric that is a function of the input documents and the summary can be used instead (e.g., other metrics from previous works), and also any combination of such metrics. Provided that human evaluation scores are available, we could even learn such a combination metric with the training objective that it correlates well with the human scores. This makes our optimization framework highly customizable to more specific summarization tasks, or genres other than newswire, where JS divergence might not be the best optimization objective (such as opinion and biographical texts as studied by Saggion et al. (2010)). To encourage the community to experiment with different metrics and summarization tasks, we provide the implementation of both the genetic and swarm summarizer at `https://github.com/UKPLab/`

coling2016-genetic-swarm-MDS.

6 Conclusion

We presented a general optimization framework for extractive MDS where any function of input documents and summary can be plugged in and used as an optimization objective. The algorithms we consider belong to the class of metaheuristics, and we developed an adaptation of genetic algorithms and of a swarm intelligence approach called Artificial Bee Colony to the task of extractive MDS. The Python implementation of the genetic and swarm summarizers is freely available[4] and can be extended with any other scoring function. As exemplary optimization objectives, we studied the summary evaluation metrics KL divergence and JS divergence. Our evaluation on the DUC-02 and DUC-03 datasets shows a competitive performance of our framework. In our detailed analysis we found interesting complementary properties of the genetic and swarm summarizers and of a strong baseline.

Acknowledgments

This work has been supported by the German Research Foundation (DFG) as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under grant No. GRK 1994/1, and via the German-Israeli Project Cooperation (DIP, grant No. GU 798/17-1).

References

Rasim M. Alguliev, Ramiz M. Aliguliyev, and Nijat R. Isazade. 2013. Multiple Documents Summarization Based on Evolutionary Optimization Algorithm. *Expert Systems with Applications*, 40(5):1675–1689.

Ramiz M. Aliguliyev. 2009. A New Sentence Similarity Measure and Sentence Based Extractive Technique for Automatic Text Summarization. *Expert Systems with Applications*, 36(4):7764–7772.

Gerardo Beni and Jing Wang. 1993. Swarm Intelligence in Cellular Robotic Systems. In *Robots and Biological Systems: Towards a New Bionics?*, pages 703–712, Berlin, Heidelberg. Springer.

Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J. Gutjahr. 2009. A Survey on Metaheuristics for Stochastic Combinatorial Optimization. *Natural Computing*, 8(2):239–287.

Christian Blum and Andrea Roli. 2003. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, 35(3):268–308.

Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. 1999. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Inc., New York, NY, USA.

Florian Boudin, Hugo Mougard, and Benot Favre. 2015. Concept-based Summarization using Integer Linear Programming: From Concept Pruning to Multiple Optimal Solutions. In Llus Mrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1914–1918, Lisbon, Portugal.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality As Salience in Text Summarization. *Journal of Artificial Intelligence Research*, pages 457–479.

Lawrence J. Fogel, Alvin J. Owens, and Michael J. Walsh. 1966. Intelligent decision making through a simulation of evolution. *Behavioral Science*, 11(4):253–272.

Dan Gillick and Benoit Favre. 2009. A Scalable Global Model for Summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, ILP '09, pages 10–18, Boulder, Colorado.

David E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring Content Models for Multi-document Summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, Boulder, Colorado.

[4] https://github.com/UKPLab/coling2016-genetic-swarm-MDS

Yan-xiang He, De-xi Liu, Dong-hong Ji, Hua Yang, and Chong Teng. 2006. MSBGA: A Multi-Document Summarization System Based on Genetic Algorithm. In *2006 International Conference on Machine Learning and Cybernetics*, pages 2659–2664.

Kai Hong, John Conroy, benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A Repository of State of the Art and Competitive Baseline Summaries for Generic News Summarization. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1608–1616, Reykjavik, Iceland.

Dervis Karaboga and Bahriye Basturk. 2007. A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. *Journal of Global Optimization*, 39(3):459–471.

Dervis Karaboga, Beyza Gorkemli, Celal Ozturk, and Nurhan Karaboga. 2014. A Comprehensive Survey: Artificial Bee Colony (ABC) Algorithm and Applications. *Artificial Intelligence Review*, 42(1):21–57.

Andreas Krause and Daniel Golovin. 2014. Submodular function maximization. *Tractability: Practical Approaches to Hard Problems*, pages 71–104.

Hui Lin and Jeff A. Bilmes. 2011. A Class of Submodular Functions for Document Summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 510–520, Portland, Oregon.

Chin-Yew Lin, Guihong Cao, Jianfeng Gao, and Jian-Yun Nie. 2006. An Information-Theoretic Approach to Automatic Evaluation of Summaries. In *Proceedings of the Human Language Technology Conference at NAACL*, pages 463–470, New York, NY, USA.

Marina Litvak, Mark Last, and Menahem Friedman. 2010. A New Approach to Improving Multilingual Summarization Using a Genetic Algorithm. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 927–936, Uppsala, Sweden.

Annie Louis and Ani Nenkova. 2013. Automatically Assessing Machine Summary Content Without a Gold Standard. *Computational Linguistics*, 39(2):267–300.

Hans Peter Luhn. 1958. The Automatic Creation of Literature Abstracts. *IBM Journal of Research Development*, 2:159–165.

K. Nandhini and S. R. Balasundaram. 2013. Use of Genetic Algorithm for Cohesive Summary Extraction to Assist Reading Difficulties. *Applied Computational Intelligence and Soft Computing*, 2013:8–16.

Paul Over, Hoa Dang, and Donna Harman. 2007. DUC in Context. *Information Processing and Management*, 43(6):1506–1520.

Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An Assessment of the Accuracy of Automatic Evaluation in Summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9, Montreal, Canada.

Konstantinos E. Parsopoulos and Michael N. Vrahatis. 2002. Recent approaches to global optimization problems through Particle Swarm Optimization. *Natural Computing*, 1(2):235–306.

Maxime Peyrard and Judith Eckle-Kohler. 2016. Optimizing an Approximation of ROUGE - a Problem-Reduction Approach to Extractive Multi-Document Summarization. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, volume 1: Long Papers, pages 1825–1836, Berlin, Germany.

Horacio Saggion, Juan-Manuel Torres Moreno, Iria da Cunha, Eric San Juan, and Patricia Velazquez-Morales. 2010. Multilingual Summarization Evaluation without Human Models. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2010)*, pages 1059–1067, Beijing, China.

Alexander Schrijver. 1986. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA.

Alexander Schrijver. 2003. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, New York, NY, USA.

Ruben Sipos, Pannaga Shivaswamy, and Thorsten Joachims. 2012. Large-margin Learning of Submodular Summarization Models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 224–233, Avignon, France.

Wei Song, Lim Cheon Choi, Soon Cheol Park, and Xiao Feng Ding. 2011. Fuzzy Evolutionary Optimization Modeling and Its Applications to Unsupervised Categorization and Extractive Summarization. *Expert Systems with Applications*, 38(8):9112–9121.

Sewall Wright. 1932. The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of the Sixth International Congress of Genetics*, 1:356–66.

Exploiting Sentence and Context Representations in Deep Neural Models for Spoken Language Understanding

Lina M. Rojas-Barahona, Milica Gašić, Nikola Mrkšić, Pei-Hao Su
Stefan Ultes, Tsung-Hsien Wen and Steve Young
Department of Engineering, University of Cambridge, Cambridge, UK
lmr46, mg436, nm480, phs26, su259, thw28, sjy@cam.ac.uk

Abstract

This paper presents a deep learning architecture for the semantic decoder component of a Statistical Spoken Dialogue System. In a slot-filling dialogue, the semantic decoder predicts the dialogue act and a set of slot-value pairs from a set of n-best hypotheses returned by the Automatic Speech Recognition. Most current models for spoken language understanding assume (i) word-aligned semantic annotations as in sequence taggers and (ii) delexicalisation, or a mapping of input words to domain-specific concepts using heuristics that try to capture morphological variation but that do not scale to other domains nor to language variation (e.g., morphology, synonyms, paraphrasing). In this work the semantic decoder is trained using unaligned semantic annotations and it uses distributed semantic representation learning to overcome the limitations of explicit delexicalisation. The proposed architecture uses a convolutional neural network for the *sentence representation* and a long-short term memory network for the *context representation*. Results are presented for the publicly available DSTC2 corpus and an In-car corpus which is similar to DSTC2 but has a significantly higher word error rate (WER).

1 Introduction

In most existing work on Spoken Language Understanding (SLU), semantic decoding is usually seen as a sequence tagging problem with models trained and tested on datasets with word-level annotations (Tür et al., 2013; Mesnil et al., 2015; Yao et al., 2013; Sarikaya et al., 2011; Deoras and Sarikaya, 2013; Sarikaya et al., 2014). Spoken language understanding from *unaligned data*, in which utterances are annotated with an abstract semantics, faces the additional challenge of not knowing which specific words are relevant for extracting the semantics. This problem was tackled in (Zhou and He, 2011), by using conditional random fields (CRFs) driven by finely-tuned hand-crafted features. Other discriminative approaches that deal with unaligned data use some form of *delexicalisation* or mapping of the input to known ontological concepts (Henderson et al., 2012; Henderson et al., 2014a). The main disadvantage of delexicalisation is the difficulty in scaling it, not only to larger and more complex dialogue domains but also to handle the many forms of language variation.

We propose in this paper a semantic decoder that learns from unaligned data (Figure 1) and that exploits rich semantic distributed word representations instead of delexicalisation. The semantic decoder predicts the dialogue act and the set of slot-value pairs from a set of n-best hypotheses returned by an automatic speech recognition (ASR). The prediction is made in two steps. First, a deep learning architecture is used for the joint prediction of dialogue acts and the presence or absence of slots. Second, the same architecture is reused for predicting the values of the slots that were detected by the first joint-classifier. The deep architecture combines sentence and context representations. A convolutional neural network (CNN) (Collobert et al., 2011) is used to generate the sentence representation, while a long-short term memory (LSTM) network (Hochreiter and Schmidhuber, 1997) is used to generate the context representation. A non-linear function then combines the top layers of these neural networks and distinct softmax layers are used to predict the dialogue act and slots in the first joint model. In the second model, a single softmax predicts the possible values for each slot.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 258–267, Osaka, Japan, December 11-17 2016.

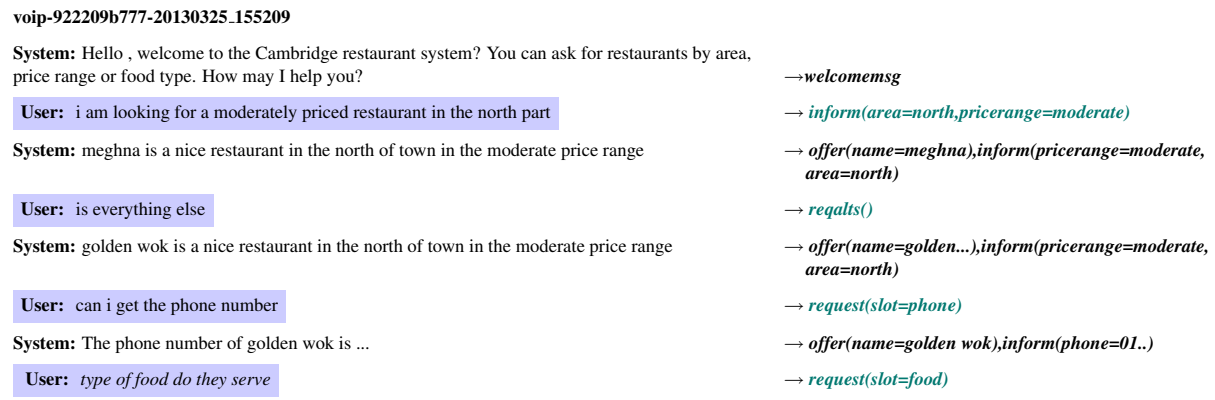


Figure 1: Excerpt from a dialogue in the DSTC2 corpus. The top-best ASR hypothesis is shown highlighted on the left, and the corresponding user semantics is shown highlighted on the right.

Our models are evaluated on two datasets DSTC2 (Henderson et al., 2014b) and In-car (Tsiakoulis et al., 2012) using accuracy, f-measure and the Item Cross Entropy (ICE) score (Thomson et al., 2008). We show that these models outperform previous proposed models, without using manually designed features and without any preprocessing of the input (e.g., stop words filtering, delexicalisation). They do this by exploiting distributed word representations and we claim that this allows semantic decoders to be built that can easily scale to larger and more complex dialogue domains.

The remainder of this paper is structured as follows. We first present related work in Section 2 and then we describe our architecture in Section 3. We describe the experimental setup in 4 and the evaluation results are introduced in Section 5. Finally, we present conclusions and future work in Section 6.

2 Related Work

Sequence tagging discriminative models such as CRFs and sequence neural networks have been widely explored for spoken language understanding. For instance, Recurrent Neural Networks have been proposed in (Yao et al., 2013; Mesnil et al., 2015) and generative Deep Neural Networks consisting of a composition of Restricted Boltzmann Machines (RBM) have been studied by (Sarikaya et al., 2011; Deoras and Sarikaya, 2013; Sarikaya et al., 2014). A combination of neural networks and triangular CRFs is presented in (Celikyilmaz and Hakkani-Tur, 2010), in which a convolutional neural network is used for extracting the input features of a triangular CRF in order to perform joint intent detection and slot filling. All these models use word-level semantic annotations. However, providing these word-level semantic annotations is costly since it requires specialised annotators. (Zhou and He, 2011) has proposed learning CRFs from unaligned data, however they use manually tuned lexical or syntactic features. In this work we avoid the need for word-level annotation by exploiting distributed word embeddings and using deep learning for feature representation.

Convolutional Neural Networks (CNNs) have been used previously for sentiment analysis (Kim, 2014; Kalchbrenner et al., 2014) and in this work we explore a similar CNN to the one presented by Kim (2014) for generating a sentence representation. However unlike Kim (2014), the input in not a single well formed sentence but a set of ill-formed ASR hypotheses. Additionally, the softmax layer used for binary classification (i.e., positive or negative sentiment) is replaced by a softmax layer for multiclass dialogue act prediction and a further softmax layer is added for each distinct slot in the domain. (Chen and He, 2015) proposed a CNN for generating intent embeddings in SLU, which uses tri-letter input vectors. Instead, in this paper the models are initialised with GloVe word embeddings (Pennington et al., 2014). These GloVe embeddings were trained in an unsupervised fashion on a large amount of data to model the contextual similarity and correlation between words. Chen and He's model aims to learn the embeddings for utterances and intents such that utterances with similar intents are close to each other in the continuous space. Although we share the same spirit, we use sentence embeddings not only for intent

(or dialogue act) recognition but also for slot-filling within a dialogue system and we combine them with embeddings for dialogue context.

Approaches for adaptive SLU have been proposed in (Ferreira et al., 2015; Zhu et al., 2014), however they focused more on domain adaptation on top of an existing SLU component. Moreover, they use classical discriminative models for SLU such as CRFs and SVMs that require manually designed features. In contrast, the focus of this paper is to exploit deep learning models for SLU, which learn feature representations automatically.

Recently, some researchers have focused on mapping word level hypotheses directly to beliefs without using an explicit semantic decoder step (Henderson et al., 2014a; Mrkšić et al., 2015). These systems track the user's goal through the course of the dialogue by maintaining a distribution over slot-value pairs. Such systems are interesting, but it is not clear that they can be scaled to very large domains due to the constraint of delexicalisation. Furthermore, they still require an explicit semantic decoding layer for domain identification and general *Topic Management*.

3 Deep Learning Semantic Decoder

We split the task of semantic decoding into two steps: (i) training a joint model for predicting the dialogue act and presence or absence of slots and (ii) predicting the values for the most probable slots detected in (i). As shown in Figure 2, we use the same deep learning architecture in both steps for combining sentence and context representations to generate the final hidden unit that feeds one or many softmax layers. In the first step, as shown in the Figure, there are distinct softmax layers for the joint optimisation of the dialogue act and each possible slot. In the second step there is a single softmax layer that predicts the value of each specific slot. In the following we explain this architecture in more detail.

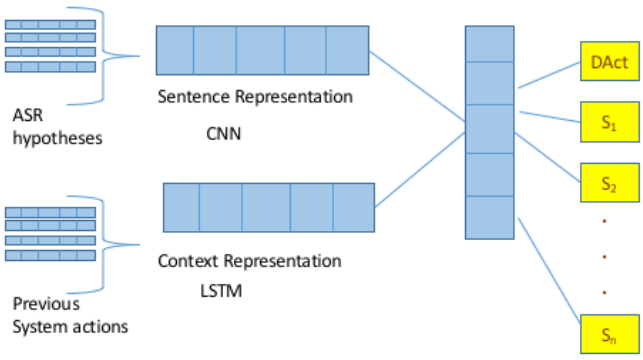


Figure 2: Combination of sentence and context representations for the joint prediction of dialogue acts and slots.

3.1 Sentence Representation

A CNN is used for generating the hypothesis representation, then these representations are weighted by their confidence scores and then summed up to obtain the sentence representation (Figure 3).

The CNN is a variant of (Kim, 2014), in which the inputs are the word vectors in each *ASR hypothesis*. Let x_i be a $k-$dimensional word embedding for the i-th word in a hypothesis. A hypothesis of length m is represented as: $x_{1:m} = \mathbf{x}_1 \bigoplus \mathbf{x}_2 \bigoplus ... \bigoplus \mathbf{x}_m$ where $\bigoplus$ is the concatenation operator. A convolutional operation is applied to a window of l words to produce a new feature.

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+l-1} + b) \tag{1}$$

where f is the hyperbolic tangent function; $w \in \mathbb{R}^{lk}$ is a filter applied to a window of l words and $b \in \mathbb{R}$ is a bias term. The filter is applied to every window of words in the sentence to produce a feature map.

$$\mathbf{c} = [c_1, c_2, ..., c_{n-l+1}] \tag{2}$$

with $\mathbf{c} \in \mathbb{R}^{n-l+1}$. A max pooling operation is then applied to give the maximum value $c = max\{\mathbf{c}\}$ as the representative feature for that filter. Multiple filters can be applied by varying the window size to obtain several adjacent features for a given hypothesis. These features $\hat{f}_j$ for the hypothesis $j \in H$ are then multiplied by the ASR confidence score p_j[1] and summed over all ASR hypotheses to generate a representation for the sentence s_t (Equation 3), as shown in Figure 3.

$$s_t = \sum_{j \in H} \hat{f}_j * p_j \tag{3}$$

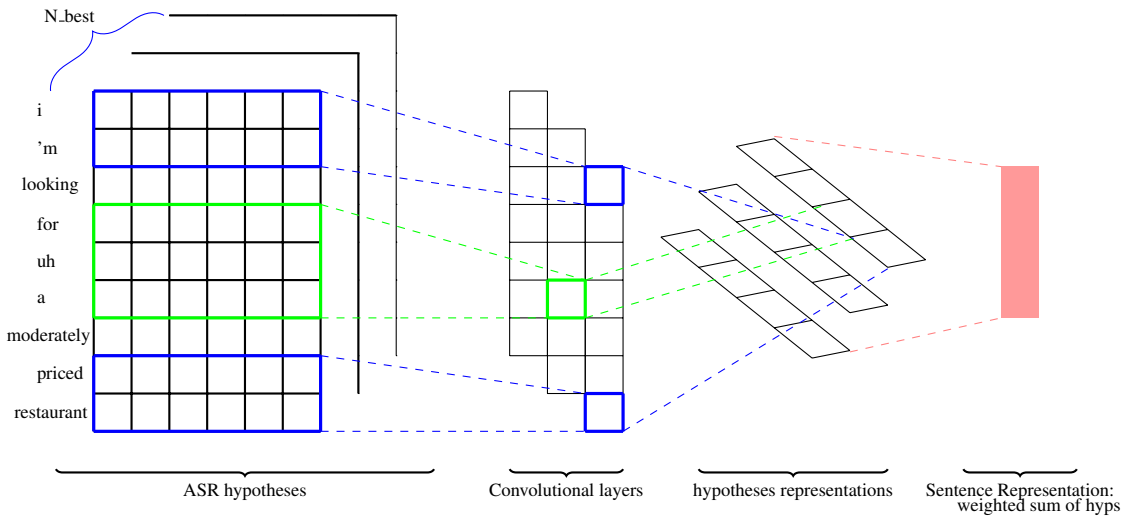


Figure 3: Sentence Representation: after applying convolution operations on the N-best list of ASR hypotheses, the resulting hidden layers are weighted by the ASR confidence scores and summed.

3.2 Context Representation

An LSTM (Hochreiter and Schmidhuber, 1997) is used for tracking the context implied by previous dialogue system actions. The top layer of this LSTM network then provides the context representation for decoding the current input utterance.

An LSTM is a sequence model that utilises a memory cell capable of preserving states over long periods of time. This cell is recurrently connected to itself and it has three multiplication units, an input gate, a forget gate and an output gate. These gating vectors are in [0,1]. The cell makes selective decisions about what information is preserved, and when to allow access to units, via gates that open and close. The LSTM transition equations are as follows:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}^{(i)} \cdot x_t + \mathbf{U}^{(i)} \cdot h_{t-1} + b^{(i)}), \\
\mathbf{f}_t &= \sigma(\mathbf{W}^{(f)} \cdot x_t + \mathbf{U}^{(f)} \cdot h_{t-1} + b^{(f)}), \\
\mathbf{o}_t &= \sigma(\mathbf{W}^{(o)} \cdot x_t + \mathbf{U}^{(o)} \cdot h_{t-1} + b^{(o)}), \\
\mathbf{u}_t &= tanh(\mathbf{W}^{(u)} \cdot x_t + \mathbf{U}^{(u)} \cdot h_{t-1} + b^{(u)}), \\
\mathbf{c}_t &= \mathbf{i}_t \bigodot \mathbf{u}_t + \mathbf{f}_t \bigodot \mathbf{c}_{t-1}, \\
\mathbf{h}_t &= \mathbf{o}_t \bigodot tanh(\mathbf{c}_t)
\end{aligned} \tag{4}$$

where h_t is the hidden unit at time step t, x_t is the input at the current time step, b is a bias, σ is the logistic sigmoid function and $\bigodot$ denotes elementwise multiplication.

As shown in Figure 1, system actions are encoded in the form of a system dialogue act plus one or more slot-value pairs. To track the history of system actions, slots and values are treated as words and the input x_t is formed from its corresponding word vectors. The length of the context can vary. We consider

[1]The posterior probability of hypothesis j in the N-best list.

all the system actions previous to the current user utterance, or a window l of the previous system actions. For instance, if we are currently processing the last user input in Figure 1, in which L is the total number of system actions, we can consider all previous system actions (L=4), or the last l system actions, where $l < L$.

3.3 Combining Sentence and Context

We study in this paper two ways of combining the sentence $\mathbf{s}_t$ and the context $\mathbf{h}_t$ representations. The first straightforward way is to apply a non linear function to their weighted sum:

$$\hat{\mathbf{h}}_\mathbf{t} = tanh(\mathbf{Ws} \cdot \mathbf{s}_t + \mathbf{Wc} \cdot \mathbf{h}_t) \tag{5}$$

The second way is to let the sentence representation be the last input to the LSTM network, then $\hat{\mathbf{h}}_\mathbf{t} = \mathbf{h}_t$.

For classification a softmax layer is used for each prediction:

$$P(Y = k|\hat{h}, W, b) = \frac{e^{(W_k \hat{h} + b_k)}}{\sum_{k\prime} e^{(W_{k\prime} \hat{h} + b_{k\prime})}} \tag{6}$$

where k is the index of the output neuron representing one class. For dialogue act classification k is one of the possible values: inform, request, offer, ... etc. For the slot prediction k is either 0 for absent or 1 for present. For slot-value prediction k will correspond to one of the possible values for each slot. For instance, for the slot price-range the possible values are cheap, moderate, expensive and dontcare. The result of the prediction is the most probable class:

$$\hat{y} = \text{argmax}_k(P(Y = k|\hat{h}, W, b)) \tag{7}$$

The back-propagation optimisation is done by minimising the negative log-likelihood loss function through stochastic gradient descent.

4 Experimental Evaluation

In this section we introduce the corpora, and describe the experiments performed and the evaluation metrics used.

4.1 Corpora

Experimental evaluation used two similar datasets: DSTC2 (Henderson et al., 2014b) and In-car (Tsiakoulis et al., 2012). Both corpora were collected using a spoken dialogue system which provides restaurant information system for the city of Cambridge. Users can specify restaurant suggestions by area, price-range and food type and can then query the system for additional restaurant specific information such as phone number, post code and address. The first dialogue corpus was released for the dialogue state tracking challenge and we use here the semantic annotations that were also provided [2]. The trainset has 2118 dialogues and 15611 turns in total while the testset has 1117 dialogues and 9890 turns in total.

The second corpus contains dialogues collected under various noisy in-car conditions. In a stationary car with the air conditioning fan on and off, in a moving car and in a car simulator (Tsiakoulis et al., 2012) [3]. The trainset has 1508 dialogues and 10532 turns in total and the testset has 641 dialogues and 4861 turns in total. Because of the noise, the average word error rate (WER = 37%) is significantly higher than for DSTC2 (around 29%).

4.2 Hyperparameters and Training

Dropout was used on the penultimate layers of both the CNN and the LSTM networks to prevent co-adaptation of hidden units by randomly dropping out a proportion of the hidden units during forward propagation (Hinton et al., 2012). The models were implemented in Theano (Bastien et al., 2012). We

[2]The DSTC2 corpus is publicly available in: `http://camdial.org/~mh521/dstc/`

[3]This corpus has been obtained in an industry funded project and therefore it is not available for public use.

used filter windows of 3, 4, and 5 with 100 feature maps each for the CNN. A dropout rate of 0.5 and a batch size of 50 was employed, 10% of the trainset was used as validation set and early stopping was adopted. Training is done through stochastic gradient descent over shuffled mini-batches with Adadelta update rule (we used an adadelta decay parameter of 0.95). To initialise the models, GloVE word vectors were used (Pennington et al., 2014) with a dimension $d = 100$. System-action word-embeddings are tuned during training, instead hypothesis word-embeddings are not because of the heavy computations.

4.3 Experiments

Step I: Joint classification of dialogue-acts and slots: We evaluated five different model configurations for the joint classification of dialogue-acts and presence or absence of slots.

- **CNN**: the softmax layers for the joint classification of dialogue acts and slots are connected directly to the CNN sentence representation with no context.
- **CNN+LSTM**: we study the influence of context by considering the previous system actions (Section 3.2, Eq. 5), here we study the different context length, by using a context window of 1, 4, and all the previous system actions, namely **CNN+LSTM_w1**, **CNN+LSTM_w4** and **CNN+LSTM_w** respectively.
- **LSTM_all**: Finally, we study the impact of long distance dependencies, by using mainly the LSTM model, with the previous system actions as input, but we inject the sentence representation as the last LSTM input.

Step II: Classification of slot value pairs: We select the best model in step I for predicting the presence of slots, then for each slot present we predict the value, by using again the best architecture from the previous step.

4.4 Evaluation Metrics

We evaluate the performance of our models by using the conventional metrics for classification, namely accuracy, precision, recall and F-measure (F1-score).

In addition, we used the ICE score (Eq. 8) between the hypotheses and the reference semantics (ie. ground-truth) to measure the overall quality of the distribution returned by the models(Thomson et al., 2008). Let U be the number of utterances and W be the number of available semantic items. Given $u = 1..U$ and $w = 1...W$, let:

$$c_{uw} = \begin{cases} p, \text{ the confidence assigned to the hypothesis that the } w^{th} \text{ semantic item is part of utterance } u, \\ 0, \text{ if none was assigned.} \end{cases}$$

$$\delta_{uw} = \begin{cases} 1, \text{ if the } w^{th} \text{ item is in the reference semantics for } u, \\ 0, \text{ otherwise} \end{cases}$$

and $N = \sum_{uw} \delta_{uw}$, be the total number of semantic items in the reference semantics.

$$\text{ICE} = \frac{1}{N_w} \sum - \log(\delta_{uw} c_{uw} + (1 - \delta_{uw})(1 - c_{uw}) \tag{8}$$

5 Results and Discussion

In this section we report the results on DSTC2 and In-car dialogue corpora.

Step I: Joint classification of dialogue-acts and slots: For this step, the classifiers must predict jointly 14 dialogue acts and 5 slots for the DSTC2 dataset as well as 14 dialogue acts and 7 slots for the In-car dataset. We evaluate both (i) using 10 fold cross-validation on the trainsets and (ii) on the corpora' testsets.

Table 1 shows the 10 fold cross-validation results on both corpora. These results suggest that for DTSC2, the context representation is not significantly impacting the prediction. Although, the model with a window of 4 ,**CNN+LSTM_w4**, improves slightly the accuracy and f1-score. On the In-car dataset, however, including the context does help to disambiguate the semantic predictions from ill-formed hypotheses. This is expected, since this data set has a much higher error rate and hence higher levels of confusion in the ASR output. Although there is no significant difference on the f1-score when using the immediate previous system act ($w1$) or a longer context, **CNN+LSTM_w** gives a better accuracy and a lower ICE score on this dataset.

Corpus	Metric	CNN	CNN+LSTM		
-	-	-	w1.	w4	w
DSTC2	acc.	$96.1\% \pm 0.002$	$95.97\% \pm 0.003$	$\mathbf{96.11}\% \pm 0.002$	$95.9\% \pm 0.003$
	P.	$90.17\% \pm 0.007$	$89.33\% \pm 0.007$	$89.77\% \pm 0.004$	$89.21\% \pm 0.008$
	R.	$85.61\% \pm 0.009$	$85.66\% \pm 0.007$	$86.40\% \pm 0.006$	$85.96\% \pm 0.006$
	F1	$87.8\% \pm 0.007$	$87.43\% \pm 0.006$	$\mathbf{88.03}\% \pm 0.004$	$87.53\%0.005$
	ICE	$\mathbf{0.245 \pm 0.013}$	0.275 ± 0.02	0.271 ± 0.02	0.277 ± 0.02
In-car	acc.	$90.45\% \pm 0.005$	$91.66\% \pm 0.003$	$91.49\% \pm 0.007$	$\mathbf{91.77}\% \pm 0.04$
	P.	$83.87\% \pm 0.01$	$84.31\% \pm 0.01$	$84.16\% \pm 0.01$	$83.89\% \pm 0.01$
	R.	$71.57\% \pm 0.007$	$74.91\% \pm 0.005$	$74.6\% \pm 0.02$	$74.76\% \pm 0.01$
	F1	$76.96\% \pm 0.008$	$\mathbf{79,16}\% \pm \mathbf{0.003}$	$78.85\% \pm 0.01$	$78.83\% \pm 0.007$
	ICE	0.498 ± 0.0013	0.457 ± 0.02	0.459 ± 0.03	$\mathbf{0.448 \pm 0.02}$

Table 1: 10 fold cross-validation evaluation of step I, the joint classification of dialogue acts and slots. Here we study the impact of the context by comparing **CNN** and **CNN+LSTM**.

Table 2 shows the results on the test sets. Consequently, when evaluating on the DSTC2 test set, a window of 4 ($w4$), performs slightly better than other window sizes and better than the simple **CNN** model. On the In-car testset, a context window of 4 outperforms all the other settings: **CNN+LSTM**. However, on this test set using the sentence representation as the last input to the LSTM context neural network (section 3.3) improves the f1-score and reduces the ICE error.

Corpus	Metric	CNN	CNN+LSTM			LSTM_all
-	-	-	w1.	w4	w	-
DSTC2	acc.	96.03%	95.79%	95.79%	95.69%	95.59%
	P.	89.73%	88.69%	88.95%	88.38%	88.15%
	R.	84.74%	85.09%	86.02%	85.96%	84.76%
	F1	87.14%	86.83%	**87.43%**	87.12%	86.42%
	ICE	**0.268**	0.278	0.292	0.297	0.308
In-car	acc.	**87.60%**	82.19%	82.25%	82.14%	82.3%
	P.	69.96%	79.52%	79.29%	80.25%	78.12%
	R.	62.14%	71.09%	71.59%	70.9%	74.04%
	F1	65.53%	74.89%	75.15%	75.02%	**75.9%**
	ICE	1.332	1.344	1.333	1.421	**1.106**

Table 2: Evaluation of the Step I on DSTC2 and In-car testsets. We also compare two ways of combining sentence and context representation: **CNN+LSTM** models (combining sentence and context representation through a non linear function) and **LSTM_all** model (embedding the sentence representation into the context model).

Step II: Prediction of slot value pairs For evaluating Step II, we selected the best model obtained during the 10-fold cross-validation experiments in terms of F1 score. For both corpora, this was the **CNN+LSTM_w4** configuration. For DSTC2, it was the 4^{th}-fold crossvalidation with $Acc = 90.42\%$,

$F1 = 88.69\%$ and ICE $= 0.251$. For In-car, it was the 5^{th}-fold crossvalidation with $Acc = 93.13\%$, $F1 = 81.49\%$ and ICE $= 0.393$. We used these models to classify whether a given slot appears in a given hypothesis or not. Then for that slot, we train another **CNN+LSTM_w4** classifier for predicting its values. In the In-car corpus the slot "type" has only one possible value "restaurant". Similarly, the slot "task" can only be the value "find". For these slots with only one value, we report values using the model of Step I, since it is enough to detect the slot in the utterance.

	DSTC2					In-car				
Slot	Acc.	P.	R.	F1	ICE	Acc.	P.	R.	F1	ICE
Slot[4]	95.29%	90.89%	95.72%	93.24%	0.478	89.92%	74.73%	61.56%	67.51%	0.743
Area	91.77%	92.66%	92.83%	92.74%	0.563	72.03%	72.56%	74.28%	73.41%	1.676
Food	71.37%	73.19%	76.02%	74.58%	1.989	66.46%	64.27%	68.70%	66.41%	2.309
Price	94.62%	91.33%	94.49%	92.89%	0.729	93.96%	88.77%	92.03%	90.37%	0.632
This[5]	98.70%	96.79%	93.92%	95.33%	0.113	97.16%	96.14%	84.72%	90.07%	0.214
Type	-	-	-	-	-	95.56%	95.09%	86.69%	90.69%	0.290
Task	-	-	-	-	-	97.12%	83.24%	64.93%	72.95%	0.175
Mean	90.35%	88.97%	90.60%	89.76%	0.774	87.47%	82.11%	76.13%	78.77%	0.863
St.Dev.	0.109	0.091	0.082	0.085	0.715	0.128	0.121	0.118	0.112	0.821

Table 3: Evaluation of the step II: the slot-value pairs classification on DSTC2 and In-car.

Given that there is no domain specific delexicalisation, the models achieve a good level of performance overall (Table 3). Note that the slot "food" has 74 possible values in DSTC2 and 25 in In-car. Hence, this slot has much higher cardinality than all the other slots.

Overall performance A baseline for assessing overall performance is provided by the model presented in (Henderson et al., 2012), in which the vector representation is obtained by summing up the frequency of n-grams extracted from the 10-best hypotheses, weighted by their confidence scores. Here we compare our performance against Henderson's model with and without context features, namely WNGRAMS+Ctxt and WNGRAMS repectively. Henderson reported his results on the In-car dataset. A similar model, namely SLU1, was evaluated on DSTC2 in (Williams, 2014). Both implementations consist of many binary classifiers for dialogue act and slot-value pairs.

Corpus	Model	F1	ICE
DSTC2	SLU1 (Williams, 2014)	80.2%	1.943
	CNN+LSTM_w4	**83.59%**	**0.758**
In-car	WNGRAMS (Henderson et al., 2012)	70.8%	1.76
	WNGRAMS+Ctxt (Henderson et al., 2012)	**74.2%**	1.497
	CNN+LSTM_w4	73.06%	**1.106**

Table 4: Overall performance of the setting CNN+LST_w4 semantic decoder.

In terms of the ICE score, the model CNN+LSTM W4 outperforms all the baselines (Table 4). In terms of the F1 score, the model significantly outperforms the SLU1 and WNGRAMS baselines. However it is slightly worse than WNGRAMS+Ctxt, which has been enhanced with context features on In-car. Remember however, that our model uses only word-embeddings for automatically generating sentence and context representations without having any manually designed features or using explicit application specific semantic dictionaries.

[4]"Slot" is used when no value is given for the slot (e.g., "What kind of food do they serve?"/request(slot=food)).
[5]"This" is used for annotating elliptical utterances (e.g., "I dont care"/inform(this='dontcare')).

6 Conclusion and Future Work

This paper has presented a deep learning architecture for semantic decoding in spoken dialogue systems that exploits semantically rich distributed word vectors. We compared different models for combining sentence and context representations. We found that context representations significantly impact slot F-measure on ASR hypotheses generated under very noisy conditions. The combination of sentence and context representations, with a context window of 4 words, outperforms all the baselines in terms of the ICE score. In terms of the F1 scores, our model outperforms the baseline on the DSTC2 corpus and the baseline without manually designed features on the In-car corpus. Although the F-score of our model does not outperforms the baseline enriched with context features on the In-car corpus, the proposed model remains competitive, especially considering that our model requires no manually designed features or application specific semantic dictionaries.

7 Future Work

Semantic distributed vector representations can be used for detecting similarity between domains. As future work, we want to study the adoption of the sentence and the contex representations generated in the Step I (i.e., the joint prediction of dialogue act and slots) within a *Topic Management* in multi-domain dialogue systems. The Topic Manager is in charge of detecting the domain and the intention behind users' utterances. Furthermore, it would be interesting to study these embeddings for domain adaptation on potentially open-domains.

Acknowledgments

This research was partly funded by the EP-SRC grant EP/M018946/1 Open Domain Statistical Spoken Dialogue Systems. The data used in this paper was produced in an industry funded project and it is not available for public use.

References

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. nips workshop on deep learning and unsupervised feature learning.

Asli Celikyilmaz and Dilek Hakkani-Tur. 2010. Convolutional neural network based semantic tagging with entity embeddings. *genre*.

Yun-Nung Chen and Xiaodong He. 2015. Learning bidirectional intent embeddings by convolutional deep structured semantic models for spoken language understanding. In *Extended Abstract of The 29th Annual Conference on Neural Information Processing Systems–Machine Learning for Spoken Language Understanding and Interactions Workshop (NIPS-SLU)*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Anoop Deoras and Ruhi Sarikaya. 2013. Deep belief network based semantic taggers for spoken language understanding. In *INTERSPEECH*, pages 2713–2717.

Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefèvre. 2015. Online adaptative zero-shot learning spoken language understanding using word-embedding. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5321–5325. IEEE.

Matthew Henderson, Milica Gašić, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young. 2012. Discriminative Spoken Language Understanding Using Word Confusion Networks. In *Spoken Language Technology Workshop, 2012. IEEE*.

M. Henderson, B. Thomson, and S. J. Young. 2014a. Word-based Dialog State Tracking with Recurrent Neural Networks. In *Proceedings of SIGdial*.

Matthew Henderson, Blaise Thomson, and Jason Williams. 2014b. The second dialog state tracking challenge. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, volume 263.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

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

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain Dialog State Tracking using Recurrent Neural Networks. In *Proceedings of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

R. Sarikaya, G. E. Hinton, and B. Ramabhadran. 2011. Deep belief nets for natural language call-routing. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5680–5683, May.

Ruhi Sarikaya, Geoffrey E Hinton, and Anoop Deoras. 2014. Application of deep belief networks for natural language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(4):778–784.

Blaise Thomson, Kai Yu, Milica Gasic, Simon Keizer, Francois Mairesse, Jost Schatzmann, and Steve J Young. 2008. Evaluating semantic-level confidence scores with multiple hypotheses. In *INTERSPEECH*, pages 1153–1156.

Pirros Tsiakoulis, Milica Gašic, Matthew Henderson, Joaquin Planells-Lerma, Jorge Prombonas, Blaise Thomson, Kai Yu, Steve Young, and Eli Tzirkel. 2012. Statistical methods for building robust spoken dialogue systems in an automobile. *Proceedings of the 4th applied human factors and ergonomics*.

Gökhan Tür, Anoop Deoras, and Dilek Hakkani-Tür. 2013. Semantic parsing using word confusion networks with conditional random fields. In *INTERSPEECH*, pages 2579–2583.

Jason D Williams. 2014. Web-style ranking and slu combination for dialog state tracking. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 282–291.

Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu. 2013. Recurrent neural networks for language understanding. In *INTERSPEECH*, pages 2524–2528.

Deyu Zhou and Yulan He. 2011. Learning conditional random fields from unaligned data for natural language understanding. In *European Conference on Information Retrieval*, pages 283–288. Springer.

Su Zhu, Lu Chen, Kai Sun, Da Zheng, and Kai Yu. 2014. Semantic parser enhancement for dialogue domain extension with little data. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 336–341. IEEE.

Predictive Incremental Parsing Helps Language Modeling

Arne Köhn and **Timo Baumann**
Department of Informatics
Universität Hamburg
{koehn,baumann}@informatik.uni-hamburg.de

Abstract

Predictive incremental parsing produces syntactic representations of sentences as they are produced, e. g. by typing or speaking. In order to generate connected parses for such unfinished sentences, upcoming word types can be hypothesized and structurally integrated with already realized words. For example, the presence of a determiner as the last word of a sentence prefix may indicate that a noun will appear somewhere in the completion of that sentence, and the determiner can be attached to the predicted noun. We combine the forward-looking parser predictions with backward-looking N-gram histories and analyze in a set of experiments the impact on language models, i. e. stronger discriminative power but also higher data sparsity. Conditioning N-gram models, MaxEnt models or RNN-LMs on parser predictions yields perplexity reductions of about 6 %. Our method (a) retains online decoding capabilities and (b) incurs relatively little computational overhead which sets it apart from previous approaches that use syntax for language modeling. Our method is particularly attractive for modular systems that make use of a syntax parser anyway, e. g. as part of an understanding pipeline where predictive parsing improves language modeling at no additional cost.

1 Introduction

N-gram models are the best performing language models capable of online decoding. Interestingly, they perform well despite discarding all information contained in a sentence except the last N-1 words to predict the next one. At least intuitively, it seems implausible that N-grams are a natural model for sentences. In particular, humans seem to use strong predictive skills that rely on non-local aspects of pragmatics (i.e., the conversational context), semantics (i.e., meaningful continuations), or syntax (i.e., syntactically correct continuations). Sturt and Lombardo (2005) show that humans have strong expectations about upcoming words based on the predicted syntactic structure of the sentence. This includes effects spanning long distances in the sentence. For example, reading a word that does not match expectations slows down the reading process.

Syntactic dependency trees provide long-spanning context information not contained in N-grams. They have, however, not widely been used for language modeling. Previous work (see Sec. 2) has integrated parsing into the language modeling process either by post-hoc (re-)scoring of full sentences or by parsing the partial sentence including the word to be predicted. The first approach is not suitable for online decoding at all and the second is computationally expensive as outlined below.

We propose to use a predictive dependency parser that produces syntactic structure for yet-unfinished sentences. The parser is able to include upcoming ('virtual') nodes in the predicted structure (see Sec. 3). We use this forward-looking information to condition our models. In a range of experiments, we confirm the value of parser predictions using a very simple setup in Sec. 4. In Sec. 5, we integrate the predictions with several types of language modeling approaches, and obtain perplexity reductions in all of them.

Author ordering based on a fair coin toss.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 268–277, Osaka, Japan, December 11-17 2016.

2 Related Work

Existing approaches that have exploited syntax for language modeling fall into 3 clusters: post-hoc reranking, the application of tree probabilities, and the use of predictions (as in our work).

Recent research focused on the reranking of n-best hypotheses or lattices either based on the probability of the syntax tree for the full sentence (Filimonov and Harper, 2009; Charniak, 2001; Tan et al., 2012) or by training a discriminative model (Collins et al., 2005). A special case of reranking is the task of sentence completion, where one word of a sentence is missing and the correct word out of five possibilities needs to be selected to fill the gap (Zweig and Burges, 2011). For evaluation purposes, the possible words are selected specifically to have a similar probability under a plain N-gram model. Zhang et al. (2016) propose a model based on LSTMs that predicts dependency trees instead of word sequences. They obtain state of the art results for the sentence completion challenge by computing the probabilities of the alternatives and choosing the best one. However, these models do not cope with sentence prefixes, and hence they can only be applied non-incrementally in a rescoring fashion, e.g. to improve ASR results after a first pass over the input data; thus, they are not applicable to interactive use-cases which require incremental processing.

Parsers that assign probabilities to the derived parse tree can be used for language modeling by comparing parse tree probabilities for all possible continuations of the sentence prefix, e. g using a limited-domain hand-written PCFG with learned rule-weights (Jurafsky et al., 1995), or by learning structure using a hierarchical HMM (Schwartz et al., 2011). As an alternative, Roark (2001) estimates the probability of a sentence prefix by measuring the probability of all parse trees (within a beam) that are derivable for the prefix ending in the queried N-gram divided by the probability of all parse trees for the prefix ending at the N-1'th word. For these approaches, sentence prefixes including the N'th word must be computed, which is computationally expensive. When obtaining a probability distribution for the next word, the cost grows with the vocabulary size. In contrast, our approach parses only once, up to the N-1'th word (i. e. the context portion of the prefix when determining the N'th word).

Chelba and Jelinek (1998) use a tree adjoining grammar to parse the sentence prefix up to (but not including) the word being queried, and the parser state (the top two elements of the stack) is used to condition the language model upon. The probability is summed over all possible trees for the prefix. Thus, again many parses are required for a single query, resulting in a high complexity. Nevertheless, this approach shows many similarities to ours as parsing is done incrementally and the parser state implicitly encodes information about upcoming structure. In our model, we make the predictions explicit which may also be useful to other modules in a system.

The main contributions of our work are as follows: first, our method only needs raw text (not treebanks) for estimating syntactic LMs (as our parser is trained independently) while retaining high run-time performance on single machines. Tan et al. (2012) use a similar amount of data but need 400 servers to apply their admittedly more sophisticated models which also include semantics and topic for full-document modelling. Second, our approach is applicable to online decoding for incremental systems such as highly interactive spoken dialogue systems, whereas other methods are often limited to post-hoc lattice rescoring.

3 Predictive Parsing

Dependency parsers generate the syntax tree for a sentence by attaching every token (the dependent of the relation) to another token (the head) or a root node. For sentence prefixes, the head of a dependent may not yet be available resulting in an incomplete tree (as in Figure 1 (a)).

Trees can be completed by using what Beuck et al. (2013) call virtual nodes which serve as stand-ins for upcoming words. Words without a matching head in the prefix can be attached to a virtual node (see Figure 1 (b)) which is fully integrated into the syntactic structure. Such nodes can also be introduced when no word needs to be attached to them, e.g. because otherwise a valency can not be satisfied (see Figure 1 (c)).

Each virtual node has a type corresponding to the PoS of the words it can stand for. We use the prediction about the upcoming word types for language modeling, following the intuition that e. g. a verb might be more likely when the parser predicts that a verb is missing to complete the sentence.

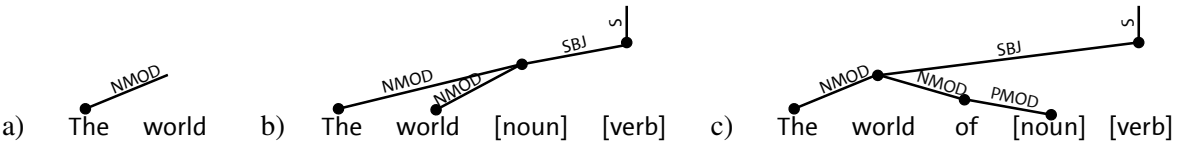


Figure 1: Dependency trees for prefixes of the sentence *The world of politics has no rules.* showing: (a) an incomplete tree (no head exists for *world*), (b) a predicted noun and verb head (lexicalization leads the parser to assume that *world* is part of a more complex noun phrase), and (c) an additional predicted noun dependent when *of* is added. (b) and (c) are actual outputs of the predictive parser.

We use a parser based on TurboParser (Martins et al., 2013) which was extended by Köhn and Menzel (2014) to produce predictive analyses of sentence prefixes. The parser predicts up to one upcoming verb and one upcoming noun, which results in a good coverage/precision trade-off for English. The parser is trained on the Penn-Treebank (Marcus et al., 1994) and not specifically tuned for language modeling nor for the corpus used for estimating the LM. Predictions of vn nodes reach an F-score of .68 (precision: .79; recall: .60). The parser output for full sentences is as accurate as the original TurboParser.

Throughout this paper, we abbreviate the prediction of the virtual verb and noun as vn where v stands for the verb and n for the noun. For specific predictions, $\overset{+}{v}$ and $\overset{+}{n}$ denotes that the corresponding virtual node was predicted, $\overset{-}{v}$ and $\overset{-}{n}$ that it was not. The distribution of predictions for sentence prefixes is shown in Table 1; as can be seen, both v and n split the prefixes roughly equally into four parts, yielding an information of 1.90 bit. Other forward-looking syntactic features (e.g. dependency direction of an expected virtual node or the dependency type) would likely be less informative, hence we focus on only these two features in our work. We experiment with models that use both or just one of the predictions.

sentence prefixes		verb predicted		
		$\overset{-}{v}$	$\overset{+}{v}$	Sum
noun predicted	$\overset{-}{n}$	41.6	22.7	**64.3**
	$\overset{+}{n}$	18.7	17.0	**35.7**
	Sum	**60.3**	**39.7**	

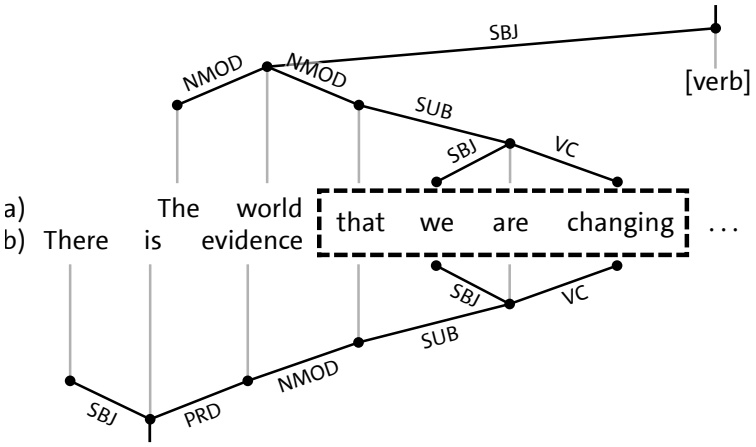
Table 1: Distribution (in percent) of virtual node predictions for sentence prefixes in the Billion Word Corpus.

4 Basic Idea and Proof-of-Concept

Our experiments are based on the assumption that probabilities for a word (such as "likes") depend on whether a verb (or noun) is structurally required to complete the sentence. In addition, we believe that this *forward-looking* structural information is not well-captured by the *backward-looking* local context of an N-gram. Notice that we do not make any assumption on the kinds of probability shifts but only the assumption that *some* shift occurs, i.e. "likes" might as well be preferred when a (plural) noun is predicted.

A language model computes the probability of a sentence: $P(w_1 \dots w_n)$. Applications such as speech recognition perform online decoding and use the decomposition according to the chain rule $P(w_1 \dots w_n) = \prod_{i=1}^{n} P(w_i|w_1 \dots w_{i-1})$, as well as the (N-gram) approximation based on the Markov assumption ($P(w_i|w_1 \dots w_{i-1}) \approx P(w_i|w_{i-N} \dots w_{i-1})$). This approximation limits language models to local contexts but makes the probability estimation problem tractable by reducing data sparsity. We enhance the model by including parser predictions $p_{i-1} \in \{\overset{+}{v}\overset{+}{n}, \overset{+}{v}\overset{-}{n}, \overset{-}{v}\overset{+}{n}, \overset{-}{v}\overset{-}{n}\}$ extracted from the predictive parse for the *full* history of the sentence $w_1 \dots w_{i-1}$. Thus, the parser predictions encode additional information about the continuation that enhance the local history in a very dense form – using only two bits.

The non-locality of parsing predictions is exemplified in Figure 2 (the example is restricted to just the verb prediction v). In the two sentences given in the figure, the local N-gram context is identical for either sentence prefix. In fact, the cause for different parser predictions can be arbitrarily far into the past (for example by exchanging *we* for a complex noun phrase in the example). For the given N-gram context, we queried language models that make use of the (non-)prediction of verbs ($\overset{+}{v}$ vs. $\overset{-}{v}$, as described below) and differences in $\log_{10}$ probabilities for some continuations of either sentence are shown in tabular form. As expected, we find that plausible continuations of Sentence 2 (a) are much more probable in the $\overset{+}{v}$ case, and plausible continuations of 2 (b) are more probable in the $\overset{-}{v}$ case.



continuation	$\log_{10}$prob difference between $\overset{+}{v}/\bar{v}$ model
has/hasn	+1.18 / +1.12
will	+1.10
is/isn	+1.08 / +0.97
and	+0.05
in	−0.06
worlds	−0.34
it	−1.28
a/an	−1.41 / −1.35
our	−1.69
lives	−2.52

Figure 2: Predictive parses for two sentence beginnings (left). While the parser predicts a verb to be neccessary to complete (a), it does not predict a verb for (b). The differences in the sentences (that cause the different parses) lie outside of the local N-gram context (dashed box). Possible continuations and the relative differences in assigned probabilities depending on $\overset{+}{v}/\bar{v}$ context are shown in tabular form (right).

To make use of the predictions in an N-gram model, we split the N-gram counts by the parser predictions of the corresponding prefix histories: an N-gram $(w_{i-N} \dots w_{i-1} w_i)$ is sorted according to p_{i-1}.[1] With the prefixes split into four bins ($\overset{+}{v}\overset{+}{n}$, $\overset{+}{v}\bar{n}$, $\bar{v}\overset{+}{n}$, and $\bar{v}\bar{n}$), we train four separate N-gram models. During decoding, we switch between these four models: given a sentence prefix $(w_1 \dots w_{i-1})$, we extract p_{i-1} and query the corresponding model for the probability of w_i. Note that the output of the parser does not depend on w_i and we therefore only need to parse once for every sentence prefix even if we query the full probability distribution (as in speech recognition). We also perform experiments with only two bins by splitting either by v or by n, obtaining two N-gram models.

All experiments in this paper are carried out on the Billion Word Corpus (BWC; Chelba et al. (2013)), for which we parsed every sentence prefix.[2] N-gram models are generated with SRILM (Stolcke et al., 2011) using standard settings (modified Kneser-Ney smoothing (Chen and Goodman, 1996) and interpolation, a limited vocabulary of 100,000 words, and dropping singleton N-grams for N>2).[3]

4.1 Exemplary Effects of Parsing Splits on Trigrams

We first illustrate the effects of parser predictions on probability distributions for prefixes ending in *the world*. The probabilities of the most frequent continuations for *the world* (i.e., standard N-gram probabilities) as well as probabilities under the four different split models are shown in Table 2 (middle part). We discuss the highlighted numbers: *the world 's* has a high probability when a noun is predicted (particularly $\bar{v}\overset{+}{n}$). This subsumes cases where a verb has a valency to which *world* is a bad fit.[4] Hence, *the world* is regarded as a possessive adjective to the predicted argument and *'s* makes the connection between the two. The continuation "." has a high probability when neither a verb nor a noun is predicted to continue the sentence. A comma is more likely under the $\overset{+}{v}\overset{+}{n}$ model, presumably because this constellation dominates at the end of introductory prepositional phrases. Lastly, *and* is most likely under $\bar{v}\bar{n}$, similarly to the full stop, as it connects material to an already complete structure.

While these observations may or may not be interesting linguistically, our main point is that probabilities shift (sometimes radically, as highlighted in the right part of the table) when dissecting the N-gram history by our longer-range vn information. This increased discriminatory power is what our method is about.

[1] Simply including parser prediction in an N-gram model like a context word (using the N-grams $(w_{i-N} \dots w_{i-1} p_{i-1} w_i)$) would break the count-of-count assumptions for Kneser-Ney smoothing.

[2] This takes 30 ms per prefix, totalling about one CPU year; parsing results as well as code to replicate our experiments are available at `https://nats-www.informatik.uni-hamburg.de/PIPLM` to encourage further research.

[3] SRILM allows to manipulate N-gram counts and is still able to compute correct backoff values unlike e.g. KenLM which, however, would allow to include singleton N-grams. Data sparsity (see below) would be less grave if singletons were included.

[4] Note that since our parser is lexicalized, it automatically learns valency relations.

		probabilities				relative change			
tri-gram	avg	$\overset{+}{v}\overset{+}{n}$	$\overset{+}{v}\overset{-}{n}$	$\overset{-}{v}\overset{-}{n}$	$\overset{-}{v}\overset{+}{n}$	$\overset{+}{v}\overset{+}{n}$	$\overset{+}{v}\overset{-}{n}$	$\overset{-}{v}\overset{-}{n}$	$\overset{-}{v}\overset{+}{n}$
the world 's	.33	.41	.37	.28	**.81**	×1.2	×1.1	÷1.2	×2.5
the world .	.14	.02	.03	**.19**	.01	**÷7.0**	**÷4.9**	×1.4	**÷13**
the world ,	.10	**.17**	.07	.11	.02	×1.6	÷1.5	×1.1	**÷4.9**
the world and	.02	.01	.01	**.03**	.004	÷2.2	÷2.2	×1.3	**÷5.3**
the world *everything else*	.41	.39	.52	.39	.16	÷1.1	×1.3	÷1.1	÷2.6

Table 2: Some tri-grams and their probabilities in the full corpus (avg) vs. split by parser prediction.

4.2 The Detrimental Effect of Data Sparsity

The splitting method outlined above increases data sparsity in the sub-models for each of the four vn bins. We here differentiate between gains from parsing prediction-induced discriminatory power and losses from splitting-induced data sparsity. For this purpose, we devise a baseline model which replaces the parser predictions with random bins, using the same distribution as in Table 1. If the parser's predictions were to provide no additional information, both models should obtain the same perplexity, as they are comparably impacted by data sparsity.

Table 3 shows the cross-entropies obtained by our syntax splitting model, the baseline model, and standard N-gram models trained on the full material.[5] We find that data sparsity inhibits the results for N>2-grams. While our method achieves a gain from syntax predictions (e. g. ~.3 bit for 4/5-grams), additional data sparsity deteriorates results slightly more (~.4 bit for 4/5-grams). Nevertheless, we find that syntax gain even increases with N-gram order, that is higher-order N-grams do not implicitly encode longer-range information in the same way that syntax predictions do. We also find that both v and n help similarly and that their gains roughly add up (i. e. they are orthogonal). In total, each of v/n in isolation shows slightly better performance than combined vn splitting given their lower data sparsity.

4.3 Exemplary Comparison of Split and Standard Model

To further analyze the strengths and weaknesses of the split model, we look word-by-word at the probability assignments of standard 5-gram models and vn state-specific 5-gram models. We restrict this analysis to 500 test sentences with 6-14 tokens and focus attention on sentences with large differences.

In the qualitative analysis, we find some patterns that are illustrated by the sentence plots in Figure 3. Each plot shows for every word the probability assigned by either model, the parser's prediction for the prefix leading up to this word, and the backoff order necessary if the queried 5-gram is not contained in the model. The backoff order is a strong indicator for data sparsity pressure.

[5]Entropies reported in Table 3 are higher than for the final N-gram results in Section 5.1, as the preliminary models employed here are not order-interpolated and sentence-ends are excluded from perplexity computation.

		syntax pred. A:	random splits B:	standard C:	syntax gain B-A:	splitting loss B-C:
2-grams	vn	7.819	7.968	7.881	0.149	0.087
	v	7.837	7.918		0.081	0.037
	n	7.861	7.917		0.057	0.036
3-grams	vn	6.988	7.209	6.938	0.221	0.271
	v	6.969	7.071		0.102	0.133
	n	6.959	7.067		0.108	0.129
4-grams	vn	6.735	7.025	6.633	0.290	0.392
	v	6.703	6.839		0.136	0.206
	n	6.671	6.830		0.159	0.197
5-grams	vn	6.685	6.994	6.566	0.309	0.428

Table 3: Cross-entropies (in bit; lower is better) obtained by splitting the training data according to the parser's prediction (verb, noun, or both), vs. random splitting and comparison to standard models.

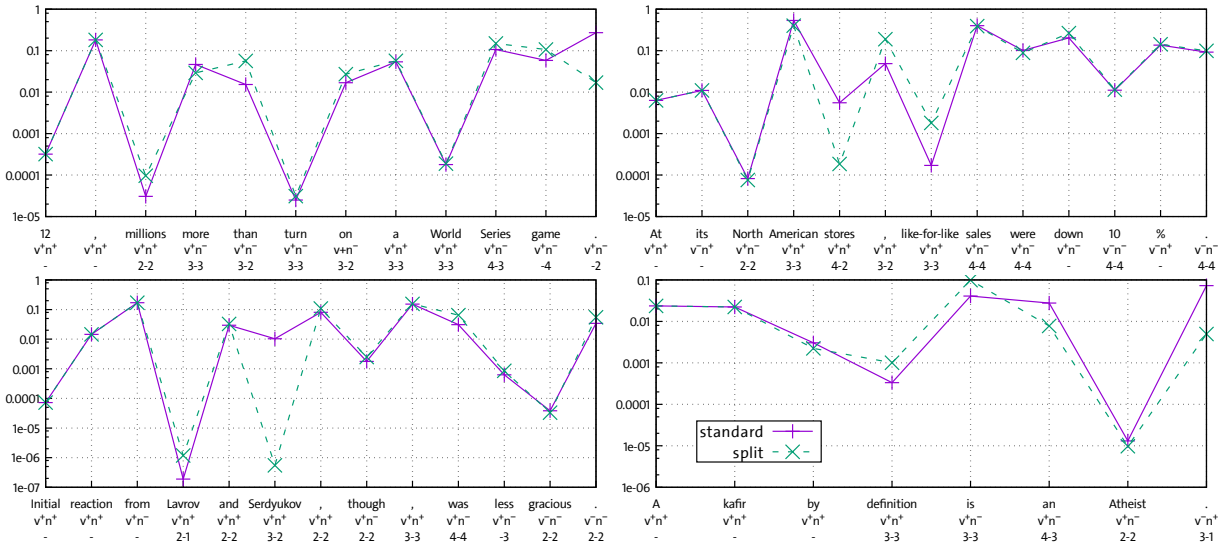


Figure 3: Sentences with their per-word probabilities in standard and split models. For each word, we show syntax prediction and the N-gram order backed off by the standard and split model respectively, if backoff is needed (e.g., "Lavrov $\overset{+}{v}\overset{+}{n}$ 2-1" indicates the parser predicting both $\overset{+}{v}$ and $\overset{+}{n}$ for the prefix up to "from" and both models backing off: the standard model to bigrams, the split model even to unigrams).

Probability estimates are **identical for the N-1 sentence-initial words** as syntax predictions are implicitly contained in the N-gram history and both models provide the same information (lower N if a model is forced into backoff). Models perform similarly under no or little data sparsity, i. e. no backoff (or just to 4-grams). **Moderate improvements** often occur **under data sparsity** when both models back off to the same order. The first sentence shows some improvements for the syntax predictions even when using lower N-gram orders (syntax being more useful than one more word of history) but fails at predicting the sentence final full stop (the parser still expects a verb and this could be classified as a parsing error), in this case due to the incompleteness of the sentence (it is both authors' feeling that this sentence *should* go on).

Inherently, the standard model is never impacted more strongly by data sparsity than the split model. Thus, **critical cases occur when the split model needs to back off more** than the standard model. While positive effects are also noticeable to some extent for the second and third sentence, these sentences are hurt badly by not 'knowing' infrequent co-occurrences, i. e. N-grams in which the last word has a high conditional probability given the full history of the N-gram: "North American stores" (where "stores" is a likely continuation for "North American" but not for "American" *per se*), and "Lavrov and Serdyukov" (where "Lavrov" must be in the history to assign a relevant probability, yet the split model backs off to "and Serdyukov"). Further analysis of this last example shows that there are two occurrences of the trigram in the training data, which fall into two different prediction states ($\overset{+}{v}\overset{+}{n}$ and $\overset{-}{v}\overset{+}{n}$). As singleton N-grams are ignored, both occurrences of the trigram are lost and not used in the split model.

A histogram of $\log_{10}$prob differences is shown in Figure 4 (left side; notice the logarithmic y-axis). The majority of the differences are positive (55 %), indicating that our method is helpful more often than not. However, the very long tail of rare gross mistakes results in a overall slightly negative $\log_{10}$prob loss of -0.025, in line with our analysis of the examples in Figure 3 and the overall loss in Table 3.

In summary, we find that most often, probability estimates are similar or slightly improve with syntax predictions as used by the split model due to the better discrimination from non-local context. On the other hand, splitting increases the random variation of N-gram counts in particular for rarely seen N-grams; more often than not, the positive effect appears to outweigh the negative effect. Unfortunately, few but large errors are introduced systematically by the fact that each of the split model's sub-models is trained on less data. We deal with data sparsity in the following section, where we build prediction-enhanced models that outperform their baselines.

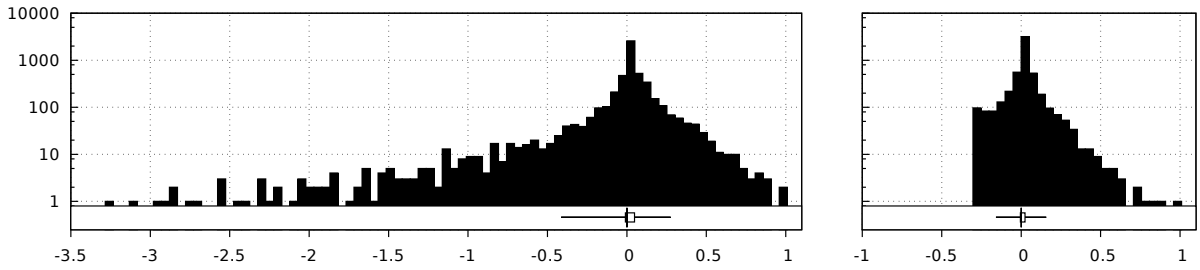


Figure 4: Left: Histogram over $\log_{10}$prob differences between the split and the standard model. Right: Differences between the interpolated and standard model (see Section 5.1). Notice the logarithmic y-axis. The boxplots below show 25/75% (box) and 5/95% (whiskers) intervals and median.

5 Experiments

To assess the merit of predictive parsing, we perform a wide range of experiments. We use SRILM (Stolcke et al., 2011) for N-grams on the BWC, for which Chelba et al. (2013) quote a cross-entropy of 6.08 bit for unpruned interpolated Kneser-Ney smoothed 5-grams. We build MaxEnt N-gram models (Rosenfeld, 1994) as well as RNN-LMs (Mikolov et al., 2011) using faster-rnnlm.[6] For the BWC, faster-rnnlm in its best setting (after extensive hyper-parameter search) results in a cross-entropy of 6.8 bit and combined with its MaxEnt model of 6.4 bit (the MaxEnt model alone has not previously been benchmarked).[6]

The numbers reported below are not necessarily directly comparable to reference numbers for several reasons: we use limited-vocabulary N-grams and separate OOV handling as we plan to apply our models to speech recognition. In contrast, faster-rnnlm by default discards all sentences containing OOV tokens which results in lower perplexity. We use SRILM defaults (pruned N-grams) to create models that easily fit in memory, and do not search for optimal RNN hyper-parameters. Nonetheless, given the breadth of our experiments, the general findings of our work are likely independent of these particularities.

5.1 Interpolated N-gram models

We first present interpolated N-gram models that reduce the impact of additional data sparsity in the plain syntax splitting models presented in Section 4.2. Interpolation exploits the averaging effect when combining models with different characteristics. As noted above, the syntax-enhanced model is slightly better *often* but much worse sometimes (see Figure 4, left side). Using interpolation, the outliers can be greatly reduced in magnitude by averaging the models; in particular, a probability from the interpolated model is at worst half as probable as the probability from the better model. Therefore, no $\log_{10}$prob difference is smaller than $log_{10}(0.5) \approx -0.3$ (cmp. Figure 4, right side). Although the magnitude of improvements is also reduced, that effect is much smaller and the overall average advantage over the standard model turns from negative to positive. In effect, interpolation allows to optimize the tradeoff between discrimination and accumulation of evidence under data sparsity, and we try several strategies:

1. The split model uses two binary predictions, thus each of the sub-models is trained on (roughly) one quarter of the training data, only. One possible route to improvements is to build separate sub-models for the v and the n variable (**1a**), each of which can be trained with roughly half of the training data, i.e. with less data sparsity. We can then select the two models that match the vn state and interpolate between them. An alternative approach for reducing data sparsity (**1b**) is to join the N-gram counts of the corresponding v and n state and to build pre-aggregated models. (E. g. the model for $\bar{v}\overset{+}{n}$ would accumulate counts from the $\bar{v}$ and the $\overset{+}{n}$ bins.) Either way, we account for the fact that an N-gram may be seen during testing in a different vn state than in training (that is nevertheless related by one of the variables).

2. We interpolate between the standard model which is trained on all data and the split models which are more discriminative (**2a**). We also combine this method with **1b** as **2b**.

The cross-entropies of the plain syntax-splitting method (**0**) and the standard 5-gram model as well as the experimental conditions described above are given in Table 4. We find that interpolating the vn

[6] github.com/yandex/faster-rnnlm

model type	setting	standard	syntax	gain
5-grams	**0.** plain splitting	6.253	6.374	−0.12
	1a. interp. from $v \times n$		6.254	−0.001
	1b. joint counts		6.187	0.07
	2a. vn interp. w/ std		6.234	0.02
	2b. joint counts interp. w/ std		6.170	**0.08**
MaxEnt	2-grams	8.41	8.31	0.10
	3-grams	7.86	7.78	0.08
	4-grams	7.88	7.87	0.01
	4-grams (large GPGPU)	7.55	7.49	**0.06**
RNN-LM	syntax in input	7.94	7.90	0.04
	syntax in NCE		7.89	**0.05**

Table 4: Cross-entropies (in bit; lower is better) of the experiments presented in Section 5, for 5-grams, MaxEnt models, and RNN-LMs, as discussed in the respective sub-sections, with the standard model, syntax-enhanced model, and gain (or loss) between the two. As can be seen, the baseline standard models are outperformed for all model types.

models from v and n leads to a large improvement, in particular when using more training data for each individual model (**1b**) rather than interpolating from v and n models (**1a**). Interpolating between the standard and plain syntax-splitting models (**2a**) leads to good results (which peaked at an interpolation weight of .367 for the splitting model). Finally, interpolation of **1b** with the standard model (weight .675) outperforms the standard model by 0.083 bit (a perplexity improvement of 6 % from 76.3 down to 72.0).

5.2 Maximum Entropy Models

We extend the faster-rnnlm MaxEnt model with parser predictions. In the model, the score for a word w_i is computed by summing over all N-gram scores: $s(w_i) = \sum_{j=0}^{n} f(w_i \dots w_{i-j})$. During decoding, the softmax over all possible words w_i yields a probability distribution. f hashes the N-gram (as well as lower order N-grams) and then looks up their scores in a fixed-size table. We extend s by adding vn-annotated (N…1)-grams to the summation: $s(w_i) = \sum_{j=0}^{n} f(w_i \dots w_{i-j}) + f(p_{i-1} w_i \dots w_{i-j})$ We use a hash table with 400M elements, the maximum for our GPGPU card.

Results are presented in Table 4, showing averages of several runs (to account for random initialization). We find that the vn-enhanced models perform significantly better than the standard ones.[7] Performance did not improve with N>3 with our setup, likely due to an increase in hash collisions, as our method puts almost five times as many items into the table. We tested 4-grams on a larger GPGPU card allowing a hash table with 1,600M elements and see that both overall performance and syntax gain improve as expected.

5.3 RNN-based Models

We explore how to integrate parser predictions into a RNN-LM in two different ways as depicted in Figure 5. Either, we add vn as two additional dimensions to the input layer, with positive weight if the corresponding virtual node was predicted and with negative weight otherwise. For example, $\bar{v}\bar{n}$ translates to $(-0.5, 0.5)$. Alternatively, we add vn to the output layer, which is used for noise contrastive estimation (NCE) because intuitively, parser predictions do not need to be fed forward to the next prediction.

For all experiments, we use faster-rnnlm with noise contrastive estimation and a sigmoid layer of size 100 (we ignore MaxEnt here and test RNN in isolation). To account for random initialization (and to allow the estimation of significance), each experiment condition is run 6 times. For both versions, we compared results with a corresponding standard RNN model as reported in Table 4. Both methods significantly[7] outperform the baseline model, and the addition of vn to the NCE layer is slightly better. We conclude that explicitly encoded syntactic long-range information even helps for RNN-LMs which in principle are able to capture longer-range dependencies, at least implicitly.

[7] Wilcoxon rank sum test based on the multiple runs of each experiment condition, $p < .01$.

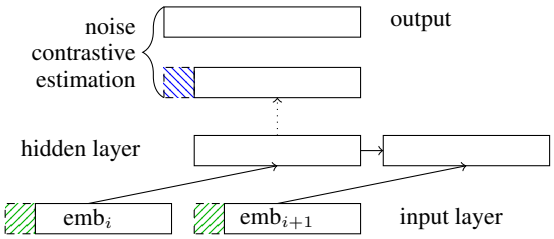


Figure 5: Choice for RNN-LM enhancement, hatched. Either as input (bottom,) or in NCE (top,).

6 Summary and Discussion

We have explored the integration of syntactic structure for online decodable language models, which we base on predictive parsing. This approach incurs a far lower computational cost than previous work on integrating parsing into language models as we parse just once for the history and use the prediction to generate the full probability distribution over possible next words. Our model is built with loose coupling in mind: the parser model is tuned for high syntax parsing accuracy and in no way for language modeling. It can hence be used for other purposes (such as incremental understanding) in an integrated system as well. If a predictive parser is already used in such a system, our proposed approach yields a perplexity improvement for free by feeding back parser predictions to the language model.

Predictions encode information about the next words that are not present in N-grams, i. e. they are able to transport information over long distances in a dense form. Explicit syntax predictions also help RNN-LMs which at least implicitly already model longer-range dependencies. We find that both verb and noun predictions are valuable for language modeling. Combined, they provide a gain of ~0.3 bit for 4- and 5-grams over the split baseline (cmp. Section 4.2). Despite the additional data sparsity introduced by the vn predictions, using count-merging and interpolation yields a perplexity improvement of 6 % over standard N-grams. We also integrated predictions into MaxEnt and RNN-LMs and find consistent and modest improvements for both.

We have used a limited vocabulary and discarded singleton N-grams for computational efficiency. We do not believe that our results will be fundamentally different at larger vocabularies or with unpruned models, but we plan to improve our setup to assess the effect nonetheless in the future. We also plan to integrate our models with online speech recognition to asses whether the observed perplexity reduction carries over to WER. Additional information extracted from the dependency tree may yield further improvements, especially with RNN models, which are robust against data sparsity. Finally, improvements to the parser that result in better predictions should transfer to language modeling as well.

We plan to extend our work beyond English data, as predictive parsing is language independent. However, Beuck and Menzel (2013) find a larger set of virtual nodes to be optimal for German for which we want to assess the merit to our method.

Acknowledgements We would like to thank Wolfgang Menzel for helpful discussion and comments, and the anonymous reviewers for valuable suggestions and remarks. This work is supported by a Daimler and Benz Foundation PostDoc Grant to the second author.

References

Niels Beuck and Wolfgang Menzel. 2013. Structural prediction in incremental dependency parsing. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 7816 of *Lecture Notes in Computer Science*, pages 245–257. Springer, Berlin.

Niels Beuck, Arne Köhn, and Wolfgang Menzel. 2013. Predictive incremental parsing and its evaluation. In Kim Gerdes, Eva Hajičová, and Leo Wanner, editors, *Computational Dependency Theory*, volume 258 of *Frontiers in Artificial Intelligence and Applications*, pages 186 – 206. IOS press.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 124–131, Toulouse, France, July. ACL.

Ciprian Chelba and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th Int. Conf. on Computational Linguistics*, volume 1, pages 225–231, Montréal, Canada, August. ACL.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

Stanley F Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, USA, June. ACL.

Michael Collins, Brian Roark, and Murat Saraclar. 2005. Discriminative syntactic language modeling for speech recognition. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 507–514, Ann Arbor, USA, June. ACL.

Denis Filimonov and Mary Harper. 2009. A joint language model with fine-grain syntactic tags. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP '09): Volume 3*, pages 1114–1123, Singapore, August. ACL.

Daniel Jurafsky, Chuck Wooters, Jonathan Segal, Andreas Stolcke, Eric Fosler, Gary Tajchaman, and Nelson Morgan. 1995. Using a stochastic context-free grammar as a language model for speech recognition. In *Acoustics, Speech, and Signal Processing. ICASSP-95., International Conference on*, volume 1, pages 189–192, Detroit, USA, May. IEEE.

Arne Köhn and Wolfgang Menzel. 2014. Incremental predictive parsing with TurboParser. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2: Short Papers, pages 803–808, Baltimore, USA, June. ACL.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 114–119, Plainsboro, USA, March. ACL.

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 617–622, Sofia, Bulgaria, August. ACL.

Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. 2011. Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 196–201, Waikoloa, USA, December. IEEE.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Ronald Rosenfeld. 1994. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, USA.

Lane Schwartz, Chris Callison-Burch, William Schuler, and Stephen Wu. 2011. Incremental syntactic language models for phrase-based translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 620–631, Portland, USA, June. ACL.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. SRILM at sixteen: Update and outlook. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, Waikoloa, USA, December. IEEE.

Patrick Sturt and Vincenzo Lombardo. 2005. Processing coordinated structures: Incrementality and connectedness. *Cognitive Science*, 29(2):291–305.

Ming Tan, Wenli Zhou, Lei Zheng, and Shaojun Wang. 2012. A scalable distributed syntactic, semantic, and lexical language model. *Computational Linguistics*, 38(3):631–671.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016. Top-down tree long short-term memory networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 310–320, San Diego, USA, June. ACL.

Geoffrey Zweig and Christopher J.C. Burges. 2011. The Microsoft Research sentence completion challenge. Technical Report MSR-TR-2011-129, Microsoft Research, Redmond, USA, December.

A Neural Attention Model for Disfluency Detection

Shaolei Wang, Wanxiang Che, and **Ting Liu**
Center for Social Computing and Information Retrieval, Harbin Institute of Technology, China
{slwang, car, tliu}@ir.hit.edu.cn

Abstract

In this paper, we study the problem of disfluency detection using the encoder-decoder framework. We treat disfluency detection as a sequence-to-sequence problem and propose a neural attention-based model which can efficiently model the long-range dependencies between words and make the resulting sentence more likely to be grammatically correct. Our model firstly encodes the source sentence with a bidirectional Long Short-Term Memory (BI-LSTM) and then uses the neural attention as a pointer to select an ordered subsequence of the input as the output. Experiments show that our model achieves the state-of-the-art f-score of 86.7% on the commonly used English Switchboard test set. We also evaluate the performance of our model on the in-house annotated Chinese data and achieve a significantly higher f-score compared to the baseline of CRF-based approach.

1 Introduction

Disfluency detection is the task of detecting the infelicities in spoken language transcripts. The task is important for natural language understanding, since most downstream NLU systems are built on the fluent utterances. Disfluency of a sentence can be categorized into five classes (Wu et al., 2015): uncompleted words, filled pauses (e.g. "uh", "um"), editing terms (e.g. "you know"), discourse markers (e.g. "i mean") and repairs that are discarded, or corrected by its following words. Typically, as shown in Figure 1, a repair type disfluency consists of a filled pause ("um"), a reparandum ("Boston") and an Interregnum("I mean") followed by its repair ("Denver"). The goal of the repair type disfluency detection is to detect the reparandum. The former four classes of disfluencies are easy to detect as they often consist of fixed phrases (e.g. "uh", "you know"). However, the repair type disfluency (see Table 1 for more examples) is more difficult to detect, because reparandums are in arbitrary form. Most of the previous disfluency detection works focus on detecting the repair type disfluencies.

A flight to $\underbrace{\text{um}}_{\text{FP}}$ $\underbrace{\text{Boston}}_{\text{RM}}$ $\underbrace{\text{I mean}}_{\text{IM}}$ $\underbrace{\text{Denver}}_{\text{RP}}$ Tuesday

Figure 1: A sentence with disfluencies annotated in the style of (Shriberg, 1994) and the Switchboard corpus. FP=Filled Pause, RM=Reparandum, IM=Interregnum, RP=Repair. We follow previous works in evaluating the system on the accuracy with which it identifies speech-repairs, marked *reparandum* above.

Modeling long-range dependencies between repair phrases is one of the core problems for disfluency detection. Previous sequence tagging methods (Ferguson et al., 2015; Georgila, 2009; Qian and Liu, 2013) require carefully designed features to capture the information of long distance, but usually suffer from the sparsity problem. Another line of syntax-based disfluency detection works (Honnibal and Johnson, 2014; Wu et al., 2015) try to model the repair phrases on a syntax tree by compressing the unrelated

* Corresponding author: Wanxiang Che

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 278–287, Osaka, Japan, December 11-17 2016.

they/E had/E they/E used/E to/E have/E well they do have television monitors stationed throughout our buildings.
and the/E other/E one/E is/E her husband is in the navy.
they/E in/E fact/E they/E just/E it was just a big thing recently.

Table 1: Sentences in which the former four classes of disfluencies have been removed. "/E" means that the word belong to a reparandum and should be deleted.

phrases and allowing the repair phrases to interact with each other. However, data that both have syntax trees and disfluency annotations are scarce. The performance of syntax parsing models (about 92% on English) also hinders disfluency detection's performance. There are also works that try to use recurrent neural network (RNN), which can capture dependencies at any length in theory, on disfluency detection problem (Hough and Schlangen, 2015). The RNN method treats sequence tagging as classification on each input token and doesn't model the transition between tags which is important in recognizing the repair phrases of multi-words. Another core problem for disfluency detection is to keep the generated sentences grammatical. However, the sequence tagging methods and RNN method have no power of modeling the linguistic structural integrity. The output of syntax-based methods is a syntax tree and can keep the output sentence grammatical in theory, but limited by the performance of syntax parsing models.

In this paper, we address the above challenges by seeing disfluency detection as a sequence-to-sequence problem and presenting a neural attention-based model to learn the conditional probability of the output which is an ordered subsequence of the input, inspired by the work of (Rush et al., 2015; Vinyals et al., 2015). Our model repurposes the attention mechanism (Bahdanau et al., 2014) to create pointers to the input elements. More specifically, we first encode the input sentence with a bidirectional Long Short-Term Memory (BI-LSTM) and then use the neural attention as a pointer to select a member of the input sequence as the output. In the decoding process, we use the attention mechanism only over the candidate words within a window and select the word with the maximum attention weight in each step. Once a word is selected, all the candidate words before it will be labeled as reparandum and be deleted.

While our neural attention-based model is structurally simple, it is very suitable for disfluency detection for that (1) taking into account the language model and a constrained vocabulary (words appear in the input sentence) during generation, which makes the resulting sentence more likely to be grammatically correct, and (2) utilizing the global representation of the input for generating, and selecting the word by joint decision with our neural attention mechanism, which can effectively model the long-range dependencies.

Experiments show that our model achieves the state-of-the-art f-score of 86.7% on the commonly used English Switchboard test set. We also evaluate the performance of our model on Chinese annotated data. As there is no public disfluency data in Chinese, we annotate 200k Chinese sentences manually for training and testing. We achieve a 61.4% f-score with more than 7 points gained compared to the CRF-based baseline, showing that our models are robust.

Our original major contributions in this paper include:

- We firstly study the problem of disfluency detection using the encoder-decoder framework.
- We propose a novel neural attention-based model for disfluency detection, and demonstrate its effectiveness on both English Switchboard corpus and in-house annotated Chinese corpus.

2 Background: Neural Models for Sequence-to-sequence Learning

Sequence-to-sequence learning can be expressed in a probabilistic view as maximizing the likelihood of observing the output (target) sequence given an input (source) sequence.

2.1 RNN Encoder-Decoder

RNN-based Encoder-Decoder is successfully applied to real world sequence to sequence tasks, first by (Sutskever et al., 2014; Cho et al., 2014). In the Encoder-Decoder framework, the source sequence $X = [x_1, ..., x_T]$ is converted into a fixed length vector c by the encoder RNN, i.e.

$$h_t = f(x_t, h_{t-1}); \quad c = q(\{h_1, ..., h_T\}) \tag{1}$$

where $h_t \in R^n$ is a hidden state at time t, and c is a vector generated from the sequence of the hidden states. f and q are some nonlinear functions. (Sutskever et al., 2014) used an LSTM as f and $q(\{h_1, ..., h_T\}) = h_T$, for instance.

The decoder is often trained to predict the next word y_t given the context vector c and all the previously predicted words $\{y_1, ..., y_{t-1}\}$. In other words, the decoder defines a probability over the translation y by decomposing the joint probability into the ordered conditionals:

$$p(y) = \prod_{t=1}^{T} p(y_t \mid \{y_1, ..., y_{t-1}\}, c), \tag{2}$$

where $y = \{y_1, ..., y_{t-1}\}$. With an RNN, each conditional probability is modeled as

$$p(y_t \mid \{y_1, ..., y_{t-1}\}, c) = g(y_{t-1}, s_t, c), \tag{3}$$

where g is a nonlinear, potentially multi-layered, function that outputs the probability of y_t, and s_t is the hidden state of the RNN. Note that y_t belongs to a fixed output vocabulary dictionary.

2.2 The Attention Mechanism

The attention mechanism was first introduced to sequence to sequence (Bahdanau et al., 2014) to release the burden of summarizing the entire source into a fixed-length vector as context. Instead, the attention uses a dynamically changing context c_t in the decoding process. c_t at each output time t is computed as follows:

$$\begin{aligned} u_{tj} &= v^T tanh(W_1 h_j + W_2 s_{t-1}) \quad j \in (1, ..., n) \\ a_{tj} &= \frac{exp(u_{tj})}{\sum_{k=1}^{T} exp(u_{tk})} \quad j \in (1, ..., n) \\ c_t &= \sum_{j=1}^{n} a_{tj} h_j \end{aligned} \tag{4}$$

where s_{t-1} is the decoder hidden states for time $t-1$. v, W_1, and W_2 are learnable parameters of the model. Lastly, c_t and s_{t-1} are concatenated as the hidden states which will be used for making predictions and fed to the next time step in the decoder RNN. We call u_{tj} a relevance score, or an alignment weight of the j-th input. Note that the vanilla attention mechanism compute the relevance among all the input each step.

3 Model Description

Disfluency detection requires that the output should be an ordered subsequence of the input. The vanilla sequence-to-sequence approach(Sutskever et al., 2014) requires the size of the output dictionary to be fixed a priori, which means that it can only generate one word of the output dictionary at any step. Because of this constraint, we cannot directly apply the vanilla sequence-to-sequence framework to disfluency detection, since it (1) may generate a word out of the input, (2) can't generate the word out of the fixed output dictionary but in the input when applying the model on the test set, and (3) has no power of modeling the order of the generated words. To address the constraint, we propose a new attention-based model as illustrated in Figure 2. Our model is still an encoder-decoder framework in a slightly generalized sense.

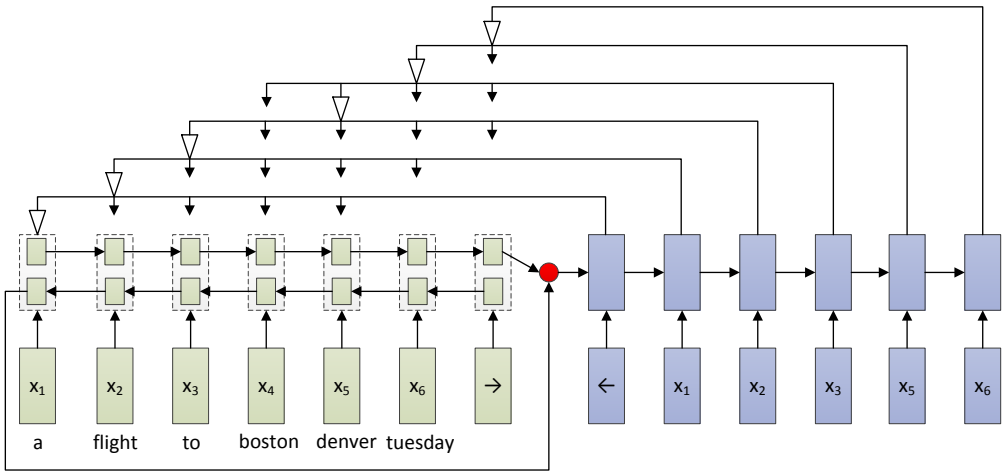


Figure 2: our attention-based network: An encoding RNN converts the input sequence to a vecter that is fed to the generating network. At each step, the generating network produces a vector that modulates a content-based attention mechanism over the words in a window. The word with maximum attention value is selected.

3.1 Input representation

To represent each input token, we use four vectors: a learned word embedding w; a fixed word embedding $\widetilde{w}$; a learned POS-tag embedding p; a hand-crafted feature representation d. The four vectors are concatenated together, transformed by a matrix V and fed to a rectified layer to learn feature combination, as

$$x = max\{0, V[\widetilde{w}; w; p; d] + b\} \tag{5}$$

where $[\widetilde{w}; w; p; d]$ means the concatenation.

We extract two kinds of hand-crafted discrete features (as shown in Table 2) for each token in a sentence and incorporate them into our neural networks by translating them into the 0-1 vector d. The dimension of d is 78, which equals to the number of discrete features. For a token x_t, d_i fires if x_t matches the i-th pattern of the feature templates. The duplicate features care whether x_t has a duplicated word/POS-tag in certain distance. The similarity features care whether the surface string of x_t resembles its surrounding words.

duplicate features
$Duplicate(i, w_{i+k}), -15 \leq k \leq +15$ and $k \neq 0$: if w_i equals w_{i+k}, the value is 1, others 0
$Duplicate(p_i, p_{i+k}), -15 \leq k \leq +15$ and $k \neq 0$: if p_i equals p_{i+k}, the value is 1, others 0
$Duplicate(w_i w_{i+1}, w_{i+k} w_{i+k+1}), -4 \leq k \leq +4$ and $k \neq 0$: if $w_i w_{i+1}$ equals $w_{i+k} w_{i+k+1}$, the value is 1, others 0
$Duplicate(p_i p_{i+1}, p_{i+k} p_{i+k+1}), -4 \leq k \leq +4$ and $k \neq 0$: if $p_i p_{i+1}$ equals $p_{i+k} p_{i+k+1}$, the value is 1, others 0
similarity features
$fuzzyMatch(w_i, w_{i+k}), k \in \{-1, +1\}$: $similarity = num\_same\_letters/(len(w_i) + len(w_{i+k}))$. if $similarity > 0.8$, the value is 1, others 0

Table 2: Discrete features used in our neural attention-based networks. p is the POS tag. w is the word. Duplicate indicates if the two units are same. fuzzyMatch indicates the similarity of two words.

3.2 Encoder

We use bidirectional LSTM-based RNN which consists of forward and backward RNNs to transform the source sequence into a series of hidden states, with each hidden state h_t corresponding to word x_t. The

Algorithm 1 : Learning algorithm of our neural attention-based model for disfluency detection

Function: Training instance $(x_i, y_i)_{i=1}^{N}$ {y_i is the positions of the words selected from x_i}

for $e = 1, T$ **do**
  for $i = 1, N$ **do**
    $MAX\_DISF\_LEN \leftarrow 10$, $len \leftarrow length(x_i)$
    $(h_1, ..., h_{len}) \leftarrow encode(x_i)$
    $Decode\text{-}LSTM \leftarrow initial\_lstm(h_1, h_{len})$
    $log\_probs \leftarrow \{\}$
    $p \leftarrow 0, t \leftarrow 0$
    while $p < len$ **do**
      $Attention\_Start \leftarrow p, Attention\_End \leftarrow min\{p + MAX\_DISF\_LEN, len\}$
      $attention \leftarrow []$
      for k in $(Attention\_Start, Attention\_End)$ **do**
        compute the attention weight u_{tk} using Equation 6
        APPEND(attention, u_{tk})
      end for
      $position\_of\_correct = y_i^t$, $position\_of\_predict = p + attention.index(max\{attention\}))$
      APPEND(log_probs, $neg\_log\_softmax(attention, position\_of\_correct - p)$)
      $t \leftarrow t + 1, p \leftarrow position\_of\_correct + 1$
      $Decode\text{-}LSTM$.input($x_i^{position\_of\_correct}$)
    end while
    Stochastic gradient descent update on the loss $\lambda = sum(log\_probs)$
  end for
end for

forward RNN reads the input sequence $x = (x_1, ..., x_T)$ in a forward direction, resulting in a sequence of hidden states $(\overrightarrow{h_1}, ..., \overrightarrow{h_T})$. The backward RNN reads x in an opposite direction and outputs $(\overleftarrow{h_1}, ..., \overleftarrow{h_T})$. We concatenate a pair of the hidden states at each time step to build a sequence of annotation vectors $(h_1, ..., h_T)$ where $h_j = [\overrightarrow{h_j}; \overleftarrow{h_j}]$. Each annotation vector h_j encodes the information about the j-th word with respect to all the other surrounding words in the sentence.

3.3 Attention-based Decoder

We address the constraint of the fixed output dictionary in vanilla sequence-to-sequence approaches by modifying the decoder framework. The mainly changes are (as shown in Algorithm 1) as bellows:

- We select a word from the candidate words $(x_i, ...x_j)$ rather than on a fixed output dictionary in each step. This mechanism keeps all the generated words come from the input sentence and has the ability to generate a word that occurs in the test sentence but not in the train sentence.
- Once a word x_k is selected, all the words $(x_i, ..., x_{k-1})$ will be deleted and the candidate words in the next step will be $(x_{k+1}, ...x_l)$. This mechanism keeps the selected words in the order that they appear in the input sentence.

Now the key question is how to choose the correct word from the candidate words $(x_i, ...x_j)$ in each step t. We achieve it by selecting the word x_k with the highest relevance weight u_{tk}, which is computed by repurposing the attention mechanism of Section 2.2. The attention mechanism of Section 2.2 computes the relevance weight vector u_t on all the input sentence $(x_1, ..., x_T)$ in each step t. This mechanism is unsuited to our task since we only need to consider the relevance weights of the subsequence $(x_i, ...x_j)$ in each step t. Hence, we make a modification of the attention mechanism in Equation 4 and model $p(y_t \mid \{y_1, ..., y_{t-1}\}, c)$ in Equation 3 as follows:

$$\begin{aligned} u_{tk} &= v^T tanh(W_1 h_k + W_2 s_{t-1} + W_3 d_{k-i}) \quad k \in (i, ..., j) \\ p(y_t \mid \{y_1, ..., y_{t-1}\}, c) &= softmax(u_t) \end{aligned} \tag{6}$$

where d_{k-i} is the embedding of the distance between previous selected word x_{i-1} and the word x_k. The distance information is very important for our model, since the latter word is more likely to be selected when two words have similar context embeddings in disfluency detection. Once x_k is selected, we will use it to update the state of the decoder RNN and select another word from the words ranging from

$(x_{k+1}, ..., x_l)$ (as shown in Figure 2). To learn the parameters of our neural attention-based model, we minimize the negative log-probability of the output sequence over the input data $\{(x_i, y_i)\}_{n=1}^{N}$ during training:

$$-\sum_{i=1}^{N} log(p(y_i|x_i)) = -\sum_{i=1}^{N} log(\prod_{t=1}^{T} softmax(u_t)) \quad (7)$$

The detailed learning algorithm is shown in Algorithm 1.

4 Network training

4.1 Parameters

Pretrained word embedding. There are lots of methods for creating word embeddings. As (Dyer et al., 2015) does, we use a variant of the skip n-gram model introduced by (Ling et al., 2015), named "structured skip n-gram", where a different set of parameters are used to predict each context word depending on its position relative to the target word. The hyperparameters of the model are the same as in the skip n-gram model defined in word2vec (Mikolov et al., 2013). We set the window size to 5, and use a negative sampling rate to 10. The AFP portion of English Gigaword corpus (version 5) is used as the training corpus.

Hyper-Parameters. The LSTMs of both encoder and decoder has two hidden layers and their dimensions are set to 100. Pretrained word embeddings have 100 dimensions and the learned word embeddings have also 100 dimensions. Pos-tag embeddings have 12 dimensions. The dimension of distance d in Equation 6 is set to 8.

Parameter initialization. The learned parameters in the neural networks are randomly initialized with uniform samples from $[-\sqrt{\frac{6}{r+c}}, +\sqrt{\frac{6}{r+c}}]$, where r and c are the number of rows and columns in the parameter structure.

4.2 Optimization Algorithm

Parameter optimization. Parameter optimization is performed with stochastic gradient descent (SGD) with an initial learning rate of $\eta_0 = 0.1$ and a gradient clipping of 5.0. The learning rate is updated on each epoch of training as $\eta_t = \eta_0/(1 + \rho t)$, in which t is the number of epoch completed and the decay rate $\rho = 0.05$.

Early Stopping. We use early stopping (Giles, 2001) based on performance on dev sets. The best parameters appear at around 12 epochs, according to our experiments.

Dropout Training. To reduce overfitting, we apply the dropout method (Srivastava et al., 2014) to regularize our model. We apply dropout not only on input and output vectors of LSTM, but also between different hidden layers of LSTM. We observe a significant improvement on model performances after using dropout.

Unknown Word Handling. As described in section 3, the input layer contains a learned vector representation for the word w and a corresponding fixed pretrained vector representation $\widetilde{w}$. We randomly replace the singleton word in the training data with the UNK token during training, but keep corresponding $\widetilde{w}$ unchanged. This technique can deal with out-of-vocabulary words.

5 Experiments

5.1 Settings

Dataset. Firstly, we conduct our experiments on the English Switchboard corpus. Following the experiment settings in (Charniak and Johnson, 2001; Honnibal and Johnson, 2014; Wu et al., 2015), we use directory 2 and 3 in PARSED/MRG/SWBD as our training set and split directory 4 into test set, development set and others. We extract the repair disfluencies according to the EDITED label in the Switchboard corpus. Following (Honnibal and Johnson, 2014), we lower-case the text and remove all punctuations and partial words[1]. We also discard all the 'um' and 'uh' tokens and merge 'you know'

[1]words are recognized as partial words if they are tagged as 'XX' or end with '-'

Method	Dev			Test		
	P	R	F1	P	R	F1
CRF	93.8%	77.7%	85.0%	92.0%	74.5%	82.3%
Attention-based method	93.0%	81.6%	86.9%	91.6%	82.3%	86.7%

Table 3: Experiment results on the development and test data of English Switchboard data.

Method	**P**	**R**	**F1**
Attention-based	91.6%	82.3%	**86.7**%
M^3N (Qian and Liu, 2013)	-	-	84.1%
Joint Parser (Honnibal and Johnson, 2014)	-	-	84.1%
semi-CRF (Ferguson et al., 2015)	90.1%	80.0%	84.8%
UBT (Wu et al., 2015)	90.3%	80.5%	85.1%

Table 4: Comparison of our neural attention-based model with the previous state-of-the-art methods on the test set of English Switchboard data.

and 'i mean' into single token. Automatic POS-tags generated from pocket_crf (Qian and Liu, 2013) are used as POS-tag in our experiments.

No public Chinese corpus is available now. For our Chinese experiments, we collect about 200k spoken sentences from minutes of meetings and annotate them with only disfluency annotations according to the guideline proposed by (Meteer et al., 1995). We respectively select about 20k sentences for development and testing. The rest are used for training. We use the word segmentation and POS-tag tools provided by the Language Technology Platform (Che et al., 2010) for preprocessing the original data in our experiments.

Metric. Following previous works (Ferguson et al., 2015; Wu et al., 2015), token-based precision (P), recall (R), and f-score (F1) are used as the evaluation metrics.

5.2 Performance of disfluency detection on English Swtichboard corpus

We build a baseline system using the Conditional Random Field (CRF) model. The hand-crafted discrete features of our CRF refer to those in (Ferguson et al., 2015). Table 3 shows the result of our model on both the development and test set.

We compare our neural attention-based model to four previous top performance systems. Our model outperforms state-of-the-art work and achieves a 86.7% f-score as shown in Table 4. Our model achieves 1.6 point improvements over UBT (Wu et al., 2015), which is the best syntax-based method for disfluency detection. The best performance by linear statistical sequence labeling methods is the semi-CRF method (Ferguson et al., 2015), achieving a 84.8% F1 score without leveraging prosodic features. Our model beats the semi-CRF model, obtaining 1.9 point improvements. Note that our method performs better than previous methods not only on precision , but also on recall. The comparison shows that our model is a good solution to disfluency detection.

5.3 Ablation Test

As described in section 3.1, we extract two kinds of hand-crafted discrete features for each token in a sentence. The duplicate features are extracted, because the reparandum(RM) and the following repair(RP) often share some identical words/POS-tags. In some cases, the word in a reparandum is the singular or plural of the word in the following repair, hence we extract the similarity features.

To test the individual effectiveness of duplicate features and similarity features, we conduct feature ablation experiments for our neural attention-based model. Table 5 shows the result. We can see that both the two kinds of features contribute to the performance improvements of disfluency detection and we can achieve a higher performance by integrating all of them into our model. This indicates that duplicate features and similarity features are both important to the neural model and they provide different kinds of information for disfluency detection.

Method	Dev			Test		
	P	R	F1	P	R	F1
Attention-based	93.0%	81.6%	86.9%	91.6%	82.3%	**86.7**%
- Duplicate	93.6%	78.0%	85.1%	90.8%	76.3%	82.9%
- Similarity	93.3%	82.0 %	87.3%	92.3%	80.9%	86.2%
- Duplicate - Similarity	93.2%	76.8%	84.2%	89.4%	76.2%	82.3%

Table 5: Results of feature ablation experiments on English Switchboard data.

Method	Dev			Test		
	P	R	F1	P	R	F1
CRF	76.5%	42.0%	54.2%	75.9%	41.6%	53.8%
Attention-based method	83.7%	50.6%	63.1%	82.4%	48.9%	61.4%

Table 6: Disfluency detection performance on Chinese annotated data.

5.4 Performance of disfluency detection on Chinese annotated corpus

In addition to English experiments, we also apply our method on Chinese annotated data. As there is no standard Chinese corpus, no Chinese experimental results are reported in (Honnibal and Johnson, 2014) and (Qian and Liu, 2013). We only use the CRF-based labeling model as our baselines. Table 6 shows the results of Chinese disfluency detection. Our models outperform the CRF model by more than 7 points on f-score which shows that our method is more effective.

6 Related Work

Most related works on disfluency detection are aimed at detecting repair type of disfluencies. (Johnson and Charniak, 2004) proposed a TAG-based noisy channel model for disfluency detection. The TAG model was used to find rough copies. Following the work of (Johnson and Charniak, 2004), (Zwarts and Johnson, 2011) extended the TAG model using minimal expected f-loss oriented n-best reranking with additional corpus for language model training. (Qian and Liu, 2013) proposed a muiti-step learning method using weighted max-margin markov network (M^3N). They showed that M^3N model outperformed many other labeling models such as CRF model. (Ferguson et al., 2015) used the Semi-Markov CRF model for disfluency detection and achieved high f-score by integrating prosodic features.

Many syntax-based approaches have been proposed which jointly perform dependency parsing and disfluency detection. (Lease and Johnson, 2006) involved disfluency detection in a PCFG parser to parse the input along with detecting disfluencies. (Rasooli and Tetreault, 2013) designed a joint model for both disfluency detection and dependency parsing. (Honnibal and Johnson, 2014) presented a new joint model by extending the original transition actions with a new "Edit" transition. This model achieved good performance on both disfluency detection and parsing. (Wu et al., 2015) proposed a right-to-left transition-based joint method and achieved the state-of-the-art performance compared with previous syntax-based approaches.

RNN had been used to disfluency detection. (Hough and Schlangen, 2015) explored incremental detection, with an objective that combines detection performance with minimal latency. This approach achieved worse performance compared with other works for the latency constraints. (Cho et al., 2013) used word embeddings learned by an RNN as features in a CRF classifiers.

7 Conclusion and Future Work

In this paper, we firstly regard the disfluency detection as a sequence-to-sequence problem and propose a neural attention-based model. Our model break the constraint of vanilla sequence-to-sequence method that the size of the output dictionary should be fixed a priori by modifying the structure of decoder. Experimential results show that our method achieves the best reported performance on the commonly used English Switchboard corpus and a better performance than CRF model on in-house Chinese annotated data.

In the future, we will try to incorporate character-based representations into the encoder model. We would also like to jointly model disfluency detection and automatic punctuation using some neural network.

Acknowledgments

We thank the anonymous reviewers for their valuable suggestions. This work was supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grant 61370164 and 61632011.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *In ICLR 2015, arXiv preprint arXiv:1409.0473*.

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–9. Association for Computational Linguistics.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *Coling 2010: Demonstrations*, pages 13–16, Beijing, China, August. Coling 2010 Organizing Committee.

Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2013. Crf-based disfluency detection using semantic features for german to english spoken language translation. *IWSLT, Heidelberg, Germany*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343. Association for Computational Linguistics, July.

James Ferguson, Greg Durrett, and Dan Klein. 2015. Disfluency detection with a semi-markov model and prosodic features. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 257–262. Association for Computational Linguistics.

Kallirroi Georgila. 2009. Using integer linear programming for detecting speech disfluencies. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 109–112. Association for Computational Linguistics.

Rich Caruana Steve Lawrence Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, volume 13, page 402. MIT Press.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association for Computational Linguistics*, 2:131–142.

Julian Hough and David Schlangen. 2015. Recurrent neural networks for incremental disfluency detection. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Mark Johnson and Eugene Charniak. 2004. A tag-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 33. Association for Computational Linguistics.

Matthew Lease and Mark Johnson. 2006. Early deletion of fillers in processing conversational speech. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 73–76. Association for Computational Linguistics.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304.

Marie W Meteer, Ann A Taylor, Robert MacIntyre, and Rukmini Iyer. 1995. *Dysfluency annotation stylebook for the switchboard corpus*. University of Pennsylvania.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In *HLT-NAACL*, pages 820–825.

Mohammad Sadegh Rasooli and Joel R Tetreault. 2013. Joint parsing and disfluency detection in linear time. In *EMNLP*, pages 124–129.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal, September. Association for Computational Linguistics.

Elizabeth Ellen Shriberg. 1994. *Preliminaries to a theory of speech disfluencies*. Ph.D. thesis, Citeseer.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Shuangzhi Wu, Dongdong Zhang, Ming Zhou, and Tiejun Zhao. 2015. Efficient disfluency detection with transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 495–503. Association for Computational Linguistics.

Simon Zwarts and Mark Johnson. 2011. The impact of language models and loss functions on repair disfluency detection. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 703–711. Association for Computational Linguistics.

Detecting Sentence Boundaries in Sanskrit Texts

Oliver Hellwig
Düsseldorf University, SFB 991
ohellwig@phil-fak.uni-duesseldorf.de

Abstract

The paper applies a deep recurrent neural network to the task of sentence boundary detection in Sanskrit, an important, yet underresourced ancient Indian language. The deep learning approach improves the F scores set by a metrical baseline and by a Conditional Random Field classifier by more than 10%.

1 Introduction

Most NLP tasks that deal with written texts take it for granted that sentences are separated reliably by punctuation marks, although punctuation has been added quite late to many writing systems. The large corpora in Old- and Middle-Indian languages, which belong to the central sources for understanding the history of South Asia, generally lack dedicated punctuation marks. This paper applies deep recurrent neural networks (RNN) to a combination of morphological and lexical features for detecting sentence boundaries (SB) in Sanskrit, the oldest and most important of these Indian languages.

Traditional editions of Sanskrit texts use a *scriptio continua*, which lacks several orthographic elements that structure texts in modern Western languages. Single words are frequently not separated by blank spaces due to missing orthographic regulation, or because the words are merged through the euphonic rules called *sandhi* ("connection").[1] Moreover, Sanskrit texts don't have a consistent and unambiguous system for marking SBs. Editors and scribes insert so called (double) *daṇḍa*s ("sticks", indicated by | and || in this paper) to mark the end of metrical structures. The position of these *daṇḍa*s can be derived directly from the prosodic structure of a text, and *daṇḍa*s always occur at the end of text lines, which coincide with half-verses in most printed editions. While single *daṇḍa*s mark the end of a half-verse, double *daṇḍa*s should, at least theoretically, indicate, where a stanza in the given metre is completed. Double *daṇḍa*s typically occur after every second line or half-verse of a metrical text, because the stanzas are finished at these points. In this function, they are meant to improve the readability of a text. As many sentences terminate at the end of a half-verse or of a stanza, *daṇḍa*s provide a good baseline for punctuation prediction (refer to Table 3). Many editors, however, also insert double *daṇḍa*s after a single or after three metrical lines, when they feel that a sentence is completed at these positions.[2] In this way, the purely metrical motivation of double *daṇḍa*s is mingled with the new function of a punctuation mark – leaving aside the fact that the philologically interesting inner-line SBs cannot be marked in the *daṇḍa* system.

Linguistic peculiarities of Sanskrit complicate the task. English, for example, encodes a large amount of its syntax through a strongly regulated word order, and structures its sentences by subordinating conjuctions. While these data provide a lot of the information necessary for restoring punctuation, Sanskrit has a rather loose word order with a tendency to subject-object-verb constructions, it uses conjunctions quite sparingly, and their position provides only weak indications for the presence of SBs. As the Indian

[1] The two words *parvatasya agre*, for example, are merged into one string *parvatasyāgre* by the rule *a+a=ā*; refer to Kielhorn (1888, 6ff.) for an overview. *Sandhi* is one of the main problems for Sanskrit NLP.

[2] Refer to Hopkins (1901, 194): "The number of verses in a (...) stanza may be decreased or increased by one or two (...). Sometimes, however, where one or three hemistichs make a stanza, it is merely a matter of editing."

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 288–297, Osaka, Japan, December 11-17 2016.

grammatical tradition has emphasized (Section 2), determining the boundaries of a sentence is equivalent to grasping its full semantic meaning.

The need for reliable punctuation on sentence level is beyond question. Access to full sentences is central for NLP tasks such as dependency parsing or role labeling. In addition, detecting SBs is also important from a philological perspective. The metrical texts considered in this paper belong to a tradition of (pseudo-)oral poetry that still survives in parts of India (Smith, 1987). The constituent structure of sentences (e.g., extensive right branchings) or the presence of enjambements, which are easily detected when SBs are known, provide important evidence for understanding the transition of these epics from an oral to a written state (Sellmer, 2015; Parry, 1930). More generally, Sanskrit provides a challenging application scenario for NLP due to the richness of its phonetics (*sandhi*), morphology, lexicon, and semantics. In spite of its historical importance, it is heavily underressourced from the perspective of NLP, and the size of its corpus prevents a purely manual annotation of linguistic phenomena.[3]

The remainder of the paper is structured as follows. Section 2 sketches how a sentence was defined in the tradition of classical Indian grammar, and summarizes related research from NLP and automatic speech recognition. Section 3 reports results of a test annotation, details the annotation guideline, and describes the data prepared for this study. Section 4 introduces the features and the deep learning model. Section 5 describes the evaluation baselines given by prosodical markers and a CRF model, discusses the performance of the model, and identifies critical areas. Section 6 summarizes the paper.

2 Related Work

Classical Sanskrit was systematically de- and prescribed in the famous grammar Aṣṭādhyāyī of Pāṇini (around 350 BCE, Scharfe (1977)), who used Sanskrit as a metalanguage, and applied methods such as rewrite rules and rule inheritance for minimizing the text length (Kiparsky, 2009). While the Aṣṭādhyāyī deals exhaustively with phonetics and morphology, syntax only plays a subordinate role. Its main syntactic contribution is the *kāraka* theory, which describes the interaction between nominal case suffixes and verbs (Cardona, 1976, 215ff.). The grammatical tradition following Pāṇini provided empirical, verb-centered definitions of sentences (Matilal, 1966, 377ff.). Because many Sanskrit sentences don't overtly express the copula "to be", these definitions gave rise to extended discussions about the underlying grammatical and cognitive structures of sentences such as *puṣpaṃ raktam* (flower:NSG red:NSG), which may mean "a red flower" or the complete sentence "the flower is red" (Deshpande, 1991). Missing copulae introduce a high degree of ambiguity in the SB detection task, as will be seen in Section 5.3. The later philosophical school of Nyāya concentrated on the conditions that make a sentence meaningful and complete for a competent speaker of Sanskrit (Matilal, 1966, 385ff.), and that include the semantic compatibility of the words (*yogyatā*) and their correct grouping (*saṃnidhi*; see Kulkarni et al. (2015)). If an utterance fulfills these conditions, it creates the intended cognition (*śābdabodha*) in the listener. So, the Indian tradition claims that only a competent speaker can determine the boundary of a sentence, but does not provide formal criteria for deciding if a sentence is complete or not.

Related research in NLP mainly deals with punctuation restoration in speech transcripts, and in languages such as Chinese that traditionally don't use punctuation marks for structuring syntactic sequences. Liu et al. (2005) contrast Hidden Markov Models (HMM), Maximum Entropy classifiers and Conditional Random Fields (CRF). They obtain a significant decrease of the SB detection error when processing lexical and automatically induced POS features using a CRF. Baldwin and Joseph (2009) perform simultaneous case and punctuation restoration in English texts. They process automatically annotated lexical, POS, and chunk features with a linear kernel Support Vector Machine. The authors report the highest F score for punctuation restoration, when they iteratively label the training and test sets with the output of the classifier, and retrain with the augmented feature space ("iterative retagging"). Zhao et al. (2012) train CRFs on the task of inserting punctuation in Chinese text, using features from different annotation levels of a Chinese treebank, and observe an increase in the F score, when higher level features such as

[3]There exist no reliable estimations of the real size of Sanskrit literature. The GRETIL website (`http://gretil.sub.uni-goettingen.de/`), which provides digital transcripts of a few percent of all printed Sanskrit texts, may contain around 15 million lexical tokens (estimation of the author; numbers may be significantly higher due to Sandhi). Large parts of the Sanskrit literature are still only transmitted as manuscripts.

POS tags or chunks are combined with lexical information. Tilk and Alumäe (2015) model the restoration of commas and periods in Estonian speech transcripts with a two-stage Long Short-term Memory (LSTM) approach. The first LSTM is trained on a large written corpus with lexical information in 1-hot-encoding as predictors and the associated punctuation as predicted classes. Following Seide et al. (2011, 26), the authors combine the output of the last hidden layer of this text LSTM with duration features from a smaller corpus of punctuated speech transcripts. This combined feature set is fed into a second LSTM that performs the final classification.

Algorithms based on short range models (n-grams, HMMs) or those requiring strict positional information may not be applicable to Sanskrit for several reasons. Sanskrit has a relatively free word order (Gillon and Shaer, 2005; Hock, 2013), and encodes many syntactical relations through its morphology, so that the positional information inherent in an n-gram model may not contribute as strongly as in English or Chinese. In addition, Sanskrit NLP suffers from data sparsity in the lexical domain. The corpus on which the models are trained contains 3,950,000 disambiguated lexical tokens. New data for pretraining a lexical model cannot be generated on the fly, because the phonetic phenomenon of Sandhi introduces a high degree of ambiguity (Hellwig, 2015b), and sufficiently large digitized Sanskrit corpora are missing.

CRFs as used by Liu et al. (2005) and Zhao et al. (2012) are more flexible than HMMs in modeling the feature space involved in SB detection, because their input features can, in principle, come from arbitrarily long ranges around a focus word, and because they are trained to maximize the classification accuracy. RNNs as used by Tilk and Alumäe (2015) are equally able to capture the long-range interactions between morphology, lexicon, and output symbols that can be hypothesized to play an important role in SB detection. Section 5 will compare their efficiency in the present task. The problems of exploding and vanishing gradients (Pascanu et al., 2013) can be handled with Long Short-Term Memory units (LSTM, for the vanishing ones (Hochreiter and Schmidhuber, 1997), combined with a gradient cutoff) or with Hessian free training of the network (Martens and Sutskever, 2011). Stacked LSTMs as used by Sutskever et al. (2014) with bidirectional units (Schuster and Paliwal, 1997) seem to provide a promising approach for labeling SBs in Sanskrit.

3 Data

3.1 Test annotation

The discussion in Section 2 has shown, that the Sanskrit grammatical tradition does not provide a solid basis for developing a practical annotation guideline for SBs. As a consequence, ten sequences of at least two metrical lines that contain complex syntactic phenomena were annotated independently by three external annotators and the author of the paper. Given the small size of the data set, this annotation was not primarily meant to determine the true inter-annotator agreement (IAA), but rather to obtain quantitative support for ambiguous cases in the annotation guideline. The lines were tokenized according to Western editorial standards without resolving Sandhis and compounds.

Assuming that a period can be inserted after each of the 360 tokens, the annotation yielded an IAA of 0.805, using Fleiss' κ (Fleiss, 1971). When only those tokens are considered after which at least one annotator inserted a period, the IAA drops to $\kappa = 0.312$. A detailed analysis shows that almost all unanimous annotations concern periods that coincide with (double) *daṇḍa*s, while there is substantial disagreement about inner-line periods.

3.2 Guideline

Drawing from the results of the initial annotation and from ideas proposed in Matilal (1966), this paper defines a Sanskrit sentence as a sequence of words that contains at least an overtly expressed finite verb (type s_1; minimal sentence length: one word[4]), or two non-verbal elements with an unexpressed copula denoting equivalence or existence[5] (type s_2). s_1 and s_2 can be expanded by (recursive and/or compound) subordinate clauses and matrix sentences. As a direct consequence, sentences on the s_1 or s_2 levels that

[4]Sentences such as *gacchāni* 'I shall go' don't need to overtly express the personal pronoun *aham*.

[5]Existence: *hastināpure vaṇik* "[there is/was a] merchant in [the city of] Hastināpura"; equivalence: *puṣpaṃ raktam*, see page 2.

are connected by a (coordinating) conjunction such as *ca* 'and' are interpreted as separate sentences in this paper. The following three cases need special consideration:

Overtly Expressed Subjects No period is inserted between main clauses separated by a coordinating conjunction such as *ca* 'and', if the first sentence overtly expresses the subject, and the following sentences use the same subject without overtly expressing it.

The particle *iti* The particle *iti* 'thus' marks the end of a direct speech, or of a personal opinion presented as a direct speech. The direct speech terminated by *iti* is interpreted as a matrix sentence and, therefore, not separated by an SB.

Formulae and interjections Interjections and formulaic phrases are marked as separate clauses, if they are not embedded as matrix sentences.[6]

3.3 Data

One annotator used the guideline (Section 3.2) to mark sentence ends in 226 chapters with 96,292 lexical tokens, which were drawn from the metrical texts in the Digital Corpus of Sanskrit (DCS, Hellwig (2015a)[7]). Although each chapter constitutes a single long sequence with unknown punctuation, most metrical texts simulate an oral presentation by inserting the stock line "[some person] said[8]" between closed narrative blocks. Therefore, the chapters have been split up into a total of 609 of such blocks ("sequences"), which represent the individual statements of the persons participating in a conversation. The epic Mahābhārata (MBH) contributes most of the data. Because the text has probably grown over centuries and incorporated diverse written and oral sources (Brockington, 1998), the predominance of the MBH does not bias the data unduly towards the style of one author.

A total of 9,562 SBs has been annotated. 85.6% of the SBs coincide with (double) *daṇḍa*s, which provide a strong baseline for SB detection (Table 3). The annotated chapters contain a total of 9,027 word types, 3,838 of which are hapax legomena. The sentences have a mean length of 10 tokens (median: 8), and 90% of all sentence lengths are found in the interval [3,20].

4 Experimental Settings

4.1 Features

This section describes which features were considered for SB detection, and motivates their use. Their influence on the prediction accuracy is reported in Section 5.1.

***Daṇḍa* information** Because *daṇḍa*s provide a strong baseline for SB detection (see Table 3), and omitting them drastically reduces the F score in all configurations, they are used as features in all settings. (Double) *daṇḍa*s are encoded as dummy variables.

Morphological Information Sanskrit has a rich, though partly ambiguous Indo-Aryan morphology. Nouns, adjectives, pronouns, and declinable verbal participles are inflected in eight cases, three numbers (including dual), and three genders, while finite verbal forms occur in three numbers, three persons, and over tenses and modes (aspects). Although Sanskrit also uses conjunctions to join subordinate and main clauses, verbal subordination is typically expressed by the indeclinable absolutive (gerund; *tvānta* and *lyabanta*).[9]

As morphology provides strong indications for the inner structure of a sentence, it is included in the feature set either in 1-hot- (**1h**, each observed combination of morphological subfeatures is mapped to

[6] Refer to Wackernagel (1978 (reprint from 1896, II, 5) on short sentences with a particle-like function.

[7] This corpus collects 279 texts from different domains with 3,950,000 tokens with gold-annotations on the morphological, lexical, and word semantic level.

[8] *vyāsa uvāca* "[The sage] Vyāsa said" is a typical example of these stock lines, which are always terminated by a single *daṇḍa*, and not written in śloka metre.

[9] A typical toy example for this construction runs like: *rāmo* ('Rāma' NSG) *vanaṃ* ('forest' ASG) *gatvā* ('go' ABS) *sītāṃ* ('Sītā' ASG) *paśyati* ('see' PR3.SG), "Rāma, having gone to the forest, sees Sītā." = "After Rāma has gone to the forest, he sees Sītā."

a distinct position in a 1-hot vector v_M) or in a decomposed encoding, in which each position of v_M encodes the presense or absence of a subfeature such as 'nominative' or 'perfect tense' (**dec**, refer to the featurization in Cotterell and Schütze (2015, 1289)). Both encoding modes (**1h**, **dec**) don't distinguish between different morphological derivations of tenses and modes.

Because the DCS does not contain syntactic annotations, morphological information is also used to generate possible syntactic links. Given a context size of $s = 5$, a word w_p at position p in a sequence, and the set of words $W_q = \{w_q | \, |q - p| \leq s, q \neq p\}$, a link between w_p and w_q is generated, if (1) $p < q$ and w_p belongs to the same compound (*samāsa*) as w_q, (2) w_p and w_q are nominal forms with the same case, number, and gender, (3) if one of w_p and w_q is a verb and the other one a congruent nominative, (4) if one of w_p and w_q is an absolutive and the other one a finite verb, or (5) if w_p and w_q belong to a set of correlative conjunctions and pronouns such as *yadā* 'when'-*tadā* 'then' or *yad* 'which'-*tad* 'that'. These links are encoded as two sums weighted with $\frac{1}{|p-q|}$ for the left and right contexts. Although the existence of a link does not guarantee that w_p and w_q belong to the same sentence, t-tests of the weighted values with the SB labels as binary factor yield highly significant test statistics of $t = -31.99$ (left) and $t = 45.70$ (right), such that testing the predictive power of these features appears justified.

Lexical Information Sanskrit has a rich vocabulary, and Sanskrit authors put importance on the use of synonyms. An unsophisticated text such as the MBH, for example, uses 35 synonyms to denote the warrior Arjuna, or 14 for the concept "mountain". As a consequence, one faces considerable data sparsity, when lexical information is used in 1-hot encoding. Low dimensional embeddings built from reduced vector space models (VSM, Turney and Pantel (2010)) or from neural networks (Bengio et al., 2003; Mikolov et al., 2011) have been shown to offer a workaround for this problem. Therefore, word embeddings generated with the `word2vec` tool (Mikolov et al., 2011)[10] are used as lexical features in all configurations marked with **w2v**.

As an alternative to a fully lexicalized model, the setting **indecl** uses the set of the most frequent 100 conjunctions and indeclinables in 1-hot encoding as the sole lexical information. This setting is motivated by the idea that these words indicate the basic structure of a sentence.

Yuret (1998) has shown that pointwise mutual information (PMI) between words can be used for building dependency structures. Therefore, normalized PMI for a window of size $s = 5$ around each w_p is added to the feature space in analogy to the syntactic links described above. PMI is either calculated from lexical information (**lpmi**), or from a mixture of lexical and word semantic data (**lspmi**).

4.2 Network Architecture and Settings

The RNN consists of a linear input layer with a dropout rate of 0.1 (Hinton et al., 2012), one or more bidirectional LSTM layers without peephole units, and a softmax output layer. All network weights are randomly initialized with a uniform distribution in $[-0.01, +0.01]$. The initial learning rate is set to 0.0008, and linearly decreased to the value of 0.0001. Training is performed with stochastic gradient descent, gradient clipping at the LSTM units, and a constant momentum of 0.95. Because the output of the network is a single binary variable, it is decoded using a threshold of 0.5. The model is implemented in C++.

The experiments reported in Section 5 are performed with a ten-fold cross-validation (CV). In order to make the results comparable to each other, the same random split of the data was used in all experiments.

5 Evaluation

5.1 Feature Selection

In order to assess how the features (Section 4.1) influence the classification results, flat networks with dropout and one bidirectional LSTM are trained on subsets of morphological and lexical features. Table 1 shows that the decomposed morphological encoding creates better results than the 1-hot encoding. It may be conjectured that the decomposed version can estimate the relevance of rare morphological features (e.g., genitive dual) from their more frequent subfeatures (e.g., genitive in all numbers). The

[10] Settings: Trained with full chapters, i.e. *daṇḍa*s were ignored; embedding size: 200, bow, window size: 10, 5 iterations.

Enc.	Links?	P	R	F
dec	no	85.08	**79.41**	**82.15**
dec	yes	**85.58**	77.92	81.57
1h	no	84.83	78.42	81.50
1h	yes	84.62	78.02	81.19

Table 1: Influence of morphological features on precision and recall of LSTMs. Enc.: encoding type; links: hardcoded morphological links used? LSTM architecture: dropout → bidirectional LSTM → softmax, 100 hidden units, 10 CVs, 50 iterations; lexical features: **freqindecl**, lexical links: **lpmi**

Lexicon	Links	P	R	F
none	**none**	83.50	78.59	80.97
freqindecl	**lpmi**	84.44	78.68	81.46
freqindecl	**none**	84.86	77.65	81.10
freqindecl	**lspmi**	85.08	79.24	82.05
w2v	**lpmi**	= Table 1, row 1		
w2v	**none**	**85.78**	79.01	82.26
w2v	**lspmi**	85.50	**79.31**	**82.28**

Table 2: Influence of lexical features; settings as in Table 1; decomposed morphological features (**dec**), no morphological links

hard-coded morphological links don't improve the performance, and are therefore discarded from the feature set.

Table 2 shows that lexical features have a noticeable, though not too large effect on the classification results. While the fully unlexicalized setting produces the worst results, the combination of word embeddings (**w2v**) with semantically enriched lexical links (**lspmi**) produces the best F scores (Table 2). This result demonstrates that neural language models create meaningful embeddings even from small training corpora, although the full lexical disambiguation of the training data is certainly helpful in learning proper representations.

The LSTM models for the final evaluation are trained on decomposed morphological features, neural embeddings of size 200, and semantically enriched lexical links. The basic architecture follows the description in Section 4.2, and the number of inner bidirectional LSTM layers is set to 2 or 3.

5.2 Comparing baseline models and LSTMs

Table 3 presents the baselines and the results for different LSTM architectures. The prosodical baselines are calculated by inserting an SB either at each *daṇḍa* or double *daṇḍa* ("baseline *daṇḍa*"), or at each double *daṇḍa* only ("baseline double *daṇḍa*"). As remarked in Section 1, editors tend to move double *daṇḍa*s by one line, if they are able to indicate an SB in this way. Therefore, double *daṇḍa*s present a rather precise baseline for SB detection in metrical Sanskrit texts, although their recall is low.

As many previous papers apply CRFs to SB detection (Section 2), CRFs trained with morphological features and different levels of lexical features are used as a second set of baselines. The central rows in Table 3 show that the F scores of CRFs are only slightly higher than those of the prosodical baselines. Error analysis reveals that CRFs base their predictions mainly on *daṇḍa* information, which explains the comparatively small differences between the F scores of the two baselines (75.55 vs. 73.80).

Bidirectional LSTMs significantly outperform both baselines. Comparing Tables 1, 2 and 3 shows that their F scores increase with their depth, i.e. the number of stacked LSTM layers. When using a deeper architecture, the strongest improvements are observed in the model recall. The best single model in Table 3 almost reaches precision and recall of the two metrical baselines, and improves the F score of the double *daṇḍa* baseline by almost 13%.

Classifier	Architecture	P	R	F
Prosodical baseline	*daṇḍa*	59.34	**85.54**	70.07
	double *daṇḍa*	**89.34**	62.86	73.80
CRF	no lex.	83.85	68.75	75.55
	freqindecl	83.82	68.74	75.54
	w2v	85.73	67.24	75.37
LSTM	2, dropout	86.92	80.70	83.70
	3	87.07	75.62	80.94
	3, dropout	88.53	85.13	**86.79**

Table 3: Comparison of baselines and LSTM architectures; settings for CRF: **lpmi**, **dec**, features extracted from a window of size ± 6 around each word, 10 CVs; settings for LSTM: **w2v**, embedding size: 200, **lspmi**, no morph. links; 100 hidden units in each bidirectional LSTM layer, 10 CVs, 50 epochs

		Length class				
b_i-e_i-i_i	Proportion	1 (≤ 4 words)	2 (5-9 w.)	3 (10-14 w.)	4 (15-29 w.)	5 (≥ 30 w.)
1-1-1	65.08	733 (41.91)	2234 (64.94)	2348 (80.25)	865 (66.03)	43 (31.39)
0-1-1	12.01	516 (29.50)	507 (14.74)	82 (2.80)	36 (2.75)	7 (5.11)
1-0-1	11.47	369 (21.10)	499 (14.51)	171 (5.84)	50 (3.82)	8 (5.84)
1-1-0	6.98	8 (0.46)	90 (2.62)	238 (8.13)	272 (20.76)	59 (43.07)
0-0-0	0.38	5 (0.29)	9 (0.26)	10 (0.34)	9 (0.69)	3 (2.19)
0-0-1	1.41	89 (5.09)	32 (0.93)	6 (0.21)	6 (0.46)	2 (1.46)
1-0-0	1.61	15 (0.86)	33 (0.96)	49 (1.67)	46 (3.51)	11 (8.03)
0-1-0	1.07	14 (0.80)	36 (1.05)	22 (0.75)	26 (1.98)	4 (2.92)

Table 4: Sentence based evaluation of the output of the best RNN from Table 3, stratified by sentence length classes (columns 3ff.). Column 1: $b_i = 1$: b(eginning) of sentence s_i is detected; $e_i = 1$: e(nd) detected; $i_i = 1$: the i(nner) part of s_i does not contain superfluous SBs.

5.3 Discussion

Table 3 demonstrates that RNNs clearly outperform both baselines, and that stacking the bidirectional LSTM layers further improves the performance. However, the results don't tell much about the actual usability of the SB labeler, especially about how many full sentences were labeled correctly, and which sentence structures or types are prone for errors. To assess these questions, a metric similar to the "strict" evaluation in Liu and Shriberg (2007) is used. Instead of measuring the annotation precision for single instances of SBs, this metric considers if full sentences have been annotated correctly, and where errors occur in their annotation. For every sentence s_i in the gold annotation it is tested, if the RNN has marked the beginning b_i of s_i (= the end of s_{i-1}) and the end e_i of s_i correctly, and if it has inserted additional SBs in between b_i and e_i (variable i_i). A sentence is accepted as correct in this evaluation, if b_i and e_i are correct ($b_i = 1$, $e_i = 1$), and if there exist no superfluous SBs between them ($i_i = 1$).

Table 4 shows the proportions of the $2^3 = 8$ combinations of these three binary labels for the output of the best RNN from Table 3 (3 hidden layers, dropout). The model achieves an overall "strict" accuracy of 65.08% (configuration 1-1-1, column 2). If the configurations in which only one SB has been missed (0-1-1, 1-0-1) or in which wrong SBs have been inserted between two correctly labeled SBs (1-1-0) are accepted as partial matches, this "lenient" accuracy goes up to 95.54%.

It has been noted in Section 1 that sentences starting or ending in the middle of a text line convey important text historical information. In addition, the CRF failed almost completely to detect these more unusual SBs. In order to examine how these cases are handled by the LSTM, the output of the best LSTM from Table 3 has been stratified according to the start and end positions of the sentences. It may be expected that sentences starting at the beginning of a text line (b) and ending at a *daṇḍa* (d) or double *daṇḍa* (d2) may have lower error rates than those starting and/or ending in the middle of a line (m). The results displayed in Table 5 support this hypothesis. The highest accuracy rates are observed for the

Start-end	Corr.	1 err.	>1 err.	Acc.
b-m	463	673	149	36.03
b-d	1185	495	72	67.64
b-d2	4034	1044	65	78.44
m-m	27	37	34	27.55
m-d	146	202	68	35.10
m-d2	368	461	39	42.40

Table 5: Sentence based evaluation of the output of the best RNN from Table 3, stratified by start and end positions of sentences. b: Sentence starts at the b(eginning) of a text line, m: in the m(iddle); d/d2: sentence ends at a *daṇḍa*/double *daṇḍa*

configurations b-d and b-d2, while accuracy rates for *-m-* configurations are clearly lower. Although these cases constitute only a minority of the training data, and their accuracy rates may rise when more labeled data are available, the presence of the (double) *daṇḍa* feature (page 4) is certainly most relevant for the observed differences in the accuracy levels.

Columns 3ff. of Table 4 provides another view of the same data, which have been stratified with regard to length classes of sentences. As could be hypothesized from Table 5, the highest accuracy is observed for length class 3, which contains, among others, all sentences that extend over two lines between two double *daṇḍa*s (subset of configuration b-d2). A closer inspection of class 5 (sentences containing at least 30 words) shows, that most of the correct instances of this class have between 30 and 50 words, although the model also marks two very long sentences correctly. MBH 1.19.3-15, a description of the ocean, is a right-branching construction typical for poetic style (*kāvya*). The initial phrase *dadṛśāte tadā tatra samudram* ("Then, both of them saw the ocean there") is expanded by several lines of accusative constructions that depend on the head word *samudram*. Apart from congruent adjectives and appositions, the expansions also contain subordinate participle clauses. This means that the whole sentence can not be reduced to an easily memorizable pattern in the form verb-adverb*-acc*, and demonstrates that stacked LSTM units are in principle able to capture such long-range syntactic dependencies.

A considerable number of errors is produced by short sentences that start or end in the middle of a text line (*-m-* configurations), and for which only one boundary is detected correctly (configurations 0-1-1 and 1-0-1 with length class 1 in Table 4). One of the syntactical patterns that produce most of the errors in this class consists of sequences of words in nom. sg. lacking a copula as observed in MBH 1.147.11:

ātmā	*putraḥ•*	*sakhā*	*bhāryā•*
self:NSGM	son:NSGM	friend:NSGM	wife:NSGF

"[The] son [is the] self. [The] wife [is a] friend."

Another problematic pattern is formed by sequences of the form verb-sg acc-sg* (MBH 6.41.64):

anumānaye	*tvāṃ•*	*yotsyāmi* ...
ask:PR1.SG	you:ASG	fight:FUT1.SG

"I ask you [for permission]. I will fight ..."

As soon as more training data are available, a Viterbi search over decoded sentence patterns or an additional CRF layer (Huang et al., 2015) may help to reduce the number of such errors.

6 Conclusion

Although the proposed deep bidirectional LSTM model clearly outperforms the metrical and CRF baselines, its accuracy is currently not high enough for performing a reliable unsupervised annotation of SBs. As the evaluation of short sentences has shown, many of the problematic cases cannot be solved on the morpho-syntactic level, but require comprehensive lexical and word semantic information. This finding suggests that a larger amount of training data, including more tokenized texts and more annotated SBs, may improve the performance. In this context, the LSTM model will be used for pre-annotating SBs. Another line of future research will concentrate on the representation of input features. Recent studies

such as Labeau et al. (2015), but also Hellwig (2015b) for Sanskrit have demonstrated that the processing of morphologically rich languages may benefit from using sub-word units, skipping lexicalization altogether, or integrating it into a "deeper level" of the network architecture. Given the complexity of Sanskrit phonetics and the richness of its vocabulary, such an approach may prove useful, as soon as more SB annotations are available.

References

Timothy Baldwin and Manuel Paul Anil Kumar Joseph. 2009. Restoring punctuation and casing in English text. In *Advances in Artificial Intelligence*, Springer, pages 547–556.

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

John Brockington. 1998. *The Sanskrit Epics*. Brill, Leiden.

George Cardona. 1976. *Pāṇini. A Survey of Research*. Mouton, The Hague - Paris.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proceedings of the 2015 Annual Conference of the NACL*, ACL, pages 1287–1292.

Madhav M. Deshpande. 1991. Pāṇinian syntax and the changing notion of sentence. In Hans Heinrich Hock, editor, *Studies in Sanskrit Syntax*, Motilal Banarsidass Publishers, Delhi, pages 31–43.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 76(5):378–382.

Brendan Gillon and Benjamin Shaer. 2005. Classical Sanskrit, "wild trees", and the properties of free word order languages. In Katalin É. Kiss, editor, *Universal Grammar in the Reconstruction of Ancient Languages*, De Gruyter, Berlin, Boston, pages 457–494.

Oliver Hellwig. 2015a. Morphological disambiguation of Classical Sanskrit. In Cerstin Mahlow and Michael Piotrowski, editors, *Systems and Frameworks for Computational Morphology*. Springer, Cham, pages 41–59.

Oliver Hellwig. 2015b. Using Recurrent Neural Networks for joint compound splitting and Sandhi resolution in Sanskrit. In Zygmunt Vetulani and Joseph Mariani, editors, *Proceedings of the 7th LTC*. pages 289–293.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* .

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

Hans Henrich Hock. 2013. Some issues in Sanskrit syntax. In Peter M. Scharf and Gérard Huet, editors, *Proceedings of the Seminar on Sanskrit syntax and discourse structures*. Paris.

E. Washburn Hopkins. 1901. *The Great Epic of India. Its Character and Origin*. Charles Scribner's Sons, New York.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* .

Franz Kielhorn. 1888. *Grammatik der Sanskrit-Sprache*. Dümmler Verlag, Berlin.

Paul Kiparsky. 2009. On the architecture of Pāṇini's grammar. In Gérard Huet, Amba Kulkarni, and Peter Scharf, editors, *Sanskrit Computational Linguistics*, Springer, Berlin, Heidelberg, pages 33–94.

Amba Kulkarni, Preeti Shukla, Pavankumar Satuluri, and Devanand Shukl. 2015. How free is 'free' word order in Sanskrit? In Peter M. Scharf, editor, *Sanskrit syntax*. pages 269–304.

Matthieu Labeau, Kevin Löser, and Alexandre Allauzen. 2015. Non-lexical neural architecture for fine-grained POS tagging. In *Proceedings of the 2015 Conference on EMNLP*. pages 232–237.

Yang Liu and Elizabeth Shriberg. 2007. Comparing evaluation metrics for sentence boundary detection. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing*. volume 4, pages 182–185.

Yang Liu, Andreas Stolcke, Elizabeth Shriberg, and Mary Harper. 2005. Using Conditional Random Fields for sentence boundary detection in speech. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. pages 451–458.

James Martens and Ilya Sutskever. 2011. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. pages 1033–1040.

Bimal Krishna Matilal. 1966. Indian theorists on the nature of the sentence (vākya). *Foundations of Language* 2:377–393.

Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. 2011. Strategies for training large scale neural network language models. In *2011 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. pages 196–201.

Milman Parry. 1930. Studies in the epic technique of oral verse-making i: Homer and Homeric style. *Harvard Studies in Classical Philology* 41:73–147.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*.

Hartmut Scharfe. 1977. *Grammatical Literature*. A History of Indian Literature, Volume 5, Fasc. 2. Otto Harrassowitz, Wiesbaden.

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

Frank Seide, Gang Li, Xie Chen, and Dong Yu. 2011. Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *Automatic Speech Recognition and Understanding (ASRU)*. IEEE, pages 24–29.

Sven Sellmer. 2015. *Formulaic Diction and Versification in the Mahābhārata*. Adam Mickiewicz University Press, Poznań.

John D. Smith. 1987. Formulaic language in the epics of India. In B. Almqvist, S.Ó. Catháin, and P.Ó. Héalaí, editors, *The heroic process: Form, Function, and Fantasy in Folk Epic*. Glendale Press, pages 591–611.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Ottokar Tilk and Tanel Alumäe. 2015. LSTM for punctuation restoration in speech transcripts. In *Interspeech 2015*. Dresden, Germany.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research* 37(1):141–188.

Jakob Wackernagel. 1978 (reprint from 1896). *Altindische Grammatik*. Vandenhoek und Ruprecht, Göttingen.

Deniz Yuret. 1998. *Discovery of Linguistic Relations Using Lexical Attraction*. Ph.D. thesis, Massachusetts Institute of Technology.

Yanqing Zhao, Chaoyue Wang, and Guohong Fu. 2012. A CRF sequence labeling approach to Chinese punctuation prediction. In *26th Pacific Asia Conference on Language, Information and Computation (PACLIC 26)*. pages 508–514.

Consistent Word Segmentation, Part-of-Speech Tagging and Dependency Labelling Annotation for Chinese Language

Mo Shen[1], Wingmui Li[2], Hyunjeong Choe[1], Chenhui Chu[3], Daisuke Kawahara[4], Sadao Kurohashi[4]
[1] Google Inc., California, USA
[2] The Chinese University of Hong Kong, Shatin, Hong Kong
[3] Japan Science and Technology Agency
[4] Graduate School of Informatics, Kyoto University, Kyoto, Japan
{moshen|wjli|hyunjeongc}@google.com,
chu@pa.jst.jp, {dk|kuro}@i.kyoto-u.ac.jp

Abstract

In this paper, we propose a new annotation approach to Chinese word segmentation, part-of-speech (POS) tagging and dependency labelling that aims to overcome the two major issues in traditional morphology-based annotation: Inconsistency and data sparsity. We re-annotate the Penn Chinese Treebank 5.0 (CTB5) and demonstrate the advantages of this approach compared to the original CTB5 annotation through word segmentation, POS tagging and machine translation experiments.

1 Introduction

The definition of "word" is an open problem in Chinese linguistics. In previous studies of Chinese corpus annotation (Duan et al., 2003; Huang et al., 1997; Xia, 2000), the judgement of word-hood of a meaningful string is based on the analysis of morphology: A morpheme in Chinese is defined as the smallest combination of meaning and phonetic sound in Chinese language, which can be classified into two major types:
1). **Free morphemes**, which can either be words by themselves or form words with other morphemes; and
2). **Bound morphemes**, which can only form words by attaching to other morphemes.

An issue with word definition using morpheme classification is that, it potentially undermines the consistency of the representation of words. For example, "论" (theory) is a bound morpheme, therefore the string "进化论" (theory of evolution) is treated as a word; on the other hand the string "进化 | 理论" (theory of evolution) are treated as two words, despite the fact that the two strings have the same meaning and structure. In another example, "者" (person) is considered as a bound morpheme, therefore "反对自由贸易者" (people who are against free trade) is treated as one word, while the string without the bound morpheme, i.e. "反对 | 自由 | 贸易" (be against free trade), can only be treated as a phrase of three words.

The morphology-based word definition can also make the data sparsity problem worse in corpus annotation. As an evidence, in the Penn Chinese Treebank 5.0 (CTB5) which is an annotated corpus widely used to train Chinese morphological analysis systems, we found that one of the major sources of the out-of-vocabulary (OOV) words is the compounds that end with a monosyllabic bound morpheme. For example, compounds 利用率 (utility rate) and 次品率 (rate of defective product) end with the bound morpheme 率 (rate); 完成度 (degree of completion) and 活跃度 (degree of activity) end with the bound morpheme 度 (degree); 持续性 (sustainability) and 挥发性 (property of volatile) end with the bound morpheme 性 (property). While these compounds are sparse in the corpus, the morphemes which they

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 298–308, Osaka, Japan, December 11-17 2016.

POS Pattern	Example
pronoun + noun	我校 (this university)
locative + noun	后门 (back door)
locative + verb	前述 (described above)
noun + locative	室内 (indoor)
pronoun + locative	此外 (besides)
adverb + verb	猝死 (sudden death)
noun + noun	厂房 (factory plant)
noun + measure	车辆 (vehicles)
adjective + noun	佳酿 (wines)
adjective + measure	高层 (high level)
verb + verb	抽取 (extract)
verb + particle	写完 (finish writing)
verb + adjective	打碎 (break)
verb + locative	综上 (accordingly)
verb + noun	辞职 (resign)
adjective + adjective	优雅 (elegant)
adverb + adjective	最新 (latest)
determiner + noun	各界 (all walks of life)
determiner + temporal	翌日 (the next day)

Table 1. Disyllabic character-level POS patterns.

CTB5 Example	Re-annotation
副主席/NN (vice president)	副/JJ (vice) 主席/NN (president)
透明度/NN (transparency)	透明/JJ (transparent) 度/SFN (degree)
非生产性/NN (unproductiveness)	非/JJ (none) 生产/VV (produce) 性/SFN (property)
中央集权式/JJ (politically centralized)	中央/NN (center) 集权/NN (centralization) 式/SFA (type)

Table 2. Some examples of the word and POS annotation in the original CTB5 and our re-annotation.

consist of can be frequently observed; this means these OOV words can be observed and learnt by a word segmenter if we split the morphemes as individual words in the annotation.

In this paper, we propose a simple annotation approach for Chinese word segmentation that overcomes the two issues: inconsistency and data sparsity, which are found in the traditional morphology-based annotation approach. We further propose a tagset for part-of-speech tagging and a label set for dependency labelling, which are consistent with our word segmentation strategy and capture more Chinese-specific syntactic structures. We re-annotate the entire CTB5 using this approach, and through word segmentation, POS tagging and machine translation experiments we demonstrate the advantages of our annotation approach compared to the original approach adopted in CTB5.

The remainder of this paper is organized as follows: in section 2 we will describe our proposed annotation approach to word segmentation; in section 3 we will present a POS tagset which is consistent with our word segmentation strategy and a new dependency label set; in section 4 we will demonstrate the effectiveness of our approach compared to the original CTB5 through experiments; we will conclude our work in the last section.

2 Word Segmentation Annotation

We categorize the words in CTB5 into three categories: Common words, names, and idioms. For names and idioms, we keep them as individual words since their word boundaries are relatively easy to recognize and the consistency in manual annotation can be achieved with less efforts. We will mainly focus on describing the treatments of common words in this section.

Tag	Description	Count in CTB5	Proposed annotation
NN	Noun	134,321	137,816
PU	Punctuation	75,794	75,935
VV	Verb	68,789	75,033
AD	Adverb	36,122	35,922
NR	Proper Noun	29,804	30,985
P	Preposition	17,280	17,721
CD	Cardinal Number	16,030	21,493
M	Measure Word	13,668	18,091
JJ	Adjective	12,979	13,898
DEC	Complementizer	12,310	12,346
DEG	Genitive Marker	12,145	12,145
NT	Temporal Noun	9,467	4,524
LC	Locative	7,676	0
VA	Predicative Adjective	7,630	7,518
CC	Coordinating Conjunction	7,137	7,134
PN	Pronoun	6,536	6,646
DT	Determiner	5,901	5,970
VC	Copula	5,338	5,521
AS	Aspect Particle	4,027	4,033
VE	“you3” (“have”)	2,980	2,980
OD	Ordinal Number	1,661	1,661
MSP	Other Particles	1,316	1,316
ETC	“deng3” (“etc.”)	1,287	0
CS	Subordinating Conjunction	888	888
BA	Causative Auxiliary	751	756
DEV	Manner Marker	621	627
SP	Sentence-final Particle	466	466
SB	Short Passive Auxiliary	451	451
DER	Resultative Marker	258	258
LB	Long Passive Auxiliary	245	245
FW	Foreign Word	33	391
IJ	Interjection	12	17
X	Unknown	6	6
SFN*	Nominal Suffix	0	13,212
SFA*	Adjectival Suffix	0	438
SFV*	Verbal Suffix	0	129

Table 3. Proposed tagset for part-of-speech tagging. The underlined characters in the examples correspond to the tags on the left-most column. The CTB POS are also shown.

The key in our method to define the boundaries of common words is the character-level POS pattern. Character-level POS has been introduced in previous studies (Zhang et al., 2013; Shen et al., 2014) which captures the grammatical roles of Chinese characters inside words; we further develop this idea and use it as a criterion in word definition.

We treat a meaningful disyllabic strings as a word if it falls into one of the character-level POS patterns listed in Table 1. The reason we focus on disyllabic patterns instead of other polysyllabic ones is that, based on our observation, meaningful strings with 3 or more syllables (other than names and idioms) are always compounds in Chinese, and therefore can be segmented into a sequence of monosyllabic and disyllabic tokens based on their internal structures. On the other hand, the internal structure in a disyllabic token, though still exists, is more implicit and harder to describe with syntactical relations; we believe that it would increase the difficulties for subsequent tasks, such as dependency parsing, if we further segment these disyllabic strings.

Following this strategy, a polysyllabic word can be then segmented based on its structure. This is illustrated with examples in Table 2.

3 Part-of-Speech and Dependency Label Set

To perform POS tagging re-annotation on CTB5 together with our proposed word segmentation approach, we use a POS tagset which is derived from the one used in the original CTB5 annotation. We show the tagset in Table 3 with comparison of number of occurrences of each tag in the original CTB5 and the re-annotated version, respectively. The tagset introduces several changes: First, we eliminate the use of the "LC" tag for locative words. This tag is assigned to all words that indicate locations and directions, such as 上 (up), 下(down), 左 (left), 右 (right), 内 (inside), 外 (outside) etc.. We instead tag these words based on their real syntactic roles in sentences, such as "NN" (noun), "AD" (adverb) or "VV" (verb). Second, we add three new tags into the tagset for suffixes: "SFN" (nominal suffix), "SFA" (adjectival suffix), and "SFV" (verbal suffix). These tags are given to monosyllabic tokens appearing at the end of compounds, which are the bound morphemes in the traditional view. Based on our observation, these tokens have the ability to determine the syntactic role of the entire compound. For example, any compound that end with a nominal suffix "度" (degree) always act as nouns in a sentence. It should be noted that because of this characteristic of suffixes, we can tag the children of suffixes in compounds based on their meaning but not their syntactic roles. We show some examples in Table 2 to illustrate our POS tagging strategy for compounds.

In Table 4 we present a dependency label set developed based on the Stanford Dependencies (De Marneffe et al., 2006) and its Chinese version (Chang et al., 2009), which defines 45 dependency relations for Chinese sentences. This label set is also closely related to the Universal Dependency[1] with many of their labels compatible with each other. We explain the major characteristics of our label set in the following subsection.

3.1 Chinese Specific Labels

dislocated The label "dislocated" is originally defined in the *universal dependencies* for languages such as Japanese to describe the syntactic relation of words in a topic–comment structure, but is not defined for Chinese. However, in Chinese it is frequent to see the topic–comment structure in a sentence, for example:

1. 這/this 本/-measure- 書/book 他/he 買/buy 的/-particle- (This book, he bought it)

In this sentence, 这本书 (this book) is the topic and 他买的 (he bought) is the comment. One common view of the syntactic structure of this sentence is that, 他 (he) is the subject of the predicate 他 (buy), and 书 (book) is the direct object. This treatment sees a topic–comment structure as having an OSV (object-subject-verb) word order, which is acceptable; it however has some problems in certain cases, for example:

2. 這/this 本/-measure- 書/book 他/he 買/buy 的/-particle- 昨天/yesterday 不見/disappear 了/-particle- (This book that he bought disappeared yesterday)

In this sentence, 书 (book) is still the direct object of 买 (buy), while it is also the subject of 不见 (disappear). Because of the nature of the dependency grammar we adopted, for such a structure we would have to choose one relation for 书 (book), either "nsubj" or "dobj", and discard the other relation which would cause a loss of the syntactic information encoded in the parse tree.

Moreover, the OSV word order cannot explain all topic-comment structure such as the following example:

3. 這/this 場/-measure- 火/fire 幸虧/fortunately 消防/firefighting 隊/team 來/come 得/-particle- 早/early (This fire, fortunately the firefighters came in time)

[1] http://universaldependencies.org/

Label	Description	Example Phrase	Example Dependency
acomp	adjectival complement	鞋是全新的 (the shoes are brand new)	**acomp**(是 are, 全新 brand new)
advmod	adverbial modifier	他看上去很疲倦 (he looks very tired)	**advmod**(疲倦 tired, 很 very)
amod	adjectival modifier	漂亮的首飾 (cute accessory)	**amod**(首飾 accessory, 漂亮 cute)
appos	appositional modifier	總統奧巴馬 (president Obama)	**appos**(奧巴馬 Obama, 總統 president)
asp	aspect marker	他給了我一本書 (he gave me a book)	**asp**(給 gave, 了 -aspect-)
attr	attributive modifier	他是個醫生 (he is a doctor)	**attr**(是 is, 醫生 doctor)
aux	auxiliary verb	必須解決 (must solve)	**aux**(解決 solve, 必須 must)
auxpass	passive auxiliary	他被刺殺了 (he was assassinated)	**auxpass**(刺殺 assassinated, 被 -auxiliary-)
auxcaus	causative auxiliary	把問題解決(solve the problem)	**auxcaus**(解決 solve, 把 -auxiliary-)
cc	coordinating conjunction	聰明又可愛 (smart and cute)	**cc**(聰明 smart, 又 and)
ccomp	closed clausal complement	他說他喜歡游泳 (He said that he likes swimming)	**ccomp**(說 said, 喜歡 likes)
conj	conjunct	聰明又可愛 (smart and cute)	**conj**(聰明 smart, 可愛 cute)
csubj	clausal subject	能夠代表祖國參賽是他的夢想 (being able to play in the game for his country is his dream)	**csubj**(是 is, 參賽 play)
csubjpass	clausal passive subject	他在考試中作弊被老師發現了 (that he cheated during the exam is found out by the teacher)	**csubjpass**(發現 find out, 作弊 cheet)
dep	undefined dependency	添加一個日程安排時間星期二地點 3 樓 (add an event, time Tuesday, location 3rd floor)	**dep**(時間, 添加)
det	determiner	那本書 (that book)	**det**(本 -measure-, 那 that)
discourse	discoursal modifier	唉，終於到星期五了 (oh, thank God it's Friday)	**discourse**(到 is, 唉 oh) **discourse**(到 is, 了-sentence-final particle-)
dislocated	dislocated modifier	書是他買的 (book he bought) 這場火幸虧消防隊來得早. (this fire, fortunately the firefighters came in time)	**dislocated**(書 book, 買 buy) **dislocated**(火 fire, 來 come)
dobj	direct object	買了一本書(bought a book)	**dobj**(買 bought, 書 book)
foreign	foreign compound	職棒大聯盟（Major League Baseball）	**foreign**(Major, League) **foreign**(Major, Baseball)
iobj	indirect object	他給了我一本書 (he gave me a book)	**iobj**(給 gave, 我 me)
list	list relation	添加一個日程安排時間星期二地點 3 樓 (add an event, time Tuesday, location 3rd floor)	**list**(時間 time, 地點 location)
mark	clause marker	他把信件給我之後就走了 (he left after he gave me the letter)	**mark**(給 give, 之後 after) **mark**(走 leave, 就 then)
mes	measure relation	一本書(a book)	**mes**(書 book, 本 -measure-)

ncomp	nominal complement	坐在椅子上 (sit on a chair)	**ncomp**(椅子 chair, 上 -complementizer-)
neg	negation modifier	不擅長 (be not good at)	**neg**(擅長 be good at, 不 not)
nn	noun compound modifier	原油期貨價格 (oil futures price)	**nn**(價格 price, 原油 oil) **nn**(價格 price, 期貨 futures)
npadvmod	noun phrase adverbial modifier	大約十米左右寬 (about 10 m wide)	**npadvmod**(寬 wide, 米 m)
nsubj	nominal subject	他給了我一本書 (he gave me a book)	**nsubj**(給 gave, 他 he)
nsubjpass	passive nominal subject	他被刺殺了 (he was assassinated)	**nsubjpass**(刺殺 assassinated, 他 he)
num	numeric modifier	一本書 (a book)	**num**(本 -measure-, 一 a)
p	punctuation	梨、橘子和香蕉 (pears, oranges, bananas)	**p**(梨 pears, 、)
pcomp	prepositional complement	由於路上人太多，我遲到了 (because it was so crowded, I was late)	**pcomp**(由於 because, 太多 so crowded)
pobj	prepositional object	他坐在椅子上 (he sits on a chair)	**pobj**(在 on, 椅子 chair)
ps	associative marker	這是我的家 (this is my home)	**ps**(我 me, 的 ‘s)
poss	possessive modifier	這是我的家 (this is my home)	**poss**(家 home, 我 me)
prep	prepositional modifier	他坐在椅子上 (he sits on a chair)	**prep**(坐 sits, 在 on)
prt	phrasal verb particle relation	他們打起來了 (they started a fight) 把數據整理成報告 (summarize the data into a report)	**prt**(打 fight, 起來 -auxiliary-) **prt**(整理 summarize, 成 become)
rcmodrel	relative clause complementizer	他回來的時候 (by the time he came back)	**rcmodrel**(回來 come back, 的 -complementizer-)
		這本是他買的書 (this is the book he bought)	**rcmodrel**(買 buy, 的 -complementizer-)
rcmod	relative clause modifier	他回來的時候 (by the time he came back)	**rcmod**(時候 time, 回來 come back)
		這本是他買的書 (this is the book he bought)	**rcmod(**書 book, 買 buy)
suff	suffix relation	科技界 (sci-tech industry)	**suff**(界 industry, 科技 sci-tech)
tmod	temporal modifier	他回來的時候天已經亮了 (it was bright outside by the time he came back)	**tmod**(亮 bright, 時候 time)
topic	topic marker	這本書是他買的 (this is the book he bought)	**topic**(書 book, 是 is)
		這是他們所不能想像的 (this is what they can’t imagine)	**topic**(這 this, 是 is)
vmod	verb modifier	他打開門發現屋裏有人 (he opened the door and found out there is somebody inside)	**vmod**(發現 found out, 打開 open)
xcomp	open clausal complement	他不喜歡打網球 (he doesn’t like to play tennis)	**xcomp**(喜歡 like, 打 play)

Table 4. Proposed dependency label set.

Unlike in the other two examples, the topic here, 這場火 (this fire), is not the direct object of the verb in the comment, 幸虧消防隊來得早 (fortunately the firefighters came in time).

To overcome these difficulties, we employ a different view which treats the topic–comment structure as having double subjects in a SSV word order. We define the first subject, 这本书 (this book) in example 2, as the head in a "dislocated" relation, and the subject-verb phrase, 他买的 (he bought) in example 2, as the modifier. The head in this dislocated relation can then form a "nsubj" (nominal subject) relation with the main predicate of the sentence, 不见 (disappear). Similarly, in example 3, the topic and the comment still form a dislocated relation even though the topic is not a direct object of the verb in the comment.

prt and ***prep*** We define the "prt" relation in two ways:

i. A relation between a verb and a particle. For example, 想像 (imagine) is the head in a "prt" relation of 所 (particle) in the sentence 這是他們所不能想像的 (this is what they can't imagine).
ii. A relation between a verb and its succeeding complement. For example, 打掃 (clean) is the head in a "prt" relation of 完 (finish) in the sentence 房間打掃完了 (the room has been cleaned).

We use the "prt" relation in the second case to capture the predicate-complement structure in Chinese. The verb 完 (finish) in the second example above functions to complement the meaning of the main verb, 打掃 (clean), and the sentence is still grammatical when the complement verb is removed: 房間打掃了 (the room is cleaned).

The complement verb sometimes also functions as a coverb in a serial verb construction, which takes its own direct object. For example:

4. 把/-auxiliary- 數據/data 整理/summarize 成/become 報告/report (summarize the data into a report)

Here the two verbs 整理 (summarize) and 成(become) form a "prt" relation, while they are the heads of 數據 (data) and 報告 (report) in the "dobj" relation.

A difficulty with labeling "prt" is that, it can be easily confused with the "prep" (prepositional modifier) relation. For example, one can argue that 成 (become) is a preposition instead of a verb and should be tagged as IN, so that the relation between 整理 (summarize) and 成 (become) would be "prep". To overcome this ambiguity, we apply a simple test: If the phrase headed by the word with a VV vs. IN ambiguity can be moved to a position before the main verb, then this word is a preposition and a prepositional modifier of the main verb; otherwise it is a verb. Here since the phrase "成 報告" (into report) cannot be moved to the position before 整理 (summarize), it should in fact be a verb phrase, not a prepositional phrase.

suff We define the suffix relation in a compound which has a "stem-suffix" structure. The suffix word with a POS tag SFN, SFA, or SFV is the root of the subtree formed by the words in the compound. It has one and only one child in this subtree, which is the head of the "stem", and the dependency relation between them is labelled as "suff".

The motivation of employing the "suff" label is to relieve the data sparseness problem of word forms in annotated corpora. Compounds, especially those with a "stem-suffix" structure, is a major source of new words in Chinese language. These compounds, however, often share a set of suffix words which has a limited amount of instances. We think it is more effective for a parser to learn from features with word forms by treating the suffix words as the heads of compounds.

4 Evaluation

4.1 Re-annotated Corpus

We re-annotated the entire CTB5 with our proposed word segmentation and POS tagging annotation strategies. We further re-annotated 3,000 sentences which are randomly sampled from the training set of CTB5 using our proposed dependency label set. This re-annotated set is compared with the same sentences with the original annotation in a machines translation experiment in section 4.3.

	CTB5	Re-annotated
Number of tokens	493,938	516,581
Avg. token length	1.63	1.55
Ratio of unknown words	14.67%	12.82%
Ratio of unknown word-POS pairs	15.02%	13.28%

Table 5. Statistics of the original CTB5 and our re-annotated version.

(a) Word Segmentation Results

Corpus	P	R	F
Original	97.21	97.36	97.28
Re-annotated	97.97	97.56	97.76
Re-annotated-partial	97.68	97.63	97.65

(b) Joint Segmentation and POS Tagging Results

Corpus	P	R	F
Original	93.42	93.56	93.49
Re-annotated	94.55	94.16	94.35

Table 6. Experimental results for morphological analysis on CTB5.

To evaluate the consistency of our annotation, 4 trained annotators were divided into two equal groups to perform 2-way annotation on a small subset (first 100 sentences in files 301-325), and each pair of annotators were assigned with 50 sentences. The inter-annotator agreement is 99.10% for segmentation, 98.37% for POS tagging, and 95.62% for dependency labeling.

Table 5 shows some of the statistics of the original and the re-annotated CTB5. We split CTB5 in the same data division as in previous studies (Jiang et al., 2008a; Jiang et al., 2008b; Kruengkrai et al., 2009; Zhang and Clark, 2010; Sun, 2011). The training, development and test set have 18,089, 350 and 348 sentences, respectively. Compared to the original CTB5, the re-annotated training set has a lower percentage of unknown words and unknown word-POS pairs found in the corresponding test set. This is consistent with our observation that compounds with internal structures are one of the major sources of OOV words.

4.2 Morphological Analysis Experiments

We compared the performance of a state-of-the-art joint word segmentation and part-of-speech tagging system (Kruengkrai et al., 2009) on the original and our re-annotated CTB5. We used the position-of-character (POC) tagset and the baseline feature set described in (Shen et al., 2014).

We trained all models using the averaged perceptron (Collins, 2002), which is an efficient and stable online learning algorithm. The models applied on all test sets are those that result in the best performance on the dev sets. To learn the characteristics of unknown words, we built the system's lexicon using only the words in the training data that appear at least 2 times.

We use precision, recall and the F-score to measure the performance of the systems. Precision (P) is defined as the percentage of output tokens that are consistent with the gold standard test data, and recall (R) is the percentage of tokens in the gold standard test data that are recognized in the output. The balanced F-score (F) is defined as $\frac{2 \cdot P \cdot R}{P+R}$.

We compared the performance of the morphological analyzer on the original and the re-annotated CTB5. The results of the word segmentation experiment and the joint experiment of segmentation and POS tagging are shown in Table 6(a) and Table 6(b), respectively. Each row in these tables shows the performance of the same system trained on the corresponding corpus.

For "Re-annotated-partial" in Table 6(a), we applied a different setting in order to directly compare the annotation consistency and data sparsity between the two corpora: We used the training set from the re-annotated corpus to train the system but the test set from the original corpus in the evaluation. To make the evaluation meaningful, we added an extra criterion when calculating the precision and the

System	BLEU-4
Character	31.60
Original	31.46
Re-annotated	**32.08**

Table 7. Experimental results for Chinese-Japanese machine translation on ASPEC corpus using Moses system.

System	BLEU-4
Original	32.00
Re-annotated	**32.97**

Table 8. Experimental results for Chinese-Japanese machine translation on ASPEC corpus using KyotoEBMT system.

recall: If the outmost boundaries of a sequence (two or more) of output tokens are consistent with a token in the test set, we consider that the output correctly identifies this token in the test set.

The results show that, the morphological analyzer can obtain higher accuracies in both word segmentation (0.48 points absolute in F-score) and joint (0.86 points absolute in F-score) experiments. Furthermore, in the word segmentation experiment "Re-annotated-partial" where we mapped the output of the system which is trained using the re-annotated training data to the original CTB5 test set, the accuracy is significantly higher[2] than that of the "Original", which demonstrates the better consistency in our re-annotation corpus.

4.3 Machine Translation Experiments

To show that a morphological analysis system and a dependency parsing system can both benefit from our re-annotation, we conducted two sets of Chinese-to-Japanese machine translation experiments where a morphological analyzer and a dependency parser are used respectively.

The parallel corpus we used is the Chinese-Japanese part of the Asian Scientific Paper Excerpt Corpus (ASPEC)[3], containing 672k sentence pairs. We used 2,090 and 2,107 additional sentence pairs for tuning and testing, respectively.

In the first set of experiments, we segmented the Japanese sentences using JUMAN (Kurohashi et al., 1994), and the Chinese sentences using the same morphological analyzer described in the last subsection. For decoding, we used the state-of-the-art phrase based statistical machine translation toolkit Moses (Koehn et al., 2007) with default options. We trained the 5-gram language models on the target side of the parallel corpora using the SRILM toolkit[4] with interpolated Kneser-Ney discounting. Tuning was performed by minimum error rate training (MERT) (Och, 2003), and it was re-run for every experiment.

In the second set of experiments, we used the same morphological analyzers to segment and tag the POS of Japanese and Chinese sentences as in the first set. We further parsed the dependency structures of the Japanese sentences using KNP (Kawahara and Kurohashi, 2006), a lexicalized probabilistic dependency parser, and for the Chinese sentences we used a second-order graph-based parser proposed in (Shen et al., 2012). For decoding, we used the tree-to-tree example-based machine translation framework KyotoEBMT[5] (Richardson et al., 2015) with default options.

We report results on the test set using BLEU-4 score, which was evaluated using the multi-bleu.perl script in Moses based on Juman segmentations. The significance test was performed using the bootstrap resampling method proposed by Koehn (2004).

In Table 7 we compare the performance of three Moses models: In "Character" we used a simple segmentation strategy for the Chinese sentences where we treated each character as a token; in "Original" and "Re-annotated" we segmented the Chinese sentences using the corresponding models described in

[2] $p < 0.05$ in McNemar's test.
[3] http://lotus.kuee.kyoto-u.ac.jp/ASPEC/
[4] http://www.speech.sri.com/projects/srilm
[5] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?KyotoEBMT

the last subsection. The results show, with the underlying machine translation system being the same, the segmenter trained with the original CTB5 failed to support the system to outperform the simple character-based segmentation, while on the other hand the system using the segmenter trained with our re-annotated CTB5 significantly outperformed both "Character"[6] and "Original"[7].

In Table 8 we show the result of the experiment with KyotoEBMT, a tree-to-tree machine translation system which requires unlabeled dependency annotation in the model training. 3,000 sentences with original and re-annotated dependency labels were used for training the parsers in "original" and "re-annotated" settings, respectively. The result shows that, the model "Re-annotated" which used the training set with the proposed annotation, it significantly outperformed[8] the baseline model "Original" by 0.97 point in BLEU-4 score.

5 Conclusion

We have proposed a new annotation approach for Chinese word segmentation, part-of-speech tagging, and dependency labelling. By re-annotating the CTB5 and conducting word segmentation, POS tagging and machine translation experiments, we have demonstrated that this approach has the advantages in achieving higher annotation consistency as well as less data sparsity, compared to the original annotation of CTB5. We couldn't show the comparison in dependency parsing experiments as we currently have only 3,000 annotated sentences; this experiment will be included in our future work.

Reference

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D. Manning. 2009. Discriminative Reordering with Chinese Grammatical Relations Features. In Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation, pages 51-59.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In Proceedings of EMNLP, pages 1–8.

Daisuke Kawahara and Sadao Kurohashi. 2006. A Fully-Lexicalized Probabilistic Model for Japanese Syntactic and Case Structure Analysis. In Proceedings of the Human Language Technology Conference of the NAACL, pages 176–183.

HuiMing Duan, XiaoJing Bai, BaoBao Chan, and ShiWen Yu. 2003. Chinese word segmentation at Peking University. In Proceedings of the second SIGHAN workshop on Chinese language processing, pages 152-155.

ChuRen Huang, KehJiann Chen, FengYi Chen, and LiLi Chang. 1997. Segmentation Standard for Chinese Natural Language Processing. Computational Linguistics and Chinese Language Processing vol. 2, no. 2, August 1997, pages 47-62.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008a. A Cascaded Linear Model for Joint Chinese Word Segmentation and Part-of-speech Tagging. In Proceedings of ACL.

Wenbin Jiang, Haitao Mi, and Qun Liu. 2008b. Word Lattice Reranking for Chinese Word Segmentation and Part-of-speech Tagging. In Proceedings of COLING.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In Proceedings of EMNLP 2004, pages 388-395.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion, Demo and Poster Session, pages 177-180.

[6] $p < 0.5$
[7] $p < 0.1$
[8] $p < 0.1$

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, YiouWang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An Error-Driven Word-Character Hybird Model for Joint Chinese Word Segmentation and POS Tagging. In Proceedings of ACL-IJCNLP, pages 513-521.

Sadao Kurohashi, Toshihisa Nakamura, Yuji Matsumoto, and Nagao Makoto. 1994. Improvements of Japanese Morphological Analyzer JUMAN. In Proceedings of the International Workshop on Sharable Natural Language, pages 22-28.

Marie-Catherine de Marneffe, Bill MacCartney and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In Proceedings of LREC 2006, pages 449-454.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, pages 160-167.

John Richardson, Raj Dabre, Chenhui Chu, Fabien Cromières, Toshiaki Nakazawa, and Sadao Kurohashi. 2015. KyotoEBMT System Description for the 2nd Workshop on Asian Translation. In Proceedings of the 2nd Workshop on Asian Translation, pages 54-60.

Mo Shen, Daisuke Kawahara, and Sadao Kurohashi. 2012. A Reranking Approach for Dependency Parsing with Variable-sized Subtree Features. In Proceedings of 26th Pacific Asia Conference on Language Information and Computing, pages 308-317.

Mo Shen, Hongxiao Liu, Daisuke Kawahara, and Sadao Kurohashi. 2014. Chinese Morphological Analysis with Character-level POS Tagging. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 253–258.

Weiwei Sun. 2011. A Stacked Sub-word Model for Joint Chinese Word Segmentation and Part-of-speech Tagging. In Proceedings of ACL-HLT, pages 1385–1394.

Fie Xia. 2000. The Segmentation Guidelines for the Penn Chinese Treebank (3.0). http://www.cis.upenn.edu/~chinese/segguide.3rd.ch.pdf.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2013. Chinese Parsing Exploiting Characters. In Proceedings of ACL, page 125-134.

Yue Zhang and Stephen Clark. 2010. A Fast Decoder for Joint Word Segmentation and POS-tagging Using a Single Discriminative Model. In Proceedings of EMNLP, pages 843–852.

Attending to Characters in Neural Sequence Labeling Models

Marek Rei
The ALTA Institute
Computer Laboratory
University of Cambridge
United Kingdom
marek.rei@cl.cam.ac.uk

Gamal K.O. Crichton **Sampo Pyysalo**
Language Technology Lab
Dept. of Theoretical & Applied Linguistics
University of Cambridge
United Kingdom
{gkoc2,smp66}@cam.ac.uk

Abstract

Sequence labeling architectures use word embeddings for capturing similarity, but suffer when handling previously unseen or rare words. We investigate character-level extensions to such models and propose a novel architecture for combining alternative word representations. By using an attention mechanism, the model is able to dynamically decide how much information to use from a word- or character-level component. We evaluated different architectures on a range of sequence labeling datasets, and character-level extensions were found to improve performance on every benchmark. In addition, the proposed attention-based architecture delivered the best results even with a smaller number of trainable parameters.

1 Introduction

Many NLP tasks, including named entity recognition (NER), part-of-speech (POS) tagging and shallow parsing can be framed as types of sequence labeling. The development of accurate and efficient sequence labeling models is thereby useful for a wide range of downstream applications. Work in this area has traditionally involved task-specific feature engineering – for example, integrating gazetteers for named entity recognition, or using features from a morphological analyser in POS-tagging. Recent developments in neural architectures and representation learning have opened the door to models that can discover useful features automatically from the data. Such sequence labeling systems are applicable to many tasks, using only the surface text as input, yet are able to achieve competitive results (Collobert et al., 2011; Irsoy and Cardie, 2014).

Current neural models generally make use of word embeddings, which allow them to learn similar representations for semantically or functionally similar words. While this is an important improvement over count-based models, they still have weaknesses that should be addressed. The most obvious problem arises when dealing with out-of-vocabulary (OOV) words – if a token has never been seen before, then it does not have an embedding and the model needs to back-off to a generic OOV representation. Words that have been seen very infrequently have embeddings, but they will likely have low quality due to lack of training data. The approach can also be sub-optimal in terms of parameter usage – for example, certain suffixes indicate more likely POS tags for these words, but this information gets encoded into each individual embedding as opposed to being shared between the whole vocabulary.

In this paper, we construct a task-independent neural network architecture for sequence labeling, and then extend it with two different approaches for integrating character-level information. By operating on individual characters, the model is able to infer representations for previously unseen words and share information about morpheme-level regularities. We propose a novel architecture for combining character-level representations with word embeddings using a gating mechanism, also referred to as *attention*, which allows the model to dynamically decide which source of information to use for each word. In addition, we describe a new objective for model training where the character-level representations are optimised to mimic the current state of word embeddings.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 309–318, Osaka, Japan, December 11-17 2016.

We evaluate the neural models on 8 datasets from the fields of NER, POS-tagging, chunking and error detection in learner texts. Our experiments show that including a character-based component in the sequence labeling model provides substantial performance improvements on all the benchmarks. In addition, the attention-based architecture achieves the best results on all evaluations, while requiring a smaller number of parameters.

2 Bidirectional LSTM for sequence labeling

We first describe a basic word-level neural network for sequence labeling, following the models described by Lample et al. (2016) and Rei and Yannakoudakis (2016), and then propose two alternative methods for incorporating character-level information.

Figure 1 shows the general architecture of the sequence labeling network. The model receives a sequence of tokens $(w_1, ..., w_T)$ as input, and predicts a label corresponding to each of the input tokens. The tokens are first mapped to a distributed vector space, resulting in a sequence of word embeddings $(x_1, ..., x_T)$. Next, the embeddings are given as input to two LSTM (Hochreiter and Schmidhuber, 1997) components moving in opposite directions through the text, creating context-specific representations. The respective forward- and backward-conditioned representations are concatenated for each word position, resulting in representations that are conditioned on the whole sequence:

$$\overrightarrow{h_t} = LSTM(x_t, \overrightarrow{h_{t-1}}) \qquad \overleftarrow{h_t} = LSTM(x_t, \overleftarrow{h_{t+1}}) \qquad h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \tag{1}$$

We include an extra narrow hidden layer on top of the LSTM, which proved to be a useful modification based on development experiments. An additional hidden layer allows the model to detect higher-level feature combinations, while constraining it to be small forces it to focus on more generalisable patterns:

$$d_t = tanh(W_d h_t) \tag{2}$$

where W_d is a weight matrix between the layers, and the size of d_t is intentionally kept small.

Finally, to produce label predictions, we use either a softmax layer or a conditional random field (CRF, Lafferty et al. (2001)). The softmax calculates a normalised probability distribution over all the possible labels for each word:

$$P(y_t = k|d_t) = \frac{e^{W_{o,k} d_t}}{\sum_{\tilde{k} \in K} e^{W_{o,\tilde{k}} d_t}} \tag{3}$$

where $P(y_t = k|d_t)$ is the probability of the label of the t-th word (y_t) being k, K is the set of all possible labels, and $W_{o,k}$ is the k-th row of output weight matrix W_o. To optimise this model, we minimise categorical crossentropy, which is equivalent to minimising the negative log-probability of the correct labels:

$$E = -\sum_{t=1}^{T} log(P(y_t|d_t)) \tag{4}$$

Following Huang et al. (2015), we can also use a CRF as the output layer, which conditions each prediction on the previously predicted label. In this architecture, the last hidden layer is used to predict confidence scores for the word having each of the possible labels. A separate weight matrix is used to learn transition probabilities between different labels, and the Viterbi algorithm is used to find an optimal sequence of weights. Given that y is a sequence of labels $[y_1, ..., y_T]$, then the CRF score for this sequence can be calculated as:

$$s(y) = \sum_{t=1}^{T} A_{t,y_t} + \sum_{t=0}^{T} B_{y_t, y_{t+1}} \tag{5}$$

$$A_{t,y_t} = W_{o,y_t} d_t \tag{6}$$

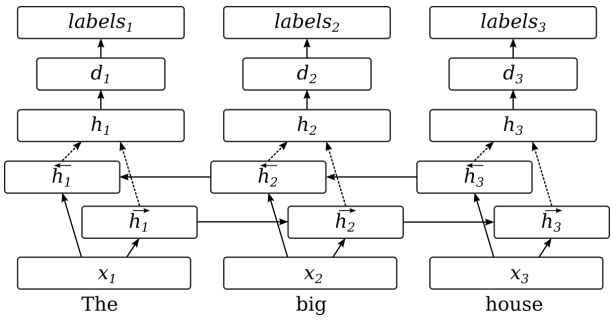


Figure 1: Neural sequence labeling model. Word embeddings are given as input; a bidirectional LSTM produces context-dependent representations; the information is passed through a hidden layer and the output layer. The outputs are either probability distributions for softmax, or confidence scores for CRF.

where A_{t,y_t} shows how confident the network is that the label on the t-th word is y_t. $B_{y_t,y_{t+1}}$ shows the likelihood of transitioning from label y_t to label y_{t+1}, and these values are optimised during training. The output from the model is the sequence of labels with the largest score $s(y)$, which can be found efficiently using the Viterbi algorithm. In order to optimise the CRF model, the loss function maximises the score for the correct label sequence, while minimising the scores for all other sequences:

$$E = -s(y) + log \sum_{\tilde{y} \in \widetilde{Y}} e^{s(\tilde{y})} \tag{7}$$

where $\widetilde{Y}$ is the set of all possible label sequences.

3 Character-level sequence labeling

Distributed embeddings map words into a space where semantically similar words have similar vector representations, allowing the models to generalise better. However, they still treat words as atomic units and ignore any surface- or morphological similarities between different words. By constructing models that operate over individual characters in each word, we can take advantage of these regularities. This can be particularly useful for handling unseen words – for example, if we have never seen the word *cabinets* before, a character-level model could still infer a representation for this word if it has previously seen the word *cabinet* and other words with the suffix *-s*. In contrast, a word-level model can only represent this word with a generic out-of-vocabulary representation, which is shared between all other unseen words.

Research into character-level models is still in fairly early stages, and models that operate exclusively on characters are not yet competitive to word-level models on most tasks. However, instead of fully replacing word embeddings, we are interested in combining the two approaches, thereby allowing the model to take advantage of information at both granularity levels. The general outline of our approach is shown in Figure 2. Each word is broken down into individual characters, these are then mapped to a sequence of character embeddings $(c_1, ..., c_R)$, which are passed through a bidirectional LSTM:

$$\overrightarrow{h_i^*} = LSTM(c_i, \overrightarrow{h_{i-1}^*}) \qquad \overleftarrow{h_i^*} = LSTM(c_i, \overleftarrow{h_{i+1}^*}) \tag{8}$$

We then use the last hidden vectors from each of the LSTM components, concatenate them together, and pass the result through a separate non-linear layer.

$$h^* = [\overrightarrow{h_R^*}; \overleftarrow{h_1^*}] \qquad m = tanh(W_m h^*) \tag{9}$$

where W_m is a weight matrix mapping the concatenated hidden vectors from both LSTMs into a joint word representation m, built from individual characters.

We now have two alternative feature representations for each word – x_t from Section 2 is an embedding learned on the word level, and $m^{(t)}$ is a representation dynamically built from individual characters in

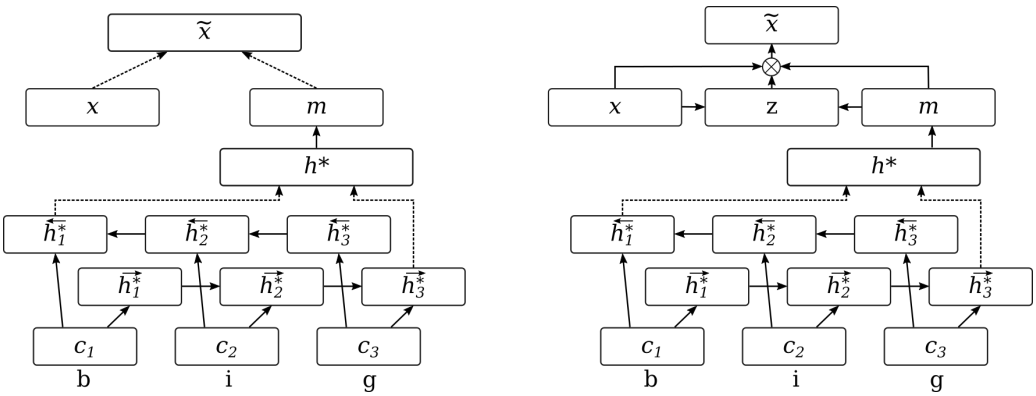


Figure 2: Left: concatenation-based character architecture. Right: attention-based character architecture. The dotted lines indicate vector concatenation.

the t-th word of the input text. Following Lample et al. (2016), one possible approach is to concatenate the two vectors and use this as the new word-level representation for the sequence labeling model:

$$\widetilde{x} = [x; m] \tag{10}$$

This approach, also illustrated in Figure 2, assumes that the word-level and character-level components learn somewhat disjoint information, and it is beneficial to give them separately as input to the sequence labeler.

4 Attention over character features

Alternatively, we can have the word embedding and the character-level component learn the same semantic features for each word. Instead of concatenating them as alternative feature sets, we specifically construct the network so that they would learn the same representations, and then allow the model to decide how to combine the information for each specific word.

We first construct the word representation from characters using the same architecture – a bidirectional LSTM operates over characters, and the last hidden states are used to create vector m for the input word. Instead of concatenating this with the word embedding, the two vectors are added together using a weighted sum, where the weights are predicted by a two-layer network:

$$z = \sigma(W_z^{(3)} tanh(W_z^{(1)} x + W_z^{(2)} m)) \qquad \widetilde{x} = z \cdot x + (1 - z) \cdot m \tag{11}$$

where $W_z^{(1)}$, $W_z^{(2)}$ and $W_z^{(3)}$ are weight matrices for calculating z, and $\sigma()$ is the logistic function with values in the range $[0, 1]$. The vector z has the same dimensions as x or m, acting as the weight between the two vectors. It allows the model to dynamically decide how much information to use from the character-level component or from the word embedding. This decision is done for each feature separately, which adds extra flexiblity – for example, words with regular suffixes can share some character-level features, whereas irregular words can store exceptions into word embeddings. Furthermore, previously unknown words are able to use character-level regularities whenever possible, and are still able to revert to using the generic OOV token when necessary.

The main benefits of character-level modeling are expected to come from improved handling of rare and unseen words, whereas frequent words are likely able to learn high-quality word-level embeddings directly. We would like to take advantage of this, and train the character component to predict these word embeddings. Our attention-based architecture requires the learned features in both word representations to align, and we can add in an extra constraint to encourage this. During training, we add a term to the loss function that optimises the vector m to be similar to the word embedding x:

Name	Task	# labels	# train tokens	# dev tokens	# test tokens
CoNLL00	Chunking	22	158,795	52,932	47,377
CoNLL03	NER	8	203,621	51,362	46,435
PTB-POS	POS	48	912,344	131,768	129,654
FCEPUBLIC	Error det	2	452,833	34,599	41,477
BC2GM	NER	3	355,405	71,042	143,465
CHEMDNER	NER	3	891,948	886,324	766,033
JNLPBA	NER	11	445,090	47,461	101,039
GENIA-POS	POS	42	397,690	52,697	50,556

Table 1: Details for each of the evaluation datasets.

$$\widetilde{E} = E + \sum_{t=1}^{T} g_t(1 - cos(m^{(t)}, x_t)) \qquad g_t = \begin{cases} 0, & \text{if } w_t = OOV \\ 1, & \text{otherwise} \end{cases} \tag{12}$$

Equation 12 maximises the cosine similarity between $m^{(t)}$ and x_t. Importantly, this is done only for words that are not out-of-vocabulary – we want the character-level component to learn from the word embeddings, but this should exclude the OOV embedding, as it is shared between many words. We use g_t to set this cost component to 0 for any OOV tokens.

While the character component learns general regularities that are shared between all the words, individual word embeddings provide a way for the model to store word-specific information and any exceptions. Therefore, while we want the character-based model to shift towards predicting high-quality word embeddings, it is not desireable to optimise the word embeddings towards the character-level representations. This can be achieved by making sure that the optimisation is performed only in one direction; in Theano (Bergstra et al., 2010), the *disconnected_grad* function gives the desired effect.

5 Datasets

We evaluate the sequence labeling models and character architectures on 8 different datasets. Table 1 contains information about the number of labels and dataset sizes for each of them.

- **CoNLL00**: The CoNLL-2000 dataset (Tjong Kim Sang and Buchholz, 2000) is a frequently used benchmark for the task of chunking. Wall Street Journal Sections 15-18 from the Penn Treebank are used for training, and Section 20 as the test data. As there is no official development set, we separated some of the training set for this purpose.

- **CoNLL03**: The CoNLL-2003 corpus (Tjong Kim Sang and De Meulder, 2003) was created for the shared task on language-independent NER. We use the English section of the dataset, containing news stories from the Reuters Corpus[1].

- **PTB-POS**: The Penn Treebank POS-tag corpus (Marcus et al., 1993) contains texts from the Wall Street Journal, annotated for part-of-speech tags. The PTB label set includes 36 main tags and an additional 12 tags covering items such as punctuation.

- **FCEPUBLIC**: The publicly released subset of the First Certificate in English (FCE) dataset contains short essays written by language learners and manual corrections by examiners (Yannakoudakis et al., 2011). We use a version of this corpus converted into a binary error detection task, where each token is labeled as being correct or incorrect in the given context.

- **BC2GM**: The BioCreative II Gene Mention corpus (Smith et al., 2008) consists of 20,000 sentences from biomedical publication abstracts and is annotated for mentions of the names of genes, proteins and related entities using a single NE class.

[1] http://about.reuters.com/researchandstandards/corpus/

	CoNLL00		CoNLL03		PTB-POS		FCEPUBLIC	
	DEV	TEST	DEV	TEST	DEV	TEST	DEV	TEST
Word-based	91.48	91.23	86.89	79.86	96.29	96.42	46.58	41.24
Char concat	92.57	92.35	89.81	83.37	97.20	97.22	46.44	41.27
Char attention	**92.92**	**92.67**	**89.91**	**84.09**	**97.22**	**97.27**	**47.17**	**41.88**

	BC2GM		CHEMDNER		JNLPBA		GENIA-POS	
	DEV	TEST	DEV	TEST	DEV	TEST	DEV	TEST
Word-based	84.07	84.21	78.63	79.74	75.46	70.75	97.55	97.39
Char concat	87.54	87.75	82.80	83.56	76.82	72.24	98.59	98.49
Char attention	**87.98**	**87.99**	**83.75**	**84.53**	**77.38**	**72.70**	**98.67**	**98.60**

Table 2: Comparison of word-based and character-based sequence labeling architectures on 8 datasets. The evaluation measure used for each dataset is specified in Section 6.

- **CHEMDNER**: The BioCreative IV Chemical and Drug (Krallinger et al., 2015) NER corpus consists of 10,000 abstracts annotated for mentions of chemical and drug names using a single class. We make use of the official splits provided by the shared task organizers.

- **JNLPBA**: The JNLPBA corpus (Kim et al., 2004) consists of 2,404 biomedical abstracts and is annotated for mentions of five entity types: CELL LINE, CELL TYPE, DNA, RNA, and PROTEIN. The corpus was derived from GENIA corpus entity annotations for use in the shared task organized in conjuction with the BioNLP 2004 workshop.

- **GENIA-POS**: The GENIA corpus (Ohta et al., 2002) is one of the most widely used resources for biomedical NLP and has a rich set of annotations including parts of speech, phrase structure syntax, entity mentions, and events. Here, we make use of the GENIA POS annotations, which cover 2,000 PubMed abstracts (approx. 20,000 sentences). We use the same 210-document test set as Tsuruoka et al. (2005), and additionally split off a sample of 210 from the remaining documents as a development set.

6 Experiment settings

For data prepocessing, all digits were replaced with the character '0'. Any words that occurred only once in the training data were replaced by the generic OOV token for word embeddings, but were still used in the character-level components. The word embeddings were initialised with publicly available pretrained vectors, created using word2vec (Mikolov et al., 2013), and then fine-tuned during model training. For the general-domain datasets we used 300-dimensional vectors trained on Google News[2]; for the biomedical datasets we used 200-dimensional vectors trained on PubMed and PMC[3]. The embeddings for characters were set to length 50 and initialised randomly.

The LSTM layer size was set to 200 in each direction for both word- and character-level components. The hidden layer d has size 50, and the combined representation m has the same length as the word embeddings. CRF was used as the output layer for all the experiments – we found that this gave most benefits to tasks with larger numbers of possible labels. Parameters were optimised using AdaDelta (Zeiler, 2012) with default learning rate 1.0 and sentences were grouped into batches of size 64. Performance on the development set was measured at every epoch and training was stopped if performance had not improved for 7 epochs; the best-performing model on the development set was then used for evalua-

[2]https://code.google.com/archive/p/word2vec/
[3]http://bio.nlplab.org/

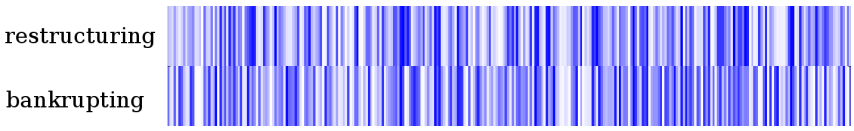


Figure 3: Visualisation of attention values for two words, trained on the PTB-POS dataset. Darker blue indicates features with higher weights for the character-level representation. *Restructuring* was present in the vocabulary, while *bankrupting* is an OOV.

tion on the test set. In order to avoid any outlier results due to randomness in the model initialisation, we trained each configuration with 10 different random seeds and present here the averaged results.

When evaluating on each dataset, we report the measures established in previous work. Token-level accuracy is used for PTB-POS and GENIA-POS; $F_{0.5}$ score over the erroneous words for FCEPUBLIC; the official evaluation script for BC2GM which allows for alternative correct entity spans; and micro-averaged mention-level F_1 score for the remaining datasets.

7 Results

While optimising the hyperparameters for each dataset separately would likely improve individual performance, we conduct more controlled experiments on a task-independent model. Therefore, we use the same hyperparameters from Section 6 on all datasets, and the development set is only used for the stopping condition. With these experiments, we wish to determine 1) on which sequence labeling tasks do character-based models offer an advantange, and 2) which character-based architecture performs better.

Results for the different model architectures on all 8 datasets are shown in Table 2. As can be seen, including a character-based component in the sequence labeling architecture improves performance on every benchmark. The NER datasets have the largest absolute improvement – the model is able to learn character-level patterns for names, and also improve the handling of any previously unseen tokens.

Compared to concatenating the word- and character-level representations, the attention-based character model outperforms the former on all evaluations. The mechanism for dynamically deciding how much character-level information to use allows the model to better handle individual word representations, giving it an advantage in the experiments. Visualisation of the attention values in Figure 3 shows that the model is actively using character-based features, and the attention areas vary between different words.

The results of this general tagging architecture are competitive, even when compared to previous work using hand-crafted features. The network achieves 97.27% on PTB-POS compared to 97.55% by Huang et al. (2015), and 72.70% on JNLPBA compared to 72.55% by Zhou and Su (2004). In some cases, we are also able to beat the previous best results – 87.99% on BC2GM compared to 87.48% by Campos et al. (2015), and 41.88% on FCEPUBLIC compared to 41.1% by Rei and Yannakoudakis (2016). Lample et al. (2016) report a considerably higher result of 90.94% on CoNLL03, indicating that the chosen hyperparameters for the baseline system are suboptimal for this specific task. Compared to the experiments presented here, their model used the IOBES tagging scheme instead of the original IOB, and embeddings pretrained with a more specialised method that accounts for word order.

It is important to also compare the parameter counts of alternative neural architectures, as this shows their learning capacity and indicates their time requirements in practice. Table 3 contains the parameter counts on three representative datasets. While keeping the model hyperparameters constant, the character-level models require additional parameters for the character composition and character embeddings. However, the attention-based model uses fewer parameters compared to the concatenation approach. When the two representations are concatenated, the overall word representation size is increased, which in turn increases the number of parameters required for the word-level bidirectional LSTM. Therefore, the attention-based character architecture achieves improved results even with a smaller parameter footprint.

	CoNLL03		FCEPUBLIC		CHEMDNER	
	# total	# noemb	# total	# noemb	# total	# noemb
Word-based	4,507,658	1,230,158	2,972,052	1,230,252	5,862,878	1,070,278
Char concat	4,987,658	1,710,158	3,452,052	1,710,252	6,182,878	1,390,278
Char attention	4,687,958	1,410,458	3,152,352	1,410,552	5,943,078	1,150,478

Table 3: Comparison of trainable parameters in each of the neural model architectures. *# total* shows the total number of parameters; *# noemb* shows the parameter count excluding word embeddings, as only a small fraction of the embeddings are utilised at every iteration.

8 Related work

There is a wide range of previous work on constructing and optimising neural architectures applicable to sequence labeling. Collobert et al. (2011) described one of the first task-independent neural tagging models using convolutional neural networks. They were able to achieve good results on POS tagging, chunking, NER and semantic role labeling, without relying on hand-engineered features. Irsoy and Cardie (2014) experimented with multi-layer bidirectional Elman-style recurrent networks, and found that the deep models outperformed conditional random fields on the task of opinion mining. Huang et al. (2015) described a bidirectional LSTM model with a CRF layer, which included hand-crafted features specialised for the task of named entity recognition. Rei and Yannakoudakis (2016) evaluated a range of neural architectures, including convolutional and recurrent networks, on the task of error detection in learner writing. The word-level sequence labeling model described in this paper follows the previous work, combining useful design choices from each of them. In addition, we extended the model with two alternative character-level architectures, and evaluated its performance on 8 different datasets.

Character-level models have the potential of capturing morpheme patterns, thereby improving generalisation on both frequent and unseen words. In recent years, there has been an increase in research into these models, resulting in several interesting applications. Ling et al. (2015b) described a character-level neural model for machine translation, performing both encoding and decoding on individual characters. Kim et al. (2016) implemented a language model where encoding is performed by a convolutional network and LSTM over characters, whereas predictions are given on the word-level. Cao and Rei (2016) proposed a method for learning both word embeddings and morphological segmentation with a bidirectional recurrent network over characters. There is also research on performing parsing (Ballesteros et al., 2015) and text classification (Zhang et al., 2015) with character-level neural models. Ling et al. (2015a) proposed a neural architecture that replaces word embeddings with dynamically-constructed character-based representations. We applied a similar method for operating over characters, but combined them with word embeddings instead of replacing them, as this allows the model to benefit from both approaches. Lample et al. (2016) described a model where the character-level representation is combined with word embeddings through concatenation. In this work, we proposed an alternative architecture, where the representations are combined using an attention mechanism, and evaluated both approaches on a range of tasks and datasets. Recently, Miyamoto and Cho (2016) have also described a related method for the task of language modelling, combining characters and word embeddings using gating.

9 Conclusion

Developments in neural network research allow for model architectures that work well on a wide range of sequence labeling datasets without requiring hand-crafted data. While word-level representation learning is a powerful tool for automatically discovering useful features, these models still come with certain weaknesses – rare words have low-quality representations, previously unseen words cannot be modeled at all, and morpheme-level information is not shared with the whole vocabulary.

In this paper, we investigated character-level model components for a sequence labeling architecture, which allow the system to learn useful patterns from sub-word units. In addition to a bidirectional LSTM operating over words, a separate bidirectional LSTM is used to construct word representations from

individual characters. We proposed a novel architecture for combining the character-based representation with the word embedding by using an attention mechanism, allowing the model to dynamically choose which information to use from each information source. In addition, the character-level composition function is augmented with a novel training objective, optimising it to predict representations that are similar to the word embeddings in the model.

The evaluation was performed on 8 different sequence labeling datasets, covering a range of tasks and domains. We found that incorporating character-level information into the model improved performance on every benchmark, indicating that capturing features regarding characters and morphmes is indeed useful in a general-purpose tagging system. In addition, the attention-based model for combining character representations outperformed the concatenation method used in previous work in all evaluations. Even though the proposed method requires fewer parameters, the added ability of controlling how much character-level information is used for each word has led to improved performance on a range of different tasks.

References

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved Transition-Based Parsing by Modeling Characters instead of Words with LSTMs. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

James Bergstra, Olivier Breuleux, Frederic Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math compiler in Python. *Proceedings of the Python for Scientific Computing Conference (SciPy)*.

David Campos, Sergio Matos, and Jose L. Oliveira. 2015. A document processing pipeline for annotating chemical entities in scientific documents. *Journal of Cheminformatics*, 7.

Kris Cao and Marek Rei. 2016. A Joint Model for Word Embedding and Word Morphology. In *Proceedings of the 1st Workshop on Representation Learning for NLP (RepL4NLP-2016)*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural Computation*, 9.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv:1508.01991*.

Ozan Irsoy and Claire Cardie. 2014. Opinion Mining with Deep Recurrent Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the Bio-entity Recognition Task at JNLPBA. *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-Aware Neural Language Models. *In Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI16)*.

Martin Krallinger, Florian Leitner, Obdulia Rabal, Miguel Vazquez, Julen Oyarzabal, and Alfonso Valencia. 2015. CHEMDNER: The drugs and chemical names extraction challenge. *Journal of Cheminformatics*, 7(Suppl 1).

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of NAACL-HLT 2016*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015a. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015b. Character-based Neural Machine Translation. *arXiv preprint arXiv:1511.04586*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19.

Tomáš Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated Word-Character Recurrent Language Model. *arXiv preprint arXiv:1606.01700*.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The GENIA corpus: An annotated research abstract corpus in molecular biology domain. *Proceedings of the second international conference on Human Language Technology Research*.

Marek Rei and Helen Yannakoudakis. 2016. Compositional Sequence Labeling Models for Error Detection in Learner Writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Larry Smith, Lorraine K Tanabe, Rie Johnson nee Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M Friedrich, Kuzman Ganchev, Manabu Torii, Hongfang Liu, Barry Haddow, Craig A Struble, Richard J Povinelli, Andreas Vlachos, William A Baumgartner, Lawrence Hunter, Bob Carpenter, Richard Tzong-Han Tsai, Hong-Jie Dai, Feng Liu, Yifei Chen, Chengjie Sun, Sophia Katrenko, Pieter Adriaans, Christian Blaschke, Rafael Torres, Mariana Neves, Preslav Nakov, Anna Divoli, Manuel Maña-López, Jacinto Mata, and W John Wilbur. 2008. Overview of BioCreative II gene mention recognition. *Genome biology*, 9 Suppl 2.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning*, 7.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*.

Yoshimasa Tsuruoka, Yuka Tateishi, Jin Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun'ichi Tsujii. 2005. Developing a robust part-of-speech tagger for biomedical text. In *Proceedings of Panhellenic Conference on Informatics*.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems*.

GuoDong Zhou and Jian Su. 2004. Exploring Deep Knowledge Resources in Biomedical Name Recognition. *Workshop on Natural Language Processing in Biomedicine and Its Applications at COLING*.

A Word Labelling Approach to Thai Sentence Boundary Detection and POS Tagging

Nina Zhou
Institute for Infocomm Research
zhoun@i2r.a-star.edu.sg

Aw AiTi
Institute for Infocomm Research
aaiti@i2r.a-star.edu.sg

Nattadaporn Lertcheva
Institute for Infocomm Research
lertchevan@i2r.a-star.edu.sg

Wang Xuangcong
Institute for Infocomm Research
wangxc@i2r.a-star.edu.sg

Abstract

Previous studies on Thai Sentence Boundary Detection (SBD) mostly assumed a sentence ends at a space and formulated the task SBD as a disambiguation problem, which classified a space either as an indicator for Sentence Boundary (SB) or non-Sentence Boundary (nSB). In this paper, we propose a word labelling approach which treats the space character as a normal word, and detects SB between any two words. This removes the restriction for SB to be occurred only at spaces and makes our system more robust for modern Thai writing. It is because in modern Thai writing, the space is not consistently used to indicate SB. As syntactic information contributes to better SBD, we further propose a joint Part-Of-Speech (POS) tagging and SBD framework based on Factorial Conditional Random Field (FCRF) model. We compare the performance of our proposed approach with reported methods on ORCHID corpus. We also performed experiments of FCRF model on the TaLAPi corpus. The results show that the word labelling approach has better performance than previous space-based classification approaches and FCRF joint model outperforms LCRF model in terms of SBD in all experiments.

1 Introduction

Sentence Boundary Detection (SBD) is a fundamental task for many Natural Language Processing (NLP) and analysis tasks, including POS tagging, syntactic, semantic, and discourse parsing, parallel text alignment, and machine translation (Gillick, 2009). Most research on SBD focus on languages that already have a well-defined concept of what a sentence is, typically indicated by sentence-end markers like full-stops, question marks, or other punctuations. However, as we study more contexts of language use (e.g. speech output which lacks punctuations) as well as look at many more different languages, the assumption of clearly-punctuated sentence boundary becomes less valid. One such language is Thai.

In prior research on Thai, the space character has been regarded as a very important element in Thai SBD (Pradit *et al.*, 2000; Paisarn *et al.*, 2001; Glenn *et al.*, 2010). These regard that space characters are always present between sentences. However, in actual fact, as prescribed by Thai linguistic authorities (www.royin.go.th) as well as what can be observed in real texts, spaces do exist in Thai texts not only in such sentence-end contexts. There is some pressure from linguistic authorities (Wathabunditkul, 2003) to set orthographic standards in Thai, prescribing the use of spaces in the context of certain words, following the rules of the Thai Royal Institute Dictionary[1]. Examples of these rules include: using of the space before and after an interjection or an onomatopoeiac word โอ๊ย (Ouch!), อุ๊ย (ow!); before conjunctions และ (*and*), หรือ (*or*), and แต่ (*but*); before and after a numeric expression: มีนักเรียน 20 คน (have 20 students), เวลา 10.00 น. (time 10.00 a.m.). Unfortunately, the rules are not strictly followed in practice and the use of spaces between words, phrases, clauses and sentences vary across different users of the Thai language. According to TaLAPi (Aw et al. 2014), a news domain corpus, it has about 23% sentences ending without a space character. One example of the Thai text from TaLAPi corpus is

[1] https://en.wikipedia.org/wiki/Royal_Institute_Dictionary

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 319–327, Osaka, Japan, December 11-17 2016.

shown in Figure 1, in which the space character is used within a sentence, but not as a sentence-end indicator.

In view of this complexity of spaces in Thai in light of the SBD task, we propose a word-based labelling approach which regards Thai SBD as a word labelling problem instead of a space classification problem. The approach treats the space as a normal word and labels each word as SB or nSB (non-Sentence Boundary). Figure 2 illustrates the space-based classification approach versus the word-based labelling approach.

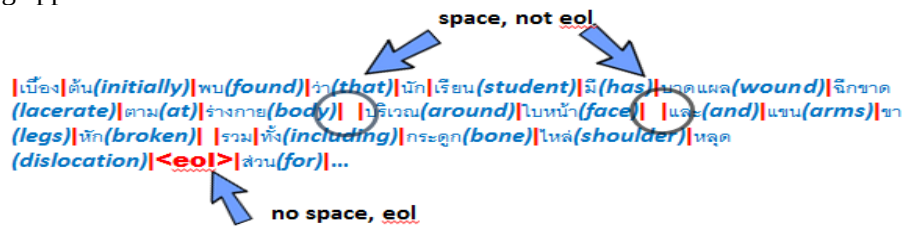


Figure 1. Example of a written Thai text in which there are two space characters within the first sentence, but there is no space character at the end of the sentence, i.e., at highlighted <eol>". "eol" refers to end-of-line.

The proposed word labelling approach formulates SBD as a typical sequence labelling task, i.e., labelling each word including spaces as a SB or nSB. It is tested on ORCHID corpus and demonstrates higher accuracy on SB than previous methods (Pradit *et al.*, 2000; Paisarn *et al.*, 2001; Glenn *et al.*, 2010). Furthermore, the contribution of POS in this task is investigated and a Joint framework for POS tagging and SBD is formulated. The results on TaLAPi corpus show that POS information can improve the accuracy of SBD, for both the sequential task of POS tagging followed by SBD and the proposed joint framework. Moreover, in the joint framework, we propose a two-layer classification for POS tagging, which is called as "2-step" Joint approach in the following paper. For comparison, the joint approach in which POS tagging realized in one step is called as "1-step" Joint approach. The proposed "2-step" Joint approach runs considerable faster and achieves similar performance when compared with the Cascade approach and "1-step" Joint approach of POS tagging and SBD. By adding enhanced features, the "2-step" Joint approach yields better SBD accuracy and comparable POS tagging accuracy.

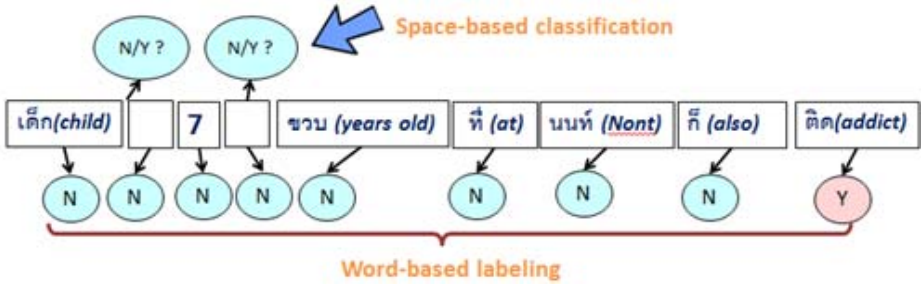


Figure 2. Space-based SBD vs word-based labelling SBD. Space-based SBD detects spaces and assigns Y (SB) or N (nSB) to each space. Word-based labelling assigns Y(SB) or N(nSB) to every word. In this case, the space character is considered as a word.

The rest of the paper is organized as follows. Section 2 reviews the previous studies on Thai SBD. Section 3 describes the proposed word labelling framework and the approaches. Section 4 compares the performance between the proposed word-based methods and reported space-based methods on ORCHID corpus (Sornlertlamvanich *et al.*, 1997), and also studied the different frameworks of word-based approaches. Section 5 concludes the paper.

2 Previous Studies

There have been limited studies carried out in Thai SBD over the past twenty years. Longchupole (1995) presented a method to segment a paragraph into small units and then used verbs to estimate the number of sentences. That was a grammatical rule based approach to extract sentences from paragraphs. The reported SBD accuracy was 81.18% (Longchupole, 1995). Pradit *et al.* (2000) applied the statistical POS tagging technique (Brants, 2000) on the detection of SB. They considered SB and non-

SB as POS tags and distinguished SB from other POS tags based on a trigram model. Their method yielded an accuracy of 85.26% on ORCHID corpus. Paisarn *et al.* (2001) utilized the Winnow algorithm to extract features from the context around the target space. The Winnow functioned like a neuron network model where a few nodes were connected to a target node. Each node examined only two features for simplicity. In total, there were 10 features including words around target space and their POS information. The space-correct accuracy for the Winnow on ORCHID was 89.13%. Later, Glenn *et al.* (2010) proposed to use maximum entropy classifier to distinguish each space as SB or non-SB and their results were shown to be consistent with the Winnow (Paisam *et al*., 2001).

Nearly all Thai SBD studies are based on the assumption that there is a space at the position of the SB. While we have shown in the Introduction part that sentence break is not always indicated by a space, especially in modern Thai writing. That inspired us to propose the word-based approach to consider a space as a word and treats SBD as a word labelling task instead of a space disambiguation problem

The word-based approach is further enhanced to label POS tags and SB jointly using joint inferencing. The advantages of this approach are: 1) it relies on contexts instead of spaces to detect SB, 2) it solves SBD and POS tagging jointly to relax the dependency of POS tagging for SBD, 3) it demonstrates higher accuracy on SBD than previous methods (Pradit *et al.*, 2000; Paisarn *et al.*, 2001; Glenn *et al*., 2010).

3 The Models

CRFs (Lafferty *et al.*, 2001; Sutton *et al*., 2011) have demonstrated their strengths of sequence labelling in NLP tasks (McCallum *et al.*, 2003; Liu *et al*., 2005; Sun *et al.*, 2011). They rely on the capacity to capture the sequence's observation O_n {i=1,2…n} (abbreviated as O) and at the same time the local dependency L_i {i=1,2,…n} (abbreviated as L) among nodes in the sequence (see Figure 3 for the example of linear-chain CRF (LCRF) (Sutton et al., 2011)). Conditioned on observations O, dependencies of L form the chain. In the model, the probability of labelling an observed input O with a label sequence L is defined by a conditional probability as in Equation (1):

$$p_\lambda(L \mid O) = \frac{1}{Z(O)} \exp\left(\sum_t \sum_k \lambda_k f_k(O, L, t)\right) \quad (1)$$

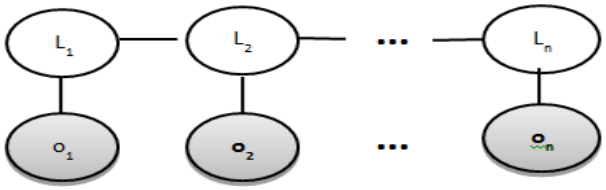


Figure 3. Linear-chain graph CRF (LCRF)

where $\{f_k\}$ is a set of feature functions defined over the observation O and label sequence L at each position t, together with the set of corresponding weights $\{\lambda_k\}$; z(O) is a normalization factor.

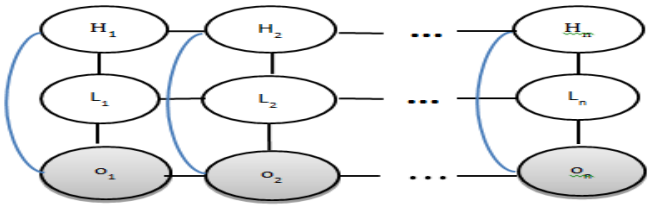


Figure 4. Two-layer Factorial CRF (FCRF)

Dynamic CRF (DCRF) (Sutton *et al.,* 2004; Sutton *et al*., 2011) is a generalization of LCRF, which supports any arbitrary structure graph. It is formally defined as in Equation (2):

$$p_\lambda(L \mid O) = \frac{1}{Z(O)} \exp\left(\sum_t \sum_{c \in C} \sum_k \lambda_k f_k(O, L_{(c,t)}, t)\right) \quad (2)$$

where C is a set of cliques indices which connect the nodes in a sequence in a single layer or among different layers. As a special case of DCRF, Factorial CRF (FCRF) model allows multiple layers' la-

belling simultaneously for a given sequence. The graphical illustration of two-layer FCRF is shown in Figure 4 where *H* indicates the 1st layer labels and *L* indicates the 2nd layer labels. *O* is the observation sequence. Through the connections between different layers of labels and the given observation, joint conditional distributions of the labels are learnt.

3.1 Isolated and Cascade SB and POS Tagging

We use LCRF for the single task of SB detection or POS tagging. In this scenario, as Thai SB is to detect word sequence and find the end of each sentence, we consider this to be similar to a sentence-end punctuation prediction task with only two labels. Words are labelled with SB if they are at the beginning of a sentence otherwise, they will be labelled as nSB. For POS tagging, we use all the 35 sub-categories as described in Aw et al. (2014).

Feature Template	Window Size
w_0	3 or 5
$w_{-1}+w_0$	3 or 5
$w_{-1}+w_0+w_1$	3
$wtype_0$	3 or 5
$wtype_{-1}+wtype_0$	3 or 5
$wtype_{-1}+wtype_0+wtype_1$	3

Table 1. The feature template for Isolated SBD and POS tagging. Window size 3 is used for Isolated SBD and POS tagging. Window size 5 is used in "2-step" Joint model (vii and viii) in Table 6.

Considering POS tagging has much more labels to recognize than SBD, it will increase the memory use for system training, therefore the number of the features and feature template have to be carefully selected. It is essential to use a simple feature set, as shown in Table 1, to make a comparison between Isolated models, Cascade models and the Joint models. It is important for "1-step" Joint model as more features make the process run extremely slow. In Table 1, w_i refers to the word at the i^{th} position relative to the current node; window size is the maximum span of words centered at the current word that the template covers, e.g., $w_{-1}+w_0$ with a window size of 3 refers to $w_{-2}+w_{-1}$, $w_{-1}+w_0$, and w_0+w_1; $wtype_i$ indicates the word type at the i^{th} position relative to the current node; In total, five word types are defined, i.e., English, Thai, punctuation, digits and spaces, for the data used in our experiments.

Feature Template	Window size
pos_0	3 or 5
$pos_{-1}+pos_0$	3
$pos_{-1}+pos_0+pos_1$	3

Table 2. The additional feature template for Cascade models, besides the feature templates in Table 1. Window size 5 is used in Cascade model (iv) in Table 6.

As POS tag provides additional syntactic and some semantic information to the word, they are utilized as additional features to the Cascade approach for detecting the sentence boundary. Besides the features listed in Table 1, more POS features listed in Table 2 are used in the Cascade models.

	Original POS	Pseudo POS	Total No.		Original POS	Pseudo POS	Total No.
1	NN,NR,PPER,PINT, PDEM	NPs	104271	7	CL	CL	5747
2	REFX	REFX	1357	8	OD,CD	OCD	8453
3	DPER,DINT,DDEM, PDT	DPs	7267	9	FXN, FXG, FXAV, FXAJ	FXs	13887
4	JJA, JJV	JJs	14335	10	P, COMP, CNJ	PCs	50301
5	VV, VA, AUX	VVs	72769	11	FWN,FWV,FWA,FWX	FWs	24
6	ADV,NEG	ADs	12275	12	PAR, PU, IJ, X	Os	6270

Table 3. The mapping from original POS to Pseudo POS tags

3.1.1. "1-step and "2-step" Joint Models

The joint model realizes the 2-layer labelling of one sequence using FCRF. We consider the first layer as labels of SBD, and the second layer as labels of POS tagging (see Table 3). However, due to the large number of POS tags, combining the feature templates of both tasks increases the search space tremendously and has a large impact on the processing speed. To address this problem, we propose a "2-step" Joint-model in which we first predict 12 top categories of the POS tags as classified in (Aw *et al.*, 2014) and then restore the pseudo POS tags back to the original POS tags (see Figure 5). On the other side, the "1-step" Joint model uses all the 35 POS tags to realize POS tagging in the 2nd layer of FCRF.

To train the "2-step" Joint model, all train data are labelled with two SB labels (i.e., SB and nSB) and 12 pseudo POS tags. The 12 pseudo POS tags are obtained by combining similar POS tags into one category as illustrated in Table 3. The Original POS column lists the original 35 POS tags and the Pseudo POS column lists the corresponding 12 pseudo POS tags. To restore the pseudo POS tags back to the original tags, we train different LCRF models for each pseudo POS tag. As no restoration is required for "CL" and "REFX", a total of 10 LCRF models are built to restore the original POS tags. The diagram of the proposed "2-step" Joint model is shown as follows (Figure 5).

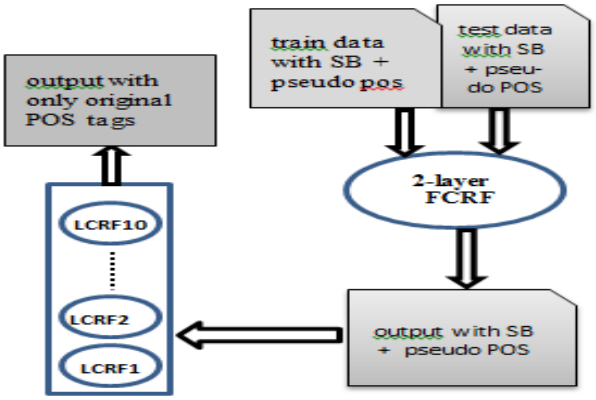


Figure 5. The proposed "2-step" Joint model for Thai SBD and POS Tagging based on two-layer FCRF and the LCRF

For fair comparison between Isolated and Joint models, we used the same feature templates in Table 1 in two of the "2-step" Joint models, i.e., (v) and (vi) in Table 6. Since the "2-step" Joint model run much faster than "1-step" Joint, more features can be added. As in Table 4 shown, name entity recognition (NER) information was added to improve the performance of the "2-step" joint models besides the feature template in Table 1.

Feature Template	Window Size
NER_0	3
NER_{-1}+NER_0	3

Table 4. The enhanced feature template for "2-step" Joint model (viii) in Table 6

4 Experimentation

4.1 Data Preparation

Our experiments were performed on the ORCHID corpus (Sornlertlamvanich *et al.*, 1997) and the TaLAPi corpus (Aw *et al.*, 2014).

The processing of the ORCHID corpus follows the work of Sornlertlamvanich *et al.* (1997) to remove all comments and concatenate all sentences and paragraphs. Different from the previous experiments, we did not insert a space at the end of sentence if it was not originally present. As such, the percentage of sentences ending without a space was almost 100% for the ORCHID corpus used in our experiment. We portioned the ORCHID data into 10 parts with equal size and used 10 fold cross validation for evaluation.

The experiments on TaLAPi corpus were performed only on the news domain which was annotated with word segmentation, POS tags and name entities. It had 3633 paragraphs, 10,478 sentences and

311,637 words. We split 80% for training and 20% for testing. During the splitting, we tried to balance the distribution of spaces and POS tags. Thus in the training data, we have a total of 282,678 words, of which 8,034 words (2.842%) are SB and 274,644 words are nSB. In the test data we have 2,091 (2.836%) SB and 71,635 nSB.

4.2 Experimental results

The GRMM toolkit (Sutton, 2006) was used in our experiments to build the 2-layer FCRF models and one layer LCRF models. To demonstrate the proposed methods, we performed 5 different experiments as follows:

Orchid Corpus

i. Isolated LCRF model to detect SBD using POS information to make it comparable with reported work.

TaLAPi Corpus

ii. Isolated LCRF model for POS tagging and SBD without POS information for SBD.
iii. Cascade LCRF model on SB utilizing same feature as (i) and POS information with different feature templates.
iv. "1-step" Joint model using same features as (ii)
v. "2-step" Joint model using same features as (ii) and with additional features and different feature configurations.

	POS-trigram(%)	**Winnow(%)**	**ME(%)**	**our work(%)**
sb-precision	74.35	92.69	86.21	93.64
sb-recall	79.82	77.27	83.50	89.84
sb-fscore	76.98	84.28	84.83	91.70
nsb-precision	90.27	91.48	93.18	99.27
nsb-recall	87.18	97.56	94.41	99.56
nsb-fscore	88.70	94.42	93.79	99.41
space correct	85.26	89.13	91.19	95.91

Table 5. Comparison of our word-labelling approach based on LCRF (last column) with previous studies on ORCHID corpus. POS-trigram (Pradit *et al*., 2000); Winnow (Paisarn *et al*., 2001); ME (*Glenn et al.*, 2010). Space correct =(#correct sb+#correct nsb)/(total # of space tokens). '#' indicate the number of items followed.

In the ORCHID corpus experiment, we used the features described in Table 1 and Table 2. Table 5 shows the result of the word-labelling approach and its comparison with reported methods. Compared to the reported results (Glenn *et al.*, 2010), our word-labelling approach yielded consistent improvement on precision, recall, F-score for both SB and non-SB and also "space correct". Our SB precision is 1% higher than Winnow method and our recall is 6.3% higher than ME method. The F-score is 7% higher than Winnow and ME. As mentioned in 4.1, not all sentence boundaries in ORCHID are indicated by space. To have a fair comparison, we consider all sentence boundaries as a "space" when calculating "space correct" (Glenn *et al.*, 2010). In Table 5, the short form "sb" and "nsb" refers to the sentence break and non-sentence break respectively.

For the experiments on TaLAPi corpus, we study the performance in Isolated, Cascade and Joint model. The same experiment can be run on ORCHID corpus, but due to time and space limitation, we only show the results of the experiments on TaLAPi corpus in Table 6.

Comparing Cascade with Isolated Model

All Cascade models have higher F-score than the Isolated model. The best F-score of the Cascade model is 67.29% when we used 18 features in the experiment (iv). The experiment affirms that POS information is helpful in sentence boundary detection.

Comparing Isolated, Cascade with Joint Model

With the same set of features as in (i), "1-step" Joint (v) yields 3% increase on recall and 2% increase on F-score for SBD when compared to the Isolated model in (i). Comparing (vi) with (i), a similar in-

crease in accuracy for SBD, with the same features, is observed. These results demonstrate that SBD can benefit from the other layer's label information, i.e., POS tagging labels, in the Joint model (v). When compared to Cascade models, "1-step" Joint shows comparable SB detection performance with the Cascade model (iii) that uses additional 3-gram POS features. By enhancing the feature set for SB detection in the Cascade model (iv), we yielded 1% increase on F-score when compared to the Cascade model (iii).

	Description	**POS**	**SBD**		
		Accuracy	**Precision**	**Recall**	**F-score**
i	Isolated without pos information for SB	94.24	82.67	53.37	64.86
ii	Cascade with same features as (i) and 1-gram POS		81.93	54.85	65.71
iii	Cascade with same features as (i) and 3-gram POS		80.26	56.38	66.24
iv	Cascade with same features as (iii) and window size of 5		79.12	58.53	67.29
v	"1-step" Joint with same features as (i)	94.64	81.72	56.24	66.63
vi	"2-step" Joint with same features as (i)	95.46	82.29	55.57	66.34
		94.49			
vii	"2-step" Joint with same features as (vi) and window size of 5	95.63	80.52	58.29	67.62
		94.99			
viii	"2-step" Joint with same features as (vii) and NER	95.64	80.36	60.26	68.87
		94.99			

Table 6. Comparison of our methods based on FCRF and LCRF on TaLAPi corpus.

Comparing "1-step" with "2-step"

While the Cascade models and "1-step" Joint model was limited by the running speed due to the number of POS tags, the "2-step" Joint model was therefore proposed to improve the running speed and not degrade the accuracy. With the same set of features, the "2-step" Joint (vi) run much faster than "1-step" Joint (v), while yielded almost the same SBD F-score as (v). The run time comparison can be found in Table 7. Experiments were run on Intel(R) Xeon(R) 8 core processor E5-2667 V2 3.30GHz, 25M cache with multi-thread 16.

Different models	**Train time (s)**	**Test time (s)**	**All process (**hr)
Isolated SB (i)	389.1	4.012	0.11
Isolated POS (i)	33230.8	98.2	9.26
Cascade (ii)	33938.2	101.3	9.46
Cascade (iii)	33992.7	102.4	9.48
Cascade (iv)	34181.3	103.8	9.54
"1-step" Joint (v)	41009.8	125.79	11.43
"2-step" Joint (vi)	12895.7	60.147	3.61
"2-step" Joint (vii)	16543.8	74.065	4.62
"2-step" Joint (viii)	18210.2	76.080	5.08

Table 7. Comparison of speed among different methods (test time does not include the serializing time).

The "2-step" Joint (vi) reduces more than half of the running time, compared to "1-step" Joint (v). This decrease in processing time enables us to include more feature set to further improve the performance of SBD in the "2-step" Joint model. By increasing window size from 3 to 5 (i.e., from (vi) to (vii)), (vii) yields 1.3% increase on F-score for SBD, compared to (vi). To further improve the performance, we added NER information with different grams on top of experiment (vii) and found that NER information with unigram (i.e., NER_0) and bigram (i.e., $NER_{-1}+NER_0$) improves the performance, i.e., (viii) shown in Table 6. Undoubtedly, with increased features, the running time of "2-step" Joint model (viii) is more than (vi) and (vii), but it is still faster than the "1-step" Joint model (v). More importantly, it achieved 2% increase on F-score for SBD. Compared to Cascade model (iv), it saved 50% time and achieved 1.6% increase on F-score for SBD.

5 Conclusion

In this paper, we have demonstrated for the first time a word-based labelling approach to Thai SBD. The word-based labelling approach achieved very good performance compared to reported results on ORCHID data. The Cascade model is to use evaluated POS information as features to help SB detection. Higher accuracy in the POS information will yield better accuracy in the Thai SBD. In fact, we also used manually annotated POS tags in SB detection, and it yielded better accuracy, i.e., 79.31% in precision, 62.70% in recall and 70.03% in F-score, compared to the Cascade approach (iv).

Different from Cascade models, Joint models are supposed to make SBD benefit from POS tagging labels in the second layer. Different features are tried in our experiments. Additional features do not always yield better accuracy. For example, when we use more features, e.g., "w_{-1}+w_1" and "$wtype_{-1}$+$wtype_1$, on the top of "2-step" Joint (viii), it does not improve the performance. We noticed that the pseudo-POS tagging performance was not improved in the same way as SBD when more features were added. Besides, more experiments will be explored in the future to see how word boundary information, POS and sentence boundary information affect each other.

In this paper, we demonstrated for the first time a word-based labelling approach to Thai SBD. The word-based labelling approach was proposed to leverage LCRF to do sequence labelling which achieved very good performance compared to reported results on ORCHID data. Furthermore, the performance of SBD with the help of POS tagging was investigated on the corpus TaLAPi. Cascade models and Joint models were compared and the "2-step" Joint POS tagging with SB detection was proposed. This proposed model saved more than half of the time, while obtaining almost the same accuracy for SBD as "1-step" Joint model, when using the same feature set. With increased speed, more features were therefore used to improve SBD and yields comparable POS tagging performance.

Reference

AiTi Aw, Sharifah Mahani Aljunied, Nattadaporn Lertcheva and Sasiwimon Kalunsima. 2014. TaLAPi – A Thai Linguistically Annotated Corpus for Language Processing, LREC, 125-132.

Thorsten Brants. 2000. TnT-A Statistical Part-of-Speech Tagger, ANLP, 224-231.

Paisarn Charoen Pornsawat and Virach Sorlertlamvanich. 2001. Automatic Sentence Break Disambiguation for Thai. ICCPOL.

Dan Gillick. 2009. Sentence Boundary Detection and the Problem with the U.S., Proceedings of NAACL HLT, 241-244.

Evang, Kilian and Basile, Valerio and Chrupala, Grzegorz and Bos, Johan. 2013. Elephant: Sequence Labelling for Word and Sentence Segmentation., EMNLP, 1422--1426

J. Laferty, Andrew McCallum and F. C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labelling Sequence Data. In ICML Proceedings of the Eighteenth International Conference on Machine Learning.

S. Longchupole. 1995. Thai Syntactical Analysis System by Method of Splitting Sentences from Paragraph form Machine Translation. Master Thesis. King Mongkut's institute of technology Ladkrabang (in Thai).

Yang Liu, Andreas Stolcke, Elizabeth Shriberg and Mary Harper. 2005. Using Conditional Random Fields for Sentence Boundary Detection in Speech, ACL, 451-458.

Andrew McCallum and Wei Li. 2003. Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons, CONLL.

Pradit Mittrapiyanuruk and Virach Sornlertlamvanich. 2000. The Automatic Thai Sentence Extraction. The Fourth Symposium on Natural Language Processing.

Xian Qian and Yang Liu. 2012. Joint Chinese Word Segmentation, POS tagging and Parsing, ACL.

Glenn Slayden, Mei-Yuh Hwang and Lee Schwartz. 2010. Thai Sentence-Breaking for Large-Scale SMT, WSSANLP, pp. 8-16.

Weiwei Sun and Jia Xu. 2011. Enhancing Chinese Word Segmentation Using Unlabeled Data, ACL.

Virach Sornlertlamvanich, Naoto Takahashi and Hitoshi Isahara. 1997. Building A Thai Part Of-Speech Tagged Corpus (ORCHID).

Charles Sutton, 2006. GRMM: Graphical Models in Mallet. http://mallet.cs.umass.edu/grmm/.

Charles Sutton and Andrew McCallum. 2011. An introduction to conditional random fields, Machine Learning.

Charles Sutton, Khashayar Rohanimanesh and Andrew McCallum. 2004. Dynamic Conditional Random Fields: Factorized Probabilistic Models for Labelling and Segmenting Sequence Data. In International Conference on Machine Learning (ICML).

Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. Sentence and token splitting based on conditional random fields. In Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, pages 49–57, Melbourne, Australia.

Aobo Wang and Min-Yen Kan, 2013. Mining Informal Language from Chinese Microtext: Joint Word Recognition and Segmentation, ACL.

Xuancong Wang, Khe Chai Sim and Hwee Tou Ng. Combining Punctuation and Disfluency Prediction: An Empirical Study, EMNLP 2014, pp.121-130

Xuancong Wang, Hwee Tou Ng, Khe Chai Sim. 2012. Dynamic Conditional Random Fields for Joint Sentence Boundary and Punctuation Prediction. INTERSPEECH 2012, 1384-1387

Suphawut Wathabunditkul. 2003. Spacing in the Thai Language. http://www.thailanguage.com/ref/spacing

Yue Zhang and Stephen Clark. 2008. Joint Word Segmentation and POS Tagging using a Single Perceptron, ACL.

Assigning Fine-grained PoS Tags based on High-precision Coarse-grained Tagging

Tobias Horsmann and Torsten Zesch
Language Technology Lab
Department of Computer Science and Applied Cognitive Science
University of Duisburg-Essen, Germany
{tobias.horsmann,torsten.zesch}@uni-due.de

Abstract

We propose a new approach to PoS tagging where in a first step, we assign a coarse-grained tag corresponding to the main syntactic category. Based on this high-precision decision, in the second step we utilize specially trained fine-grained models with heavily reduced decision complexity. By analyzing the system under oracle conditions, we show that there is a quite large potential for significantly outperforming a competitive baseline. When we take error-propagation from the coarse-grained tagging into account, our approach is on par with the state of the art. Our approach also allows tailoring the tagger towards recognizing single word classes which are of interest e.g. for researchers searching for specific phenomena in large corpora. In a case study, we significantly outperform a standard model that also makes use of the same optimizations.

1 Introduction

When a part-of-speech (PoS) tagger assigns word class labels to tokens, it has to select from a set of possible labels whose size usually ranges from fifty to several hundred labels depending on the language. Especially for new domains or under-resourced languages, there is usually not enough training data to reliably learn all the subtle differences between a large set of labels. We thus propose to split the PoS tagging task into two steps. A first high precision step, where we only assign a coarse-grained tag which can be reliably learned also with rather limited training data, and which benefits from additional unlabeled data in the form of clusters or embeddings. And a second step, where we apply specialized tagging models that only have to choose from a much smaller set of possible labels.

Figure 1 gives an overview of our approach using, in the first step, a coarse-grained tagset (similar to the universal tagset (Petrov et al., 2012)) and in the second step the fine-grained PTB tagset (Marcus et al., 1993). In the figure, we can see how knowing the coarse-grained tag informs the second step. For example, if we already know that *beautiful* is an adjective, we only have to choose between three possible tags (`JJ`, `JJR`, or `JJS`) instead of 45 tags for the full PTB tagset.

Our approach requires that the coarse tagging in the first step is more accurate than using fine-grained tagging itself, as we will loose some accuracy through error propagation between the steps. We will thus first analyze how well coarse tagging can actually be done, and also focus on whether coarse-grained models transfer better between different kinds of texts, as e.g. Ritter et al. (2011) shows that a fine-grained tagger trained on newswire data doesn't work well on social media.

Creating robust and accurate coarse-grained taggers is a worthwhile task on its own, as many NLP applications actually do not require fine-grained distinctions. For example, the popular TextRank algorithm (Mihalcea and Tarau, 2004) for keyphrase extraction uses coarse grained PoS tags to build the underlying co-occurrence graph, or Benikova and Biemann (2016) use coarse-grained tags to annotate semantic relations between nominals.

Another advantage of our approach is that the second tagging step can be easily customized for specific needs, e.g. if a scholar wants to analyze the usage of a specific PoS tag. In this case, we can use fine-grained models with additional features that are informative for this sub-problem, but might not be helpful for the overall tagging task.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 328–336, Osaka, Japan, December 11-17 2016.

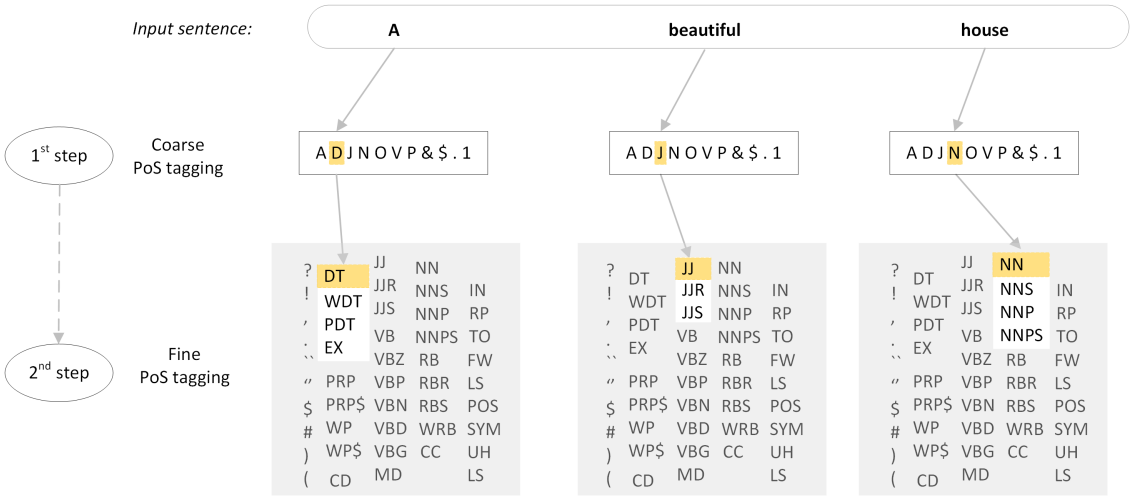


Figure 1: PoS tagging in two steps: The first step assigns a coarse tag (main word class) of a word. The second step determines the fine PoS tag based on the coarse PoS tag of the first step.

Related Work We are not aware of prior work using a similar approach. Some rule-based approaches (Brill, 1992; Hepple, 2000) assign the most probable tag to each token and then use transformation rules to correct the initial assignment. However, both steps use the same granularity and the initial assignment only reflects the prior probability but is not usable in itself.

There is also relatively little research on coarse-grained tagging. Gimpel et al. (2011) developed a tagger for Twitter data that uses a specialized coarse-grained tagset which is equivalent to our first step, but they do not aim at refining these assignments further as we do in the second step. Most approaches do fine-grained tagging first, and then map back to the universal set in order to make the different tagsets compatible. For example, Horsmann et al. (2015) use a coarse tagset to evaluate PoS tagging models using a set of corpora that are annotated with different fine-grained PoS tags.

2 Coarse-grained Tagging

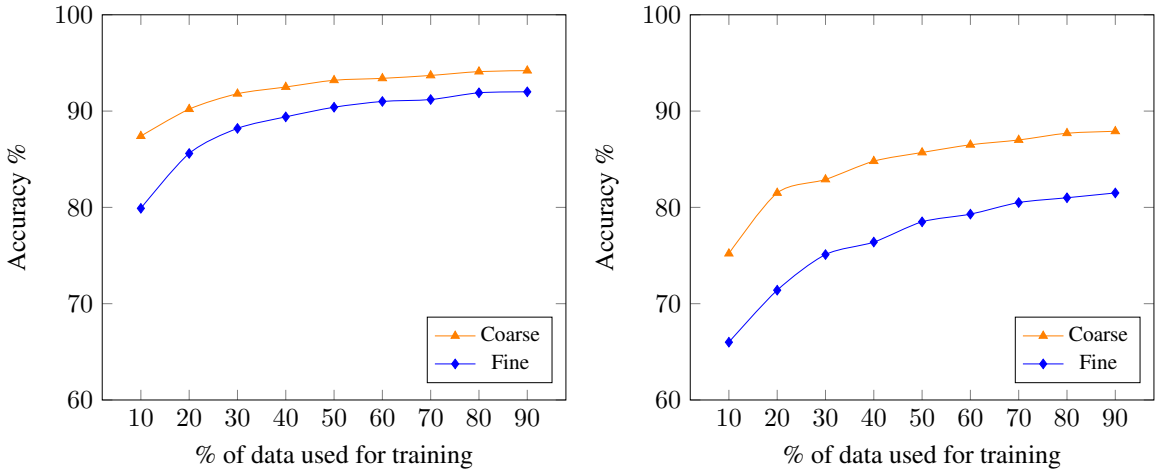
The first thing we need for our approach is a robust and highly accurate coarse PoS tagging. For domains or languages with little training data, accuracy is mainly limited by out-of-vocabulary tokens. In such situations, it seems easier to first assign a highly generalizing model that can be learned from relatively little training data instead of forcing the model to make an uninformed fine-grained decision. Thus, in this section we explore how well coarse-grained tagging actually works.

For our experiments, we use corpora from different genres (*News*, *Web*, *Chat*, and *Twitter*) in order to ensure that results are not bound to text properties only. Newswire text has a formal nature and contains few language errors, as it is usually carefully edited. Text from the web is (on average) less formal containing informal expressions and orthographic errors. Chat conversations are highly informal and often similar to spoken language, as the communication takes place between a smaller group of people with many non-standard abbreviations and orthographic errors. Twitter contains highly diverse text, as the platform is used for all kinds of purposes ranging from chat-like discussions to formal announcements. As *News* corpus, we use 46k tokens of the Wall Street Journal (WSJ) (Marcus et al., 1993), for *Web* we use 44k tokens from the GUM corpus (Zeldes, 2016) containing semi-formal text from various Wiki-platforms, as *Chat* corpus we use the NPS (Forsyth and Martell, 2007) chat corpus with 32k tokens, and for *Twitter* we use the 15k tokens Twitter messages provided by Ritter et al. (2011).

For our purposes, we need to map the fine-grained tags in these corpora to an inventory of coarse grained tags. We rely on the mappings provided by DKPro Core (Eckart de Castilho and Gurevych, 2014). We train our models using Conditional Random Fields (Lafferty et al., 2001) as implemented in FlexTag (Zesch and Horsmann, 2016) which relies on the machine learning framework DKPro TC (Daxenberger et al., 2014). As basic feature set which is common to all our models we use use ± 2 tokens

	Accuracy (%)			
	News	**Web**	**Chat**	**Twitter**
fine	92.0	88.6	86.8	80.2
coarse	94.2*	92.5*	91.4*	87.2*

Table 1: Accuracy of fine tagging vs. coarse tagging. Marked values are statistically significant against the fine-grained baseline (McNemar's test, $p < 0.05$)



(a) News

(b) Twitter

Figure 2: Learning curves for 'cleanest' dataset vs. 'noisiest' dataset

as local context, the 1,000 most frequently occurring character 2-grams to 4-grams over all tokens, and whether the target token contains capitalized characters, numeric values, or special characters.

Table 1 shows the results of 10-fold cross validation on each of the four corpora. We find that coarse tagging outperforms fine tagging on all datasets (statistically significant, McNemar's test, $p < 0.05$). As expected, the size of the gain is connected to the type of corpus, ranging from 2.2 percent point on *News* to 7.0 points on *Twitter*. The improvement can almost exclusively be attributed to intra-class errors of the fine-grained model, i.e. our coarse model is differentiating between the coarse grained classes as well as the fine-grained system, but isn't forced to make an uninformed decision. In the next section, we will investigate if this gives us enough head start to improve overall performance, but before that we further investigate the properties of our coarse tagger.

2.1 Amount of required training data

A practical benefit of coarse tagging should be that the amount of required training data is considerably lower than for fine tagging, as fewer labels should need less data to be learned. We thus adapted our 10-fold cross validation experiment to train on decreasing numbers of chunks, i.e. instead of training on 9 chunks and test on the remaining chunk, we train only on 8 and test on the 10th chunk, then train on 7 and test on the 10th chunk, and so on.

Figure 2 shows the results of this learning curve experiment on *News* (our 'cleanest' corpus) and *Twitter* (the 'noisiest'). For *News*, we see the expected behavior of a larger advantage of coarse tagging for smaller amount of training data, while for *Twitter* the distance between coarse and fine tagging does not change much. Note that we have much less data for *Twitter* than for *News*, so that we could still see a similar behavior for *Twitter* if more annotated data was available.

2.2 Word class performance

So far, we have seen that coarse tagging yields in general higher accuracy than fine tagging, but it would also be interesting to see which word classes benefit the most. The possible increase in performance

	Word class	# Token 10^3	Accuracy fine	coarse
News	Adjective	0.3	77.6	78.1
	Adverb	1.6	84.4	84.2
	Conjunction	1.1	99.0	98.9
	Determiner	4.2	98.6	98.5
	Noun	14.3	89.9	93.5
	Numeral	1.4	96.2	95.0
	Preposition	6.0	96.6	97.4
	Pronoun	1.5	98.3	98.9
	Punctuation	6.0	99.7	99.8
	Verb	6.6	86.4	93.4
	Other	0.4	95.3	96.9
		46.4	92.0	94.2

	Word class	# Token 10^3	Accuracy fine	coarse
Twitter	Adjective	0.7	58.6	57.7
	Adverb	0.8	80.1	80.9
	Conjunction	0.3	94.3	95.6
	Determiner	0.9	92.4	95.1
	Noun	3.5	69.1	84.1
	Numeral	0.3	74.1	69.5
	Pronoun	1.4	92.6	96.3
	Preposition	1.5	87.5	92.2
	Punctuation	2.0	96.5	98.7
	Verb	2.5	76.6	88.3
	Other	1.5	75.4	79.9
		15.0	80.2	87.2

Table 2: Coarse tagging results by coarse word class

is linked to at least two factors: (i) the size of the remaining fine-grained label set, and (ii) how well the remaining labels can be separated. For example, the set of fine-grained labels for the noun class is rather small (four tags: `NN`, `NNS`, `NNP`, `NNPS`) and the singular/plural noun decision is rather simple in English. However, the remaining problem to differentiate between normal nouns and proper nouns is far from trivial because the distinction is often not syntactically realized, but relies on world knowledge and context. For example, *I bought an apple.* vs. *I bought an Apple.* can only be decided based on capitalization, but especially in the social media datasets this signal is not very reliable. In contrast to nouns, we do not expect adjectives to improve much, as comparative and superlative can be quite easily distinguished from the base form of the adjective and from each other.

Table 2 shows the results per coarse-grained word class, where fine-grained results are mapped to the corresponding coarse-grained value for evaluation. Again, for space reasons, we only show *News* and *Twitter* as the most extreme representatives of our evaluation datasets. As expected, nouns and verbs are responsible for most of the improvement, as they are the bigger classes and the difficult decision faced in both classes are deferred to the second step.

2.3 Cross-domain performance

So far, we have conducted all our experiments in an in-domain setting, i.e. we train and test on the same kind of data (although of course not on exactly the same data). Since for new domains there usually is no training data in the first place, it is common to fall back on models trained on the more easily available *News* corpora. In this section, we want to evaluate the difference between fine and coarse tagging in such a setting. We will train on *News* enriched with additional training data and external resources in order to create a competitive model. We then evaluate this enhanced *News* model on the remaining datasets *Web*, *Chat*, and *Twitter*.

Fine Baseline Model In order to create a strong baseline, we augment our fine-grained tagger with more training data and external knowledge. We use the same feature set as before and train the model on the WSJ sections 0-21 and additional 250k tokens taken from the Switchboard corpus. We add distributional knowledge in form of Brown clusters (Brown et al., 1992) that we trained on 100 million tokens of English tweets crawled between 2011 and 2016. When we evaluate on the WSJ sections 22-24, we achieve an accuracy of 96.4% which is on par with the state of the art for fine-grained tagging which ranges from 96.5% (Brants, 2000) to 97.6% (Huang et al., 2015) according to the ACL wiki.[1]

Coarse Model Our coarse model is also trained on the WSJ section 0-21. An advantage of using coarse tags is the availability of many annotated corpora that can be easily mapped to coarse tags regardless of

[1] http://aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art)

	Accuracy (%)		
	News → Web	**News → Chat**	**News → Twitter**
fine	92.3	73.6	80.8
coarse	96.0*	88.9*	90.6*

Table 3: Accuracy of a fine trained and coarse trained model for tagging across domains. Marked values are statistically significant against the baseline (McNemar's test, $p < 0.05$)

the actually used fine tags. This immensely extends the pool of annotated training data that we can choose from. We thus additionally add 250k tokens of the Switchboard corpus as well as a PoS dictionary feature based on the three most frequent coarse tags of each word derived from the British National corpus (Clear, 1993). Furthermore, we add additional annotated Twitter data (Owoputi et al., 2013) to inform our classifier about phenomenons found in the social media domain. Please note that neither of these extension is possible for the baseline model because the tagsets are not compatible on the fine-grained level. However, we can use those resources for coarse grained tagging by mapping the fine tags to the same coarse tagset.

Table 3 shows the results of tagging the *Web*, *Chat*, and *Twitter* datasets, while we exclude *News* as we have been using it for training the models.[2] We find that in this cross-domain setting, our coarse-grained tagger significantly outperforms the fine-grained version for all three datasets. Especially the results on the *Chat* dataset are informative, as the news-trained model can obviously not be transferred well to chat data. In contrast, our coarse-grained version performs even better on *Chat* than on *Twitter* showing that it generalizes well.

3 Fine-grained Tagging

As we have shown in the previous section, we can assign coarse tags with higher accuracy than fine tags. This gives us the necessary head start for our second tagging step, as errors from the first stage will be propagated into the second stage and cannot be corrected anymore. For example, if we wrongly assigned the coarse tag `N` to a verb in the first step, we will in the second step apply the specialized classifier for nouns that has no chance of assigning a verb tag.

We implement the fine tagging by using a dedicated model which assigns fine PoS tags belonging to one coarse word class. For this step, we use a Support Vector Machine (Joachims, 1998) as implemented in Weka (Holmes et al., 1994), because this sub-task is not easily implemented as a sequence classification task. We use the same feature set as before and train the second step models on the WSJ considering only words belonging to the same coarse word class. Two coarse word classes (conjunction and numbers) do not have further fine-grained specializations in the PTB tagset, i.e. they can be mapped one-to-one to their fine PoS tag.

In order to assess the full potential of our approach, we will first ignore error propagation between the two tagging stages. This means, we assume an oracle condition that assigns correct coarse-grained tags. Afterwards, we will evaluate our full model with error propagation and check against the hypothetical performance of the oracle.

We show the results of fine tagging under oracle condition in Table 4 in comparison to our fine baseline. Marked values show statistically significant improvements against the baseline (McNemar's test, $p < 0.05$). Under oracle condition our model performs significantly better on all our for evaluation corpora. In the last row of Table 4, we show the results of our two-step approach, where errors made in the coarse step are propagated to the fine-grained tagging. Our results are on par with the state-of-the-art fine baseline for three out of four datasets, and are significantly better for the *Chat* corpus.

In order to better understand the influence of coarse errors, we simulate a certain level of error propa-

[2]Dataset with additional tags not occurring in the training data pose special problems. We treat the few unknown tags as always correct in order not to influence the relative difference between the fine and coarse setup too much.

	Accuracy (%)			
	News	**Web**	**Chat**	**Twitter**
Fine baseline	96.4	92.3	73.6	80.8
Oracle 100%	98.4*	95.8*	83.0*	89.3*
Two-step	96.1	92.4	75.8*	81.9

Table 4: Accuracy of fine-grained tagging using oracle coarse-grained tagging and two-step tagging. Marked values are statistically significant against the baseline (McNemar's test, $p < 0.05$)

gation by using an oracle that only returns the correct tag with a certain probability (uniformly sampled over all tags). Figure 3 shows the resulting relationship between a certain level of coarse-grained accuracy and the resulting fine-grained accuracy. We see that both are in a similar linear relationship for all datasets. Measuring the slope, we can approximate that a 1 percent point improvement in coarse accuracy translates into an improvement between .8 and .9 percent points in fine-grained accuracy. The accuracy of our two-step tagging is marked by a black dot. Its location is always on or very close to the hypothetical performance of the oracle. We can thus conclude that our predictions are quite accurate and better coarse performance will really lead to better fine performance. On *Chat* our approach performs slightly better than predicted, which we take as an indicator that our error propagation experiment is a rather conservative lower-bound for the impact of coarse tagging errors.

4 Tag-specific Optimization

While it is hard to optimize the tagging accuracy for a specific fine-grained tag using traditional taggers, our approach offers a straightforward way to train a second stage model that is tailored towards such a task. For example, a scholar might be only interested in finding past participle verbs (word class `VBN` in the PTB tagset). However, they are easily confused with past tense verbs (word class `VBD`) due to their identical word surface form as in the example below.

> *Mr. Smith was arrested/VBN and **charged/VBN** along with the others when he returned to Namibia this month*

A tagger is easily mislead to tag the second VBN (bold faced) as past tense verb rather than past participle, because the auxiliary is outside a narrowly defined ± 2 context window. Note that the past tense decision might even be correct in a very similar context like *Mr. Smith was arrested/VBN and **went/VBD** along anyway.* So the tagger either needs to learn the compositional semantics of 'charged along' from a large number of annotated examples or by using additional external knowledge. We thus argue that for our specialized `VBN` classifier it is easier than in a global setting to utilize a wider context window and use more lexical features.

If we wish to optimize the detection accuracy for VBN in our approach, we can ignore the performance on the other tags. Thus, we transform the problem into a one vs. all classification, i.e. we further reduce the decision complexity in the second tagging step to a binary decision between `VBN` and $\neg$`VBN`. In order to avoid a class bias, we ensure a balanced distribution of both classes in the training data and train our model again on the WSJ. We extend the feature set to provide additional hints for the classifier that one of the words outside of the ± 2 context window is an inflected form of *have*, *be*, or a modal verb.

Table 5 shows the results of our optimization experiment. We evaluated the accuracy of tagging only the word class VBN. We report accuracy for the fine baseline and our two-step model in (i) standard configuration and (ii) the `VBN` optimized version. The first thing to notice is that without optimization our standard two-step model performs worse than the fine-grained baseline. However, our optimized two-step model significantly outperforms the fine-grained baseline on all datasets. This shows that our separation of the tagging process in two steps allows for an easy way of customizing a PoS tagger towards specific needs by incorporating linguistically motivated features.

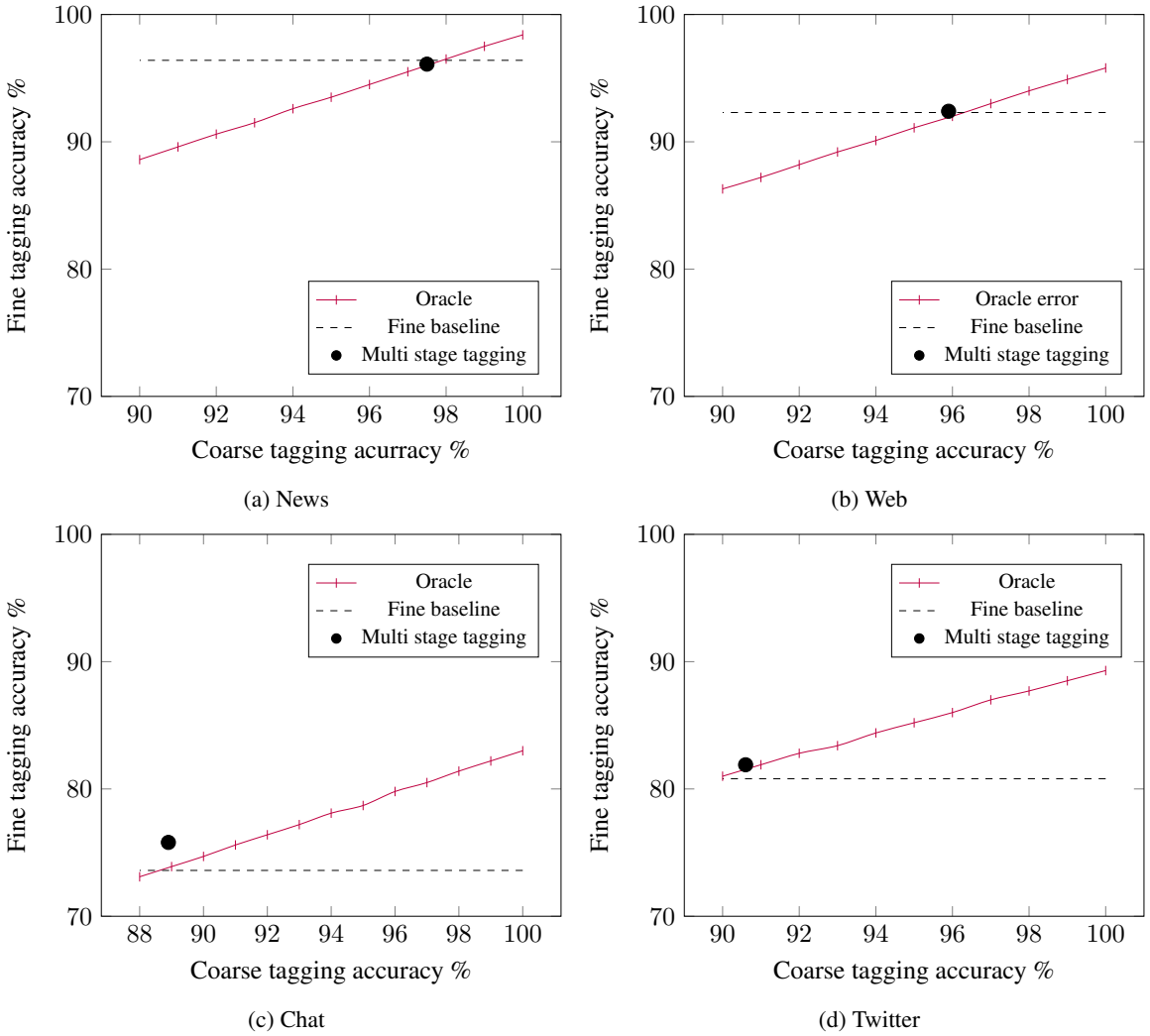


Figure 3: Relationship between fine-grained accuracy and error-level of the coarse-grained tagging. The horizontal dashed line shows the baseline performance and the black dot shows the accuracy of our two-step PoS tagging.

	VBN Accuracy (%)							
	News		**Web**		**Chat**		**Twitter**	
	w/o	w	w/o	w	w/o	w	w/o	w
fine	86.7	87.4	81.2	82.4	75.7	77.0	70.0	76.4
two-step	84.6	90.7*	79.0	86.7*	74.0	81.3*	64.3	81.4*

Table 5: Results for detecting **VBN** with and without optimization. Marked values are statistically significant against the fine-grained baseline (McNemar's test, $p < 0.05$)

5 Conclusions

We introduce a new two-step PoS tagging approach, where first a coarse-grained tag is assigned with high precision, and in a second step a specialized fine-grained classification with heavily reduced decision complexity is applied. We show that the accuracy of coarse-grained tagging is significantly higher when directly learning a coarse model compared with a fine-grained model. This especially holds in a cross-domain setting where coarse models generalize much better than fine models. As many NLP applications rely on coarse-grained tags, this finding has a huge potential for practical improvements in downstream tasks. If we evaluate the fine-grained accuracy, we find that our two-step approach performs on par

with the classical single-step paradigm, except for the especially challenging chat messages where it is significantly better. Using an oracle for coarse-grained tagging, we show that our approach has a huge potential for improving the overall accuracy, as improving coarse-grained accuracy translates almost directly into coarse-grained accuracy. In the common setting that a researcher is only interested in the performance of a specific fine-grained tag, we show that our approach can be easily optimized and then significantly outperforms the classical approach that does not improve even if the same optimizations are used.

In future work, we aim to further improve coarse grained tagging, as our analysis shows that coarse-grained improvements nearly linearly translate into fine-grained improvements. We also aim at further improving fine-grained tagging, as we see large potential for further specializing the feature sets used for different word classes.

Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant No. GRK 2167, Research Training Group "User-Centred Social Media".

References

Darina Benikova and Chris Biemann. 2016. SemRelData Multilingual Contextual Annotation of Semantic Relations between Nominals: Dataset and Guidelines. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 4154–4161, Portoroz, Slovenia.

Thorsten Brants. 2000. TnT: A Statistical Part-of-speech Tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, pages 224–231, Seattle, Washington.

Eric Brill. 1992. A Simple Rule-based Part of Speech Tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 152–155, Trento, Italy.

Peter F Brown, Peter V DeSouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-Based n-gram Models of Natural Language. *Computational Linguistics*, 18:467–479.

Jeremy H. Clear. 1993. The British National Corpus. pages 163–187. Cambridge, MA, USA.

Johannes Daxenberger, Oliver Ferschke, Iryna Gurevych, and Torsten Zesch. 2014. DKPro TC: A java-based framework for supervised learning experiments on textual data. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 61–66, Baltimore, Maryland.

Richard Eckart de Castilho and Iryna Gurevych. 2014. A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT (OIAF4HLT) at COLING 2014*, pages 1–11, Dublin, Ireland.

Eric N. Forsyth and Craig H. Martell. 2007. Lexical and Discourse Analysis of Online Chat Dialog. In *Proceedings of the International Conference on Semantic Computing*, pages 19–26, Washington, DC, USA.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech Tagging for Twitter: Annotation, Features, and Experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, pages 42–47, Portland, Oregon.

Mark Hepple. 2000. Independence and Commitment: Assumptions for Rapid Training and Execution of Rule-based POS Taggers. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 278–277, Hong Kong.

Geoffrey Holmes, Andrew Donkin, and Ian H. Witten. 1994. *WEKA: A Machine Learning Workbench*. Working paper series (University of Waikato. Department of Computer Science).

Tobias Horsmann, Nicolai Erbs, and Torsten Zesch. 2015. Fast or Accurate ? – A Comparative Evaluation of PoS Tagging Models. In *Proceedings of the International Conference of the German Society for Computational Linguistics and Language Technology (GSCL 2015)*, pages 22–30, Essen, Germany.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *ArXiv e-prints 1508.01991*.

Thorsten Joachims, 1998. *Text categorization with Support Vector Machines: Learning with many relevant features*, pages 137–142.

John D Lafferty, Andrew McCallum, and Fernando C N Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 8th International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390, Atlanta, Georgia.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A Universal Part-of-Speech Tagset. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC 2012)*, pages 2089–2094, Istanbul, Turkey.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named Entity Recognition in Tweets: An Experimental Study. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*, pages 1524–1534, Edinburgh, Scotland.

Amir Zeldes. 2016. The GUM corpus: creating multilayer resources in the classroom. *Language Resources and Evaluation*, pages 1–32.

Torsten Zesch and Tobias Horsmann. 2016. FlexTag: A Highly Flexible Pos Tagging Framework. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 4259–4263, Portorož, Slovenia.

Data-Driven Morphological Analysis and Disambiguation for Morphologically Rich Languages and Universal Dependencies

Amir More
IDC Herzliya
habeanf@gmail.com

Reut Tsarfaty
Open University of Israel
reutts@openu.ac.il

Abstract

Parsing texts into *universal dependencies (UD)* in realistic scenarios requires infrastructure for *morphological analysis and disambiguation (MA&D)* of typologically different languages as a first tier. MA&D is particularly challenging in *morphologically rich languages (MRLs)*, where the ambiguous space-delimited tokens ought to be disambiguated with respect to their constituent morphemes. Here we present a novel, language-agnostic, framework for MA&D, based on a transition system with two variants, word-based and morpheme-based, and a dedicated transition to mitigate the biases of variable-length morpheme sequences. Our experiments on a Modern Hebrew case study outperform the state of the art, and we show that the morpheme-based MD consistently outperforms our word-based variant. We further illustrate the utility and multilingual coverage of our framework by morphologically analyzing and disambiguating the large set of languages in the UD treebanks.

1 Problem Statement

A decade following the emergence of statistical parsers for English (Charniak, 1996; Bod, 1995), the CoNLL data sets presented a new challenge: the development of data-driven statistical parsers that can be trained to parse any language given an appropriately annotated treebank (Buchholz and Marsi, 2006; Nivre et al., 2007). These data sets facilitated the development of accurate, language-agnostic, dependency parsers (Nivre et al. (2006), McDonald (2006) etc.), but not without shortcomings: they require that input tokens be *morphologically analyzed and disambiguated* in advance.

This latter assumption breaks down in realistic parsing scenarios where the morphological analysis of an input token may consist of multiple syntactic words to participate in the parse tree (Tsarfaty et al., 2010). The *universal dependencies (UD)* initiative aims to remedy this by presenting a harmonized set of treebanks, now 54 and counting, with a unified annotation scheme and multilayered annotation. Specifically, UD data distinguishes the input space-delimited tokens from the (morpho)syntactic words that participate in the parse tree (Nivre et al., 2016).

Efforts towards parsing texts into *universal dependencies* in realistic scenarios thus require language-agnostic infrastructure for automatic *morphological analysis and disambiguation (MA&D)* of data from typologically different languages. MA&D is particularly challenging in *morphologically rich languages (MRLs)*, where space-delimited input tokens may have multiple analyses, only one relevant in context. This *morphological ambiguity* of a token is typically represented as a lattice. The term *Morphological Disambiguation* (MD) refers to selecting a single path through the *morphological analysis* (MA) lattice.

In Semitic languages, MD is particularly challenging (Adler, 2007; Bar-haim et al., 2008; Shacham and Wintner, 2007; Pasha et al., 2014; Habash and Rambow, 2005). To illustrate, Figure 1 shows the MA lattice of the Hebrew phrase 'bclm hneim'[1] (literally: in-shadow-of-them the-pleasant, meaning: in their pleasant shadow). Different paths in the lattice represent different disambiguation decisions. In

[1] We transliterate as in Sima'an et al. (2001).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 337–348, Osaka, Japan, December 11-17 2016.

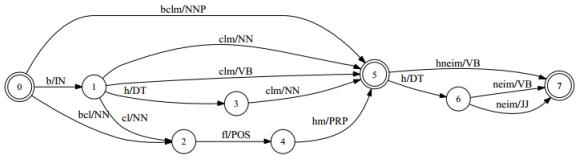


Figure 1: An example of an MA lattice of the phrase "bclm hneim" ("in their pleasant shadow") in transliterated Hebrew. Edges mark syntactic words, and double circles mark white spaces.

the context of 'bclm hneim', the correct path of 'bclm' is 'b-cl-(fl)-hm', "in-shadow-(of)-them". In the context of another sentence, the token 'bclm' may be "Betzelem", the name of a famous organization.

In MRLs, MD is also more subtle than simply segmenting the space-delimited tokens. Many MRLs are fusional, and clitics may be fused into hosts. In Figure 1, in the phrase 'b-cl-(fl)-hm' (literally: in-shadow-(of)-them), the possessive 'fl' (of) is fused into the pronoun 'hm' (them) and remains implicit in the surface form. Such fusion results in an ambiguous number of morphosyntactic nodes that participate in the analysis, impacting downstream applications as syntactic and semantic parsing, translation, etc.

Previous work on MA&D in MRLs, and in Semitic languages in particular (Adler, 2007; Bar-Haim et al., 2005; Shacham and Wintner, 2007; Pasha et al., 2014), relied on language-specific lexica and cannot be executed cross-linguistically. General CRF implementations, such as MarMoT (Müller et al., 2013), that can be applied across languages, assume an unrealistic, gold pre-segmented setting (Bjorkelund et al., 2013). For generic morphological segmentation, Morfessor (Smit et al., 2014) uses a max-likelihood in semi-supervised settings, but it cannot handle the rich labeling of morphological segments.

In this paper we present a general, language-agnostic solution the for the MA&D task. We target *joint* morphological segmentation and tagging, as has been advocated in monolingual cases (Zhang and Clark, 2011; Bar-haim et al., 2008; Adler and Elhadad, 2006; Habash and Rambow, 2005), in *universal* settings. Our technical approach extends the transition-based framework for structured prediction of Zhang and Clark (2011). We define and implement two MD variants: word-based and morpheme-based. We present the best MA&D results to date, and demonstrate that the morpheme-based variant consistently outperforms our word-based one, while providing state-of-the-art results on full-fledge, fine-grained, MD of Hebrew. Furthermore, our MA&D framework is intentionally designed with language independence in mind. Devoid of requiring language-specific resources, we show robust and competitive MA&D performance on MRLs and non-MRLs alike, for circa 50 languages in the most recent release of UD treebanks (Nivre et al., 2016).

2 Challenges and Formal Settings

We propose a data-driven framework for MA&D of MRLs and non-MRLs alike. The MA component implements a function that maps each input sentence to its MA lattice, and the MD component implements a transition-based model that accepts the lattice as input and returns a selected path as output.

Formally, our transition system is a quadruple $S = (C, T, c_s, C_t)$, where C is a set of configurations, T is a set of transitions, c_s is an initialization function, and $C_t \subseteq C$ is a set of terminal configurations. A transition sequence y of length n, $y = c_0, t_1(c_0), ..., t_n(c_{n-1})$, starts with an initial configuration $c_0 = c_s(x)$ for the input sentence x and ends with a terminal configuration $c_n \in C_t$, where $t_i \in T$ and $c_n = t_n(c_{n-1}) \in C_t$. We employ an objective function F where x is the input sentence and $GEN(x)$ is the set of possible transition sequences for x:

$$F(x) = argmax_{y \in GEN(x)} Score(y) \quad (1)$$

To compute $Score(y)$, $y \in GEN(x)$ is mapped to a global feature vector $\Phi(y) \in N_d$, where each feature is a count of occurrences of a pattern defined by a feature function ϕ. The feature vector $\Phi(y)$ is defined via a set of d feature functions $\{\phi_i\}_{i=1}^{d}$. The way Φ is defined effectively determines the quality of the parser, since the feature model captures linguistic information to which the model learns to assign

weights. Given this vector, $Score(y)$ is computed by multiplying $\Phi(y)$ with a weights vector $\vec{\omega} \in R^d$.

$$Score(y) = \Phi(y) \cdot \vec{\omega} = \sum_{c_j \in y} \sum_{i=1}^{d} \omega_i \phi_i(c_j) \quad (2)$$

Following Zhang and Clark (2011), our system learns the weights vector $\vec{\omega} \in R^d$ via the generalized perceptron, using the early-update averaged variant of Collins and Roark (2004). The algorithm iterates through a gold-annotated corpus, each sentence is disambiguated (decoded) with the last known weights, and if the decoded result differs from the gold standard, the weights are updated. As in Zhang and Clark (2011), decoding is based on the beam search algorithm, where a number of possible parse sequences are evaluated concurrently to mitigate irrecoverable prediction errors. At each step, the transition system applies all valid applicable transitions to all candidates. The B highest scoring expanded candidates are maintained and passed on to the next step. Those that don't make the B mark, fall off the beam.

Our MRL setting (cf. Tsarfaty et al. (2010)) poses three technical challenges to this general scheme:
(i) the formal challenge: how should we define a transition system for MA&D?
(ii) the learning challenge: how can we define feature functions that learn morphological phenomena?
(iii) the decoding challenge: how can we effectively compare morpheme sequences of variable length?

3 Our Proposed Solution

Let $x = x_1...x_k$ be an input sentence of k tokens and $L = MA(x_1), ..., MA(x_k)$ be the morphological ambiguity lattice for x, where L is a contiguous series of word-lattices $L_i = MA(x_i)$ connected top to bottom, as illustrated in Figure 1. Each word lattice L_i is a set of sequences of morphemes, and each sequence is a single disambiguated analysis for x_i. We define the *morphosyntactic representation* (MSR) of an arc in the lattice as a tuple $m = (s, e, f, t, g)$ with lattice nodes s and e marking the start and end of a morpheme, a form f, a part-of-speech tag t, and a set g of attribute:value grammatical properties.

Defining Configurations. A configuration for the MD transition system is a quadruple (L, n, i, M) where $L = MA(x)$ is the sentence-lattice, n is a node in L, i is the 0-based index of a word-lattice in L, and M is a set of disambiguated morphemes (i.e., selected arcs). The terminal configuration is defined to be $C_t = \{(L, top(L), tokens(L), M)\}$ for any L, M, where $tokens(L)$ is the number of word-lattices that form L. The initial configuration function c_s concatenates the L_i lattices of the tokens into a single structure $L = MA(x_1) + ... + MA(x_k)$, and sets $n = bottom(L)$, $i = 0$ and $M = \emptyset$.

Defining Transitions. There are two conceivable ways to make morphological disambiguation decisions, in a *word-based* (WB), and in a *morpheme-based* (MB), fashion, in the terminology of Tsarfaty and Goldberg (2008). In WB models (a.k.a *token-level* in the UD terminology), the disambiguation decision determines a complete path of morphemes between token-boundaries. In the lattice, this refers to selecting a path between two token-boundary nodes (double circles). MB disambiguation decisions (also termed *lexical-level*, or *word-level* in UD) occur at any node in the lattice indicating a morpheme boundary, with more than one outgoing arc, choosing a specific arc m among them.

Interim: Word-Based or Morpheme-Based? WB and MB strategies face contradicting, and complementary, challenges. In WB models, disambiguation decisions are complex, and learning how to score them is expected to suffer from data sparseness. MB models, on the other hand, over-generalize in terms of possible morphological combinations, and learning to score combinations may fail to generalize and be prone to over-fitting. On top of that, morpheme sequences are longer than word sequences, which, in a transition-based system, is known to be more error prone. Finally, variable-length sequences introduce length biases which negatively impact performance. Since MA&D is the base for the NLP pipeline, it is critical to settle this debate empirically and establish the basis for downstream tasks.

Parameterizing Transitions. A transition system is required to distinguish between all possible decisions it can make at a given point. At the same time, the model should be able to generalize from seen decisions to unseen ones, and effectively learn to disambiguate open-class words and out-of-vocabulary

items. To satisfy these desiderata, we define a *delexicalization projection* for a pre-defined set of parts-of-speech tags O capturing open-class categories. Simply put, this projection neutralizes the lattice-nodes specific indices, and, for any tag $t \in O$, it further neutralizes the lexical form. Formally:

$$DLEX_O(m) = \begin{cases} (\_, \_, \_, t, g) & \text{if } t \in O \\ (\_, \_, f, t, g) & \text{otherwise} \end{cases} \quad (3)$$

3.1 Word-Based Modeling

The Transition System For word-based (WB) modeling, a single transition morphologically disambiguates whole word-lattices such that the node n of a configuration is always at a word boundary (a node that is a bottom, top, or both, of word-lattices of L). We define the transitions in the WB system as an open set of transitions termed MD_s, specifying the parameter s as a single path:

$$MD_s : (L, n, i, M) \rightarrow (L, q, i+1, M \cup \{m_0, .., m_j\}) \quad (4)$$

Here, $\{m_0, ..., m_j\} \in L$ form a contiguous path of arcs, where m_0 starts at node n, m_j ends at node q (they can be the same arc), and s is the projected paths $s = DLEX_O(m_0), ..., DLEX_O(m_j)$. A terminal configuration will therefore contain the union of contiguous paths of word-lattices in L, together forming a complete morphological disambiguation of the ambiguous tokens of the input sentence.

Learning We define three types of word-lattice properties: o - the surface form of the token itself, a - the $DLEX$-projected lattice (all MSRs projected by the delexicalization function), and p - a chosen disambiguated path, which only exists for previously processed lattices. Using these properties, we define baseline feature templates modeled after POS tagging: unigram, bigram, and trigram combinations of o and a, and p-based features, which predict the next disambiguation decision based on the previous one(s).

3.2 Morpheme-Based Modeling

The Transition System For morpheme-based (MB) modeling, a single transition chooses an outgoing arc of the current node n in the lattice, requiring a disambiguation decision if (and only if) there is more than one outgoing arc. Again we define the transitions as an open set of transitions termed MD_s, now specifying s as a single *arc*:

$$MD_s : (L, n, i, M) \rightarrow (L, q, j, M \cup \{m\}) \quad (5)$$

Here, m is a morpheme $(n, q, f, t, g) \in L$, and $s = DLEX_O(m)$. If node q is at a word boundary, then $j = i + 1$, otherwise $j = i$. For a terminal configuration, each $m \in M$ is an outgoing arc of the end node of another arc in M (with the exception of the first morpheme, starting at $bottom(L)$) forming a contiguous path that disambiguates x.

Learning In the MB model we can access specific information concerning the current node inside the word-lattice. We define the properties f, t and g, corresponding to arcs' form, part-of-speech and morphological attribute:value pairs. We use these properties in various unigram, bigram, and trigram combinations, in parallel with the WB model. As in the WB model we also define the property p as the path in the previously disambiguated word-lattices. We define the property n to be the set of $DLEX$-projected outgoing morphemes of the current node (this parallels the property a of WB models, but at morpheme granularity). Similarly to the WB case, we use unigram, bigram, and trigram combinations of these properties as well.

Decoding Since the number of arcs in lattices' paths for x may vary, so do the number of transitions in our morpheme-based transition system. This violates a basic assumption of standard beam search decoding — that the number of transitions is a deterministic function of the input.

There are two inherent biases in varied-length transition sequences driven by the general perceptron algorithm. The beam search algorithm tests the best candidate after each step for goal fulfillment. A short

sequence may temporarily be the best candidate and fulfill the goal, while longer (and possibly correct) sequences are incomplete and may be lost. On the other hand, long sequences have more features, therefore their score may be arbitrarily inflated. So the score may be higher for longer paths, even though a shorter one may be correct and may fall off the beam.

To address these challenges we introduce a special transition we call *ENDTOKEN (ET)*, that explicitly increments i, instead of implicitly in MD_s. So, in equation (4) we set $j = i$ and apply:

$$ET : (L, n, i, M) \rightarrow (L, n, i + 1, M) \tag{6}$$

ET is required to occur exactly once at the end of the word-lattice, when n is the top of some word-lattice in L. Set aside from other transitions, ET has its own set of features. Other than incrementing i, ET has no effect on configurations, but it does cause a re-ordering of candidates in the beam during decoding, at each token boundary. Note that ET kicks in only for variable length lattices. On same-length lattices, ET is skipped and equation (4) remains as is — the process essentially falls back on the standard, same-length decoding.

An MD transition sequence thus becomes the union of disjoint sets of configurations $y = y_{md} \cup y_{et}$, and changes $Score$ in Equation (2), where $|y_{et}|$ is the # of tokens in L with variable length paths. :

$$\sum_{i=1}^{d} \omega_i^{md} \phi_i^{md}(y_{md}) + \sum_{j=1}^{d} \omega_j^{et} \phi_j^{et}(y_{et}) = \sum_{c_k \in y_{md}} \sum_{i=1}^{d} \omega_i^{md} \phi_i^{md}(c_k) + \sum_{c_l \in y_{et}} \sum_{j=1}^{d'} \omega_j^{et} \phi_j^{et}(c_l) \tag{7}$$

While the number of morphemes, and therefore $|y_{md}|$, can vary, $|y_{et}|$ is deterministic per lattice. Using this anchor, the features of the ET transition provide a counter-balance to the effects of varied-length sequences by scoring fully disambiguated paths of each word-lattice individually, occuring a fixed amount of times for all paths.

ENDTOKEN vs. IDLE transitions Variable-length sequences in beam search also exist in the structured prediction of constituency trees. Zhu et al. (2013) introduced an IDLE transition (also adopted in Honnibal and Johnson (2014) and Zhang et al. (2014)) that, like ET, has no effect on configuration, but unlike ET, occurs only at the end of the parsing sequence, an arbitrary number of times, until all parsing sequences are complete.

While IDLE transitions make sense when applied after a complete hierarchical structure is predicted — where they may learn to rerank candidates based on features that are visible at the top of the structure (the root) — it is futile to use last-seen features that arbitrarily exist at the end of a morphological disambiguation (linear) sequence, to rerank candidates again and again. This is because at the end of the sequence, we can no longer save candidates that were lost earlier on due to length discrepancies .

To mitigate this, one might try to create IDLE padding with global features spanning the entire disambiguated path. Even then, the learned model parameters would not generalize well, since these features will be applied an arbitrary number of times — the maximal length of an occasional word lattice we are at — which has no linguistic significance, and may arbitrarily inflate certain scores.

ET transitions, in contrast, occur right when they are needed — at the boundary of a word-lattice. This position enables the reordering of candidates right after a length discrepancy may have been introduced. Moreover, ET scores are counted against the global score a fixed number of times per lattice, for all, any length, candidates. This enables a fair comparison of all paths per lattice.

4 Empirical Evaluation

We empirically evaluate the proposed models, and investigate their strengths, weaknesses, and bounds. We start with a detailed investigation of MA&D in the Semitic language Modern Hebrew, which is known for its rich morphology and significant ambiguity. We then verify the cross-linguistic coverage of the models on the set of UD treebanks (Nivre et al., 2016), to validate their efficacy.

We implement *yap* (*yet another parser*), a general-purpose transition-based framework for structured prediction, based on beam search and the generalized perceptron. We extend it with the models described herein. We implement the WB, MB variants, ET transitions, and evaluate different feature settings.

(a)

	Word-Based				
	unigram	+bigram	+trigram	+next unigram	+next with bigram
	85.73 (86.72)	86.9 (87.88)	86.7 (87.66)	**92.19 (92.76)**	91.98 (92.59)
+ET	89.09 (89.81)	89.93 (90.71)	90 (90.85)	**93.39 (93.94)**	92.84 (93.46)

(b)

	Morpheme-Based				
	unigram	+bigram	+trigram	+next unigram	+next with bigram
	90.67 (91.41)	91.12 (91.88)	90.09 (91.82)	93.56 (94.16)	**93.89 (94.49)**
+ET	92.68 (93.36)	92.74 (93.55)	92.64 (93.47)	94.27 (94.92)	**94.33 (94.9)**

Table 1: Dev. set results for Word-Based (a) and Morpheme-based (b) MD: F_1 for full morphological disambiguation (form, part of speech, morphological properties). (n parenthesis: F_1 for form and POS only. The +ET lines indicate a variant that employs the ENDTOKEN transition at token boundaries.

We report the F_1 metric comparing the MSRs of *predicted* vs. *gold* lattice arcs, for full morphological disambiguation (segmentation, POS tags, and all morphological properties), and for segmentation and POS tags only. For comparison with previous work, we also report token-level accuracy (while F_1 awards partial success on word-lattices, token-level accuracy requires exact match on a whole path per token).

4.1 The Case for Modern Hebrew

Setup We evaluate MA&D performance on the Modern Hebrew section of the SPMRL 2014 Shared Task (Seddah et al., 2014), which has been derived from the Unified-SD treebank of Tsarfaty (2013). We updated the treebank to provide consistent theories for the treebank annotation and lexicographic resources (Itai and Wintner, 2008), a consistency that we found lacking in the SPMRL 2014 Hebrew section. We use the standard split, and train on the standard train set (5k sentences). Here we provide results and in-depth analysis on *dev* and confirm our findings on *test*.

A pre-condition for the execution of our MD models is an $MA(x)$ function that generates word-lattices for x (§2). We start off with a morphological analyzer that we implemented, called HEBLEX, which relies on the Ben-Gurion Hebrew Lexicon used by Adler and Elhadad (2006). The lexicon contains full analyses for 567,483 words and 102 prefixes. HEBLEX uses the lexicon to determine the various combinations of prefixes and words that form valid tokens. This process is far from trivial due to morphological *fusion*, as some morphemes are implicit (§1).

Although the lexicon is quite large, there are still tokens which are *out-of-vocabulary (OOV)*. OOV tokens may be of two types: it may be that an entire string is out of the lexicon (mostly proper nouns) or that the affixes and the open class items are seen, but their combination has not yet been encountered. We address OOV by assigning proper noun analyses to entire tokens, as well as to all arcs combined with seen affixes. This adds ambiguity to the lattices, but gives the MD the chance to select a correct path.

In our experiments we aim to quantify exactly the effect of lexical coverage of the MA on MD accuracy. To this end, we add an option to *infuse* missing gold analyses into the MA lattices provided by HEBLEX and present two sets of results: once disambiguating lattices with infused gold analyses (ideal MA), and once without infusing gold analyses (realistic MA).

Results Tables (1a), (1b) present our investigation of the WB and MB models on the dev test, respectively, with different feature templates. Our results show that the MB disambiguation consistently outperforms our WB variant, in various feature template settings. Moreover, the ET transition consistently improves performance, with best results for Hebrew at F_1 scores of 94.3 (94.9) for full MD (seg/POS only). The token-level accuracy for our best results are 93.07 (93.9) for full MD (seg/POS).

These results were obtained on *infused* lattices, that include the gold path as one of the alternatives. In order to gauge the effect of incomplete lexical coverage, we disable infusion of the gold analyses into the HEBLEX lattices. We then observe a drop to F_1 scores of 89.62 (92.06) and token accuracy of 87.72(90.85). To set our results in context, we applied our best model in "English-like" settings for tagging, with gold pre-segmented text. F_1 then increases to 96.82 (97.44). That is, in "English-like" settings, our tagging accuracy (97.44) is as high as state-of-the-art English tagging (Manning, 2011).

MarMoT (Müller et al., 2013), a state of the art CRF tagger, obtains F_1 93.38 on full MD on this set.

Next we aim to verify that ET transitions indeed act as intended. We classify a sequence length error as either an overshoot (predicted morphological disambiguation sequence is longer than gold) or undershoot (predicted shorter than gold). Without ET, in the infused setting, 36.8% of sentences have incorrect length and the overshot:undershoot ratio is 6.6:1. Adding ET transitions results in 31.8% length errors, correcting 20.62% of the overshoot errors, resulting in ratio of 4:1. In the un-infused setting, 41.4% of the sentences have incorrect length with a ratio of 6.39:1. Adding ET results in 36% length errors, correcting 20.67% of the overshoot errors, resulting in ratio of 3.7:1.

Hebrew previous results are non-trivial to compare to due to significant changes of the treebank along the way and unavailable code of previous work. The most relevant results to ours are by Adler (2007), who reports state-of-the-art results for Modern Hebrew in realistic (non-infused) setting, with self-reported token accuracy of 90% (93%) on a different evaluation set. For his prediction on our dev set, F_1 evaluation yields 85.74 (87.95), much lower than ours. Segmentation F_1 for Adler is 96.35, while ours is 97.6. We confirm our findings on the test set, for which Adler F_1 yields 82.91 (85.56). Our best model now yields 86.23 (88.85) and 92.96 (93.73) in realistic and infused settings, respectively.

4.2 The Case for Universal Dependencies

Setup We evaluate the cross-linguistic coverage of our MD models on the UD set. We parse 48 UD treebanks from the UD1.3 release (Nivre et al., 2016), training on the *train* set and evaluating on *test*.[2]

We implement a *universal* data-driven morphological analyzer, that can be trained out-of-the-box along with our MD models for any treebank in the CoNLL-U format. We generate a dictionary for each language from its train set by collecting all seen analyses of each token in the training data, where an analysis is composed of MSRs that contain a lemma, POS, and the full set of morphological features. The dictionary maps each token to a set of MSR sequences, which then compose their ambiguous MA lattices. For out-of-vocabulary (OOV) tokens, the MA pre-computes the cardinality of each coarse POS — the number of unique tokens per coarse POS — and consider the top 5 POS as "open-class". For these top 5 POS, the MA computes the 50 highest-frequency MSRs (POS + morph. properties) to be used as the OOV lattice of an OOV token. When applying MA to the training set, we add the OOV lattice to tokens whose known analysis contains an open-class POS. The model thus encounters during training a larger space of states than the observed one, and learns to accurately apply transitions in OOV circumstances at test time.

Results Table 2 reports F_1 accuracy for full MD and seg/POS for UD languages that do not require segmentation. For most large train sets ($> 200k$ tokens), we observe 2-3 points absolute drop from infused (ideal) to uninfused (realistic) setting. This suggests that when large train sets exist, our data-driven MA is a viable economic alternative to costly hand-crafted monolingual lexical resources. To scrutinize our realistic results we also report non-OOV-only F_1 for 5k limited uninfused setting. Here we see that for ~80% of languages, our results are on a par with a state-of-the-art tagger, MarMot, retrained on these data, within 0.035 (or less) points gap. This demonstrates that our disambiguation capacity is on-par with MarMot, where performance gaps come mostly from our OOV strategy (which is intentionally restrained, to allow handling of MRL segmentation that is handled by MarMot). Table 3 shows results for UD MRLs that require segmentation, contrasting results on gold pre-segmented input and un-segmented raw data. For raw data we see a minor, ~0.02, drop in F_1, compared to gold-segmented settings. That is, our model still retains the competitive MA&D performance, in a *single, universal, trainable* model — we attribute this to our *joint* segmentation and tagging strategy, which overcomes error propagation.[3]

[2] We do not present results for 6 languages: cs,kk,es_ancora,en_esl,pt_br,ja_ktc, as some or all form fields are empty.

[3] Shortly before submitting this article, UDPipe (Straka et al., 2016), a tool for tokenization, morphological analysis, tagging and parsing, had been released. As its name suggests, UDPipe is a pipeline implementation packaging together separate tools for different tasks. Our approaches and ultimate goals are rather different. We present *joint* morphological segmentation and tagging, as opposed to a pipeline. Moreover, our framework can be extended into a *single* transition-based system performing all tasks jointly, overcoming overheads and error propagation, as we intend to address next.

Lang.	sents	5k Sent. Trainset			Full Trainset		
	words	inf.	no inf.	MM	inf.	no inf.	MM
bg	8907	0.933	0.914 (0.959)	0.937 (0.964)	0.946	0.926	0.948
	124474	0.969	0.96 (0.983)	0.976 (0.987)	0.979	0.969	0.982
cu	5077	0.901	0.853 (0.911)	0.893 (0.927)	0.908	0.851	0.897
	46025	0.964	0.931 (0.977)	0.959 (0.978)	0.965	0.929	0.962
da	4868	0.932	0.88 (0.944)	0.943 (0.966)	0.934	0.894	0.943
	88979	0.953	0.91 (0.954)	0.961 (0.972)	0.954	0.922	0.961
el	1929	0.912	0.876 (0.918)	0.921 (0.944)	0.914	0.876	0.921
	47449	0.977	0.965 (0.989)	0.979 (0.99)	0.978	0.965	0.979
en	12543	0.904	0.887 (0.927)	0.904 (0.937)	0.927	0.914	0.93
	204586	0.923	0.911 (0.951)	0.923 (0.954)	0.942	0.932	0.945
en_	3650	0.95	0.944 (0.955)	0.96 (0.969)	0.95	0.944	0.96
lines	66374	0.95	0.944 (0.955)	0.96 (0.969)	0.95	0.944	0.96
et	14510	0.866	0.821 (0.911)	0.883 (0.933)	0.913	0.884	0.928
	187814	0.923	0.895 (0.953)	0.933 (0.961)	0.951	0.932	0.959
eu	5396	0.877	0.79 (0.895)	0.874 (0.93)	0.88	0.797	0.877
	72974	0.948	0.904 (0.964)	0.944 (0.971)	0.948	0.908	0.946
fi	12217	0.884	0.756 (0.942)	0.889 (0.961)	0.925	0.864	0.932
	162721	0.915	0.863 (0.96)	0.928 (0.973)	0.95	0.909	0.958
ga	720	0.781	0.743 (0.87)	0.819 (0.909)	0.791	0.747	0.819
	16701	0.896	0.873 (0.946)	0.924 (0.961)	0.9	0.882	0.924
gl	2276	0.971	0.968 (0.985)	0.971 (0.984)	0.971	0.968	0.971
	79329	0.971	0.968 (0.985)	0.971 (0.984)	0.971	0.968	0.971
got	4360	0.874	0.824 (0.873)	0.877 (0.901)	0.871	0.83	0.877
	44722	0.955	0.927 (0.964)	0.961 (0.972)	0.953	0.924	0.961
grc	13185	0.862	0.762 (0.867)	0.844 (0.883)	0.862	0.776	0.859
	196083	0.926	0.853 (0.937)	0.935 (0.946)	0.92	0.861	0.941
grc_	13306	0.829	0.719 (0.89)	0.774 (0.887)	0.909	0.853	0.9
proiel	166061	0.911	0.827 (0.972)	0.893 (0.966)	0.973	0.939	0.971
hi	13304	0.82	0.799 (0.823)	0.867 (0.891)	0.846	0.832	0.891
	281057	0.944	0.934 (0.955)	0.947 (0.964)	0.953	0.948	0.963
hr	3557	0.86	0.812 (0.877)	0.87 (0.913)	0.86	0.813	0.87
	78817	0.951	0.936 (0.974)	0.954 (0.973)	0.951	0.933	0.954
hu	1433	0.763	0.701 (0.831)	0.751 (0.862)	0.758	0.697	0.751
	33016	0.92	0.895 (0.951)	0.945 (0.973)	0.915	0.896	0.945
id	4477	0.932	0.926 (0.934)	0.937 (0.945)	0.932	0.926	0.937
	97531	0.932	0.926 (0.934)	0.937 (0.945)	0.932	0.926	0.937

Lang.	sents	5k Sent. Trainset			Full Trainset		
	words	inf.	no inf.	MM	inf.	no inf.	MM
la	2660	0.797	0.696 (0.814)	0.797 (0.876)	0.787	0.709	0.797
	37819	0.904	0.847 (0.927)	0.935 (0.967)	0.897	0.854	0.935
la_	16258	0.916	0.895 (0.922)	0.922 (0.938)	0.935	0.914	0.945
ittb	276941	0.978	0.965 (0.985)	0.979 (0.986)	0.986	0.977	0.988
la_	11986	0.857	0.773 (0.859)	0.841 (0.886)	0.892	0.843	0.892
proiel	132376	0.949	0.896 (0.959)	0.943 (0.967)	0.969	0.943	0.97
lv	673	0.797	0.745 (0.927)	0.817 (0.931)	0.801	0.739	0.817
	12629	0.876	0.841 (0.971)	0.898 (0.971)	0.88	0.835	0.898
nl	13000	0.836	0.797 (0.889)	0.823 (0.911)	0.88	0.863	0.872
	197134	0.866	0.841 (0.923)	0.861 (0.928)	0.902	0.891	0.897
nl_	6641	0.939	0.934 (0.967)	0.95 (0.971)	0.943	0.94	0.952
lassysmall	88929	0.954	0.95 (0.973)	0.961 (0.974)	0.956	0.953	0.963
no	15696	0.926	0.887 (0.952)	0.93 (0.963)	0.947	0.917	0.952
	244776	0.959	0.944 (0.97)	0.964 (0.976)	0.971	0.958	0.975
pl	6800	0.858	0.772 (0.882)	0.854 (0.916)	0.87	0.778	0.866
	69499	0.955	0.915 (0.976)	0.961 (0.978)	0.961	0.922	0.967
pt	8800	0.913	0.881 (0.919)	0.916 (0.938)	0.927	0.906	0.93
	214812	0.96	0.947 (0.965)	0.967 (0.977)	0.967	0.958	0.972
ro	4759	0.915	0.894 (0.917)	0.927 (0.939)	0.916	0.895	0.927
	108618	0.962	0.946 (0.967)	0.962 (0.973)	0.961	0.947	0.962
ru	4029	0.869	0.791 (0.895)	0.861 (0.925)	0.872	0.791	0.861
	79772	0.957	0.925 (0.976)	0.956 (0.979)	0.956	0.925	0.956
ru_	46750	0.896	0.818 (0.919)	0.89 (0.944)	n/a	n/a	n/a
syntagrus	815485	0.965	0.935 (0.977)	0.968 (0.98)	n/a	n/a	n/a
sl	6471	0.874	0.791 (0.876)	0.884 (0.927)	0.882	0.804	0.892
	112334	0.954	0.923 (0.967)	0.963 (0.976)	0.958	0.927	0.967
sl_	2472	0.83	0.796 (0.86)	0.857 (0.902)	0.832	0.793	0.857
sst	23575	0.896	0.88 (0.92)	0.919 (0.947)	0.895	0.878	0.919
sv	4303	0.928	0.921 (0.951)	0.946 (0.966)	0.931	0.921	0.946
	66645	0.952	0.947 (0.965)	0.966 (0.975)	0.954	0.947	0.966
sv_	3650	0.955	0.952 (0.964)	0.957 (0.966)	0.955	0.952	0.957
lines	63949	0.955	0.952 (0.964)	0.957 (0.966)	0.955	0.952	0.957
zh	3997	0.903	0.889 (0.918)	0.913 (0.932)	0.903	0.889	0.913
	98608	0.914	0.903 (0.933)	0.923 (0.943)	0.914	0.903	0.923

Table 2: MA&D for unsegmented languages : F_1 scores of the languages in UD that do not require morphological segmentation, the upper line indicates full MD, the lower line indicates segmentation and POS only. MM columns are results for MarMoT. Results in parentheses are F_1 scores for non-OOV-only tokens.

Lang.	sents	Gold Segmented 5k Train. Set (non-OOV accuracy)					Gold Segmented Full Trainset					Un-Segmented Full Trainset			
	words	inf.	+ET	no inf.	+ET	MM	inf.	+ET	no inf.	+ET	MM	inf.	+ET	no inf.	+ET
ar	6174	0.887	0.887	0.853	0.853 (0.87)	0.903 (0.924)	0.892	0.892	0.86	0.86	0.907	0.867	0.871	0.8	0.799
	225853	0.956	0.956	0.948	0.948 (0.956)	0.956 (0.972)	0.959	0.959	0.951	0.951	0.959	0.929	0.933	0.882	0.882
ca	13123	0.953	0.953	0.919	0.919 (0.958)	0.957 (0.965)	0.963	0.963	0.939	0.939	0.968	0.961	0.961	0.938	0.937
	429157	0.969	0.969	0.936	0.936 (0.972)	0.973 (0.977)	0.977	0.977	0.954	0.954	0.98	0.975	0.975	0.952	0.951
cs_	23478	0.865	0.857	0.758	0.758 (0.862)	0.86 (0.921)	0.901	0.901	0.831	0.831	0.904	0.897	0.899	0.827	0.827
cac	472608	0.973	0.969	0.93	0.928 (0.984)	0.975 (0.988)	0.983	0.983	0.961	0.961	0.987	0.983	0.983	0.961	0.961
cs_	860	0.845	0.844	0.812	0.801 (0.871)	0.887 (0.922)	0.848	0.847	0.816	0.812	0.887	0.832	0.822	0.804	0.8
cltt	26234	0.956	0.959	0.942	0.938 (0.988)	0.98 (0.991)	0.958	0.957	0.94	0.942	0.98	0.953	0.951	0.937	0.938
de	14118	0.928	0.928	0.921	0.921 (0.936)	0.927 (0.941)	0.929	0.929	0.921	0.921	0.927	0.93	0.928	0.921	0.92
	269626	0.928	0.928	0.921	0.921 (0.936)	0.927 (0.941)	0.929	0.929	0.921	0.921	0.927	0.93	0.928	0.921	0.92
es	14187	0.929	0.928	0.888	0.888 (0.943)	0.927 (0.956)	0.939	0.939	0.908	0.908	0.936	0.93	0.933	0.9	0.903
	382436	0.947	0.947	0.931	0.931 (0.959)	0.945 (0.967)	0.955	0.955	0.943	0.943	0.953	0.948	0.951	0.935	0.938
fa	4798	0.953	0.961	0.958	0.958 (0.972)	0.963 (0.978)	0.954	0.953	0.956	0.957	0.963	0.957	0.956	0.947	0.949
	121020	0.96	0.968	0.964	0.964 (0.974)	0.969 (0.98)	0.96	0.959	0.962	0.963	0.969	0.962	0.962	0.954	0.955
fi_	14981	0.876	0.88	0.794	0.793 (0.921)	0.858 (0.944)	0.856	0.847	0.811	0.809	0.915	0.916	0.929	0.814	0.856
ftb	127602	0.914	0.917	0.856	0.855 (0.953)	0.904 (0.957)	0.883	0.869	0.843	0.844	0.943	0.939	0.95	0.849	0.899
fr	14554	0.931	0.93	0.921	0.918 (0.935)	0.939 (0.954)	0.943	0.951	0.925	0.925	0.949	0.944	0.942	0.92	0.921
	356216	0.949	0.947	0.941	0.939 (0.952)	0.955 (0.967)	0.959	0.965	0.942	0.942	0.964	0.958	0.957	0.937	0.938
he	5241	0.934	0.933	0.888	0.888 (0.907)	0.921 (0.953)	0.934	0.93	0.891	0.886	0.922	0.914	0.917	0.724	0.724
	135496	0.968	0.968	0.937	0.938 (0.947)	0.955 (0.974)	0.967	0.966	0.939	0.937	0.957	0.945	0.947	0.768	0.769
it	11699	0.945	0.946	0.91	0.911 (0.953)	0.943 (0.962)	0.96	0.958	0.945	0.945	0.97	0.953	0.957	0.935	0.937
	249330	0.959	0.96	0.924	0.925 (0.961)	0.958 (0.972)	0.97	0.967	0.956	0.955	0.978	0.961	0.965	0.946	0.947
ta	400	0.761	0.793	0.761	0.732 (0.912)	0.82 (0.929)	0.794	0.797	0.757	0.756	0.82	0.72	0.722	0.659	0.664
	6329	0.835	0.859	0.825	0.813 (0.927)	0.877 (0.938)	0.856	0.861	0.827	0.821	0.877	0.765	0.772	0.714	0.717
tr	3947	0.781	0.873	0.765	0.806 (0.935)	0.855 (0.933)	0.794	0.787	0.768	0.805	0.855	0.84	0.781	0.742	0.78
	40609	0.884	0.938	0.87	0.897 (0.968)	0.934 (0.964)	0.893	0.885	0.879	0.899	0.934	0.907	0.87	0.846	0.871

Table 3: MA&D in segmented languages: F_1 scores of the languages in UD which require morphological segmentation. The upper line indicate full MD, the lower line indicates segmentation and POS only. The left hand side shows results for GOLD segmentation, the right hand side for input lattices. MM columns are results for MarMoT. Results in parentheses are F_1 scores for non-OOV-only tokens.

5 Conclusion

We present an MD transition-based system that can effectively cope with extreme morphological ambiguities in MRLs. To the best of our knowledge, this is the first joint framework for MRL segmentation and tagging in a transition-based setup. Moreover, we present the best MA&D results for Modern Hebrew to date, and the first ever set of MA&D results for the most recent release of UD treebanks (UD1.3).[4] Our system provides a first tier for dependency parsing in real-world scenarios, dispensing with the need of external pre-processing. Furthermore, this transition-based model can be extended into a *joint* model for *complete* morphological and syntactic analysis, as has been previously advanced in phrase-based parsing (Tsarfaty, 2006; Goldberg and Tsarfaty, 2008; Cohen and Smith, 2007; Green and Manning, 2010).

[4]Upon publication we will make our source code, executables and trained models available at `https://github.com/habeanf/yap`.

References

Meni Adler and Michael Elhadad. 2006. An unsupervised morpheme-based hmm for Hebrew morphological disambiguation. In Nicoletta Calzolari, Claire Cardie, and Pierre Isabelle, editors, *ACL*. The Association for Computer Linguistics.

Meni Adler. 2007. *Hebrew Morphological Disambiguation: An Unsupervised Stochastic Word-based Approach*. Ph.D. thesis, Ben-Gurion University of the Negev, Beer-Sheva, Israel.

Roy Bar-Haim, Khalil Simaan, and Yoad Winter. 2005. Part-of-speech tagging for Hebrew and other semitic languages. *Master's thesis, Technion, Haifa, Israel*.

Roy Bar-haim, Khalil Sima'an, and Yoad Winter. 2008. Part-of-speech tagging of Modern Hebrew text. *Natural Language Engineering*, 14(2):223–251.

Anders Bjorkelund, Ozlem Cetinoglu, Richard Farkas, Thomas Mueller, and Wolfgang Seeker. 2013. (re)ranking meets morphosyntax: State-of-the-art results from the SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 135–145, Seattle, Washington, USA, October. Association for Computational Linguistics.

Rens Bod. 1995. *Enriching Linguistics With Statistics*. Ph.D. thesis, University of Amsterdam.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL-X*, pages 149–164.

Eugene Charniak. 1996. Tree-Bank Grammars. In *AAAI/IAAI, Vol. 2*, pages 1031–1036.

S. B. Cohen and N. A. Smith. 2007. Joint morphological and syntactic disambiguation. In *Proceedings of EMNLP*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yoav Goldberg and Reut Tsarfaty. 2008. A single framework for joint morphological segmentation and syntactic parsing. In *Proceedings of ACL*.

Spence Green and Christopher D. Manning. 2010. Better Arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 394–402, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 573–580, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association of Computational Linguistics – Volume 2, Issue 1*, pages 131–142.

Alon Itai and Shuly Wintner. 2008. Language resources for Hebrew. *Language Resources and Evaluation*, 42(1):75–98, March.

Christopher D. Manning, 2011. *Computational Linguistics and Intelligent Text Processing: 12th International Conference, CICLing 2011, Tokyo, Japan, February 20-26, 2011. Proceedings, Part I*, chapter Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?, pages 171–189. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ryan McDonald. 2006. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvanya.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order crfs for morphological tagging. *In Proceedings of EMNLP*.

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

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Yevgeni Berzak, Riyaz Ahmad Bhat, Cristina Bosco, Gosse Bouma, Sam Bowman, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Çağrı Çöltekin, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Sebastian Garza, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gokirmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Normunds Grūzītis, Bruno Guillaume, Jan Hajič, Dag Haug, Barbora Hladká, Radu Ion, Elena Irimia, Anders Johannsen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Jessica Kenney, Simon Krek, Veronika Laippala, Lucia Lam, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Keiko Sophie Mori, Shunsuke Mori, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Vitaly Nikolaev, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Loganathan Ramasamy, Laura Rituma, Rudolf Rosa, Shadi Saleh, Baiba Saulīte, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Carolyn Spadine, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jing Xian Wang, Jonathan North Washington, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2016. Universal dependencies 1.3. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University in Prague.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the spmrl 2014 shared task on parsing morphologically-rich languages. pages 103–109.

Danny Shacham and Shuly Wintner. 2007. Morphological disambiguation of Hebrew: A case study in classifier combination. In *Proceedings of ACL*.

Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman, and N. Nativ. 2001. Building a tree-bank of Modern Hebrew text. *Traitment Automatique des Langues*, 42(2).

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–24. Association for Computational Linguistics.

Milan Straka, Jan Hajic, and Jana Straková. 2016. Udpipe: Trainable pipeline for processing conll-u files performing tokenization, morphological analysis, pos tagging and parsing. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Reut Tsarfaty and Yoav Goldberg. 2008. Word-based or morpheme-based? annotation strategies for modern Hebrew clitics. In *Proceedings of LREC*.

Reut Tsarfaty, Djame Seddah, Yoav Goldberg, Sandra Kuebler, Marie Candito, Jennifer Foster, Yannick Versley, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing for morphologically rich language (spmrl): What, how and whither. In *Proceedings of the first workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL) at NA-ACL*.

Reut Tsarfaty. 2006. Integrated morphological and syntactic disambiguation for modern Hebrew. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, COLING ACL '06, pages 49–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Reut Tsarfaty. 2013. A unified morphosyntactic scheme for stanford dependencies. In *Proceedings of ACL*.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Comput. Linguist.*, 37(1):105–151, March.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Character-level chinese dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1326–1336, Baltimore, Maryland, June. Association for Computational Linguistics.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 434–443. The Association for Computer Linguistics.

Automatic Syllabification for Manipuri language

Loitongbam Gyanendro Singh, Lenin Laitonjam, Sanasam Ranbir Singh
Computer Science and Engineering Department
Indian Institute of Technology Guwahati, Assam, India
Email: {gyanendrol9,lenin.lai,ranbir}@iitg.ernet.in

Abstract

Development of hand crafted rule for syllabifying words of a language is an expensive task. This paper proposes several data-driven methods for automatic syllabification of words written in Manipuri language. Manipuri is one of the scheduled Indian languages. First, we propose a language-independent *rule-based* approach formulated using *entropy* based phonotactic segmentation. Second, we project the syllabification problem as a sequence labeling problem and investigate its effect using various sequence labeling approaches. Third, we combine the effect of sequence labeling and rule-based method and investigate the performance of the hybrid approach. From various experimental observations, it is evident that the proposed methods outperform the baseline rule-based method. The entropy based phonotactic segmentation provides a word accuracy of 96%, CRF (sequence labeling approach) provides 97% and hybrid approach provides 98% word accuracy.

1 Introduction

Manipuri language, one of the scheduled Indian languages, belonging to a Tibeto-Burman languages family(Chelliah, 1990) is syllabic in nature. In general, a syllable follows *onset-nucleus-coda* (consonant-vowel-consonant) structure, where nucleus is the core component defined by a vowel. The preceding and following consonants defined by the onset and coda respectively may or may not be present in a syllable. However every syllable must have a nucleus. Formation of an onset, nucleus and coda, while producing an uninterrupted syllabic sound, greatly affects the pronunciation of a language. For various applications such as Text-to-Speech Synthesis (TTS) (Kishore and Black, 2003; Bellur et al., 2011), Automatic Speech Recognition (ASR) (Wu et al., 1998) etc., proper syllabification of a word is one of the important core issues, especially for syllabic Indian languages. Though there have been several studies in this direction for the rich resource Indian languages such as Hindi, Bengali, Tamil, Telegu, Malayalam, Marathi (Bellur et al., 2011; Narendra et al., 2011; Kurian et al., 2011), only very few studies have been reported for low resource Manipuri language (Abbi and Awadhesh, 1985; Chelliah, 1990). To the best of our knowledge, no corpus is available for Manipuri language in UTF-8 to study various aspects of corpus-based linguistic understanding. This has motivated us to generate Manipuri corpus in UTF-8 and analyse various characteristics such as onset and coda distributions, syllable formulation and syllabification rules etc.

Manipuri language is written generally using two scripts; *Bengali scripts* and *Meitei Mayek* [1]. At present, majority of the Manipuri documents are written using Bengali scripts and, hence the investigation in this paper focuses on Manipuri words written using Bengali script. Further, processing of Bengali scripts poses more challenges than processing Meitei Mayek because of the phoneme imbalance. Bengali script has 55 symbols to represent 38 phonemes in Manipuri language (Singh et al., 2007).

One of the major challenges in syllabifying Indian languages such as Hindi, Urdu, Bengali, Marathi, Kashmiri, Punjabi, Gujarati is the *schwa* deletion in their written forms (Kishore and Black, 2003;

[1] http://www.unicode.org/L2/L2000/00259-MeeteiMayek.pdf

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 349–357, Osaka, Japan, December 11-17 2016.

Choudhury et al., 2004). Majority of the existing studies for syllabifying Indian languages are based on hand crafted rules built on language characteristics. Building such rules requires domain knowledge and it is an expensive task. Further, such rules often fail to capture *heteronym* (word having same spelling, but different pronunciations).

In general, languages evolves over time. New vocabularies are added. One language borrows vocabularies from other languages and become parts of the regular usage. For example, words like `institution`, `election`, `daddy` etc. have become more preferred words than their Manipuri counterparts. While syllabifying such borrowed or loan words, the rules crafted using the characteristics of the native language often fail to capture the syllabic structures of such word.

In this paper, we first explored two rule-based approaches which are *Baseline System* (C*VC* structured) and *Entropy Based Phonotactic Segmentation* (EPS). Secondly, we transform the syllabification problem as a *sequence labeling problem* and investigate the effect of two state-of-art methods, namely *Conditional Random Field* (CRF) (Sutton and McCallum, 2006) and *Maximum Entropy Markov Model* (MEMM) (McCallum et al., 2000). The intuition behind such transformation is that with an appropriate training set, these models can learn the inherent dependencies automatically and address the problem of *schwa*, *heteronym* and *loan* words. Another advantage of such machine learning methods is the language independent model. Given an appropriate annotated dataset, these methods can syllabify with reasonable accuracy without the knowledge of the underlying language. In summary, this paper has the following five contributions.

- Generate Manipuri corpus in UTF-8.
- Statistical analysis of the syllabified corpus for understanding syllable components (i.e. onset, nucleus and coda).
- Development of a data driven rule-based syllabification methods by exploring entropy distribution.
- Investigating the syllabification performance of CRF, MEMM and comparing with rule-based counterparts.
- Evaluating CRF model over Assamese language dataset.

2 Related Works

Two approaches are mainly considered for automatic syllabification of a language.

(i) **Rule-based approach**

(a) **Obligatory Onset Principle** (Hooper, 1972), or **Principle of Maximum Open Syllabicity** (Pulgram, 1970): It is one of the simplest principles, which is based on the assumption that open syllable i.e., a syllable with no coda, is significance. This is in fact very naive approach and it is impractical for Manipuri as it has various legal consonant clusters other that just CV syllable structure.

(b) **Sonority principles** (Selkirk, 1984): In this method, syllabification depends on sonority of each phoneme i.e., its quality of being sonorous. Sonorous is the capability of giving out sound especially resonance, deep sound. This principle assigns numerical values to every phoneme of a syllable depending on its sonority level where vowels have the highest value followed by nasals, fricatives, and plosives. The main disadvantage of sonority information is the position dependency of phonemes. The same phoneme present at a different location of a syllable may have different sonorous property. There exists several instances where phoneme does not fit its typical sonority rules.

(c) **The legality principle** (Goslin and Frauenfelder, 2001): Syllabification is based on the validation of the syllable structure considering its onsets and codas. It also considers the legality of the consonant clusters to detect its splitting points. It allows consonant clusters to be valid if they appear in some syllables. Legality principle requires a big corpus to study the legality of

its constituents structures. Its main problem arises when several valid splitting instances of a consonant cluster are possible resulting on ambiguous syllabification rules.

(d) **Maximum onset principle** (MOP) (Kahn, 2015): It is very similar to legality principle. Here, if multiple legal splits are possible, it gives preference to longer onsets irrelevant of the legality of the syllable coda.

Rule-based approaches are language dependent. A prior knowledge of the syllable structure and phonotactics of the language is necessary to derive such rules. Further, syllabification purely based on rule-based approached is mostly inadequate due to the presence of various ambiguities. There are various instances where correct syllabification cannot be obtained by a definite rule or may even break the conventional syllabification principle.

(ii) **Data driven approach**

(a) Zhang and Hamilton (1997) suggested Learning English Syllabification Rules system that learn rules using a symbolic pattern recognition approach.

(b) Adsett and Marchand (2009) provide a comparison between various data-driven syllabification algorithms namely, IB1 (94.36%) and Look-up Procedure (91.44%) algorithms along with Hidden Markov Support Vector Machines (95.17%), Liangs algorithm (95.48%) and Syllabification by Analogy (96.70%) which incorporate structure information.

(c) (Marchand et al., 2009; Adsett and Marchand, 2009) showed that rule-based approaches perform poorly as compared to the data driven approaches.

To the best of our knowledge, there have not been any work related to corpus-based machine learning approach using sequence labeling method for syllabifying Manipuri words. In the similar line of approach, Dinu et al. (2013) used CRF using two level tagging (phone boundary and phone distance) to perform syllabification for Romanian language. However, unlike their study, our approach uses only one level tagging as discussed in the section 5.2.

Further, Rogova et al. (2013) proposed a language-independent probabilistic syllabification of phonetic transcriptions of words using Segmental Conditional Random Fields (SCRF). This method works with phonetic transcriptions of word as input. Due to the existance of inherent $schwa$ in the script, phonetic transcriptions of a word may not be correct for Manipuri words.

3 Corpus generation

One of the challenges working with low resource Manipuri language is the unavailability of UTF-8 data sources. To create UTF-8 textual corpus for this study, we have identified two Manipuri local newspapers [2]. The local Manipuri newspapers written in Bengali script are not unicode compatible. The news articles are published and archived on the news website in PDF format. In order to generate unicode compatible text corpus, we deploy a Bengali OCR [3]. However, the accuracy of the OCR is very poor i.e., 52.6% words accuracy. We, therefore, manually correct the text extracted from the OCR. Thus, our corpus consists of 733 documents with 89,084 words and 26,203 unique words covering all possible Manipuri phone types.

4 Statistical analysis of Manipuri syllables

The 26,203 unique words present in the corpus are manually syllabified to study their characteristics. Each syllable is annotated with its positional information (`beg`, `mid` and `end`). For each word, the beginning syllable is marked as `beg`, ending syllable as `end` and all syllables that are between `beg` and `end` as `mid`. For example, the word *particular* is divided into *par*_`beg`, *ti*_`mid`, *cu*_`mid` and *lar*_`end`. Analysis of the distribution of onset, nucleus, and coda over different syllabic positions is important for understanding various linguistic aspects of a language. For the proposed rule-based approach for

[2] `http://www.poknapham.in/`, `http://naharolgithoudang.in/`

[3] `https://www.newocr.com/`

Table 1: Types of syllable structure with their positional distribution (in percentage)

Syllable structures	beg	mid	end	total
CV	12.61628	24.82558	21.98335	59.42521
CVC	12.13398	10.99894	2.97172	26.10465
CVV	1.72568	1.557875	0.56950	3.85306
CVCC	1.96749	1.346458	0.47040	3.78435
V	3.10253	0.202167	0.38583	3.69054
CCV	0.12552	0.447938	0.72013	1.29360
VC	0.71353	0.066067	0.04756	0.82716
CCVC	0.18498	0.121563	0.12684	0.43339
VV	0.31580	0.046247	0.00924	0.37130
CVVC	0.03699	0.005285	0.01453	0.05681
CCVV	0.02906	0.001321	0.00264	0.03303
CCVCC	0.02378	0.002642	0.00528	0.03171
VCC	0.02246	0.00	0.00	0.02246
CVCCC	0.00924	0.00	0.00528	0.01453
CVVCC	0.01189	0.00264	0.00	0.01453
CCVVC	0.00792	0.00	0.00132	0.00924
CCVVCC	0.00264	0.00264	0.00132	0.00660
CVVC	0.00264	0.00396	0.00	0.00660
CCCC	0.00528	0.00	0.00132	0.00660
VVC	0.00132	0.00132	0.00132	0.00396
VVCC	0.00264	0.00132	0.00	0.00396
CCVCCC	0.00264	0.00	0.00	0.00264
CCCV	0.00132	0.00	0.00132	0.00264
VCCC	0.00132	0.00	0.00	0.00132

syllabifying Manipuri word, we study distributional characteristics of onset, nucleus, and coda as one of the core parameters.

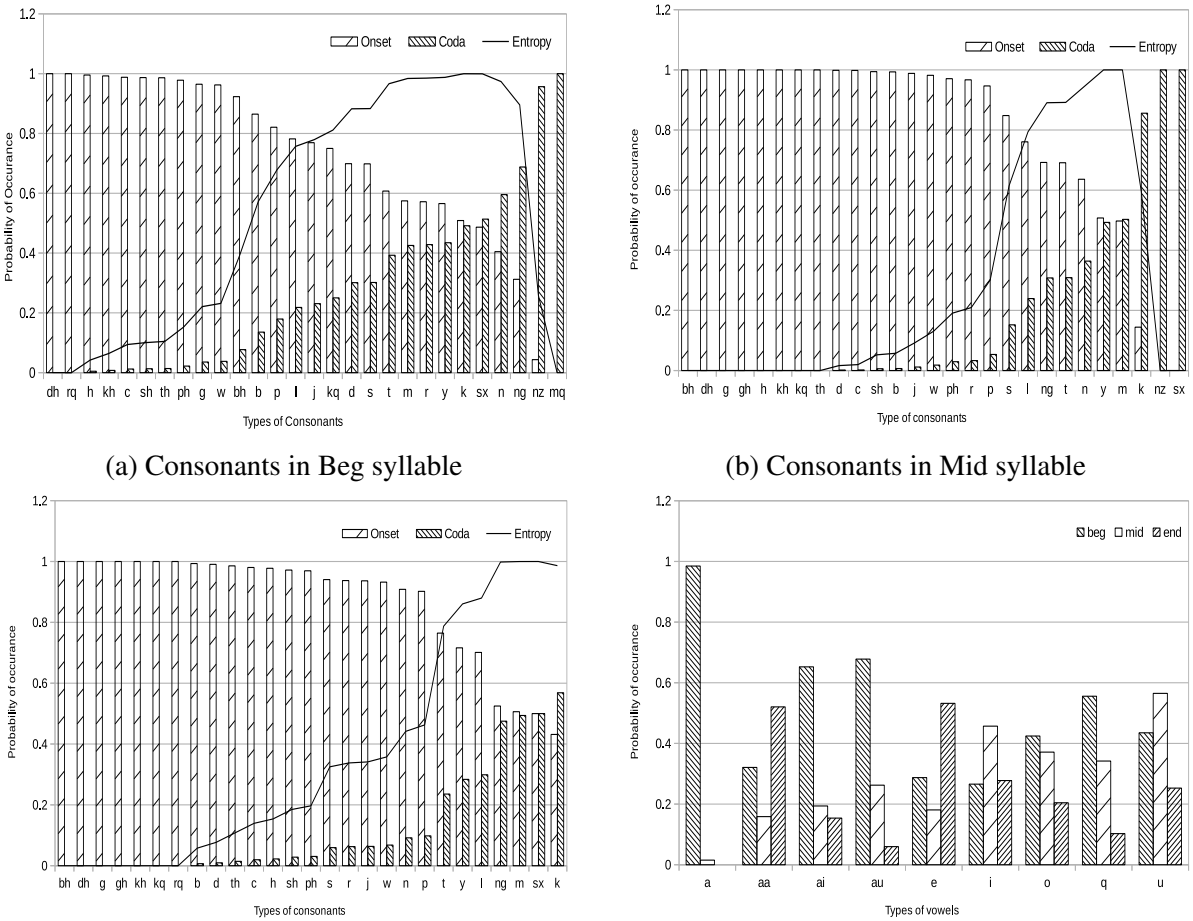
Table 1 shows the types of syllable structure with their percentage distribution of syllabic position (i.e. `beg`, `mid`, `end`) present across words in the corpus where C and V represent consonant and vowel respectively. The dashed line is the partition between the typical 9 syllable structures of Indian languages (covering almost 99.78%) over the nontypical syllable structures found in this corpus. All the root patterns of Manipuri language, as describe in Abbi and Awadhesh (1985), are found in the corpus. The appearance of the nontypical structures are due to the presence of loan words such as *headquarters*, *match*, *annual*, etc.

In Table 1, we observed that CV structure is the most commonly used syllabic structure with more than 59.43% of occurrences, followed by CVC with about 26.1%. We also observed that some syllable structures have dominant positional distribution. For example, V and CVC dominate with `beg`, while CV dominates with `mid` and CCV with `end` syllable position.

4.1 Phonotactics

Phonotactics is an important area of research in phonology which explores the rules governing the phoneme sequence of a language. To understand phonotactics of Manipuri language, we explore the distribution of onset, nucleus and coda across the syllables at different positions.

Figure 1[a,b,c] show the probability of onset and coda distribution for consonant phones at different syllabic positions (beg, mid and end) and their entropy i.e. the information contain in onset and coda probability distribution. A consonant with low entropy shows its structural bias toward either onset or coda. The figures show that a significant number of consonants have structural (onset or coda) as well as positional bias. Further, Figure 1[d] shows the probability distribution of nucleus vowel phones across different syllabic positions (beg, mid and end). It clearly shows that some vowels such as /a/, /au/, /ai/ have positional bias towards `beg`. In the proposed EPS method, discussed in section 5.1.2, the entropy



(a) Consonants in Beg syllable

(b) Consonants in Mid syllable

(c) Consonants in End syllable

(d) Vowels

Figure 1: Probability distribution of different phones

distribution is used to determine syllable boundary.

5 Proposed Methods

In this paper, we propose three different corpus-based automatic syllabification methods; (i) Rule-based approach, (ii) Sequence labelling approach and (iii) Hybrid of rule-based and sequence labelling approaches.

5.1 Rule-based approach

In this section, we discuss two rule-based approaches: (i) Purely based on C*VC* rule of the language and (ii) Corpus driven EPS. Both of them depends on the legality of syllable structures and consonant clusters by checking each phone along with its adjacent units until a legal split point is obtained. These two approaches differ only when a current phone α to be checked is consonant. When α is a dependent vowel and if its previous phone is a consonant then add α to the current syllable phone sequence else there is spelling error in the syllabifying word. When α is an independent vowel, then add it to the current syllable phone sequence only if it satisfies one of the following conditions.

a) If previous phone is one of the following short vowels {/a/, /aa/, /i/, /ii/} and followed by either /u/ or /uu/.

b) If previous phone is a short vowels /aa/ and followed by /o/.

c) If previous two phones do not satisfy condition (a) and previous phone is one of the following vowels {/a/, /aa/, /u/, /uu/, /o/} and followed by either /i/ or /ii/.

Full consonants				
ক = /ka/	খ = /kha/	গ = /ga/	ঘ = /gha/	ঙ = /nga/
চ =/ca/	ছ =/cha/	জ = /ja/	ঝ = /jha/	ঞ = /nza/
ট = ত = /ta/	ঠ = থ = /tha/	ড = দ = /da/	ঢ = ধ = /dha/	ণ = ন = /na/
প = /pa/	ফ = /pha/	ব = /ba/	ভ = /bha/	ম = /ma/
য় = /ya/	র = /ra/	ল = /la/	ৱ = /wa/	ক্ষ= /kq/
শ = /sha/	ষ = /sxa/	স = /sa/	হ = /ha/	

Pure consonants				
ক্ = /k/	খ্ = /kh/	গ্ = /g/	ঘ্ = /gh/	ং = ঙ্ = /ng/
চ্ =/c/	ছ্ =/ch/	জ্ = /j/	ঝ্ = /jh/	ঞ = /nz/
ট্ = ত্ = /t/	ঠ্ = থ্ = /th/	ড্ = দ্ = /d/	ঢ্ = ধ্ = /dh/	ণ্ = ন্ = /n/
প্ = /p/	ফ্ = /ph/	ব্ = /b/	ভ্ = /bh/	ম্ = /m/
য় = /y/	র্ = /r/	ল্ = /l/	ৱ্ = /w/	◌ৃ = /rq/
শ্ = /sh/	ষ্ = /sx/	স্ = /s/	হ্ = /h/	◌ঁ = /mq/

Independent vowels				
অ = /a/	আ = /aa/	ই = /i/	ঈ = /ii/	উ = /u/
ঊ = /u/	এ = /ae/	ঐ = /ai/	ও =/o/	ঔ = /au/

Dependent vowels				
া = /aa/	ি = /i/	ী = /uu/	ু = /u/	ূ = /uu/
ে = /ae/	ৈ = /ai/	ো = /o/	ৌ = /au/	

Table 2: Types of consonants and vowels and their orthographic phoneme representation

Otherwise, mark α as the beginning phone of the new syllable.

5.1.1 Baseline System

Since pure consonant cannot be a stand-alone phone, it must always be merged to either previous or following phone. Assumption in this approach is that a diacritic character (such as halant or virama) is properly placed to distinguish pure consonant from full consonant phone. Table 2 shows different types of consonants and vowels, and their corresponding orthographic representations. In this approach, if the input phone α is a pure consonant and not followed by one of the following semivowels {/y/, /r/, /l/, /w/}, then add α to the current syllable phone sequence, otherwise mark it as the beginning phone of the new syllable. We consider this approach as a *baseline* model to compare with other proposed methods.

5.1.2 Entropy Based Phonotactic Segmentation (EPS)

As the problem of schwa deletion exists in the script, the assumption made for the baseline system does not happen in reality. The proposed approach (described in algorithm 1), is an improvement over the baseline to handle the schwa deletion problem, by using the entropy estimate of phone and cluster of phones. It is designed as a generalized recursive algorithm where the baseline approach is treated as terminating case. Initially for each α, the two parameters β and γ, which are the preceding and following phones or string of phones of α, are taken as `NULL` value.

- `H(x,y)` calculate the cluster entropy of `x` and `y` when both are not `NULL`. If one of them is `NULL`, then the entropy of onset-coda distribution for the other is calculated.
- P_{split}`(x,y)` and P_{merge}`(x,y)` give the probability of splitting and merging respectively when `x` and `y` appear together.
- `CheckSplit(x,y)` function either split or merge `x` and `y` based on P_{split}`(x,y)` and P_{merge}`(x,y)` score.
- x^* represents the expansion of x by adding more context information.
- θ represents a threshold of the entropy.
- `MAX` = 5, since the typical syllable structures are not greater that 5.

5.2 Sequence labeling approach

In this section, we transform the problem of segmenting a word into its syllable units as a problem of sequence labeling task. To apply sequence labeling task on a word in the corpus, every letters present in each word are tagged as `C` or `S` where `C` denotes *continuous character* and *S* denotes *splitting character*. In each manually syllabified word, all letters except the last letter in a syllable are tagged as `C` and the last letter as `S`. For example, the word `Manipuri` which is syllabified as `ma ni pu ri` is tagged as `m/C a/S n/C i/S p/C u/S r/C i/S`.

EPS(β,α,γ)
if Length($\beta + \alpha + \gamma$) $\leq$ MAX then
  if *H(β,α)* $\leq \theta$ **then**
    CheckSplit(β,α)
  end
  else if *H(α,γ)* $\leq \theta$ *and* $P_{split}(\alpha,\gamma) < P_{merge}(\alpha,\gamma)$ **then**
    EPS(β,α + γ,NULL)
  end
  else
    EPS(β^*,α,γ^*)
  end
end
else
  Baseline(α)
end

Algorithm 1: EPS: If α is consonant

Now, considering the observed input sequence $o_1, o_2, ..., o_k$ (a Manipuri word), where o_i denotes an alphabet of Manipuri language (Bengali script in this study), the task is to find the best state sequence i.e., $s_1, s_2, ..., s_k$ that might have resulted the observed sequence where $s_i \in \{C, S\}$. For the word `Manipuri`, the desire output is the sequence $\{$`C S C S C S C S`$\}$.

In this study, we have considered two state-of-art sequence labeling methods namely CRF (Sutton and McCallum, 2006) and MEMM (McCallum et al., 2000). For this experiment, each phone that occurs less than 5 times in the corpus is considered as a rare phone. Each phone that occurs more than 2 times in the corpus will be a feature with their corresponding tags. Those features having a frequency less than 2 will be ignored. The rare phone features having a frequency less than 10 will be ignored and the common phone where feature frequency is more than 250 will form an equivalent class. For improving iterative scaling, 100 iterations have been used and to avoid the overfitting problem L1 regularizer have been employed.

5.3 Hybrid approach

The sequence labeling approach does not verify its output syllable structure with the phonotactic structure of the underlying language. In the proposed hybrid approach, the output of the sequence labeling method is passed to the proposed EPS method to verify the phonotactic structure and rectify the possible error.

6 Syllabification Results

This section discusses the experimental results of different syllabification methods proposed in this paper. 5-fold cross validation has been used to investigate the performance of different labeling methods. *Precision*, *Recall*, *F-measure* and *word accuracy* have been used as the evaluation metrices.

Table 3 compares the performance of different methods in terms of word accuracy, Precision, Recall and F1 measure and shows the distribution of wrongly syllabified words among the loan words and Manipuri words. It clearly shows that all the proposed methods outperform the baseline method. The EPS outperforms the baseline at least by 8.5% in accuracy. Similarly, CRF outperforms both baseline and EPS of about 9.6% and 1% respectively. Interestingly MEMM underperforms all other methods because of the existance of structural bias i.e. label bias problem. EPS and CRF can overcome this problem because they can generalize the global context information. It is also evident from the table that hybrid approach further enhanced the classification accuracy over CRF and EPS of about 0.5% and 1.6%. All the proposed approaches in this paper, were able to handle most of the schwa deletion and loan words problems.

Table 3: Evaluation for different methods

Manipuri Dataset				
Methods	Word Accuracy	Precision	Recall	F1 Score
Baseline	0.875	0.92950	0.93577	0.93212
EPS	0.95992 (+0.085)	0.98007	0.98004	0.98
MEMM	0.73225 (-0.143)	0.80968	0.80519	0.80603
CRF	0.97080 (+0.096)	0.98293	0.98343	0.98287
Hybrid	0.97576 (+0.101)	0.98624	0.98731	0.98653

Assamese Dataset				
Methods	Word Accuracy	Precision	Recall	F1 Score
MEMM	0.33808	0.55486	0.54891	0.54817
CRF	0.95258	0.97798	0.97777	0.97745

Evaluation for different types of word in Manipuri Dataset					
Methods	Word type	Word accuracy	Precision	Recall	F1 Score
Baseline	Loan	0.3227	0.6218	0.6518	0.6339
	Manipuri	0.8909	0.9389	0.9443	0.9412
EPS	Loan	0.7773	0.8791	0.9027	0.8890
	Manipuri	0.9657	0.9846	0.9825	0.9832
MEMM	Loan	0.2182	0.4241	0.4406	0.4289
	Manipuri	0.8100	0.8869	0.8814	0.8828
CRF	Loan	0.7864	0.8874	0.8950	0.8888
	Manipuri	0.9772	0.9868	0.9867	0.9865
Hybrid	Loan	0.8182	0.8961	0.9117	0.9022
	Manipuri	0.9805	0.9894	0.9897	0.9894

6.1 Language Independency

To investigate the capability of CRF handling across different languages, we have further considered another publicly available Assamese dataset consisting of 17,925 unique words. Second part of the Table 3 shows the response of CRF and MEMM over this dataset. It clearly shows that CRF provides a word accuracy of 0.95, while MEMM provides just 0.34.

7 Conclusion

In this paper, we first present generation of Manipuri text corpus using Bengali OCR for linguistics studies. Using the corpus generated from the news articles, we analysed phone characteristics across structural and positional properties. Using the observation from this study, we proposed entropy based phonotactic segmentation (EPS) method (a corpus driven rule based automatic syllabification method) for automatic segmentation of Manipuri words. This approach achieves a word accuracy of 96%. Further, the effect of CRF has been investigated and found to provide a word accuracy of 97%. Then, we combine both the CRF and EPS to enhance the classification accuracy and achieve an accuracy of 98%.

8 Acknowledgment

Authors would like to thank Miss Nanaobi Huidrom, Department of CSE, IIT Guwahati, for her support during corpus generation. This work is funded by Deity, Government of India.

References

Anvita Abbi and K. Mishra Awadhesh. 1985. Consonant clusters and syllabic structures of meitei. *Linguistics of the Tibeto-Burman Area*, 8(2):81–92.

Connie R Adsett and Yannick Marchand. 2009. A comparison of data-driven automatic syllabification methods. In *International Symposium on String Processing and Information Retrieval*, pages 174–181. Springer.

Ashwin Bellur, K Badri Narayan, K Raghava Krishnan, and Hema A Murthy. 2011. Prosody modeling for syllable-based concatenative speech synthesis of hindi and tamil. In *Communications (NCC), 2011 National Conference on*, pages 1–5. IEEE.

Shobhana L Chelliah. 1990. Level-ordered morphology and phonology in manipuri. *Linguistics of the Tibeto-Burman Area*, 13(2):27–72.

Monojit Choudhury, Anupam Basu, and Sudeshna Sarkar. 2004. A diachronic approach for schwa deletion in indo aryan languages. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology: Current Themes in Computational Phonology and Morphology*, pages 20–26. Association for Computational Linguistics.

Liviu P Dinu, Vlad Niculae, and Octavia-Maria Sulea. 2013. Romanian syllabication using machine learning. In *Text, Speech, and Dialogue*, pages 450–456. Springer.

Jeremy Goslin and Ulrich H Frauenfelder. 2001. A comparison of theoretical and human syllabification. *Language and Speech*, 44(4):409–436.

Joan B Hooper. 1972. The syllable in phonological theory. *Language*, pages 525–540.

Daniel Kahn. 2015. *Syllable-based generalizations in English phonology*, volume 15. Routledge.

S Prahallad Kishore and Alan W Black. 2003. Unit size in unit selection speech synthesis. In *INTERSPEECH*.

Anila Susan Kurian, Badri Narayan, Nagarajan Madasamy, Ashwin Bellur, Raghava Krishnan, G Kasthuri, MV Vinodh, Hema A Murthy, and Kishore Prahallad. 2011. Indian language screen readers and syllable based festival text-to-speech synthesis system. In *Proceedings of the Second Workshop on Speech and Language Processing for Assistive Technologies*, pages 63–72. Association for Computational Linguistics.

Yannick Marchand, Connie R Adsett, and Robert I Damper. 2009. Automatic syllabification in english: A comparison of different algorithms. *Language and Speech*, 52(1):1–27.

Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *Icml*, volume 17, pages 591–598.

NP Narendra, K Sreenivasa Rao, Krishnendu Ghosh, Ramu Reddy Vempada, and Sudhamay Maity. 2011. Development of syllable-based text to speech synthesis system in bengali. *International journal of speech technology*, 14(3):167–181.

Ernst Pulgram. 1970. *Syllable, word, nexus, cursus*. Number 81-85. Mouton.

Kseniya Rogova, Kris Demuynck, and Dirk Van Compernolle. 2013. Automatic syllabification using segmental conditional random fields. In *BOOK OF ABSTRACTS OF THE 23RD MEETING OF COMPUTATIONAL LINGUISTICS IN THE NETHERLANDS: CLIN 2013*, page 41. Citeseer.

Elisabeth O Selkirk. 1984. On the major class features and syllable theory. In *Language Sound Structure: Studies in Phonology*, pages 107–136. Cambridge: The MIT Press.

Leihaorambam Sarbajit Singh, Kabita Thaoroijam, and Pradip Kumar Das. 2007. Written manipuri (meiteiron) from phoneme to grapheme. *Language in India*, 7(6).

Charles Sutton and Andrew McCallum. 2006. *An introduction to conditional random fields for relational learning*, volume 2. Introduction to statistical relational learning. MIT Press.

Su-Lin Wu, ED Kingsbury, Nelson Morgan, and Steven Greenberg. 1998. Incorporating information from syllable-length time scales into automatic speech recognition. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 2, pages 721–724. IEEE.

Jian Zhang and Howard J Hamilton. 1997. Learning english syllabification for words. In *International Symposium on Methodologies for Intelligent Systems*, pages 177–186. Springer.

Learning to Distill: The Essence Vector Modeling Framework

Kuan-Yu Chen, Shih-Hung Liu, Berlin Chen*, Hsin-Min Wang

Academia Sinica, Taiwan
*National Taiwan Normal University, Taiwan

{kychen,journey,whm}@iis.sinica.edu.tw, berlin@csie.ntnu.edu.tw*

Abstract

In the context of natural language processing, representation learning has emerged as a newly active research subject because of its excellent performance in many applications. Learning representations of words is a pioneering study in this school of research. However, paragraph (or sentence and document) embedding learning is more suitable/reasonable for some tasks, such as sentiment classification and document summarization. Nevertheless, as far as we are aware, there is relatively less work focusing on the development of unsupervised paragraph embedding methods. Classic paragraph embedding methods infer the representation of a given paragraph by considering all of the words occurring in the paragraph. Consequently, those stop or function words that occur frequently may mislead the embedding learning process to produce a misty paragraph representation. Motivated by these observations, our major contributions in this paper are twofold. First, we propose a novel unsupervised paragraph embedding method, named the essence vector (EV) model, which aims at not only distilling the most representative information from a paragraph but also excluding the general background information to produce a more informative low-dimensional vector representation for the paragraph. We evaluate the proposed EV model on benchmark sentiment classification and multi-document summarization tasks. The experimental results demonstrate the effectiveness and applicability of the proposed embedding method. Second, in view of the increasing importance of spoken content processing, an extension of the EV model, named the denoising essence vector (D-EV) model, is proposed. The D-EV model not only inherits the advantages of the EV model but also can infer a more robust representation for a given spoken paragraph against imperfect speech recognition. The utility of the D-EV model is evaluated on a spoken document summarization task, confirming the practical merits of the proposed embedding method in relation to several well-practiced and state-of-the-art summarization methods.

1 Introduction

Representation learning has gained significant interest of research and experimentation in many machine learning applications because of its remarkable performance. When it comes to the field of natural language processing (NLP), word embedding methods can be viewed as pioneering studies (Bengio et al., 2003; Mikolov et al., 2013; Pennington et al., 2014). The central idea of these methods is to learn continuously distributed vector representations of words using neural networks, which seeks to probe latent semantic and/or syntactic cues that can in turn be used to induce similarity measures among words. A common thread of leveraging word embedding methods to NLP-related tasks is to represent a given paragraph (or sentence and document) by simply taking an average over the word embeddings corresponding to the words occurring in the paragraph. By doing so, this thread of methods has recently

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 358–368, Osaka, Japan, December 11-17 2016.

enjoyed substantial success in many NLP-related tasks (Collobert and Weston, 2008; Tang et al., 2014; Kageback et al., 2014).

Although the empirical effectiveness of word embedding methods has been proven recently, the composite representation for a paragraph (or sentence and document) is a bit queer. Theoretically, paragraph-based representation learning is expected to be more suitable for such tasks as information retrieval, sentiment analysis and document summarization (Huang et al., 2013; Le and Mikolov, 2014; Palangi et al., 2015), to name but a few. However, to the best of our knowledge, unsupervised paragraph embedding has been largely under-explored on these tasks. Classic paragraph embedding methods infer the representation of a given paragraph by considering all of the words occurring in the paragraph. Consequently, those stop or function words that occur frequently in the paragraph may mislead the embedding learning process to produce a misty paragraph representation. In other words, the frequent words or modifiers may overshadow the indicative words, thereby drifting the main theme of the semantic content in the paragraph. As a result, the learned representation for the paragraph might be undesired. In order to address this shortcoming, we propose a novel unsupervised paragraph embedding method, named the essence vector (EV) model, which aims at not only distilling the most representative information from a paragraph but also excluding the general background information to produce a more informative and discriminative low-dimensional vector representation for the paragraph.

On a separate front, with the popularity of the Internet and the increasing development of the digital storage capacity, unprecedented volumes of multimedia information, such as broadcast news, lecture recordings, voice mails and video streams, among others, have been quickly disseminated around the world and shared among people. Consequently, spoken content processing has become an important and urgent demand (Lee and Chen, 2005; Ostendorf, 2008; Liu and Hakkani-Tur, 2011). Obviously, speech is one of the most important sources of information about multimedia (Furui et al., 2012). A common school of processing multimedia is to transcribe the associated spoken content into text or lattice format by an automatic speech recognizer. After that, well-developed text processing frameworks can then be readily applied. However, such imperfect transcripts usually limit the associated efficacy. To bridge the performance gap between perfect and imperfect transcripts, we hence extend the proposed essence vector model to a denoising essence vector (D-EV) model, which not only inherits the advantages of the EV model but also can infer a more robust representation for a given spoken paragraph that is more resilient to imperfect speech recognition.

The remainder of this paper is organized as follows. We first briefly review two classic paragraph embedding methods in Section 2. Section 3 sheds light on our proposed essence vector model and its extension, the denoising essence vector model. Then, a series of experiments are presented in Section 4 to evaluate the proposed representation learning methods. Finally, Section 5 concludes the paper.

2 Literature Review

In contrast to the large body of work on developing various word embedding methods, there are relatively few studies concentrating on learning paragraph representations in an unsupervised manner (Huang et al., 2013; Le and Mikolov, 2014; Chen et al., 2014; Palangi et al., 2015). Representative methods include the distributed memory model (Le and Mikolov, 2014) and the distributed bag-of-words model (Le and Mikolov, 2014; Chen et al., 2014).

2.1 The Distributed Memory Model

The distributed memory (DM) model is inspired and hybridized from the traditional feed-forward neural network language model (NNLM) (Bengio et al., 2003) and the recently proposed word embedding methods (Mikolov et al., 2013). Formally, given a sequence of words, $\{w^1, w^2, \cdots, w^L\}$, the objective function of feed-forward NNLM is to maximize the total log-likelihood,

$$\sum_{l=1}^{L} \log P(w^l | w^{l-n+1}, \cdots, w^{l-1}). \quad (1)$$

Obviously, NNLM is designed to predict the probability of a future word, given its $n-1$ previous words. The input of NNLM is a high-dimensional vector, which is constructed by concatenating (or

taking an average over) the word representations of all words within the context (i.e., $w^{l-n+1}, \cdots, w^{l-1}$), and the output can be viewed as that of a multi-class classifier. By doing so, the n-gram probability can be calculated through a softmax function at the output layer:

$$P\left(w^l \middle| w^{l-n+1}, \cdots, w^{l-1}\right) = \frac{\exp(y_{w^l})}{\sum_{w_i \in V} \exp(y_{w_i})}, \tag{2}$$

where y_{w_i} denotes the output value for word w_i, and V is the vocabulary.

Based on the NNLM, the notion underlying the DM model is that a given paragraph also contributes to the prediction of the next word, given its previous words in the paragraph (Le and Mikolov, 2014). To make the idea work, the training objective function is defined by

$$\sum_{t=1}^{\mathrm{T}} \sum_{l=1}^{L_t} \log P(w^l | w^{l-n+1}, \cdots, w^{l-1}, D_t), \tag{3}$$

where T denotes the number of paragraphs in the training corpus, D_t denotes the t-th paragraph, and L_t is the length of D_t. Since the model acts as a memory unit that remembers what is missing from the current context, it is named the distributed memory (DM) model.

2.2 The Distributed Bag-of-Words Model

Opposite to the DM model, a simplified version is to only rely on the paragraph representation to predict all of the words occurring in the paragraph (Le and Mikolov, 2014; Chen et al., 2014). The training objective function can then be defined by maximizing the predictive probabilities all over the words occurring in the paragraph:

$$\sum_{t=1}^{\mathrm{T}} \sum_{l=1}^{L_t} \log P(w^l | D_t). \tag{4}$$

Since the simplified model ignores the contextual words at the input layer, the model is named the distributed bag-of-words (DBOW) model. In addition to being conceptually simple, the DBOW model only needs to store the softmax weights, whereas the DM model stores both softmax weights and word vectors (Le and Mikolov, 2014).

3 Learning to Distill: The Proposed Essence Vector Modeling Framework

3.1 The Essence Vector Model

Classic paragraph embedding methods infer the representation of a paragraph by considering all of the words occurring in the paragraph. However, we all agree upon that the number of content words in a paragraph is usually less than that of stop or function words. Accordingly, those stop or function words may mislead the representation learning process to produce an ambiguous paragraph representation. In other words, the frequent words or modifiers may overshadow the indicative words, thereby making the learned representation deviate from the main theme of the semantic content expressed in the paragraph. Consequently, the associated capacity will be limited. In order to complement such deficiency, we hence strive to develop a novel unsupervised paragraph embedding method, which aims at not only distilling the most representative information from a paragraph but also diminishing the impact of the general background information (probably predominated by stop or function words), so as to deduce an informative and discriminative low-dimensional vector representation for the paragraph. We henceforth term this novel unsupervised paragraph embedding method the essence vector (EV) model.

To turn the idea into a reality, we begin with an assumption that each paragraph (or sentence and document) can be assembled by two components: the paragraph specific information and the general background information. This assumption also holds in the low-dimensional representation space. Accordingly, the proposed method consists of three modules: a paragraph encoder $f(\cdot)$, which can automatically infer the desired low-dimensional vector representation by considering only the paragraph-specific information; a background encoder $g(\cdot)$, which is used to map the general background information into a low-dimensional representation; and a decoder $h(\cdot)$ that can reconstruct the original paragraph by combining the paragraph representation and the background representation.

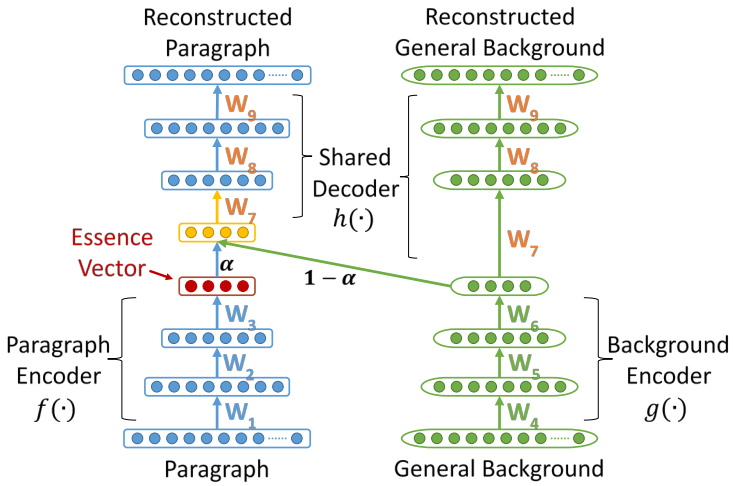


Figure 1: Illustrations of the essence vector model.

More formally, given a set of training paragraphs $\{D_1, \cdots, D_t, \cdots, D_T\}$, in order to modulate the effect of different lengths of paragraphs, each paragraph is first represented by a bag-of-words high-dimensional vector $P_{D_t} \in \mathbb{R}^{|V|}$, where each element corresponds to the frequency count of a word/term in the vocabulary V, and the vector is normalized to unit-sum. Then, a paragraph encoder is applied to extract the most specific information from the paragraph and encapsulate it into a low-dimensional vector representation:

$$f(P_{D_t}) = v_{D_t}. \tag{5}$$

At the same time, the general background is also represented by a high-dimensional vector with normalized word/term frequency counts, $P_{BG} \in \mathbb{R}^{|V|}$, and a background encoder is used to compress the general background information into a low-dimensional vector representation:

$$g(P_{BG}) = v_{BG}. \tag{6}$$

Both $f(\cdot)$ and $g(\cdot)$ are fully connected deep networks with different model parameters θ_f and θ_g, respectively. It is worthy to note that $f(\cdot)$ and $g(\cdot)$ can have same or different architectures. Since each learned paragraph representation v_{D_t} only contains the most informative/discriminative part of P_{D_t}, we assume that the weighted combination of v_{D_t} and v_{BG} can be mapped back to P_{D_t} by a decoder $h(\cdot)$:

$$h(\alpha_{D_t} \cdot v_{D_t} + (1 - \alpha_{D_t}) \cdot v_{BG}) = P'_{D_t}, \tag{7}$$

where $h(\cdot)$ is also a fully connected multilayer neural network with parameter θ_h, and the interpolation weight can be determined by an attention function $q(\cdot,\cdot)$:

$$\alpha_{D_t} = q(v_{D_t}, v_{BG}). \tag{8}$$

The attention function can be realized by a trainable network or a simple linear/non-linear function. Further, to ensure the quality of the learned background representation v_{BG}, it should also be mapped back to P_{BG} by $h(\cdot)$ appropriately:

$$h(v_{BG}) = P'_{BG}. \tag{9}$$

In a nutshell, the training objective function of the proposed essence vector model is to minimize the total KL-divergence measure:

$$\min_{\theta_f, \theta_g, \theta_h} \sum_{t=1}^{T} \left(P_{D_t} \log \frac{P_{D_t}}{P'_{D_t}} + P_{BG} \log \frac{P_{BG}}{P'_{BG}} \right). \tag{10}$$

The activation function used in the EV model is the hyperbolic tangent, except that the output layer in the decoder $h(\cdot)$ is the softmax (Goodfellow et al., 2016), the cosine distance is used to calculate the

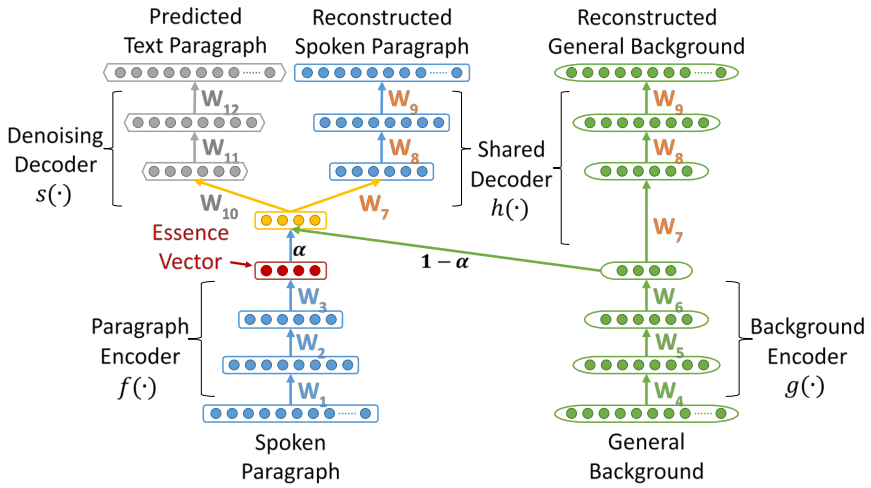


Figure 2: Illustrations of the denoising essence vector model.

attention coefficients, and the Adam (Kingma and Ba, 2015) is employed to solve the optimization problem. At test time, a given paragraph can obtain its own representation by being passed through the paragraph encoder (i.e., $f(\cdot)$). Figure 1 illustrates the architecture of the EV model.

3.2 The Denoising Essence Vector Model

Next, we turn to focus on learning representations for spoken paragraphs. In addition to the stop/function words and modifiers, the additional challenge facing spoken paragraph learning is the imperfect transcripts generated by automatic speech recognition. Therefore, our goal is not only to inherit the advantages of the EV model, but also to infer a more robust representation for a given spoken paragraph that withstands the errors of imperfect transcripts. The core idea is that the learned representation of a spoken paragraph should be able to interpret its corresponding manual transcript paragraph as much as possible. With the intention of equipping the ability that can distill the *true* information from a given *spoken* paragraph, we further incorporate a multi-task learning strategy in the EV modeling framework. To put the idea into a reality, an additional module, a denoising decoder $s(\cdot)$, is introduced on top of the EV model. More formally, given a set of training spoken paragraphs $\{D_1, \cdots, D_t, \cdots, D_{\mathrm{T}}\}$ and their manual transcripts $\{D_1^m, \cdots, D_t^m, \cdots, D_{\mathrm{T}}^m\}$, the EV model can first be constructed by referring to each pair of D_t and the general background information (cf. Section 3.1). Since we target at making the learned paragraph representation v_{D_t} contain the true information of D_t^m, we assume that the weighted combination of v_{D_t} and v_{BG} can also be well mapped back to $P_{D_t^m}$ by the decoder $s(\cdot)$:

$$s\left(\alpha_{D_t} \cdot v_{D_t} + \left(1-\alpha_{D_t}\right) \cdot v_{BG}\right) = P'_{D_t^m}, \tag{11}$$

where $s(\cdot)$ is a fully connected neural network with parameter θ_s. The activation function used in $s(\cdot)$ is the hyperbolic tangent, except that the last layer is the softmax. We will henceforth term this extended unsupervised paragraph embedding method the denoising essence vector (D-EV) model. The training objective of the D-EV model is to minimize the following total KL-divergence measure:

$$\min_{\theta_f,\theta_g,\theta_h,\theta_s} \sum_{t=1}^{\mathrm{T}} \left(P_{D_t} \log \frac{P_{D_t}}{P'_{D_t}} + P_{D_t^m} \log \frac{P_{D_t^m}}{P'_{D_t^m}} + P_{BG} \log \frac{P_{BG}}{P'_{BG}} \right). \tag{12}$$

Figure 2 illustrates the architecture of the proposed D-EV model.

	Books	DVD	Electronics	Kitchen	Average
PCA	0.762	0.769	0.807	0.824	0.790
EV	**0.796**	**0.812**	**0.839**	**0.858**	**0.826**
Unigrams	0.797	0.805	0.837	0.860	0.824
Bigrams	0.798	0.779	0.819	0.857	0.813
Unigrams+Bigrams	0.810	0.821	0.852	0.884	0.842
Unigrams+PCA	0.799	0.812	0.835	0.860	0.826
Unigrams+EV	0.806	0.813	0.833	0.871	0.831
Unigrams+Bigrams+PCA	0.810	0.821	0.852	0.884	0.842
Unigrams+Bigrams+EV	**0.838**	**0.824**	**0.862**	**0.890**	**0.853**

Table 1: Experimental results on sentiment analysis achieved by the proposed EV model and other baseline features, including unigrams, bigrams, PCA, and the combinations.

4 Experimental Setup & Results

4.1 Experiments on the EV Model for Sentiment Analysis

At the outset, we evaluate the proposed EV model on the sentiment polarity classification task. Four widely-used benchmark multi-domain sentiment datasets are used in this study[1] (Blitzer et al., 2007). They are product reviews taken from Amazon.com in four different domains: Books, DVD, Electronics, and Kitchen. Each of the reviews, ranging from Star-1 to Star-5, were rated by a customer. The reviews with Star-1 and Star-2 were labelled as Negative, and those with Star-4 and Star-5 were labeled as Positive. Each of the four datasets contains 1,000 positive reviews, 1,000 negative reviews, and a number of unlabeled reviews. Labeled reviews in each domain are randomly split up into ten folds (with nine folds serving as the training set and the remaining one as the test set). All of the following results are reported in terms of an average accuracy of ten-fold cross validation. The linear kernel SVM (Chang and Lin, 2011) is used as our classifier and all of the parameters are set to the default values. All of the unlabeled reviews are used to obtain the general background information and train the EV model.

In this set of experiments, we first compare the EV model with PCA (Bengio et al., 2013), which is a standard dimension reduction method. It is worthy to note that PCA is a variation of an auto-encoder (Bengio et al., 2013) method; thus it can be treated as our baseline system. All of the experimental results are listed in Table 1. As expected, the proposed EV model consistently outperforms PCA in every domain by a significant margin. The reason might be that PCA maps data to a low-dimensional space by maximizing the statistical variance of data, but the implicitly denoising strategy and the linear formulation limit its model capability. On the contrary, the proposed EV model is designed to distill the most useful information from a given paragraph and exclude the general background information explicitly; it thus can deduce a more informative and discriminative representation.

Next, we make a step forward to compare the EV model with other baseline systems based on literal bag-of-words features, including unigrams and bigrams. The results are also shown in Table 1. Several observations can be drawn from the results. First, although bigram features (denotes as Bigrams in Table 1) are believed to be more discriminative than unigram features (denotes as Unigrams in Table 1), the results indicate that Unigrams outperform Bigrams in most cases. The reason might be probably due to the curse of dimensionality problem. Second, as expected, the combination of unigram and bigram features (denotes as Unigrams+Bigrams) achieves better results than using Unigrams and Bigrams in isolation for all cases. Third, both the proposed EV model and PCA can make further performance gains when paired with Unigrams, Bigrams, and their combination. Fourth, the proposed EV model demonstrates its ability in the sentiment classification task since it consistently outperforms PCA for all cases in the experiments.

[1] https://www.cs.jhu.edu/~mdredze/datasets/sentiment/

		ROUGE-1	ROUGE-2
2001	Peer T	0.330	0.079
	VSM	0.286	0.049
	LexRank	0.334	0.061
	EV	0.332	0.059
	CNN	0.352	0.076
	PriorSum	0.360	0.079
2002	Peer 26	0.352	0.076
	VSM	0.304	0.056
	LexRank	0.353	0.075
	EV	0.354	0.074
	CNN	0.357	0.087
	PriorSum	0.366	0.090
2004	Peer 65	0.379	0.092
	VSM	0.337	0.072
	LexRank	0.379	0.089
	EV	0.376	0.084
	CNN	0.379	0.099
	PriorSum	0.389	0.101

Table 2: Experimental results of multi-document summarization achieved by the proposed EV model and several state-of-the-art summarization methods.

4.2 Experiments on the EV Model for Multi-Document Summarization

We further investigate the capability of the EV model on an extractive multi-document summarization task. In this study, we carry out the experiments with the DUC 2001, 2002, and 2004 datasets[2]. All the documents were compiled from newswires, and were grouped into various thematic clusters. The summary length was limited to 100 words for both DUC 2001 and DUC 2002, and 665 bytes for DUC 2004. The general background information was inferred from the LDC Gigaword corpus[3] (including Associated Press Worldstream (AP), New York Times Newswire Service (NYT), and Xinhua News Agency (XIN)). The most common belief in the document summarization community is that relevance and redundancy are two key factors for generating a concise summary. In this paper, we leverage a density peaks clustering summarization method (Rodriguez and Laio, 2014; Zhang et al., 2015), which can take both relevance and redundancy information into account at the same time. That is, a concise summary for a given document set can be automatically generated through a one-pass process instead of an iterative process. Recently, the summarization method has proven its empirical effectiveness (Zhang et al., 2015). For evaluation, we adopt the widely-used automatic evaluation metric ROUGE (Lin, 2003), and take ROUGE-1 and ROUGE-2 (in F-scores) as the main measures following Cao et al., (2015).

We compare the proposed EV model with two baseline systems (the vector space model (VSM) (Gong and Liu, 2001) and the LexRank (Erkan and Radev, 2004) method), the best peer systems (including Peer T, Peer 26, and Peer 65) participating DUC evaluations, and the recently elaborated DNN-based systems (including CNN and PriorSum) (Cao et al., 2015). Owing to the space limitation, we omit the detailed introduction to these summarization methods; interested readers may refer to Penn and Zhu (2008), Liu and Hakkani-Tur (2011), Nenkova and McKeown (2011), and Cao et al., (2015) for more in-depth elaboration. It is worthy to note that the proposed EV model, the two baseline systems, and the best peer systems are unsupervised methods, while the DNN-based systems are supervised ones. The experimental results are listed in Table 2. Several interesting observations can be concluded from the results. First, the proposed EV model outperforms VSM by a large margin in all cases, and performs comparably to other well-designed unsupervised summarization methods. Second, both LexRank and EV (with the density peaks clustering method) take pairwise information into account globally, so their results are almost the same. Third, although the proposed EV model is an unsupervised method and is

[2] http://www-nlpir.nist.gov/projects/duc/
[3] https://catalog.ldc.upenn.edu/LDC2011T07

	ROUGE-1	ROUGE-2	ROUGE-L
DM	0.387	0.242	0.337
DBOW	0.396	0.250	0.344
EV	**0.414**	0.264	0.361
D-EV	**0.414**	**0.278**	**0.374**
MRW	0.332	0.191	0.291
LexRank	0.305	0.146	0.254
SM	0.332	0.204	0.303
ILP	0.348	0.209	0.306

Table 3: Experimental results of spoken document summarization achieved by the proposed EV and D-EV models and several state-of-the-art summarization methods.

not specifically designed toward summarization, it almost achieves the same performance level as the complicated DNN-based supervised methods (i.e., CNN and PriorSum), which confirms the power of the EV model again.

4.3 Experiments on the D-EV Model for Spoken Document Summarization

In order to assess the utility of the proposed D-EV model, we perform a series of experiments on the extractive spoken document summarization task. All of experiments are conducted on a Mandarin benchmark broadcast new corpus[4] (Wang et al., 2005). The MATBN dataset is publicly available and has been widely used to evaluate several NLP-related tasks, including speech recognition (Chien, 2015), information retrieval (Huang and Wu, 2007) and summarization (Liu et al., 2015). As such, we follow the experimental setting used in previous studies for speech summarization in the literature. The vocabulary size is about 72 thousand words. The average word error rate of the automatic transcripts of these broadcast news documents is about 38%. The reference summaries were generated by ranking the sentences in the manual transcript of a broadcast news document by importance without assigning a score to each sentence. Each document has three reference summaries annotated by three subjects. For the assessment of summarization performance, we adopt the commonly-used ROUGE metric (Lin, 2003), and take ROUGE-1, ROUGE-2 and ROUGE-L (in F-scores) as the main measures. The summarization ratio is set to 10%. An external set of about 100,000 text news documents, which was assembled by the Central News Agency (CNA) during the same period as the broadcast news documents to be summarized (extracted from the Chinese Gigaword Corpus[5] released by LDC), is used to obtain the background representation.

To begin with, we compare the performance levels of the proposed EV and D-EV models and two classic paragraph embedding methods (i.e., DM and DBOW) for spoken document summarization. All the models are paired with the density peaks clustering summarization method. The results are shown in Table 3, from which several observations can be drawn. First, DBOW outperforms DM in our experiments, though DBOW is a simplified version of DM. Second, the proposed EV model outperforms DM and DBOW by a large margin, as expected. The results confirm that EV can modulate the impact of those stop or function words when inferring representations for paragraphs. That is to say, the proposed paragraph embedding method EV can indeed distill the most important aspects of a given paragraph and meanwhile suppress the impact of the general background information for producing a more discriminative paragraph representation. Thus, the relevance degree between any pair of sentence and document representations can be estimated more accurately. Third, the D-EV model consistently outperforms other paragraph embedding methods, including our own EV model. The outcome reveals that, although EV can achieve better performance than other classic paragraph embedding methods, the recognition errors inevitably make the inferred representations deviate from the original semantic content of spoken paragraphs. Accordingly, the results signal that the D-EV model can complement the

[4] http://slam.iis.sinica.edu.tw/corpus/MATBN-corpus.htm
[5] https://catalog.ldc.upenn.edu/LDC2011T13

deficiency of the EV model in spoken document summarization; we thus believe that it is more suitable for use in spoken content processing.

In the last set of experiments, we compare the results mentioned above with that of several well-practiced, state-of-the-art unsupervised summarization methods, including the graph-based methods (i.e., the Markov random walk (MRW) method (Wan and Yang, 2008) and the LexRank method (Erkan and Radev, 2004)) and the combinatorial optimization methods (i.e., the submodularity-based (SM) method (Lin and Bilmes, 2010) and the integer linear programming (ILP) method (Riedhammer et al., 2010)). Among them, the ability of reducing redundant information has been aptly incorporated into the submodular-based method and the ILP method. Interested readers may refer to Penn and Zhu (2008), Liu and Hakkani-Tur (2011), and Nenkova and McKeown (2011) for comprehensive reviews and new insights into the major methods that have been developed and applied with good success to a wide range of spoken document summarization tasks. The results are also listed in Table 3. Several noteworthy observations can be drawn from the results of these methods. First, although the two graph-based methods (i.e., MRW and LexRank) have similar motivations, MRW outperforms LexRank by a large margin. Second, although both SM and ILP have the ability to reduce redundant information when selecting indicative sentences to form a summary for a given document, ILP consistently outperforms SM. The reason might be that ILP performs a global optimization process to select representative sentences, whereas SM chooses sentences with a recursive strategy. Comparing the results of these strong baseline systems to that of the paragraph embedding methods (including DM, DBOW, EV, and D-EV) paired with the density peaks clustering summarization method, it is clear that all the paragraph embedding methods are better than the baseline methods. The results corroborate that, instead of only considering literal term matching for determining the similarity degree between a pair of sentence and document, incorporating concept (semantic) matching into the similarity measure leads to better performance. In particular, the proposed D-EV model is the most robust among all the methods compared in the paper, which supports the important notion of the proposed "*learning to distilling*" framework. We also want to note that the proposed methods (i.e., EV and D-EV) can also be incorporated with the graph-based methods and the combinatorial optimization methods. We leave this exploration for future work.

5 Conclusions

In this paper, we have proposed a novel paragraph embedding framework, which is embodied with the essence vector (EV) model and the denoising essence vector (D-EV) model, and made a step forward to evaluate the proposed methods on benchmark sentiment classification and document summarization tasks. Experimental results demonstrate that the proposed framework is the most robust among all the methods (including several well-practiced or/and state-of-the-art methods) compared in the paper, thereby indicating the potential of the new paragraph embedding framework. For future work, we will first focus on pairing the (denoising) essence vector model with other summarization methods. Moreover, we will explore other effective ways to integrate extra cues, such as speaker identities and relevance information, into the proposed framework. Furthermore, we also plan to extend the applications of the proposed framework to information retrieval and language modeling, among others.

References

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

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: a review and new perspectives. *Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: domain adaptation for sentiment classification. In *Proceedings of ACL*, pages 187–205.

Ziqiang Cao, Furu Wei, Sujian Li, Wenjie Li, Ming Zhou, and Houfeng Wang. 2015. Learning Summary Prior Representation for Extractive Summarization. In *Proceedings of ACL*, pages 829–833.

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity based reranking for reordering documents and producing summaries. In *Proceedings of SIGIR*, pages 335–336.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(27):1–27.

Kuan-Yu Chen, Hung-Shin Lee, Hsin-Min Wang, and Berlin Chen. 2014. I-vector based language modeling for spoken document retrieval. In *Proceedings of ICASSP*, pages 7083–7088.

Jen-Tzung Chien. 2015. Hierarchical Pitman-Yor-Dirichlet language model. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 23(8): 1259–1272.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.

Gunes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligent Research*, 22(1):457–479.

Sadaoki Furui, Li Deng, Mark Gales, Hermann Ney, and Keiichi Tokuda. 2012. Fundamental technologies in modern speech recognition. *IEEE Signal Processing Magazine*, 29(6):16–17.

Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of SIGIR*, pages 19–25.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning.* Cambridge, MA: MIT Press.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. I*n Proceedings of CIKM*, pages 2333–2338.

Chien-Lin Huang and Chung-Hsien Wu. 2007. Spoken Document Retrieval Using Multi-Level Knowledge and Semantic Verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(8): 2551–2560.

Mikael Kageback, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. 2014. Extractive summarization using continuous vector space models. In *Proceedings of CVSC*, pages 31–39.

Diederik Kingma and Jimmy Ba. 2015. ADAM: A method for stochastic optimization. In *Proceedings of ICLR*, pages 1–15.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of ICML*, pages 1188–1196.

Chin-Yew Lin. 2003. ROUGE: Recall-oriented understudy for gisting evaluation. [Online]. Available: http://haydn.isi.edu/ROUGE/.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of NAACL HLT*, pages 912–920.

Yang Liu and Dilek Hakkani-Tur. 2011. *Speech summarization*. *Chapter 13* in *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. G. Tur and R. D. Mori (Eds), New York: Wiley.

Shih-Hung Liu, Kuan-Yu Chen, Berlin Chen, Hsin-Min Wang, Hsu-Chun Yen, and Wen-Lian Hsu. 2015. Combining relevance language modeling and clarity measure for extractive speech summarization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(6): 957–969.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*, pages 1–12.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2–3): 103–233.

Mari Ostendorf. 2008. Speech technology and information access. *IEEE Signal Processing Magazine*, 25(3):150–152.

Lin-shan Lee and Berlin Chen. 2005. Spoken document understanding and organization. *IEEE Signal Processing Magazine*, 22(5):42–60.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2015. Deep sentence embedding using the long short term memory network: analysis and application to information retrieval. In *Proceedings of arXiv:1502.06922*.

Gerald Penn and Xiaodan Zhu. 2008. A critical reassessment of evaluation baselines for speech summarization. In *Proceedings of ACL*, pages 470–478.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vector for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-Tur. 2010. Long story short - Global unsupervised models for keyphrase based meeting summarization. *Speech Communication*, 52(10):801–815.

Alex Rodriguez and Alessandro Laio. 2014. Clustering by fast search and find of density peaks. *Science*, 344(6191): 1492–1496.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of ACL*, pages 1555–1565.

Juan-Manuel Torres-Moreno (Eds.). 2014. *Automatic text summarization*. WILEY-ISTE.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. In *Proceedings of SIGIR*, pages 299–306.

Hsin-Min Wang, Berlin Chen, Jen-Wei Kuo, and Shih-Sian Cheng. 2005. MATBN: A Mandarin Chinese broadcast news corpus. *International Journal of Computational Linguistics and Chinese Language Processing*, 10(2):219–236.

Yang Zhang, Yunqing Xia, Yi Liu, and Wenmin Wang. 2015. Clustering sentences with density peaks for multi-document summarization. In *Proceedings of NAACL*, pages 1262–1267.

Continuous Expressive Speaking Styles Synthesis based on CVSM and MR-HMM

Jaime Lorenzo-Trueba [1,3]
Speech Technology Group[1]
Universidad Politecnica de Madrid
Spain
jaime@nii.ac.jp

Roberto Barra-Chicote [1]
Speech Technology Group[1]
Universidad Politecnica de Madrid
Spain

Ascension Gallardo-Antolin [2]
Signal Theory and Communications[2]
Universidad Carlos III de Madrid
Spain

Junichi Yamagishi [3]
National Institute of Informatics[3]
Tokyo
Japan

Juan M. Montero[1]
Speech Technology Group[1]
Universidad Politecnica de Madrid
Spain

Abstract

This paper introduces a continuous system capable of automatically producing the most adequate speaking style to synthesize a desired target text. This is done thanks to a joint modeling of the acoustic and lexical parameters of the speaker models by adapting the CVSM projection of the training texts using MR-HMM techniques. As such, we consider that as long as sufficient variety in the training data is available, we should be able to model a continuous lexical space into a continuous acoustic space. The proposed continuous automatic text to speech system was evaluated by means of a perceptual evaluation in order to compare them with traditional approaches to the task. The system proved to be capable of conveying the correct expressiveness (average adequacy of 3.6) with an expressive strength comparable to oracle traditional expressive speech synthesis (average of 3.6) although with a drop in speech quality mainly due to the semi-continuous nature of the data (average quality of 2.9). This means that the proposed system is capable of improving traditional neutral systems without requiring any additional user interaction.

1 Introduction

It is clear that in recent times there has been an increase in the penetration rates of speech technologies and applications based in speech recognition or speech synthesis are more and more common. Speech synthesis systems in particular have improved greatly in terms of speech intelligibility, speech quality or naturalness in neutral read speech situations regardless of the technology (Barra-Chicote, 2011). The problem appears when one needs to develop applications such as dialogue systems or robotic interfaces, for which a more expressive way of speaking is more appropriate.

One of the main problems regarding expressive speech synthesis is the vast amount of possibilities that have to be taken into account. Human expressiveness is not a discrete space but a continuous one, and speaking styles vary greatly from person to person and even from time period to time period. As such, obtaining enough data to cover all the requirements can become a very difficult task and scalability a significant problem. This is why statistical parametric speech synthesis is better fitted to the task. While unit-selection based systems have been proven to be more than capable of providing good quality

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 369–376, Osaka, Japan, December 11-17 2016.

expressive speech (Adell et al., 2012; Andersson et al., 2010), the adaptability of HMM-based systems allows us to better face the scalability problem.

The objective of the present paper is to introduce a complete TTS system capable of predicting the most adequate speaking style in order to automatically adapt the produced voice to the target text. For that objective we propose a continuous system based both in Continuous Vector Space Modeling (CVSM) (Tonta and Darvish, 2010; Klein et al., 2011) and Multi-regression HMM (MR-HMM) (Fujinaga et al., 2001), CVSM to model the expressive training texts as a continuous space and MR-HMM to make use of that continuous space as auxiliary features for the HMM modeling. This results in a system capable of characterizing input texts as a vector, which at the same time ends up producing the adequate acoustic model associated to the input text. As such, as long as sufficient variety in the training data is available, we would be able to model a continuous lexical space into a continuous acoustic space.

The rest of the paper is organized as follows. In section 2 we describe corpus considered for training the system. Section 3 gives a theoretical background to the proposed system and then explains in detail the proposed method. Then, section 4 describes the perceptual evaluations environment, whose results are described in section 4.1. Finally in section 5 we present the conclusions to be drawn from this paper together with some ideas for future work.

2 Speech Corpus

For the present research we wanted to combine text processing techniques and speech synthesis techniques, so the considered corpus had to not only cover a number of speaking styles with a reasonable amount of text data, but also consist of speech from a single speaker, so that the synthesis models could provide high quality synthetic speech. For that reason we utilized our self-designed and recorded database: Spanish Speaking Styles.

2.1 Spanish Speaking Styles

Spanish Speaking Styles (SSS) is a speaking styles corpus recorded for a single male professional speaker in 4 different speaking styles, with about 1 hour of speech per speaking style. The 4 speaking styles (news broadcasting, interviews, live sports broadcasts and political speech) cover a large spectrum of the expressive map (as can be seen in the F0-rhythm map that we can see in figure 1) while being recognizable.

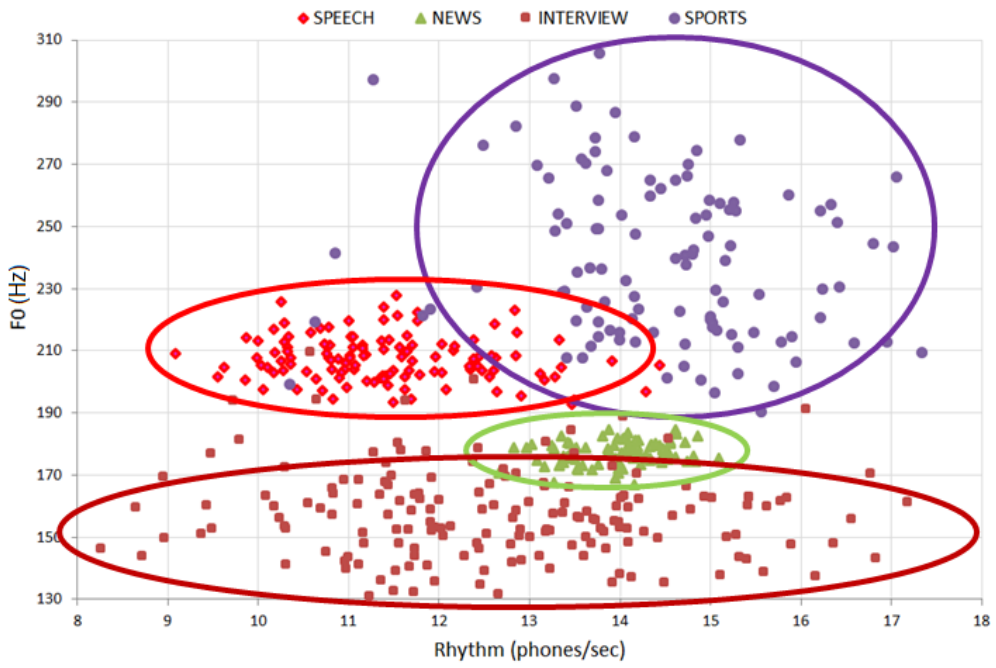


Figure 1: F0 vs. rhythm map of the 4 recorded speaking styles in SSS.

Two of the speaking styles (news and political speech) are scripted and the other two (live sports broadcasts and interviews) unscripted. A summary of the database can be seen in table 1.

Speaking Style	Train set	Test set
NEWS	102 utts, 1h3min	21 utts, 11min
INTERVIEW	332 utts, 45min	21 utts, 7min
SPORT	200 utts, 56min	21 utts, 7min
POLITICAL SPEECH	116 utts, 56min	21 utts, 11min
TOTAL	750 utts, 3h40min	84 utts, 36min

Table 1: Detailed description of the SSS database.

3 Expressive Speech Synthesis based on CVSM and MR-HMM

The ideal expressive TTS system should be able to handle any kind and any number of expressiveness without requiring costly labeling or manipulations every time we want to add any new one. With this purpose in mind the concept of a continuous system (i.e. a system in which expressiveness is treated as a complete space instead of as discrete entities) fits perfectly. That is, in real life expressiveness present overlap between them because there are not clear frontiers, so being able to take into account those overlaps in the shape of a dynamic synthesizer would be ideal.

3.1 Continuous Vector Space Modeling

Traditional information retrieval techniques such as TF-IDF (Fautsch and Savoy, 2010) are commonly based on the assumption that all the terms of the vocabulary lists of the documents do not have any relationship to each other, which is ultimately false as language has semantic relationships that we should be able to model (Tonta and Darvish, 2010; Klein et al., 2011). With that consideration in mind Latent Semantic Analysis (LSA) (Deerwester et al., 1990; Landauer et al., 2013), nowadays referred to as Continuous Vector Space Modeling (Krishnamurthy and Mitchell, 2013; Andreas and Ghahramani, 2013) was born.

CVSM aims to exploit the relationships between terms t_i and documents d_j by transforming them into an alternate "semantic" vector space, where both terms and documents are described by vectors of similar dimensionality and directions, enabling direct comparisons (Olmos et al., 2013; Cosma and Joy, 2012).

Typically this step is done by means of Singular Value Decomposition (SVD) (Golub and Reinsch, 1970; Henry et al., 2010), through which the latent semantic structure of the WTDM is shown.

3.2 Multi-Regression HMM

Multi-regression HMM is a particular kind of adaptation in which the model parameters are adapted depending on auxiliary features instead of the acoustic features themselves. Initially this was developed to exploit the correlation between F0 and the spectral features (Fujinaga et al., 2001), which significantly improved isolated word recognition rates in a speech recognition task. Numerically speaking, the multi-regression formula for M auxiliary features is defined as follows:

$$\mu = r_0 + r_1\epsilon_1 + \ldots + r_1\epsilon_M \tag{1}$$

Where $r_{0\ldots M}$ are the regression coefficients and $r_{\epsilon\ldots M}$ the M auxiliary features. This particularity can be exploited in speech synthesis to model additional information in the speaker models such as speaking styles or emotions (Nose and Kobayashi, 2012; Ling et al., 2013), and it has been used for applications as varied as generating walking motion models (Niwase et al., 2005).

3.3 Proposed System

Figure 2 shows the proposed flowchart for the continuous, MRHMM-based system. There we can see many fundamental differences with the discrete and semi-continuous approach, both at training time and at synthesis time.

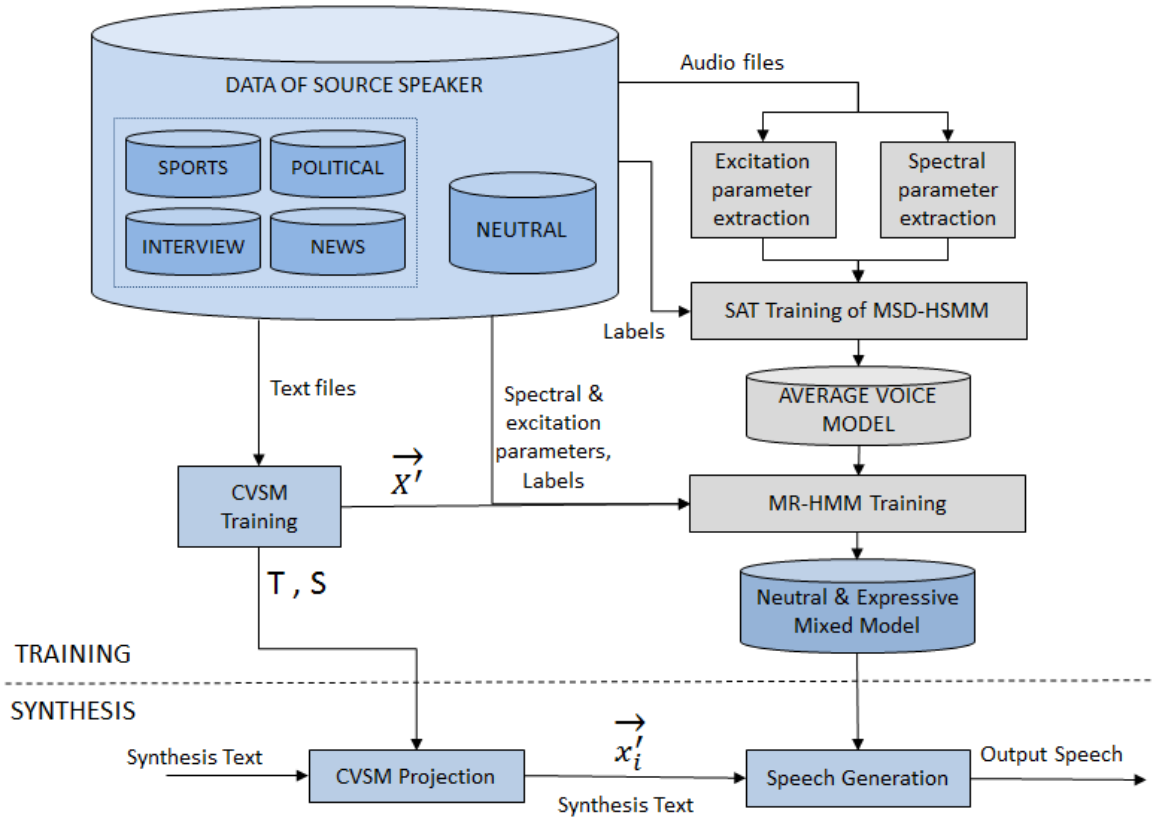


Figure 2: Schematic of the proposed continuous approach to the expressive TTS system.

Most notably, there is no traditional adaptation process involved in the system, but a MR-HMM adaptation that takes the CVSM projections of each training text file as the control vector $\vec{x}'$ for a training process that outputs a new model that combines both neutral and expressive information. Also we can see how there is no need for a centroid estimation, as there is no genre prediction carried out at synthesis time. Instead, only the CSVM matrices are kept as the output so that at synthesis time the synthesis text can be projected to obtain $\vec{x}'_i$, which will be used directly at the speech generation process. This process is then capable of producing an expressive speech output without relying in any genre prediction system, only in the CVSM projections of the synthesis texts.

4 Perceptual Evaluation

For the proposed perceptual evaluation we considered two approaches to the continuous modeling: a first one that directly utilized the CVSM-projected vectors $\vec{x}'$ and a second one that normalized each component by the maximum of their respective component $\vec{\bar{x}}' = \{v_1/v_{MAX1}, \ldots, v_4/v_{MAX4}\}$ where $v_{MAXi} = max\,(v_i)\,\forall v_i$ in the training data. The second approach was considered in case reducing the dispersion of the control vector values helped the MR-HMM re-estimation process. The same normalization was applied to the synthesis control vectors. In total 8 systems were evaluated (2 versions with 4 speaking styles each), which following the Latin Square approach (Gao, 2005), meant that we needed 8 different utterances to be synthesized for all the systems to be presented to the listeners in a random order without repetitions.

The test itself was carried out by means of a web interface, where the evaluator was presented with a button to play the audio sample and the transcription of the uttered texts. The samples could be played as many times as desired. Then, the listener was asked to rate the utterances in the traditional 5 point MOS evaluation in terms of adequacy of the utterance to the text (from not adequate to very adequate), speech quality (from very bad to very good) and perceived expressive strength when comparing to a hypothetical neutral version (very low to very high).

Regarding the speaker models, the whole train section of SSS database and the neutral speech of the same speaker present in SEV database (Barra-Chicote et al., 2008) were modeled into an average voice model (AVM) by applying Speaker Adaptive Training (SAT) (Anastasakos et al., 1997) with three feature streams with their Δ and Δ^2 coefficients: logarithm of the fundamental frequency (1 coefficient), mel-cepstral analysis coefficients (MCEP, 60 coefficients) and aperiodicity bands (25 coefficients). The models were adapted by means of the CSMAPLR algorithm (Yamagishi et al., 2009).

In terms of the statistical significance of the results, we applied the Wilkoxon Signed-Rank Test for a 95% confidence ratio in order to obtain the error margins. 16 subjects took part in each evaluation to guarantee double coverage of the Latin square matrix.

4.1 Evaluation Results

Figures 4 to 3 show the results for both continuous system evaluations (C_Normalized for the normalized version and Continuous for the non-normalized one), together with neutral speech (N), traditional emotional system (S) and natural voice (NAT). The results for the non-continuous systems have been extracted from other works to serve as a reference. It is important to emphasize that this evaluation considered only utterances that could be synthesized, which represented a 65% for the normalized version and 60% for the non-normalized one, details for each system-style interaction can be seen in table 2.

Synth. Rate	C_Normalized	Continuous
Interview	48%	43%
News	81%	67%
Speech	81%	90%
Sports	52%	38%
Average	65%	60%

Table 2: Percentage of synthesizable test sentences for each system-style pair. C_Normalized represents the normalized implementation of the continuous system and Continuous the non-normalized one.

Speech quality (figure 3) does show some bad results, an average quality of 2.93 for the basic system and 2.73 for the normalized system, which is significantly worse than all other systems. This is supposed to be mainly because the continuous system introduces a large amount of artifacts. This effect was somewhat expected due to the inherent semi-continuous nature of the SSS database: the professional speaker was asked to interpret the four speaking styles showing as little variation as possible so that they were clear representatives of their paralinguistic characteristics, so modeling them in a continuous fashion is not possible without additional data. More varied data or a more naturally continuous task is expected to fare better for the continuous system.

On the other hand, in the adequacy results (figure 4) we can see how the system provides significant increases when compared to traditional neutral systems. In the case of the normalized system, the average adequacy is 3.47 and 3.60 for the non-normalized version, not significantly worse than the 3.78 of the traditional expressive synthesis. Interviews, due to its extremely conversational nature was not modeled adequately.

Finally perceived expressive intensity results (figure 5) show a similar image to adequacy. Significantly better than using a neutral system, both proposed continuous systems show a 3.51 average perceived expressive intensity, which is once again comparable to the 3.65 of the non-predictive speech synthesis. Even so the systems are far from the 4.07 of natural speech and are much better than the 2.51 of neutral speech.

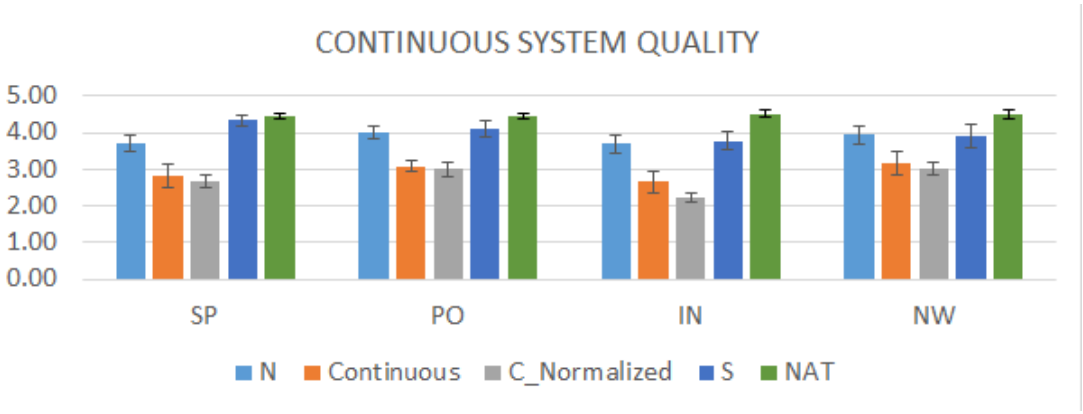


Figure 3: Results in MOS scale of the quality evaluation for the continuous system.

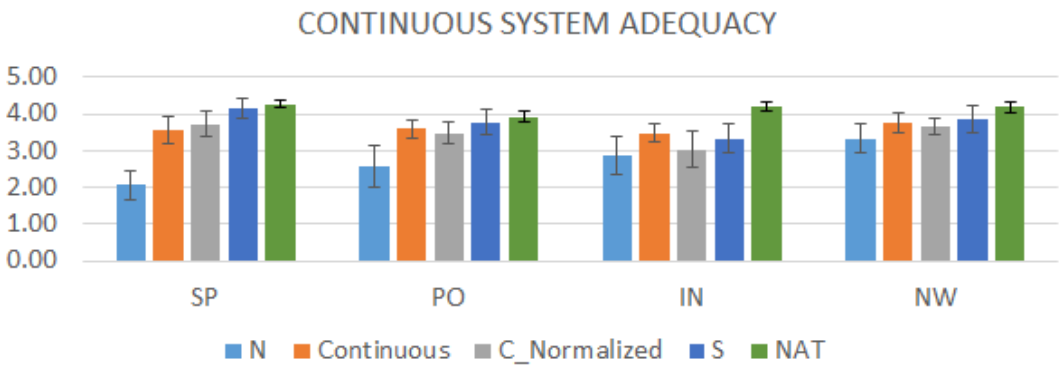


Figure 4: Results in MOS scale of the adequacy evaluation for the continuous system.

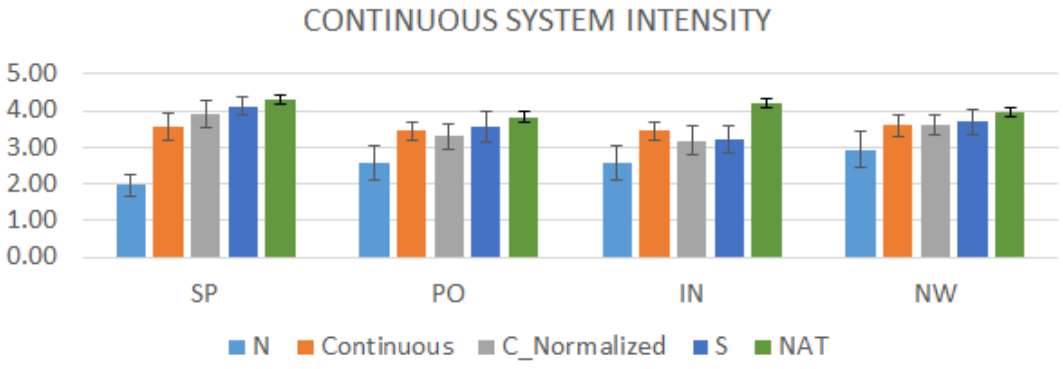


Figure 5: Results in MOS scale of the expressive intensity evaluation for the continuous system.

5 Conclusions and Future Work

We have introduced a complete TTS system is capable of synthesizing the most adequate expressiveness to the target text without relying in genre prediction techniques, just by making use of the CVSM projection of the input text as a control factor of the speaker model, which vastly increases the versatility of the system as we remove the need for labeling the training data genre. This system proved to be capable of conveying the correct expressiveness (average adequacy of 3.6) with an expressive strength comparable to oracle traditional expressive speech synthesis (average of 3.6) although at a significant drop in speech quality mainly due to the semi-continuous nature of the data (average quality of 2.9).

All in all we have introduced a different approach to expressive speech synthesis where the system automatically adjusts the produced speaking style according to the text to be synthesized without requiring any output from the user besides providing adequate training data. The system has shown that it is capable of significantly improving the traditional neutral speech synthesis systems in the task, and also of providing similar adequacy and perceived expressive strength rates than those of natural voice.

For future work we want to consider a broader array of expressiveness, in order to find a problem where the continuous modeling fits naturally: a more complete speaking styles collection, or even considering sub-spaces of speaking styles, including more conversational speaking styles in an attempt to solve the problems that arose with interviews. Another field that we want to work on is on bringing our systems into the DNNs and RNNs world. This task will prove challenging as there is still not many researches underway on DNN-based expressive speech synthesis. Finally, carrying out evaluations in real life systems such as car navigation systems or robotic assistants in scenarios not as constrained as the ones we evaluated would provide much needed information on how research systems fare in the real world, which would undoubtedly give hints on where more to focus our efforts.

Acknowledgements

The work leading to these results has received funding from the European Union under grant agreement 287678. It has been supported by NAVEGABLE (DPI2014-53525-C3-2-R), ASLP-MULN (TIN2014-54288-C4-1-R) projects and by Spanish Government grant TEC2014-53390-P. Jaime Lorenzo has been funded by Universidad Politécnica de Madrid under grant SBUPM-QTKTZHB. The authors want to specially thank the members of the Speech Technology Group, NAVEGASE and Simple4All, specially Simon and Oliver from CSTR, for the continuous and fruitful discussion on these topics.

References

Jordi Adell, David Escudero, and Antonio Bonafonte. 2012. Production of filled pauses in concatenative speech synthesis based on the underlying fluent sentence. *Speech Communication*, 54(3):459–476.

Tasos Anastasakos, John McDonough, and John Makhoul. 1997. Speaker adaptive training: A maximum likelihood approach to speaker normalization. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 2, pages 1043–1046. IEEE.

Sebastian Andersson, Kallirroi Georgila, David Traum, Matthew Aylett, and Robert AJ Clark. 2010. Prediction and realisation of conversational characteristics by utilising spontaneous speech for unit selection. *Speech Prosody*.

Jacob Andreas and Zoubin Ghahramani. 2013. A generative model of vector space semantics. *ACL 2013*, page 91.

R. Barra-Chicote, J. M. Montero, J. Macias-Guarasa, S. Lufti, J. M. Lucas, F. Fernandez, L. F. D'haro, R. San-Segundo, J. Ferreiros, R. Cordoba, and J. M. Pardo. 2008. Spanish expressive voices: Corpus for emotion research in spanish. *Proc. of LREC*.

Roberto Barra-Chicote. 2011. *Contributions to the analysis, design and evaluation of strategies for corpus-based emotional speech synthesis*. Ph.D. thesis, ETSIT-UPM.

Georgina Cosma and Mike Joy. 2012. An approach to source-code plagiarism detection and investigation using latent semantic analysis. *Computers, IEEE Transactions on*, 61(3):379–394.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.

Claire Fautsch and Jacques Savoy. 2010. Adapting the tf idf vector-space model to domain specific information retrieval. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1708–1712. ACM.

Katsuhisa Fujinaga, Mitsuru Nakai, Hiroshi Shimodaira, and Shigeki Sagayama. 2001. Multiple-regression hidden markov model. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 513–516. IEEE.

Lei Gao, 2005. *Latin Squares in Experimental Design*. Michigan State University.

Gene H Golub and Christian Reinsch. 1970. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420.

ER Henry, J Hofrichter, et al. 2010. Singular value decomposition: application to analysis of experimental data. *Essential Numerical Computer Methods*, 210:81–138.

Richard Klein, Angelo Kyrilov, and Mayya Tokman. 2011. Automated assessment of short free-text responses in computer science using latent semantic analysis. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, pages 158–162. ACM.

Jayant Krishnamurthy and Tom M Mitchell. 2013. Vector space semantic parsing: A framework for compositional vector space models. *ACL 2013*, page 1.

Thomas K Landauer, Danielle S McNamara, Simon Dennis, and Walter Kintsch. 2013. *Handbook of latent semantic analysis*. Psychology Press.

Zhen-Hua Ling, Korin Richmond, and Junichi Yamagishi. 2013. Articulatory control of hmm-based parametric speech synthesis using feature-space-switched multiple regression. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(1):207–219.

Naotake Niwase, Junichi Yamagishi, and Takao Kobayashi. 2005. Human walking motion synthesis with desired pace and stride length based on hsmm. *IEICE transactions on information and systems*, 88(11):2492–2499.

Takashi Nose and Takao Kobayashi. 2012. An intuitive style control technique in hmm-based expressive speech synthesis using subjective style intensity and multiple-regression global variance model. *Speech Communication*.

Ricardo Olmos, José A León, Guillermo Jorge-Botana, and Inmaculada Escudero. 2013. Using latent semantic analysis to grade brief summaries: A study exploring texts at different academic levels. *Literary and linguistic computing*, 28(3):388–403.

Yaşar Tonta and Hamid R Darvish. 2010. Diffusion of latent semantic analysis as a research tool: A social network analysis approach. *Journal of Informetrics*, 4(2):166–174.

J. Yamagishi, T. Kobayashi, Y. Nakano, K. Ogata, and J. Isogai. 2009. Analysis of speaker adaptation algorithms for hmm-based speech synthesis and a constrained smaplr adaptation algorithm. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(1):66–83.

An Automatic Prosody Tagger for Spontaneous Speech

Mónica Domínguez
Universitat Pompeu Fabra
C. Roc Boronat, 138
08018, Barcelona, Spain
monica.dominguez@upf.edu

Mireia Farrús
Universitat Pompeu Fabra
C. Roc Boronat, 138
08018, Barcelona, Spain
mireia.farrus@upf.edu

Leo Wanner
ICREA & UPF
C. Roc Boronat, 138
08018, Barcelona, Spain
leo.wanner@upf.edu

Abstract

Speech prosody is known to be central in advanced communication technologies. However, despite the advances of theoretical studies in speech prosody, so far, no large scale prosody annotated resources that would facilitate empirical research and the development of empirical computational approaches are available. This is to a large extent due to the fact that current common prosody annotation conventions offer a descriptive framework of intonation contours and phrasing based on labels. This makes it difficult to reach a satisfactory inter-annotator agreement during the annotation of gold standard annotations and, subsequently, to create consistent large scale annotations. To address this problem, we present an annotation schema for prominence and boundary labeling of prosodic phrases based upon acoustic parameters and a tagger for prosody annotation at the prosodic phrase level. Evaluation proves that inter-annotator agreement reaches satisfactory values, from 0.60 to 0.80 Cohen's kappa, while the prosody tagger achieves acceptable recall and f-measure figures for five spontaneous samples used in the evaluation of monologue and dialogue formats in English and Spanish. The work presented in this paper is a first step towards a semi-automatic acquisition of large corpora for empirical prosodic analysis.

1 Introduction

Speech prosody is known to be central in advanced communication technologies. It is decisive in structuring the message, stressing parts of the message that the interlocutor considers important, and revealing information about the interlocutor's attitude and affection state (Nooteboom, 1997; Wennerstrom, 2001). However, despite the advances of theoretical studies in speech prosody, so far, no sufficiently large, well-annotated prosody material has been created to support empirical studies and drive the research on empirical techniques for analysis and generation of prosodic cues, especially for application in human-computer interaction technologies. Common annotation conventions, such as the ToBI convention (Beckman et al., 2005), provide a descriptive framework of intonation contours and phrasing based upon labels that are language-dependent and rather subjective, which makes it difficult to reach a satisfactory inter-annotator agreement for creating gold standard annotations to train and evaluate algorithms.

It is, therefore, not surprising that empirical research is still based upon rather small laboratory experiments. A further consequence of the lack of sound universal prosody annotation conventions is that current methodologies applied to speech prosody segmentation are still based upon textual and linguistic units (usually words or syntax) rather than on acoustic and phonological units (prosodic phrases and prosodic words). These limitations become an insurmountable barrier for technologies that aim at grasping prosodic cues in spontaneous speech, where many complex prosodic, linguistic and affective phenomena occur (hesitations, incoherent discourse structure, false starts, continuation rising tunes for holding the floor, expression of emotions, speech acts, prosodic disambiguation, etc.). These inherent characteristics of oral language cannot be dealt with using strategies that belong to written language. For instance, sentences with false starts including a filled pause (e.g., *They've never . . . mmm well, my brother's been to Barcelona*).

To overcome the limitations of the current annotation practice and advance in the derivation of more meaningful communicative units from speech as well as in the generation of more natural synthesized speech in the field of human-machine interaction technologies, we need:

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 377–386, Osaka, Japan, December 11-17 2016.

- a parametric language-independent annotation schema of prosody at the acoustic level that can be used by computational models for automatic segmentation and prominence detection;
- prosody taggers and acoustic feature extractors that distill acoustic features from raw speech signals.

In what follows, we address both tasks. First, we present an annotation schema that is implemented as a modular script, deployed as an extended version of the Praat software (Boersma, 2001) into which a functionality for feature annotation and retrieval is incorporated. Such a feature annotation functionality contributes to the independent modular structure and also helps visualization and manual revision of the output within the same Praat environment. Then, we introduce our prosody tagger. Results on inter-annotator agreement and tagger performance compared to a baseline using only F0 cues show that our work is a relevant contribution to the state of the art in the field of speech prosody processing.

The rest of the paper is structured as follows. Section 2 provides an overview of the theoretical approaches in speech prosody, existing automatic tools for labeling prosody and a brief description of the Praat software used for the implementation of our prosody tagger. Section 3 describes the adapted methodology, which is based on theoretical studies of hierarchical prosody and the annotation schema used for manual annotation and for the implementation of the automatic prosody tagger. Section 4 covers the architecture and technical description of the prosody tagger and Praat's extended functionality developed for feature annotation. Section 5 discusses the inter-annotator agreement and evaluation of the performance of our methodology for automatic segmentation and prominence labeling at the prosodic phrase level. Finally, conclusions and future work are drawn in Section 6.

2 Related Work

Theoretical studies of speech prosody have claimed since the 1980's the hierarchical nature of prosodic events in self-contained units (Selkirk, 1984; Ladd, 2008; Gussenhoven, 1984). They describe intonation as a suprasegmental feature, which goes beyond textual segments and has its own structure and phonology. These theoretical studies, especially the one by Selkirk (1984), describe prosody as a hierarchical entity composed of embedded prosodic levels. In our work, we focus on the prosodic phrase (PPh) and the phenomenon of prominence at the PPh level. Our interest in the PPhs stems from the fact that the PPh level has been shown to correlate with the extended thematic structure advocated by Mel'čuk (2001); see, e.g., Domínguez et al. (2014).[1] This correlation proved to be instrumental for the prediction of expressive prosodic contours (Domínguez et al., 2016a).

Recently, a series of models for a computational representation of the intonation contour have been developed and made available in an open initiative for their comparison and further study under the Common Prosody Platform (CPP)[2] (Prom-On et al., 2016): the *Command-Response* (CR) model, the *Autosegmental-Metrical* (AM) model, the *Task-Dynamic* (TD) model and the *Target Approximation* (TA) model. However, these models are limited to a representation of fundamental frequency (F0) contours, which is known to be only one element of speech prosody (Tseng, 2004).

Regarding automatic annotation for prosody involving machine learning techniques, AuToBI[3] (Rosenberg, 2010) was the first publicly available tool to automatically annotate intonation (again, mainly F0 contours and also breaks) with ToBI labels (Silverman et al., 2010). However, AuToBI has several limitations. Thus, it outputs word-by-word annotation, which is a handicap for obtaining a higher-level representation. Furthermore, it is trained on an English corpus of broadcasting radio news, making it domain- and language-specific. In line with AuToBI, ANALOR[4] (Avanzi et al., 2008) is a tool for semi-automatic annotation of French prosodic structure, trained on a small corpus of radio broadcast.

[1] Contrary to other theories of information structure based upon simple theme-rheme division (Steedman,), Mel'čuk (2001)'s extended thematic structure allows embeddedness of thematicity spans and propositions.

[2] http://commonprosodyplatform.org/

[3] http://eniac.cs.qc.cuny.edu/andrew/autobi/

[4] http://www.lattice.cnrs.fr/Analor.html?lang=fr

Like AuToBI, it is domain- and language-specific, but it allows segmentation of an utterance into major prosodic units.

Praat (Boersma, 2001) is an open-source platform for phonetic research widely used by the speech community for annotation, analysis and synthesis purposes. Praat allows the annotation of sound files by means of tiers. Each tier, which is mapped to the whole time-stamp of the associated sound file, includes interval or point annotations that cannot overlap. Each annotation has a label and this label is the only information that can be included into any annotation. Since the labels cannot be extracted as objects in the main Praat window, no action can be scripted based upon smaller units than tiers. Notwithstanding, Praat is a powerful tool, user-friendly, programmable, freely available, running on many platforms, and actively maintained (Mertens, 2004). Due to all these characteristics, a number of Praat-based tools have appeared over the last decade, among them, e.g., ProsodyPro (Xu, 2013). However, many of these tools create a set of parallel tiers, assigning different labels to these tiers, and then, output extracted acoustic features in a text format for further processing using other platforms. For example, Praaline (Christodoulides, 2014) process the *txt* file externally using the R statistical package.

3 Methodology

The main goal of the present work is the development and implementation of a methodology that serves as a scaffolding upon which further improvements and empirical studies can be built upon. One of the requirements, as introduced before, is that this methodology is versatile in the sense that it is language-independent and is able to describe prosodic cues in natural language using a parametric approach. A second goal of our work is to implement a prosody tagger embedded into a Praat-based platform that allows feature annotation and retrieval of any segment below the tier level. In what follows, we introduce the methodology used in the annotation and implementation of a prosody tagger following a discrete representation based upon normalized acoustic parameters (see Section 3.1). The specific annotation guidelines for manual annotation are detailed in Section 3.2. A technical specification of the implementation of the prosodic tagger is outlined in Section 4.

3.1 Acoustic Parameters in Prosodic Units

A key element in our methodology is the concept of prosody as a multidimensional acoustic entity, which involves F0, intensity, and duration acoustic elements. Our methodology involves a combination of normalized acoustic parameters that has been proven by recent studies to yield better results in the task of the prediction of prosodic labels (Domínguez et al., 2016b).

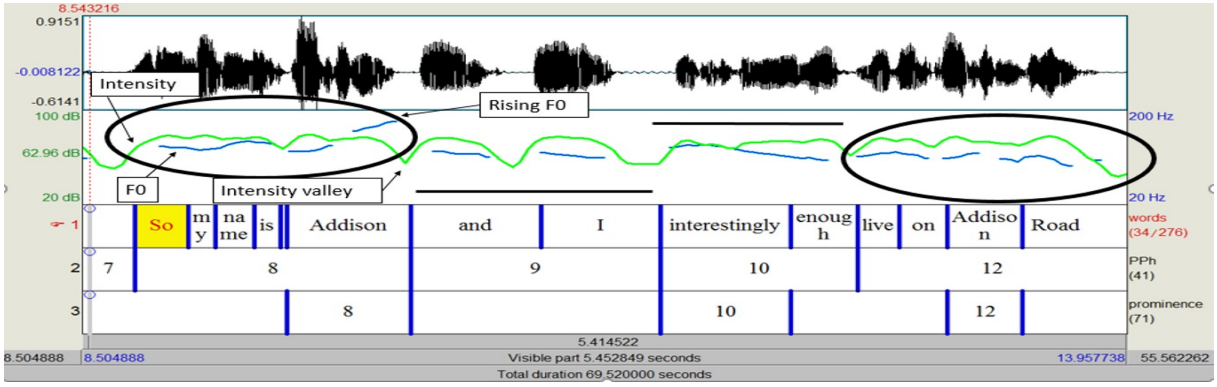


Figure 1: Example of prosodic units.

Figure 1 shows a snapshot of an instance of an utterance containing two propositions (in Mel'čuk (2001)'s terminology). The snapshot corresponds to a continuation rise (L H-H% according to the ToBI convention (Silverman et al., 2010)) in the first proposition and a typical falling tune (H H-L%) in the second. Looking at the graphic representation of the intonation and intensity contours provided by the Praat algorithm, we can clearly observe how these lines form a homogeneous picture (marked by an

ellipse in Figure 1) that corresponds exactly with the PPh division. Nevertheless, as we are dealing with spontaneous speech, there are some areas where the division is not that clear, as can be observed in the central part of the utterance. This homogeneity observed in Figure 1 (which is also auditorily perceived by an expert annotator) can be translated into a vector of normalized acoustic values, which are computed at different levels.

For the representation of the vectors, we adopt a practical approach based upon the actual functions already provided by Praat for speech signal processing. We establish different levels of abstraction, which do not strictly correspond to theoretical representations as such, but follow the idea that prosodic units to be tagged must follow a certain parametric logic in terms of positive or negative deviations at each level in their associated segments. Consequently, we consider as Level 1 (L1) the whole utterance or speech sample; as Level 2 (L2) each voiced segment within the utterance; Level 3 (L3) PPhs once they are segmented. Domínguez et al. (2016b) describe boundary tones in terms of a combination of normalized acoustic parameters, showing also that such a combination performs better than each acoustic element individually for the prediction of prosodic cues.

3.2 Annotation Guidelines

In this section, a set of guidelines for the annotation of prominence and boundaries at the PPh level is introduced. Since the corpus used for evaluation is spontaneous speech, which has inherent difficulties especially for segmentation, specific notes on how to proceed in controversial points are included in these guidelines.

A PPh is defined as a prosodic unit that forms a homogeneous unit in terms of F0 and intensity curves and is signaled by one or a combination of acoustic parameters as outlined in the previous section. A PPh is marked according to the following criteria:

- In case there is one or (usually) a combination of the following conditions: pause, final rising intonation, lengthening of the last word, sharp fall in intensity, a PPh boundary is to be marked.
- In terms of content packaging, a PPh must contain at least one complete concept (usually a predicate with its arguments) of a considerable length relative to the whole utterance and associated voiced segment respectively.
- In spontaneous speech, disfluencies such as disruptions, truncated phrases and hesitations may influence manual labeling of prosodic units. Therefore, all these events are to be included in the closest PPh, unless a pause precedes or follows such disfluencies.
- If the contour following an unvoiced phoneme (with *undefined* F0 value) is perceived as a continuation of the previous F0 contour (forming an homogeneous unit), no boundary is to be inserted. On the contrary, if the F0 contour is significantly different after the F0 phonemic disruption, a boundary is to be marked.

Prominence within each PPh is marked in accordance with the following criteria:

- Prominent words are defined as a combination of one or (usually) several of the following parameters: F0 peak, high intensity, longer duration within its PPh.
- At least one word must be labeled as prominent within each PPh.
- Perceived relevant content must not be used as a criterion to label prosodic prominence (e.g., in noun compounds, an element tends to be perceived more prominent as it carries the semantic meaning of the unit).
- If a combination of acoustic parameters occurs within a word,[5] this word should have more weight than another word showing, for example, an F0 peak and no other acoustic cue.

[5]We refer to textual units in this case, as we are not aiming at segmenting prosodic words yet

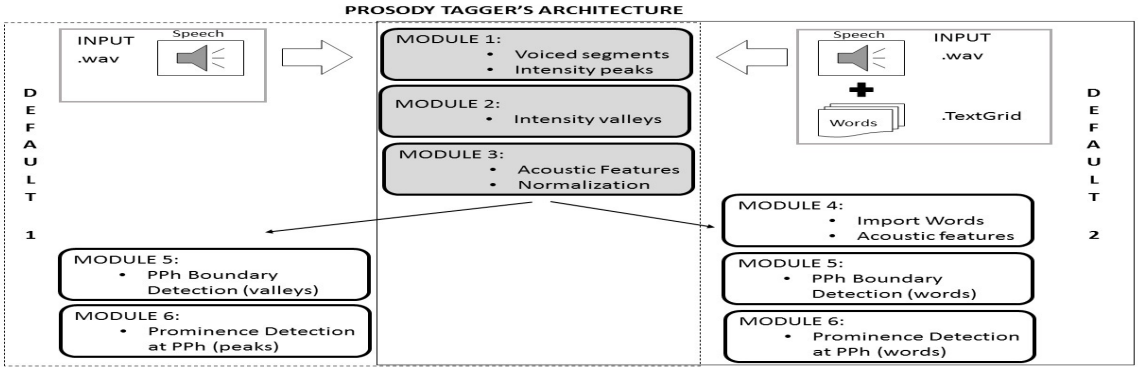


Figure 2: Prosody tagger's architecture.

4 Prosody Tagger Implementation

Our prosody tagger is available as a web service.[6] Any speech sample in *wav* format and associated *TextGrid* with word division can be uploaded, such that the segmentation into PPhs and prominence within the segmented PPhs is displayed on screen and also for download in TextGrid format. All scripts and the extended Praat version for feature annotation are also available under a Creative Common's license[7].

The architecture of the prosody tagger has been conceived as a modular platform such that its optimization and further development (including prosodic word detection) can be attained focusing on specific intermediate steps within the whole pipeline. Acoustic information extracted from different modules is annotated in terms of feature vectors in each segment, including computed z-scores within different prosodic units (so far, from Levels 1 to 3), as introduced in previous sections. Those features can be visualized, retrieved and used for processing at any stage thanks to the extension for feature annotation performed on Praat. Acoustic parameters include, but are not limited to, F0, intensity and duration elements, as Praat allows extraction of a wider range of acoustic parameters (such as jitter, shimmer and pulses, among others).

Figure 2 sketches the modular architecture of the prosody tagger with two possible configurations: (i) Default 1: using only raw audio (as wav file), and (ii) Default 2: using both raw audio (wav file) and importing external word segmentation (in TextGrid format), which must be uploaded by the user. For the Default 2 configuration presented in this study, we have used for word segmentation the proprietary Automatic Speech Recognition system *Scribe*[8] by Vocapia Research.[9] The output of Scribe is converted from xml into TextGrid format. In what follows, a description of each module's functionality is outlined and annotated acoustic features are specified at each stage.

Module 1 uses the wav file and creates a TextGrid using the built-in function in Praat *To TextGrid (silences)*, which automatically detects unvoiced and voiced segments as intervals. Then, a pitch object and an intensity object are extracted from the sound file. The function *To IntensityTier (peaks)* is performed on the intensity object to select salient peaks. The F0 information is extracted at standard frame rates from the pitch object to associate extracted intensity peaks to the ones that involve F0; the distance between these peak candidates is also considered for syllable nuclei detection. A point tier is created and points matching the combination of intensity, F0 and time distance within each voiced segment are annotated. As features, absolute intensity, F0 and the associated voiced interval are stored in each point segment.

Module 2 makes use of the intensity object created in Module 1 to extract intensity valleys using the Praat function *To IntensityTier (valleys)*. Standard intensity frames are selected if their intensity z-score

[6] http://kristina.taln.upf.edu/praatweb/

[7] https://github.com/monikaUPF/modularProsodyTagger

[8] https://scribe.vocapia.com/; Scribe is currently run as a beta version.

[9] http://www.vocapia.com/

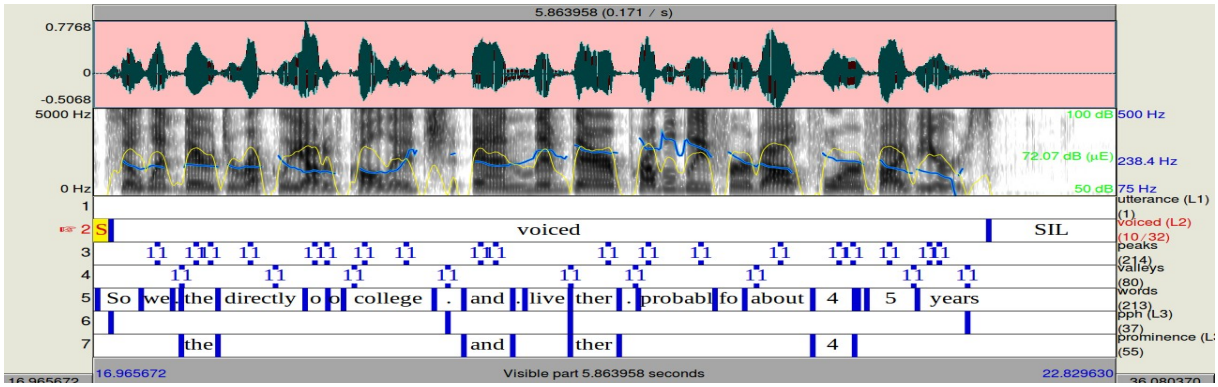


Figure 3: All modules' output for calculation.

(relative to L1) is lower than 0. Then, the lowest values in intensity relative to each voiced fragment (L2) are labeled in a new point tier taking into account the distance between them. Annotated features from this module are: intensity z-score relative to the whole sound (L1) and intensity z-score relative to the associated voiced segment (L2) at each valley point.

Module 3 extracts acoustic values, computes z-scores at available levels, and annotates results as features in each segment. At L1, mean and standard deviation of intensity and F0, together with duration for the whole file, are annotated. These values serve for calculation of z-scores at lower levels in the hierarchy. At L2, annotated features include both absolute values for F0, intensity and duration for further calculation of z-scores in peak and valley tiers (created in Module 1 and 2 respectively) and z-scores derived from L1 values. In the peak and valley tiers, the distance to the previous point is also annotated as a feature. For the first point in the tier, the distance to the boundary of its associated voiced segment is specified as reference.

If a TextGrid with the word segmentation is available, Module 4 exports this tier and annotates features at each marked interval. Consequently, prominence predicted in Module 6 outputs prominent words if these segments are provided by the user. Extracted acoustic parameters and annotated features in this module include: (i) z-scores relative to their associated L2 voiced interval (the z-score values for intensity and F0 are extracted and annotated as features for each word segment obtained by Module 1); (ii) time landmarks, i.e., time of minimum value of intensity and maximum F0 within each word; (iii) duration: absolute duration of the word, and relative duration to the corresponding voiced segment and to the whole sample.

Module 5 uses voiced segments and valleys to predict PPh boundaries. They are derived from the information extracted in the L2 voiced/silence segments detected by Module 1 and from the valleys marked in Module 2. Valleys and peaks contained in each L2 voiced segment are looped in to find the smallest z-score values of intensity within each voiced range and measuring the distance of these valleys to the closest peaks. If the distance of one of the closest peaks is greater than or equal to 0.2 seconds, the z-score is among the minimum in the range, and F0 value is *undefined*, then a PPh boundary is marked.

Finally, Module 6 performs prominence detection on each PPh predicted in the previous module. If no word alignment is available, only syllable peaks predicted in Module 1 are used. Consequently, this module outputs prominent points that correspond to peak points in configuration 1 and prominent word intervals in configuration 2. For calculation of prominence, a combination of F0, intensity and duration cues are taken into account as described in Section 3. Figure 3 displays all tiers created by each module as described above for computation and the final output with the tagging of PPh boundaries and prominent words after the whole pipeline has been executed running a default 2 configuration.

5 Evaluation

A total of five different spontaneous speech samples are used in the evaluation, both for inter-annotator agreement and the prosody tagger's performance: three dialogues in Spanish and two monologues in

American English. Table 1 shows the specific information details for each sample. Dialogues in Spanish are set in a medical context; a male doctor is involved in all of them talking to a patient. Gender is represented in all file names with the convention "f" and "m" for female and male respectively. Files "es_01mm" and "es_02mm" include the same speakers in the same conversational context, where a patient complains to the doctor, but in "es_01mm", the doctor shows a negative response, while in "es_02mm", he acts in a comprehensive and pro-active way. Monologues in English are biographical introductions of the speakers (birthplace, family, recent activities, etc.). Original sound files, transcripts and annotations used in this evaluation will be made available upon request to the authors and acceptance of associated license terms.

		Length	
Filename	Format	Seconds	Words
es_01mm	dialogue	36	196
es_02mm	dialogue	28	150
es_03fm	dialogue	152	545
en_04m	monologue	70	213
en_05f	monologue	30	282
	TOTAL	**316**	**1386**

Table 1: Corpus used in evaluation.

Filename	Prominence	Boundary
es_01mm	0.55	**0.98**
es_02mm	**0.63**	**0.72**
es_03fm	0.51	**0.78**
en_04m	**0.72**	**0.93**
en_05f	**0.69**	**0.70**

Table 2: Inter-annotator agreement Cohen's kappa.

Two expert annotators, proficient in both English and Spanish, have independently labeled both speech samples following the guidelines outlined above in Section 3.2. Cohen's kappa (Cohen, 1960) has been calculated for inter-annotator agreement for each prominence and boundary labeling task. Evaluation is performed on the Default 2 configuration using word segmentation to facilitate the computation and objectiveness of the validation process. A baseline pipeline using only duration and F0 parameters for the same task has been implemented. Inter-annotator kappa results are presented in Section 5.1. The tagger's accuracy, precision and recall compared to the baseline is reported in Section 5.2.

5.1 Inter-annotator agreement

Table 2 provides kappa values for PPh boundary and prominence labeling of our corpus. If annotators label words that are part of the same prosodic word (e.g, they coincide in the final initial word boundary or are separated by a function word, which is usually unstressed), we count this as a partial match for the kappa computation. In order to count matches automatically under Praat, annotators are asked to insert interval boundaries duplicating the word boundaries which are automatically marked, so that we can compare boundary times for the computation of matches.

A kappa within the range of 0.6-0.8 (within a scale between 0 and 1) is considered *satisfactory*, and above 0.8 *perfect* (Cohen, 1960). In Table 2, kappa values that are in line with these thresholds are highlighted in bold for each task (i.e., prominence and boundary labeling within PPh level). Results prove that agreement ranges from 0.51 and 0.98. A higher agreement is observed in the boundary labeling task for all voice samples. No significant differences are observed between English and Spanish samples in boundary detection. However, in prominence labeling, two Spanish samples (files "es_01mm" and "es_-03fm") only reach a *moderate* agreement of 0.55 and 0.51 respectively and, in the overall picture, kappa values for prominence in Spanish are lower than those for English, which might be due to the dialogue format of the samples with shorter interventions, affective states displayed by participants (perceived emotional behavior conveyed by prosody) and quick turn movements between speakers. Nevertheless, we cannot infer that prominence annotation could be language-dependent as such: the corpus we use for evaluation is simply too small for such conclusions. Further research could exploit these techniques, or even a combination of semi-automatic annotation using the prosody tagger presented in this paper, to explore how linguistic parameters such as the discourse type (dialogue in Spanish versus monologue in English), register, gender or speaker idiosyncrasies may affect inter-annotator agreement and tagger's performance in this respect.

5.2 Automatic prosody labeling performance

In order to evaluate the performance of the automatic prosody tagger, we count as full matches those matches that have been labeled as full either by one or by both annotators. For prominence labeling only, we include into the match count also partial matches, i.e., words that coincide in one interval boundary or belong to the same prosodic word – as already done in the inter-annotator agreement exercise. Boundaries that match with a time margin of ± 0.25 seconds are considered to be partial matches. For PPh boundaries, we count it as a match (or true positive), if the automatic tool labels a boundary which has only been labeled by one annotator. Table 3 presents the accuracy, precision, recall and F-measure scores for full matches (F) and full and partial matches (F&P), both for the baseline and our tagger (recall that the baseline uses F0 only).

	Accuracy		Precision		Recall		F-Measure	
	P	B	P	B	P	B	P	B
baseline (F)	0.83	0.89	0.49	0.88	0.22	0.28	0.30	0.42
tagger (F)	**0.84**	0.88	**0.52**	0.58	**0.32**	**0.43**	**0.36**	**0.55**
baseline (F&P)	0.90	0.90	0.84	0.88	0.37	0.28	0.51	0.42
tagger (F&P)	**0.91**	0.89	0.80	0.63	**0.49**	**0.49**	**0.61**	**0.55**

Table 3: Automatic prosody tagger's accuracy, precision and recall.

Table 3 shows that the prosody tagger performs at accuracy rates higher than 0.84 in both prominence (P) and boundary (B) detection tasks. The baseline achieves higher precision figures (especially in boundary detection) than our tagger. A closer look at the output reveals that the baseline marked only those boundaries that included a clear pause, i.e., "safe" candidates. In contrast, the tagger marked not only those clear pauses, but also more subtle boundaries that involved an intensity decrease and not necessarily a pause. On the other side, the tagger reaches considerably higher recall figures than the baseline for both prominence and boundary detection tasks. The F-measure figures show that overall, the tagger performs better. Still, since our methodology is based upon the deviation of normalized values, neutral speech might pose a problem when trying to tag both prominence and boundaries, as there is a tendency towards less variable prosodic cues in this register. Further empirical studies using a semi-automatic approach and optimization of the tagger are needed to have a deeper insight into this and other issues.

6 Conclusions

The integration of a parametric prosody annotation methodology into speech signal processing research is essential in order to reach a close to natural segmentation of spontaneous speech samples and to facilitate the task of the annotation of large corpora for training algorithms for the generation of expressive synthesized speech.

We presented an annotation schema that facilitates a hierarchical acoustic representation of prosody and a tagger that automatically tags speech samples in accordance with this schema. The rather high inter-annotator agreement figures show that our annotation schema is coherent and objective, i.e., does not depend on potentially subjective criteria of the individual annotators. Recall results surpassing the baseline prove that our automatic prosody tagger is flexible enough to support language independent speech signal analysis and detection of prominence and boundaries at the PPh level using a combination of acoustic features, rather than merely F0 contours, as previous empirical and theoretical studies claimed. Improved recall scores also indicate that the number of true positives from the total number of words which actually belong to the positive class, i.e., labeled as positive by the manual annotators, is much higher than the baseline's.

The present implementation may be extended for other applications or further smaller prosodic unit detection (such as prosodic words) due to its modular architecture and open-source philosophy. Moreover, this paper briefly presents an extended Praat functionality for feature annotation to provide easy access and manual revision of the tagger's output as well as retrieval of annotated features at any stage of the computation process. The modularity and prosodically-oriented methodology used in the development of the tagger provides a suitable framework for the deployment of more complex data-driven

approaches, which are able to learn from prosodic units instead of textual units, and thus, get closer to more natural and expressive results when applied to generation of prosodic cues.

All in all, the presented methodology and implementation serves as a platform upon which further research lines and experiments can be run to increase the knowledge in the area of speech technologies and test advanced implementations for human-machine interaction technologies. In the future, we plan to explore the re-implementation of the tagger as a neural network application. Extracted acoustic features combined with linguistic features such as part of speech tag, syntactic dependencies and communicative structure, will be put to the test to observe whether prediction is enhanced, as previously suggested by empirical studies such as (Domínguez et al., 2016b).

Acknowledgements

This work is part of the KRISTINA project, which has received funding from the *European Unions Horizon 2020 Research and Innovation Programme* under the Grant Agreement number H2020-RIA-645012. It has been also partly supported by the Spanish Ministry of Economy and Competitiveness under the Maria de Maeztu Units of Excellence Programme (MDM-2015-0502). The second author is partially funded by the Spanish Ministry of Economy and Competitivity through the *Juan de la Cierva* program. The authors also want to acknowledge Yvan Josse and Bianca Vieru from Vocapia Research for their support on speech to text scripts, and Iván Latorre and Joan Codina from Universitat Pompeu Fabra for their support on the implementation of Praat extension for feature annotation and the "Praat on the Web" tool.

References

M. Avanzi, A. Lacheret-Dujour, and B. Victorri. 2008. ANALOR: A tool for semi-automatic annotation of french prosodic structure. In *Proceedings of the 4th International Conference on Speech Prosody*, pages 119–122, Campinas, Brazil.

M. E. Beckman, J. Hirschberg, and S. Shattuck-Hufnagel. 2005. The original ToBI system and the evolution of the ToBI framework. In S A. Jun, editor, *Prosodic Typology – The Phonology of Intonation and Phrasing*.

P. Boersma. 2001. Praat, a system for doing phonetics by computer. *Glot International*, 5(9/10):341–345.

G. Christodoulides. 2014. Praaline: Integrating tools for speech corpus research. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, Reykjavik, Iceland.

J. Cohen. 1960. A Coefficient of Agreement for Nominal Scales. In Sage Publications Inc., editor, *Educational and Psychological Measurement*, pages 37–46.

M. Domínguez, M. Farrús, A. Burga, and L. Wanner. 2014. The Information StructureProsody Language Interface Revisited. In *Proceedings of the 7th International Conference on Speech Prosody*, pages 539–543, Dublin, Ireland.

M. Domínguez, M. Farrús, A. Burga, and L. Wanner. 2016a. Using hierarchical information structure for prosody prediction in content-to-speech applications. In *Proceedings of the 8th International Conference on Speech Prosody*, pages 1019–1023, Boston, USA.

M. Domínguez, M. Farrús, and L. Wanner. 2016b. Combining acoustic and linguistic features in phrase-oriented prosody prediction. In *Proceedings of the 8th International Conference on Speech Prosody*, pages 796–800, Boston, USA.

C. Gussenhoven. 1984. *On the Grammar and Semantics of Sentence Accents*. Foris, Dordrecht.

R. Ladd. 2008. *Intonational Phonology*. Cambridge University Press, Cambridge.

I. A. Mel'čuk. 2001. *Communicative Organization in Natural Language: The semantic-communicative structure of sentences*. Benjamins, Amsterdam, Philadephia.

P. Mertens. 2004. The Prosogram: Semi-automatic transcription of prosody based on a tonal perception model. In *Proceedings of the 2nd International Conference on Speech Prosody*, pages 549–552, Nara, Japan.

S. Nooteboom. 1997. The prosody of speech: Melody and rhythm. In *The Handbook of Phonetic Sciences*. Blackwell Publishers Ltd, Oxford.

S. Prom-On, Y. Xu, W. Gu, A. Arvaniti, H. Nam, and D. H. Whalen. 2016. The common prosody platform (cpp): Where theories of prosody can be directly compared. In *Proceedings of the 8th International Conference on Speech Prosody*, pages 1–5, Boston, USA.

A. Rosenberg. 2010. AutoBI - A tool for automatic ToBI annotation. In *Proceedings of Interspeech*, pages 146–149, Makuhari, Japan.

E. O. Selkirk. 1984. *Phonology and Syntax: The relation between sound and structure*. The MIT Press, Cambridge, Massachussetts.

K. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, C. Wightman, P. Price, J. Pierrehumbert, and J. Hirschberg. 2010. ToBI: A standard for labeling English prosody. In *Proceedings of Interspeech*, pages 146–149, Makuhari, Japan.

M. Steedman. Information structure and the syntax-phonology interface. In *Linguistic inquiry*, volume 31, pages 649–689. The MIT Press, Cambridge, Massachussetts.

C. Tseng. 2004. Intensity in relation to prosody organization. In *International Symposium on Chinese Spoken Language Processing*, pages 217–220. IEEE.

A. Wennerstrom. 2001. *The Music of Everyday Speech. Prosody and Discourse Analysis*. Oxford University Press, Oxford.

Y. Xu. 2013. Prosodypro a tool for large-scale systematic prosody analysis. In *Proceedings of Tools and Resources for the Analysis of Speech Prosody (TRASP)*, pages 7–10, Aix-en-Provence, France.

Frustratingly Easy Neural Domain Adaptation

Young-Bum Kim
Microsoft
Redmond, WA
ybkim@microsoft.com

Karl Stratos*
Bloomberg L. P.
New York, NY
me@karlstratos.com

Ruhi Sarikaya†
Amazon
Seattle, WA
rsarikay@amazon.com

Abstract

Popular techniques for domain adaptation such as the feature augmentation method of Daumé III (2009) have mostly been considered for sparse binary-valued features, but not for dense real-valued features such as those used in neural networks. In this paper, we describe simple neural extensions of these techniques. First, we propose a natural generalization of the feature augmentation method that uses $K + 1$ LSTMs where one model captures global patterns across all K domains and the remaining K models capture domain-specific information. Second, we propose a novel application of the framework for learning shared structures by Ando and Zhang (2005) to domain adaptation, and also provide a neural extension of their approach. In experiments on slot tagging over 17 domains, our methods give clear performance improvement over Daumé III (2009) applied on feature-rich CRFs.

1 Introduction

There are often multiple types of data that share common characteristics. For example, in this work we are interested in slot tagging of user queries under as many as 17 different domains. Correctly labeling a given query requires the knowledge of the domain that the query is in. A word such as "home" can be labeled as either `contact_name` under the COMMUNICATION domain, `place_type` under the PLACES domain, or `home_screen` under the XBOXENTERTAINMENTSEARCH domain. On the other hand, a function word such as "the" is labeled as `others` across all domains.

Ideally, we would like to train on all sources of data in order to learn about words that are shared across domains, but naively combining these sources for training does not work well since this ignores label inconsistencies across domains. The other extreme is to only use data only from the domain of interest, but it misses out opportunities to learn global patterns shared by multiple domains. The goal of domain adaptation is precisely to address this conundrum: how can we learn from multiple sources of data but retain the integrity of individual domains?

There has been much progress in domain adaptation. A notable example is the feature augmentation method of Daumé III (2009), whose key insight is that if we partition the model parameters to those that handle *common patterns* and those that handle *domain-specific patterns*, the model is forced to learn from all domains yet preserve domain-specific knowledge. The method of Daumé III (2009) is usually considered for sparse binary-valued features which underlie conventional NLP systems. With such features, the parameter partitioning can be achieved with trivial data preprocessing: by conjoining feature types with domain indicators and using them alongside the original feature types. But it is not clear how this approach can be extended to dense real-valued feature values, which are used in many recent NLP systems based on neural networks.

In this paper, we describe natural generalizations of such domain adaptation techniques to neural networks. First, we propose a neural extension of the feature augmentation method of Daumé III (2009)

* Work done while at Columbia University.

† Work done while at Microsoft.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 387–396, Osaka, Japan, December 11-17 2016.

in which we achieve the effect of model partitioning by having a global LSTM used across all domains and independent LSTMs used within individual domains, and then combining their outputs in the top layer. Second, we propose using the framework for learning predictive structures by Ando and Zhang (2005) for domain adaptation which has not previously been considered for this task (the original work only considers multi-tasking in the context of semi-supervised learning): we likewise consider a neural extension of this framework.

We perform slot tagging experiments on 17 different personal digital assistant domains that Cortana handles (Tur, 2006; Anastasakos et al., 2014; Kim et al., 2015a; Kim et al., 2015c; Kim et al., 2015b; Kim et al., 2016a; Kim et al., 2016b). Our methods give clear performance improvement over naive baselines such as training K independent models on individual domains or training one model on the union of all domains. Our methods also significantly outperform the feature augmentation method of Daumé III (2009) with standard sparse binary features implemented with conditional random fields (CRFs).

The rest of the paper is organized as follows. In Section 2, we discuss related works. In Section 3, we provide background information on domain adaptation and sequence modeling. First, we review the feature augmentation method of Daumé III (2009) (Section 3.1). Then we review the framework for learning predictive structures by Ando and Zhang (2005) and observe how it can be naturally considered for domain adaptation (Section 3.2). We also give a brief introduction to neural networks for sequence modeling (Section 3.3). In Section 4, we present a neural extension of the feature augmentation method. In Section 5, we present a neural extension of the framework of Ando and Zhang (2005). In Section 6, we give experimental results.

2 Related Works

Domain adaptation and multi-taking with neural networks have been an active research area. We discuss some examples of previous works and how our work differs.

Many past approaches to domain adaptation simply augment the network with a parameter that activates on the current domain. For instance, Alumäe (2013) and Tilk and Alumäe (2014) tackle multi-domain neural language modeling, with both feedforward and recurrent networks, by introducing a domain-indicator parameter. Similarly, Ammar et al. (2016) address multi-lingual dependency parsing with the stack LSTM by introducing a language-indicator parameter that exploits various resources for the given language such as the language identity and WALS typological properties.

Our work is distinguished from these works in that we use a *global* $(K+1)$-th model that captures general patterns across K domains, rather than just K models augmented with domain indicators. This explicit modeling of domain-independent patterns is more in the spirit of the original feature augmentation method of Daumé III (2009). The problem of adapting a general network to particular task has been studied in the context of speech recognition acoustic models (Yao et al., 2012; Mirsamadi and Hansen, 2015).

A task closely related to domain adaptation is multi-tasking: training a single model to perform different tasks (not just different domains) in hope to exploit common properties across tasks. Multi-tasking has also been investigated extensively in the NLP neural network community, notably the "NLP from scratch" work by Collobert et al. (2011) in which various sequence labeling tasks are tackled by a single network with convolution and CRF layers, and its numerous followup studies such as Yang et al. (2016).

3 Background

3.1 Domain Adaptation by Feature Augmentation

Daumé III (2009) propose the following simple approach to domain adaptation. Suppose there are K distinct domains, and let d denote the total number of feature types. The usual setting in NLP is that these features are high-dimensional, sparse, and binary-valued. For example in a CRF, a feature type may ask if the current word is "apple" and the previous word is "company": it takes value 1 if the answer is yes and 0 otherwise. Thus without distinguishing domains, a feature vector takes the form $x \in \{0,1\}^d$ for any domain $k \in \{1 \dots K\}$.

Now, consider learning a naive model $f : \{0,1\}^d \rightarrow \Omega$ over all K domains where Ω denotes an output space. Without domain adaptation, the model f receives a feature vector $x \in \{0,1\}^d$ and simply outputs the value $f(x) \in \Omega$ no matter which domain x corresponds to. The proposal of Daumé III (2009) is that we train a model $f^{aug} : \{0,1\}^{(K+1)d} \rightarrow \Omega$ that instead uses an *augmented* feature vector $x^{aug} \in \{0,1\}^{(K+1)d}$ defined as

$$x^{aug} = (x, 0, \ldots, x^k, \ldots, 0)$$

where $x^k \in \{0,1\}^d$ is a feature representation of x under the k-th domain. In other words, we expand the feature space by $K+1$ times by duplicating the original feature x for K times: then given an input from the k-th domain, we only use the original x and the corresponding domain representation x^k.

Note that this essentially partitions the model parameters to those that handle the representation x used across all domains and those that handle x^k for individual domains $k \in \{1 \ldots K\}$. In the sparse binary feature regime, this can be achieved by conjoining the original feature types with a domain indicator. For example, the i-th feature value of x^k can look like

$$x_i^k = \begin{cases} 1 & \text{if the current word is "apple"} \\ & \text{AND the previous word is "company"} \\ & \text{AND the domain is } k \\ 0 & \text{otherwise} \end{cases}$$

This can be also thought of as data preprocessing in which we duplicate the data in each domain and mark one copy with the domain identity.

3.2 Shared Structure Learning Framework

Ando and Zhang (2005) propose learning a shared structure across multiple related classification tasks. Specifically, they consider K binary classification tasks each of which has its own linear classifier $f_k : \mathbb{R}^d \rightarrow \mathbb{R}$ mapping a d-dimensional feature vector $x \in \mathbb{R}^d$ to a classification score

$$f_k(x) := (u_k + \Theta v_k)^\top x = u_k^\top x + v_k^\top \Theta^\top x$$

Here, $u_k \in \mathbb{R}^d$ and $v_k \in \mathbb{R}^m$ are task-specific parameters but $\Theta \in \mathbb{R}^{d \times m}$ is a global parameter shared by all classifiers $f_1 \ldots f_K$. In particular, if Θ is zero then each classifier is an independent linear function $u_k^\top x$. The predicted label is the sign of the classification score $f_k(x)$.

The parameter sharing makes the estimation problem challenging, but Ando and Zhang (2005) develop an effective alternating loss minimization algorithm using a variational property of singular value decomposition (SVD). The original work applied this framework to semi-supervised learning and achieved strong empirical results. But note that it can be viewed as domain adaptation where the K classification tasks are different "domains" and we aim to learn a global parameter Θ shared across the domains.

3.3 Neural Networks for Sequence Modeling

Recently, there has been much success in tackling sequence modeling problems using neural networks. By far the most popular network for sequence modeling is long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), which is a special case of more general recurrent neural network (RNN) architecture. An RNN extends a traditional feedforward network by recursively receiving inputs generated by the model itself. This essentially defines a feedforward network in which the same set of parameters are used multiple times and is well-suited for sequential data. But a simple RNN suffers from the vanishing gradient problem and does not model long-term dependency very well (Pascanu et al., 2013). An LSTM addresses this issue by introducing a memory cell in RNN architecture that can control how much past information to retain or to forget. This modification has been shown to be crucial in practice. Many recent works in NLP have achieved state-of-the-art results using variants of LSTM, for example in dependency parsing (Dyer et al., 2015) and machine translation (Bahdanau et al., 2014).

4 A Neural Extension of the Feature Augmentation Method

We now describe a neural extension of the feature augmentation method for domain adaptation (Daumé III, 2009). Our model consists of $K + 1$ LSTMs: one LSTM θ is used on all domains, and the remaining K LSTMs $\theta^1 \dots \theta^K$ are used only for the corresponding domains. More specifically, we predict one of L labels at the t-th time step in a given user query in domain $k \in \{1 \dots K\}$ as follows. The common LSTM θ produces an output vector $h_t \in \mathbb{R}^d$ and the domain-specific LSTM θ^k produces an output vector $h_t^k \in \mathbb{R}^d$. The output dimension d of these LSTMs is empirically chosen. The model has parameters $W \in \mathbb{R}^{L \times d}$ and $W^k \in \mathbb{R}^{L \times d}$ at the top layer and combines the LSTM outputs as $z_t^k \in \mathbb{R}^L$ where

$$z_t^k = [W\ W^k] \begin{bmatrix} h_t \\ h_t^k \end{bmatrix} = W h_t + W^k h_t^k \tag{1}$$

which is a direct analogue of the feature augmentation method with sparse binary-valued features:

$$x^{aug} = (x, 0, \dots, x^k, \dots, 0)$$

This combined representation (1) is put through a softmax function to produce a distribution over L labels. Figure 1 (a) provides an illustration of the proposed architecture The model can be trained by maximizing the log likelihood of correct predictions in the training data:

$$J(W, W^1 \dots W^k, \theta, \theta^1 \dots \theta^k) = \sum_t \log \left(\frac{\exp [z_t]_{ans(t)}}{\sum_{l=1}^{L} \exp [z_t]_l} \right) \tag{2}$$

where $ans(t) \in \{1 \dots L\}$ is the ground-truth label for the t-th training example. Note that we do not consider the sentence-level log likelihood (which requires an additional CRF layer for global normalization) for simplicity.

5 A Neural Extension of the Shared Structure Learning Framework

We describe a neural extension of the shared structure learning framework of Ando and Zhang (2005). Recall that Ando and Zhang (2005) consider K binary classifiers $f_k : \mathbb{R}^d \to \mathbb{R}$ with a global parameter $\Theta \in \mathbb{R}^{d \times m}$:

$$f_k(x) = u_k^\top x + v_k^\top \Theta^\top x \tag{3}$$

The model can be seen as combining the given input x (the first term) and a transformed input $\Theta^\top x$ (the second term) where the transformation is learned across K tasks.

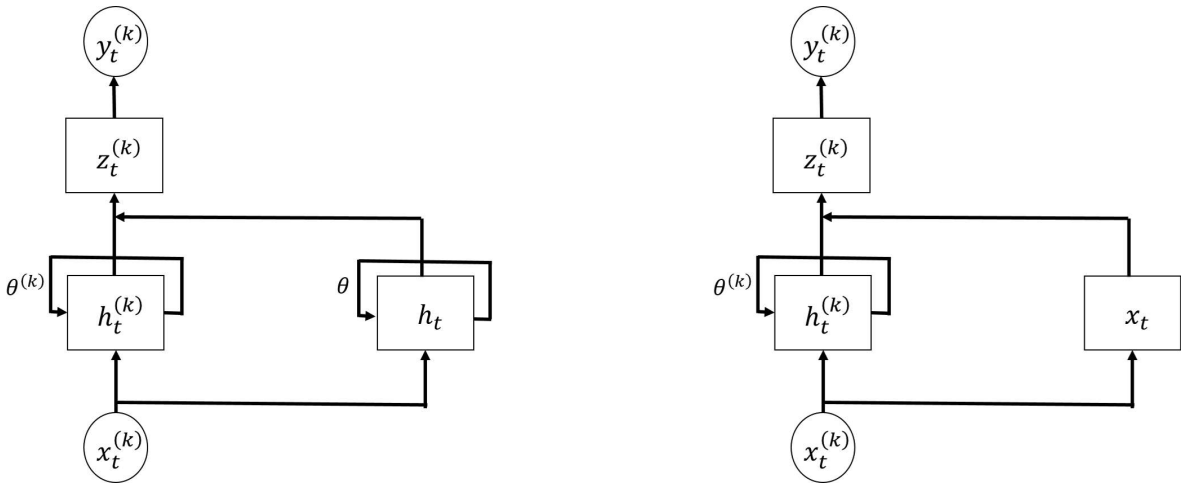
We extend this framework with the following model. The model consists of K LSTMs $\theta^1 \dots \theta^K$ corresponding to K domains. At the t-th time step in a given user query in domain $k \in \{1 \dots K\}$, we compute a direct analogue of (3) by adding the output vector of the k-th LSTM θ^k on the input vector (word embedding) x_t to the original input vector to create

$$z_t^k = W x_t + W^k \theta^k(x_t)$$

where $\theta^k(x_t) \in \mathbb{R}^d$ denotes the output vector of the k-th LSTM on input x_t.

Figure 1 (b) provides an illustration of the proposed architecture This combined representation is put through a softmax function to produce a distribution over L labels. The model can be trained by maximizing the log likelihood of correct predictions in the training data similarly in (2).

Note that the proposed extensions of Daumé III (2009) and Ando and Zhang (2005) only differ in how the domain-independent information is derived. In particular, the extension of Ando and Zhang (2005) can be seen as a weaker version of the extension of Daumé III (2009) since the domain-independent information is used as is (i.e., x_t) in the former while first transformed by a domain specific LSTM in the latter (i.e., h_t).



(a) Feature augmentation method

(b) Shared structure learning framework

Figure 1: Illustration of the proposed neural extensions of Daumé III (2009) and Ando and Zhang (2005).

6 Experiments

In this section, we turn to experimental findings to provide empirical support for our proposed methods. First, we note that our experimental settings are rather different from previously considered settings for domain adaptation in many aspects:

- *Sufficient target data:* In a typical setting for domain adaptation, one generally assumes that the source domain has a sufficient amount of labeled data but the target domain has an insufficient amount of labeled data. However, we have sufficient amounts of labeled data for all domains: our goal is to effectively utilize all of them.
- *Variant output:* In a typical setting for domain adaptation, the label space is invariant across all domains. Here, the label space can be different in different domains, which is a more challenging setting. See Kim et al. (2015d) for details of this setting.
- *Multiple source domains:* In most previous works, only a pair of domains (source vs. target) have been considered, although they can be easily generalized to $K > 2$. Here, we experiment with $K = 17$ domains.

To test the effectiveness of our approach, we apply it to the slot sequence tagging task in a suite of 17 personal assistant domains. The goal is to find the correct semantic tagging of the words in a given user utterance. For example, a user could say *"reserve a table at joeys grill for thursday at seven pm for five people"*; then the model needs to tag *"joeys grill"* with `restaurants`, *"thursday"* with `date`, *"seven pm"* with `time`, and "five" with `numberpeople`. The data statistics and the short descriptions of the domains are shown in Table 1. As the table indicates, the domains have very different attributes. We have 131 unique labels across these domains (a different subset of these labels is used in each domain). The total numbers of training, development and test queries across domains are 2640K, 129K and 64K, respectively.

6.1 Setting

In all our experiments, we train our models using stochastic gradient descent (SGD) with Adam (Kingma and Ba, 2015)—an adaptive learning rate algorithm. We use the initial learning rate of 2×10^{-4} and leave all the other hyperparameters as suggested in Kingma and Ba (2015). Each SGD update is computed using a minibatch of size 128. We use the dropout regularization (Srivastava et al., 2014) with the keep probability of 0.5. The dimension of the hidden layer is $d = 100$. For simplicity, we use simple forward-directional LSTMs rather than bi-direction LSTMs, as we observed that they yielded similar results on

	# of labels	#train	#test	#dev	#vocab	Description
Alarm	8	157K	13K	7K	3504	Set alarms
Calendar	20	130K	12K	7K	11077	Set appointments and meeting in calendar
Comm.	21	750K	60K	26K	69414	Make a call and sent messages
Entertain.	15	150K	8K	3.7K	17239	Find songs and movies
Events	6	7K	1.5K	1K	691	Find events and book a ticket
Hotel	17	4.5K	2.6K	1.8K	8022	Book hotel
Mediactrl	10	126K	11K	6K	12589	Set up a music player
Mvtickets	7	4K	1.2K	1K	2235	Find movie theater and book a ticket
Mystuff	18	3.9K	1.8K	1K	8711	Find and open a document
Note	3	6.7K	1.2K	0.6K	4269	Edit and create note
Ondevice	6	228K	2.5K	1.5K	5332	Set up a phone
Orderfood	11	7K	1.4k	0.6K	3686	Order food using app
Places	32	479K	4.5K	2.5K	51424	Find location and direction
Reminder	16	307K	2.7K	1.2K	27510	Remind appointment and to-do list
Reservations	12	2.4K	1K	0.6K	1269	Make a restaurant reservations
Taxi	10	6.3K	2K	1.2K	4391	Find and book a cab
Weather	9	281K	2.5K	1.5K	11878	Ask weather
Overall	131	2640K	129K	64K	139310	

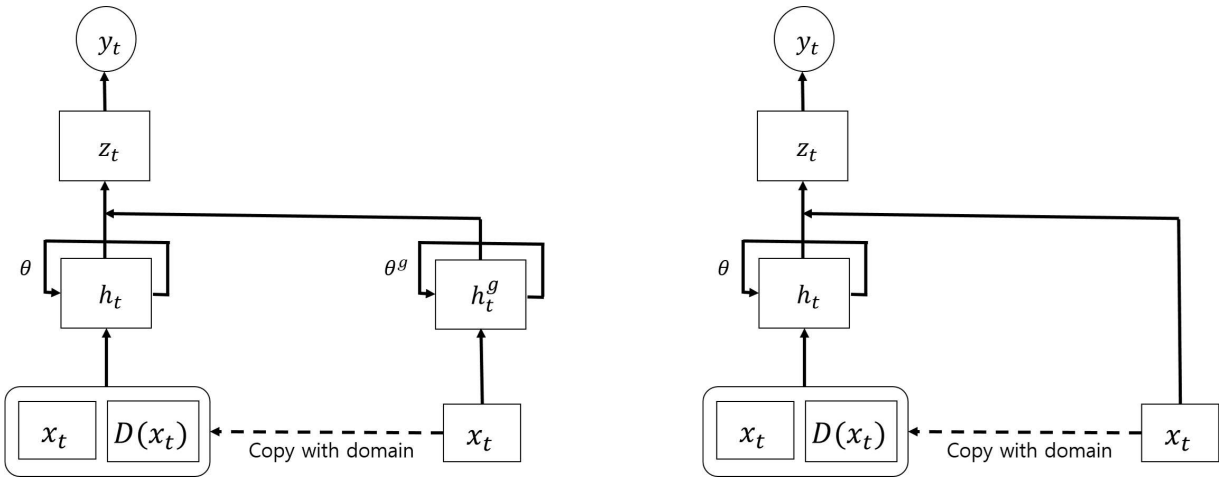
Table 1: Data sets used in the experiments. For each domain, the number of unique labels, the number of quires in the training, development, and test sets, input vocabulary size of the training set, and short description about domain.

the development set in preliminary experiments. To initialize word embedding, we used word embedding trained on 6 billion tokens (6B-200d) from Wikipedia 2014 plus Gigaword 5 (Pennington et al., 2014). These configurations were selected by observing the models performance on held-out development set.

We compare the following methods for the slot tagging tasks:

- *NoAdapt*: train a feature-rich CRF only on target training data.
- *Union*: train a feature-rich CRF on the union of source and target training data.
- *Daume*: train a feature-rich CRF with the discrete feature duplication method of Daumé III (2009).
- *1D&E*: train a domain specific LSTM with a generic embedding on all domain training data, shown on the right in Figure 2.
- *1D&L*: train a domain specific LSTM with a generic LSTM on all domain training data, shown on the left in Figure 2.
- *KD&E*: train domain specific K LSTMs with a generic embedding on all domain training data, shown on the right in Figure 1.
- *KD&L*: train domain specific K LSTMs with a generic LSTM on all domain training data, shown on the left in Figure 1.

The methods *1D&E* and *1D&L* are slight variants of the main proposed neural networks *KD&E* and *KD&L* in Figure 1. These only use a single LSTM for K domains in addition to a common LSTM: it is mostly presented for comparison purposes. Note that we have a single model to tag all domains (except for *NoAdapt* with which we have K models).



(a) 1 domain specific LSTM + generic LSTM

(b) 1 domain specific LSTM + generic embedding

Figure 2: Illustration of slight variants of the proposed neural extensions of Daumé III (2009) and Ando and Zhang (2005) in which we have 2 instead of $K + 1$ LSTMs: one is used for individual domains, one is shared across all domains.

Domain	Noadapt	Union	Daumé III (2009)
Alarm	95.89	89.24	**97.36**
Calendar	91.03	82.09	**94.4**
Comm.	94.94	81.79	**96.6**
Entertain.	**93.83**	89.91	92.61
Events	87.84	58.74	**89.34**
Hotel	91.75	84.54	**92.76**
Mediactrl	90.39	77.76	**92.3**
Mvtickets	92.75	79.21	**94.43**
Mystuff	**90.92**	79.87	89.95
Note	87.9	63.21	**90.86**
Ondevice	93.59	89.48	**96.44**
Orderfood	93.52	85.71	**95.78**
Places	92.75	87.46	**94.13**
Reminder	91.81	84.25	**94.02**
Reservations	92.68	74.45	**94.43**
Taxi	90.27	79.22	**94.75**
Weather	97.27	91.27	**98.44**
Average	92.30	81.07	**94.04**

Table 2: F1 scores across seventeen personal assistant domains for *Noadapt*, *Union* and Daumé III (2009) in a setting with sparse binary-valued features.

6.2 Results

For non-neural models, we use conditional random fields (CRFs) (Lafferty et al., 2001) with a rich set of binary-valued features such as lexical features, gazetteers, Brown clusters (Brown et al., 1992) and context words. For parameter estimation, we used L-BFGS (Liu and Nocedal, 1989) with 100 as the maximum iteration count and 1.0 for the L2 regularization parameter.

Their results are shown in Table 2. A few observations: the model without domain adaptation (*Noadapt*) is already very competitive because we have sufficient training data. However, simply training a single model with aggregated queries across all domains significantly degrades performance (*Union*). This is because in many cases the same query is labeled differently depending on the domain and the

context. This is also because the amounts of training data are widely different across different domains. For example, slots (e.g. `app name`) in other domains are overwhelmed by slots in PLACES domain such as `place name` since our PLACES domain is the largest in terms of number of slot types and 2nd largest in terms of dataset size. Finally, the feature augmentation method of Daumé III (2009) dramatically improves performance across all domains, achieving the average F1 score of 94.04.

Daumé III (2009)	Domain	NoAdapt	1D&E	1D&L	KD&E	KD&L
97.36	Alarm	94.64	97.23	97.26	97.46	**97.5**
94.4	Calendar	93.49	95.03	95.06	95.14	**96.67**
96.6	Comm.	95.36	96.82	96.83	96.96	**97.08**
92.61	Entertain.	94.73	94.34	94.28	**94.61**	94.59
89.34	Events	87.12	90.33	**92.82**	90.92	92.65
92.76	Hotel	92.28	94.78	96.11	96.4	**97.71**
92.3	Mediactrl	91.91	90.85	92.23	92.22	**93.49**
94.43	Mvtickets	92.75	94.98	96.76	96.41	**97.55**
89.95	Mystuff	89.34	88.42	88.23	88.68	**90.71**
90.86	Note	89.89	91.77	94.29	91.38	**94.61**
96.44	Ondevice	96.65	97.29	97.31	97.52	**97.64**
95.78	Orderfood	93.28	95.23	**96.65**	96.45	96.26
94.13	Places	94.47	95.85	96.52	96.52	**96.64**
94.02	Reminder	91.53	92.96	93.14	93.27	**93.37**
94.43	Reservations	92.82	94.43	95.45	95.76	**96.09**
94.75	Taxi	88.32	91.7	91.94	91.51	**92.07**
98.44	Weather	98.07	98.69	98.46	98.45	**98.8**
94.04	Average	92.74	94.16	94.90	94.69	**95.50**

Table 3: F1 scores for Daumé III (2009) and LSTM model variants across seventeen personal assistant domains.

The results with our proposed LSTM domain adaptation models are also shown in Table 3. For easy comparison, the results with Daumé III (2009) applied on feature-rich CRFs are duplicated at the leftmost column.

The *NoAdapt* yields 92.74% average F1-measure which is higher than the performance of CRFs without the rich hand-crafted features. We consistently and significantly improve performance by using adaptation techniques. First, the *1D&E* improves F1 performance to 94.16%. Using not generic embedding, but generic LSTM *(1D&L)* boosts the performance up to 94.9%. Also, having K domain specific LSTMs with generic embeddings (*KD&E*) which is very close to Ando and Zhang (2005) achieves performance gain from *1D&E*, but fail to provide any improvement from 1D&L. Finally, KD&L, which is directly inspired by Daumé III (2009), achieves the best performance of 95.5%, indicating that having K different domain specific LSTMs and a generic LSTM yields significant gains on most of the domains.

7 Conclusion

In this work, we described novel neural approaches to domain adaptation. Adding to the rich history of works in domain adaptation and multi-tasking with neural networks, we proposed a neural analogue of the well-established feature augmentation method of Daumé III (2009) and the shared structure learning framework of Ando and Zhang (2005). Our extensions are natural and simple. In experiments on slot tagging over 17 domains, we demonstrated the effectiveness of our methods by delivering clear performance gains over both naive and strong baselines.

Acknowledgements

We thank Minjoon Seo and anonymous reviewers for their constructive feedback.

References

Tanel Alumäe. 2013. Multi-domain neural network language model. In *INTERSPEECH*, pages 2182–2186. Citeseer.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Many languages, one parser. *CoRR, abs/1602.01595*.

Tasos Anastasakos, Young-Bum Kim, and Anoop Deoras. 2014. Task specific continuous word representations for mono and multi-lingual spoken language understanding. In *ICASSP*, pages 3246–3250. IEEE.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Hal Daumé III. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

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

Young-Bum Kim, Minwoo Jeong, Karl Stratos, and Ruhi Sarikaya. 2015a. Weakly supervised slot tagging with partially labeled sequences from web search click logs. In *Proceedings of the NAACL*. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, Xiaohu Liu, and Ruhi Sarikaya. 2015b. Compact lexicon selection with spectral methods. In *Proc. of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2015c. Pre-training of hidden-unit crfs. In *Proc. of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 192–198.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015d. New transfer learning techniques for disparate label sets. *ACL. Association for Computational Linguistics*.

Young-Bum Kim, Alexandre Rochette, and Ruhi Sarikaya. 2016a. Natural language model re-usability for scaling to different domains. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016b. Scalable semi-supervised query classification using matrix sketching. In *Proc. of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *The International Conference on Learning Representations (ICLR)*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

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

Seyedmahdad Mirsamadi and John HL Hansen. 2015. A study on deep neural network acoustic model adaptation for robust far-field speech recognition. In *Proceedings of Interspeech*, pages 2430–2434.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Ottokar Tilk and Tanel Alumäe. 2014. Multi-domain recurrent neural network language model for medical speech recognition. In *Baltic HLT*, pages 149–152.

Gokhan Tur. 2006. Multitask learning for spoken language understanding. In *In Proceedings of the ICASSP*, Toulouse, France.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

Kaisheng Yao, Dong Yu, Frank Seide, Hang Su, Li Deng, and Yifan Gong. 2012. Adaptation of context-dependent deep neural networks for automatic speech recognition. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 366–369. IEEE.

A House United: Bridging the Script and Lexical Barrier between Hindi and Urdu

Riyaz Ahmad Bhat **Irshad Ahmad Bhat** **Naman Jain** **Dipti Misra Sharma**
LTRC, IIIT-Hyderabad, India
{riyaz.bhat,irshad.bhat,naman.jain}@research.iiit.ac.in, dipti@iiit.ac.in

Abstract

In Computational Linguistics, Hindi and Urdu are not viewed as a monolithic entity and have received separate attention with respect to their text processing. From part-of-speech tagging to machine translation, models are separately trained for both Hindi and Urdu despite the fact that they represent the same language. The reasons mainly are their divergent literary vocabularies and separate orthographies, and probably also their political status and the social perception that they are two separate languages. In this paper, we propose a simple but efficient approach to bridge the lexical and orthographic differences between Hindi and Urdu texts. With respect to text processing, addressing the differences between their texts would be beneficial in the following ways: (a) instead of training separate models, their individual resources can be augmented to train single, unified models for better generalization, and (b) their individual text processing applications can be used interchangeably under varied resource conditions.

To remove the script barrier, we learn accurate statistical transliteration models which use sentence-level decoding to resolve word ambiguity. Similarly, we learn cross-register word embeddings from the harmonized Hindi and Urdu corpora to nullify their lexical divergences. As a proof of the concept, we evaluate our approach on the Hindi and Urdu dependency parsing under two scenarios: (a) resource sharing, and (b) resource augmentation. We demonstrate that a neural network-based dependency parser trained on augmented, harmonized Hindi and Urdu resources performs significantly better than the parsing models trained separately on the individual resources. We also show that we can achieve near state-of-the-art results when the parsers are used interchangeably.

1 Introduction

Hindi and Urdu are spoken primarily in northern India and Pakistan and together constitute the third largest language spoken in the world.[1] They are two standardized registers of what has been called the Hindustani language, which belong to the Indo-Aryan language family. Masica (1993) explains that, while they are different languages officially, they are not even different dialects or sub-dialects in a linguistic sense; rather, they are different literary styles based on the same linguistically defined sub-dialect. He further explains that at the colloquial level, Hindi and Urdu are nearly identical, both in terms of core vocabulary and grammar. However, at formal and literary levels, vocabulary differences begin to loom much larger (Hindi drawing its higher lexicon from Sanskrit and Urdu from Persian and Arabic) to the point where the two styles/languages become mutually unintelligible. In written form, not only the vocabulary but the way Urdu and Hindi are written makes one believe that they are two separate languages. They are written in separate orthographies, Hindi being written in Devanagari, and Urdu in a modified Perso-Arabic script. Given these differences in script and vocabulary, Hindi and Urdu are

[1] see http://www.ethnologue.com/statistics/size and https://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 397–408, Osaka, Japan, December 11-17 2016.

socially and even officially considered two separate languages. These apparent divergences have also led to parallel efforts for resource creation and application building in computational linguistics. The Hindi-Urdu treebanking project is one such example where the influence of differences between Hindi and Urdu texts have led to the creation of separate treebanks for Hindi and Urdu (Bhatt et al., 2009; Bhat et al., 2015). However, pursuing them separately in computational linguistics makes sense. If the two texts differ in form and vocabulary they can not be processed with same models unless the differences are accounted for and addressed. In this paper, we aim to remove these differences between Hindi and Urdu texts. We learn accurate machine transliteration models for the common orthographic representation of their texts. To resolve their lexical divergences, we learn cross-register word embeddings from the harmonized Hindi and Urdu corpora. So as to evaluate our approach, we empirically demonstrate the impact of text harmonization on the dependency parsing of both Hindi and Urdu under varied supervised training conditions. We show that a neural network-based parser trained on cross-register word embeddings sets the new benchmark for dependency parsing of Hindi and Urdu. A summary of our overall contributions is provided as follows:

- **Common Representation:** To remove the orthographic differences between Hindi and Urdu texts, we evaluate Devanagari and Perso-Arabic scripts for their common representation. We propose accurate statistical transliteration models which use sentence-level decoding on n-best transliterations to resolve word ambiguity. We empirically show that the Devanagari script is better suited for automatic text processing of both Hindi and Urdu. We show significant gains in accuracy in different modules of Hindi and Urdu dependency parsing pipeline when the training and evaluation data are represented in Devanagari instead of Perso-Arabic.
- **Resource Sharing and Augmentation:** To facilitate resource sharing and augmentation, we use statistical transliteration to harmonize the orthographic differences between Hindi and Urdu texts, while their lexical divergences are resolved by learning cross-register word embeddings. We augment their harmonized treebanks and train neural network-based parsing models using different supervised domain adaptation techniques. We empirically show that our augmented models perform significantly better than the models trained separately on individual treebanks. Moreover, we also demonstrate that the individual parsing models trained on harmonized Hindi and Urdu resources can be used interchangeably to parse both Hindi and Urdu texts and give near state-of-the-art results.

2 Experimental Setup

To experiment on resource sharing and augmentation between Hindi and Urdu, we need to mitigate their orthographic and lexical differences. To that end, we propose a simple approach which uses machine transliteration and distributional similarity-based methods (see §3 and §4). After script harmonization, we learn distributed word representations to project Hindi and Urdu lexicon in the same distributional space. To make effective use of these word representations, we employ the non-linear neural network architecture for transition-based dependency parsing proposed by Chen and Manning (2014). We use a similar architecture for sequence labeling as well.

2.1 Parsing Models

Our parsing model is based on transition-based dependency parsing paradigm (Nivre, 2008). Particularly, we use an arc-eager transition system, which is one of the famous transition-based parsing systems. Arc-eager algorithm defines four types of transitions to derive a parse tree namely: 1) Shift, 2) Left-Arc, 3) Right-Arc, and 4) Reduce. To predict these transitions, a classifier is employed. We follow Chen and Manning (2014) and use a non-linear neural network to predict these transitions for any parser configuration. The neural network model is the standard feed-forward neural network with a single layer of hidden units. The output layer uses softmax function for probabilistic multi-class classification. The model is trained by minimizing cross entropy loss with an $l2$-regularization over the entire training data. We also use mini-batch Adagrad for optimization and apply dropout (Chen and Manning, 2014).

From each parser configuration, we extract features related to the top four nodes in the stack, top four nodes in the buffer and leftmost and rightmost children of the top two nodes in the stack and the leftmost

child of the top node in the buffer. For each node, we use the distributed representation of its lexical form, POS tag, chunk tag and/or dependency label. We use the Hindi and Urdu monolingual corpora to learn the distributed representation of the lexical units. The Hindi monolingual data contains around 40M raw sentences, while the Urdu data is comparatively smaller and contains around 6M raw sentences. The distributed representation of non-lexical units such as POS, chunk and dependency labels are randomly initialized within a range of -0.01 to +0.01 (Chen and Manning, 2014).

To decode a transition sequence, we use dynamic oracle recently proposed by Goldberg and Nivre (2012) instead of the vanilla static oracle. Dynamic oracle allows training by exploration that helps to mitigate the effect of error propagation. We use the same value for the exploration hyperparameter as suggested by Goldberg and Nivre (2012).

2.2 Sequence Labeling Models

For the training of POS tagging and chunking models, we use a similar neural network architecture as discussed above. Unlike Collobert et al. (2011), we do not learn separate transition parameters. Instead we include the structural features in the input layer of our model with other lexical and non-lexical units. For POS tagging, we use second-order structural features, 2 words to either side of the current word, and last 3 letters of the current word. Similarly, for chunking we use POS tags of the current word and the 2 words surrounding it on the either side, in addition to the features used for POS tagging.

2.3 Hindi and Urdu Treebanks

We use the annotations in the Hindi and Urdu treebanks (henceforth HDTB and UDTB) for conducting all the experiments. Both treebanks are multi-layered and multi-representational (Bhatt et al., 2009). They contain three layers of annotation namely **dependency structure (DS)** for annotation of modified-modifier relations, **PropBank-style annotation** for predicate-argument structure, and an independently motivated **phrase-structure annotation**. For our experiments, we only need annotations in the first layer of the treebanks i.e., annotations in the DS layer. Dependency Structure involves dependency analysis based on the Pāṇinian Grammatical framework (Bharati et al., 1995). In addition to dependency analysis, the sentence annotation also includes morphological analysis, part-of-speech (POS) tagging and chunking. POS tagging and chunking are based on Indian Language Machine Translation (ILMT) guidelines (Bharati et al., 2006). There are around 32 POS tags and 11 chunk tags used in the treebanks, while the dependency labels are around 82.

We split the treebank data with a ratio of 80:10:10 for training, testing and tuning separate models for POS tagging, chunking and parsing. For both treebanks, the internal structure of annotation files is preserved. However, we randomly distribute the files across training, testing and development sets. Each document or annotation file mainly contains newswire articles. Statistics about the data are provided in Table 1.

Count of	Hindi			Urdu		
	Training	Testing	Development	Training	Testing	Development
Tokens	3,47,744	43,556	43,556	1,53,317	19,065	19,065
Chunks	1,87,029	23,418	23,417	72,319	9,010	9,010
Sentences	16,629	2,077	2,077	5,432	677	677

Table 1: Statistics of training, testing and development sets used in all the experiments reported in this paper.

3 Common Representation

Despite their similarities, we can not use Hindi text processing tools for Urdu as such and vice versa as they are written in two different scripts. To address this problem, we need to represent both Hindi and Urdu texts in a single script. For this purpose, we can either use Devanagari or Perso-Arabic script. We can transliterate Hindi texts in Devanagari to Perso-Arabic or Urdu texts in Perso-Arabic to Devanagari. Either way, transliteration between these two scripts is a non-trivial task. There are genuine cases of character ambiguity due to one-to-many character mappings in both directions of transliteration. A detailed description of the challenges in Hindi-Urdu transliteration can be found in the works of Malik et al. (2008) and Lehal and Saini (2014). In addition to character ambiguity, Perso-Arabic to Devanagari

transliteration has to deal with missing short vowels in Urdu texts also. In Urdu writing, short vowels are hardly represented, even though the Perso-Arabic script has the provision for their representation. A major drawback of dropping short vowels in Urdu writing is that it generates homographs. For example, without an appropriate short vowel on the first letter, ہوا could mean 'air' (ہَوا) or 'become' (ہُوا) depending on the context. These homographs would lead to ambiguity in the Devanagari script. There would be more than one genuine Devanagari representation for such homographs, since Devanagari represents each phoneme uniquely and explicitly. Usually word-level transliteration models do not deal with word ambiguity and leave it unresolved. However, we need our transliteration model to pick a transliteration that best fits the sentential context.

It should be noted that both Devanagari and Perso-Arabic scripts are a natural choice for the common representation. The use of a third script (e.g. Roman script) for this purpose would be computationally expensive, as we need to transliterate both Hindi and Urdu resources. Moreover, the transliteration errors would also double. More importantly, if we choose a third script, we have to manually develop a reasonably-sized corpus of transliteration pairs for training the transliteration models. On the other hand, transliteration pairs in Devanagari and Perso-Arabic scripts can be automatically extracted from the corpora available in these scripts (see §3.1.1 for more details).

To measure the suitability of both scripts for the common representation of Hindi and Urdu texts, we perform extrinsic evaluation on the dependency parsing pipeline which involves POS tagging, chunking and dependency parsing. The script that maximizes the accuracy across the pipeline would imply its feasibility for uniformly representing the Hindi and Urdu texts for computational purposes.

3.1 Hindi-Urdu Transliteration

Hindi and Urdu transliteration has received a lot of attention from the NLP research community of South Asia (Malik et al., 2008; Lehal and Saini, 2012; Lehal and Saini, 2014). It has been seen to break the barrier that makes the two look different. Most of the existing works on Hindi-Urdu transliteration have considered basic rule-based models which use character tables coupled with a set of heuristics to resolve ambiguous mappings. Statistical approaches have hardly been explored (Sajjad et al., 2011). Unlike rule-based systems, statistical approaches are more robust and efficient. Among data-driven approaches, machine learning methods like noisy-channel model and structured prediction algorithms have been widely used for machine transliteration (Knight and Graehl, 1998; Zelenko and Aone, 2006). In this work, we model Hindi-Urdu transliteration as a structured prediction problem using a linear model. We use the structured perceptron (DHMM) of Collins (Collins, 2002) to learn the parameters of our transliteration model. Given an input training data of aligned character sequences $D = d_1 ... d_n$, a vector feature function $\vec{f}(d)$, and an initial weight vector $\vec{w}$, the algorithm performs two steps for each training example $d_i \in D$: (1) **Decode**: $\hat{t} = \underset{t_1 \cdots t_n}{\arg\max}(\vec{w} \cdot \vec{f}(d))$, and (2) **Update**: $\vec{w} = \vec{w} + \vec{f}(d) - \vec{f}(\hat{t})$. We use Viterbi-search for decoding in case of Devanagari to Perso-Arabic transliteration, while we use beam-search for Perso-Arabic to Devanagari transliteration to decode the best letter sequence in the target script. The latter is used to extract n-best transliterations for resolving word ambiguity.

In case of Perso-Arabic to Devanagari transliteration, to resolve the word ambiguity as discussed above, we perform sentence-level decoding on the n-best transliterations from the perceptron model. We use a noisy channel model and exact Viterbi search to find the most likely Hindi (Devanagari) sentences. The noisy-channel model can be formally defined as follows:

$$\mathbf{h}^* = \arg\max \; p(h) \times p(h|u) \tag{1}$$

$p(h)$ is the language model score which gives a prior distribution over the most likely sentences in Hindi and $p(h|u)$ is the perceptron score which indicates how likely the Hindi (Devanagari) sentence h is a word by word transliteration of the Urdu sentence u. Since $p(h|u)$ is not a probability score, we assign uniform probabilities to all the transliteration options. Thus redefining our model without $p(h|u)$ as:

$$\mathbf{h}^* = \arg\max \; p(h) \tag{2}$$

Thus, our model only relies on language model to find the best sentence from the n-best transliterations. We use trigram language model learned from 40M multi-domain Hindi corpus with Kneser-Ney smoothing. Here, it should be noted that it is plausible to score Urdu sentences in Devanagari using a language model trained on Hindi data, since there is a considerable overlap in the Hindi and Urdu grammar and vocabulary.

3.1.1 Transliteration Pair Extraction and Character Alignment

Like any other supervised machine learning approach, supervised machine transliteration requires a strong list of transliteration pairs to learn the model parameters. We use the sentence aligned ILCI Hindi-Urdu parallel corpora (Jha, 2010) to extract the transliteration pairs. Initially, the parallel corpus is word-aligned using GIZA++ (Och and Ney, 2003). We extract all the word pairs which occur as 1-to-1 alignments in the word-aligned corpus as potential transliteration equivalents. We extracted a total of 54,035 translation pairs from the parallel corpus of 50,000 sentences. To further complement the translation pairs, we also extracted 66,668 pairs from IndoWordNet (Narayan et al., 2002) synset mappings.[2] A rule-based approach with edit distance metric is used to extract the transliteration pairs from these translation pairs. To compute the edit distances, we use the Hindi-Urdu character mappings presented in (Lehal and Saini, 2014). We compute the levenshtein distance between the translation pairs based on insertion, deletion and replace operations. The distance scores are normalized by dividing them with the length of the longest string in a translation pair. Translation pairs with a normalized score of less than a small threshold of $\sim$0.1 are considered as transliteration pairs. Using this procedure, we extracted 21,972 unique transliteration pairs from the Hindi-Urdu parallel corpus and 24,614 unique transliteration pairs from the Hindi-Urdu synsets. After mining the transliteration pairs, we character align them using Giza++ for training and testing the transliteration models.

3.1.2 Experiments and Results

We train two structured perceptron models on the transliteration pairs discussed above. We maintain 80:10:10 data split for training, testing and tuning both models. Additionally, we also manually transliterate 1000 Urdu sentences in Devanagari script to tune and evaluate our noisy-channel model which we use on top of the Perso-Arabic to Devanagari transliteration system to resolve word ambiguity. The parameters such as number of training iterations, order of ngram context, number of transliterations for noisy-channel model are tuned on the respective development sets. We found that the top 5 transliterations gave the best results.

To compare our results with the existing systems, we choose HUMT, Malerkotla (MAL) and SANGAM available on the Internet,[3] while choosing the SMT-based transliteration as a baseline. The baseline model is a phrase-based machine translation system (PSMT) for transliteration built using the Moses toolkit.[4] We train the system with the default settings with the distortion limit set to 0. We list the performance of all these systems in Table 2 for comparison. The performance of our noisy-channel models is reported in Table 3.

System	Devanagari → Perso-Arabic	Perso-Arabic → Devanagari
PSMT	96.23%	74.30%
HUMT	90.34%	40.75%[5]
MAL	93.78%	78.23%
SANGAM	97.38%	87.56%
DHMM	**98.03%**	**88.03%**

Table 2: Comparison of type-level accuracies of the available systems on the Internet with our system.

[2] http://www.cfilt.iitb.ac.in/~sudha/bilingual_mapping.tar.gz

[3] http://www.sanlp.org/HUMT/HUMT.aspx, translate.malerkotla.co.in/ and http://sangam.learnpunjabi.org/

[4] http://www.statmt.org/moses/

[5] HUMT system performed worst on Perso-Arabic to Devanagari transliteration because it relies on diacritical marks in Urdu texts for correct transliteration.

System	Testing Set	Development Set
DHMM	94.21%	94.63%
+Noisy-channel Model	**96.37%**	**96.72%**

Table 3: Token-level performance of noisy-channel model in resolving word ambiguity in Perso-Arabic to Devanagari transliteration.

As shown in Table 2, we have established a new best system for the transliteration of Hindi and Urdu texts bidirectionally. Our Devanagari to Perso-Arabic system outperforms SANGAM by 0.65%. There is also an improvement of 0.47% over SANGAM in case of Perso-Arabic to Devanagari transliteration. Out of the two transliteration models, Perso-Arabic to Devanagari performs worst because of the missing vowels in the Urdu texts. Our noisy-channel model improved the results of our basic perceptron model for Perso-Arabic to Devanagari transliteration. It improved the accuracy by an absolute 2% on the test set. This clearly shows how often homographs are generated due to missing vowels in Perso-Arabic script.

3.1.3 Extrinsic Evaluation

As we already mentioned, the script that fares well in an extrinsic evaluation on a dependency parsing pipeline will be chosen for the common representation of Hindi and Urdu texts. For training and evaluation in each script, we made two copies of Hindi and Urdu training and evaluation sets. For each module, we trained two models–one in each script. In Table 4, we report the results on the respective evaluation sets in both scripts. Besides using predicted POS and chunk tag features for chunking and parsing, we also conduct experiments using the gold features. This is important for capturing the impact of orthographic representation on each module independently.

Data	POS tagging	Chunking	Parsing (LAS)
		Gold Features	
Hindi Devanagari	**96.48**	**98.40**	91.70
Hindi Perso-Arabic	96.00	98.35	91.52
Urdu Perso-Arabic	93.13	96.62	88.08
Urdu Devanagari	**93.42**	**96.65**	88.38
		Predicted Features	
Hindi Devanagari	-	**97.84**	**88.32**
Hindi Perso-Arabic	-	97.58	87.95
Urdu Perso-Arabic	-	95.60	81.67
Urdu Devanagari	-	**96.03**	**82.28**

Table 4: Performance of different modules of a dependency parsing pipeline trained and evaluated on Hindi and Urdu treebank data in Devanagari and Perso-Arabic scripts.

The results presented in Table 4 clearly favor Devanagari script over Perso-Arabic for the orthographic representation of Hindi and Urdu texts. For all the modules, accuracy decreases in Perso-Arabic script, while there is a significant increase in accuracy in Devanagari script. The reason mainly lies in the fact that Devanagari represents information explicitly while Perso-Arabic script does not. Devanagari is a phonetic script which represents phonemes uniquely and explicitly. Perso-Arabic on the other hand is not a phonetic script. The better results in Devanagari script clearly would, therefore, be due to the less ambiguous representation of words in this script as can be seen in Table 5. The table represents the impact of transliteration on lexical and POS-tag merging.

Script	**Lexical Merging** (%)	**Tag Ambiguity** (%)
Hindi in Perso-Arabic	-11.87	+3.12
Urdu in Devanagari	+11.26	-2.51

Table 5: Comparison of lexical and POS-tag merging rates in Devanagari and Perso-Arabic transliterated data.

We define *lexical merging rate* as the amount of percentage drop in the size of the vocabulary after transliteration. Similarly *tag ambiguity* captures the increase in ambiguity of POS categories due to lexical merging. Both *lexical merging* and *tag ambiguity* rates are higher in the case of Devanagari to Perso-Arabic transliteration, which explains the drop in accuracies when the models are trained and evaluated on data represented in Perso-Arabic script. There is an 11% drop in type counts when we transliterate HDTB data in Perso-Arabic, while on the other hand, there is a similar percentage increase in

vocabulary when UDTB in represented in Devanagari. Interestingly, not all the lexical merging/expansion leads to/resolves syntactic ambiguity. In HDTB, Perso-Arabic script created 3% homographs which are syntactically ambiguous, while Devanagari scripts resolves ambiguity for around 3% words in UDTB. Given these statistics, it seems reasonable to conclude that Devanagari is better suited for automatic text processing of Hindi and Urdu texts. However, it is also interesting that the POS models are able to resolve most of the syntactic ambiguity created by Perso-Arabic script.

4 Resource Sharing and Augmentation

In the above section, we empirically showed that the Devanagari script can serve as a common representation for both Hindi and Urdu texts without harming the performance of text processing applications such as a POS tagger and a parser. Given the fact that Hindi and Urdu are syntactically or grammatically similar, resource sharing and augmentation should become feasible just by removing the script barrier. A common orthographic representation would, however, only affect that part of Hindi and Urdu vocabularies which is shared. It will not fill the lexical gaps. It is the lexical differences between Hindi and Urdu texts that leave them mutually unintelligible (Masica, 1993). The severity of lexical differences can be clearly seen by comparing the OOV rates of Hindi and Urdu evaluation sets. As shown in Table 6, almost half of the tokens in both Hindi and Urdu development sets are missing from the Urdu and Hindi training sets respectively. Such excessive OOV rates would worsen the problem of lexical data sparsity for any statistical model.

Training	Development	OOV (%)
Hindi	Urdu	**51.6**
Urdu	Urdu	15.40
Urdu	Hindi	**62.76**
Hindi	Hindi	22.06

Table 6: OOV rates of Hindi and Urdu development sets. Both Hindi and Urdu data sets are represented in Devanagari.

Lexical data sparseness is considered as one of the major challenges in tackling the problem of data sparsity in data-driven approaches to natural language processing. A common approach to bridge the lexical gaps between the source and the target data is to use distributional similarity-based methods. The distributional similarity methods exploit Harris' distributional hypothesis which states that words that occur in the same contexts tend to have similar meanings (Harris, 1954). To address the lexical differences between Hindi and Urdu, distributional similarity-based methods seem as an appropriate choice. Consider a case of **pratikshā** and **intizār**, both words are semantic equivalents and have the same meaning i.e., **wait**. **pratikshā** is a Sanskrit word used in Hindi texts, while **intizār** is its Perso-Arabic equivalent used in Urdu texts. In both Hindi and Urdu, **pratikshā** and **intizār** form complex predicates with similar light verbs. One such complex predicate is **pratikshā/intizār kar** 'wait do'. The complex predicate takes a genitive-marked theme argument, licenses ergative case on its agentive argument in perfective aspect and can take similar tense, aspect and modal auxiliaries. Even though, **pratikshā** and **intizār** are different word forms, they have identical syntactic distributions which could be used as an approximation of their semantic similarity.

To capture the similarity between Sanskrit and Perso-Arabic words in Hindi and Urdu vocabularies, we could apply distributional similarity methods on the union of harmonized (represented in same script) Hindi and Urdu corpora. Augmenting source domain corpus with the target domain data to learn distributional representation of words is a common practice to address lexical sparseness encountered in domain adaptation tasks (Candito et al., 2011; Plank and Moschitti, 2013). We could also use bilingual word clustering or word embedding approaches which have been used to address the loss of lexical information in delexicalized parsing (Täckström et al., 2012; Xiao and Guo, 2014). In case of Hindi and Urdu, the former approach is more simple and direct way to capture the distributional similarity. The similar grammar and partially shared vocabularies would ensure semantically similar words of Hindi and Urdu are assigned similar distributional representations. The cross-lingual approaches, on the other hand, are computationally complex, while capture the distributional similarity indirectly using a seed bilingual

lexicon.

The distributional similarity can be incorporated in a statistical model either by using word clusters or word vectors (Turian et al., 2010). Similar to Collobert et al. (2011) and Chen and Manning (2014), we represent lexical units in the input layer of our neural network model by the word embeddings instead of one-hot vectors. We augment Hindi monolingual data with the transliterated Urdu data and use word2vec toolkit[6] to learn the word embeddings. The toolkit provides an efficient implementation of the continuous bag-of-words (CBOW) and skip-gram (SG) approaches of Mikolov et al. (2013) to compute distributed representation of words. To learn distributed word representations, we considered context windows of 2 to 5 words to either side of the central element. We varied vector dimensionality within the 50 to 100 range in steps of 10. The model choice, window size and vector dimensionality were selected on the development set in a POS tagging task. The optimal parameters are: learning algorithm as skip-gram, window size as 1, embedding dimensionality as 50 and minimum word frequency as 2.

Once we have represented both Hindi and Urdu texts in same distributional space, we model sharing and augmentation of their resources as a supervised domain adaptation task. Supervised domain adaptation assumes the availability of annotated data in both source and target domains to improve model performance in the target domain. We discuss the best practices in supervised domain adaptation and evaluate their performance for Hindi and Urdu resource sharing and augmentation. An overview of the supervised domain adaptation methods can be found in (Daumé III, 2007), which we repeat here briefly.

- The SRCONLY method trains a single model on the source data while ignoring any target data.
- The TGTONLY method trains a single model only on the target data and acts as the baseline.
- In the ALL method, we simply train our model on the union of the Hindi and Urdu training sets.
- In the WEIGHTED method, we weight the instances of a data set with larger instances to train a single unbiased model. The weights are appropriately chosen by cross-validation.
- In the LININT method, we train SRCONLY and TGTONLY models and linearly interpolate their predictions at the inference time. The interpolation weights are tuned via cross-validation.
- PRIOR method relies on the use of SRCONLY model as a prior on the weights of the target model while training. In our neural network model, we simply replace the regularization term with $\lambda||w - w^s||_2^2$ where w^s is the weight vector from the SRCONLY model.

Among these supervised methods, SRCONLY will address resource sharing, while WEIGHTED, LININT and PRIOR are the methods for resource augmentation.

4.1 Experiments and Results

In any non-linear neural network model, we need to tune a number of hyperparameters for an optimal performance. Tuning these parameters is usually as cumbersome as designing appropriate feature combinations in a linear model. The hyperparameters include number of hidden units, choice of activation function, learning rate, dropout, dimensionality of input units, etc. Furthermore, we had to tune these parameters for each individual task separately. Interestingly, we found that the hyperparameters take similar optimal values for all the three tasks. This could be due to the fact that POS tagging, chunking and parsing are correlated to each other. There is, however, some variation in learning rate and dropout. The optimal parameters include: 200 hidden units, rectilinear activation function, 20 batch size, 20 dimensional non-lexical input units, 0.01 learning rate for POS tagging and chunking and 0.3 for parsing, and 15 training iterations.

After tuning the hyperparameters of our neural network models, we trained multiple models for both Hindi and Urdu to evaluate the performance of each domain adaptation method. To use uniform POS and chunk features, we used 10-fold jackknifing to assign these features to the training data instead of using gold features. For chunking, we used the auto POS features from the best performing POS tagger. Similarly for parsing, we used the best POS and chunk taggers to generate these features. The results of our experiments are reported in Table 7.

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

Source	Target	SRCONLY	TGTONLY	ALL	WEIGHTED	LININT	PRIOR
POS tagging							
Hindi	Urdu	89.34	93.42	93.74	93.77	**93.93**	93.52
Urdu	Hindi	86.06	96.48	96.50	96.54	**96.61**	96.22
Chunking							
Hindi	Urdu	92.53	96.03	96.40	96.31	**96.52**	96.13
Urdu	Hindi	90.27	97.64	**97.77**	97.63	97.71	97.44
Dependency Parsing							
Hindi	Urdu	78.93	82.28	82.64	82.53	**82.65**	82.32
Urdu	Hindi	75.12	88.32	**88.41**	88.37	88.39	88.18

Table 7: Results of different supervised domain adaptation methods on Hindi and Urdu resource sharing and augmentation.

For resource augmentation, it is encouraging to note that all the methods led to some improvement over the TGTONLY baseline. Surprisingly, PRIOR method did not perform well in our case. It was one of the best methods reported by Daumé III (2007) when used with a linear model. On the other hand, LININT consistently performed better than other methods. Nevertheless, we achieved substantial improvements in accuracies in all the three tasks for both Hindi and Urdu. Particularly, improvements in Urdu are more prominent. Our augmentation results clearly show that Hindi and Urdu annotations can be complementary to each other. In our case, large number of annotations in HDTB proved useful for parsing of Urdu, which comparatively has a smaller-sized treebank.

The SRCONLY models, under resource sharing experiments, did not perform at par with the TGTONLY baseline models. The performance is still better for Urdu (as a target domain) than it is for Hindi which again could be attributed to the sheer size of the Hindi training data. The larger gaps in accuracies between the SRCONLY and TGTONLY models can be attributed to domain shift problem inherent in data-driven approaches. To empirically verify it, we explored the impact of domain shift on the Hindi tagger and parser trained on newswire data by applying it on Hindi texts other than news articles. For this task we used annotated data from four different domains of Hindi which include cricket, recipes, gadgets and box office. Each domain contains around 500 hundred sentences annotated with POS, chunk and dependency structures. The accuracies of the Hindi parser and tagger on these domains is shown in Table 8. If we compare the accuracies of the Hindi POS tagger and parser on Urdu and the four domains, it appears that the performance on the Urdu test set is comparable. The tagging and parsing accuracies on gadget and recipe data are lower than the accuracies on the Urdu test data. This encourages us to suggest that we can consider Hindi and Urdu as two separate domains representing the same language at least in computational sense and use their tools interchangeably instead of building tools for them separately.

Domains	Parsing			POS tagging
	UAS	LS	LAS	
Cricket	87.90	85.26	79.72	94.02
Box-office	86.64	83.43	78.98	89.55
Gadget	83.27	81.30	75.35	85.93
Recipe	81.12	79.31	71.37	88.95
Urdu	86.13	83.15	78.93	89.34

Table 8: Comparison of parsing and tagging accuracy of Hindi parser and POS tagger on Urdu test data and four different domains of Hindi.

5 Related Work

Recently, there have been several attempts at leveraging the similarity between Hindi and Urdu for sharing and interoperability of their individual resources. Most of the works suggest that Hindi and Urdu resources can be used interchangeably with some modifications (Sinha, 2009; Visweswariah et al., 2010). Sinha (2009) show that an English-Urdu machine translation system can be easily build by using Hindi as a bridge language. Urdu translations of English sentences are derived from the output of an existing English-Hindi MT system. They use lexical mappings between Hindi and Urdu words, lexical and syntactic disambiguation rules, and a transliteration module for converting the MT output to Urdu.

Adeeba and Hussain (2011) used a transliteration-based approach to create an Urdu WordNet from an existing Hindi WordNet (Narayan et al., 2002). They used a rule-based transliteration system to convert

the lexical database of the Hindi WordNet to Urdu (Perso-Arabic). They manually pruned typical Sanskrit words that are not used in Urdu texts and added additional entries specific to Urdu. Similarly Ahmed and Hautli (2010) proposed to use a simple transliteration-based approach to access Hindi WordNet for Urdu texts, instead of creating a separate Urdu WordNet.

Mukund et al. (2010) have explored the use of Hindi specific POS tagger and chunker on Urdu text. Both training and testing data are transliterated to a common form for model transfer. They show that Hindi POS tagger performs worst on Urdu text which suffered an absolute loss of 32.5% in accuracy from an Urdu specific tagger. Their observation is same for chunking, however the results are not reported due to lack of an evaluation set. Visweswariah et al. (2010) explore complementary role of linguistic resources present in Hindi and Urdu for better system performance. They show improvements in machine translation, bitext alignment and POS tagging.

Besides these empirical studies that show Hindi and Urdu can mutually benefit from sharing their individual resources, there are also arguments against any such endeavour (Riaz, 2009; Mukund et al., 2010; Prasad et al., 2012). In a theoretical study, Riaz (2009) argues that lexical divergences between Hindi and Urdu hinder the interoperability of their computational resources. On the basis of the extensive variation in their vocabularies, he argues that any method that relies on maximum likelihood estimation may not work jointly for both Hindi and Urdu.

Our work differs from these related works in multiple ways. Firstly, we showed that Urdu text processing suffers substantially by the use of an ambiguous script. For computational purposes, we argued to represent Urdu texts in Devanagari instead of Perso-Arabic. To that end, we proposed an efficient and accurate transliteration method that resolves the lexical ambiguity due to missing short vowels and ambiguous characters in Perso-Arabic writing. Secondly, in addition to resource sharing, we also showed that resource augmentation can improve the performance of individual text processing modules of Hindi and Urdu. Furthermore, to mitigate the effect of lexical sparsity, we also used distributional similarity-based method besides transliteration.

6 Conclusion

In this paper, we explored the possibility of sharing and augmenting annotation resources of Hindi and Urdu to improve the performance of their individual text processing modules. To bridge the script and lexical differences between their texts, we proposed a simple and efficient technique based on script transliteration and distributional similarity. We showed that we can easily abstract away from the orthographic differences between Hindi and Urdu texts by representing their lexicons in a same distributional space. As a proof-of-the-concept, we showed that by bridging their script and lexical differences we can enhance the performance of Hindi and Urdu dependency parsers by simply merging their training data. Moreover, our experimental results suggest that Hindi and Urdu parsers can even be used interchangeably with reasonable accuracies.

In the future, we would like to explore the possibility of merging the semantic role annotations in HDTB and UDTB for training a better semantic role labeler. It would also be interesting to see whether our observations related to resource sharing between Hindi and Urdu would hold for applications other than parsing.

References

Farah Adeeba and Sarmad Hussain. 2011. Experiences in building the Urdu WordNet. In *Proceedings of Asian Language Resources collocated with IJCNLP*.

Tafseer Ahmed and Annette Hautli. 2010. Developing a basic lexical resource for Urdu using Hindi WordNet. In *Proceedings of CLT10, Islamabad, Pakistan*.

Akshar Bharati, Vineet Chaitanya, Rajeev Sangal, and KV Ramakrishnamacharyulu. 1995. *Natural language processing: a Paninian perspective*. Prentice-Hall of India New Delhi.

Akshar Bharati, Rajeev Sangal, Dipti Misra Sharma, and Lakshmi Bai. 2006. Anncorra: Annotating corpora

guidelines for POS and Chunk annotation for Indian languages. Technical report, (TR-LTRC-31), LTRC, IIIT-Hyderabad.

Riyaz Ahmad Bhat, Rajesh Bhatt, Annahita Farudi, Prescott Klassen, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Misra Sharma, Ashwini Vaidya, Sri Ramagurumurthy Vishnu, et al. 2015. The Hindi/Urdu treebank project. In *Handbook of Linguistic Annotation*. Springer Press.

Rajesh Bhatt, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Misra Sharma, and Fei Xia. 2009. A multi-representational and multi-layered treebank for Hindi/Urdu. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 186–189. Association for Computational Linguistics.

Marie Candito, Enrique Henestroza Anguiano, and Djamé Seddah. 2011. A word clustering approach to domain adaptation: Effective parsing of biomedical texts. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 37–42. Association for Computational Linguistics.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 1, pages 740–750.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of Association of Computational Linguistics*.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 959–976.

Zellig S Harris. 1954. Distributional structure. *Word*.

Girish Nath Jha. 2010. The TDIL program and the Indian language corpora initiative (ILCI). In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC 2010). European Language Resources Association (ELRA)*.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

Gurpreet Singh Lehal and Tejinder Singh Saini. 2012. Development of a complete Urdu-Hindi transliteration system. In *Proceedings of the 24th International Conference on Computational Linguistics (Posters)*, pages 643–652.

Gurpreet Singh Lehal and Tejinder Singh Saini. 2014. Sangam: A Perso-Arabic to Indic script machine transliteration model. In *Proceedings of the 11th International Conference on Natural Language Processing*.

Muhammad G Malik, Christian Boitet, and Pushpak Bhattacharyya. 2008. Hindi Urdu machine transliteration using finite-state transducers. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 537–544. Association for Computational Linguistics.

Colin P Masica. 1993. *The Indo-Aryan Languages*. Cambridge University Press.

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

Smruthi Mukund, Rohini Srihari, and Erik Peterson. 2010. An information-extraction system for Urdu—a resource-poor language. *ACM Transactions on Asian Language Information Processing (TALIP)*, 9(4):15.

Dipak Narayan, Debasri Chakrabarti, Prabhakar Pande, and Pushpak Bhattacharyya. 2002. An experience in building the IndoWordNet-a WordNet for Hindi. In *Proceedings of the First International Conference on Global WordNet, Mysore, India*.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (acl)*. Association for Computational Linguistics.

KVS Prasad, Shafqat Virk, John Camilleri, Krasimir Angelov, Kaarel Kaljurand, Olga Caprotti, and Aarne Ranta. 2012. Computational evidence that Hindi and Urdu share a grammar but not the lexicon. In *Proceedings of the 3rd Workshop on South and Southeast Asian Natural Language Processing (SANLP), COLING 2012*, volume 7427.

Kashif Riaz. 2009. Urdu is not Hindi for information access. In *Proceedings of the Workshop on Multilingual Information Access, SIGIR*.

Hassan Sajjad, Nadir Durrani, Helmut Schmid, and Alexander Fraser. 2011. Comparing two techniques for learning transliteration models using a parallel corpus. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 129–137, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

R Mahesh K Sinha. 2009. Developing English-Urdu machine translation via Hindi. In *Proceedings of the Third Workshop on Computational Approaches to Arabic-Script-based Languages*.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 477–487. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Karthik Visweswariah, Vijil Chenthamarakshan, and Nandakishore Kambhatla. 2010. Urdu and Hindi: Translation and sharing of linguistic resources. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1283–1291. Association for Computational Linguistics.

Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, page 119.

Dmitry Zelenko and Chinatsu Aone. 2006. Discriminative methods for transliteration. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 612–617. Association for Computational Linguistics.

Deeper syntax for better semantic parsing

Olivier Michalon[⋆] **Corentin Ribeyre**[†∗] **Marie Candito**[◇] **Alexis Nasr**[⋆]
⋆ Aix Marseille Univ, CNRS, Centrale Marseille, LIF, Marseille, France
† Faculty of Humanities, University of Geneva, Switzerland
◇ Alpage, Université Paris Diderot, Sorbonne Paris Cité, INRIA, Paris, France
⋆ firstname.lastname@lif.univ-mrs.fr
† firstname.lastname@unige.ch
◇ firstname.lastname@inria.fr

Abstract

Syntax plays an important role in the task of predicting the semantic structure of a sentence. But syntactic phenomena such as alternations, control and raising tend to obfuscate the relation between syntax and semantics. In this paper we predict the semantic structure of a sentence using a deeper syntax than what is usually done. This deep syntactic representation abstracts away from purely syntactic phenomena and proposes a structural organization of the sentence that is closer to the semantic representation. Experiments conducted on a French corpus annotated with semantic frames showed that a semantic parser reaches better performances with such a deep syntactic input.

1 Introduction

FrameNet (Baker et al., 1998) is an English resource containing a set of inter-related semantic frames, each frame containing a set of semantic roles (*frame elements* in FrameNet's terminology). Frames offer semantic generalizations over individual predicates, since different lexical units can evoke the same frame, and semantic roles offer generalizations over syntactic arguments. Hence FrameNet parsing can be viewed as mixing predicate disambiguation and semantic role labelling.[1]

Although FrameNet is more semantically-oriented than other semantic role labeling resources such as PropBank (Palmer et al., 2005), syntactic information has been shown to be decisive for predicting (FrameNet) semantic roles since the early days of the task (Gildea and Jurafsky, 2002). Linking regularities provide the theoretical justification of this result: there exist regularities in how semantic arguments are realized in syntax. Yet it is well known that the mapping from syntactic arguments to semantic ones is not straightforward. First, lexical idiosyncrasies can come into play, for instance the *Addressee* of communication verbs may correspond to the indirect object for verbs like *to say* and to the direct object for a verb like *to inform*. Second, it is also well known that surface syntax exhibits variation that can obfuscate regularities. For instance though the *Speaker* is generally the subject of communication verbs, this does not hold when the verb is passivized. This difference disappears if syntactic alternations are neutralized, and the "canonical" diathesis of a verb is made explicit: the *Speaker* is the canonical subject in both active and passive voices.

In this paper, we investigate the syntax-semantic interface in FrameNet annotated data, and study the impact of using "deeper" syntactic features to predict semantic frames and roles. More precisely, we take advantage of a deep syntactic dependency graphbank for French (Candito et al., 2014b; Ribeyre et al., 2014), which provides a level of representation that abstracts away from purely syntactic variation. The main contributions of the paper are (i) a comparison of the syntax/semantic regularities observed when using plain "surface" syntax to those observed when using deep syntax and (ii) a study of how and why the switch from surface to deep syntax impacts FrameNet semantic parsing. In the remaining of the

[∗] All of his work has been done during his PhD at Alpage.

[1] In the following, we will use shorter terms than those of FrameNet terminology : we use the term *trigger* for a lexical unit that can evoke a frame, the term *role* for frame element, and *role filler* for the sequence of words that instantiates a role.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 409–420, Osaka, Japan, December 11-17 2016.

paper we will use the terms Surface Syntactic Representations (SSR) and Deep Syntactic Representations (DSR) to refer to surface syntactic trees and deep syntactic graphs.

Using abstract syntactic representations as an intermediate representation level between syntax and semantics has been proposed in different theoretical frameworks, such as derived trees of Tree Adjoining Grammars (Joshi and Schabes, 1997) or deep syntactic structures of the Meaning Text Theory (Mel'čuk, 1988). But we only found few works showing, empirically, that using such representations can effectively help predict the semantic roles of predicates.Two of them concern PropBank semantic role labeling. The early (Gildea and Hockenmaier, 2003) work shows that using CCG-derived predicate-argument features predicted by a CCG parser improves the identification of core PropBank arguments. Vickrey and Koller (2008) investigate the use of simplified syntactic paths and report a slight improvement when applying transformation rules to simplify phrase-structure parses.

As far as FrameNet parsing is concerned, we don't know of any work using more abstract syntactic input than plain "surface" syntactic trees, whether phrase-structure (Gildea and Jurafsky, 2002) or dependency trees (Johansson and Nugues, 2007; Das et al., 2014). We focus on French, first because of the availability of the afore-mentioned DSR, and second because in the French FrameNet corpus (Djemaa et al., 2016) the annotated semantic roles are restricted to essential arguments. On the contrary, both essential ("core") and non essential participants are annotated in the English FrameNet, including modifiers such as time, location, purpose etc... But syntactic variation such as syntactic alternations, VP coordination, control etc... does concern primarily the most salient grammatical functions (subject, direct object, indirect object etc...), which are typically the ones that essential arguments bear. Hence, neutralizing syntactic variation is expected to have an impact primarily on essential semantic roles.

The structure of the paper is the following: in section 2, we present (i) the French FrameNet corpus that we use, (ii) the deep syntactic representations whose impact for FrameNet parsing we wish to investigate, (iii) we compare the syntax/semantic interface when using surface dependency trees and deep dependency graphs and (iv) we compare such deep representations to other deep representations proposed mainly for English. Section 3 and 4 are devoted to the frame-semantic parser and the deep-syntactic parsing architecture we used. We present and discuss the frame-semantic parsing experiments in section 5, and conclude in section 6.

2 Deep syntax and frame semantics

2.1 French FrameNet corpus

The French FrameNet annotated corpus (Djemaa et al., 2016) was produced within the ASFALDA ANR project on French shallow semantic parsing[2]. Two corpora have been annotated with frames and roles: the French Treebank (Abeillé and Barrier, 2004) (hereafter FTB) and the Sequoia Treebank (Candito and Seddah, 2012b). The first one contains $18,535$ sentences from the *Le Monde* newspaper. The second one is much smaller and was originally created for domain adaptation experiments for statistical parsing. It contains $3,099$ sentences from a regional newspaper, from Europarl, from the European Medicine Agency and from the French Wikipedia.

The French FrameNet corpus annotation is restricted to four semantic domains: commercial transactions, cognitive stances, verbal communication and causality. For all lexical items of the lexicon, associated with frames pertaining to these domains, the first 100 occurrences have been annotated. For each occurrence to annotate, annotators were proposed the pertaining frames, plus a special null frame for the cases in which the occurrence evoked a sense not pertaining to the four domains. We provide quantitative characteristics of the corpus in Table 1. The semantic annotations cover 105 frames, and the lexicon extracted from the annotations contains 1112 frame/lemma pairs (i.e. senses). The corpus contains $15,990$ annotated frame occurrences (plus 8727 occurrences of the null frame[3]), 56.2% of which correspond to verbal triggers and 33.0% to noun triggers.

[2]Version 1.0, https://sites.google.com/site/anrasfalda/

[3]The null frame is used to annotate words that would trigger a frame that has not been defined yet. Note that trigger occurrences ahead of the first 100 occurrences do not bear any frame at all, and are not to be considered.

Nb. Sentences	21 634	Nb. annot. frame occurrences	15 990
Nb. Tokens	625 951	Nb. annot. role occurrences	24 147
Nb. distinct annot. frames	105	Mean nb. annot. frames per lemma	18.3
Nb. distinct annot. lemma/frame pairs	1112	Median nb. annot. frames per lemma	6

Table 1: Quantitative characteristics of the French FrameNet annotated corpus (excluding the null frame).

2.2 Deep syntactic representations

We now turn to the deep syntactic graphbank that we use as an alternative syntactic representation for FrameNet parsing. DSRs are available for the two corpora that were annotated with frames and roles (the Sequoia corpus and the French Treebank). The development set of the Sequoia corpus was used to set up the deep syntactic annotation scheme, as well as a surface-to-deep syntax conversion module (Ribeyre et al., 2014) based on a graph-rewriting tool (Ribeyre et al., 2012). While the DSRs were manually validated for the full Sequoia corpus, those for the FTB sentences were automatically obtained using this surface-to-deep syntax conversion module, described in section 4. The quality of the resulting DSRs is high enough to use them as a reference for evaluation[4].

Candito et al. (2014b) define DSRs as dependency graphs which abstract away from purely syntactic variations, as far as verbal and adjectival predicates are concerned, making explicit their predicate-argument structure. SSR and DSR differ on three aspects:

- **Saturation:** The predicate-argument structure of all verbs is saturated for verbs that are not the head of a saturated clause (e.g. coordinated verbs, infinitival verbs). Any element that does not locally depend on the verb but that would do so if the verb were the head of a clause is added as (deep) dependent of the verb. First, this means that arguments shared by several verbs, e.g. in elliptic coordinations or control verb constructions, are attached to all their deep governors. For instance in *Paul loves to eat pies*, *Paul* is the subject of both *loves* and *eat*, and in *Paul loves and often eats pies*, the two coordinated predicates *loves* and *eats* share the same subject *Paul* and direct object *pies*. Second, noun-modifying verbs get the noun as deep syntactic dependents. For instance in *People born before 1969 fear the moon*, the verb *born* gets *People* as subject.
- **Syntactic alternations:** Productive syntactic alternations are neutralized. Syntactic arguments of verbs get their *canonical* grammatical function, which may differ from the observed grammatical function. The most frequent alternations are the passive alternation, then middle and neuter alternations, each marked with a *se* clitic. Other more marginal alternations are impersonal, impersonal passive and causatives. Note that alternations frequently interact with predicate-argument structure saturation. For instance, in *Paul would like to get an interview and then be hired*, *Paul* is added as canonical subject of *get* but canonical object of *hired*. In noun-modifying participial clauses, if the verb is transitive, the past participle is analyzed as a passive. For instance in *People hired after march are few*, the verb *hired* gets *People* as canonical direct object (see also the verb *poussée (urged)* in figure 1).
- **Abstraction:** Most grammatical markers are discarded. Auxiliaries in particular are replaced by deep features on the lexical verb. Empty prepositions and complementizers are bypassed For instance in *Le chat sourit à la souris (The cat smiles to the mouse)*, the preposition *à* is discarded, and the indirect object of the verb is the NP *la souris (the mouse)*.

By extension, the subjects[5] of adjectives are made explicit in the DSRs.

The DSRs are closer to predicate-argument structures than SSRs are, yet predicates are not disambiguated, and thus canonical grammatical functions are used and not semantic roles.

Figure 1 shows the SSR, DSR and FrameNet annotations for one sentence (the role fillers are reduced to their syntactic head, cf. section 2.3). It can be seen, for instance, that the past participle *poussée*

[4]Ribeyre et al. (2014) report a 98.4 Fscore evaluated on manually validated DSRs for 200 sentences from the FTB.

[5]The subject of the adjective is either the noun it modifies in case of an attributive adjective, or the subject of the copular verb in case of a predicative adjective.

(urged) modifies the proper noun *EDF* in both syntactic representations, but the noun is its canonical direct object in the deep representation.

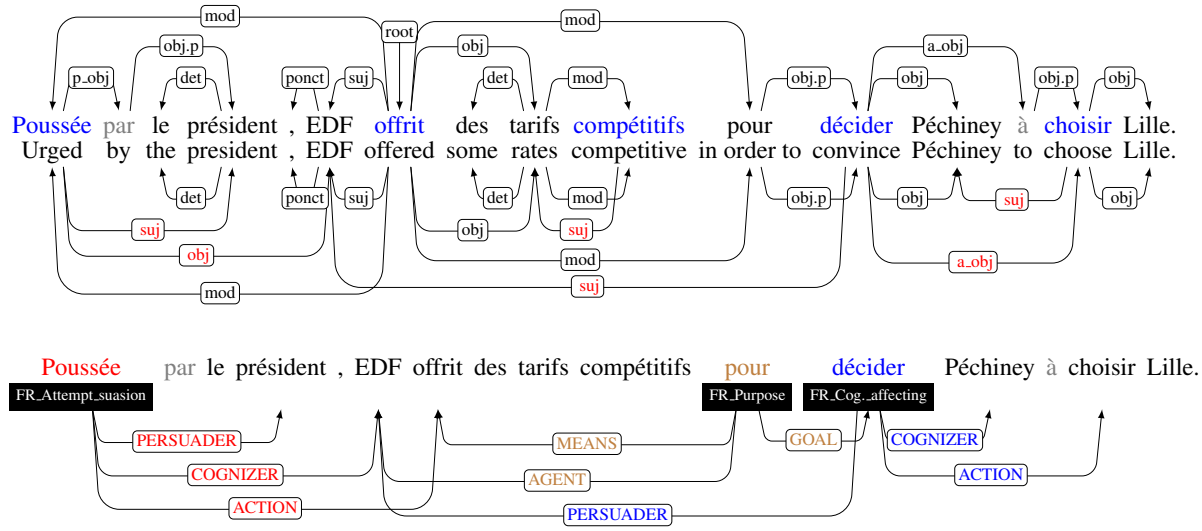


Figure 1: Example of syntactic and semantic annotations for a sentence. **Top**: Surface and deep syntactic representations (edges above: SSR, edges below: DSR). Verbs and adjectives are in blue. Tokens discarded in the DSR are in gray. Grammatical functions added and/or normalized when switching from SSR to DSR are in red. **Bottom**: frame and role annotations (from trigger to syntactic head of role fillers), for three triggers (2 verbs and 1 preposition).

2.3 Syntax semantics interface

As already mentionned in the introduction, syntax is a major feature when predicting semantic roles. Reducing the variety of syntactic features might therefore help fighting against data sparsity and improve this prediction task. Because they are meant to neutralize syntactic variations, DSRs are good candidates for such a reduction. In all the following, we will use as syntactic features the syntactic paths that link a frame trigger to the syntactic head of each of its role fillers (the head is taken as the leftmost root of the subtrees composing the role filler). In this section we will measure how much the use of DSRs helps to reduce the variety of syntactic paths. In order to do so, we will compute the entropy of the probability distribution of the syntactic paths that correspond to a role. Two entropies will be compared: the *surface* syntactic entropy and the *deep* syntactic entropy.

Before defining surface and deep syntactic entropy, we need to define precisely the notions of semantic path: deep syntactic path (DSP) and surface syntactic path (SSP). Given sentence S that contains an occurrence of frame F having word t as trigger and which role R is filled by a sequence of tokens W (the role filler, which may be discontinuous). We will call the tuple (t, R, W) a semantic path of S.

We associate to every semantic path $p = (t, R, W)$ of sentence S a surface syntactic path $SSP(p)$ and a deep syntactic path $DSP(p)$, which link the trigger to the head of the role filler W, noted $h(W)$.

$SSP(p)$ is the shortest path linking t and $h(W)$ in the SSR of S. The SSR being a tree, such a path exists and is unique, it is the sequence of dependencies that link t to $h(W)$. We represent it formally as a sequence of tuples *(direction,label)*, where *direction* is + if a dependency is traversed from the governor to the dependent and - otherwise.

Defining DSP is not as straigtforward: the DSR being a graph, there can be several shortest paths[6] from t to $h(W)$. We select a unique shortest path using the following hierarchy of grammatical functions to rank paths of length one: $suj > obj > ats/ato > a\_obj > de\_obj > p\_obj > mod$.[7] The left part of Table 2 shows the five most frequent syntactic paths, when the trigger is a verb, using either surface or

[6] Actually, since deep syntax can form non connected oriented graphs, there can be no path at all between t and $h(W)$ (due to errors in deep syntax or in semantic annotations). We use the special tag *__no_path__* in such cases.

[7] With $a > b$ meaning a has priority over b and a/b meaning a has the same priority as b.

surface syntax		deep syntax	
(+suj)	25.02%	(+suj)	33.1%
(+obj)	17.01%	(+obj)	32.79%
(-mod)	8.04%	(+a_obj)	4.73%
(+obj,+obj.cpl)	4.42%	(-mod)	3.15%
(+a_obj,+obj.p)	4.09%	(+mod,+obj.p)	2.46%
Total	58.58%	Total	76.23%

role	synt. head	syntax	path
PERSUADER	*EDF*	surface	(-obj.p,-mod,+suj)
		deep	(+suj)
COGNIZER	*Péchiney*	surface	(+obj)
		deep	(+obj)
ACTION	*choisir*	surface	(+a_obj,+obj.p)
		deep	(+a_obj)

Table 2: **Left**: Most frequent gold syntactic paths in training corpus, when the trigger is a verb. **Right**: surface and deep paths for the *FR_cognizer_affecting* frame evoked by *décider* in the sentence of Figure 1.

deep syntax. We can see that the distribution of paths is much more compact when using deep syntax : the first five paths represent more than 76% of deep paths, compared to 58% for surface paths. (obj) and (suj) paths represent 42.03% of SSP but 65.89% of DSP (in order to reach that coverage with SSP, the 8 most frequent SSP are needed).

The right part of Table 2 shows the deep and surface paths corresponding to the roles of the *FR_cognizer_affecting* frame evoked by *décider* in the sentence of Figure 1.

In order to measure the reduction of the variety of the syntactic realization when moving from surface to deep syntax, we have computed the average entropy over all roles R of the probability distributions $P(p|R)$ where p is a path. These distributions have been estimated on the training corpus. The average entropy when computed on surface syntax is equal to 1.65 and to 1.32 when computed on deep syntax. This decrease is a direct measure of the normalizing effect of the deep syntactic frame we used. Note though that an entropy reduction could be artificially obtained by neutralizing meaningful syntactic distinctions. Yet, the DSRs were designed following syntactic principles and experiments in section 5 are intended to check that such a normalization is indeed beneficial for downstream semantic parsing.

2.4 Comparison with other deep representations

There has been various previous works proposing deeper syntactic annotation schemes that can represent information absent in plain constituency or dependency trees, such as long-distance dependencies, subjects of control verbs, subjects of coordinated verbs etc. This additional information is sometimes viewed as pertaining to semantic representations, sometimes retained as still syntactic.

English has been the first focus language, along with Czech thanks to the Prague Dependency Treebank (Hajič et al., 2006). For English, several works automatically convert Penn Treebank constituency trees into deeper representations, based on lexicalized grammar formalisms such as LFG, CCG or HPSG. Cahill et al. (2004) automatically construct LFG f-structures from PTB trees, a work adapted for various other languages including French (Schluter and van Genabith, 2008). Hockenmaier and Steedman (2007) extracted a corpus of CCG derivations and dependency structures from the Penn Treebank. These two kinds of deeper representations do capture long distance dependencies, subjects of non finite verbs, argument sharing between coordinated verbs. When compared to the DSRs we use though, the main missing trait is the neutralization of syntactic alternations, which we believe is a major source for the syntactic path normalization effect described in section 2.3[8].

The Stanford dependencies (SD, De Marneffe and Manning (2008)) constitute another proposal for obtaining dependencies not directly present in surface syntactic trees. The stanford parser comprises a dependency extraction system, which can output several variants of typed word-to-word dependencies, from plain dependency trees to more semantically-oriented graphs. The deepest variant ('collapsed with propagation of conjunct dependencies' variant) does cope with some of the aforementioned phenomena such as subject of infinitival verbs or coordinated verbs. Compared with the DSRs for French, the major differences are that syntactic alternations are not neutralized, and that all prepositions are collapsed and injected in the labels (while only void prepositions are collapsed in the French DSRs).[9]

[8]Passive alternations is by far the most frequent alternation, and also happens to be rather easy to identify, so we hypothesize that using such representations on top of passive neutralization would be an alternative to the DSRs we use.

[9]We actually did some unfruitful experiments on the English FrameNet data, comparing the use of syntactic features ex-

Taking a further step towards semantic representations, predicate-argument structure graphs such as those used for the Broad-Coverage Semantic Dependency Parsing task at SemEval 2014 (Oepen et al., 2014) are also very close to the DSRs we use, with respect to the covered linguistic phenomena. The three datasets used in this shared task are (i) predicate-argument semantic graphs extracted from the HPSG-grounded DeepBank of Flickinger et al. (2012), (ii) predicate-argument structures from the Enju HPSG Treebank [10], and (iii) the Prague Czech-English Dependency Treebank (Hajič et al., 2012). These three datasets differ in how far they differ from syntactic representations. While some traits are common to the DSRs we use, one major difference lies in the more semantically-oriented labelling of the word-word dependencies: the semantic arguments are simply numbered (arg0, arg1, etc...). We believe that in the absence of word sense disambiguation at the level of predicates, this plain numbering obfuscates syntactic clues that are crucial for FrameNet semantic role labelling. If we take a French example, the verb *convenir* has two senses (among others), in which the arguments bear different FrameNet roles, and which can be disambiguated by the canonical subcategorization frame: we have *X(subject) convenir à Y(a-object)* meaning "X suits Y" versus *X(subject) convenir de Y(de-object)* meaning "X admit to Y".

To sum up, while the various deep representations cited above do capture the topology of predicate-argument structures, by coping with major phenomena such as control verbs or coordinated verbs, the DSRs are appealing for framenet parsing for two reasons: first they are still syntactic in nature (they are thus recoverable deterministically from surface syntax, cf. section 4), while a semantic graph would represent a too sophisticated input for the task. Second, the DSRs use canonical grammatical functions, which are both more abstract than surface grammatical function labels, but do not obfuscate important syntactic clues for predicate and role disambiguation.

3 Semantic parser

The semantic prediction system (FastSem) is a baseline system based on a cascade of linear classifiers[11]. For every word w of a sentence, we proceed in two steps. A *frame* identification step (which frame (if any) does w trigger?) followed by a *role* identification step (which role (if any) is w the head of?). This architecture is based on two strong independence hypotheses: frames are independent from one another in a sentence and roles inside a frame are independent[12].

We chose to use a simple architecture as our focus here is to assess whether normalized syntactic paths help semantic parsing. It remains to be proved, although it can be easily supposed, that it would also help with less naive hypotheses.

In the first step we use for each lexical unit a different linear classifier, each using the following features: the fine- and coarse-grained PoS of the target word t, and for each word w of the sentence, its lemma, its PoS (fine and coarse) and the syntactic path that links t to w. The classifier used for the second step is frame specific. To predict the role of word f, we use as features the lemma and PoS (coarse and fine) of f, t's lemma and fine-grained PoS, the syntactic path between t and f, plus the combination of the syntactic path and the lemma of t.

4 Predicting deep syntax

In order to evaluate the impact of deep syntax on semantic parsing in realistic conditions, we need to obtain predicted deep syntactic representations. Although directly training a graph parser would be an option (as in (Ribeyre et al., 2016)), we retain the rule-based architecture that was used to bootstrap the deep syntactic annotations. Our motivation is to be able to apply the surface-to-deep rewriting rules step-by-step, in order to study the impact of each phenomenon.

tracted from two variants of SD (basic versus collapsed with propagation of conjunct dependencies). We concluded that the collapsed dependencies are not adapted for our purpose: they no not neutralize syntactic alternations, and multiply labels by collapsing all prepositions. We could measure that this has the result of actually increasing the entropy of the syntactic paths that correspond to a role. Preposition collapsing has a negative impact on predicting non essential semantic roles, such as temporal or locative modifiers.

[10] See http://kmcs.nii.ac.jp/enju

[11] The classifier library used is LIBLINEAR (Fan et al., 2008).

[12] These hypothesis are known to be too strong. For instance Das et al. (2014) show that collectively predicting all role fillers of a given frame occurrence improves performance.

	Surface dependency parser		Conversion to deep syntax			
			on predicted surf. parses		on gold surf. parses	
	UAS	LAS	UF1	LF1	UF1	LF1
trainjk	86.9	83.5	83.5	80.4	99.7	99.5
dev	87.5	84.1	84.1	81.0	99.7	99.4
test	86.6	83.3	83.5	80.5	99.7	99.5

Table 3: Parsing performance. Columns 2 and 3: unlabeled and labeled attachment scores of the (surface) dependency parser. Last four columns: unlabeled and labeled F-scores after classification of *il/se* clitics and conversion to deep syntax, applied either on the predicted surface dependency parses (columns 4 and 5) or on the gold dependency parses (last 2 columns). Results on the training set are obtained using a 10-fold jackknifing. Results on the dev and test set are obtained using training on the full training set.Punctuation tokens are taken into account.

The surface-to-deep syntax conversion module of Ribeyre et al. (2014) takes as input surface dependency trees in which a few linguistic phenomena have already been made explicit, because they were considered difficult to capture by a rule-based approach. This is in particular the case for the status of the *il* and *se* clitics, which results from complex syntactic and lexical factors. In order to do so, we designed two classifiers that predict the status of these clitics. We omit to describe here these classifiers as well as their evaluation, for reason of lack of space.

The architecture of our deep syntactic parser is to apply sequentially **(i)** part-of-speech tagging and lemmatization, **(ii)** surface dependency parsing and **(iii)** surface-to-deep syntax rewriting rules.

Tagging and syntactic parsing were performed with MACAON (Nasr et al., 2011), a tool suite for standard NLP tasks. The tagging is based on a CRF model whereas the dependency parser is a second order graph-based parser, with standard features. We report parsing performance in Table 3 (first two colums). The scores are comparable to the baseline scores obtained by the SPMRL shared task participants on French (Björkelund et al., 2013), without any special handling of multi-word expressions.

The last step consists in applying the surface-to-deep syntax conversion module (Ribeyre et al., 2014). This module uses OGRE (Ribeyre et al., 2012), a deterministic two-stage graph rewriting system.

The first stage follows the Single Pushout Approach (SPO) (Rozenberg, 1997), a widely used method when dealing with graph rewriting system. This stage identifies graph patterns and applies rewriting operations such as adding an edge, removing an edge, changing a label, and so on. This is done in one pass and contrary to the SPO approach, the first stage is executed only once.

The second stage is a propagation step. During the first stage, the rewriting rules may have left what we call *triggers* on edges. Those are special actions that, given a specific edge configuration, apply a serie of rewriting steps using a fixed-point algorithm: when all possible rewritings have been done, the algorithm terminates. It is especially helpful in case of linguistic phenomena interacting with each other. In the SSR of the sentence *John seems to want to give a book to Mary*, for example, *John* is the subject of *seems* and *want* is a dependent of *seems* and *give* a dependent of *want*. Ultimately, in the DSR, *John* is the subject of both *want* and *give*. The interaction between raising and control verbs is obtained through the propagation of rules of the form "if V_1 taking V_2 as complement has or gets a final subject X then add X as final subject of V_2". Moreover, the two-stage rewriting system ensures that the algorithm terminates and the system is confluent. See (Ribeyre, 2016) for more details and proofs.

The surface-to-deep syntax module applies sequentially 5 sets of rewriting rules:

1. The first set converts tense auxiliaries into mood and tense features on the lexical verb.
2. The second set distributes dependents of coordinated predicates and identifies the final subject of non finite verbs and by extension, of adjectives also, whether used as predicative complements or noun modifiers.
3. Syntactic alternations are mainly handled in the third set, which identifies the canonical grammatical functions for arguments of verbs (whether already present in the surface tree, or added by the second

	Prec.		Recall		F-measure		Prec.		Recall		F-measure	
Input	SSR	DSR	SSR	DSR	SSR	DSR	SSR	DSR	SSR	DSR	SSR	DSR
trigger detec.	89.3	89.4	88.7	88.7	89	89	88.6	88.8	88.4	88.3	88.5	88.6
frame selec.	81.1	81.2	80.6	80.6	80.8	80.9	80.3	80.5	80.2	80	80.3	80.2
role detec.	85.1	86.2	59.2	62.6	69.8	72.5	79.6	81.3	51.7	55.7	62.7	66.1
role selec.	77.9	80.9	54.2	58.7	**63.9**	**68.1**	72	75.9	46.7	52	**56.7**	**61.7**

Table 4: FastSem results for **all triggers**, using **gold** (left) and **predicted** (right) SSR and DSR.

module).

4. The fourth set handles comparative and superlative constructions mostly.
5. The last set exclusively deals with bypassing the semantically empty words.

We provide the performance evaluation of DSR prediction step in Table 3. Columns 4 and 5 show the result of the whole parsing architecture, where steps (i), (ii) and (iii) are predicted. The last two columns show the result of applying step (iii) on gold SSR. Not surprisingly, the DSR built from gold SSR are almost perfect. This is due to the fact that the deep syntactic corpus contains gold DSR for the small Sequoia part only, the other part, which corresponds to the FTB, is made of pseudo-gold DSR obtained by the application of step (iii) on gold SSR ! The table shows the sharp drop in quality when DSR are computed on predicted SSR.

5 Experiments and discussion

We now turn to FrameNet parsing experiments, meant primarily to compare the use of surface versus deep syntactic paths as features. All experiments were used using the same split.[13] Feature engineering was performed on the development set.

5.1 Evaluation metrics

The train, dev and test examples are made of the set of annotated frame occurrences of the train, dev and test sets, including the null frame cases. For each setting, we trained word specific classifiers for the frame selection step and frame specific classifiers for the role selection step. But, since selecting the null frame is a rather easy task, we chose to evaluate each of the two steps using two different metrics. For frame selection, we first evaluate the task of deciding whether a word triggers an actual frame or the null frame. The results are reported in lines "trigger detection" of Table 4. The "frame selection" lines report the precision, recall and F-scores of choosing a frame, computed when setting aside the triggers whose gold frame is the null frame.

For role labeling, prediction and evaluation is made on heads of role fillers only. It is also broken in two: we first evaluate the task of deciding whether a word plays a role or not with respect to the trigger (reported in the "role detection" lines in the result tables). Then, for words that are actually head of role fillers in gold data, we compute precision, recall and F-score of the head and role pairs predicted by our semantic parser (reported in the "role selection" lines in the tables). Note that in both cases, the role is counted as incorrect if the frame was not predicted correctly.

5.2 Results and discussion

The experiments conducted vary according to two dimensions: the use of surface vs. deep syntactic paths (SSP or DSP) and whether they are predicted or gold. The predicted SSP are obtained using predicted PoS, lemmas, morphological features and surface dependency syntax. The predicted DSP are obtained by applying *il/se* classification and rewriting rules on predicted surface dependency trees (cf. section 4). All results are computed on the test set.

The left part of Table 4 shows results using gold syntactic structure, whether surface or deep. As can be seen, the results for the first three metrics slightly increase when switching from SSP to DSP, but

[13]The training set is the concatenation of the usual training sets of the Sequoia and FTB corpus. Same for the development and test sets.

	Prec.		Recall		F-measure		Prec.		Recall		F-measure	
Input	SSR	DSR	SSR	DSR	SSR	DSR	SSR	DSR	SSR	DSR	SSR	DSR
frame selec.	80.1	80.7	80.1	80.7	80.1	80.7	80	80.5	80.8	80.9	80.4	80.7
role selec.	81.4	86.4	59.1	66.1	**68.5**	**74.9**	75.7	80.3	51.6	59	**61.3**	**68**

Table 5: FastSem results for **verbs**, using **gold** (left) and **predicted** (right) SSR and DSR.

	SSP	DSP				
		all	alt	byp	subj	coo
gold	68.5	74.3	70.6	69.1	69.3	70.2
predicted	61.3	68	63.3	63.1	62.4	63.1

Table 6: FastSem F-measure for role selection with application of deep rewriting rule sets in isolation, for verbal triggers. Rules are applied on SSP that are either gold (first row) or predicted (last row). The first column reports the results when using SSP. The second when using DSP with all rules applied. See text for description of the rule sets (alt) to (coo).

we obtain a 4.2 point improvement for the overall result of role selection when using DSP instead of SSP (63.9 to 68.1). Because our DSR focus on the predicate argument structure of verbs and adjectives, and because the number of adjectival triggers is marginal in the French FrameNet corpus, we chose to provide, in Table 5, the same metrics as in Table 4, computed on verbal triggers only. As one can see, using deep syntax provides substantial help for predicting roles: we obtain a 6.4 point improvement for role selection for verbal triggers (68.5 to 74.9).

We now turn to a more realistic setting in which all features for the semantic parser are predicted: lemmas, PoS, SSP and DSP. Not surprisingly, the results shown in Table 4 (all triggers) and 5 (verbal triggers) are overall lower than when using gold features. However, switching from surface to deep syntax leads to higher gain for predicted data than for gold data: 5.1 points (56.7 to 61.7) for all trigger, instead of 4.2 for gold data and 6.7 points (61.3 to 68) instead of 6.4 on gold data for verbal triggers. These results clearly show the benefit of using deep syntactic features.

The differences between SSP and DSP are of various kinds, as seen in section 2.2. We propose to study the impact of each phenomenon, by applying in isolation each set of graph-rewriting rules of the surface-to-deep syntax conversion module. More precisely, we applied in isolation (alt) the rules for syntactic alternations, (byp) the bypassing of empty prepositions and complementizers, (subj) the addition of subjects for non finite verbs and adjectives and (coo) the distribution of dependents to coordinated predicates. We provide the results in Table 6, for the role selection task, computed on verbal triggers only. It shows that every rule set contributes to a better prediction of the semantic structure.

5.3 Error analysis

In order to perform error analysis, we analyzed the changes in role selection when switching from SSP to DSP (table 7). The number of corrected errors (W→C) is more than four times the number of introduced errors (C→W). We reproduce below three cases of errors that were corrected when switching from surface to deep syntax. They correspond to syntactic alternation (1), coordination of VPs (2) and control verb (3). The trigger is in capital letters, and the (head of) role fillers we focus on are in bold:

	C→C	C→W	W→C	W→W
predicted	1163	47	218	481
gold	1362	48	203	316

Table 7: Improvements and degradations for role selection when switching from SSP to DSP, using either gold syntax (first row) or predicted syntax (second row). Break-down of the non-null gold roles of the dev set, when frames are correctly identified by both systems. C stands for correct, W stands for wrong.

freq. range	G1 (frequent) $> 10\%$		G2 (medium) $< 10\%$ and $> 1\%$		G3 (rare) $< 1\%$	
	Prop.	F1	Prop.	F1	Prop.	F1
SSR	42.1%	89.2	33.1%	78.3	24.8%	38.3
DSR	65.9%	92.3	14.3%	75	19.8%	23.5

Table 8: Role selection task results on the dev set, using gold frames triggered by verbs: break-down by frequency (in the training set) of the gold syntactic path. "Prop." columns provide the proportion of each sub-group.

1. *Cette **thérapie** a été DÉCIDÉE par le **gouvernement***
   (This therapy has been decided by the government.)
   thérapie: DSP=(+obj) SSP=(+subj) **gouvernement**: DSP=(+subj) SSP=(+p_obj,+obj.p)

2. ***Grandier** avait publié un pamphlet et S'OPPOSAIT fermement à la destruction des murailles.*
   Grandier had published a pamphlet and was firmly opposed to the destruction of the walls.
   Grandier: DSP=(+subj) SSP=(-dep.coord,-coord,+subj)

3. ***Ils** ont essayé de les PERSUADER de bouleverser le calendrier.*
   They have tried to them persuade to change the schedule.
   Ils: DSP=(+subj) SSP=(-obj.p, -de_obj, +subj)

We also took a closer look at the introduced errors. They mostly correspond to cases in which the role filler has same surface and deep syntactic path, the path being rather unusual for the role filler. This may indicate that increased regularity of the DSP makes role fillers with unusual syntactic path more difficult to detect. We tried to assess this hypothesis by breaking-down the performance of the role selection task by frequency of the syntactic paths between the head of the role filler and the trigger. Results are shown in table 8. The frequent paths (G1) lead to better role prediction than the other two groups, and this is even more true when using DSRs than SSRs (92.3 versus 89.2). This explains most of the improvements, since this group represents a higher proportion when using DSRs than SSRs (65.9 versus 42.1). For less frequent paths (G2 and G3), results are either slightly (G2) or much (G3) better when using SSRs than DSRs, but these two groups represent a much lower proportion in the DSR paths than in the SSR paths. To sum up, frequent paths are even more frequent when using DSRs, and thus lead to better role prediction, whereas the non frequent paths exhibit the opposite trend.

6 Conclusion

In this paper we showed that frame semantic structure prediction can benefit from a deeper syntactic representation, in which the syntactic paths between a verb and its arguments are normalized. This reduces the variety of the syntactic realization of semantic roles, which we assessed by measuring a decrease of the entropy of the syntactic paths of a given role. We then showed that a FrameNet semantic parser can take advantage of this simpler syntax/semantic interface and reach better performance when switching from surface syntax to deep syntax.

Acknowledgments

This work was funded by the French National Research Agency (ASFALDA project ANR-12-CORD-023), and supported by the French Investissements d'Avenir - Labex EFL program (ANR-10-LABX-0083).

References

Anne Abeillé and Nicolas Barrier. 2004. Enriching a french treebank. In *Proceedings of LREC 2004*, Lisbon, Portugal.

American Psychological Association. 1983. *Publications Manual*. American Psychological Association, Washington, DC.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *COLING-ACL '98: Proceedings of the Conference*, pages 86–90, Montreal, Canada.

Anders Björkelund, Ozlem Cetinoglu, Richárd Farkas, Thomas Mueller, and Wolfgang Seeker. 2013. (re)ranking meets morphosyntax: State-of-the-art results from the SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 135–145, October.

A. Burchardt, K. Erk, A. Frank, A. Kowalski, S. Padó, and M. Pinkal. 2006. The salsa corpus: a german corpus resource for lexical semantics. In *Proceedings of LREC 2006*.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage pcfg-based lfg approximations. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marie Candito and Djamé Seddah. 2012a. Effectively long-distance dependencies in French: annotation and parsing evaluation. In *Proceedings of TLT 11*.

Marie Candito and Djamé Seddah. 2012b. Le corpus sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical. In *Proceedings of TALN 2012 (in French)*, Grenoble, France, June.

Marie Candito, Pascal Amsili, Lucie Barque, Farah Benamara, Gaël de Chalendar, Marianne Djemaa, Pauline Haas, Richard Huyghe, Yvette Yannick Mathieu, Philippe Muller, Benoît Sagot, and Laure Vieu. 2014a. Developing a French FrameNet: Methodology and first results. In *Proceedings of LREC 2014*, Reykjavik, Iceland, May.

Marie Candito, Guy Perrier, Bruno Guillaume, Corentin Ribeyre, Karën Fort, Djamé Seddah, and Éric De La Clergerie. 2014b. Deep Syntax Annotation of the Sequoia French Treebank. In *Proceedings of LREC 2014*, Reykjavik, Islande, May.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Stanford University.

Marianne Djemaa, Marie Candito, Philippe Muller, and Laure Vieu. 2016. Corpus annotation within the french framenet: methodology and results. In *Proceedings of LREC 2016*, Portoroz, Slovenia, May.

Marianne Djemaa. 2014. Traitement framenet des constructions à attribut de l'objet. In *Proceedings of the 16e Rencontres des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RECITAL 2014)*, pages 13–24, Marseille, France, July.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Daniel Flickinger, Yi Zhang, and Valia Kordoni. 2012. Deepbank: A dynamically annotated treebank of the wall street journal. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories (TLT-11)*, pages 85–96, Lisbon, Portugal. Edições Colibri, Lisbon.

Association for Computing Machinery. 1983. In *Computing Reviews*, volume 24, pages 503–512.

Daniel Gildea and Julia Hockenmaier. 2003. Identifying semantic roles using combinatory categorial grammar. In Michael Collins and Mark Steedman, editors, *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 57–64.

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

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing prague czech-english dependency treebank 2.0. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 3153–3160, İstanbul, Turkey. ELRA, European Language Resources Association.

Jan Hajič, Jarmila Panevová, Eva Hajicová, Petr Sgall, Petr Pajas, Jan Štepánek, Jiří Havelka, Marie Mikulová, Zdenek Zabokrtskỳ, and Magda Ševcıková Razımová. 2006. Prague dependency treebank 2.0. *CD-ROM, Linguistic Data Consortium, LDC Catalog No.: LDC2006T01, Philadelphia*, 98.

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank. *Comput. Linguist.*, 33(3):355–396, September.

Richard Johansson and Pierre Nugues. 2007. Lth: Semantic structure extraction using nonprojective dependency trees. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 227–230, Stroudsburg, PA, USA. Association for Computational Linguistics.

Aravind K Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In *Handbook of formal languages*, pages 69–123. Springer.

Igor Mel'čuk. 1988. *Dependency syntax: theory and practice*. SUNY press.

Alexis Nasr, Frédéric Béchet, and Jean-François Rey. 2010. Macaon : Une chaîne linguistique pour le traitement de graphes de mots. In *Traitement Automatique des Langues Naturelles - session de démonstrations*, Montréal.

Alexis Nasr, Frederic Bechet, Jean-Francois Rey, Benoit Favre, and Joseph Le Roux. 2011. Macaon: An nlp tool suite for processing word lattices. In *The 49th Annual Meeting of the Association for Computational Linguistics: demonstration session*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, March.

Corentin Ribeyre, Djamé Seddah, and Éric Villemonte De La Clergerie. 2012. A Linguistically-motivated 2-stage Tree to Graph Transformation. In *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms*, Paris, France.

Corentin Ribeyre, Marie Candito, and Djamé Seddah. 2014. Semi-Automatic Deep Syntactic Annotations of the French Treebank. In *Proceedings of the 13th International Workshop on Treebanks and Linguistic Theories*, Tübingen, Germany, December.

Corentin Ribeyre, Eric Villemonte de la Clergerie, and Djamé Seddah. 2016. Accurate deep syntactic parsing of graphs: The case of french. In *Proceedings of LREC 2016*, may.

Corentin Ribeyre. 2016. *Data-driven methods for syntax-semantic interface*. Theses, Université Paris Diderot, January.

Grzegorz Rozenberg, editor. 1997. *Handbook of Graph Grammars and Computing by Graph Transformation: Volume I. Foundations*. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Natalie Schluter and Josef van Genabith. 2008. Treebank-based acquisition of lfg parsing resources for french. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA, October. Association for Computational Linguistics.

David Vickrey and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of ACL-08: HLT*, pages 344–352, Columbus, Ohio, June. Association for Computational Linguistics.

Language Independent Dependency to Constituent Tree Conversion

Young-Suk Lee
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
USA
ysuklee@us.ibm.com

Zhiguo Wang
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
USA
zhigwang@us.ibm.com

Abstract

We present a dependency to constituent tree conversion technique that aims to improve constituent parsing accuracies by leveraging dependency treebanks available in a wide variety in many languages. The technique works in two steps. First, a partial constituent tree is derived from a dependency tree with a *very simple* deterministic algorithm that is both language and dependency type independent. Second, a complete high accuracy constituent tree is derived with a constraint-based parser, which uses the partial constituent tree as external constraints. Evaluated on Section 22 of the WSJ Treebank, the technique achieves the state-of-the-art conversion F-score 95.6. When applied to English Universal Dependency treebank and German CoNLL2006 treebank, the converted treebanks added to the human-annotated constituent parser training corpus improve parsing F-scores significantly for both languages.

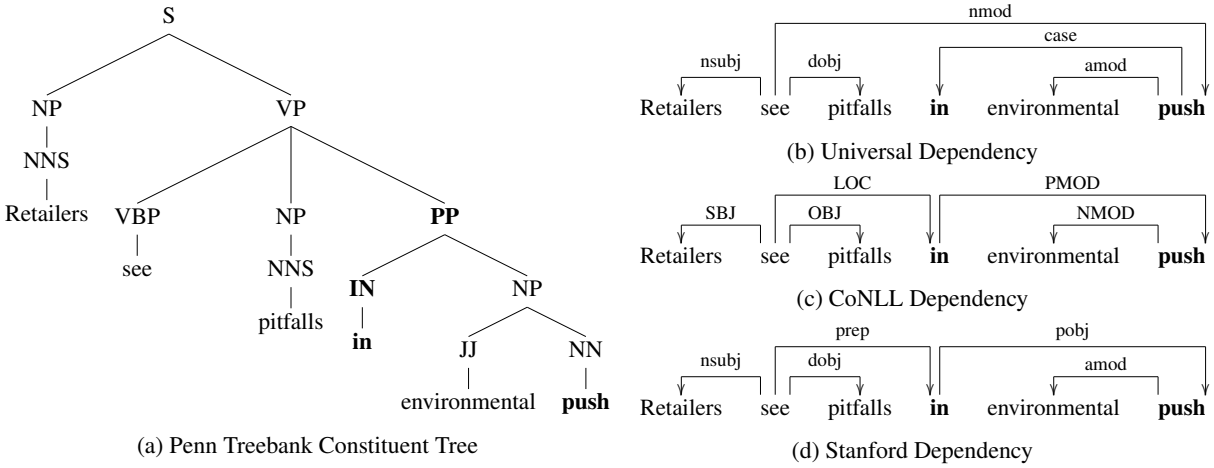
1 Introduction

State-of-the-art parsers require human annotation of a training corpus in a specific representation, e.g. constituent structure in Penn Treebank (Charniak and Johnson, 2005; Petrov and Klein, 2007) or dependency relations in a dependency treebank (Yamada and Matsumoto, 2003; McDonald et al., 2005) . Creation of human-annotated treebanks, however, is knowledge and labor intensive and it is desired that one can improve parsing performance by leveraging treebanks annotated in representations of a wide variety.

While there have been quite a few papers on automatic conversion from dependency to constituent trees and vice versa (Wang et al., 1994; Collins et al., 1999; Forst, 2003; de Marneffe et al., 2006; Johansson and Nugues, 2007; Xia et al., 2008; Hall and Nivre, 2008; Rambow, 2010; Wang and Zong, 2010; Zhang et al., 2013; Simkó et al., 2014; Kong et al., 2015) , very few papers address the issue of whether or not the converted treebank actually improves the performance of the target parser when added to the human-annotated gold treebanks for parser training. In addition, much of the work on dependency to constituency conversion relies on dependency trees automatically derived from the Penn Treebank (Marcus et al., 1993) via head rules and assumes that the head-modifier definitions are consistent between the constituent and dependency trees (Xia et al., 2008). However, such techniques cannot easily generalize to dependencies that diverge from the Penn Treebank in head-modifier definitions and dependency labels, e.g. Universal Dependency (Nivre et al., 2015) in Figure 1(b), and the dependencies of a wide variety available in CoNLL shared tasks.

In this paper, we propose a *very simple* dependency to constituent tree conversion technique which is applicable to any languages and any dependencies, e.g. Universal Dependency (UD), CoNLL dependencies (CoNLL), Stanford dependencies (Stanford), while achieving the state-of-the-art conversion accuracy. The technique works in two steps. We first derive a partial constituent tree from a dependency tree according to a simple deterministic algorithm without any external knowledge sources such as head rules. The partial constituent tree retains the gold part-of-speech tags (POStags) and partial contituent brackets inferred from the dependency tree (in Section 2). We then recover the complete constituent

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 421–428, Osaka, Japan, December 11-17 2016.



(a) Penn Treebank Constituent Tree

(b) Universal Dependency

(c) CoNLL Dependency

(d) Stanford Dependency

Figure 1: Penn Treebank Constituent Tree (a), Universal Dependency (b), CoNLL Dependency (c) and Stanford Dependency (d) representations for *Retailers see pitfalls in environmental push*. The preposition *in* is the head of *push* in the Penn Treebank, CoNLL and Stanford Dependency, whereas it is the modifier of *push* in the Universal Dependency. None of the dependency labels overlap with the Penn Treebank phrase labels. A dependency arrow goes from a head to its modifier.

structure and labels by a constraint-based parsing which uses the gold POStags and partial brackets as parsing constraints (in Section 3).

Evaluated on WSJ-22 for conversion accuracy, the proposed technique achieves the labeled F-score of 95.62 for conversion from the Stanford (de Marneffe et al., 2006) basic dependency (in Section 4). When applied to the English Universal Dependency (UD) treebank and German CoNLL2006 treebank, the converted treebanks added to the human-annotated constituent parser training corpus improve the F-scores of BerkeleyParser [1] (Petrov and Klein, 2007) and Maximum Entropy (MaxEnt) parsers significantly for both languages (in Section 5). While most of the previous work applies dependency to constituent tree conversion on the dependencies automatically derived from the Penn Treebank, the current work applies the technique to human-annotated English UD treebank as well. The constituent parser performance improvement due to the addition of converted treebanks is the first reported for English and German (in Section 6).

Throughout the paper, we use the notation CTree for a constituent tree, DTree for a dependency tree and UDTree for a universal dependency tree. We use the term 'constituent' and 'phrase' interchangeably. Conversion and parsing accuracies are reported in labeled F-scores.

2 Dependency to Partial Constituent Tree Conversion

We first derive a partial constituent tree from the source dependency tree. The partial constituent tree retains all of the human annotated part-of-speech tags and partial constituent brackets inferred from the source dependencies. Figure 2 is the deterministic algorithm that derives a partial CTree from any given DTree, where the dependency span of a word is a consecutive word sequence reachable from the word by head modifier relations.

Note that the algorithm in Figure 2 does not require any external knowledge sources such as head rules learned from the target CTrees. It applies to any DTrees that make a reasonable linguistic assumption on head-modifier relations regardless of languages and dependency types. This *simplicity* sets the current proposal apart from all of the previous proposals that rely on linguistic rules, as in (Xia et al., 2008), statistical model utilizing manually acquired head rules and the phrase labels of the target constituent treebank, as in (Kong et al., 2015), or a scoring function that computes the similarity between the source DTree and the nbest parsing output of the DTree sentences by the target constituent parser, as in (Niu et al., 2009).

[1] https://github.com/slavpetrov/berkeleyparser

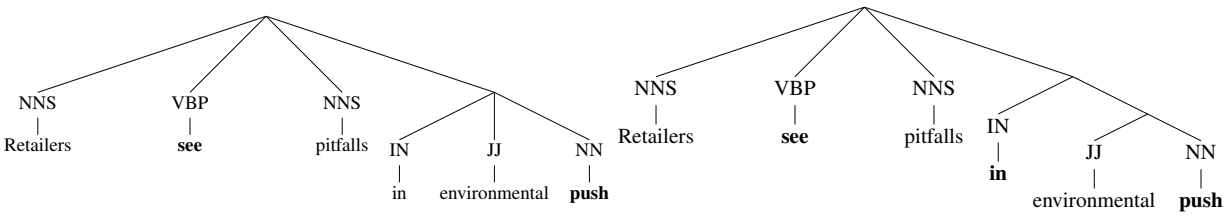
input: DTree (labeled or unlabeled) with *n* input words
output: Unlabeled CTree with gold POStags and partial constituent brackets

Step 1: Identify the dependency span D_i of each word w_i
if *the word* w_i *does not have any dependent* **then**
  D_i is length 1, containing only w_i itself;
else
  D_i subsumes all of its dependents recursively;

Step 2: Convert a dependency span D_i to a constituent C_i
Vertex of C_i dominates the immediate dependents of the head word and the head word itself.

Step 3: Remove all constituent brackets containing only one word.

Figure 2: DTree to unlabeled partial CTree Conversion Algorithm



(a) Partial CTree derived from UDTree in Figure 1(b)

(b) Partial CTree derived from DTrees in Figure 1(c, d)

Figure 3: Partial CTrees derived from the DTrees in Figure 1 according to the algorithm in Figure 2

The UDTree in Figure 1(b) is converted to the partial CTree in Figure 3(a) and the DTrees in Figure 1(c, d) are converted to the partial CTree in Figure 3(b) according to the algorithm in Figure 2. The head word of each constituent is in bold-face. Similarity of the head-modifier definitions between the target CTree and the source DTree is reflected on the partial CTrees. The partial CTree in Figure 3(a) derived from the UDTree leaves more ambiguity within the prepositional phrase covered by *in environmental push* than the one derived from CoNLL or Stanford DTrees. Similarity between a given DTree representation and the Penn Treebank CTree is reflected on the conversion accuracy reported in Section 4.

3 Constraint-based Maximum Entropy Parsing

To derive the fully specified labeled CTree from a partial CTree, we parse the input sentence with a constraint-based constituent parser that utilizes the gold POStags and partial brackets as model external constraints.

We implement the constraint-based parsing algorithm on the maximum entropy parser of (Ratnaparkhi, 1997; Ratnaparkhi, 1999), which works robustly regardless of the grammar coverage of the baseline parsing model and therefore well-suited for constraint-based parsing of partial CTrees derived from out-of-domain as well as in-domain DTrees.

3.1 Baseline Maximum Entropy Parser

The baseline MaxEnt parser takes one of the four actions to parse an input sentence: *tag* , *chunk* , *extend* and *reduce*. Four models corresponding to each action are built separately during training.

The model score in (1) is integrated into the parser scoring function (2). In (1) and (2), a_i is an action from *tag*, *chunk*, *extend* or *reduce*, and b_i is the context for a_i.

$$q(a_i|b_i) = p_{a_i}(a_i|b_i) \quad (1)$$

$$score(T) = \prod_{a_i \in deriv(T)} q(a_i|b_i) \quad (2)$$

deriv(T) in (2) is the derivation of a parse *T*, which may not be complete. Given the scoring function (2), a beam search heuristic attempts to find the best parse T^*, defined in (3) where *trees(S)* are all the complete parses for an input sentence *S*.

input: Input sentence with partial CTree
output: Complete labeled CTree

Parser Initialization;
M = 20 & K = 80 & Q = 0.95;
C = ¡empty heap¿ h_0 = ¡input sentence¿;
while —C— ¡ M **do**
  if *($\forall i$, h_i is empty)* **then**
    break
  else
    i = max $\{i$ — h_i is non-empty$\}$;
    sz = min $(K, |h_i|)$;
    for j = *1 to sz* **do**
      if $\exists$ h_c **then**
        d_c=advance(extract (h_c))
      else
        $d_1 d_p$ = advance(extract (h_i), Q)
      for q = *1 to p* **do**
        if *completed* (d_q) **then**
          insert (d_q, C)
        else
          insert (d_q, h_{i+1})

Figure 4: Constraint-based parsing algorithm

Constraints	WSJ-22	BOLT-DF
Baseline w/o constraints	88.57	82.43
Gold POStag	89.50	85.09
Gold bracket	98.52	96.88
Gold POStag+bracket	98.74	98.02

Table 1: Impact of model external constraints on parsing F-scores. The constraints Gold POStag, Gold bracket denote the POStags and constituent brackets read off from the human annotated gold CTrees. Combination of gold brackets and gold labels are equivalent to gold CTrees.

$$T^* = arg \max_{T \in trees(S)} score(T) \tag{3}$$

The parser explores the top K scoring parses and terminates when M complete parses are found or all hypotheses are exhausted. Possible actions $a_1 a_n$ on a derivation are sorted according to the model score $q(a_i|b_i)$. Only the actions $a_1 a_m$ with the highest probabilities are considered.

3.2 Constraint-based Maximum Entropy Parsing

In constraint-based parsing, the parser actions are based not only on the trained model scores but also on external constraints, which aim to improve the parsing qualities not achievable by parsing models alone.

The model external constraints include gold (i.e. human annotated) POStags, gold constituent brackets and/or gold labels. We enforce the parser to choose the gold tags, gold constituent brackets and labels over those selections made by the parsing model scores. When gold tags are provided as constraints, the tag action accepts the gold tag as the output. When gold constituent brackets (and labels) are given, the parser $chunk$, $extend$ and $reduce$ actions accept the gold constituent spans and their labels over the highest scoring model hypotheses. Figure 4 shows the constraint-based parsing algorithm.

The parameters M, K are described in Section 3.1. C denotes the heap of completed parses. h_i contains the derivations of length i. h_c contains the derivation with a constraint. Q is the probability pruning threshold. Advance applies relevant actions to a derivation d and returns a list of new derivations $d_1 d_n$. If there is a model external constraint for an action, it returns the derivation with the constraint d_c. Otherwise, it returns the derivations with the highest probabilities until the probability mass of the actions is greater than the threshold Q. Insert inserts a derivation d in heap h. Extract returns a derivation in h. Completed returns true if and only if d is a complete derivation.

Applying the constraint-based parsing algorithm in Figure 4 to the input sentence *Retailers see pitfalls in environmental push* with the partial CTrees in Figure 3 as the constraints, the parser produces the labeled CTree in Figure 1(a). Impact of model external constraints on parsing F-scores is shown in Table 1. The constraints Gold POStag, Gold bracket denote the POStags and constituent brackets read off from the human annotated gold CTrees. Combination of gold brackets and gold labels are equivalent to gold CTrees. Note that gold constituent brackets alone lead to very high F-scores for WSJ-22, 98.52 and BOLT-DF, 96.88. Our proposal capitalizes on the effectiveness of human annotated gold POStags

Techniques	Dependencies	F-score
(Xia et al., 2008)	CoNLL	89.4
(Niu et al., 2009)	Unlabeled	93.8
Current 2-stage	CoNLL	95.5
Current 2-stage	Stanford-v1.6.8	95.6

Table 2: DTree to CTree conversion F-scores on WSJ-22

Dependencies	DevSet	EvalSet
Stanford-v1.6.8	92.88	92.06
CoNLL	92.50	91.74
Universal Dependency	91.22	90.48

Table 3: DTree to CTree conversion F-scores on EWT according to various dependencies

and constituent brackets on parsing even when they are provided only partially, and utilize the partial CTrees derived from human annotated DTrees to recover the complete CTrees.

4 Conversion Accuracy

To compare the performance of the current conversion technique (Current 2-stage) with the previous work, all of which use the DTrees automatically derived from the Penn Treebank as the source dependency, we show the conversion accuracy on WSJ-22 in Table 2. The proposed 2-stage technique achieves the state-of-the-art conversion F-score 95.6 without relying on language and/or target treebank specific head rules. The constraint-based MaxEnt parser is trained on WSJ02-21.[2]

We also show the conversion accuracy of the current technique on English Web Treebank (EWT, LDC2012T13) from three types of dependencies in Table 3: Stanford basic dependency converted from the Penn Treebank by Stanford parser v1.6.8, CoNLL dependency converted from the Penn Treebank by pennconverter.jar[3], and human-annotated UD of (Nivre et al., 2015)[4]. MaxEnt parser for the constraint-based parsing is trained on English Ontonotes-5 treebank. The EWT train/development/evaluation data partitions are the same as those available from the UD.[5] Conversion F-scores are computed with `evalb`, excluding punctuations.

5 Parsing Experimental Results

Our ultimate goal is to improve constituent parsing accuracy by leveraging dependency treebanks available in a wide variety. To achieve this objective, we first convert dependency treebanks into constituent representations using the proposed conversion technique. Then we merge the converted treebanks with the human-annotated constituent treebank to enlarge the training set of constituent parser. We finally re-train the constituent parsers with the enlarged training set. We report the parsing experimental results for English and German.

English parser training and evaluation data sets from Ontonotes-5 (LDC2013T19) and EWT are shown in Table 4. Ontonotes-5 is the biggest constituent treebank available in English and includes sub-corpora from 7 genres. German parser training and evaluation data sets are shown in Table 5.

We experiment with two constituent parsers. The MaxEnt parser which we adapted for the constraint-based parsing and the BerkeleyParser. We measure the labeled F-scores including punctuations so that all sentences are scored correctly even when there is a mismatch of punctuation tags between the reference and machine parses.[6]

5.1 English Results

We train the baseline parser on the Ontonotes-5 training corpus only (Baseline in Tables 6 and 7). UD treebank corresponding to the training portion of EWT is converted to CTrees, using the proposed conversion technique with the constraint-based MaxEnt parser, and the converted treebank is added to the Ontotnotes-5 treebank for parser training (+Converted in Tables 6 and 7). We also train parsers on both Ontonotes-5 treebank and the EWT training corpus (+Gold in Tables 6 and 7).

[2] (Niu et al., 2009) automatically derive their dependencies from the Penn Trees using head percolation table.

[3] Downloaded from http://nlp.cs.lth.se/coftware/treebank-converter

[4] v1.1 downloaded from http://universaldependencies.org

[5] The gold stanford English UD was built over the source material of the EWT. That is, UD and EWT are parallel.

[6] Ontonotes-5 and EWT are quite noisy and quite a few sentences contain punctuation tag mismatches.

	Training Data		Development Data		Evaluation Data	
Genre	sent #	token #	sent #	token #	sent #	token #
WB	~14k	~323k	800	~18k	800	~18k
MZ	~6.7k	~159k	800	~19k	800	~19k
NW	~42k	~1m	800	~20k	800	~19k
BN	~12k	~229k	800	~15k	800	~15k
BC	~15k	~221k	800	~12k	800	~12k
TC	~14k	~109k	800	~6.2k	800	~6k
PT	~24k	~326k	800	~11k	800	~11k
EWT	~125k	~205k	2,002	~25k	2,077	~25k

Table 4: English Ontonotes (WB, MZ, NW, BN, BC, TC, PT) and English Web Treebank (EWT) data partition into baseline parser train (Ontonotes), converted train (EWT), development and evaluation data sets

Data Sets	sent #	token #
Baseline train	~18.5k	~332k
Converted train	~18.5k	~328k
Development	1,061	~18.5k
Evaluation	1,060	~18k

Table 5: German Tiger Treebank data partition into baseline parser train, converted train, development and evaluation data sets

Eval Set	Baseline	+Converted	+Gold
EWT	79.30	80.78	82.22
WB	82.94	83.50	83.67
MZ	85.01	85.64	85.96
NW	86.82	87.01	87.35
BN	87.25	87.31	87.43
BC	82.21	*81.72*	*82.14*
TC	81.45	*81.41*	*81.23*
PT	94.72	94.78	95.11

Table 6: English MaxEnt parser F-scores

Eval Set	Baseline	+Converted	+Gold
EWT	78.34	79.12	80.21
WB	83.48	*82.94*	*83.15*
MZ	86.10	86.48	86.35
NW	86.17	86.46	86.64
BN	85.49	85.73	86.31
BC	81.67	*81.34*	*81.64*
TC	77.40	*76.65*	*76.12*
PT	91.92	*91.79*	*91.91*

Table 7: English BerkeleyParser F-scores

For both MaxEnt and Berkeley parsers, addition of the converted treebank improves the F-scores of the EWT evaluation data much more than other evaluation data sets from the Ontnotes-5 treebank, as expected. The converted treebank also improves the F-scores of WB, MZ, NW, BN and PT for the MaxEnt parser and MZ, NW and BN for BerkeleyParser. Not surprisingly, addition of the gold EWT improves the parser performance more than addition of the converted treebank. When the addition of the converted treebank hurts the parser performance, we see that the same downward pattern holds even with the addition of the gold EWT, as indicated by italics in Tables 6 and 7.

5.2 German Results

Tiger constituent treebank has the corresponding CoNLL2006 dependency treebank. We split the Tiger treebank training data into two parts, one for the baseline constituent parser training, and the other for conversion from the CoNLL dependency treebank. Experimental results are shown in Table 8. We observe the same pattern of improvement as English in a bigger margin.

6 Related Work and Conclusions

In the family of DTree to CTree conversion technique, the current work is closest in spirit to (Niu et al., 2009). They generate N-best parses of the dependency treebank sentences using the constituent parser and compare the similarity between N-best constituent parses and the source dependencies by converting the N-best parses back to dependencies. They show that addition of converted Chinese dependency treebank to CTB, (Xue et al., 2005), improves the Chinese constituent parsing accuracy modestly. (Xia

training data / parser	Baseline	+ Converted Treebank	+ Gold Treebank
MaxEnt parser	75.74	76.88	78.01
BerkeleyParser	71.88	73.42	76.12

Table 8: Contituent parsing improvement due to the DTree-to-CTree converted treebank and the gold constituent treebank

et al., 2008) propose a rule-based DTree to CTree conversion technique, assuming that the input DTree is identical to a flattened version of the desired CTree. They decompose the input DTree into multiple DTree segments, replacing each segment with the CTree counterparts and glue the CTree segments to form a complete CTree. The idea of utilizing dependency boundaries as constraints on constituent parsing has been explored in (Wang and Zong, 2010).

In the family of bi-directional conversion between CTrees and DTrees, (Hall and Nivre, 2008) present a dependency driven parser that parses both dependency and constituent structures. They automatically transform constituent representations into complex dependency representations so that they can recover the constituent structure. (Kong et al., 2015) propose a statistical model to transform DTrees into CTrees. They first convert CTrees to DTrees, which encode the rich head-modifier and phrase label information from the CTrees.[7] They train a statistical model to restore the CTrees from the feature-rich DTrees. While they report their DTree to CTree conversion accuracy on WSJ-22, their accuracy is not directly comparable to those we report in Tables 2 and 3 since their DTrees encodes head-modifier relations and phrase labels read off from the corresponding gold CTrees. (Fernández-Golzález and Martins, 2015) derive head-ordered DTrees from CTrees, train an off-the-shelf dependency parser on the DTrees, and recover the constituent information from the head-ordered DTrees. These bi-directional techniques practically reduce constituent parsing to dependency parsing and are applied to DTrees that encode the same complex information as the corresponding CTrees in order to easily recover the phrase structures.

We presented a simple DTree to CTree conversion technique that aims to improve constituent parsing accuracies by leveraging dependency treebanks available in a wide variety in many languages. Evaluated on WSJ-22, the technique achieves the state-of-the-art conversion F-score 95.6. When applied to English and German, the converted treebanks added to the constituent parser training corpus improve parsing F-scores significantly for both languages.

Acknowledgements

We would like to acknowledge the anonymous reviewers for their helpful comments.

References

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

Michael Collins, Lance Ramshaw, Jan Hajic, and Christoph Tillmann. 1999. A Statistical Parser for Czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 505–512.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*.

Daniel Fernández-Golzález and André F. T. Martins. 2015. Parsing as Reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Martin Forst. 2003. Treebank Conversion - Establishing a Testsuite for a Broad-Coverage LFG from the TIGER Treebank. In *Proceedings of LING at EACL*, pages 25–32.

Johan Hall and Joakim Nivre. 2008. A dependency-driven parser for German dependency and constituency representations. In *Proceedings of the Workshop on Parsing German*, pages 47–54.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, pages 105–112, Tartue, Estonia.

Lingpeng Kong, Alexander M. Rush, and Noah A. Smith. 2015. Transforming Dependencies into Phrase Structures. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the penn treebank. *Computational Linguistics - Special issue on using large corpora: II, Volume 19 Issue 2, June 1993*, pages 313–330.

[7] https://github.com/ikekonglp/PAD/tree/master/python

Ryan McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP) 2005*.

Zheng-Yu Niu, Haifeng Wang, and Hua Wu. 2009. Exploiting Heterogeneous Treebanks for Parsing. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing*, pages 46–54.

Joakim Nivre, Cristina Bosco, Jinho Choi, MarieCatherine de Marneffe, Timothy Dozat, Richard Farkas, Jennifer Foster, Filip Ginter, Yoav Goldberg, Jan Hajic, Jenna Kanerva, Veronika Laippala, Alessandro Lenci, Teresa Lynn, Christopher Manning, Ryan McDonald, Anna Missila, Simonetta Montemagni, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Maria Simi, Aaron Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. 2015. Universal Dependencies 1.0.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Proceedings of NAACL-HLT*, pages 404–411.

Owen Rambow. 2010. The Simple Truth about Dependency and Phrase Structure Representations: An Opinion Piece. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 337–340.

Adwait Ratnaparkhi. 1997. A Linear Observed Time Statistical Parser Based on Maximum Entropy Models. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Adwait Ratnaparkhi. 1999. Learning to Parse Natural Language with Maximum Entropy Models. *Machine Learning 34*, pages 151–175.

Katalin Ilona Simkó, Veronika Vencze, Zsolt Szánto, and Richárd Farkas. 2014. An Emripical Evaluation of Automatic Conversion from Constituency to Dependency in Hugarian. In *Proceedings of the 25th COLING*, pages 1392–1401.

Zhiguo Wang and Chengqing Zong. 2010. Phrase structure parsing with dependency structure. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1292–1300. Association for Computational Linguistics.

Jong-Nae Wang, Jing-Shin Chang, and Keh-Yih Su. 1994. An Automatic Treebank Conversion Algorithm for Corpus Sharing. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 248–254.

Fei Xia, Rajesh Bhatt, Owen Rambow, Martha Palmer, and Dipti Misra Sharma. 2008. Towards a Multi-Representational Treebank. In *Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories*, pages 159–170.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering, 11(2)*, pages 207–238.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of International Workshop on Parsing Technologies, Volume 3*.

Yuan Zhang, Regina Barzilay, and Amir Globerson. 2013. Transfer Learning for Constituency-Based Grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 291–301.

Promoting multiword expressions in A* TAG parsing

Jakub Waszczuk and **Agata Savary**
Université François-Rabelais Tours,
3 place Jean-Jaurès,
41000 Blois, France
first.last@univ-tours.fr

Yannick Parmentier
LIFO - Université d'Orléans,
6, rue Léonard de Vinci,
45067 Orléans, France
first.last@univ-orleans.fr

Abstract

Multiword expressions (MWEs) are pervasive in natural languages and often have both idiomatic and compositional readings, which leads to high syntactic ambiguity. We show that for some MWE types idiomatic readings are usually the correct ones. We propose a heuristic for an A* parser for Tree Adjoining Grammars which benefits from this knowledge by promoting MWE-oriented analyses. This strategy leads to a substantial reduction in the parsing search space in case of true positive MWE occurrences, while avoiding parsing failures in case of false positives.

1 Introduction

Multiword expressions (MWEs), e.g. *by and large*, *red tape*, and *to pull one's socks up* 'to correct one's work or behavior', are linguistic objects containing two or more words and showing idiosyncratic behavior at different levels. Notably, their meaning is often not deducible from the meanings of their components and from their syntactic structure in a fully compositional way. Thus, interpretation-oriented NLP tasks, such as semantic calculus or translation, call for MWE-dedicated procedures. Syntactic parsing often underlies such tasks, and the crucial issue is at which point the MWE identification should take place: before (Nivre and Nilsson, 2004), after (Constant et al., 2012) or during parsing (Wehrli et al., 2010; Green et al., 2013; Candito and Constant, 2014; Nasr et al., 2015; Constant and Nivre, 2016). The last, joint, approach proves the most efficient due to at least two reasons. Firstly, some MWEs coincide with word combinations that cross phrase boundaries, which is hard to detect prior to parsing, as in example (1). Secondly, while most MWEs have both an idiomatic and a compositional reading, as in examples (2)–(3), the former occurs much more frequently than the latter for large classes of MWEs. In Sec. 6 we show that, indeed, the *idiomaticity rate*, i.e. the ratio of occurrences with idiomatic reading to all occurrences in a corpus, exceeds 0.95 for verbal MWEs and compounds. This suggests that promoting MWE-oriented analyses in parsing might lead to rapidly achieved correct parses. (Wehrli, 2014) shows that, indeed, the quality of symbolic parsing significantly increases if an occurrence of a MWE is admitted as soon as the necessary syntactic constraints are fulfilled. Our goal is to apply a similar strategy, i.e. to systematically promote MWE-oriented interpretations, while parsing with Tree Adjoining Grammars (TAGs).

(1) **After all** the preparations we finally left.
(2) After being criticized, she **pulled her socks up**.
(3) When the kid shivered with cold, **she pulled its socks up**.
(4) **Acid rains** in Ghana are equally grim.

Consider the sentence in example (4). At least two competing analyses are syntactically valid for the first 4 words: *rains* is (a) a verb with the subject *acid*, or (b) the head noun of a nominal phrase. In the latter case, the nominal phrase has either (i) a compositional reading (*acid* is a regular nominal modifier) or (ii) an idiomatic one (*acids rains* is an NN compound).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 429–439, Osaka, Japan, December 11-17 2016.

Our objective is to propose a parsing strategy which would promote analysis (b) and reading (ii). More precisely, the parser should only provide grammar-compliant MWE-oriented analyses each time they are feasible. Thus, we wish to both avoid the parsing failure for (1), and rapidly achieve the correct syntactic parses of (2)–(4), due to imposing their idiomatic interpretations. In this way, the parser's search space is reduced, with virtually no loss of correct parses, and with rare errors at the level of MWE identification, as in (3). The rate of such errors is the complement of the idiomaticity rate of the text to be parsed (here: 0.05).

Note that promoting the most probably correct analysis, whether containing MWEs or not, is the goal of probabilistic parsers in general. Thus, instead of designing a custom parsing architecture for promoting MWEs, it would be more adequate to simply train a general-purpose parser on a treebank containing MWE annotations. This solution is however hindered by data insufficiency. Firstly, many languages still lack large-size treebanks. Secondly, very few treebanks contain a full-fledged range of MWE annotations, even for English (Rosén et al., 2015). Thirdly, MWEs are subject to sparseness problems even more than single words: most existing MWEs occur never or rarely in MWE-annotated corpora (Czerepowicka and Savary, 2015), let alone treebanks. Here, we partly cope with these problems by an Earley-style A* parser using a MWE-oriented heuristic, which takes advantage of a potential occurrence of MWEs in a sentence. While it is designed to systematically promote MWEs regardless of their probabilities, the parser could be very well used with a weighted TAG and the weights assigned to individual elementary trees could be estimated on the basis of training data.

In Sec. 2 we remind basic facts about TAGs. In Sec. 3 we explain the MWE-promoting strategy in TAG parsing. In Sec. 4 we describe the parsing algorithm on a running example and we formalize its heuristics in Sec. 5. In Sec. 6 we show experimental results on a Polish TAG grammar extracted from a treebank. The choice of Polish is due to the fact that high-quality MWE resources compatible with the treebank are available for this language. In Sec. 7 we compare our approach with related work. Finally, we conclude and comment on future work.

2 Tree Adjoining Grammars

A TAG (Joshi et al., 1975) is a tree-rewriting system defined as a tuple $\langle \Sigma, N, I, A, S \rangle$, where Σ (resp. N) is a set of terminal (resp. non-terminal) symbols, I and A are sets of elementary trees (ETs), and $S \in N$ is the axiom. Trees in I are called initial trees (ITs), their internal and leaf nodes are labeled with symbols in N and in $\Sigma \cup N$, respectively. Their non-terminal leaf nodes are called substitution nodes and marked with $\downarrow$. Trees in A are called auxiliary trees (ATs) and are similar to trees in I except that they contain a leaf node (called a foot and marked with $\star$) whose label is the same as the one of the root. Consider the toy TAG in Fig. 2 covering three competing interpretations for *acid rains* in example (4). Notably, tree t_5 represents its idiomatic reading. We have $I = \{t_1, t_3, t_4, t_5, t_6\}$ and $A = \{t_2\}$.

ETs are combined to derive new trees using *substitution* and *adjunction*. Substitution consists in replacing a leaf with an ET whose root is labeled with the same non-terminal (cf. the dotted arrow in Fig. 1). Adjunction consists in inserting an AT t inside any tree t' provided that the root/foot label of t is the same as the label of the insertion point in t' (cf. the dashed arrows in Fig. 1). The result of a TAG derivation is twofold: a *derived tree*, and a *derivation tree*. The former represents the syntactic tree resulting from tree rewriting. The latter shows which ETs have been combined and how, as shown in Fig. 1(b). The derived tree of a sentence containing a syntactically regular MWE is identical to the one with its compositional reading, but their derivation trees differ. Thus, in the context of joint syntactic parsing and MWE identification (cf. Sec. 1), the derived and the derivation trees can be seen as the results of the former and of the latter task, respectively.

A TAG whose every ET contains at least one terminal leaf is called an LTAG (lexicalized TAG). The reason why we are particularly interested in LTAGs is that we consider MWEs a central challenge in NLP, and LTAGs show several advantages with respect to them (Abeillé and Schabes, 1989). Firstly, each MWE, together with the lexical and morphosyntactic constraints that it imposes, can be represented as a unique ET. Unification constraints on feature structures attached to tree nodes allow one to naturally express dependencies between arguments at different depths in the ETs (e.g. the subject-possessive

agreement in *to pull one's socks up*). This is not the case for most other grammatical formalism, which handle long-distance dependencies by feature percolation. Secondly, the so-called *extended domain of locality* offers a natural framework for representing two different kinds of discontinuities. Namely, discontinuities coming from the internal structure of a MWE (e.g. required but non-lexicalized arguments) are directly visible in elementary trees and are handled in parsing mostly by substitution. Discontinuities coming from insertion of adjuncts (e.g. *a bunch of NP*, *a whole bunch of NP*) are invisible in elementary trees but are handled by adjunction.

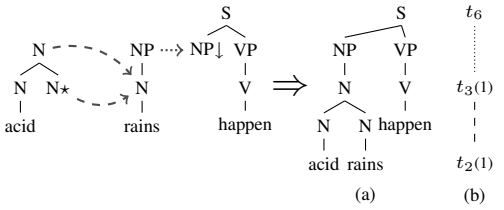


Figure 1: Tree rewriting in TAG resulting in a derived tree (a), and a derivation tree (b).

(t_1)	(t_2)	(t_3)	(t_4)	(t_5)	(t_6)
NP N acid	N N N⋆ acid	NP N rains	S NP↓ VP V rains	NP N N acid rains	S NP↓ VP V happen
$NP \rightarrow N_0$ $N_0 \rightarrow$ acid	$N \rightarrow N_1\ N\star$ $N_1 \rightarrow$ acid	$NP \rightarrow N_2$ $N_2 \rightarrow$ rains	$S \rightarrow NP\ VP_3$ $VP_3 \rightarrow V_4$ $V_4 \rightarrow$ rains	$NP \rightarrow N_5\ N_6$ $N_5 \rightarrow$ acid $N_6 \rightarrow$ rains	$S \rightarrow NP\ VP_7$ $VP_7 \rightarrow V_8$ $V_8 \rightarrow$ happen

Figure 2: A toy TAG grammar converted into flat rules

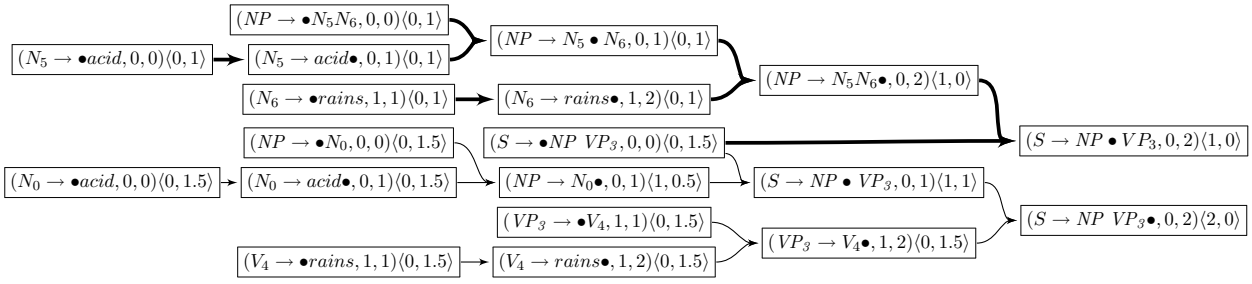


Figure 3: Hypergraph representing the chart parsing of the substring *acid rains* with ETs t_1, t_4 and t_5 from Fig. 2. The lowest-cost path representing the idiomatic interpretation is highlighted in bold.

3 Promoting MWEs in weighted TAG parsing

The fact that MWEs are represented in LTAGs as ETs allows us to propose a very simple and yet powerful strategy of promoting them in parsing. As seen in Sec. 2, parsing with an LTAG consists in combining ETs via substitution or adjunction. We define the weight of a full parse as the sum of the weights of the participating ETs. Note that the more sentence words belong to MWEs, and the longer are those MWEs, the less ETs are needed to cover the sentence. Suppose, for instance, that the sequence *acid rains* in Fig. 1 is covered by its idiomatic interpretation represented by tree t_5 from Fig. 2, instead of being handled by adjunction. In this case parsing *acid rains happen* produces the same derived tree as before but the derivation tree is smaller: it involves 2 ETs instead of 3.

This simple observation underlies our idea of promoting MWE-oriented analyses. Namely, suppose the input LTAG trivially weighted, i.e., each ET having weight 1. Then, finding analyses containing the maximum number of MWEs boils down to achieving the lowest-weight parses. Our objective is to find them more rapidly than other parses, which can be achieved by an A* algorithm using a MWE-driven heuristics, as described in the following sections. See also Sec. 8 for considerations on how this solution might generalize to non-trivially weighted grammars, notably with weights estimated on the basis of treebanks.

4 Weighted parsing with a flattened TAG

In (Waszczuk et al., 2016) we presented a TAG parsing architecture based notably on grammar flattening, subtree sharing and finite-state-based compression. Here, we sketch a simplified version of this architecture, and explain how it implements parsing as an A* graph traversal algorithm. Then in Sec. 5 we

define the heuristic implementing the MWE promoting strategy, which – to the best of our knowledge – is totally novel.

Consider again the LTAG in Fig. 2. For the sake of presentation and compression (cf. Sec. 6), we represent TAG ETs as sets of flat production rules (Alonso et al., 1999) with indexed non-terminals.[1] For instance, the two N non-terminals in t_5 receive different indexes so as to avoid spurious analyses like $[[rains]_N[acid]_N]_{NP}$. A rule headed by the root of an ET (e.g., $S \rightarrow NP\ VP_3$) is called a *top rule*. The other rules are called *inside rules*.

Suppose that only the first two words of sentence (4) are to be parsed with a grammar subset limited to t_1, t_4 and t_5. With a flattened grammar representation, TAG parsing comes close to CFG parsing (even if dedicated inference rules are needed for adjunction, which is neglected in this paper). Like for CFG, an Earley-style parsing process for TAGs defined within a deductive framework (Shieber et al., 1995), involving an *agenda* (queue of weighted items) and a *chart*, can be represented as a hypergraph (Klein and Manning, 2001), more precisely a B-graph (Gallo et al., 1993), whose nodes are items of the chart and of the agenda, and whose hyperarcs represent applications of inference rules, as shown in Fig. 3. Each item $\mathcal{I} = (r, k, l)$ contains a dotted rule r and the span (k, l) over which the symbols to the left of the dot have been parsed.[2] For instance, the hyperarc leading from $(N_5 \rightarrow \bullet acid, 0, 0)$ to $(N_5 \rightarrow acid\bullet, 0, 1)$ means that the terminal $acid$ has been scanned from position 0 to 1. The latter item can then be combined with $(NP \rightarrow \bullet N_5 N_6, 0, 0)$ to yield $(NP \rightarrow N_5 \bullet N_6, 0, 1)$, etc. $\mathcal{I}$ and r are called *passive*, if the dot occurs at the end of r, and *active* otherwise. A sentence s has been parsed if a target item has been reached (spanning over the whole sentence, with a passive top rule headed by S).

The specificity of such a hypergraph lies in the fact that it is dynamically generated as the parsing process goes on. The main objectives include the generation of the smallest possible portion of this hypergraph, while including all the requested parses. In our case those are all optimal parses[3], in the sense of the MWE-promoting strategy.

Each derivation traversing $\mathcal{I} = (r, k, l)$ and resulting in a full parse tree T can be divided into two parts: (i) $\mathcal{I}$'s *inside derivation*, i.e., the part of the derivation corresponding to a (possibly partial) subtree of T rooted at r' head and spanning over (k, l), (ii) $\mathcal{I}$'s *outside derivation*, the part of the derivation corresponding to a partial tree obtained from T but excluding $\mathcal{I}$'s *inside derivation*. The weights of $\mathcal{I}$'s best inside and outside derivations are denoted by $\beta(\mathcal{I})$ and $\alpha(\mathcal{I})$. They are calculated according to the strategy described in Sec. 3, i.e. as numbers of ETs involved.

In symbolic CFG parsing, and in deductive parsing in general, the sentence parsability problem boils down to target node B-reachability in the (gradually constructed) hypergraph, and can be solved e.g. by a depth-first search generalized to hypegraphs. In probabilistic CFG parsing, parse trees and hypergraph B-paths are scored, and discovering the best parse is equivalent to finding the shortest B-path, which can be done by Dijkstra's algorithm generalized to hypergraphs (Gallo et al., 1993). The search space of this basic algorithm can be reduced in the A* algorithm (Klein and Manning, 2003), by introducing a heuristic which estimates the distance of each node to a target node. Namely, each $\mathcal{I}$ is assigned two values: $\beta(\mathcal{I})$ and $h(\mathcal{I})$, the latter being an estimation of $\alpha(\mathcal{I})$. The parsing items are popped from agenda in increasing order of $\beta(\mathcal{I}) + h(\mathcal{I})$. The heuristic used to calculate $h(\mathcal{I})$ should be *admissible*, i.e. should never overestimate ($h(\mathcal{I}) \leq \alpha(\mathcal{I})$). Additionally, if the heuristic is *monotonic* (i.e. $\beta(\mathcal{I}) + h(\mathcal{I})$ never increases), then an item is never re-introduced into the agenda once is has been popped, and the algorithm runs faster.

We apply the A* algorithm in a slightly adapted version in that we do not search for one but for all optimal parses, i.e. those containing grammar-compliant idiomatic interpretations. Thus, we do not quit when the first target item has been reached, but only when we are sure that no more optimal derivations can be found. As long as $\mathcal{I}$ stays on the agenda, $\beta(\mathcal{I})$ has to be recalculated each time a new hyperarc with head node $\mathcal{I}$ is added. Once $\mathcal{I}$ moves to the chart, $\beta(\mathcal{I})$ remains constant. In Fig. 3, the couple $\langle \beta(\mathcal{I}), h(\mathcal{I}) \rangle$ decorates each node. Note that in case of parsing with a flattened TAG, only an ET t, not its

[1] Our proposal applies, however, to other LTAG representations as well.

[2] For simplicity, we ignore the fact that an item's span can include a gap accounting for adjunction.

[3] In probabilistic CFG parsing, the 1-best parse (Klein and Manning, 2003) or k-best parses (Pauls and Klein, 2009) are usually considered.

individual flat rules, is assigned a weight. Therefore, t's weight contributes to $\beta(\mathcal{I})$ only when t has been fully parsed, and it contributes to $h(\mathcal{I})$ otherwise. For instance, going from items $(NP \rightarrow N_5 \bullet N_6, 0, 1)$ and $(N_6 \rightarrow rains\bullet, 1, 2)$ to $(NP \rightarrow N_5 N_6 \bullet, 0, 2)$ we have completed parsing the top rule of t_5, thus the weight of this ET (1) is added to W_1. However, item $(N_6 \rightarrow rains\bullet, 1, 2)$ is decorated with $\langle 0, 1\rangle$, since no ET has been fully parsed so far but we are parsing tree t_5 (with weight 1), whose terminals fully cover the intended span $(0, 2)$.

5 MWE-driven heuristic

The proper choice of the heuristic is crucial for the performance of the A* algorithm. We propose a heuristic $h(\mathcal{I})$ specifically designed to handle MWEs and, more generally, ETs with multiple anchors, which allows to use the A* parsing algorithm with MWE-aware weighted TAG grammars. In case weight 1 is assigned to all ETs, the heuristic closely models the strategy of promoting MWEs described in Sec. 3. Namely, it admits that if a given MWE has a chance to occur in the part of the sentence that remains to be parsed (i.e., in its outside derivation), then this MWE probably occurs. More precisely, the yet unparsed portion of the sentence can be divided into two parts: (i) the terminals yet to be covered by the tree that we are currently parsing, (ii) the remaining terminals. The heuristic consists in considering each terminal s_i from (ii) separately and assuming that it will be parsed with the ET containing s_i within the longest possible MWE.

Formally, let $\mathcal{S} = s_1 s_2 \dots s_{|\mathcal{S}|}$ be the input sentence and $Pos(\mathcal{S})$ the set of positions between its words, ranging from 0 to $|\mathcal{S}|$. Since the same word can occur more than once in a sentence or a tree, we manipulate multisets of words. For a set X, a multiset over X is a set of pairs $\{(x, m(x)) : x \in X\}$, where $m(x) \in \mathbb{N}^+$ is called the multiplicity of x. We extend set notations and operators to multisets. For instance, $\{(a, 2), (b, 1)\}$ is noted as $\{a, a, b\}_{ms}$, and we have $\{a, b\}_{ms} \cup \{a\}_{ms} = \{a, a, b\}_{ms}$, $\{a, a, b\}_{ms} \setminus \{a, b\}_{ms} = \{a\}_{ms}$, $\{a, b\}_{ms} \subseteq \{a, a, b\}_{ms}$, $\{a, a, b\}_{ms} \not\subseteq \{a, b\}_{ms}$, $|\{a, a, b\}_{ms}| = 3$, etc. For any set X, let $\mathcal{M}(X)$ be the set of all multisets over X. Let $Rest(\mathcal{I})$ denote a multiset of words in the input sentence $\mathcal{S}$ outside of $\mathcal{I}$'s span, i.e., $Rest(\mathcal{I}) = \{s_1, \dots, s_k, s_{l+1}, \dots, s_{|\mathcal{S}|}\}_{ms}$.[4] Let $tree(r)$ be the ET from which r stems, and $W(t) \in [0, \infty)$ the weight of the ET t. For instance, in Fig. 2 and 3, for $r = N_5 \rightarrow acid\bullet$ we have $tree(r) = t_5$ and $W(t_i) = 1$ for $i = 1, \dots, 6$.

Let $sub(t) \in \mathcal{M}(\Sigma)$ be the multiset of terminals in tree t. For instance, $sub(t_5) = \{acid, rains\}_{ms}$. For each word w, let $minw(w)$ denote the minimal weight of scanning w by an ET, i.e., the minimum proportion of w among all terminals of a single ET. More precisely,

$$minw(w) = \min_{t:(w,i)\in sub(t)} \frac{W(t)}{|sub(t)|}. \tag{5}$$

For instance, the proportion of *acid* in the terminals of t_1, t_2 and t_5 is, 1, 1 and 0.5, respectively, so $minw(acid) = 0.5$. Similarly $minw(rains) = 0.5$.[5] Thus, with all ET weights equal to 1, the longer a MWE, the lower are the $minw$ values of its components.

Let $sub(r), super(r) \in \mathcal{M}(\Sigma)$ be the multisets of terminals occurring in $tree(r)$ inside and outside of the subtree rooted at r's head, respectively. For instance, $sub(N_5 \rightarrow acid\bullet) = \{acid\}_{ms}$ and $super(N_5 \rightarrow acid\bullet) = \{rains\}_{ms}$. Note that for any top rule r, $super(r) = \emptyset_{ms}$.

Let $\mathit{suff}(r)$ be the set of passive non-top rules headed by the symbols in r' body after the dot. For instance, $\mathit{suff}(NP \rightarrow N_5 \bullet N_6) = \{N_6 \rightarrow rains\bullet\}$ and $\mathit{suff}(S \rightarrow \bullet NP\, VP_3) = \{VP_3 \rightarrow V_4\bullet\}$. Note that if r is passive, $\mathit{suff}(r) = \emptyset$.

Finally, let $Req(\mathcal{I})$ be the multiset of words required by the yet unparsed part of the current tree, i.e.,

$$Req(\mathcal{I}) = super(r) \cup \bigcup_{p\in \mathit{suff}(r)} sub(p). \tag{6}$$

[4] In case of adjunction $\mathcal{I}$'s span includes two additional indices denoting the gap, and the words within the gap also belong to $Rest(\mathcal{I})$.

[5] Variants of the $minw(w)$ definition include distributing the weights of individual terminals in an ET proportionally to their frequencies in the corpus. Our experiments did not show any advantage of such a distribution over the uniform one.

For instance in Fig.3, for item $\mathcal{I} = (NP \rightarrow N_5 \bullet N_6, 0, 1)$ we have $super(NP \rightarrow N_5 \bullet N_6) = \emptyset_{ms}$, $sub(N_6 \rightarrow rains\bullet) = \{rains\}_{ms}$, and $Req(\mathcal{I}) = \{rains\}_{ms}$.

For any item $\mathcal{I} = (r, k, l)$ we define a primary heuristic $h_0(\mathcal{I})$ as in equation (7).

$$h_0(\mathcal{I}) = \begin{cases} \infty, \text{ if } Req(\mathcal{I}) \not\subseteq Rest(\mathcal{I}) \\ \sum\limits_{(s,i) \in Rest(\mathcal{I}) \setminus Req(\mathcal{I})} minw(s) \times i, \text{ otherwise} \end{cases} \tag{7}$$

Then the estimation for the weight of $\mathcal{I}$'s best outside derivation, i.e. $\alpha(\mathcal{I})$, is given by equation (8).

$$h(\mathcal{I}) = \begin{cases} h_0(\mathcal{I}), \text{ if } \mathcal{I} \text{ is a top-rule passive item} \\ W(tree(r)) + h_0(\mathcal{I}), \text{ otherwise} \end{cases} \tag{8}$$

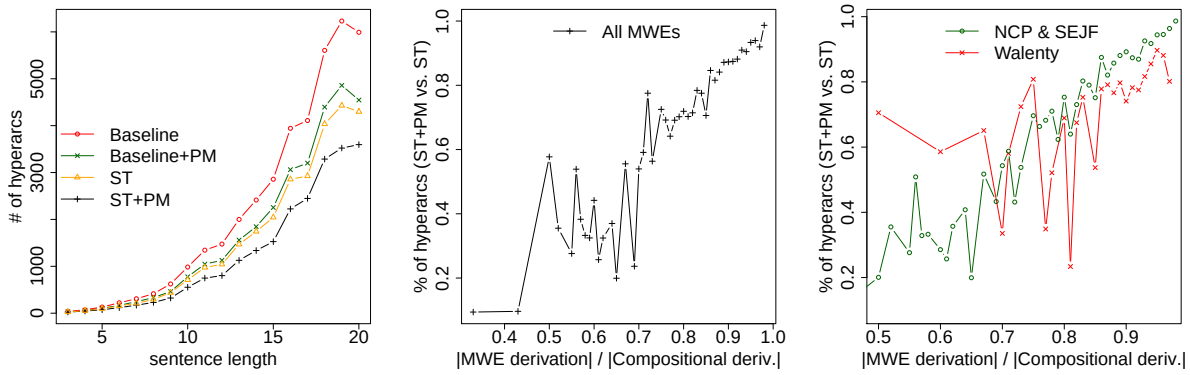
For instance, in the top-rule passive item $(NP \rightarrow N_0\bullet, 0, 1)$ we have finished parsing t_1 ($\beta(\mathcal{I}) = 1$) and we still have to consume *rains*, which implies a weight at least equal to $h(\mathcal{I}) = minw(rains) = 0.5$. In the inside-rule passive item $\mathcal{I} = (N_5 \rightarrow acid\bullet, 0, 1)$ we have $Rest(\mathcal{I}) = \{rains\}_{ms}$, $Req(N_5 \rightarrow acid\bullet) = \{rains\}_{ms}$, thus $h(\mathcal{I}) = W(t_5) = 1$. Finally, in the active item $\mathcal{I} = (NP \rightarrow N_5 \bullet N_6, 0, 1)$ we have $Rest(\mathcal{I}) = \{rains\}_{ms}$, $super(NP \rightarrow N_5 \bullet N_6) = \emptyset_{mt}$, and $Req(\mathcal{I}) = \{rains\}_{ms}$, thus $h(\mathcal{I}) = W(t_5) = 1$.

With this heuristic, and weight 1 assigned to individual ETs, the derivations containing MWEs are often reached before the paths towards compositional ones are even followed. For instance the item $(N_0 \rightarrow acid\bullet, 0, 1)$ has the estimated cost 1.5, and it will be created later than $(S \rightarrow NP\, VP_3\bullet, 0, 2)$. Thus, the hyperpath (highlighted in bold) assuming the idiomatic reading of *acid rains*, will be followed before the path assuming that *rains* is a verb.

For a given item the heuristic assumes that each remaining word w from the input sentence (with the exception of the words required by the rule underlying the item) will be scanned with the lowest possible cost, i.e. $minw(w)$ – see Eq. (5). The heuristic never over-estimates the cost of parsing the remaining part of the sentence and is thus *admissible*. All but one inference rules of the parser are also *monotonic*, in the sense that the estimation, stemming from the application of an inference rule, of the total weight $\beta(\mathcal{I}) + h(\mathcal{I})$ of an item $\mathcal{I}$ is greater or equal to the total weight, $\beta(\mathcal{I}') + h(\mathcal{I}')$, of any premise item $\mathcal{I}'$ of this rule. The sole exception concerns the inference rule – called *foot adjoin (FA)*, see (Waszczuk et al., 2016) – responsible for recognizing the so-called *gaps* over which adjoining could be performed. This is related to the fact that the weight of the item inferred with FA does not depend on the $\beta(\mathcal{I}')$ weight of its premise item $\mathcal{I}' = (r, k, l)$, where item $\mathcal{I}'$ provides an evidence that adjunction could possibly take place over span (k, l). Nonetheless, the algorithm guarantees that when item $\mathcal{I}$ is popped from the agenda, one of the hyperarcs representing an optimal derivation of $\mathcal{I}$ is already inferred, and thus the $\beta(\mathcal{I})$ value is correctly calculated.

6 Experimental results

We evaluated our parsing strategy with Składnica, a Polish treebank with over 9,000 manually disambiguated constituency trees (Świdziński and Woliński, 2010). As it contains no MWE annotations, we produced them automatically, by projecting 3 existing MWE resources: (i) the named entity (NE) layer of the National Corpus of Polish (NCP) (Savary et al., 2010) (only the multiword NEs were taken into account), (ii) SEJF, an extensional lexicon of Polish nominal, adjectival and adverbial MWEs (Czerepowicka and Savary, 2015), (iii) Walenty, a Polish valence dictionary (Przepiórkowski et al., 2014) with over 8,000 verbal MWEs. The mapping for (i) was straightforward and did not require manual validation, since Składnica is a subcorpus of the NCP, whose NE annotation and adjudication were performed manually. The mapping for (ii) and (iii), followed by a manual validation, consisted in searching for syntactic nodes satisfying all lexical constraints and part of syntactic constraints of a MWE entry. The required lexical nodes were to be contiguous for (ii) but not for (iii). As a result, 2026 idiomatic occurrences (1303 from NCP-NE, 368 from SEJF and 355 from Walenty) and 40 compositional ones (22 for SEJF and 18 for Walenty) were identified, which implies the idiomaticity rate about 0.95 (0.95 for Walenty and 0.94 for SEJF).



(a) (b) (c)

Figure 4: (a) Average number of hyperarcs explored depending on the parsing strategy (for clarity using only sentence of length < 20), (b) Average % of hyperarcs explored with the PM+ST strategy, using the ST strategy as a reference, and (c) Average % of hyperarcs explored depending on the type of MWEs.

A TAG grammar with 28652 lexicalized elementary trees was then extracted from the MWE-marked treebank, similarly to (Krasnowska, 2013) or (Chen and Shanker, 2004). Each treebank subtree marked for a MWE yielded: (i) a MWE-dedicated ET containing all paths leading to the lexical (co-)anchors, (ii) ETs covering the compositional interpretations. Various compression techniques can be applied to a flattened TAG (Waszczuk et al., 2016). We used a representation in which common subtrees and prefixes of flat rules are shared.

We assess our parser's efficiency in terms of the size of its parsing hypergraph. We believe it to be a more objective measure to compare different parsing strategies than the absolute parsing time, since each hypergraph edge corresponds to an application of an inference rule, i.e. to a basic parsing step (as in theoretical complexity considerations).[6] Conversely, the parsing time is highly dependent on the low level implementation details.[7]

The baseline hypergraph is the one generated with the full grammar, when no MWE-promoting strategy is used and all grammar-compliant parses are generated for each sentence. The MWE-promoting (PM) hypergraph, compared to this baseline, includes mainly the optimal parses (the algorithm ensures that, in PM, all optimal parses are achieved, but some sub-optimal parses may also be reached, since heuristic h is an imperfect estimation of α), i.e. those in which the maximum number of words belongs to potential MWEs.

The experiment was carried out on the same dataset from which the grammar was extracted. Therefore, for each sentence, the baseline hypergraph contained both its gold (i.e., conforming to Składnica) parse (derived tree) and its gold MWE identification (derivation tree). The PM hypergraphs, in turn, contained the correct parses for virtually 100% of the sentences,[8] and correct MWE identification for around 95% of them (due to the idiomaticity rate equal to 0.95). Thus, the parsing efficiency gain due to the PM strategy occurred with no loss of accuracy.

The PM strategy is comparable to supertagging (ST), i.e. pre-selecting, for each sentence, a subset of ETs which have good chances to be used in the derivation, in order to reduce the parsing search space. We experimented with a simple form of ST, which restricts the grammar to ETs whose terminals occur in the given sentence. Namely, we examined the ST hypergraph containing all parses for each sentence, and the one when ST was combined with PM (where mainly optimal parses were achieved).

Fig. 4a shows the absolute sizes of the hypergraphs for these 4 strategies in function of the sentence

[6]The overhead related to computing the values of the heuristic is at most linear in the size of the sentence, and may be much lower with efficient low-level optimizations.

[7]In an optimized implementation, TAG parsing time is proportional to the number of hyperarcs, as reported by (Waszczuk et al., 2016).

[8]A sanity check showed that for 54 sentences the gold parse was not found, mainly due to some abbreviation- and letter-case-related specificities, as well as to missing MWE annotations in Składnica.

length. The PM strategy brings enhancement regardless of whether supertagging is used or not. The supertagging alone outperforms, on average, the baseline MWEs-promoting strategy. Since the combination of ST and PM strategies proves the most efficient, we restrict further experiments to this version.

Note that Fig. 4a does not fully reflect the potential advantages of the PM strategy, whose behavior does not directly depend on the length of the parsed sentence, but rather on the number and the size of the MWEs potentially occurring in it. These 2 values can be together represented as the ratio of the size of the MWE-based derivation tree to the size of the corresponding compositional derivation tree (i.e. the one assuming no MWE occurrence). Expectedly, as shown in Fig. 4b, the lower this ratio (i.e. the more words in the sentence belong to MWEs, and the longer are these MWEs), the more significant the hypergraph size reductions. Moreover, the resulting graph suggests that the hypergraph size reductions are linear with respect to this ratio. Note that the vertical axis now shows the proportional gain in the hypergraph size due to the ST+PM strategy with respect to the ST strategy alone.

Finally, we investigated the behavior of the PM+ST strategy for two types of MWEs independently: verbal MWEs from Walenty and compounds from NCP and SEJF. As shown in Fig. 4c, verbal MWEs, while less frequent, prove to be better in reducing ambiguity for sentences with low number of potential MWEs. It is hard to ascertain this claim for sentences with lower gold derivation size ratio. While compounds seem to outperform verbal MWEs in this case, sentences with verbal MWEs for which this ratio is low are also very short in our dataset (of length 5, on average, for the 20 sentences with the lowest ratio), and thus exhibit low syntactic ambiguity.

7 Related work

While A* algorithms have been widely used for AI inference problems where a lightest derivation is to be found (Felzenszwalb and McAllester, 2007), this is to our knowledge the first attempt at using them within the context of MWE parsing with TAG. This work was inspired by Lewis and Steedman (2014) who applied A* to parsing with another strongly lexicalized grammar formalism, namely CCG. Unlike in this work, our grammar rules are not constrained to have a single lexical item, hence they can explicitly represent MWEs. This calls for a more elaborate heuristic, since a not yet parsed terminal can either be consumable by the currently parsed tree or not, as is the case with *rains* in item $(NP \rightarrow N_5 \bullet N_6, 0, 1)$ as opposed to $(NP \rightarrow N_0 \bullet, 0, 1)$ in Fig. 3. Distinguishing these two cases leads to a more precise weight estimation.

Angelov and Ljunglöf (2014) proposed to apply A* top-down parsing to parallel multiple context-free grammars, a formalism strictly more expressive than TAGs. In their approach weights are assigned to production rules and the grammar is not assumed to be strongly lexicalized, which complicates the design of an efficient heuristic. Their evaluation showed that a non-admissible heuristic can be orders of magnitude faster than the admissible version, at the expense of parsing quality.

Other ways of dealing with MWEs in the context of TAG would involve pre- or post-processing. A post-processing step would consist in identifying MWE interpretations in derivation structures (potentially with an additional processing cost). Regarding pre-processing, current state-of-the-art techniques are related to probabilistic supertagging (Bangalore and Joshi, 1999), as opposed to the simple symbolic supertagging applied in Sec.6. While labeling the words of a sentence with candidate ETs, one may either keep for each word the most probable ET, or all ETs whose probabilities are above a given threshold. Large MWE annotations are needed to train such supertaggers. Probabilistic treatment of contiguous MWEs has been applied to Tree-Substitution Grammar with encouraging results (Green et al., 2013). The main drawback of such probabilistic pre-processing is the fact that it can prevent the parser from finding the right derivations in case when the supertagging was wrong. This situation is avoided in A* parsing which, while requiring that candidate ETs be annotated with the corresponding probabilities, performs a filtering of unlike ET candidates on the fly.

An alternative to probabilistic supertagging has been proposed by Boullier (2003). There, an approximated CFG grammar is computed from an input TAG, and used to parse the input sentence so as to decide which ETs should be selected for TAG parsing. This approach has been enhanced by Gardent et al. (2014) to take word order into account. We consider such a supertagging technique an interesting

candidate for future work. One could indeed not only select ETs that are compatible with the sentence to parse but also distinguish ETs for literal interpretations from ETs for MWEs. Like non-statistical supertagging, using an A* algorithm has the advantage to process MWEs while keeping ambiguity as long as possible to avoid dismissing valid interpretations.

Relatively few works have explicitly addressed the idiomaticity rate of MWEs. (Savary et al., 2012) perform a straightforward matching of a Polish economic MWE lexicon, containing extensional descriptions of morpho-syntactic variants, against a corpus and obtain only 0.12%–0.21% of false positives. (El Maarouf and Oakes, 2015) examine 10 verbal MWEs in the British National Corpus and find out that the idiomaticity measure for half of them exceeds 0.95, and for 9 most frequent of them is above 0.676.

8 Conclusions and future work

We have presented a novel LTAG parsing architecture in which parses potentially containing MWEs are given higher priorities so as to be achieved faster than the competing compositional analyses. The underlying A* algorithm uses a distance estimation heuristic based on the number of terminal nodes in elementary trees. The results obtained with a Polish TAG grammar show that this strategy can considerably reduce the number of parsing items to be explored in order to generate a subset of parses very likely to contain the correct parse. The tests used a grammar extracted from a MWE-annotated treebank but the method also applies to hand-crafted grammars.

Future work includes possible enhancements of the A* heuristic. It currently does not require that, if an ET is used to scan an input terminal, then all the other terminals of this ET also have to be present. It does not require either that the terminals need to be scanned in the appropriate order. Taking such constraints into account might enhance both the parsing quality and speed. Note also that the heuristic ignores ETs which contain no lexical anchors, so it is mainly adapted to strongly lexicalized TAGs. Relieving this constraint, while preserving MWE promotion, would be worth consideration.

Another perspective is to evaluate the computational overhead of the MWE-based heuristic, as opposed to identifying MWEs in a post-parsing step. Also, a fine-grained estimation of the idiomaticity rate of different types of MWEs might give us hints as to which of them should best be identified before, during or after parsing. With such data at hand, it should be possible to construct a multi-stage MWE-aware parsing architecture, tunable for optimum trade-off between accuracy and speed.

Even with MWE lexicon mapping on a treebank, as shown in Sec. 6, sufficiently large MWE-annotated treebanks are hard to obtain, and if they do exist, they are still concerned by MWE sparseness. In the long run, we aim at a hybrid parsing architecture in which a MWE-driven parser is fed with a probabilistic TAG grammar combined with MWE lexicons. We believe that such an extension of our solution to a hybrid setting is possible due to two factors. Firstly, the heuristic described in Sec. 5 generalizes to any weighted TAG with non-negative weights assigned to individual ETs. Secondly, systematically promoting MWE-oriented analyses in probabilistic parsing can be achieved even if MWEs are underrepresented in the training corpus. Namely, MWE-oriented ETs could stem from a syntactic MWE lexicon, such as Walenty (Przepiórkowski et al., 2014), while their weights could be calculated from the weights of the ETs corresponding to their compositional analyses. Alternatively, the weights could be represented as lexicographically ordered pairs, consisting of (i) the number of ETs participating in the underlying derivations, and (ii) the actual weights stemming from the weighted grammar.

Finally, integrating feature structures and unification within this parsing framework might lead to faster pruning of spurious analyses, and enable a more precise MWE identification, especially for inflectionally rich languages like Polish.

Acknowledgments

This work has been supported by the European Framework Programme Horizon 2020 via the PARSEME[9] European COST Action (IC1207), as well as by the French National Research Agency (ANR) via the PARSEME-FR[10] project (ANR-14-CERA-0001).

[9] www.parseme.eu

[10] http://parsemefr.lif.univ-mrs.fr

References

Anne Abeillé and Yves Schabes. 1989. Parsing idioms in lexicalized tags. In Harold L. Somers and Mary McGee Wood, editors, *Proceedings of the 4th Conference of the European Chapter of the ACL, EACL'89, Manchester*, pages 1–9. The Association for Computer Linguistics.

Miguel Alonso, David Cabrero, Eric Villemonte de la Clergerie, and Manuel Vilares Ferro. 1999. Tabular algorithms for TAG parsing. In *EACL 1999*, pages 150–157.

Krasimir Angelov and Peter Ljunglöf. 2014. Fast statistical parsing with parallel multiple context-free grammars. In *EACL*, volume 14, pages 368–376.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Comput. Linguist.*, 25(2):237–265, June.

Pierre Boullier. 2003. Supertagging: a Non-Statistical Parsing-Based Approach. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 55–65, Nancy, France.

Marie Candito and Matthieu Constant. 2014. Strategies for Contiguous Multiword Expression Analysis and Dependency Parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 743–753.

John Chen and Vijay K. Shanker. 2004. New developments in parsing technology. chapter Automated Extraction of Tags from the Penn Treebank, pages 73–89. Kluwer Academic Publishers, Norwell, MA, USA.

Matthieu Constant and Joakim Nivre. 2016. A transition-based system for joint lexical and syntactic analysis. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 161–171, Berlin, Germany, August. Association for Computational Linguistics.

Matthieu Constant, Anthony Sigogne, and Patrick Watrin. 2012. Discriminative strategies to integrate multiword expression recognition and parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 204–212, Stroudsburg, PA, USA. Association for Computational Linguistics.

Monika Czerepowicka and Agata Savary. 2015. SEJF - a Grammatical Lexicon of Polish Multi-Word Expression. In *Proceedings of Language and Technology Conference (LTC'15), Poznań, Poland*. Wydawnictwo Poznańskie.

Ismail El Maarouf and Michael Oakes. 2015. Statistical Measures for Characterising MWEs. In *IC1207 COST PARSEME 5th general meeting*.

Pedro Felzenszwalb and David McAllester. 2007. The generalized A* architecture. *Journal of Artificial Intelligence Research*, 29:153–190.

Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. 1993. Directed hypergraphs and applications. *Discrete Appl. Math.*, 42(2-3):177–201, April.

Claire Gardent, Yannick Parmentier, Guy Perrier, and Sylvain Schmitz. 2014. Lexical Disambiguation in LTAG using Left Context. In Zygmunt Vetulani and Joseph Mariani, editors, *Human Language Technology. Challenges for Computer Science and Linguistics. 5th Language and Technology Conference, LTC 2011, Poznan, Poland, November 25-27, 2011, Revised Selected Papers*, volume 8387, pages 67–79. Springer.

Spence Green, Marie-Catherine de Marneffe, and Christopher D. Manning. 2013. Parsing Models for Identifying Multiword Expressions. *Computational Linguistics*, 39(1):195–227.

Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. 1975. Tree adjunct grammars. *Journal of the Computer and System Sciences*, 10:136–163.

Dan Klein and Christopher D. Manning. 2001. Parsing and hypergraphs. In *Proceedings of the Seventh International Workshop on Parsing Technologies (IWPT-2001), 17-19 October 2001, Beijing, China*. Tsinghua University Press.

Dan Klein and Christopher D. Manning. 2003. A* parsing: Fast exact viterbi parse selection. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Katarzyna Krasnowska. 2013. Towards a polish LTAG grammar. In Mieczyslaw A. Klopotek, Jacek Koronacki, Malgorzata Marciniak, Agnieszka Mykowiecka, and Slawomir T. Wierzchon, editors, *Language Processing and Intelligent Information Systems - 20th International Conference, IIS 2013, Warsaw, Poland, June 17-18, 2013. Proceedings*, volume 7912 of *Lecture Notes in Computer Science*, pages 16–21. Springer.

Mike Lewis and Mark Steedman. 2014. A* CCG Parsing with a Supertag-factored Model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000. Association for Computational Linguistics.

Alexis Nasr, Carlos Ramisch, José Deulofeu, and Valli André. 2015. Joint Dependency Parsing and Multiword Expression Tokenisation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL'15)*.

Joakim Nivre and Jens Nilsson. 2004. Multiword units in syntactic parsing. In *Proceedings of MEMURA 2004 – Methodologies and Evaluation of Multiword Units in Real-World Applications, Workshop at LREC 2004, May 25, 2004, Lisbon, Portugal*, pages 39–46, Lisbon, Portugal, May.

Adam Pauls and Dan Klein. 2009. K-best a* parsing. In Keh-Yih Su, Jian Su, and Janyce Wiebe, editors, *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 958–966. The Association for Computer Linguistics.

Adam Przepiórkowski, Elżbieta Hajnicz, Agnieszka Patejuk, and Marcin Woliński. 2014. Extended phraseological information in a valence dictionary for NLP applications. In *Proceedings of the Workshop on Lexical and Grammatical Resources for Language Processing (LG-LP 2014)*, pages 83–91, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Victoria Rosén, Gyri Smørdal Losnegaard, Koenraad De Smedt, Eduard Bejček, Agata Savary, Adam Przepiórkowski, Petya Osenova, and Verginica Barbu Mitetelu. 2015. A survey of multiword expressions in treebanks. In *Proceedings of the 14th International Workshop on Treebanks & Linguistic Theories conference*, Warsaw, Poland, December.

Agata Savary, Jakub Waszczuk, and Adam Przepiórkowski. 2010. Towards the annotation of named entities in the national corpus of polish. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

Agata Savary, Bartosz Zaborowski, Aleksandra Krawczyk-Wieczorek, and Filip Makowiecki. 2012. SEJFEK - a Lexicon and a Shallow Grammar of Polish Economic Multi-Word Units. In *Proceedings of the 3rd Workshop on Cognitive Aspects of the Lexicon*, pages 195–214, Mumbai, India, December. The COLING 2012 Organizing Committee.

Stuart M Shieber, Yves Schabes, and Fernando CN Pereira. 1995. Principles and implementation of deductive parsing. *The Journal of logic programming*, 24(1):3–36.

Jakub Waszczuk, Agata Savary, and Yannick Parmentier. 2016. Enhancing practical TAG parsing efficiency by capturing redundancy. In *21st International Conference on Implementation and Application of Automata (CIAA 2016)*, Proceedings of the 21st International Conference on Implementation and Application of Automata (CIAA 2016), Séoul, South Korea, July.

Eric Wehrli, Violeta Seretan, and Luka Nerima. 2010. Sentence analysis and collocation identification. In *Proceedings of the Workshop on Multiword Expressions: from Theory to Applications (MWE 2010)*, pages 27–35, Beijing, China, August. Association for Computational Linguistics.

Eric Wehrli. 2014. The relevance of collocations for parsing. In *Proceedings of the 10th Workshop on Multiword Expressions (MWE)*, pages 26–32, Gothenburg, Sweden, April. Association for Computational Linguistics.

Marek Świdziński and Marcin Woliński. 2010. Towards a bank of constituent parse trees for Polish. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech and Dialogue: 13th International Conference, TSD 2010, Brno, Czech Republic*, volume 6231 of *Lecture Notes in Artificial Intelligence*, pages 197–204, Heidelberg. Springer-Verlag.

Incrementally Learning a Dependency Parser to Support Language Documentation in Field Linguistics

Morgan Ulinski* **Julia Hirschberg*** **Owen Rambow†**
*Department of Computer Science
†Center for Computational Learning Systems
Columbia University
New York, NY, USA
{mulinski@cs,julia@cs,rambow@ccls}.columbia.edu

Abstract

We present experiments in incrementally learning a dependency parser. The parser will be used in the WordsEye Linguistics Tools (WELT) (Ulinski et al., 2014a; Ulinski et al., 2014b) which supports field linguists documenting a language's syntax and semantics. Our goal is to make syntactic annotation faster for field linguists. We have created a new parallel corpus of descriptions of spatial relations and motion events, based on pictures and video clips used by field linguists for elicitation of language from native speaker informants. We collected descriptions for each picture and video from native speakers in English, Spanish, German, and Egyptian Arabic. We compare the performance of MSTParser (McDonald et al., 2006) and MaltParser (Nivre et al., 2006) when trained on small amounts of this data. We find that MaltParser achieves the best performance. We also present the results of experiments using the parser to assist with annotation. We find that even when the parser is trained on a single sentence from the corpus, annotation time significantly decreases.

1 Introduction

Although languages have appeared and disappeared throughout history, today languages are facing extinction at an unprecedented pace. Over 40% of the estimated 7,000 languages in the world are at risk of disappearing (Alliance for Linguistic Diversity, 2013). Efforts to document languages become even more important with the increasing rate of extinction. Bird (2009) emphasizes a particular need to make use of computational linguistics during fieldwork. One way the WordsEye Linguistics Tools will address this issue is by providing field linguists with the means to easily document syntax, something that is largely missing from existing documentation tools. We model our tools for documenting syntax on the tools for documenting morphology in SIL FieldWorks Language Explorer (FLEx) (SIL FieldWorks, 2014), one of the most widely-used toolkits for field linguists.

An important part of FLEx is its "linguist-friendly" morphological parser (Black and Simons, 2006), which is fully integrated into lexicon development and interlinear text analysis. The parser is constructed "stealthily," in the background, and can help a linguist by predicting glosses and morphological analyses for interlinear texts. In the same way, WELT will provide tools for specifying the syntax of sentences in the form of dependency structures and use them to train a parser in the background. The parser will provide predictions for new sentences, and, as these are corrected and approved by the linguist, they are added to the training data and the parser is incrementally improved.

This paper makes three contributions. First, we introduce a new corpus of English, Spanish, German, and Egyptian Arabic descriptions of spatial relations and motion events, which we have annotated with dependency structures and other linguistic information. We focused on spatial relations and motion because one of the other functions of WELT will be to assist field linguists with elicitation of spatial language and documentation of spatial and motion semantics. We are making the corpus available to the public. Second, we compare the performance of two existing dependency parsing packages, MSTParser (McDonald et al., 2006) and MaltParser (Nivre et al., 2006), using incrementally increasing amounts of

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 440–449, Osaka, Japan, December 11-17 2016.

this training data. Third, we show that using a parser trained on small amounts of data can assist with dependency annotation.

In Section 2, we discuss related work. In Section 3, we describe the new publicly available corpus. In Section 4, we describe the parsing experiments and discuss the results. Section 5 discusses initial experiments with nonlexical models. We discuss the annotation experiments and results in Section 6. We conclude in Section 7.

2 Related Work

There have been a number of investigations into multilingual dependency parsing. For example, Nivre et al. (2007b) presents detailed results for 11 languages using the arc-eager deterministic parsing algorithm included in MaltParser. However, results are reported only for the parser trained on the full training set and would not generalize to situations where training data is limited. Likewise, the 2006 and 2007 CoNLL shared tasks of multilingual dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007a) relied on the existence of ample training data for the languages being investigated. Our work differs in that we are interested in the performance of a dependency parser trained on very little data.

Duong et al. (2015) approach dependency parsing for a low-resource language as a domain adaptation task; a treebank in a high-resource language is considered out-of-domain, and a much smaller treebank in a low-resource language is considered in-domain. They jointly train a neural network dependency parser to model the syntax of both the high-resource and the low-resource language. In this paper, we focus on the alternate approach of training directly on small amounts of data.

Guo et al. (2015) also investigate inducing dependency parsers for low-resource languages using training data from high-resource languages. They focus on lexical features, which are not directly transferable among languages, and propose the use of distributed feature representations instead of discrete lexical features. Lacroix et al. (2016) describe a method for transferring dependency parsers across languages by projecting annotations across word alignments and learning from the partially annotated data. However, both of these methods rely on large amounts of (unannotated) data in the target language in order to learn the word embeddings and alignments. It is unclear how well these approaches would work in the context of an endangered language where large amounts of unannotated text will not be available.

Our work also differs from the above because our goal is to incorporate a parser into tools for field linguists studying endangered languages. Currently, there are limited options for creating a syntactic parser for an endangered language. The *ParGram* project (ParGram, 2013) aims to produce wide coverage grammars for a variety of languages, but doing so requires knowledge both of the LFG formalism and the XLE development platform (Crouch et al., 2011). It is unlikely that a field linguist would have the grammar engineering skills necessary to create a grammar in this way. Similarly, the LinGO Grammar Matrix (Bender et al., 2002) is a framework for creating broad-coverage HPSG grammars. The Grammar Matrix facilitates grammar engineering by generating "starter grammars" for a language from the answers to a typological questionnaire. Extending a grammar beyond the prototype, however, does require extensive knowledge of HPSG, making this tool more feasibly used by computational linguists than by field linguists. Our work differs from both ParGram and the Grammar Matrix because we do not require the field linguist to master a particular grammar formalism. Instead, linguists will create a syntactic parser simply by labeling individual sentences, a procedure that builds easily upon an existing workflow that already includes annotating sentences with morphological information.

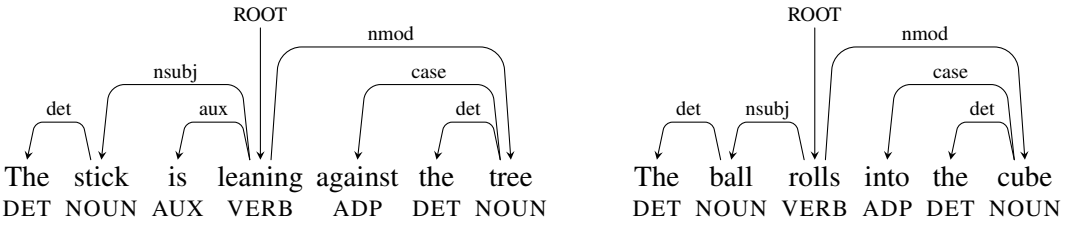
3 Corpus

To obtain a corpus that is similar to sentences that field linguists would probably analyze using WELT, we started with two stimulus kits used by field linguists to study spatial and motion language: the Picture Series for Positional Verbs (Ameka et al., 1999) and the Motion Verb Stimulus Kit (Levinson, 2001). For each picture and video clip, we elicited a one-sentence description from native speakers of English, Spanish, German, and Egyptian Arabic. We chose languages covering a range of linguistic phenomena. For example, German uses morphological case, and Spanish and Arabic both use clitics. In future work, we hope to add languages from other language families, including Chinese and Korean. Our languages

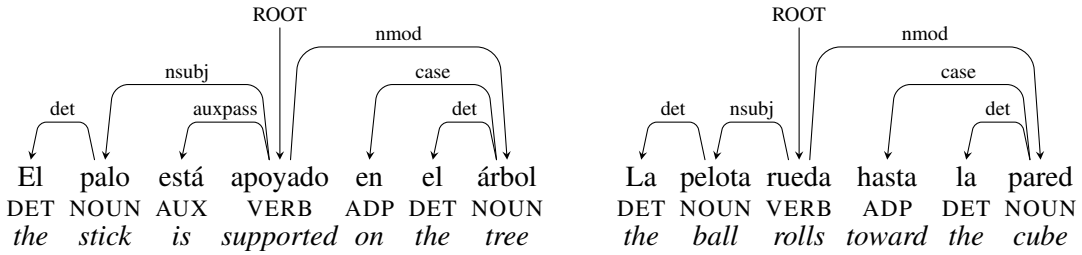
are high-resource languages because we needed to have access to linguistically trained native speakers in order to create the gold standard; however, we did not actually use any additional resources for these languages in our experiments, and we believe they can therefore stand in for low-resource languages.

	# sents	Avg. len.	# words	# lemmas	# pos tags	# features	# labels
English	163	7.21	152	135	12	26	20
Spanish	165	8.51	180	149	12	30	20
German	157	7.52	217	175	13	32	19
Arabic	158	10.04	174	117	10	37	15

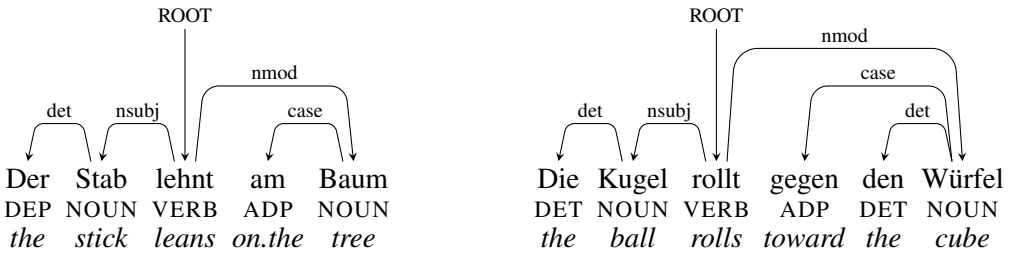
Table 1: Summary of each language in the corpus



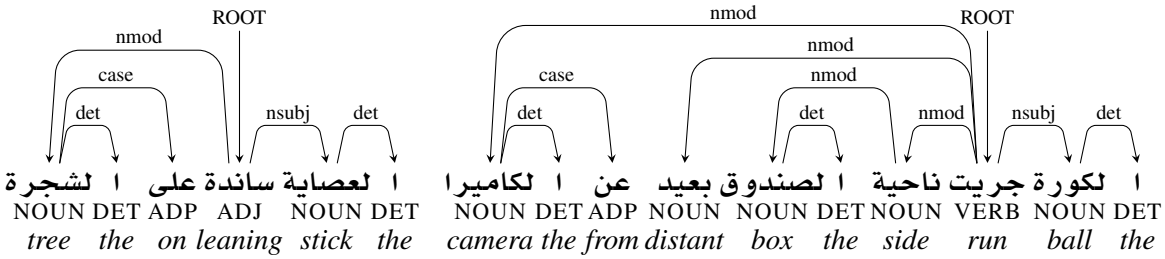
(a) English



(b) Spanish



(c) German



(d) Arabic

Figure 1: Example dependency structures

We start out by tokenizing each sentence; for Spanish and Arabic, this step includes splitting off the clitics. We then annotated each token with its lemma, morphological information, part of speech, syntactic head, and dependency label. For consistency across languages, we used part of speech tags, morphological features, and dependency labels from the Universal Dependencies project (Nivre et al., 2016) and attempted to follow the universal guidelines as closely as possible. The total number of sentences, aver-

age sentence length, and number of unique words, lemmas, part of speech tags, morphological features, and dependency labels for each language is shown in Table 1. Note the sentence count varies slightly for each language; this is because for some of the pictures and videos, the native informant gave us several possible descriptions. English and German have very similar average sentence lengths; average lengths in Spanish and Arabic are higher. German had the largest vocabulary; English had the smallest vocabulary. All languages used similar numbers of part of speech tags and dependency labels, except Arabic which used fewer of both. Arabic had the largest number of morphological features, and English the smallest. Some example sentences with dependency labels are shown in Figure 1.

4 Parsing Experiments and Results

We used four methods of training a dependency parser on our data: MSTParser (McDonald et al., 2006), two configurations of MaltParser (Nivre et al., 2006), and a baseline. All experiments used 5-fold cross validation. For each of the four training methods, we trained on a subset of the train fold ranging from 1 sentence to 100 sentences. We tested on the full test fold, and then averaged the accuracy across the five folds. Results are shown in Table 2. Arc accuracy requires selecting the correct head for a token; Lbl accuracy requires selecting the correct dependency label; Both requires that both head and dependency label are correct. The highest accuracy in each row for each metric (Arc, Lbl, Both) is shown in bold.

The baseline is determined by assigning the majority dependency label from the train data. Heads are selected using left or right attachment, whichever is more common in the train data. For most of the training sets, we did left attachment and assigned `det` as the dependency label, and the baseline usually remained constant across all train sizes. For German with train size = 2, one of the folds had a majority of right attachment, which resulted in a slight decrease in baseline accuracy. Likewise, for Arabic with train size = 1 and train size = 2, one fold used right attachment, resulting in a decrease in arc accuracy for those rows. The Arabic label accuracy was much more variable, since `nmod` was the majority label about half of the time. The Arabic baseline used for each train size and train fold is shown in Table 3.

The first parser we tested was MSTParser (McDonald et al., 2006; McDonald et al., 2005), which uses a two-stage approach to parsing: an unlabeled parser and a separate edge labeler. The parser works by finding a maximum spanning tree; the label sequence is then found using Viterbi's algorithm. MSTParser uses a combination of lexical, part of speech, and morphological features; we did not modify the default feature set.

Train size	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
1	det	det	nmod **(right)**	nmod	det
2	det	det	nmod **(right)**	det	nmod
5	det	det	det	nmod	nmod
10	nmod	det	det	det	nmod
25	nmod	det	nmod	nmod	nmod
50	nmod	det	det	nmod	nmod
100	det	det	det	nmod	nmod

Table 3: Arabic baseline: majority label per fold; if not otherwise indicated, the default attachment is left

We next tested MaltParser (Nivre et al., 2006), which implements a variety of deterministic parsing algorithms. A dependency structure is derived using features of the current parser state to predict the next action. Parser state is represented by a stack of partially processed tokens and a list of remaining input tokens. We tested two algorithms: Nivre arc-eager and Nivre arc-standard. The arc-eager algorithm adds arcs to the dependency tree as soon as the head and dependent are available; the arc-standard algorithm requires that the dependent already be complete with respect to its own dependents. We used the default feature sets for each of the algorithms. Like MSTParser, the feature set includes a combination of lexical, part of speech, and morphological features; MaltParser also adds dependency features (arcs and labels) from the current parser state.

Even with only one training sentence, both MSTParser and MaltParser performed well above the baseline. MaltParser consistently achieved higher accuracy than MSTParser for all languages and train sizes, especially when predicting the dependency labels. The performance of the arc-eager algorithm vs.

Train size	Baseline			MSTParser			MaltParser (Nivre arc-eager)			MaltParser (Nivre arc-standard)		
	Arc	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**
1	0.452	0.325	0.318	0.669	0.495	0.437	0.720	0.785	0.699	**0.741**	**0.793**	**0.708**
2	0.452	0.325	0.318	0.730	0.702	0.643	0.798	**0.831**	**0.769**	**0.801**	0.826	0.764
5	0.452	0.325	0.318	0.794	0.780	0.723	**0.852**	**0.860**	**0.822**	0.817	0.843	0.789
10	0.452	0.325	0.318	0.826	0.787	0.743	**0.872**	**0.880**	**0.846**	0.829	0.856	0.804
25	0.452	0.325	0.318	0.872	0.830	0.798	**0.935**	**0.925**	**0.902**	0.878	0.901	0.855
50	0.452	0.325	0.318	0.930	0.884	0.865	**0.950**	**0.942**	**0.919**	0.920	0.925	0.897
100	0.452	0.325	0.318	0.939	0.913	0.896	**0.961**	**0.965**	**0.946**	0.945	0.951	0.926

(a) English

Train size	Baseline			MSTParser			MaltParser (Nivre arc-eager)			MaltParser (Nivre arc-standard)		
	Arc	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**
1	0.397	0.273	0.271	0.454	0.329	0.311	0.504	0.570	0.478	**0.533**	**0.588**	**0.493**
2	0.397	0.273	0.271	0.568	0.444	0.397	0.600	0.650	**0.558**	**0.605**	**0.663**	**0.558**
5	0.397	0.273	0.271	0.662	0.608	0.541	**0.753**	0.779	**0.713**	0.752	**0.786**	0.702
10	0.397	0.273	0.271	0.758	0.729	0.662	0.797	0.836	0.773	**0.810**	**0.838**	**0.777**
25	0.397	0.273	0.271	0.837	0.813	0.770	**0.890**	**0.905**	**0.865**	0.868	0.887	0.843
50	0.397	0.273	0.271	0.880	0.861	0.817	**0.921**	**0.937**	**0.905**	0.910	0.930	0.895
100	0.397	0.273	0.271	0.923	0.898	0.871	**0.947**	**0.959**	**0.935**	0.932	0.947	0.919

(b) Spanish

Train size	Baseline			MSTParser			MaltParser (Nivre arc-eager)			MaltParser (Nivre arc-standard)		
	Arc	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**
1	0.446	0.286	0.273	0.575	0.407	0.353	0.643	0.680	0.594	**0.655**	**0.685**	**0.595**
2	0.446	0.269	0.234	0.631	0.494	0.435	0.737	0.738	0.657	**0.749**	**0.770**	**0.676**
5	0.446	0.286	0.273	0.753	0.634	0.585	**0.794**	0.800	**0.732**	0.781	**0.820**	0.730
10	0.446	0.286	0.273	0.770	0.676	0.634	**0.819**	**0.848**	**0.782**	0.810	0.844	0.770
25	0.446	0.286	0.273	0.820	0.751	0.707	**0.883**	**0.896**	**0.854**	0.836	0.895	0.819
50	0.446	0.286	0.273	0.850	0.797	0.758	**0.914**	**0.936**	**0.899**	0.896	0.935	0.879
100	0.446	0.286	0.273	0.908	0.845	0.816	**0.942**	0.953	**0.931**	0.916	**0.954**	0.908

(c) German

Train size	Baseline			MSTParser			MaltParser (Nivre arc-eager)			MaltParser (Nivre arc-standard)		
	Arc	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**
1	0.358	0.253	0.181	0.623	0.491	0.434	**0.650**	**0.707**	**0.611**	0.617	0.681	0.571
2	0.358	0.254	0.184	**0.712**	0.656	0.598	0.704	0.738	0.646	0.687	**0.760**	**0.654**
5	0.424	0.237	0.171	0.787	0.747	0.694	0.842	0.861	0.799	**0.844**	**0.871**	**0.811**
10	0.424	0.235	0.168	0.864	0.808	0.768	0.902	0.907	0.869	**0.906**	**0.923**	**0.882**
25	0.424	0.194	0.062	0.920	0.858	0.827	**0.941**	**0.939**	**0.917**	0.930	0.938	0.909
50	0.424	0.216	0.114	0.948	0.888	0.869	0.954	0.958	0.938	**0.957**	**0.965**	**0.941**
100	0.424	0.237	0.171	0.962	0.912	0.897	**0.975**	0.972	**0.961**	0.973	**0.973**	0.957

(d) Arabic

Table 2: Accuracy of each parsing method.

the arc-standard algorithm seems to vary by language and train size. For English, Spanish, and German, the arc-standard algorithm has higher performance on small training sets, while the arc-eager algorithm becomes superior as more training data is available. Results are more mixed for the Arabic data.

5 Initial Experiments with Nonlexical Models

One of the goals of WELT will be to encourage sharing of data and analyses among field linguists. Since stimulus packs (such as the picture series and video series that we used to create our corpus) are commonly reused across many languages, it would be helpful if a parser trained on a fully-annotated version of the data for one language could be used by a field linguist just starting out with another, potentially similar, language. To that end, we performed an initial experiment to see whether a parser trained on one language could be applied successfully to the other languages in our corpus. To test this, we used MaltParser (arc-eager algorithm) to train a parser on English data, using only nonlexical features: part of speech, morphological tags, and dependency labels/arcs. We then applied this model to the other three languages. Results are shown in Table 4. For this experiment, we used all 163 sentences from the English corpus.

Comparing these results to those in Table 2, we see that, for Spanish, the English nonlexical model performs similarly to MaltParser trained on 5 Spanish sentences. For German, the English nonlexical model performs similarly to MaltParser trained on 5-10 German sentences. For Arabic, the English nonlexical model has lower accuracy than MaltParser trained on a single Arabic sentence. This suggests that a simple nonlexical model such as this one may only be useful for linguists using WELT if an annotated corpus in a *related* language is available.

	Arc	Label	Arc+Label
Spanish	0.767	0.816	0.719
German	0.808	0.840	0.773
Arabic	0.629	0.713	0.567

Table 4: Accuracy of (English) nonlexical model applied to other languages

6 Annotation Experiments and Results

To test whether our parser would help with annotation, we performed annotation experiments using the English data. We timed how long it took an annotator to label a sentence, when the sentence is preprocessed in one of four ways. In the first method, a baseline assigns a flat structure and the dependency label `det` to all nodes. The other methods use MaltParser (Nivre arc-eager algorithm) trained on 1, 5, or 25 sentences to provide an initial parse for the annotator to correct.

Annotators labeled 5 trees for each parsing method, for a total of 20 trees. To ensure each of the four sets of 5 contained sentences with similar syntactic complexity, the sentences were chosen as follows. Each parsing method was assigned 1 sentence of each of 5 lengths: 7 words, 8 words, 9 words, 10-11 words, and 12-14 words. These were randomly selected from among all sentences of the required length. The 20 sentences were then presented to the annotator in random order. To keep the experiment consistent, all annotators labeled the same 20 sentences, in the same order.

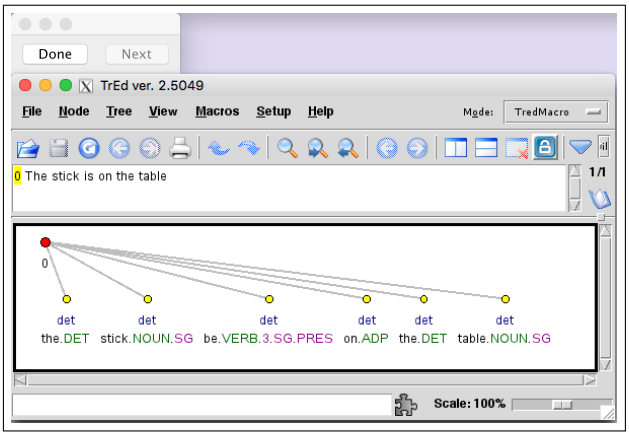


Figure 2: Screenshot of annotation software

Three annotators participated in the experiment. The first was the first author of this paper. She is an expert annotator, very familiar with the universal guidelines for dependency annotation and the annotation software. The other two annotators were undergraduate students who participated in a brief training session to familiarize them with the desired analysis and the software. They were given ref-

erence materials showing sample trees covering a variety of syntactic phenomena including: auxiliaries, copulas, coordination, secondary predication, and subordinate clauses. They were permitted to refer to this material throughout the annotation task. Before participating in the annotation task, they annotated 10 additional trees for practice.

The software used for annotation was Tree Editor (TrEd) (Pajas and Štěpánek, 2008) with a simple Java wrapper that handled opening files in TrEd and keeping track of annotation time. A screenshot of the setup is shown in Figure 2. Upon pressing the Next button, the annotator was shown the next tree in TrEd and the program recorded the start time. When the annotator finished labeling a tree, they saved the file in TrEd and pressed the Done button. The wrapper program closed the current file in TrEd and recorded the end time.

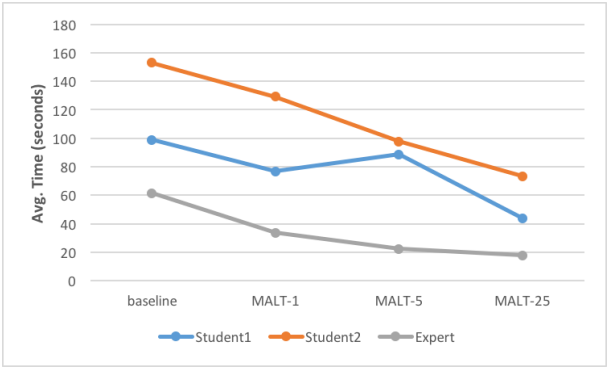


Figure 3: Annotation time across all sentence sizes

A graph showing the average time (across all sentence lengths) it took each annotator to label a tree for each of the four parsing types is shown in Figure 3. A detailed listing of the time each annotator took for each sentence size is shown in Table 5. A graph showing the average time (across annotators) for each sentence size is shown in Figure 4(a). All times are given in seconds. The accuracy of MaltParser (arc-eager) on sentences of different lengths is shown in Figure 4(b).

Sent. size	Baseline	Malt-1	Malt-5	Malt-25
7	87.0	54.9	14.4	14.3
8	73.0	83.6	65.1	17.7
9	85.9	67.2	61.7	43.3
10-11	144.3	70.6	65.7	86.6
12-14	104.5	107.5	237.1	57.3
Avg	98.9	76.8	88.8	43.8

(a) Student1

Sent. size	Baseline	Malt-1	Malt-5	Malt-25
7	132.7	54.0	23.6	26.3
8	97.8	98.9	24.3	30.4
9	90.8	153.0	96.9	102.4
10-11	203.4	66.2	139.9	138.9
12-14	240.8	274.1	203.7	68.7
Avg	153.1	129.2	97.7	73.4

(b) Student2

Sent. size	Baseline	Malt-1	Malt-5	Malt-25
7	59.2	25.4	8.1	8.7
8	44.7	42.0	8.6	9.2
9	52.3	29.7	21.4	21.0
10-11	75.8	34.9	11.2	36.8
12-14	75.5	36.0	62.0	12.8
Avg	61.5	33.6	22.3	17.7

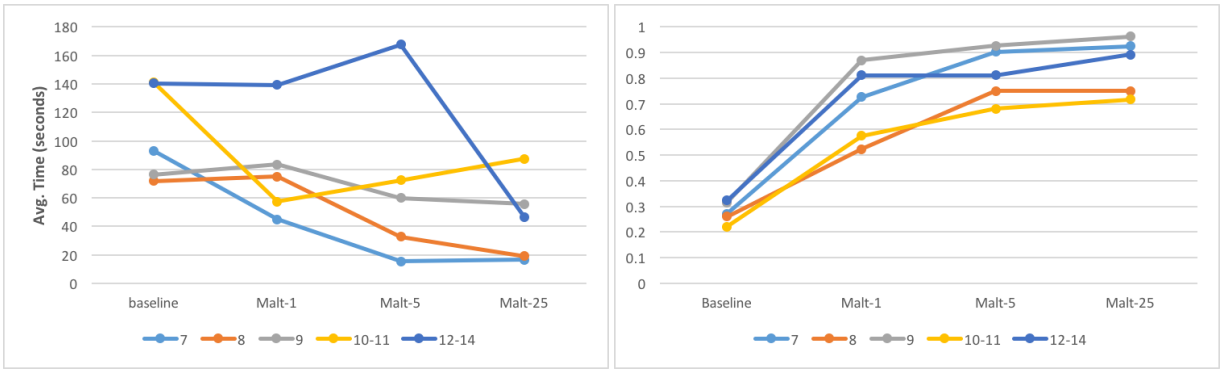
(c) Expert

Sent. size	Baseline	Malt-1	Malt-5	Malt-25
7	93.0	44.8	15.4	16.5
8	71.8	74.8	32.7	19.1
9	76.3	83.3	60.0	55.6
10-11	141.1	57.3	72.3	87.5
12-14	140.3	139.2	167.6	46.3
Avg	104.5	79.9	69.6	45.0

(d) Average

Table 5: Time (seconds) to annotate each sentence

Results vary slightly across annotators, but it is clear that, even when training on a single sentence, annotation time is improved. Average annotation time decreases from 104.5 seconds for the baseline parse to 79.9 seconds for the parser trained on one sentence. Using the parser trained on 5 sentences, average annotation time decreases again to 69.6 seconds. Using the parser trained 25 sentences, we see a decrease in annotation time to an average of 45 seconds. Statistical significance testing was done with a paired t-test. Significant decreases in annotation time are: between the baseline and 1 training sentence ($p = 0.034$), between the baseline and 5 training sentences ($p = 0.015$), between the baseline



(a) (b)

Figure 4: Graphs of (a) annotation time for each sentence size, across all annotators (b) parser accuracy on each sentence size

and 25 training sentences ($p = 2.51\text{e}{-5}$), and between 1 training sentence and 25 training sentences ($p = 0.018$).

There are several reasons to account for the fact that training on a single sentence significantly decreases annotation time. MaltParser works by predicting steps in a derivation, so one sentence actually translates into more than one data point. With only one sentence, the parser can learn that a determiner should be the left child of a noun, or that a noun should be the left child of the root predicate. Having these dependencies already attached reduces the work the annotator must do compared to a completely flat structure. In addition, our corpus consists only of descriptions of spatial relations and motion events, so we expect a much more limited range of grammatical constructs than one finds in other treebanks.

For very short sentences (length 7), the graph in Figure 4(b) shows a clear downward trend as the amount of training data increases. For sentences of length 8-9, we do not see improvement with one training sentence, but annotation time begins to decrease substantially when there are 5 training sentences. For longer sentences, the downward trend is less clear. This makes sense, since we can expect to find a wider range of syntactic structures in a longer sentence, and parser performance on these will require that a similar structure was seen in the train set. For sentences length 10-11, there is a substantial drop in annotation time from baseline to 1 training sentence, at which point it seems to plateau. For sentences length 12-14, average annotation time does not decrease until we have 25 training sentences.

Parser	**Student1**			**Student2**			**Expert**		
	Arc	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**	**Arc**	**Lbl**	**Both**
Baseline	0.951	1.000	0.951	1.000	1.000	1.000	1.000	1.000	1.000
Malt-1	0.971	1.000	0.971	1.000	1.000	1.000	1.000	1.000	1.000
Malt-5	0.930	0.950	0.905	0.980	1.000	0.980	0.980	1.000	0.980
Malt-25	0.962	0.982	0.962	1.000	1.000	1.000	1.000	1.000	1.000

Table 6: Annotation accuracy

One concern with using a parser to assist with annotation is whether there will be any effect on overall accuracy. When presented with a mostly-correct parse, will annotators be able to see all the errors and fix them? The accuracy of the annotators for each of the four parsing types is shown in Table 6. We do see a drop in accuracy for all annotators when training on 5 sentences, especially for Student1. However, this decrease is very slight for both Student2 and Expert. We suspect there may have been several difficult sentences in this set; all annotators made errors on the sentence of length 10, and Student1 had particularly low accuracy (0.625) on the sentence of length 8.

7 Summary and Future Work

We have reported results for incrementally training a dependency parser across four languages. Our results show that such a parser can improve on baseline performance even when trained on a single

sentence, making our method particularly useful in the documentation of endangered and low-resource languages. We found that MaltParser achieved the highest accuracy overall; the arc-standard algorithm seems preferable for very small training sizes and arc-eager for slightly larger training sizes. We found that using a parser to predict each sentence before annotation did significantly improve annotation time, without a substantial decrease in accuracy.

The results of our experiments demonstrate that it is possible to extend FieldWorks' "stealthy" approach to learning a morphological parser into the realm of syntax. By providing a way to assign dependency structures to sentences, WELT will allow field linguists to incorporate syntax into language documentation. The incrementally trained parser will reduce their workload by letting them correct errors in a dependency structure rather than starting from scratch. This method of syntactic documentation does not limit the field linguist to a particular syntactic theory. We chose to use the universal labels and analyses in our corpus, but WELT users will have complete control over assignment of heads and choice of dependency labels. The only requirement is that they are consistent across sentences.

In future work, we will experiment with other parsers, such as TurboParser (Martins et al., 2010), Mate (Bohnet, 2010), and Easy-First (Goldberg and Elhadad, 2010). Furthermore, we will continue to investigate methods of re-using existing parsers and dependency annotations with new languages (see Section 5); specifically, we will investigate more effective methods of adapting existing parsers to other languages. For example, we will investigate how to combine a non-lexical model with a lexical model obtained from a small number of target language sentences. We will also investigate ways for linguists to directly specify syntactic properties that can be used by the parser, similar to the way FLEx converts morphological properties specified by users into formal rules compatible with the underlying parser. Finally, we plan to try our WELT system in actual fieldwork.

Acknowledgements

We would like to thank Bob Coyne, Victor Soto, Daniel Bauer, and Heba Elfardy for their help creating and annotating the parallel corpus. We would like to thank the anonymous reviewers for their comments that helped us improve the paper.

References

Alliance for Linguistic Diversity. 2013. The Endangered Languages Project. `http://www.endangeredlanguages.com/`.

F. Ameka, C. de Witte, and D. Wilkins. 1999. Picture series for positional verbs: Eliciting the verbal component in locative descriptions. In D. Wilkins, editor, *Manual for the 1999 Field Season*, pages 48–54. Max Planck Institute for Psycholinguistics.

E. Bender, D. Flickinger, and S. Oepen. 2002. The Grammar Matrix. In J. Carroll, N. Oostdijk, and R. Sutcliffe, editors, *Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan.

S. Bird. 2009. Natural language processing and linguistic fieldwork. *Computational Linguistics*, 35(3):469–474.

H.A. Black and G.F. Simons. 2006. The SIL FieldWorks Language Explorer approach to morphological parsing. In *Computational Linguistics for Less-studied Languages: Texas Linguistics Society 10*, Austin, TX, November.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 89–97, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164.

D. Crouch, M. Dalrymple, R. Kaplan, T. King, J. Maxwell, and P. Newman. 2011. XLE Documentation. `http://www2.parc.com/isl/groups/nltt/xle/doc/xle_toc.html`.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. A neural network model for low-resource universal dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 339–348, Lisbon, Portugal, September. Association for Computational Linguistics.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 742–750, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1234–1244, Beijing, China, July. Association for Computational Linguistics.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1058–1063, San Diego, California, June. Association for Computational Linguistics.

Stephen C. Levinson. 2001. Motion verb stimul, version 2. In Stephen C. Levinson and N. J. Enfield, editors, *Manual for the field season 2001*, pages 9–13. Max Planck Institute for Psycholinguistics.

André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 34–44, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 523–530, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 216–220, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proc. of LREC-2006*.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.

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

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Petr Pajas and Jan Štěpánek. 2008. Recent advances in a feature-rich framework for treebank annotation. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 673–680, Stroudsburg, PA, USA. Association for Computational Linguistics.

ParGram. 2013. ParGram/ParSem. `http://pargram.b.uib.no/`.

SIL FieldWorks. 2014. SIL FieldWorks. `http://fieldworks.sil.org`.

Morgan Ulinski, Anusha Balakrishnan, Daniel Bauer, Bob Coyne, Julia Hirschberg, and Owen Rambow. 2014a. Documenting endangered languages with the WordsEye Linguistics Tool. In *Proceedings of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 6–14.

Morgan Ulinski, Anusha Balakrishnan, Bob Coyne, Julia Hirschberg, and Owen Rambow. 2014b. WELT: Using graphics generation in linguistic fieldwork. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 49–54.

Inducing Multilingual Text Analysis Tools Using Bidirectional Recurrent Neural Networks

Othman Zennaki
CEA, LIST, LVIC
Gif-sur-Yvette, France
othman.zennaki@cea.fr

Nasredine Semmar
CEA, LIST, LVIC
Gif-sur-Yvette, France
nasredine.semmar@cea.fr

Laurent Besacier
LIG, Univ. Grenoble-Alpes
Grenoble, France
laurent.besacier@imag.fr

Abstract

This work focuses on the rapid development of linguistic annotation tools for resource-poor languages. We experiment several cross-lingual annotation projection methods using Recurrent Neural Networks (RNN) models. The distinctive feature of our approach is that our multilingual word representation requires only a parallel corpus between source and target languages. More precisely, our method has the following characteristics: (a) it does not use word alignment information, (b) it does not assume any knowledge about foreign languages, which makes it applicable to a wide range of resource-poor languages, (c) it provides truly multilingual taggers. We investigate both uni- and bi-directional RNN models and propose a method to include external information (for instance low level information from Part-Of-Speech tags) in the RNN to train higher level taggers (for instance, super sense taggers). We demonstrate the validity and genericity of our model by using parallel corpora (obtained by manual or automatic translation). Our experiments are conducted to induce cross-lingual POS and super sense taggers.

1 Introduction

In order to minimize the need for annotated resources (produced through manual annotation, or by manual check of automatic annotation), several research works were interested in building Natural Language Processing (NLP) tools based on unsupervised or semi-supervised approaches (Collins and Singer, 1999; Klein, 2005; Goldberg, 2010). For example, NLP tools based on cross-language projection of linguistic annotations achieved good performances in the early 2000s (Yarowsky et al., 2001). The key idea of annotation projection can be summarized as follows: through word alignment in parallel text corpora, the annotations are transferred from the *source* (resource-rich) language to the *target* (under-resourced) language, and the resulting annotations are used for supervised training in the target language. However, automatic word alignment errors (Fraser and Marcu, 2007) limit the performance of these approaches.

Our work is built upon these previous contributions and observations. We explore the possibility of using Recurrent Neural Networks (RNN) to build multilingual NLP tools for resource-poor languages analysis. The major difference with previous works is that we do not explicitly use word alignment information. Our only assumption is that parallel sentences (source-target) are available and that the source part is annotated. In other words, we try to infer annotations in the target language from sentence-based alignments only. While most NLP researches on RNN have focused on monolingual tasks[1] and sequence labeling (Collobert et al., 2011; Graves, 2012), this paper, however, considers the problem of learning multilingual NLP tools using RNN.

Contributions In this paper, we investigate the effectiveness of RNN architectures — Simple RNN (SRNN) and Bidirectional RNN (BRNN) — for multilingual sequence labeling tasks without using any word alignment information. Two NLP tasks are considered: Part-Of-Speech (POS) tagging and Super Sense (SST) tagging (Ciaramita and Altun, 2006). Our RNN architectures demonstrate very competitive results on unsupervised training for new target languages. In addition, we show that the integration of

[1]Exceptions are the recent propositions on Neural Machine Translation (Cho et al., 2014; Sutskever et al., 2014)

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 450–460, Osaka, Japan, December 11-17 2016.

POS information in RNN models is useful to build multilingual coarse-grain semantic (Super Senses) taggers. For this, a simple and efficient way to take into account low-level linguistic information for more complex sequence labeling RNN is proposed.

Methodology For training our multilingual RNN models, we just need as input a parallel (or multi-parallel) corpus between a resource-rich language and one or many under-resourced languages. Such a parallel corpus can be manually obtained (clean corpus) or automatically obtained (noisy corpus).

To show the potential of our approach, we investigate two sequence labeling tasks: cross-language POS tagging and multilingual Super Sense Tagging (SST). For the SST task, we measure the impact of the parallel corpus quality with manual or automatic translations of the SemCor (Miller et al., 1993) translated from English into Italian (manually and automatically) and French (automatically).

Outline The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 describes our cross-language annotation projection approaches based on RNN. Section 4 presents the empirical study and associated results. We finally conclude the paper in Section 5.

2 Related Work

Cross-lingual projection of linguistic annotations was pioneered by Yarowsky et al. (2001) who created new monolingual resources by transferring annotations from resource-rich languages onto resource-poor languages through the use of word alignments. The resulting (noisy) annotations are used in conjunction with robust learning algorithms to build cheap unsupervised NLP tools (Padó and Lapata, 2009). This approach has been successfully used to transfer several linguistic annotations between languages (efficient learning of POS taggers (Das and Petrov, 2011; Duong et al., 2013) and accurate projection of word senses (Bentivogli et al., 2004)). Cross-lingual projection requires a parallel corpus and word alignment between source and target languages. Many automatic word alignment tools are available, such as GIZA++ which implements IBM models (Och and Ney, 2000). However, the noisy (non perfect) outputs of these methods is a serious limitation for the annotation projection based on word alignments (Fraser and Marcu, 2007).

To deal with this limitation, recent studies based on cross-lingual representation learning methods have been proposed to avoid using such pre-processed and noisy alignments for label projection. First, these approaches learn language-independent features, across many different languages (Durrett et al., 2012; Al-Rfou et al., 2013; Täckström et al., 2013; Luong et al., 2015; Gouws and Søgaard, 2015; Gouws et al., 2015). Then, the induced representation space is used to train NLP tools by exploiting labeled data from the source language and apply them in the target language. Cross-lingual representation learning approaches have achieved good results in different NLP applications such as cross-language SST and POS tagging (Gouws and Søgaard, 2015), cross-language named entity recognition (Täckström et al., 2012), cross-lingual document classification and lexical translation task (Gouws et al., 2015), cross language dependency parsing (Durrett et al., 2012; Täckström et al., 2013) and cross-language semantic role labeling (Titov and Klementiev, 2012).

Our approach described in next section, is inspired by these works since we also try to induce a common language-independent feature space (crosslingual words embeddings). Unlike Durrett et al. (2012) and Gouws and Søgaard (2015), who use bilingual lexicons, and unlike Luong et al. (2015) who use word alignments between the source and target languages[2] our common multilingual representation is very agnostic. We use a simple (multilingual) vector representation based on the occurrence of source and target words in a parallel corpus and we let the RNN learn the best internal representations (corresponding to the hidden layers) specific to the task (SST or POS tagging).

In this work, we learn a cross-lingual POS tagger (multilingual POS tagger if a multilingual parallel corpus is used) based on a recurrent neural network (RNN) on the source labeled text and apply it to tag target language text. We explore simple and bidirectional RNN architectures (SRNN and BRNN respectively). Starting from the intuition that low-level linguistic information is useful to learn more complex taggers, we also introduce three new RNN variants to take into account external (POS) information in multilingual SST.

[2]to train a bilingual representation regardless of the task

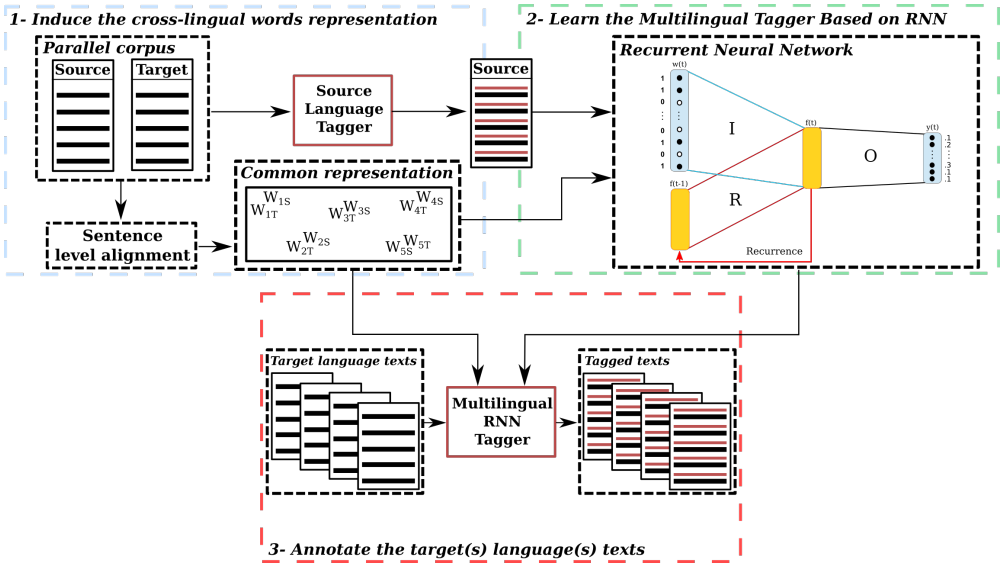


Figure 1: Overview of the proposed model architecture for inducing multilingual RNN taggers.

3 Unsupervised Approach Overview

To avoid projecting label information from deterministic and error-prone word alignments, we propose to represent the word alignment information intrinsically in a recurrent neural network architecture. The idea consists in implementing a recurrent neural network as a multilingual sequence labeling tool (we investigate POS tagging and SST tagging). Before describing our cross-lingual (multilingual if a multi-parallel corpus is used) neural network tagger, we present the simple cross-lingual projection method, considered as our baseline in this work.

3.1 Baseline Cross-lingual Annotation Projection

We use direct transfer as a baseline system which is similar to the method described in (Yarowsky et al., 2001). First we tag the source side of the parallel corpus using the available supervised tagger. Next, we align words in the parallel corpus to find out corresponding source and target words. Tags are then projected to the (resource-poor) target language. The target language tagger is trained using any machine learning approach (we use TnT tagger (Brants, 2000) in our experiments).

3.2 Proposed Approach

We propose a method for learning multilingual sequence labeling tools based on RNN, as it can be seen in Figure 1. In our approach, a parallel or multi-parallel corpus between a resource-rich language and one or many under-resourced languages is used to extract common (multilingual) and agnostic words representations. These representations, which rely on sentence level alignment only, are used with the source side of the parallel/multi-parallel corpus to learn a neural network tagger in the source language. Since a common representation of source and target words is chosen, this neural network tagger is truly multilingual and can be also used to tag texts in target language(s).

3.2.1 Common Words Representation

In our *agnostic* representation, we associate to each word (in source *and* target vocabularies) a common vector representation, namely $V_{wi}, i = 1, ..., N$, where N is the number of parallel sentences (bi-sentences in the parallel corpus). If w appears in i-th bi-sentence of the parallel corpus then $V_{wi} = 1$.

The idea is that, in general, a source word and its target translation appear together in the same bi-sentences and their vector representations are close. We can then use the RNN tagger, initially trained on source side, to tag the target side (because of our *common vector representation*). This simple representation does not require multilingual word alignments and it lets the RNN learns the optimal internal representation needed for the annotation task (for instance, the hidden layers of the RNN can be considered as multi-lingual embeddings of the words).

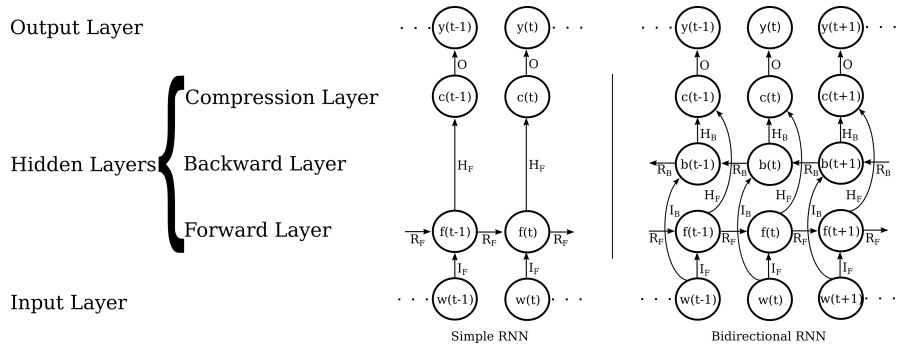


Figure 2: High level schema of RNN used in our work.

3.2.2 Recurrent Neural Networks

There are two major architectures of neural networks: Feedforward (Bengio et al., 2003) and Recurrent Neural Networks (RNN) (Schmidhuber, 1992; Mikolov et al., 2010). Sundermeyer et al. (2013) showed that language models based on recurrent architecture achieve better performance than language models based on feedforward architecture. This is due to the fact that recurrent neural networks do not use a context of limited size. This property led us to use, in our experiments, the Elman recurrent architecture (Elman, 1990), in which recurrent connections occur at the hidden layer level.

We consider in this work two Elman RNN architectures (see Figure 2): *Simple* RNN (SRNN) and *Bidirectional* RNN (BRNN). In addition, to be able to include low-level linguistic information in our architecture designed for more complex sequence labeling tasks, we propose three new RNN variants to take into account external (POS) information for multilingual Super Sense Tagging (SST).

A. Simple RNN

In the *simple* Elman RNN (SRNN), the recurrent connection is a loop at the hidden layer level. This connection allows SRNN to use at the current time step hidden layer's states of previous time steps. In other words, the hidden layer of SRNN represents all previous history and not just $n-1$ previous inputs, thus the model can theoretically represent long context.

The architecture of the SRNN considered in this work is shown in Figure 2. In this architecture, we have 4 layers: input layer, forward (also called recurrent or context layer), compression hidden layer and output layer. All neurons of the input layer are connected to every neuron of forward layer by weight matrix I_F and R_F, the weight matrix H_F connects all neurons of the forward layer to every neuron of compression layer and all neurons of the compression layer are connected to every neuron of output layer by weight matrix O.

The input layer consists of a vector $w(t)$ that represents the current word w_t in our common words representation (all input neurons corresponding to current word w_t are set to 0 except those that correspond to bi-sentences containing w_t, which are set to 1), and of vector $f(t-1)$ that represents output values in the forward layer from the previous time step. We name $f(t)$ and $c(t)$ the current time step hidden layers (our preliminary experiments have shown better performance using these two hidden layers instead of one hidden layer), with variable sizes (usually 80-1024 neurons) and sigmoid activation function. These hidden layers represent our common language-independent feature space and inherently capture word alignment information. The output layer $y(t)$, given the input $w(t)$ and $f(t-1)$ is computed with the following steps :

$$f(t) = \Sigma(w(t).I_F(t) + f(t-1).R_F(t)) \quad (1)$$

$$c(t) = \Sigma(f(t).H_F(t)) \quad (2)$$

$$y(t) = \Gamma(c(t).O(t)) \quad (3)$$

Σ and Γ are the sigmoid and the softmax functions, respectively. The softmax activation function is used to normalize the values of output neurons to sum up to 1. After the network is trained, the output $y(t)$ is a vector representing a probability distribution over the set of tags. The current word w_t (in input) is tagged with the most probable output tag.

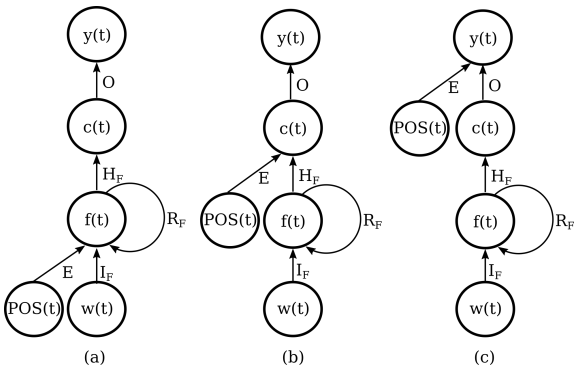


Figure 3: SRNN variants with POS information at three levels: (a) input layer, (b) forward layer, (c) compression layer.

For many sequence labeling tasks, it is beneficial to have access to future in addition to the past context. So, it can be argued that our SRNN is not optimal for sequence labeling, since the network ignores future context and tries to optimize the output prediction given the previous context only. This SRNN is thus penalized compared with our baseline projection based on TnT (Brants, 2000) which considers both left and right contexts. To overcome the limitations of SRNN, a simple extension of the SRNN architecture — namely Bidirectional recurrent neural network (BRNN) (Schuster and Paliwal, 1997) — is used to ensure that context at previous and future time steps will be considered.

B. Bidirectional RNN

An unfolded BRNN architecture is given in Figure 2. The basic idea of BRNN is to present each training sequence forwards and backwards to two separate recurrent hidden layers (forward and backward hidden layers) and then somehow merge the results. This structure provides the compression and the output layers with complete past and future context for every point in the input sequence. Note that without the backward layer, this structure simplifies to a SRNN.

C. RNN Variants

As mentioned in the introduction, we propose three new RNN variants to take into account low level (POS) information in a higher level (SST) annotation task. The question addressed here is: at which layer of the RNN this low level information should be included to improve SST performance? As specified in Figure 3, the POS information can be introduced either at input layer or at forward layer (forward and backward layers for BRNN) or at compression layer. In all these RNN variants, the POS of the current word is also represented with a vector ($POS(t)$). Its dimension corresponds to the number of POS tags in the tagset (universal tagset of Petrov et al. (2012) is used). We propose one *hot* vector representation where only one value is set to 1 and corresponds to the index of current tag (all other values are 0).

3.2.3 Network Training

The first step in our approach is to train the neural network, given a parallel corpus (training corpus), and a validation corpus (different from train data) in the source language. In typical applications, the source language is a resource-rich language (which already has an efficient tagger or manually tagged resources). Our RNN models are trained by stochastic gradient descent using usual back-propagation and back-propagation through time algorithms (Rumelhart et al., 1985). We learn our RNN models with an iterative process on the tagged source side of the parallel corpus. After each epoch (iteration) in training, validation data is used to compute per-token accuracy of the model. After that, if the per-token accuracy increases, training continues in the new epoch. Otherwise, the learning rate is halved at the start of the new epoch. Eventually, if the per-token accuracy does not increase anymore, training is stopped to prevent over-fitting. Generally, convergence takes 5–10 epochs, starting with a learning rate $\alpha = 0.1$.

The second step consists in using the trained model as a target language tagger (using our common

vector representation). It is important to note that if we train on a multilingual parallel corpus with *N* languages ($N > 2$), the same trained model will be able to tag all the *N* languages.

Hence, our approach assumes that the word order in both source and target languages are similar. In some languages such as English and French, word order for contexts containing nouns could be reversed most of the time. For example, the compound word *the European Commission* would be translated into *la Commission européenne*. In order to deal with the word order constraints, we also combine the RNN model with the cross-lingual projection model in our experiments.

3.3 Dealing with out-of-vocabulary words

For the words absent from in the initial parallel corpus, their vector representation is a vector of zero values. Consequently, during testing, the RNN model will use only the context information to tag the OOV words found in the test corpus. To deal with these types of OOV words[3], we use the CBOW model of (Mikolov et al., 2013) to replace each OOV word by its closest known word in the current OOV word context. Once the closest word is found, its common vector representation is used (instead of the vector of zero values) at the input of the RNN.

3.4 Combining Simple Cross-lingual Projection and RNN Models

Since the simple cross-lingual projection model *M1* and RNN model *M2* use different strategies for tagging (TnT is based on Markov models while RNN is a neural network), we assume that these two models can be complementary. To keep the benefits of each approach, we explore how to combine them with linear interpolation. Formally, the probability to tag a given word *w* is computed as

$$P_{M12}(t|w) = (\mu P_{M1}(t|w, C_{M1}) + (1 - \mu)P_{M2}(t|w, C_{M2})) \quad (4)$$

where, C_{M1} and C_{M2} are the context of w considered by *M1* and *M2* respectively. The relative importance of each model is adjusted through the interpolation parameter μ. The word w is tagged with the most probable tag, using the function f described as

$$f(w) = \arg\max_t(P_{M12}(t|w)) \quad (5)$$

4 Experiments

Our models are evaluated on two labeling tasks: Cross-language Part-Of-speech (POS) tagging and Multilingual Super Sense Tagging (SST).

4.1 Multilingual POS Tagging

We applied our method to build RNN POS taggers for four target languages - French, German, Greek and Spanish - with English as the source language.

In order to determine the effectiveness of our common words representation described in section 3.2.1, we also investigated the use of state-of-the-art bilingual word embeddings (using MultiVec Toolkit (Bérard et al., 2016)) as input to our RNN.

4.1.1 Dataset

For French as a target language, we used a training set of $10,000$ parallel sentences, a validation set of 1000 English sentences, and a test set of 1000 French sentences, all extracted from the ARCADE II English-French corpus (Veronis et al., 2008). The test set is tagged with the French *TreeTagger* (Schmid, 1995) and then manually checked.

For German, Greek and Spanish as a target language, we used training and validation data extracted from the Europarl corpus (Koehn, 2005) which are a subset of the training data used in (Das and Petrov, 2011; Duong et al., 2013). This choice allows us to compare our results with those of (Das and Petrov, 2011; Duong et al., 2013; Gouws and Søgaard, 2015). The train data set contains $65,000$ bi-sentences ; a validation set of $10,000$ bi-sentences is also available. For testing, we use the same test corpora as (Das and Petrov, 2011; Duong et al., 2013; Gouws and Søgaard, 2015) (bi-sentences from CoNLL shared

[3] words which do not have a known vector representation

Lang. / Model	French		German		Greek		Spanish	
	All words	OOV	All words	OOV	All words	OOV	All words	OOV
Simple Projection	80.3	77.1	78.9	73.0	77.5	72.8	80.0	79.7
SRNN MultiVec	75.0	65.4	70.3	68.8	71.1	65.4	73.4	62.4
SRNN	78.5	70.0	76.1	76.4	75.7	70.7	78.8	72.6
BRNN	80.6	70.9	77.5	76.6	77.2	71.0	80.5	73.1
BRNN - OOV	81.4	77.8	77.6	77.8	77.9	75.3	80.6	74.7
Projection + SRNN	84.5	78.8	81.5	77.0	78.3	74.6	83.6	81.2
Projection + BRNN	85.2	79.0	81.9	77.1	79.2	75.0	84.4	81.7
Projection + BRNN - OOV	**85.6**	**80.4**	82.1	**78.7**	79.9	**78.5**	**84.4**	**81.9**
(Das, 2011)	—	—	82.8	—	**82.5**	—	84.2	—
(Duong, 2013)	—	—	**85.4**	—	80.4	—	83.3	—
(Gouws, 2015a)	—	—	84.8	—	—	—	82.6	—

Table 1: Token-level POS tagging accuracy for Simple Projection, SRNN using MultiVec bilingual word embeddings as input, RNN[5], Projection+RNN and methods of Das & Petrov (2011), Duong et al (2013) and Gouws & Søgaard (2015).

tasks on dependency parsing (Buchholz and Marsi, 2006)). The evaluation metric (*per-token* accuracy) and the Petrov et al. (2012) *universal tagset* are used for evaluation.

For training, the English (source) sides of the training corpora (ARCADE II and Europarl) and of the validation corpora are tagged with the English *TreeTagger* toolkit. Using the matching provided by Petrov et al. (2012), we map the TreeTagger and the CoNLL tagsets to the common *Universal Tagset*.

In order to build our baseline unsupervised tagger (based on a Simple Cross-lingual Projection – see section 3.1), we also tag the target side of the training corpus, with tags projected from English side through word-alignments established by GIZA++. After tags projection, a target language POS tagger based on TnT approach (Brants, 2000) is trained.

The combined model is built for each considered language using cross-validation on the test corpus. First, the test corpus is split into 2 equal parts and on each part, we estimate the interpolation parameter μ (Equation 4) which maximizes the *per-token* accuracy score. Then each part of test corpus is tagged using the combined model tuned on the other part, and vice versa (standard cross-validation procedure).

We trained MultiVec bilingual word embeddings on the parallel Europarl corpus between English and each of the target languages considered.

4.1.2 Results and discussion

Table 1 reports the results obtained for the unsupervised POS tagging. We note that the POS tagger based on bidirectional RNN (BRNN) has better performance than simple RNN (SRNN), which means that both past and future contexts help select the correct tag. Table 1 also shows the performance before and after performing our procedure for handling OOVs in BRNNs. It is shown that after replacing OOVs by the closest words using CBOW, the tagging accuracy significantly increases.

As shown in the same table, our RNN models accuracy is close to that of the simple projection tagger. It achieves comparable results to Das and Petrov (2011), Duong et al. (2013) (who used the full Europarl corpus while we use only a $65,000$ subset of it) and to Gouws and Søgaard (2015) (who used extra resources such as Wiktionary and Wikipedia). Interestingly, RNN models learned using our common words representation (section 3.2.1) seem to perform significantly better than RNN models using MultiVec bilingual word embeddings.

It is also important to note that only one single SRNN and BRNN tagger applies to German, Greek and Spanish; so this is a truly multilingual POS tagger! Finally, as for several other NLP tasks such as language modelling or machine translation (where standard and NN-based models are generally combined in order to obtain optimal results), the combination of standard and RNN-based approaches (*Projection+_*) seems necessary to further optimize POS tagging accuracies.

[5]For RNN models, only one (same) system is used to tag German, Greek and Spanish

4.2 Multilingual SST

In order to measure the impact of the parallel corpus quality on our method, we also learn our SST models using the multilingual parallel corpus MultiSemCor (MSC) which is the result of manual or automatic translation of SemCor from English into Italian and French.

4.2.1 Dataset

SemCor The SemCor (Miller et al., 1993) is a subset of the Brown Corpus (Kucera and Francis, 1979) labeled with the *WordNet* (Fellbaum, 1998) senses.
MultiSemCor The English-Italian MultiSemcor (MSC-IT-1) corpus is a manual translation of the English SemCor to Italian (Bentivogli et al., 2004). As we already mentioned, we are also interested in measuring the impact of the parallel corpus quality on our method. For this we use two translation systems: (a) Google Translate to translate the English SemCor to Italian (MSC-IT-2) and French (MSC-FR-2). (b) LIG machine translation system (Besacier et al., 2012) to translate the English SemCor to French (MSC-FR-1).
Training corpus The SemCor was labeled with the *WordNet* synsets. However, because we train models for SST, we convert SemCor synsets annotations to super senses. We learn our models using the four different versions of MSC (MSC-IT-1,2 - MSC-FR-1,2), with modified Semcor on source side.
Test Corpus To evaluate our models, we used the SemEval 2013 Task 12 (Multilingual Word Sense Disambiguation) (Navigli et al., 2013) test corpora, which are available in 5 languages (English, French, German, Spanish and Italian) and labeled with *BabelNet* (Navigli and Ponzetto, 2012) senses. We map BabelNet senses to WordNet synsets, then WordNet synsets are mapped to super senses.

4.2.2 SST Systems Evaluated

The goals of our SST experiments are twofold: first, to investigate the effectiveness of using POS information to build multilingual super sense tagger, secondly to measure the impact of the parallel corpus quality (manual or automatic translation) on our RNN models (SRNN, BRNN and our proposed variants). To summarize, we build four super sense taggers based on baseline cross-lingual projection (see section 3.1) using four versions of MultiSemcor (MSC-IT-1, MSC-IT-2, MSC-FR-1, MSC-FR-2) described above. Then we use the same four versions to train our multilingual SST models based on SRNN and BRNN. For learning our multilingual SST models based on RNN variants proposed in part (C) of section 3.2.2, we also tag SemCor using *TreeTagger* (POS tagger proposed by Schmid (1995)).

4.2.3 Results and discussion

Our models are evaluated on SemEval 2013 Task 12 test corpora. Results are directly comparable with those of systems which participated to this evaluation campaign. We report two SemEval 2013 (unsupervised) system results for comparison:

- **MFS Semeval 2013** : The most frequent sense is the baseline provided by SemEval 2013 for Task 12, this system is a strong baseline, which is obtained by using an external resource (the WordNet most frequent sense).
- **GETALP** : a fully unsupervised WSD system proposed by (Schwab et al., 2012) based on Ant-Colony algorithm.

The DAEBAK! (Navigli and Lapata, 2010) and the UMCC-DLSI systems (Gutiérrez Vázquez et al., 2011) have also participated to SemEval 2013 Task 12. However, they use a supervised approach [6].

Table 2 shows the results obtained by our RNN models and by two SemEval 2013 WSD systems. SRNN-POS-X and BRNN-POS-X refer to our RNN variants: *In* means input layer, *H1* means first hidden layer and *H2* means second hidden layer. We achieve the best performance on Italian using MSC-IT-1 clean corpus while noisy training corpus degrades SST performance. The best results are obtained with combination of simple projection and RNN which confirms (as for POS tagging) that both approaches are complementary.

[6] DAEBAK! and UMCC-DLSI for SST have obtained: 68.1% and 72.5% on Italian; 59.8% and 67.6 % on French

Model		Italian		French	
		MSC-IT-1 trans man.	**MSC-IT-2 trans. auto**	**MSC-FR-1 trans. auto**	**MSC-FR-2 trans auto.**
Baseline	Simple Projection	61.3	45.6	42.6	44.5
SST Based RNN	SRNN	59.4	46.2	46.2	47.0
	BRNN	59.7	46.2	46.0	47.2
	SRNN-POS-In	61.0	47.0	46.5	47.3
	SRNN-POS-H1	59.8	46.5	46.8	47.4
	SRNN-POS-H2	63.1	48.7	47.7	49.8
	BRNN-POS-In	61.2	47.0	46.4	47.3
	BRNN-POS-H1	60.1	46.5	46.8	47.5
	BRNN-POS-H2	63.2	48.8	47.7	50
	BRNN-POS-H2 - OOV	64.6	49.5	48.4	50.7
Combination	Projection + SRNN	62.0	46.7	46.5	47.4
	Projection + BRNN	62.2	46.8	46.4	47.5
	Projection + SRNN-POS-In	62.9	47.4	46.9	47.7
	Projection + SRNN-POS-H1	62.5	47.0	47.1	48.0
	Projection + SRNN-POS-H2	63.5	49.2	48.0	50.1
	Projection + BRNN-POS-In	62.9	47.5	46.9	47.8
	Projection + BRNN-POS-H1	62.7	47.0	47.0	48.0
	Projection + BRNN-POS-H2	63.6	49.3	48.0	50.3
	Projection + BRNN-POS-H2 - OOV	**64.7**	49.8	48.6	51.0
S-E	MFS Semeval 2013	60.7		**52.4**	
	GETALP (Schwab et al., 2012)	40.2		34.6	

Table 2: Super Sense Tagging (SST) accuracy for Simple Projection, RNN and their combination.

We also observe that the RNN approach seems more robust than simple projection on noisy corpora. This is probably due to the fact that no word alignments are required in our cross language RNN. Finally, BRNN-POS-H2-OOV achieves the best performance, which shows that the integration of POS information in RNN models and dealing with OOV words are useful to build efficient multilingual super senses taggers. Finally, it is worth mentioning that integrating low level (POS) information lately (last hidden layer) seems to be the best option in our case.

5 Conclusion

In this paper, we have presented an approach based on recurrent neural networks (RNN) to induce multilingual text analysis tools. We have studied Simple and Bidirectional RNN architectures on multilingual POS and SST tagging. We have also proposed new RNN variants in order to take into account low level (POS) information in a super sense tagging task. Our approach has the following advantages: (a) it uses a language-independent word representation (based only on word co-occurrences in a parallel corpus), (b) it provides truly multilingual taggers (1 tagger for N languages) (c) it can be easily adapted to a new target language (when a small amount of supervised data is available, a previous study (Zennaki et al., 2015a; Zennaki et al., 2015b) has shown the effectiveness of our method in a weakly supervised context).

Short term perspectives are to apply multi-task learning to build systems that simultaneously perform syntactic and semantic analysis. Adding out-of-language data to improve our RNN taggers is also possible (and interesting to experiment) with our common (multilingual) vector representation.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. *CoNLL-2013*, pages 183–192.

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

Luisa Bentivogli, Pamela Forner, and Emanuele Pianta. 2004. Evaluating cross-language annotation transfer in the multisemcor corpus. In *COLING*, page 364.

Alexandre Bérard, Christophe Servan, Olivier Pietquin, and Laurent Besacier. 2016. Multivec: a multilingual and multilevel representation learning toolkit for nlp. In *The 10th edition of the Language Resources and Evaluation Conference (LREC 2016)*.

Laurent Besacier, Benjamin Lecouteux, Marwen Azouzi, and Ngoc-Quang Luong. 2012. The lig english to french machine translation system for iwslt 2012. In *IWSLT*, pages 102–108.

Thorsten Brants. 2000. Tnt: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth CoNLL*, pages 149–164.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the EMNLP-2006*, pages 594–602.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the joint SIGDAT conference on EMNLP and very large corpora*, pages 100–110.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. *Proceedings of the 49th ACL*, 1:600–609.

Long Duong, Paul Cook, Steven Bird, and Pavel Pecina. 2013. Simpler unsupervised pos tagging with bilingual projections. In *ACL (2)*, pages 634–639.

Greg Durrett, Adam Pauls, and Dan Klein. 2012. Syntactic transfer using a bilingual lexicon. In *The Joint Conference on EMNLP and CoNLL*, pages 1–11.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Alexander Fraser and Daniel Marcu. 2007. Measuring word alignment quality for statistical machine translation. *Computational Linguistics*, 33(3):293–303.

Andrew Brian Goldberg. 2010. *New directions in semi-supervised learning*. Ph.D. thesis, University of Wisconsin–Madison.

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *NAACL-HLT*, pages 1386–1390.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. *ICML 2015*.

Alex Graves. 2012. *Supervised sequence labelling*. Springer.

Yoan Gutiérrez Vázquez, Antonio Fernández Orquín, Andrés Montoyo Guijarro, Sonia Vázquez Pérez, et al. 2011. Enriching the integration of semantic resources based on wordnet.

Dan Klein. 2005. *The unsupervised learning of natural language structure*. Ph.D. thesis, Stanford University.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

H Kucera and W Francis. 1979. A standard corpus of present-day edited american english, for use with digital computers (revised and amplified from 1967 version).

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159.

Tomáš Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010*, pages 1045–1048.

Tomáš Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*.

George A Miller, Claudia Leacock, Randee Tengi, and Ross T Bunker. 1993. A semantic concordance. In *Proceedings of the workshop on HLT*, pages 303–308.

Roberto Navigli and Mirella Lapata. 2010. An experimental study of graph connectivity for unsupervised word sense disambiguation. *Pattern Analysis and Machine Intelligence*, 32(4):678–692.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. Semeval-2013 : Multilingual word sense disambiguation. In *Second Joint Conference on Lexical and Computational Semantics*, volume 2, pages 222–231.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting on ACL*, pages 440–447.

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36(1):307–340.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *LREC'12*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. Learning internal representations by error propagation. Technical report, DTIC Document.

Helmut Schmid. 1995. Treetagger| a language independent part-of-speech tagger. *Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart*, 43:28.

Jürgen Schmidhuber. 1992. A fixed size storage o (n3) time complexity learning algorithm for fully recurrent continually running networks. *Neural Computation*, 4(2):243–248.

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

Didier Schwab, Jérôme Goulian, Andon Tchechmedjiev, and Hervé Blanchon. 2012. Ant colony algorithm for the unsupervised word sense disambiguation of texts: Comparison and evaluation. In *COLING*, pages 2389–2404.

Martin Sundermeyer, Ilya Oparin, J-L Gauvain, Ben Freiberg, Ralf Schluter, and Hermann Ney. 2013. Comparison of feedforward and recurrent neural network language models. In *ICASSP*, pages 8430–8434. IEEE.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 conference of the NAACL-HLT*, pages 477–487.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers.

Ivan Titov and Alexandre Klementiev. 2012. Crosslingual induction of semantic roles. In *Proceedings of the 50th Annual Meeting of the ACL*, volume 1, pages 647–656.

J Veronis, O Hamon, C Ayache, R Belmouhoub, O Kraif, D Laurent, TMH Nguyen, N Semmar, F Stuck, and W Zaghouani. 2008. Arcade ii action de recherche concertée sur l'alignement de documents et son évaluation.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*, pages 1–8.

Othman Zennaki, Nasredine Semmar, and Laurent Besacier. 2015a. Unsupervised and lightly supervised part-of-speech tagging using recurrent neural networks. In *PACLIC 29*.

Othman Zennaki, Nasredine Semmar, and Laurent Besacier. 2015b. Utilisation des réseaux de neurones récurrents pour la projection interlingue d'étiquettes morpho-syntaxiques à partir d'un corpus parallèle. In *TALN 2015*.

Bitext Name Tagging for Cross-lingual Entity Annotation Projection

Dongxu Zhang[1], **Boliang Zhang**[2], **Xiaoman Pan**[2], **Xiaocheng Feng**[3],
Heng Ji[2], **Weiran Xu**[1]
[1]Beijing University of Posts and Telecommunications, Beijing, China
zhangdongxuu@gmail.com, xuweiran@bupt.edu.cn
[2]Rensselaer Polytechnic Institute, NY, USA
{zhangb8,panx2,jih}@rpi.edu
[3]Harbin Institute of Technology, Harbin, China
xcfeng@ir.hit.edu.cn

Abstract

Annotation projection is a practical method to deal with the low resource problem in incident languages (IL) processing. Previous methods on annotation projection mainly relied on word alignment results without any training process, which led to noise propagation caused by word alignment errors. In this paper, we focus on the named entity recognition (NER) task and propose a weakly-supervised framework to project entity annotations from English to IL through bitexts. Instead of directly relying on word alignment results, this framework combines advantages of rule-based methods and deep learning methods by implementing two steps: First, generates a high-confidence entity annotation set on IL side with strict searching methods; Second, uses this high-confidence set to weakly supervise the model training. The model is finally used to accomplish the projecting process. Experimental results on two low-resource ILs show that the proposed method can generate better annotations projected from English-IL parallel corpora. The performance of IL name tagger can also be improved significantly by training on the newly projected IL annotation set.

1 Introduction

Annotation projection task aims to deal with low resource issues where human annotations are limited or unavailable in incident languages or domains. Since supervised learning algorithms can not work without annotation sets, annotation projection methods could automatically generate annotations from another language or domain where rich annotation sets are available, such as English.

Yarowsky and Ngai (2001) proposed a method of using parallel text with word alignment results to project annotations. Fig. 1 shows an example of entity projection with word alignment results from English to Turkish. On English side, *Voice of America Radio* and *Congo* are tagged as an organization (ORG) and a location (LOC) respectively by an English name tagger. The dashed lines represent word alignment results generated by a word alignment tool. Following alignment results, we can project labels to *Amerikann Sesi* and *Congo* on Turkish side automatically. A major problem of this framework is that it suffers from noises produced by word alignment errors. Thus, some de-noising methods have been proposed (Kim et al., 2010; Wang and Manning, 2014).

Though promising, this framework has several disadvantages. One shortcoming is that it totally depends on word alignment results. In this case, noise propagation from word alignment errors is heavily troublesome. To alleviate this problem, there are post-processing methods for annotation correction (Kim et al., 2010) and soft expectations to make use of more probabilistic information inside word alignment results (Wang and Manning, 2014). Although post-processing correction is efficient to filter out wrong labels, it can hardly find back labels which have been lost in the word alignment step. The soft expectation method leverages probabilistic information from word alignment results instead of hard labels such as one and zero. It can revive some false negative cases where true answers still have quite high rankings (but not the highest one). But this method will fail if word alignment results are totally wrong. From

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 461–470, Osaka, Japan, December 11-17 2016.

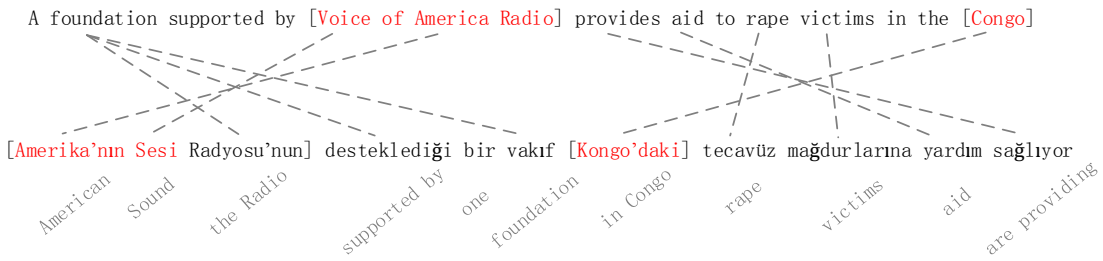


Figure 1: Errors of annotation projection with word alignment results using English and Turkish bitext.

another aspect, since word alignment task is mainly designed for the machine translation task (Och and Ney, 2003), it is not specifically tuned for other NLP tasks such as annotation projection.

Another disadvantage is that this framework depends on full-sequence word alignment where each alignment pair in the sequence will be taken into account for annotation projection. Then, the entity projection could always be disrupted by alignment errors on low-frequency word pairs and outliers, especially when the quality of bitexts cannot be guaranteed in low-resource languages. For intuition, in Fig. 1, the overall word alignment quality seems to be acceptable. But for entity projection, *Amerikann Sesi Radyosunun* is failed to be labeled completely caused by a single word alignment error on *Radyosunun*. In this circumstance, we should instead try to only focus on projecting meaningful tags, for example name tags.

In this paper, we focus on entity projection task on English-IL bitext and propose a weakly supervised framework to train a bitext name tagger and deal with issues mentioned above. The main contributions of this paper are as follows:

- We propose a new weakly-supervised framework for entity annotation projection. The framework contains two steps which can increase precision and recall step by step.
- The proposed model does not heavily depend on word alignment results. It bypasses the use of full word alignment results by taking the original English and IL data from the bitext as inputs and using training process to learn the projection. Also, the model only focuses on projecting name tags.
- We employ connections among hidden layers in recurrent neural networks to deal with sequence labeling tasks across parallel corpora.

2 Method

Given English-IL bitexts, we could start with labeling all sentences on the English side using a high-quality English name tagger [1] and find out entity names in each English sentence. For the rest of this section, our goal is to project these labeled entities from English to IL. To accomplish this goal, two separate steps are carried out. Firstly, in Sec 2.1, a **high-confidence annotation set** is generated using strict rules on parallel corpora. Secondly, in Sec 2.2, a supervised bitext name tagger is trained to recall those annotations which were lost during the first step. Besides, in Sec 2.3, we provide two different strategies to correct some errors made by the second step and give further improvements on the quality of projection.

2.1 High-confidence Annotation Projection

In order to supervise the training of the bitext name tagger, we need to firstly generate a high-quality training set out of the entire bitext. A naive way of projecting names from English to IL is to follow word alignment results [2]. Since word alignment task is not perfectly solved [3] and the performance could

[1] In our experiment, we use the Stanford NER tool (Manning et al., 2014).

[2] Here we use GIZA++ (Och and Ney, 2003)

[3] The poor alignment occurs especially when the parallel corpora is not enough (low resource issue) or the quality of bitext is poor.

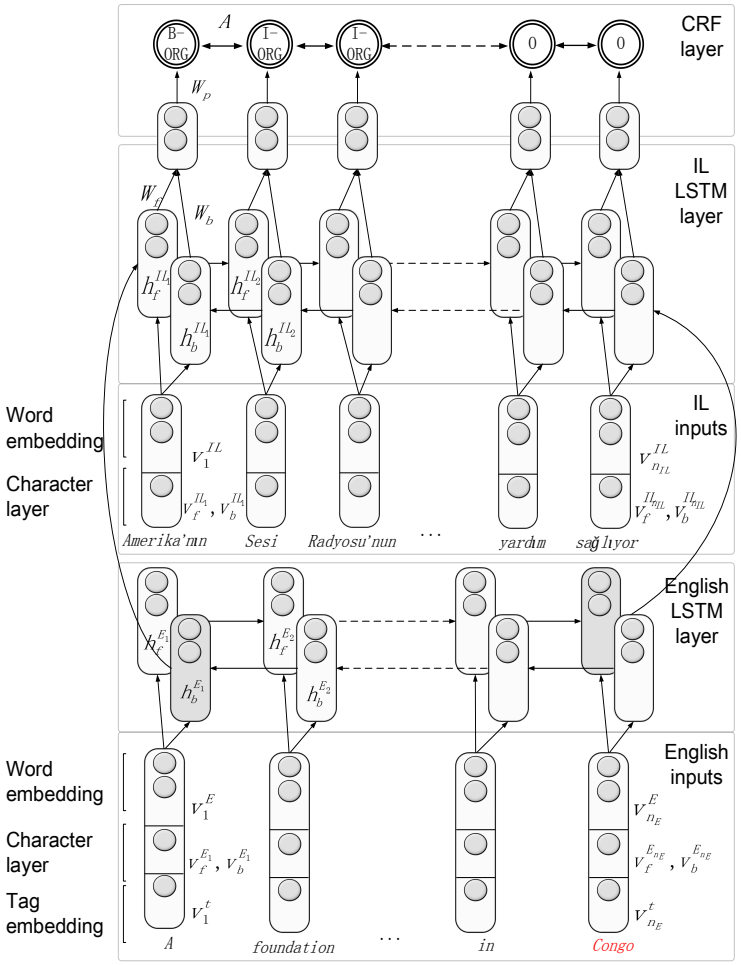


Figure 2: The framework of the bitext name tagger.

become much worse when considering the projection over low frequency phrases such as names, here we add some rules for name searching before using word alignment results.

For each labeled name in a English sentence, we search for the corresponding IL name on the IL side of the bitext as follows:

1. Firstly, if the Levenshtein distance between an IL word sequence and the English name is close enough, the word sequence will be labeled with the English name tag.

2. The second choice of measuring the similarity is to use Soundex (Raghavan and Allan, 2004).

3. If previous steps can not find the corresponding name in IL, a word-to-word translation table is used. The table is derived from GIZA++ and we only keep top 5 most credible translations for each word. And an exact word-to-word translation of entity names is carried out for string matching.

After searching steps, we use word alignment results to project names that has not been found in IL. Here we follow constraints that the number of words in a name should be the same between English and IL mentions, and words in the projected name should be contiguous. Finally, we only keep those sentence pairs where all the entity names labeled in English have been successfully projected by using previous methods.

2.2 Name Tagging on Parallel Text

In the previous section, a high-confidence annotation set is automatically generated. Using this annotation set with bitext inputs, we could train a bitext name tagger for name annotation projection. Figure

2 shows the structure of this neural-based model. Here we utilize the flexibility of recurrent neural networks to label IL sequences with the information flowing from English side. For both English and IL side, we employ bi-directional LSTM networks to handle sequence inputs of varied lengths.

Basically, our model follows the recipe of Lample et al. (2016), with several extensions designed for this task. First, there exist two sets of embedding and recurrent layers in order to handle inputs from both English and IL side. Second, to combine these two parts of signals, there are connections between two bi-directional LSTMs where the zero step of hidden layers on IL side is initialized by the last step of hidden layers on English side. Third, there are not only word and character level information, but tag sequences involved in the input signal, on both English and IL sides. Details of the bitext name tagger will be introduced in the rest of this section.

Input signal

We have three different types of input sequences for each sentence pair in the bitext:

- Word sequence $X_L = (x_1^L, x_2^L, ..., x_{n_L}^L)$, $x_i^L \in \{0, 1, ..., V_L - 1\}$, where $L \in \{E, IL\}$ represents English or IL, V_L is the size of vocabulary in L, and n_L is the number of words in the current word sequence of L.

- In order to let system know which names in English need to be projected, it is crucial to add the sequence of name types labeled by the English name tagger: $T_E = (t_1^E, t_2^E, ..., t_{n_E}^E)$, $t_i^E \in \{0, 1, ..., V_t - 1\}$, where V_t is the number of tag types. Here $V_t = 7$ since we follow the IOB format (*Inside, Outside, Beginning*) with three entity type *PERSON, LOCATION and ORGANIZATION*.

- To leverage character level information, for each word x_i^L, there is a character sequence $C^{L_i} = (c_1^{L_i}, c_2^{L_i}, ..., c_p^{L_i})$, $c_j^{L_i} \in \{0, 1, ..., V_{Lc} - 1\}$, where p is the number of characters in word X_i^L, and V_{Lc} is the number of different characters in language L.

Embeddings for each word

Then, we project these input signals from high dimensional space of token id into dense vector spaces using look-up tables. Thus, we have

$$v_i^L = W_{word}^L x_i^L,\ L \in \{E, IL\}$$

$$v_j^{L_i} = W_{char}^L c_j^{L_i},\ L \in \{E, IL\}$$

$$v_i^t = W_{tag}^E t_i^E$$

where W_{word}^L,W_{char}^L, W_{tag}^E are look-up tables for English/IL word, English/IL character, and English tag respectively.

Since character level information is helpful for name tagging task (Klein et al., 2003), we combine the word level embedding and character level embedding together by following Lample et al. (2016). For each work token, we derive a vector from the corresponding character embedding sequence using bidirectional LSTM networks as following:

$$v_f^{L_i} = LSTM_{for}^{LC}(\mathbf{v}^{L_i})[p-1],\ \ v_b^{L_i} = LSTM_{back}^{LC}(\mathbf{v}^{L_i})[0]$$

where $\mathbf{v}^{L_i} = (v_1^{L_i}, v_2^{L_i}, ..., v_p^{L_i})$, $LSTM_{for}^{LC}(\cdot)$ and $LSTM_{back}^{LC}(\cdot)$ are the forward and backward long short-term memory recurrent networks (LSTM-RNN) (Hochreiter and Schmidhuber, 1997) to encode $\mathbf{v}^{L_i}$. The output of $LSTM(\cdot)$ is a list of LSTM-RNN hidden layers.

Then, we concatenate v_i^E, $v_f^{E_i}$, $v_b^{E_i}$ and v_i^t into a vector d_i^E which represents the information of word i in each English sentence. Similarly, we concatenate v_i^{IL}, $v_f^{IL_i}$, $v_b^{IL_i}$ into a vector d_i^{IL} for each word in IL. In practice, to leverage information from rich unlabeled monolingual corpus, we initialize W_{word}^E and W_{word}^{IL} with pre-trained word embeddings using word2vec tool (Mikolov et al., 2013).

Two bi-directional LSTMs for bitext

After generating $\mathbf{d}^E = (d_1^E, d_2^E, ..., d_{n_E}^E)$ and $\mathbf{d}^{IL} = (d_1^{IL}, d_2^{IL}, ..., d_{n_{IL}}^{IL})$, we put them into English and IL bidirectional LSTM-RNNs separately:

$$\mathbf{h}_f^E = LSTM_{for}^E(\mathbf{d}^E), \;\; \mathbf{h}_b^E = LSTM_{back}^E(\mathbf{d}^E)$$

where $LSTM_{for}^E(\cdot)$ and $LSTM_{back}^E(\cdot)$ are the forward and backward LSTM-RNNs to encode $\mathbf{d}^E$. And $\mathbf{h}_f^E$ and $\mathbf{h}_b^E$ are the list of hidden layers. Similarly, for IL, we have

$$\mathbf{h}_f^{IL} = LSTM_{for}^{IL}(\mathbf{d}^{IL}, \mathbf{h}_b^E[0]), \;\; \mathbf{h}_b^{IL} = LSTM_{back}^{IL}(\mathbf{d}^{IL}, \mathbf{h}_f^E[n_E - 1])$$

To utilize information extracted from English side, we use $h_f^E[n_E - 1]$ and $h_b^E[0]$ from the last steps of bi-directional LSTM to initialize the starting states of LSTM hidden layers in IL side. Thus during training, errors can be back propagated to English side neural networks through these two vectors. The two hidden layers filled with gray color of background in Figure 2 show the intuition.

This idea is inspired by one of previous successful sequence-to-sequence deep learning method in machine translation (Sutskever et al., 2014), which encodes an input sequence into one single vector and then generates a new sequence with both the vector and the language model contained in its decoding networks. In our case, the task is much easier since we only need to predict a sequence of tag types instead of word tokens from a huge vocabulary. And the model also absorbs hints from the IL side of input sequence so it is not at all an auto-encoder.

After LSTM layers in IL, the score function is given to assess all the name types for each token in IL based on hidden layers of LSTM as follows:

$$P_i = W_p tanh(W_f \mathbf{h}_f^{IL}[i] + W_b \mathbf{h}_b^{IL}[i])$$

where P_i is a vector of dimension V_t representing scores of tags for x_i^{IL}.

Training and Projecting

Following the architecture of Lample et al. (2016), we add a first order transition matrix A on top of the previous model to simulate a CRF structure. The score function between input $\mathbf{X}$ and output tag sequence $\mathbf{y}$ is as following,

$$s(\mathbf{X}, \mathbf{y}) = s(\tilde{X}_E, \tilde{X}_{IL}, T_E, \mathbf{y}) = \sum_{i=1}^{n_{IL}} A_{y_i, y_{i+1}} + \sum_{i=1}^{n_{IL}} P_{i, y_i}$$

where $\tilde{X}_E$ and $\tilde{X}_{IL}$ represent X_E and X_{IL} with their character sequences. To normalize each score into probability, a softmax function is added,

$$p(\mathbf{y}|\mathbf{X}) = \frac{e^{s(\mathbf{X}, \mathbf{y})}}{\sum_{\tilde{\mathbf{y}} \in \mathbf{Y_X}} e^{s(\mathbf{X}, \tilde{\mathbf{y}})}}$$

Thus, the final output $\mathbf{y}_d$ for decoding is the tag sequence that has the highest probability among all possible tag sequences $\mathbf{Y_X}$.

During training, our goal is to maximize $\log p(\mathbf{y}_c|\mathbf{X})$ for the correct (high-confidence annotation) tag sequence $\mathbf{y}_c$. Stochastic gradient descent (SGD) is employed with dropout strategy [4].

Using the trained bitext tagger in this section, we can complete the name projection task by annotating the rest of sentence pairs not covered in the high-confidence annotation set, and generate the final annotation set in IL by combining both high-confidence annotation set and annotations found by the bitext tagger.

[4]For the detailed training strategy, please read the training section in Lample et al. (2016)

2.3 Error Correction Strategies

Sec 2.1 and Sec 2.2 have already introduced the main content of the proposed framework. In this section, we make use of an overlooked information to further improve the quality of the projection.

According to Sec 2.1, to generate the high-confidence annotation set, we omit those sentence pairs whose names labeled in English are not completely projected in IL. For example, we will exclude a sentence pair in the first step if the number of names in English is three while only two names are projected to IL. However, these two projected names are still *high-confidence*. So if these names can be properly utilized, the quality of projection could be further improved. Here we introduce two different strategies to make use of this information:

Post-processing strategy

Since there are two annotation results on each of these omitted sentence pairs: annotations generated by the first step and annotations generated by the second step, it is natural to implement a post-processing step and integrate these two annotation results. Compared with the second step, although the annotations from the first step is not complete, they should be more reliable if rules in Sec 2.1 are strict enough. So here we follow this assumption and force the final projected annotations to maintain names produced in the first step and add names from the second step only when there is no conflict between them.

High-confidence annotations as one of inputs

A disadvantage of the post-processing method is that the assumption that annotations from the first step are more trust-worthy is not always true. When the annotated names produced by the first step are wrong, they will not only introduce noise, but also ruin annotations from the second step if there are conflicts between them. So a better idea is to feed all the information we have to neural networks and let the model speaks the truth.

To provide information of annotations produced by the first step, we add another input signal $T_{IL} = (t_1^{IL}, t_2^{IL}, ..., t_{n_{IL}}^{IL})$ in IL side, where $t_i^{IL} \in \{0, 1, ..., V_t - 1\}$ represents the tag result from the first step, V_t is the number of tag types. Then T_{IL} will go through tag embedding layers, LSTM-RNN layers and influence the results of CRF tagging.

In order to simulate the scenario of prediction process where the number of high-confidence names found on IL side is less than the number on English side, it is crucial to randomly drop out some high-confidence annotation names in T_{IL} and replace them with *Outside* during the training time.

3 Experiments

3.1 Data and Experimental Setup

To simulate a practical scenario, we evaluate our model on two low-resource ILs: Turkish and Uzbek, using the ground-truth name tagging annotations from the DARPA LORELEI program [5]. Table 1 shows data statistics. The numbers of sentence pairs for training and development represent the high-confidence annotation sets we produced during our experiments. And we exclude all the ground truth data from bitext for training and validation. Since a small proportion of sentences in the ground truth do not have English bitext, the test sets are slightly different in Sec 3.2 and Sec 3.3.

In our experiment, we set learning rate=0.01 and dropout rate=0.5. Dimension of word embedding=300, dimension of hidden layer=100, dimension of character embedding and hidden layer=25, and dimension of tag embedding=25.

3.2 Results of Bitext Name Projection

Bitext name taggers are trained on high-confidence annotation sets. Since most of ground truth sentences also contain parallel data, we can project annotations on bitext of ground truth and directly evaluate the quality of the projection. Here we compare our results with names projected from pure word alignment and also method in Sec 2.1. Table 2 shows the result.

[5] http://www.darpa.mil /program/low-resource-languages-for-emergent-incidents

Category	Turkish	Uzbek
Full sentence pairs for projection	24,193	39,045
Sentence pairs for training	12,276	21,552
Sentence pairs for dev.	755	1,431
Ground truth sentence pairs on bitext	1759	2918
Ground truth names on bitext	1744	2894
Ground truth sentence pairs in total	2,121	3,040
Ground truth names in total	2,178	3,144

Table 1: Data statistics

Dataset	Turkish			Uzbek		
	P	R	F	P	R	F
WA	34.5	33.4	33.9	28.6	29.9	29.2
HA	66.6	38.3	48.7	**66.5**	39.6	49.6
BNT	**67.4**	**49.7**	**57.2**	62.7	**47.8**	**54.3**

Table 2: Performance of projected annotation sets with different projection strategies. *WA* represents projection results purely based on **w**ord **a**lignment with IOB constraint. *HA* represents the **h**igh-confidence **a**nnotation projection. *BNT* represents the annotation projection using **b**itext **n**ame **t**agger.

From the result, it demonstrates that, in the case of low resource languages, annotations with word alignment results perform poorly on both precision and recall rate. So it is hard to improve the projection based on word alignment framework. Even re-ranking strategies with word alignment results is not effective since searching-based method (HA) has significantly outperformed the word alignment results. Also, compared with high-confidence annotations, our final annotation set shows significant better recall rate without much precision loss. Table 3 also indicates that after the second step, there are a huge number of names (not necessarily correct names) been discovered. This indicates in another aspect the significance of the recalling process in the second step. When observing real labeled cases, our model shows stable performances without much influences from the quality of word alignment result. For instance, the bitext name tagger could successfully label *Radyosu'nun* as a I-ORG in Fig 1.

Lang.	High-confidence annotation set			Annotation set after recalling		
	PER	LOC	ORG	PER	LOC	ORG
Turkish	6469	6133	2329	13520	21346	8094
Uzbek	9889	11353	1696	20522	29769	5348

Table 3: Statistics of tag number before and after using bitext name tagger.

Further more, since in the low resource scenario, large size of parallel corpora is not always available, we do experiments on different sizes of training data to evaluate the capability of this framework with less amount of bitexts. The subset sentence pairs for training are randomly selected. Figure 3 shows the result on Turkish and Uzbek. Different from the case where RNN is employed on machine translation, we do not need a huge number of training data because the searching space of prediction is quite small, due to the small amount of tag types rather than the size of a vocabulary. From the curve, we can see that even with only 2000 parallel sentence pairs, our model shows around 50% F-value. Notice that since we can directly employ the word alignment method and also method in Sec 2.1 on the ground truth without training process, the curve of these two methods are flat.

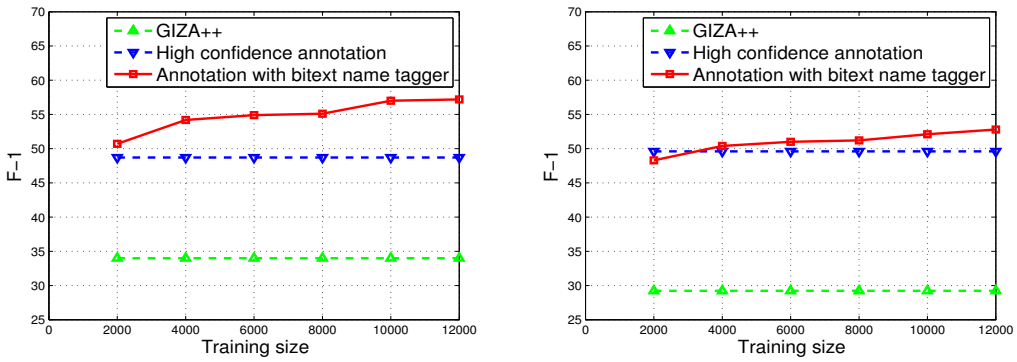


Figure 3: Performances of annotation projection with different size of training data on Turkish(left) and Uzbek(right) bitexts.

Model	Turkish			Uzbek		
	P	R	F	P	R	F
Clean	70.5	66.9	68.7	73.4	68.8	71.0
ExpDriv	45.8	**51.1**	48.3	38.3	41.0	39.6
HA *	62.3	45.3	52.5	59.4	45.2	51.3
BNT *	64.6	**51.1**	57.0	**62.6**	46.9	53.6
BNT + PP	65.6	49.7	56.5	59.5	**51.4**	55.2
BNT + HAI *	**66.4**	50.9	**57.6**	62.2	49.8	**55.3**

Table 4: Performance of IL name taggers. *Clean* represents the performance of LSTM-CRF model trained on human annotation data. *ExpDriv* refers to the baseline **Exp**ectation **driv**en method. *HA* represents LSTM-CRF model trained with **h**igh-confidence **a**nnotation set. *BNT* represents LSTM-CRF model trained with annotation set produced by the **b**itext **n**ame **t**agger. + *PP* represents *BNT* with **p**ost-**p**rocessing strategy. + *HAI* represents *BNT* with **h**igh-confidence **a**nnotations as one of **i**nputs. * indicates consistent improvement compared with results in the line above.

3.3 Results of IL Name Tagging

Since the final goal of annotation projection is to help IL tasks, we train and compare the performances of NER models on IL annotation sets produced with different projection methods. Here we choose the LSTM-CRF model proposed by Lample et al. (2016) to be the model of IL NER system because of its state-of-the-art performance on English NER task[6]. Table 4 shows the results.

From the results, the model trained with annotation set produced by the bitext name tagger outperforms both high-confidence annotations and also the state-of-the-art Expectation-driven learning method for IL name tagging (Zhang et al., 2016). Contrasting results between two different strategies in Sec 2.3, HAI shows consistent improvement across different languages while post-processing strategy fails in Turkish. Table 2 shows that, in Turkish, the precision of annotations from the second step outperforms the first step, which means in this case the assumption of the post-processing strategy is not valid.

4 Related Work

Most of related methods for annotation projection are based on word alignment results. Kim et al. (2010) employed both heuristic method and alignment correction with alignment dictionary of entity mentions. Das and Petrov (2011) designed a label propagation method to automatically induce a tag lexicon for the foreign language to smooth the projected labels. Wang and Manning (2014) project model expectations rather than labels, which facilities transfer of model uncertainty across language boundaries in word alignment projection.

[6]You can download the code of LSTM-CRF tagger from `https://github.com/glample/tagger`

Wang et al. (2013) also proposed a method to joint train word alignment and bilingual name tagging, which involves training process of word alignment. But this joint method is based on two assumptions. First, it requires a readily-trained name tagger in each languages. Even more, both taggers need to have competitive strengths so that they can correct each other. Unfortunately, in the case of low resource languages, no competitive name tagger is available.

One of most recent works linked to annotation projection was proposed by Fang and Cohn (2016) for the task of part of speech tagging (POS). Their work interestingly combined both gold annotations and projected ones by learning a global corrective matrix between gold annotation and projected annotation on IL side. The limitation is that gold annotation set on IL side must exist, which is not always the case especially in an incident language.

5 Conclusion and Future Work

We introduce a weakly-supervised framework for entity annotation projection. Our model takes original English and IL bitexts as inputs and does not heavily depend on word alignment results. Experiment results show that this method can provide significantly better annotations projected from English to IL.

Notice that in the first step of our framework, we do not consider the case where low-resource language does not use the Roman script. Although the method will still work because we also use word alignment results to measure the distance between text sequences, the performance could drop. So in the future work, we should plug in some transliteration techniques and gazetteers to make the system more robust.

Acknowledgments

We thank all the anonymous reviewers for their valuable comments and constructive suggestions. This work was supported by the 111 Project of China under Grant No. B08004, the key project of ministry of science and technology of China under Grant No. 2011ZX03002-005-01, the National Natural Science Foundation of China under Grant No. 61273217, the Natural Science Foundation of China under Grant No.61300080, the Ph.D. Programs Foundation of Ministry of Education of China under Grant No. 20130005110004, the U.S. DARPA LORELEI Program No. HR0011-15-C-0115 and ARL/ARO MURI W911NF-10-1-0533. The content is solely the responsibility of the authors and does not necessarily represent the official views of official policies or government.

References

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 600–609. Association for Computational Linguistics.

Meng Fang and Trevor Cohn. 2016. Learning when to trust distant supervision: An application to low-resource pos tagging using cross-lingual projection. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning*.

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

Seokhwan Kim, Minwoo Jeong, Jonghoon Lee, and Gary Geunbae Lee. 2010. A cross-lingual annotation projection approach for relation detection. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 564–571. Association for Computational Linguistics.

Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D Manning. 2003. Named entity recognition with character-level models. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 180–183. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Hema Raghavan and James Allan. 2004. Using soundex codes for indexing names in asr documents. In *Proceedings of the Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval at HLT-NAACL 2004*, pages 22–27. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Mengqiu Wang and Christopher D Manning. 2014. Cross-lingual pseudo-projected expectation regularization for weakly supervised learning. *Transactions of the Association for Computational Linguistics*, 2:55–66.

Mengqiu Wang, Wanxiang Che, and Christopher D Manning. 2013. Joint word alignment and bilingual named entity recognition using dual decomposition. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1073–1082.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proceedings of the 2001 Conference of the North American Chapter of the Association for Computational Linguistics*.

Boliang Zhang, Xiaoman Pan, Tianlu Wang, Ashish Vaswani, Heng Ji, Kevin Knight, and Daniel Marcu. 2016. Name tagging for low-resource incident languages based on expectation-driven learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*.

Determining the Multiword Expression Inventory of a Surprise Language

Bahar Salehi,[1] Paul Cook[2] and **Timothy Baldwin[1]**

[1] NICTA Victoria Research Laboratory
Department of Computing and Information Systems
The University of Melbourne, Victoria 3010, Australia

[2] Faculty of Computer Science
University of New Brunswick
Fredericton, NB E3B 5A3, Canada

bsalehi@student.unimelb.edu.au, paul.cook@unb.ca, tb@ldwin.net

Abstract

Much previous research on multiword expressions (MWEs) has focused on the token- and type-level tasks of MWE identification and extraction, respectively. Such studies typically target known prevalent MWE types in a given language. This paper describes the first attempt to learn the MWE inventory of a "surprise" language for which we have no explicit prior knowledge of MWE patterns, certainly no annotated MWE data, and not even a parallel corpus. Our proposed model is trained on a treebank with MWE relations of a source language, and can be applied to the monolingual corpus of the surprise language to identify its MWE construction types.

1 Introduction

Multiword expressions ("MWEs") are word combinations which have idiosyncratic properties relative to their component words (Sag et al., 2002; Baldwin and Kim, 2010), such as *taken aback* or *red tape*. The need for an explicit model of MWEs has been shown to be important in NLP tasks including machine translation (Venkatapathy and Joshi, 2006), parsing (Constant et al., 2012), and keyphrase/index term extraction (Newman et al., 2012). However, existing approaches to MWE identification/extraction typically target specific MWE types that are known to be prevalent in a given language, such as: (a) compound nouns in languages such as English (Copestake, 2003; Ó Séaghdha, 2008), German (Schulte im Walde et al., 2013) and Japanese (Tanaka and Baldwin, 2003); (b) light verb constructions (LVCs) in languages such as English (Butt, 2003), Persian (Karimi-Doostan, 1997) and Italian (Alba-Salas, 2002); and (c) compound verbs in languages such as Japanese (Uchiyama et al., 2005). Note here that the combination of highly-productive MWE types can vary greatly across languages: English is rich with compound nouns and LVCs are also common, but lacks compound verbs; Persian is rich with LVCs and adjective–noun compounds, but has very few compound nouns and compound verbs; and Japanese is rich with LVCs and compound nouns and verbs, but adjective–noun MWEs are rarer. Even for collocation extraction, this knowledge is generally assumed for a given language, in targeting only highly productive constructions such as adjective–noun or verb–noun collocations (Krenn and Evert, 2001; Pecina, 2008).

But what if the language of interest is one where no such prior knowledge exists, e.g. because it is a "surprise" language where rapid deployment of language technologies is required and there is no access to an informant with sufficient linguistic training to be able to inventorise the MWE types in the language (Oard, 2003; Maynard et al., 2003)? Here, there is little expectation of success without an automatic method for determining the inventory and relative frequency of MWEs in a given language. This provides the motivation for this paper: can we develop a method for automatically profiling the MWE inventory of a novel language based simply on a monolingual corpus of that language, and a treebank in a second language such as English?

We carry out this research in the Universal Dependency ("UD") framework (Nivre et al., 2016), using the method of Duong et al. (2015) to induce a delexicalised dependency parser for the surprise language, based on a supervised parsing model for a language such as English where we have a well-developed treebank in the UD. Given the parser output over a monolingual corpus in the surprise language, we then apply one of two methods to extract our MWE profile: (1) a baseline method, where we simply

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 471–481, Osaka, Japan, December 11-17 2016.

extract out delexicalised dependency tuples of relation type `mwe` or `compound` (including the POS tags), aggregate the counts of the pos–relation–pos triples, and extract the most frequent triples; and (2) a supervised reranker over the delexicalised dependency tuples, to better deal with noise in the output of the delexicalised dependency parser.

One additional contribution of this paper is analysis of MWE annotation across different languages in the UD. We find that there are a number of competing styles of annotation, and very different levels of thoroughness in the annotation of MWEs. As part of this, we perform an "oracle" analysis of MWE extraction based on the gold-standard treebank annotations for a given language, and find that the results vary greatly between languages, due to annotation divergences. Using the supervised reranking method, however, and incorporating more and more languages for training (but holding out the surprise language), we find that we are able to "smooth" annotation differences between languages.

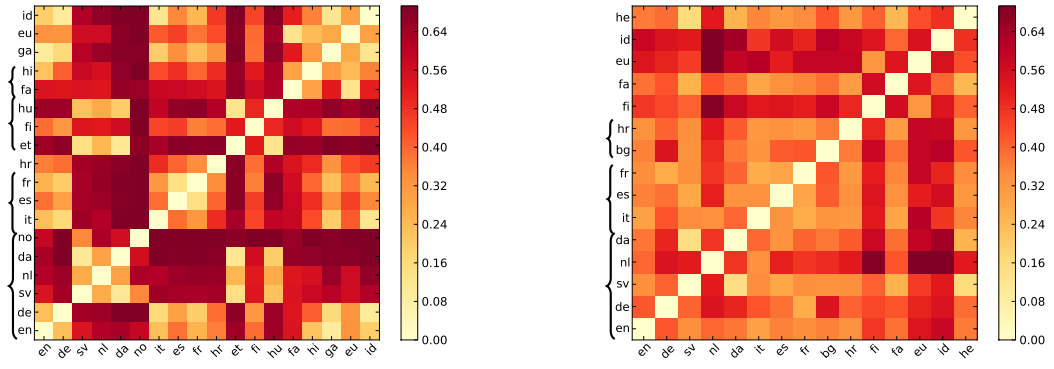
2 Related Work

There is a wealth of research on MWE identification (i.e. distinguishing MWEs from non-idiosyncratic combinations at the token level) and extraction (i.e. determining at the type level which word combinations in a corpus are MWEs). Many of these methods are customised to particular MWE constructions which are known to exist in a given corpus, e.g. noun compounds (Lapata and Lascarides, 2003; Tanaka and Baldwin, 2003), verb particle constructions ("VPCs": Baldwin and Villavicencio (2002; Baldwin (2005)), determinerless prepositional phrases (Baldwin et al., 2004; van der Beek, 2005), or compound verbs (Breen and Baldwin, 2009). There is also a significant body of work on general-purpose MWE extraction, often based on statistical association measures applied to either a monolingual corpus (Evert and Krenn, 2005; Pecina, 2008; Ramisch, 2012) or a parallel corpus (Melamed, 1997; Moirón and Tiedemann, 2006). Even here, however, POS-based constraints are generally applied on the types of MWE that are extracted (e.g. noun–noun or verb–noun bigrams). There are also methods for identifying MWEs in context using supervised models (Diab and Bhutada, 2009; Li and Sporleder, 2010; Schneider et al., 2014), which require exhaustive annotation of MWE token occurrences in a corpus. All of this research differs from our work in that it either assumes knowledge of the type(s) of MWE to extract for a given language, or requires explicitly annotated MWE data in that language.

Closer to home, there has recently been work on general-purpose, unsupervised approaches to MWE extraction, making no assumptions about the types of MWE that exist in a given language (Newman et al., 2012; Brooke et al., 2014). Here, however, the definition of MWE tends to be blurred somewhat to focus on index terms or "formulaic language", i.e. idiomatic expressions with statistically-marked properties in a given corpus — blurred in the sense that many MWEs are not statistically marked, and also that they include formulaic expressions such as *in this paper* that are not formally MWEs.

Also related is recent work on resource development for low-resource languages, such as dependency parsing based on transfer learning from a higher-density language (Naseem et al., 2012; Täckström et al., 2013; Duong et al., 2015). For example, Duong et al. (2015) proposed a neural network-based parser that transfers dependency relations across languages without requiring a parallel corpus. They learn syntactic cross-lingual word embeddings by training the skip-gram model (Mikolov et al., 2013) on a representation of the original text in which the context of each token is represented by its universal POS tags (Petrov et al., 2012). They then incorporate these word embeddings in a transition-based neural network dependency parser (Chen and Manning, 2014).

Our proposed method is the first attempt to learn the MWE profile of a language with no knowledge of the target language except for POS tags (which themselves can be induced automatically, with little or no annotated data: Garrette and Baldridge (2013), Duong et al. (2014)), and no parallel corpus. We train a delexicalised dependency parser based on transfer learning (involving no syntactic annotations for the target language), and train a reranking model based on observed MWEs in only the source language(s).



(a) `compound` (b) `mwe`

Figure 1: Cross-lingual similarity of MWE pattern distributions using JSD

3 Resources

The Universal Dependency Treebank[1] ("UD") is a universal annotation scheme for dependency parsing that is consistent among languages with the goal of cross-lingual learning (Nivre et al., 2016). It is made up of a universal part-of-speech (POS) tag set (Petrov et al., 2012) and universal dependency relation set, and a set of treebanks.

In this paper, we use v1.2 of UD. MWEs are labeled as either `name` (for named entities), `compound` (for binary compound expressions) or `mwe` (for fixed expressions). In this work, we focus specifically on `mwe` and `compound`, as named entity recognition is a specialised subtask of MWE identification with its own dedicated literature (Maynard et al., 2003; Huang et al., 2003; Steinberger and Pouliquen, 2007), and there is every expectation that all languages contain named entities. Although the documentation for UD provides definitions of how to distinguish `mwe` and `compound` in labelling MWEs, there seem to be major inconsistencies in how they have been interpreted for particular languages: some languages do not use these relations at all, while others only annotate a subset of MWE types with these relations.

The languages examined in this paper are as follows, in descending order of prevalence[2] of MWE annotations in UD (as indicated in parentheses):

> Hindi (14.0%), **Indonesian** (9.2%), **Persian** (7.5%), **Croatian** (6.7%), **English** (6.2%), **Swedish** (5.4%), Estonian (4.9%), Irish (4.7%), Finnish (4.4%), Basque (3.7%), Hungarian (2.7%), Dutch (2.2%), Norwegian (1.6%), Danish (1.5%), French (1.5%), Italian (0.8%), Spanish (0.8%), Hebrew (0.7%), Bulgarian (0.6%), German (0.4%)

These were selected based on the fact that they have at least 100 individual occurrences of the `mwe` or `compound` relation. The 5 languages in bold were selected as our test languages, based on the high prevalence of MWE annotations and diversity of MWE patterns.[3] Here and for the remainder of the paper, we define "MWE pattern" to be an ordered tuple of the form $\langle \text{pos}_h, \textsf{rel}, \text{pos}_d \rangle$, where pos_h is the POS of the head, and pos_d is the POS of the dependent in the triple. Based on this definition, English has 56 distinct MWE patterns, Croatian 49, Persian 48, Swedish 45, and Indonesian 26.

4 MWE Patterns

This paper investigates the profile of MWE patterns in a given language, in the form of delexicalised dependency tuples. The most frequent patterns in our 5 target languages with the `compound` relation

[1] `https://universaldependencies.github.io/docs/`

[2] the proportion of MWE tokens

[3] We discarded Hindi despite the high proportion of MWEs because: (1) it only covered `compound` relations, and has no `mwe` relations, and (2) it has a low number of distinct MWE patterns (23), and as such appeared skewed in its annotation.

are: NOUN–NOUN; PROPN–PROPN (i.e. proper noun dependencies, which should be annotated with the `name` relation rather than `compound`, according to the annotation guidelines); and VERB–NOUN, which includes LVCs. There are also other noticeable patterns such as VERB–ADV(erb) and VERB–ADP(osition), corresponding to VPCs (Schulte im Walde, 2004; Baldwin, 2005).

`mwe` patterns are more diverse than `compound` patterns: `compound` patterns mostly involve nouns and verbs, while `mwe` patterns involve a diverse range of POS types, such as ADP–ADP or ADV–ADV, and pairings including CONJ(unctions) or SCONJ (subordinating conjunctions).

We additionally measured the similarity between the MWE pattern probability distribution of the different languages using Jensen–Shannon divergence, as shown in Figure 1 for all languages in UD. To make comparison between related languages easier, we clustered the languages by language family. According to Figure 1, there is no clear indication that languages of the same family have similar MWE patterns, which is something that we might have expected.

These results suggest that although the ultimate goal of the UD project is to have compatible annotations, the MWE annotations are not, at present, consistent. In fact, annotation divergences would appear to be more noticeable than linguistic differences. For example, the Norwegian treebank annotates only VPCs (and not multiword compound nouns, e.g.), and the `mwe` relation is not used at all. That is, the observed differences in MWE patterns certainly reflect differences between languages, but greater than this, they capture differences in the annotation process between different languages.

We also examined the annotation consistency of MWEs between the Train+Dev sets and Test set of each language (based on the provided splits), and observed high consistency (low JSD) between the existing patterns in these sets for the same language. The JSD on `compound` patterns are all below 0.10, except for Spanish (0.23) and French (0.18). Due to the diversity of `mwe` patterns, the JSD is less consistent within each language, with Croatian (0.64) and German (0.44) being notably high, and the rest of the languages below 0.25. This shows that annotation is quite consistent within each language.

Therefore, despite the cross-lingual annotation inconsistency, our corpora appear to be internally consistent enough to train a model over, based on the observed MWEs in a language.

5 Methodology

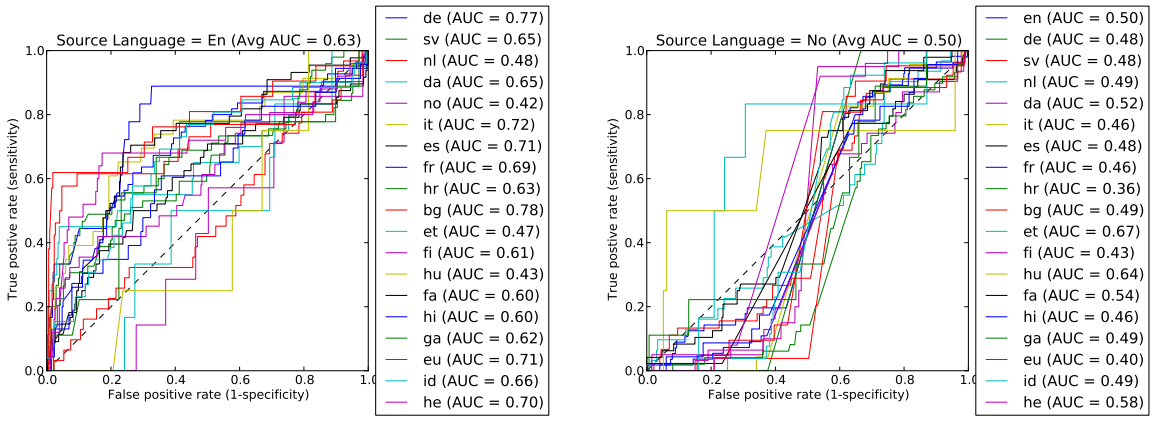
In this work, we measure the likelihood of the triple $\langle \text{pos}_h, \textsf{rel}, \text{pos}_d \rangle$ being an MWE pattern in the target language. The scores are measured according to the respective lexical instances of each triple in the source language, aggregated to compute scores for each triple, and used to train a support vector regression (SVR: Joachims (2006)) model.

The gold-standard labels to train the model are based on the dependency relations: the value is set to 1 if the dependency is `compound` or `mwe`, and 0, otherwise.

We use 8 features in our proposed method: pointwise mutual information (PMI), ϕ-square, the Dice coefficient, student's t test, log-likelihood ratio, pattern fixedness, token/type ratio for a given triple, and token/token ratio across all relations.

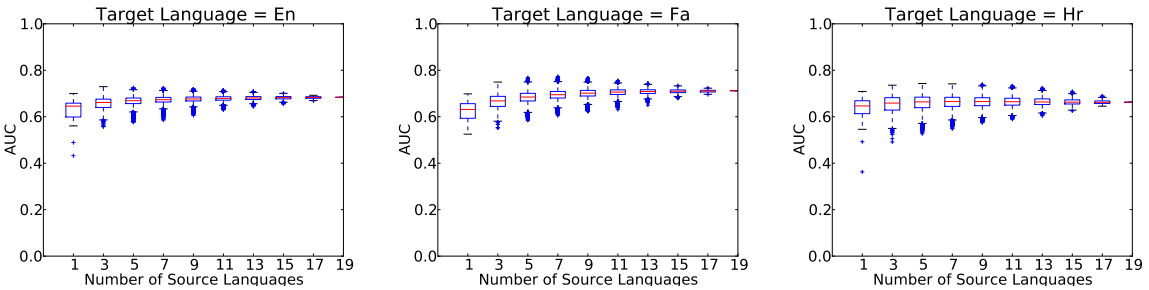
$$
\begin{aligned}
\text{PMI}(x,y) &= \log \frac{p(x,y)}{p(x)p(y)} \\
\phi^2 &= \frac{(n(x,y)n(\bar{x},\bar{y}) - n(x,\bar{y})n(\bar{x},y))^2}{n(x)n(\bar{x})n(\bar{y})n(y)} \\
\text{Dice coefficient} &= \frac{2n(x,y)}{n(x)+n(y)} \\
\text{t}(x,y) &= \frac{\frac{n(x,y)-(n(x)*n(y))}{\text{total words}}}{\sqrt{n(x,y)}}
\end{aligned}
$$

where $n(.)$ is the number of occurrences, and $\bar{x}$ is the number of all instances except x. Pattern fixedness is measured via entropy as $\text{H}(\Pr(D(x,y)))$, where $D(x,y)$ is the difference between the linear position of the head and dependent, binned as follows: $\textit{posdiff} \in \{(-\infty,-2), -2, -1, 1, 2, (2,\infty)\}$.

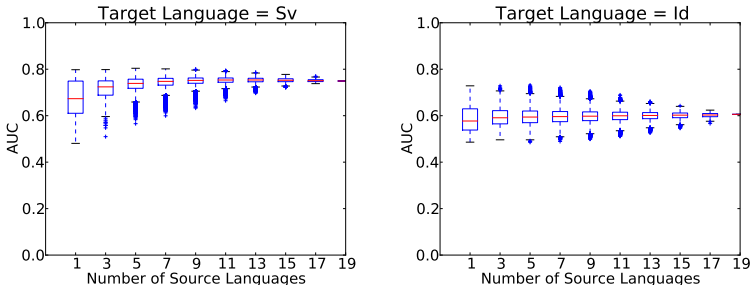


(a) English (b) Norwegian

Figure 2: Selecting a language as a source language



(a) English (b) Persian (c) Croatian



(d) Swedish (e) Indonesian

Figure 3: Distribution of ROC AUC scores when combining source languages

These features are first computed for each lexical instance of a given pattern, and then aggregated to calculate the overall feature values for each triple, using either: (a) the mean (in the case of pattern fixedness); or (b) the median (in the case of the other measures).[4]

After computing the features, we train an SVR model based on the dependency triples in the source language, and then apply the model to rank the triples in the target language. To avoid noisy annotations, we consider only those triples that occur at least twice in each corpus.

We further experiment with a simple ensemble method to combine source languages, in order to smooth over annotation and linguistic differences between languages: we combine the trained rerankers from multiple source languages by calculating the average of the predicted scores from each language.

[4]MWEs components are usually seen in a fixed order and with fixed gap size. We use mean to aggregate the pattern fixedness scores in order to capture any lexical instances which are not used in a fixed order. However, we use the median for the other measures to suppress the impact of outliers. Our preliminary results also confirm that this is the best way to aggregate the scores.

Score	Pattern	Example
0.327	⟨NOUN, ccomp, ADJ⟩	*sure place*
0.306	⟨X, compound, PROPN⟩	*Indo Lanka*
0.301	⟨NOUN, appos, SYM⟩	*$ value*
0.298	⟨ADJ, nmod:npmod, ADV⟩	*little more*
0.295	⟨NOUN, punct, NUM⟩	*5 ”*
0.285	⟨SYM, punct, SYM⟩	*— —*
0.283	⟨AUX, advcl, ADV⟩	*as can*
0.277	⟨NOUN, mwe, SCONJ⟩	*in case*
0.270	⟨SCONJ, mwe, ADP⟩	*due to*
0.268	⟨NOUN, mwe, ADP⟩	*in order*

Table 1: The top predicted MWE patterns in English, by combing all other languages.

6 Results

We report on two experiments. First, we train a model using features extracted from the gold-standard treebank in a given source language, and apply it to features extracted from the gold-standard treebank in a target language. We investigate how well our model is able to find the annotated MWE triples when gold-standard dependency relations are provided. This experiment also shows how our model can be used to find new MWE patterns in existing annotated treebanks (missing certain MWE types). Second, we investigate how our model performs in the more realistic scenario of no annotated treebank being available in the target language.

6.1 Experiment I: Learning given the gold standard treebank

In our first experiment, we assume access to gold standard annotations of POS tags and relation edges in both source and target languages, to determine the tractability of the task, assuming perfect parses.

Since the output of our model is a score in the range $[0, 1]$, we evaluate based on the area under the curve (AUC) from a ROC curve. Figure 2 shows the ROC curve for predicting MWEs when English and Norwegian are the source languages. English is among our 5 selected languages — i.e., one of the languages with the highest number of multiword expression patterns — while for Norwegian, the `mwe` relation is not used at all and only `compound:prt` is annotated. According to these results, the average AUC for predicting MWE patterns is 0.63 when English is the source language (averaged across all target languages, excluding English), while it is 0.50 when Norwegian is the source language. This shows that a source language with less annotated patterns makes for a weaker model. The average scores when our 5 selected languages are used as the source language are remarkably similar: English = 0.63, Persian = 0.64, Croatian = 0.64, Indonesian = 0.61 and Swedish = 0.65.

To investigate further, Figure 3 shows how adding more source languages affects the results for MWE pattern extraction over our 5 selected languages. According to these results, using more than one language can increase the AUC, however, using more than 3 languages does not improve the average AUC greatly.

Finally, we show the top-predicted MWE patterns in English in Table 1. We observe errors such as ⟨NOUN, punct, NUM⟩ and ⟨SYM, punct, SYM⟩, because of their idiosyncratic properties across token instances. However, our model also predicts that ⟨NOUN, ccomp, ADJ⟩ is an MWE.

6.2 Experiment II: Learning without gold standard dependency relations

In our second experiment, we evaluate under the more realistic task setting of there being no gold standard treebank in the target language. Instead, we use the cross-lingual parser proposed by Duong et al. (2015) to parse the corpus in the target language (see Section 2). Note that we still use gold-standard POS tags, but this isn't entirely unrealistic given the relative maturity of methods for inducing universal POS taggers (Das and Petrov, 2011; Täckström et al., 2013; Duong et al., 2014).

Obviously, due to the fact that the parser has no access to dependency annotations in the target language, the parser output will be noisy. However, this emulates a true surprise language setup, where we

Language	De	Sv	Da	It	Es	Fr	Hr	Bg	Hu	Fa	Ga	Eu	Id	He
Baseline	**0.816**	0.511	0.481	0.637	0.546	0.656	0.487	0.554	0.406	0.467	0.586	0.497	0.473	0.472
Reranker	0.736	**0.514**	0.428	0.631	**0.610**	**0.684**	0.461	**0.645**	0.470	**0.549**	0.673	0.578	0.513	**0.622**
PMI	0.804	0.512	**0.567**	**0.803**	0.494	0.634	**0.575**	0.471	**0.588**	0.521	**0.687**	**0.595**	**0.756**	0.476
Baseline + gold	0.797	0.750	0.846	0.885	0.879	0.799	0.696	0.917	0.457	0.919	0.799	0.891	0.931	0.788

Table 2: AUC scores when English is used as the source language to transfer dependency links and to train our reranker model. In "Baseline + gold", the trained model is applied to the gold-standard annotation of the target language rather than the parsed corpus.

have no prior knowledge of MWEs or dependency structure in the target language.

In order to evaluate our proposed method and compare it with the gold standard treebank, we change the evaluation method slightly in order to better reflect the expected inconsistencies in the parser output. In terms of gold-standard labeling, we exhaustively consider every edge between all pairs of tokens in each sentence, and consider an edge to be a positive instance if there is an MWE dependency between its token pairs in the gold-standard treebank, and a negative instance otherwise. To evaluate the parser output, which is the baseline in this experiment, we use the generated dependency edges and labels, and evaluate this against the exhaustively-generated gold-standard. To evaluate our system's performance, we use the dependency edges given by the parser and aggregate the reranker's predicted scores at the level of the delexicalised dependency triples, as per the first experiment. Unlike the first experiment, we evaluate using ROC AUC over the token pairs instead of $\langle \text{pos}_h, \mathsf{rel}, \text{pos}_d \rangle$ triples.

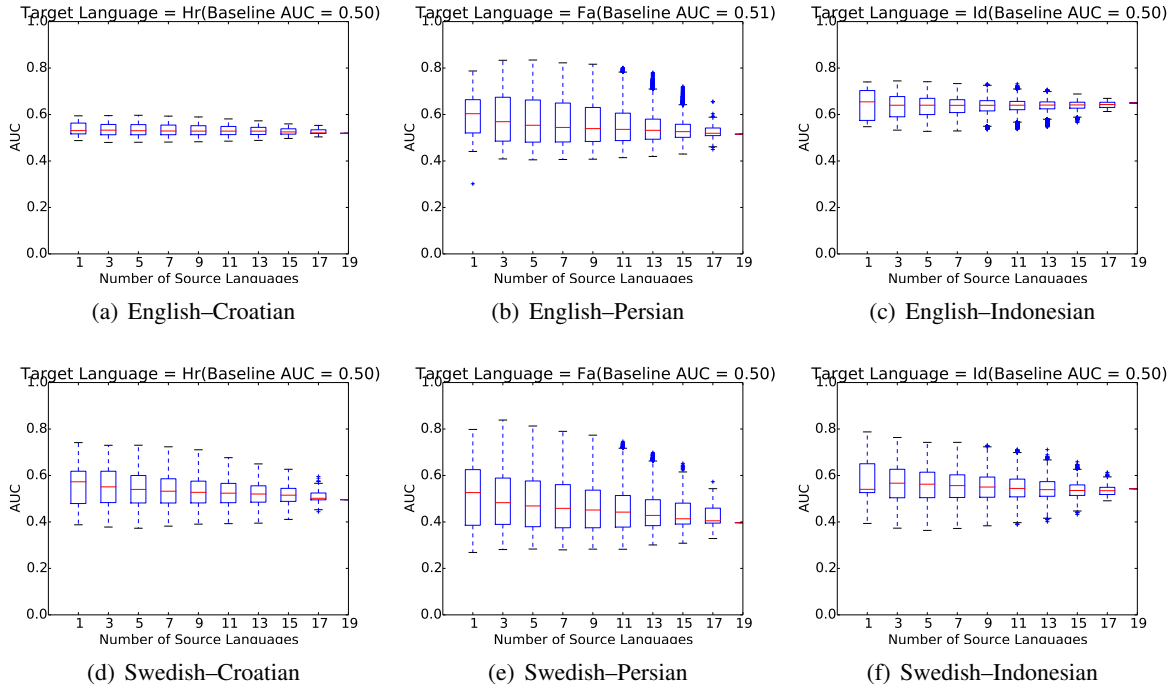
Table 2 shows the AUC scores when English is used as the source language to parse the target language, and English is also used to train our reranking model. Our proposed model ("Reranker") produces above-baseline results for all target languages except German, Danish, Italian and Croatian. We observe a very high percentage (63%) of compounds being predicted as noun–noun compounds in German, which is a large part of the strong results for that language. In order to compare with a collocation extraction methods, we contrast this with a ranking based on the average PMI score for each dependency relation ("PMI"). The results show that for half of the languages simple PMI scores can lead to higher AUC scores, while for the other half, the reranker model (which incorporates PMI scores but is trained on another language), performs better.

The final row in Table 2 is the result of providing the baseline method with gold standard dependency relations (with unknown label, to avoid trivialising the task) and applying the reranker to the gold-standard tuples. Since one source of noise (i.e. the induced parser) is removed in this baseline, we observe much higher scores than the other two approaches, except for Hungarian. For Hungarian, 90% of the annotated MWEs are NUM–NUM compounds, which is the reason that our second baseline performs worse for Hungarian compared to other languages. This result suggests that, unsurprisingly perhaps, the major cause of error in our method is the dependency parser.

Similar to the previous experiment, we also experimented with an ensemble of rerankers. We use English and Swedish as the source language to parse Persian, Croatian and Indonesian. Figure 4 shows how increasing the number of source languages and combining the trained models affects the AUC scores. According to Figure 4, our proposed method on average beats the baseline, when using only one language to train the reranker. However, unlike the previous experiment, combining multiple source languages does not improve the reranker. Additionally, comparing English with Swedish, we observe that the source language used to induce the dependency parser plays an important role.

7 Error Analysis

Finally, we perform error analysis to better understand the performance of the proposed method, focusing on two languages: Persian and Croatian, with 322 and 225 patterns to rank, respectively. We selected these two languages primarily because of the diversity of the MWE annotations in the treebanks (Section 3), and we had access to expert native-speaker annotators. The most frequent annotated patterns in the original treebanks are shown as "Gold standard" in Table 3 (the relation between pos_h and pos_d are either `mwe` or `compound` or both). The dependency parser and reranker are trained on English and



(a) English–Croatian (b) English–Persian (c) English–Indonesian

(d) Swedish–Croatian (e) Swedish–Persian (f) Swedish–Indonesian

Figure 4: Combining source languages given the noisy dependency relations. In (a)–(c), the English treebank is used as the source language for the cross-lingual parser, and in (d)–(f) Swedish is used.

Swedish as the source languages, individually. The top-10 most frequent patterns in the first quartile of the output of the reranker are shown in Table 3. The patterns which match with the gold standard pattern are marked with "†".

When English is the source language, noun compounds are correctly selected as a very frequent pattern in Persian. The pattern of ⟨ADJ, amod, NOUN⟩ is selected as the second most common pattern in Persian, for which almost all token instances are also true MWEs, such as *Islamic republic*, *Islamic revolution*, *right wing* and *fundamental law*.[5] Instances of ⟨NOUN, nmod, NOUN⟩ are more institutionalised, such as *seminar presentation* and *Iftar time*. Persian is rich with LVCs, which shows up in the first column as ⟨NOUN, nsubj, VERB⟩, i.e. misanalysed as verb–subject rather than verb–object pairs, but containing predominantly LVCs. In fact, among the top-20 most frequent token instances of this pattern, 17 are LVCs. As we work our way down the list of dependency triples in Table 3, there are fewer and fewer actual MWE token instances associated with the pattern. For example, the number of MWE instances associated with ⟨ADV, advmod, NOUN⟩ is less than non-MWEs (MWE examples are *before Christ*, and *before revolution*). The primary sources of error were parser errors or the triple being a fragment of a larger MWE. Using Swedish as the source language, we observed a similar trend.

For Croatian, almost all of the tokens associated with the top-2 patterns for both English and Swedish are MWE instances, with the tokens associated with ⟨NOUN, compound, NOUN⟩ based on English corresponding very closely with ⟨NOUN, nmod, NOUN⟩ based on Swedish. As with Persian, as we go down the list, the patterns become more noisy and the MWE tokens sparser, with the exception of ⟨NOUN, compound/nmod, PROPN⟩, for which almost all instances are MWEs (e.g. *president Erdogan*) or part of a larger MWE. Also, the instances of ⟨PROPN, compound, PROPN⟩ are all named entities. None of the token instances associated with ⟨NOUN, nsubj, VERB⟩ and ⟨AUX, aux, VERB⟩ were MWEs.

[5]Note that our model predicts the MWE patterns rather than MWE instances. Therefore, whether an individual MWE is also an MWE in the target language or not, does not affect the final results.

	Source = English	Source = Swedish	Gold standard
Persian	⟨NOUN, compound, NOUN⟩ †	⟨ADP, case, NOUN⟩	⟨NOUN, ∗, VERB⟩
	⟨ADJ, amod, NOUN⟩ †	⟨NOUN, nsubj, VERB⟩ †	⟨ADP, ∗, ADP⟩
	⟨NOUN, nmod, NOUN⟩ †	⟨NOUN, nmod:poss, NOUN⟩ †	⟨ADJ, ∗, VERB⟩
	⟨NOUN, nsubj, VERB⟩ †	⟨NOUN, nmod, VERB⟩ †	⟨NOUN, ∗, NOUN⟩
	⟨NOUN, nmod, ADJ⟩	⟨VERB, acl:relcl, NOUN⟩	⟨NUM, ∗, NOUN⟩
	⟨DET, det, NOUN⟩	⟨ADJ, amod, NOUN⟩ †	⟨CONJ, ∗, PRON⟩
	⟨ADV, advmod, NOUN⟩	⟨CONJ, cc, NOUN⟩ †	⟨CONJ, ∗, CONJ⟩
	⟨NOUN, conj, VERB⟩ †	⟨ADJ, nsubj, VERB⟩ †	⟨ADJ, ∗, NOUN⟩
	⟨NOUN, nmod, VERB⟩ †	⟨DET, det, NOUN⟩	⟨NOUN, ∗, ADP⟩
	⟨NOUN, conj, SCONJ⟩	⟨NUM, nummod, NOUN⟩ †	⟨CONJ, ∗, NOUN⟩
Croatian	⟨ADJ, amod, NOUN⟩ †	⟨ADJ, amod, NOUN⟩ †	⟨PRON, ∗, VERB⟩
	⟨NOUN, compound, NOUN⟩ †	⟨NOUN, nmod, NOUN⟩ †	⟨ADJ, ∗, NOUN⟩
	⟨ADP, case, NOUN⟩ †	⟨ADP, case, NOUN⟩ †	⟨NOUN, ∗, NOUN⟩
	⟨NOUN, nmod, NOUN⟩ †	⟨NOUN, dobj, VERB⟩	⟨PROPN, ∗, PROPN⟩
	⟨NOUN, dobj, VERB⟩	⟨NOUN, nmod, PROPN⟩	⟨PROPN, ∗, NOUN⟩
	⟨NOUN, compound, PROPN⟩	⟨NOUN, nmod:poss, NOUN⟩ †	⟨NUM, ∗, NOUN⟩
	⟨NOUN, nmod, VERB⟩	⟨NOUN, nmod, VERB⟩	⟨ADP, ∗, NOUN⟩
	⟨PROPN, compound, PROPN⟩ †	⟨NOUN, nsubj, VERB⟩	⟨X, ∗, X⟩
	⟨NOUN, nsubj, VERB⟩	⟨AUX, aux, VERB⟩	⟨PRON, ∗, ADP⟩
	⟨NOUN, conj, NOUN⟩ †	⟨PRON, nsubj, VERB⟩ †	⟨PRON, ∗, SCONJ⟩

Table 3: Top-ranking Persian and Croatian MWE patterns extracted using English and Swedish as the source language. Those patterns which match the top-ranking gold standard patterns are shown with "†".

8 Conclusion

In this paper, we proposed a method for automatically determining the MWE composition of a novel language, based on delexicalised universal dependency patterns of the form $\langle \text{pos}_h, \textsf{rel}, \text{pos}_d \rangle$. The method is based on determination of MWEs in a source language from a dependency treebank, and training of a model over delexicalised dependency patterns for that language. This is then applied to a target language to rerank patterns, in terms of MWEhood. In our initial experiments, we used gold-standard dependency information for a treebank for the target language, and found the method to be highly successful at ranking dependency patterns. This both validated the method, as well as suggesting the potential for the use of the method in cross-checking the consistency of the UD treebanks. We then applied our method under the more realistic setting of having no gold-standard dependency data for the target language, but instead the output of a dependency parser induced for the target language based only on a POS-tagged monolingual corpus in the target language (and gold-standard data in the source language). We found the method to produce above-baseline results for the majority of languages tested, and that for the false positives associated with higher token frequencies, many of the associated tokens were actually true instances of MWEs (with the wrong dependency relation).

Acknowledgements

We wish to thank Long Duong for help with the transfer-based dependency parsing, Jan Snajder for his kind assistance with the Croatian annotation, and Dan Flickinger, Lars Hellan, Ned Letcher and João Silva for valuable advice in the early stages of development of this work. We would also like to thank the anonymous reviewers for their insightful comments and valuable suggestions. NICTA is funded by the Australian government as represented by Department of Broadband, Communication and Digital Economy, and the Australian Research Council through the ICT Centre of Excellence programme.

References

Josep Alba-Salas. 2002. *Light Verb Constructions in Romance. A Syntactic Analysis*. Ph.D. thesis, Cornell University.

Timothy Baldwin and Su Nam Kim. 2010. Multiword expressions. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing*. CRC Press, Boca Raton, USA, 2nd edition.

Timothy Baldwin and Aline Villavicencio. 2002. Extracting the unextractable: A case study on verb-particles. In *Proceedings of the 6th Conference on Natural Language Learning (CoNLL-2002)*, pages 98–104, Taipei, Taiwan.

Timothy Baldwin, John Beavers, Leonoor van der Beek, Francis Bond, Dan Flickinger, and Ivan A. Sag. 2004. In search of a systematic treatment of determinerless PPs. In Patrick Saint-Dizier, editor, *Computational Linguistics Dimensions of Syntax and Semantics of Prepositions*. Kluwer Academic, Dordrecht, Netherlands.

Timothy Baldwin. 2005. The deep lexical acquisition of English verb-particle constructions. *Computer Speech and Language, Special Issue on Multiword Expressions*, 19(4):398–414.

James Breen and Timothy Baldwin. 2009. Corpus-based extraction of Japanese compound verbs. In *Proceedings of the Australasian Language Technology Workshop 2009 (ALTW 2009)*, pages 35–43, Sydney, Australia.

Julian Brooke, Vivian Tsang, Graeme Hirst, and Fraser Shein. 2014. Unsupervised multiword segmentation of large corpora using prediction-driven decomposition of n-grams. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 753–761, Dublin, Ireland.

Miriam Butt. 2003. The light verb jungle. In *Proceedings of the Workshop on Multi-Verb Constructions*, pages 1–49, Trondheim, Norway.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar.

Matthieu Constant, Anthony Sigogne, and Patrick Watrin. 2012. Discriminative strategies to integrate multiword expression recognition and parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 204–212, Jeju Island, Korea.

Ann Copestake. 2003. Compounds revisited. In *Proceedings of the 2nd International Workshop on Generative Approaches to the Lexicon*, Geneva, Switzerland.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 600–609, Portland, USA.

Mona T. Diab and Pravin Bhutada. 2009. Verb noun construction MWE token supervised classification. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*, pages 17–22, Singapore.

Long Duong, Trevor Cohn, Karin Verspoor, Steven Bird, and Paul Cook. 2014. What can we get from 1000 tokens? a case study of multilingual POS tagging for resource-poor languages. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 886–897, Doha, Qatar.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Cross-lingual transfer for unsupervised dependency parsing without parallel data. In *Proceedings of the 19th Conference on Natural Language Learning (CoNLL-2015)*, pages 113–122, Beijing, China.

Stefan Evert and Brigitte Krenn. 2005. Using small random samples for the manual evaluation of statistical association measures. *Computer Speech and Language, Special Issue on Multiword Expressions*, 19(4):450–466.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2013)*, pages 138–147, Atlanta, USA.

Fei Huang, Stephan Vogel, and Alex Waibel. 2003. Automatic extraction of named entity translingual equivalence based on multi-feature cost minimization. In *Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-language Named Entity Recognition*, Sapporo, Japan.

Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, Philadelphia, USA.

Gholam Hossein Karimi-Doostan. 1997. *Light Verb Construction in Persian*. Ph.D. thesis, University of Essex.

Brigitte Krenn and Stefan Evert. 2001. Can we do better than frequency? A case study on extracting PP-verb collocations. In *Proceedings of the ACL/EACL 2001 Workshop on the Computational Extraction, Analysis and Exploitation of Collocations*, pages 39–46, Toulouse, France.

Mirella Lapata and Alex Lascarides. 2003. Detecting novel compounds: The role of distributional evidence. In *Proceedings of the 11th Conference of the European Chapter for the Association of Computational Linguistics (EACL-2003)*, pages 235–242, Budapest, Hungary.

Linlin Li and Caroline Sporleder. 2010. Linguistic cues for distinguishing literal and non-literal usages. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010), Posters Volume*, pages 683–691, Beijing, China.

Diana Maynard, Valentin Tablan, Kalina Bontcheva, and Hamish Cunningham. 2003. Rapid customization of an information extraction system for a surprise language. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(3):295–300.

I. Dan Melamed. 1997. Automatic discovery of non-compositional compounds in parallel data. In *Proceedings of the Fifth Workshop on Very Large Corpora*. EMNLP.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at the International Conference on Learning Representations, 2013*, Scottsdale, USA.

Begona Villada Moirón and Jörg Tiedemann. 2006. Identifying idiomatic expressions using automatic word-alignment. In *Proceedings of the EACL 2006 Workshop on Multi-wordexpressions in a multilingual context*, pages 33–40.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 629–637, Jeju Island, Korea.

David Newman, Nagendra Koilada, Jey Han Lau, and Timothy Baldwin. 2012. Bayesian text segmentation for index term identification and keyphrase extraction. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 2077–2092, Mumbai, India.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*.

Douglas W Oard. 2003. The surprise language exercises. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(2):79–84.

Diarmuid Ó Séaghdha. 2008. *Learning compound noun semantics*. Ph.D. thesis, Computer Laboratory, University of Cambridge.

Pavel Pecina. 2008. *Lexical Association Measures: Collocation Extraction*. Ph.D. thesis, Faculty of Mathematics and Physics, Charles University in Prague, Prague, Czech Republic.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 2089–2096, Istanbul, Turkey.

Carlos Ramisch. 2012. A generic framework for multiword expressions treatment: from acquisition to applications. In *Proceedings of ACL 2012 Student Research Workshop*, pages 61–66, Jeju Island, Korea.

Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings of the 3rd International Conference on Intelligent Text Processing Computational Linguistics (CICLing-2002)*, pages 189–206, Mexico City, Mexico.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014. Discriminative lexical semantic segmentation with gaps: Running the mwe gamut. *Transactions of the Association of Computational Linguistics*, 2:193–206.

Sabine Schulte im Walde, Stefan Müller, and Stefan Roller. 2013. Exploring vector space models to predict the compositionality of German noun-noun compounds. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (*SEM 2013)*, pages 255–265, Atlanta, USA.

Sabine Schulte im Walde. 2004. Identification, quantitative description, and preliminary distributional analysis of German particle verbs. In *Proceedings of the COLING Workshop on Enhancing and Using Electronic Dictionaries*, pages 85–88, Geneva, Switzerland.

Ralf Steinberger and Bruno Pouliquen. 2007. Cross-lingual named entity recognition. *Lingvisticæ Investigationes*, 30(1):135–162.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2013)*, pages 1061–1071, Atlanta, USA.

Takaaki Tanaka and Timothy Baldwin. 2003. Noun-noun compound machine translation a feasibility study on shallow processing. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, pages 17–24, Sapporo, Japan.

Kiyoko Uchiyama, Timothy Baldwin, and Shun Ishizaki. 2005. Disambiguating Japanese compound verbs. *Computer Speech and Language, Special Issue on Multiword Expressions*, 19(4):497–512.

Leonoor van der Beek. 2005. The extraction of determinerless PPs. In *Proceedings of the Second ACL-SIGSEM Workshop on the Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*, pages 190–199, Colchester, UK.

Sriram Venkatapathy and Aravind Joshi. 2006. Using information about multi-word expressions for the word-alignment task. In *Proceedings of the COLING/ACL 2006 Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 53–60, Sydney, Australia.

A Hybrid Deep Learning Architecture for Sentiment Analysis

Md Shad Akhtar, Ayush Kumar, Asif Ekbal, Pushpak Bhattacharyya
Department of Computer Science & Engineering
Indian Institute of Technology Patna, India
{shad.pcs15,ayush.cs12,asif,pb}@iitp.ac.in

Abstract

In this paper, we propose a novel hybrid deep learning architecture which is highly efficient for sentiment analysis in resource-poor languages. We learn sentiment embedded vectors from the Convolutional Neural Network (CNN). These are augmented to a set of optimized features selected through a multi-objective optimization (MOO) framework. The sentiment augmented optimized vector obtained at the end is used for the training of SVM for sentiment classification.We evaluate our proposed approach for coarse-grained (i.e. sentence level) as well as fine-grained (i.e. aspect level) sentiment analysis on four Hindi datasets covering varying domains. In order to show that our proposed method is generic in nature, we also evaluate it on two benchmark English datasets. Evaluation shows that performance of the proposed method are consistent across all the datasets and often outperform the state-of-art systems. To the best of our knowledge, this is the very first attempt where such a deep learning model is used for sentiment analysis in less-resourced languages such as Hindi.

1 Introduction

Sentiment Analysis (Pang and Lee, 2008) in natural language processing (NLP) deals with the problem of identifying the polarity in a user generated content. With growing social media platforms such as Twitter, Facebook etc., copious amount of data is being generated continuously. According to Domo's Data Never Sleep 2.0[1], the global internet population is about 2.4 billion users. Online platforms such as Twitter alone generate over 300,000 tweets per minute[2]. At the same time more than 26K user reviews are posted on Yelp, an online user review portal. This tremendous amount of semi-structured data poses a great challenge in its efficient processing for any specific purpose. Sentiment analysis for web generated content e.g. tweets and online reviews, is a cumbersome problem mainly due to its unstructured and noisy nature (e.g. *gr8*, *g8* etc. for *great*) and spelling and grammatical mistakes. Considering the challenges as mentioned above, authors have proposed their sentiment analyzers for Twitter data and/or online reviews (Kim and Hovy, 2004; Mohammad et al., 2013a; Gupta et al., 2015). However, most of the works have been done on the resource-rich languages such as English.

India is a multi-lingual country with great linguistic and cultural diversities. There are 22 officially spoken languages. However, there have not been enough research works that address sentiment analysis involving Indian languages, *except* few such as (Balamurali et al., 2012; Bakliwal et al., 2012; Kumar et al., 2015). However, these existing works do not address the fine-grained sentiment analysis at the aspect level. The prime reason behind this is the scarcity of benchmark datasets and other resources/tools in Indian languages. In our work, we focus on sentiment analysis in Hindi, the official language of India and the fourth most spoken language all over in the world. We make use of benchmark datasets released as part of a shared task on sentiment analysis in Indian languages (SAIL) for Twitter (Patra et al., 2015). Recently, we (Akhtar et al., 2016) have created a dataset for aspect based sentiment analysis

[1] https://www.domo.com/learn/data-never-sleeps-2

[2] http://aci.info/2014/07/12/the-data-explosion-in-2014-minute-by-minute-infographic/

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 482–493, Osaka, Japan, December 11-17 2016.

(ABSA) (Pontiki et al., 2014) in Hindi. For sentence-level sentiment analysis we annotate these same set of reviews. Here, we evaluate our proposed approach for both coarse-grained (sentence based) and fine-grained (aspect based) sentiment analysis.

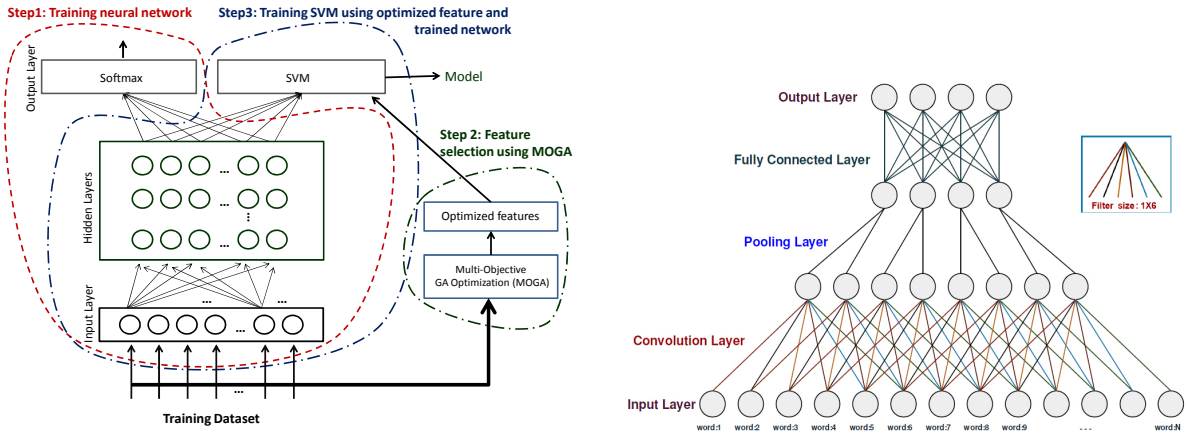
Our proposed method is based on deep learning, which has shown its premise in various NLP problems including sentiment analysis. Authors worldwide have proposed many variants of its architecture (Kim, 2014; dos Santos and Gatti, 2014), which have shown success for solving problems in varying domains. Most of these works employ traditional technique of using softmax as an activation function on top of a typical convolutional neural network (CNN). However, in our work we learn *sentiment embedded vectors* using CNN pipeline and perform final classification using a strong classifier, Support Vector Machine (SVM) (Vapnik, 1995). Replacing softmax layer with some stronger classifier might be useful as shown in very few research, such as computer vision (Tang, 2013) and NLP (Poria et al., 2015).

In this work, we do not use the traditional pipeline of CNN (c.f. Section 2.1) for sentiment classification. Rather, we learn sentiment features through CNN, which we call as '*sentiment-embedded vector*'. Parallely, a multi-objective optimization (MOO) based framework using Genetic Algorithm (GA) (Deb et al., 2002) is employed to derive optimized features for the respective optimization functions. In the final step, we augment the sentiment-embedded vector with the optimized feature set to form '*sentiment augmented optimized vector*'. This vector is used as the feature for sentiment classification using a non-linear SVM. In order to study the impact of external optimized handcrafted features, we build different models of the baseline systems. The existing works which make use of external features in CNN architecture simply append features at the input layer. This method has mainly three drawbacks: **(i)** The information in external features appended at the input layer are not properly reflected in the output due to the processes of convolution and pooling layers. **(ii)** The set of features is not optimized i.e, optimal subset of features is not extracted, instead complete feature set is appended to the word representations at the input layer. **(iii)** Softmax is a weak classifier and has limitation over SVM. We propose to tackle all these problems using our approach, the results of which are encouraging and consistent across datasets of varying domains and languages. Such hybrid model using CNN, SVM and MOGA (c.f. Section 2.3) that performs sentiment classification using sentiment augmented optimized vector is novel, impactful as well as very effective for resource-constrained languages.

We summarize the main contributions of the proposed approach as follows: **i)** a hybrid modified architecture of CNN, that learns sentiment embedded vector instead of traditional pipelined-classification; **ii)** application of MOO for the systematic selection of optimized feature set, to generate sentiment augmented optimized vector; **iii)** replacement of softmax layer to produce more robust hybrid deep learning network by using non-liner SVM based classification at the final step; and **iv)** generic approach, applicable to different languages and domains. We evaluate the approach on the datasets of varying domains, i.e. Twitter (generic as well as sarcastic) and online product reviews (sentence-level and aspect-level), across two different languages *viz.* Hindi and English for sentence-level as well as aspect-level sentiment analysis. Experiments show that the proposed hybrid deep learning architecture is highly efficient for sentiment analysis in multiple domains for Hindi. To the best of our knowledge, this is the very first attempt of using such a hybrid deep learning model for sentiment analysis, especially in less-resource languages. For English, we use the benchmark dataset of SemEval-2015 shared task on sentiment analysis in Twitter (Rosenthal et al., 2015) and SemEval-2014 shared task on aspect based sentiment analysis (Pontiki et al., 2014).

2 Methodology

Logistic regression (LR) (or Softmax regression for multi-class classification) and SVM are two algorithms that often produce comparable results. However, SVM has an edge over LR if the data is not linearly separable, i.e. SVM with non-linear kernel performs better than LR (Pochet and Suykens, 2006). Also, LR focuses on maximizing the likelihood and is prone to over-fitting. However, SVM finds a linear hyperplane by projecting input data into higher dimension and generalizes well. We incorporate this idea in our proposed research by replacing the softmax regression with SVM at the output layer of CNN. The motivation for using CNN architecture are two-fold: (i) The system can learn hidden semantics from a



(a) Proposed methodology

(b) A typical architecture of CNN

Figure 1: (a) Proposed methodology. (b) A typical architecture of CNN.

large unlabeled corpus, and (ii) limited coverage of lexical resources (Hindi SentiWordNet). The proposed approach, *CNN-SVM*$_{W+X}$, operates in three steps (Figure: 1a; red, green & blue dotted lines show the processes of Step 1, 2 and 3, respectively.):

1. Learning sentiment embedded vector using CNN architecture;
2. Generation of sentiment augmented optimized vector using a multi-objective GA (MOGA) based optimization technique; and
3. Training of SVM with non-linear kernel utilizing the network trained in first step and optimized features of Step 2.

In Step 1, we define the network and initialize its weights using Xavier initialization (Glorot and Bengio, 2010). We then train a CNN using a stochastic gradient descent back-propagation algorithm. Parallely, in Step 2, MOO based feature selection technique is employed to identify the most relevant set of features within the framework of SVM. Once the training of CNN is over, i.e. optimal parameters of the network are found, in Step 3 we concatenate the output of top hidden layer and optimized feature set reported by MOGA and feed it to SVM.

CNN performs reasonably well in capturing the relevant lexical and syntactic features on its own. Thus, the first step of the proposed approach ensures that it extracts such features from the training data automatically. The SVM in the proposed approach makes use of the features extracted from CNN along with the optimized features (from MOGA) to define a hyperplane which is more robust as compared to what defined by either CNN or SVM with optimized features alone. The pseudo code of the proposed approach is sketched in Algorithm 1. Statements 1-6 deal with the first step i.e. training of the deep learning network to learn sentiment embedded vectors while statement 7 finds out the optimized feature set. The last step is carried out by statements 8-14.

2.1 Convolutional neural network (CNN)

CNN is a special kind of multi-layer neural network which consists of one or more convolutional and pooling layers, followed by one or more fully-connected layers. The convolutional and pooling layers implicitly extract relevant feature representation from input data, and fed it to the fully connected layers for classification. The size and weights of the convolution filters determine the features to be extracted from the input data. Same convolution filter is floated over the complete input data in order to extract similar features at different spatial locations. Max pool layer is then applied to select the most significant features from the CNN features. Subsequently, after iterating several convolutional and max pooling layers, it is fed to a fully connected layer for classification. In general we use softmax as an activation

Algorithm 1 (*Pred*, *Acc*) = *CNN-SVM*$_{(W+X)}$ *(Train, Dev, Test, Test-Gold, θ)*

Require: *Train, Dev, Test, Test-Gold* - Datasets; θ - Termination criteria.
Ensure: *Pred* - Predicted output; *Acc* - Accuracy achieved.
1: $Net \leftarrow$ BuildNetwork()
2: InitializeNetwork(Net)
3: **for** $error >= \theta$ **do**
4: $error \leftarrow$ TrainNetwrok($Net, Train, Dev$)
5: **end for**
6: /* Training complete */
7: $Feature_{opt} \leftarrow$ MOGA($Train, Dev$)
8: $H_{Train} \leftarrow$ GetTopHiddenLayer($Net, Train$)
9: $Train_{combined} \leftarrow H_{Train} + Feature_{opt}$
10: $Model_{SVM} \leftarrow$ SVM$_{\text{Train}}$($Train_{combined}$)
11: $H_{Test} \leftarrow$ GetTopHiddenLayer($Net, Test$)
12: $Test_{combined} \leftarrow H_{Test} + Feature_{opt}$
13: $Pred \leftarrow$ SVM$_{\text{Test}}$($Model_{SVM}, Test_{combined}$)
14: $Acc \leftarrow$ Evaluation*(Test-Gold, Pred)*
15: **return** (Pred, Acc)

function in the fully connected layer. A typical CNN architecture is shown in Figure 1b. Feature map represents the size of the filter while each edge corresponds to a weight of the filter.

2.2 Word representation

A neural network requires word embedding (or, sentence embedding) as an input to the network, i.e. a vector representation of each word or sentence. We use word2vec tool (Mikolov et al., 2013) which efficiently captures the semantic properties of words in the corpus. We train with a corpus of 6.7 million sentences, which were collected from Wikipedia and Twitter sources. This trained model is used for translating a word into its respective vector representation. We set the vector dimension of a word to 200. Each sentence is padded with zero vectors in order to make its length uniform throughout the dataset. Hence, the vector dimension ($Vector_{dim}$) of each sentence (i.e. number of neurons at input layer) counts to 200$\times$max-sentence-length.

2.3 Multi-Objective genetic algorithm (MOGA) based feature selection

We develop a feature selection technique based on multi-objective optimization (MOO) (Deb, 2001). The problem of feature selection can be modeled as follows: Given a set of features F and $M = \langle m_1, m_2, .., m_M \rangle$ objective functions, find a subset F^* of F such that M objectives are optimized simultaneously. For instance, maximization of all objective functions can be mathematically stated as:

$$Objective_M(F^*) = \underset{M, S \epsilon F}{argmax}\{Objective_M(S)\}$$

We use a binary version of genetic algorithm (GA) for determining the best fitting feature set. The basic operations of GA are 'crossover', 'mutation' and 'selection'. First, we randomly initialize N chromosomes of length n, each representing a solution in the population. The length of each chromosome (n) corresponds to the number of features available, i.e. each bit position encodes exactly one feature. The value of 1 in a bit position denotes that the respective feature is used for classifier's training, otherwise the feature is not used. A representation of a chromosome is presented in Figure 2. Selection, crossover

1	1	0	.	.	.	.	0	0	1
F_1	F_2	F_3	.	.	.	.	F_{n-2}	F_{n-1}	F_n

Figure 2: Representation of chromosome in GA based optimization.

and mutation operations are then performed on the chromosomes.

1. **Selection:** At first we select top N solutions w.r.t fitness value. For fitness computation, we construct a SVM based classifier with the selected features, and iterate this process 5 times for 5-fold

cross-validation experiments. In multi-objective optimization we perform non-dominating sorting for the selection. Two solutions, *A* & *B*, are non-dominated to each other if solution *A* is not bad than solution *B* in at least one objective function and vice-verse. In contrast, a solution *A* dominated by *B* if for all objective functions *A* is less optimal than solution *B*. A set of solutions, that are non-dominating to each other but dominates every other solutions in the population forms a non-dominating *front-0*. Similarly, non-dominating *front-1* consists of remaining solutions that are non-dominating to each other but dominate other solutions. Hence, *front-0* solutions dominates *front-1* solutions which in turns dominates *front-2* solutions and so on. Set of solution in *front-0* forms pareto-optimal surface (Rank 1). A pictorial representation of non-dominating solutions are depicted in Fig. 3. We use binary tournament selection, as in non-dominated sorting GA (NSGA)-II (Deb et al., 2000). We use elitism operation, where non-dominated solutions among parent and child generations are propagated to the next generation. MOO provides a set of non-dominated solutions (Deb et al., 2002) on the final Pareto optimal front. Although each of these solutions is equally important from the algorithmic point of view, but user may often require to produce only a single solution. In our case we select the particular solution that yields maximum accuracy.

2. **Crossover:** In crossover, for any two solutions a random split position is chosen. Two new solutions are generated by swapping the information of the chromosomes with each other at the split point.

3. **Mutation:** Similarly, mutation operator is applied to each entry of chromosome, where an entry is randomly replaced by 0 or 1 based on mutation probability.

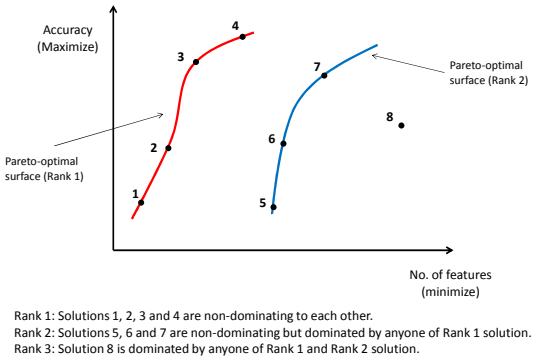


Figure 3: Representation of dominated and non- dominated solutions.

In this work we optimize two objective functions: accuracy (maximize) and number of features (minimize). We set the parameters of MOO as follows: population size=60, number of generations=30, crossover probability=0.8, mutation probability=0.03.

3 Datasets, Experiments and Analysis

3.1 Datasets

For experiments we use the following four datasets for Hindi:

1. **Twitter-Hindi ($Twitter_H$):** We use benchmark dataset released by the organizers of 'SAIL: Sentiment Analysis in Indian Languages' task (Patra et al., 2015).

2. **Online reviews for aspect based sentiment analysis in Hindi ($Review_{A_H}$)[3]:** This dataset is developed by us (Akhtar et al., 2016) for aspect based sentiment analysis (ABSA). It comprises of 5,417 product and service reviews across 12 domains. Reviews are annotated with aspect terms along with

its polarities. We consider four classes, namely *positive*, *negative*, *neutral* and *conflict*. In this work we solve only one problem of ABSA i.e. aspect term sentiment problem.

3. **Online reviews for sentence based sentiment analysis in Hindi ($Review_{S_H}$)[3]:** There is no available benchmark dataset which deals with sentence-level sentiment analysis for online product reviews in Hindi. Therefore, we extract user reviews from $Review_{A_H}$ dataset and annotate these using four classes as mentioned above.

4. **Online movie reviews-Hindi ($Movie_H$)[3]:** We collect user reviews from various news and blog websites, and annotate using four classes.

Detailed statistics of the above datasets are presented in Table 1. For each dataset $Review_{A_H}$, $Review_{S_H}$ and $Movie_H$ we distribute 70%, 20% and 10% of the data as training, test and development, respectively. For generalization, we also evaluate the proposed method on two other benchmark datasets in English *viz.* SemEval 2015 shared task on sentiment analysis in twitter (Rosenthal et al., 2015) and SemEval-2014 shared task on ABSA (Pontiki et al., 2014).

Datasets		**Sentiment**				
		#Pos	**#Neg**	**#Neu**	**#Con**	**Total**
$Twitter_H$	Train	168	559	494	-	1221
	Test	166	251	50	-	467
$Review_{A_H}$	-	1986	569	1914	40	4509
$Review_{S_H}$	-	2290	712	2226	189	5417
$Movie_H$	-	823	530	598	201	2152

Table 1: Dataset statistics. Here, pos: positive, neg:negative, neu:neutral and con:conflict

3.2 Baseline, proposed model and its variants

In order to compare our proposed approach, we define the following baseline models:

- B_{SVM}: This is a SVM based model that incorporates all the available features.

- B_{CNN_W}: It is a simple CNN based model, trained and evaluated using word embedding as features (c.f. Section 2.2).

In addition, we also try to understand the behavior of the proposed model in presence or absence of extra handcrafted features. For a comparative study we define following two models based on CNN architecture:

- CNN-SVM$_W$: This represents our proposed model in the absence of optimized feature set. It is trained and evaluated only with the word embedding features. We extract feature vectors from the top hidden layer and feed it to SVM for training.

- $B_{CNN_{(W+X)}}$: This model is similar to baseline B_{CNN_W}. The only difference is the usage of optimized features as determined by MOO based feature selection technique (in addition to word embedding).

3.3 Feature set

Table 2 shows the set of features that we use for building different models and the optimized feature subset that we obtain through the feature selection technique.

[3]Resource available at http://iitp.ac.in/~ai-nlp-ml/resources.html

Category	Dataset	Feature	Description
Lexical and syntactic features	All	PoS, Word N-grams, Character N-grams	Part-of-Speech tag, Unigram, Bigram and Trigram
Twitter specific features	$Twitter_H$, $Twitter_E$	Hashtags	Number of hashtag(#) tokens in the tweet.
		Emoticons	Binary valued feature denotes the presence or absence of the positive and negative emoticons
		Punctuation	Number of occurrences of contiguous sequence of question marks, exclamation marks etc
		URL and Username	# of url and usernames present in the tweet.
		Average length	Average length of the tokens
Lexicon features	$Review_{A_H}$, $Review_{S_H}$, $Movie_H$	SentiWordNet for Indian Language (Das and Bandyopadhyay, 2010)	# of positive tokens, negative tokens and average score.
	$Review_{A_H}$, $Review_{S_H}$, $Movie_H$, $Review_{A_E}$	Semantic Orientation (Hatzivassiloglou and McKeown, 1997)	Sum of semantic orientation score of each token.
	$Twitter_E$, $Review_{A_E}$	Bing Liu lexicon (Ding et al., 2008)	# of positive tokens and negative tokens.
		MPQA lexicon(Wiebe and Mihalcea, 2006)	Number of positive tokens and negative tokens.
	$Twitter_E$	NRC lexicons (Mohammad et al., 2013b; Mohammad and Turney, 2013)	# of tokens with positive score, negative score and zero score, total emoticons score and total sentiment score.

Dataset	Optimized feature set*
$Twitter_H$	Emoticons, Punctuation, SentiWordNet
$Review_{A_H}$, $Review_{S_H}$	Semantic Orientation
$Movie_H$	Semantic Orientation, SentiWordNet
$Twitter_E$	HashTag, Emoticons, Punctuation, Bing Lui and NRC Lexicon
$Review_{A_E}$	Bing Lui and MPQA Lexicon
*We leave out lexical and syntactic features from the optimized set as these information will be captured by the CNN itself.	

Table 2: Feature set and the optimized features

3.4 Experiments

For experiments we use DL4J[4], a java based package for deep learning implementation, and LibSVM library (Chang and Lin, 2011) for SVM. We use the development set to fine-tune the parameters of CNN. For SVM, we perform grid search to find the optimal parameter settings of RBF kernel. CNN classifier is trained for 50, 100 and 120 epochs with 300 feature maps of size $4 \times Vector_{dim}$. We use stochastic gradient descent and negative log-likelihood as the optimization algorithm and loss function, respectively. In addition, we use L2 regularization and dropout technique (Srivastava et al., 2014) to build a robust system. Results of the proposed method along with the baselines are presented in Table 3a. Our proposed method achieves 62.52% accuracy for $Twitter_H$ which convincingly outperforms SVM based baseline by 13 points, and reports approximately 2 points better accuracy as compared to B_{CNN_W}. Comparison to the participating systems of SAIL shared task shows that we are ahead of the best system as reported in (Se et al., 2015) by almost 7 points. For aspect-level Hindi review dataset, $Review_{A_H}$ the proposed approach reports an accuracy of 65.96% against 54.09% as reported in our previous attempt (Akhtar et al., 2016), which was based only on SVM. Similarly for sentence-level sentiment analysis on $Review_{S_H}$ and $Movie_H$ datasets, our proposed method performs better compared to the other baselines. Since evaluation on $Review_{S_H}$ and $Movie_H$ datasets are performed for the first time, we do not have any existing model for comparison. In Table 3b, we show the class-wise accuracies of *CNN-SVM*$_{(W+X)}$ for the Hindi datasets.

3.4.1 Effect of handcrafted features

Since the twitter-specific features are not very relevant for the product reviews dataset, we only use lexicon-based features for it. In comparison to the baseline B_{CNN_W}, augmenting external features in $B_{CNN_{(W+X)}}$ shows better accuracy. We also observe similar phenomenon for all the other settings. Addition of extra feature helps *CNN-SVM*$_{(W+X)}$ for $Twitter_H$ to achieve accuracy of 62.45% as compared

[4]http://deeplearning4j.org/

to 61.24% without it. Similarly, for $Review_{A_H}$ dataset *CNN-SVM*$_{(W+X)}$'s improvement is close to 5% by just using the sentiment lexicon features. Since word embeddings are good at capturing the semantic information, addition of lexicon based features assist it in finding the sentiment more accurately. It should be noted that augmentation of external features along with the features automatically extracted from CNN at the penultimate layer (sentiment augmented optimized vector) yields better result compared to the model where external features added to the word embeddings at the very input layer. This can be attributed to the fact that information in external features are lost through a series of convolution and max-pooling layers. While we add all features to B_{SVM} in the network, we observe that performance drops. This could be because the network itself captures lexical features on its own, and augmenting features further leads to over-fitting.

Method	**Accuracy**			
	$Twitter_H$	$Review_{A_H}$	$Review_{S_H}$	$Movie_H$
B_{SVM}	49.02	54.07	51.52	38.76
B_{CNN_W}	60.60	59.13	55.12	40.31
CNN-SVM$_W$	61.24	59.26	56.47	41.70
$B_{CNN_{(W+X)}}$	61.89	59.53	55.56	41.40
(Se et al., 2015)	55.60	-	-	-
Previous system: (Kumar et al., 2015), (Akhtar et al., 2016)	46.25	54.09	-	-
CNN-SVM$_{(W+X)}$	**62.52**	**65.96**	**57.34**	**44.88**

(a) Overall performance

Class	**Accuracy**			
	$Twitter_H$	$Review_{A_H}$	$Review_{S_H}$	$Movie_H$
Positive	24.69 (41/166)	67.43 (265/393)	65.77 (294/447)	87.71(150/170)
Negative	88.84 (223/251)	58.94 (89/151)	27.64 (47.170)	23.58 (25/106)
Neutral	56.0 (28/50)	70.46 (241/342)	65.71 (276/420)	21.68 (18/83)
Conflict	-	00.00 (0/16)	8.6 (4/46)	00.00 (0/70)
Total	**62.52 (292/467)**	**65.96 (595/902)**	**57.34 (621/1083)**	**44.88 (913/430)**

(b) Class-wise performance

Table 3: Results of baseline models and proposed method for $Twitter_H$, $Review_{A_H}$, $Review_{S_H}$ and $Movie_H$ datasets. Subscript *W+X* represents models with word embedding and optimized feature set while *W* represent models with only word embeddings. B_{SVM} and B_{CNN_W} are the two baseline systems and CNN-SVM$_{(W+X)}$ is the proposed method. CNN-SVM$_W$ represents proposed system without optimized feature set.

3.4.2 Evaluation on other benchmark datasets

In order to show the domain and language adaptability, we evaluate our proposed method i.e. *CNN-SVM*$_{(W+X)}$, on two benchmark datasets in English *viz.* $Twitter_E$ and $Review_{A_E}$. The $Twitter_E$ dataset belongs to SemEval-2015 shared task on sentiment analysis in Twitter (Rosenthal et al., 2015) and comprises of 8,210, 1,654 and 2,392 tweets for training, testing & development, respectively. Tweets in the dataset belong to different genres, i.e. **generic** as well as **sarcastic**. We evaluate the system for both genres in isolation. The second dataset belongs to SemEval-2014 shared task on ABSA (Pontiki et al., 2014). It contains approximately 3,800 user reviews from two domains, *viz.* laptop and restaurant. Table 4 depicts the results on $Twitter_E$ and $Review_{A_E}$ datasets for both the genres and domains, respectively. Results suggest that use of SVM on top of CNN, i.e. *CNN-SVM*$_W$ performs better than the typical CNN system, i.e. B_{CNN_W}. We observe the same phenomenon when optimized feature sets are concatenated with word embeddings in systems $B_{CNN_{(W+X)}}$ and *CNN-SVM*$_{(W+X)}$.

Method	Accuracy			
	$Twitter_E$		$Review_{A_E}$	
	Tweets	Sarcasm	Laptop	Restaurant
B_{SVM}	56.31	58.33	57.18	69.92
B_{CNN_W}	51.61	45.0	64.98	73.46
CNN-SVM$_W$	52.96	50.0	65.29	74.65
$B_{CNN_{(W+X)}}$	56.06	51.67	67.28	74.07
CNN-SVM$_{(W+X)}$	**58.62**	**61.67**	**68.04**	**77.16**

Table 4: Results of baseline models and proposed method for $Twitter_E$ and $Review_{A_E}$ datasets

#	Domain		Sentence	Actual	Predicted
1	$Reviews_H$	D	स्पीकर के आवाज़ की क्वालिटी अच्छी है , लेकिन हम कुछ और तेज़ आवाज़ चाहते थे ।	**Conflict**	**Positive**
		T_l	speekara ke AAvaaZ kee kvaaliTee Achchhee hai , lekin ham kuchh AOra teZ AAvaaZ chaahate the .		
		T_r	The sound quality of the speaker is good, but we expected better sound.		
	$Movie_H$	D	इन फिल्मों में कोई मजबूत कहानी नहीं थी मगर स्टंट के भरोसे फिल्म चल रही थी ।	**Conflict**	**Negative**
		T_l	In philmoN meN koEE majaboot kahaanee naheeN thee magara sTaNT ke bharose philm chal rahee thee .		
		T_r	There was no strong story in the film but has good stunts.		
2	$Twitter_H$	D	गणपति विसर्जन के मौके पर सुरक्षा कड़ी - URL	**Positive**	**Neutral**
		T_l	gaNNapati visarjan ke maoke para surakShaa kaDdee - URL		
		T_r	Tight security on the eve of Ganpati Visargan - URL.		
	$Reviews_H$	D	3जी प्लास्टिक का बना हुआ था जो हाथ से फिसलता था ।	**Negative**	**Neutral**
		T_l	3jee plaasTik kaa banaa huAA thaa jo haath se phisalataa thaa .		
		T_r	It was made up of 3G plastic which was slippery.		
3	$Reviews_H$	D	लेकिन इसमें लगे एएमडी रेडिओन एचडी7670एम ग्राफ़िक्स चिप का प्रदर्शन लाज़वाब है ।	**Positive**	**Neutral**
		T_l	lekin IsameN lage EEmaDee reDiOn EchaDee7670Em graaphiksa chip kaa pradarshan laaZvaab hai .		
		T_r	But the performance of the integrated AMD Radion HD7670M graphics chip is splendid.		

Table 5: Qualitative analysis: Examples of the error case. D, T_l and T_r represents devanagari, transliterated and translated forms

3.5 Analysis of results

Results suggest that our proposed architecture performs reasonably well for different domains and languages. The traditional CNN architectures are known to be capturing the lexical and structural features very well. We incorporate this idea into our work to learn sentiment embedded vector which helps in the attaining better results as evident from Table 3. The usage of SVM (rather than traditional softmax function) on sentiment embedded vector is able to generate the decision hyperplane more accurately by projecting the CNN features into higher dimension. Further, the (near) optimal set of features produced by MOO based feature selection technique (in addition to CNN features) assists SVM for more accurate prediction using sentiment augmented optimized vector.

We also perform Analysis of Variance (ANNOVA) (Anderson and Scolve, 1978) test which is a measure of statistical significance on the obtained results. We execute our approach 10 times with varying parameter settings. We observed that the variance in mean accuracy between proposed method and state-of-the-art methods is less than 5%. It signifies that improvements over the baselines are statistically significant.

We perform a detailed analysis (quantitative and qualitative) of the outputs to study the effectiveness as well as the shortcomings of the proposed approach.

We observe that 13% and 5.8% mis-classified test instances are correctly predicted by the *CNN-SVM*$_{(W+X)}$ model for Twitter and reviews domain, respectively. Motivations for using CNN architecture are two-fold: i) to learn hidden semantics from a large unlabeled corpus; and ii) handling limited coverage of lexical resources (e.g. Hindi SentiWordNet). In contrast to B_{SVM}, we observe that CNN architecture correctly captures instances such as "@imVkohli: धोनी के 'अतिआत्मविश्वास' के

कारण हारी टीम (@imVkohli: dhonee ke 'AtiAAtmavishvaasa' ke kaaraNN haaree Teema-@imVkohli: Team lost due to over-confidence of Dhoni.)", in which sentiment bearing words: अतिआत्मविश्वास (AtiAAtmavishvaas:Over-confidence) and हारी (haare:lost) were not found in SentiWordNet and in training set as well.

While analysing the outputs for the errors, we observe the following points:

1. Sentiment in a sentence can be either expressed by a explicit use of sentiment word e.g. अच्छा (Achchhaa:good) or by a word which carries implicit sentiment e.g. फीके (pheeke:light). For conflict sentences, explicit use of a positive or negative sentiment word drives the system to predict its output as 'positive' or 'negative'. In the first sentence of Table 5, presence of अच्छी (Achchhee:good) misguides the system to predict its sentiment as 'positive'.

2. Absence of an explicit sentiment marker in a sentence makes it harder for the system to correctly predict the sentiment. For the second sentence in Table 5, the system classifies as 'neutral' as no explicit trigger word is present.

3. The system mis-classifies some of the sentences which have explicit sentiment bearing words, but their corresponding word representations are missing due to their rare occurrences. For example, in the third sentence, word लाज़वाब (laaZvaab:splendid) has a positive sentiment. As its representation is missing from the word embedding output, system incorrectly predicts it as 'neutral'.

4 Conclusion

In this paper, we propose an efficient hybrid deep learning architecture for sentiment analysis in resource-poor languages. We learn sentiment embedded vector using CNN architecture and make a final prediction by replacing softmax function with a stronger classifier, i.e. SVM at the output layer of CNN. Training of SVM is further assisted by the optimized feature set computed by a multi-objective GA based feature selection technique to form sentiment augmented optimized vector. We build various models and evaluate our proposed method on the datasets of varying domains: Twitter (generic & sarcastic) and product/service reviews (aspect-level and sentence-level sentiment analysis). For all datasets we observed that our method consistently reports better accuracy than the various baselines and state-of-the-art systems. We observed that the usage of SVM and optimized feature set in the proposed approach helps it to achieve encouraging performance across the domains and languages compared to the state-of-the-art methods.

In this work, we include only one of the sub-problems of aspect based sentiment analysis i.e. aspect term sentiment classification. In future we would like to solve other sub-problems of ABSA such as aspect term extraction, aspect category detection and its sentiment classification. Aspect term extraction is an sequence labeling task while aspect category detection is a multi-lable classification tasks. We plan to explore recurrent neural networks (RNN) for aspect term extraction and extend our CNN based approach for multi-label classification. Also, since quality of word representation is an important factor in any neural network architecture, we plan to make use of techniques such as distance supervision for enhancing the quality of word representations.

References

Md Shad Akhtar, Asif Ekbal, and Pushpak Bhattacharyya. 2016. Aspect based sentiment analysis in hindi: Resource creation and evaluation. In *In Proceedings of the 10th edition of the Language Resources and Evaluation Conference (LREC)*.

T. W. Anderson and S.L. Scolve. 1978. *Introduction to the Statistical Analysis of Data*. Houghton Mifflin.

Akshat Bakliwal, Piyush Arora, and Vasudeva Varma. 2012. Hindi Subjective Lexicon: A Lexical Resource For Hindi Polarity Classification. *In Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC)*.

A. R. Balamurali, Aditya Joshi, and Pushpak Bhattacharyya. 2012. Cross-Lingual Sentiment Analysis for Indian Languages using Linked WordNets. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, pages 73–82.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Amitava Das and Sivaji Bandyopadhyay. 2010. SentiWordNet for Indian Languages. *In Asian Federation for Natural Language Processing, China*, pages 56–63.

Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. 2000. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II. In *Parallel problem solving from nature PPSN VI*, pages 849–858. Springer.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197.

Kalyanmoy Deb. 2001. *Multi-objective Optimization Using Evolutionary Algorithms*, volume 16. John Wiley & Sons.

Xiaowen Ding, Bing Liu, and Philip S. Yu. 2008. A Holistic Lexicon-based Approach to Opinion Mining. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08.

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In *COLING*, pages 69–78.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*.

Deepak Kumar Gupta, Kandula Srikanth Reddy, Asif Ekbal, et al. 2015. PSO-ASent: Feature Selection Using Particle Swarm Optimization for Aspect Based Sentiment Analysis. In *Natural Language Processing and Information Systems*, pages 220–233. Springer.

Vasileios Hatzivassiloglou and Kathleen R McKeown. 1997. Predicting the Semantic Orientation of Adjectives. In *Proceedings of the ACL/EACL*, pages 174–181.

Soo-Min Kim and Eduard Hovy. 2004. Determining The Sentiment of Opinions. In *In proceedings of the 20th International Conference on Computational Linguistics*, page 1367. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. *arXiv preprint arXiv:1408.5882*.

Ayush Kumar, Sarah Kohail, Asif Ekbal, and Chris Biemann. 2015. IIT-TUDA: System for Sentiment Analysis in Indian Languages Using Lexical Acquisition. In *Mining Intelligence and Knowledge Exploration*, pages 684–693. Springer.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a Word-Emotion Association Lexicon. 29(3):436–465.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013a. NRC-Canada: Building The State-of-The-Art in Sentiment Analysis of Tweets. *arXiv preprint arXiv:1308.6242*.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013b. NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. In *In Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013)*, Atlanta, Georgia, USA, June.

Bo Pang and Lillian Lee. 2008. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Braja Gopal Patra, Dipankar Das, Amitava Das, and Rajendra Prasath. 2015. Shared Task on Sentiment Analysis in Indian Languages (SAIL) Tweets-An Overview. In *Mining Intelligence and Knowledge Exploration*, pages 650–655. Springer.

NLMM Pochet and JAK Suykens. 2006. Support Vector Machines Versus Logistic Regression: Improving Prospective Performance in Clinical Decision-making. *Ultrasound in Obstetrics & Gynecology*, 27(6):607–608.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 Task 4: Aspect Based Sentiment Analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, Dublin, Ireland, August.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2015. Deep Convolutional Neural Network Textual Features and Multiple Kernel Learning for Utterance-level Multimodal Sentiment Analysis. In *In Proceedings of EMNLP*, pages 2539–2544.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. Semeval-2015 Task 10: Sentiment Analysis in Twitter. *In Proceedings of SemEval-2015*.

Shriya Se, R Vinayakumar, M Anand Kumar, and KP Soman. 2015. AMRITA-CEN@ SAIL2015: Sentiment Analysis in Indian Languages. In *Mining Intelligence and Knowledge Exploration*, pages 703–710. Springer.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Yichuan Tang. 2013. Deep Learning Using Linear Support Vector Machines. *arXiv preprint arXiv:1306.0239*.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Synthesis Lectures on Human Language Technologies. Springer.

Janyce Wiebe and Rada Mihalcea. 2006. Word Sense and Subjectivity. In *In proceedings of the COLING/ACL*, pages 1065–1072.

Word Segmentation in Sanskrit Using Path Constrained Random Walks

Amrith Krishna, Bishal Santra*, Pavankumar Satuluri,
Sasi Prasanth Bandaru[#], Bhumi Faldu, Yajuvendra Singh[##] and Pawan Goyal
*Dept. of Electronics & Electrical Communication Engineering,
[#]Dept. of Electrical Engineering, [##]Dept. of Mathematics,
Dept. of Computer Science & Engineering
Indian Institute of Technology, Kharagpur, WB, India
amrith@iitkgp.ac.in

Abstract

In Sanskrit, the phonemes at the word boundaries undergo changes to form new phonemes through a process called as *sandhi*. A fused sentence can be segmented into multiple possible segmentations. We propose a word segmentation approach that predicts the most semantically valid segmentation for a given sentence. We treat the problem as a query expansion problem and use the path-constrained random walks framework to predict the correct segments.

1 Introduction

Word segmentation is an essential step for most of the text processing tasks. In Sanskrit texts, it is very common that the phonemes at the word boundaries undergo changes to form new phonemes, through a process termed *sandhi*. It also makes the word boundary between the words that undergo *sandhi*, indistinguishable. Proximity between phonemes (*saṃhitā*) is the sole criteria for applying *sandhi*, thus phonetic transformation takes place between two consecutive words. This poses a big challenge in identifying individual words in a given sentence, as the same fused form can be segmented into many possible segmentations. Though, there has been considerable advancements in tackling word segmentation challenges faced in other languages including Chinese, Korean and Arabic, a direct application of those approaches is not possible in Sanskrit texts due to the phenomena of *sandhi*. Since the *sandhi* rules are well documented in the tradition, all the syntactically valid segmentations (splits) of a given sentence in Sanskrit can be enumerated. Sanskrit Heritage Reader provides a nice compact interface to show all the valid solutions (Huet and Goyal, 2013). The main challenge is to identify the most relevant solution from all the possible segmentations. The relevance of contextual information in identifying semantically relevant segments is a well-accepted fact (Hellwig, 2009). Mittal (2010), Natarajan and Charniak (2011) have successfully applied statistical methods for *sandhi* splitting in Sanskrit. But, both the approaches relied heavily on word co-occurrence features as their context with minimal usage of the morphological information. We, unlike the previous methods, propose a segmentation approach which effectively combines the morphological features in addition to the word co-occurrence features from a manually tagged corpus of more than 400,000 sentences (Hellwig, 2009). We treat the problem as a query expansion problem that iteratively selects a segment amongst the competing segmentations and then continues to select the next correct segment with the information from extended context. The algorithm terminates when no more candidates remain to be evaluated. We use the scalable "path-constrained random walks (PCRW)" (Lao and Cohen, 2010) framework with linguistically motivated Inductive Logic Programming (ILP) formulations for finding the correct segmentation. Using extensive experiments, we find that our approach outperforms the existing approaches by a significant margin.

2 Challenges in Sandhi Splitting

Processing of Sanskrit text poses challenges of its own due to the *sandhi* phenomena. For example, consider the phonemic change at the boundary for the case of *ramā* + *īśaḥ* → *rameśaḥ*. Here the phonemes *ā* at the end of the previous word and *ī* at the beginning of the subsequent word combine together to

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 494–504, Osaka, Japan, December 11-17 2016.

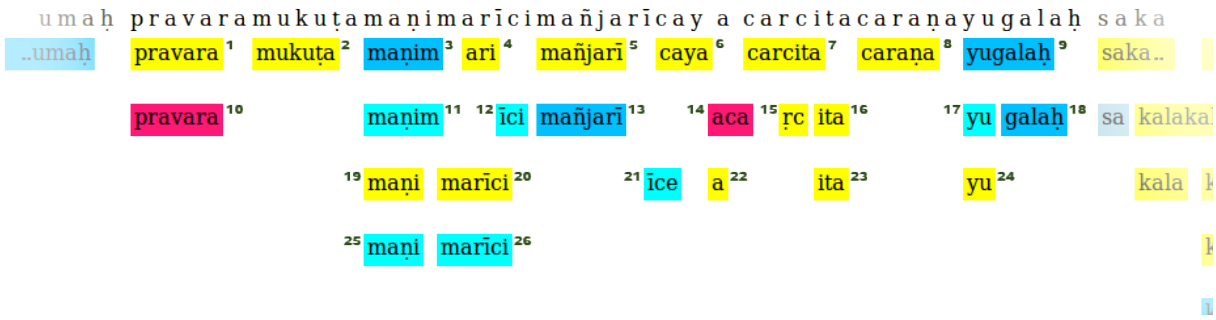


Figure 1: Possible segmentations for the compound "*pravaramukuṭamaimarīcimañjarīcayacarcitacaraṇa-yugalaḥ*" from the Sanskrit Heritage Reader. Each colour specifies a separate morphological class (Goyal and Huet, 2013; Huet and Goyal, 2013).

form the phoneme *e*. Such a transformation does not result in any change in morphological, syntactic or semantic properties of the individual words. However, there are phonetic variations resulting in the words to change their written forms.

As already mentioned, finding a semantically valid segmentation for a given sentence is a challenging task, due to the large number of possible segmentations. For example, consider the sentence, "*tatra sakalārthikalpadrumaḥ pravaramukuṭamaṇimarīcimañjarīcayacarcitacaraṇayugalaḥ sakalakalāpāraṅgato'maraśaktirnāma rājā babhūva*" - [from the text *Pañcatantram (kathāmukham)*] translates to, "There was a king namely Amaraśakthi, who was the wish-granting tree (fulfiller of all the desires) to the whole group of seekers, and the pair of whose feet was covered with a stream of rays originating from the gems in wreaths of eminent noble kings, and who was proficient in all arts". The sentence of 9 words (3 of them being compound words) can be segmented into more than half a million possible syntactically valid segmented sentences[1].

It can also be observed that multiple semantically valid splits are possible for a given string. For example, the phrase *śveto dhāvati* may be decomposed in two different ways. Both, the splits, *śvetaḥ dhāvati* (The white [one, horse] runs) and *śvā itaḥ dhāvati* (The dog runs [towards] here) are valid and give two different interpretations of the same phrase. This re-emphasizes the need for the context of the whole sentence in determining the correct set of candidates. The Sanskrit grammar poses no restriction on words in undergoing sandhi, other than the proximity of the phonemes at word boundaries. Consider the statement *dadarśāvatarantamambarāddhiraṇyagarbhāṅgabhuvaṃ muniṃ hariḥ* (Lord Viṣṇu saw Brahma's son Nārada, descending from the sky). Here, in this statement, the words *ambarāt* (from the sky) and *hiraṇyagarbhāṅgabhuvaṃ* (Nārada, son of Lord Brahma) are not related semantically, barring this they undergo euphonic transformation and fuse together to form *ambarāddhiraṇyagarbhāṅgabhuvaṃ*.

In dealing with segmentation, there are scenarios where the compound words should not be split into the constituents. Exo-centric compound refers to an external entity and the compound should be considered as a whole. Otherwise, the statement in which the compound is used need not result in a meaningful sentence. For example, the word '*daśaratha*' refers to a person and it does not refer to any of its constituents '*daśa*' (ten) or '*ratha*' (chariot). But during the analysis, it is required to use the compound component information, or else there might not be sufficient distributional information about each segment. Compound formation in Sanskrit is highly productive in nature. In the sentence from *Pañcatantram* menioned earlier, the compound,"*pravaramukuṭamaṇimarīcimañjarīcayacarcitacaraṇayugalaḥ*" (The pair of whose feet was covered with a stream of rays originating from the gems in wreaths of eminent noble kings,) is an Exo-centric (*Bahuvrīhi*) compound which consists of 9 constituents. Figure 1 shows the possible segmentation for the mentioned compound. The correct segmentation is the collection of words numbered 1-2-19-20-5-6-7-8-9. In such cases, it may be difficult to obtain distributional information about the compound if we analyse it without decomposing the compound into its components. But, after the analysis, if we do not join the compound components back to the original compound, then it alters the meaning of the sentence.

[1]Using the Sanskrit Heritage Reader available at http://sanskrit.inria.fr/

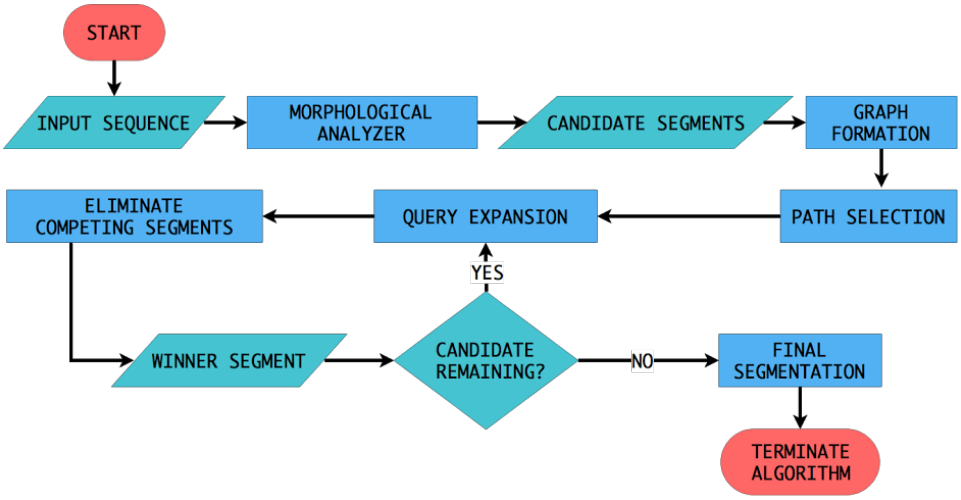


Figure 2: Flowchart for the Proposed Approach

3 Method

We treat the word segmentation problem as a query expansion problem. An input sentence is passed through the Sanskrit Heritage Reader[2]. The Sanskrit Heritage Reader provides all the possible segmentations for the input sentence along with the morphological information for each of the segments, similar to those in Figure 1. The output segments will henceforth be referred to as candidates. We now conceptualise a graph with these candidates as the nodes in the graph, in which we perform path-constrained random walks (PCRW) (Lao and Cohen, 2010) using a set of pre-defined path types. We start with the most promising candidate(s) as our initial query node and perform PCRW to obtain a winner node among the candidates. We add the winner to the extended context and repeat the process until no more candidates remain to be evaluated. We eliminate all the candidates conflicting with the winner node whenever a winner node is selected. The conflicting nodes are eliminated using positional information and *Sandhi rules*. Figure 2 provides the flowchart for the entire process.

3.1 Graph Formation

With the segments from the morphological analyser, we form a weighted multi-digraph $G(V, E, W)$, such that each node in the vertex set V represents a candidate. Every node has three attributes, the word-form, the lemma and the POS tag (morphological information) of the given segment. In this work, POS denotes the morphological information that each word-form possesses based on its usage in the sentences as directly obtained from the Sanskrit Heritage Reader. For nouns, the gender, the plurality and the grammatical case of the word-forms are obtained. For verbs the tense, number and person are the relevant morphological information that we obtain. We now construct multiple edges between all the candidate nodes except those which are conflicting with each other. We decide whether two nodes are conflicting using the position information and *sandhi* analysis. Consider two segments (k, z) and (k', z') in a given sentence. Here k and k' are the starting position (offsets) of these segments relative to the sentence, z and z' are the length of the segments. We say that (k, z) and (k', z') *conflict* if $k \leq k' < k + |z| - 1$ or $k' \leq k < k' + |z'| - 1$ (Huet and Goyal, 2013).

For every non-conflicting pair of nodes, we form edges with varying edge weights where the edge weights are decided by the possible combinations of attribute pairs at these nodes. Since each node has 3 attributes we form a total of 9 directed edges (in E) between every pair. The edge weights (in W) are defined as the co-occurrence probability of the attribute value at the target node given the attribute value at the source node. The probability values can be computed from a suitable corpus.

[2]http://sanskrit.inria.fr/DICO/reader.fr.html

3.2 Language Model

We represent our corpus, from which we obtain the distributional information for our task, as a Graph $G_2(V_2, E_2, W_2)$. We add every unique lemma, POS tag and word-form that occur in the corpus as a vertex in the graph. So, we define V_2, the vertex set as the union of the vocabulary of each of the three attributes. Now we form directed edges (in E_2) between every pair of nodes that co-occur in a sentence in the corpus. We calculate the edge weights (in W_2) using the expression 1.

$$P_{co}(j|i) = \frac{count(j,i)}{count(i)} \tag{1}$$

Here $count(i,j)$ denotes the total number of sentences in the corpus in which the nodes i and j co-occur. $count(i)$ is the count of documents in which node i occurs. For example, in order to compute the edge-weight for an edge from a POS node 'gen.m.sg.' to a lemma node '*hari*', we calculate $P(hari|gen.m.sg.)$, i.e, conditional probability of the lemma '*hari*' co-occurring with any word having the POS tag 'gen.m.sg.' in the same sentence. Similarly, we find edge weights from every possible pair of nodes that co-occur in the given corpus.

3.3 Path Selection using PCRW

With PCRW, the framework supports forming paths of any arbitrary length and also to formulate the constraints for path formations. The constraints we impose for path formation are termed as path types. We define various linguistically motivated path-types as Inductive Logic Programming (ILP) formulations. The path types are defined in Table 1. We form all possible paths in the graph G which satisfy the constraints defined in various path-types. In our formulation of the path types, a single edge path-type is defined as shown in expression 2.

$$Attribute^{constraint}_{node\ id} \xrightarrow{P_{co}} Attribute^{constraint}_{node\ id} \tag{2}$$

$$POS^{Noun}_{i} \xrightarrow{P(j|i)} Lemma^{Verb}_{j} \tag{3}$$

$$Constraint = \{\mathcal{P}(Noun), \mathcal{P}(Verb), \mathcal{P}(Compound), \mathcal{P}(Indeclinable), *\} \tag{4}$$

We define a node-type as $Attribute^{constraint}_{node\ id}$. Here $Attribute_{node\ id}$ denotes the attribute value of the node in G with the id of the node given as the subscript. The superscript is a constraint that specifies the requirements a node in G must satisfy to be in the path of the specified type. Expression 3 is a path-type of length one. Here, the source node in the expression is POS_i, i.e., it expects the POS tag value of node i. The source node in the path must be a noun and the target node must be a verb. In the sentence in Figure 1, the only eligible nodes to be the source node for this path type are nodes numbered 3,13,9 and 18. There are only two nodes which can be the target node which are 10 and 14. Hence, there are eight possible paths of the path-type given in expression 3. Now, consider one of the eight possible paths $nom.sg.f._{13}^{Noun} \xrightarrow{P_{co}(aca|nom.sg.f)} aca_{14}^{Verb}$. We obtain the edge weight as $P_{co}(aca|nom.sg.f)$ from W_2, where 'aca' and '$nom.sg.f$' are two nodes in G_2

Please note that the constraint 'Noun' is defined as the set of all the possible POS tags that a noun word-form can assume. Each POS tag in the Noun set conveys the case, gender and plurality of a word. We similarly define disjoint sets of POS tags for verbs, compounds and indeclinables. The set 'Constraint' is defined in the expression 4, where $\mathcal{P}$ denotes a power set. By defining 'Constraint' as a power set, we can group different POS tags as a single constraint to be used. For example, in $Lemma^{verb}_{i} \longrightarrow Lemma^{Noun-\{nom.sg.m,inst.sg.m.\}}_{j}$, the set $\{nom.sg.m, i.sg.m.\}$ is an element of $\mathcal{P}$, where 'nom' and 'i' signify nominative and instrumental cases respectively. The edge here starts with a verb node and looks for any noun node other than 'nom.sg.m.' and 'i.sg.m.' cases. By design, we do not allow POS tags of noun, verbs or compounds to be combined, as $\mathcal{P}$ does not contain such a combination as its element. We specify '$*$' as a wild-card constraint which allows any POS tag.

Sl. No.	Linguistic Proprety	Path types
1	General Relations	$Lemma_i^* \longrightarrow Lemma_j^*$
2		$POS_i^* \longrightarrow POS_j^*$
3	Compounds	$p(Lemma_i^{compound} \vert Lemma_j^{compound})$
4	Expectancy	$POS_i^{noun} \longrightarrow POS_j^{verb} \longrightarrow Lemma_k^{noun}$
5		$Lemma_i^{noun} \longrightarrow Lemma_j^{verb} \longrightarrow Lemma_k^{noun}$
6	Proximity Nouns	$Word_i^{noun} \longrightarrow Lemma_j^{noun}$, where,$POS_i = POS_j$
7		$Lemma_i^{noun} \longrightarrow Lemma_j^{noun}$, where,$POS_i \neq POS_j$
8	Proximity Verbs	$Lemma_i^{verb-\{absolutive\}} \longrightarrow Lemma_j^{absolutive}$
9		$Lemma_i^{verb-\{gerund\}} \longrightarrow Lemma_j^{gerund}$

Table 1: Path Types for path selection

Path-type starting from node i to node s is defined recursively as follows:

$$PathType_{POS_i}^{Lemma_s} : POS_i^{Noun} \xrightarrow{P(j_1|i)} Lemma_{j_1}^{Verb} \xrightarrow{P(j_2|j_1)} Lemma_{j_2}^{Verb} .... \xrightarrow{P(s|j_k)} Lemma_s^{Verb} \tag{5}$$

$$P(PathType_{POS_i}^{Lemma_s}) = P(j_1|i).P(s|j_k) \prod_{l=2...k} P(j_l|j_{l-1}) \tag{6}$$

In expressions 5 and 6, $P(j|i)$ denotes $P_{co}(Attribute_j|Attribute_i)$. The value of the path from node i to node s is defined as the product of all the weights (from W_2) of the edges in the path. The value so obtained can be thought as a random walk score, where a random walk traverses over graph G_2 starting at node $Attribute_i$ and terminating at $Attribute_s$ with all other nodes in the path as intermediate nodes.

Table 1 defines various path types that we use in our system. Though only a sample of path types is shown, it is implied that path types of other attributes with a similar structure are also formed. In Table 1, we also mention the linguistic properties that motivated us to formulate the path types.

General Relations: The general relations use the '$*$' constraint implying any node can form path of this type. Here we restrict ourselves to the path types which have same attribute types at both the ends of the path (edge). We also use Kneser-Ney smoothing (Kneser and Ney, 1995) for word-form to word-form co-occurrence and lemma to lemma co-occurrence values, though we do not do the same for POS to POS co-occurrence. Smoothing is not provided for POS to POS as we find that all the possible co-occurrences are already captured in the language model.

Compounds: We find bigram probabilities with various compound components since the compounds strictly follow a sequential order in their formation.

Expectancy of a verb: In Sanskrit, there always exists a modifier-modified relation between the words and the verb in the sentence, known as *kāraka* roles or semantic roles. Expectancy plays a vital role in establishing these modifier-modified relations in a sentence. Every verb expects different semantic roles in its sentence usages, which are marked with different syntactic markers in Sanskrit. The path types 4 and 5 try to capture this notion. For example, consider the sentence *rāmaḥ brāhmaṇāya gāṃ dadāti* (Ram gives a cow to the Brahmin). Here the noun words *rāmaḥ* and *brāhmaṇāya* are in different POS tags serving the roles of subject and beneficiary. Though we do not need to know the exact roles, but with the path *rāma⟶dadāti⟶brāhmaṇāya*, we attempt to capture the probability of both the words to co-occur with the verb (both the noun words need not co-occur in the same sentence in the corpus. Refer expression 6), while both being in different POS tags. The path can be an approximation for expectancy.

Proximity - Kulkarni et al. (2015) defined Proximity in Sanskrit as 'the representation of word meanings without any intervention'. But in Sanskrit poetry, this may not be strictly followed as the poet needs to maintain the meter of the verses. Kulkarni et al. (2015) conclude that there is a violation of proximity in case of *viśeṣaṇa-viśeṣya*, i.e., the modifier-modified relation between two nouns, both in case of prose as well as poetry In such situations, morphological markers are one of the ways to identify related words. Paths 6 - 9 attempt to capture this notion. In path 6, we find the co-occurrence probabilities of

two nodes when both appear in the same POS. Path 6 tries to capture the relation of *viśeṣaṇa-viśeṣya* between two nouns. Path 7 exactly finds the inverse where we find the co-occurrence probabilities of two nodes when they are not in the same POS. Path 8 is different from Expectancy path-types as there is no verb in between the nodes.

3.4 Query Expansion with PCRW

In graph G, we form paths of all possible types starting from query node and terminating at any candidate node. No two nodes in a path should be conflicting with each other. For Path types 1, 2 (General relations) and 6 in Table 1, we perform random walks with restarts over graph G to capture the structural properties of the graph. All the three paths are of length 1, i.e. they are edges. Here we relax the criteria for path formation and we form edges with all the possible node pairs in G other than the conflicting nodes. The edge weights are obtained again from W_2. With other path-types, we do not perform random walks over G, but as mentioned the path scores can be seen as random walk traversal over the graph G_2. This effectively helps us to leverage the structural properties of the content graph structure G and the graph G_2, which ideally represents the distributional properties in the language (corpus). We then do a weighted sum of scores from all the paths and the winner node is selected from the combined scores. We eliminate the conflicting nodes with winner nodes and then use the extended context to select the subsequent winner. We choose the initial query node by identifying the longest candidate from the set of candidates. The heuristic is inspired by the maximum matching approach used by Chen and Liu (1992). In case of a tie, we select the candidate with the minimum number of conflicts.

3.5 Supervised Parameter Estimation for Path-Types

Every path type can be considered as a feature and the random walk score at each instance for a pair of (query, source) node is a feature-value. We need to estimate the relative importance of each path type. Our input is a pair of words, the source word and target words. We generate all possible positive instances and down-sample the negative training instances from a training set of 5000 sentences. The label to each instance is a +1 or 0 depending on whether or not the pair of words co-occur in a sentence in the corpus. We use logistic regression with L-BFGS (Andrew and Gao, 2007) optimisation procedure to handle over-fitting. We follow the same optimisation procedure as followed in Lao and Cohen (2010). The authors argue that this approach is superior to the alternative one-weight-per-edge-label approach.

4 Experiments

4.1 Comparison with the Existing Approaches

We compare our system's performance with the existing approaches for *sandhi* splitting by presenting the results of our system on a data-set of 2148 strings, henceforth to be referred to as 'Test 2148', which was used by Natarajan and Charniak (2011) in their work. Table 2 presents the results from 4 different systems. OT2 and OT3 are the best performing variants from Mittal (2010) based on recall and precision respectively. NC4 is the best performing version of the algorithm proposed by Natarajan and Charniak (2011). It is a supervised Bayesian word segmentation approach which employs Gibbs sampling to sample from the posterior distribution of a training set of 25000 split strings. We tested our system on the test data with our pre-trained model and we do not use the training data used in Natarajan and Charniak (2011). Precision is the proportion of correct predictions amongst all the predictions made by the systems. Recall is the proportion of correct predictions which matched with the ground truth to the total number of segments in the ground truth.

Since the dataset does not provide any compound component information, we join the compound components from the prediction by applying the *sandhi* rules in order to obtain the compound before the evaluation. Our approach provides an overall improvement of 28.81% in F-score from the previously best performing system (NC4). A total of 1784 of 2148 (83.05%) were segmented with an F-score of one.

	Test 2148		
System	P	R	F
S	92.38	88.91	90.61
NC4	76.21	64.84	70.07
OT2	63.96	68.74	66.26
OT3	70.52	66.61	68.51

Table 2: Performance evaluation of our proposed system, S with the state of the art, NC4 (Natarajan and Charniak, 2011), OT2 and OT3 (Mittal, 2010).

	DCS 90K			Held-out Dataset		
System	P	R	F	P	R	F
B1	37.02	50.81	42.83	36.89	50.74	42.72
B2	42.25	58.62	49.115	42.04	58.45	48.91
B3	65.13	76.78	70.48	65.07	76.70	70.41
S	74.11	84.92	79.15	73.28	82.73	77.72

Table 3: Performance evaluation of our proposed system, S with the competing baselines B1, B2 and B3 over 100,000 sentences from DCS

4.2 Dataset

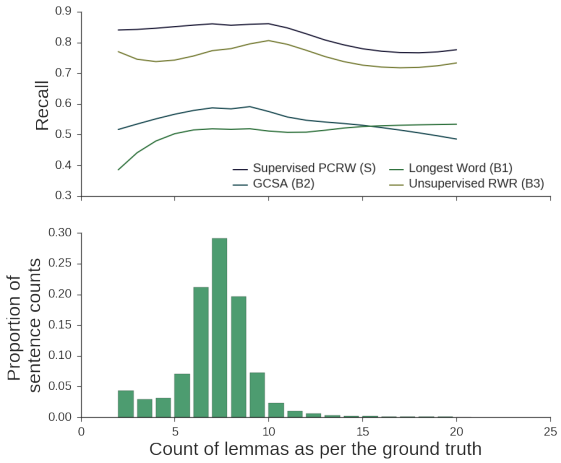
We use the Digital Corpus of Sanskrit (DCS), to build the language models and train our model. The corpus is a digitised collection of ancient works written in both prose and poetry styles with more than 430,000 segmented sentences with 66,000 unique lemmas. The corpus is a rich resource for the task as it contains over 3.2 million segmented tokens which are tagged with lemma and morphological information through expert supervision. We find only about 4067 Out Of Vocabulary (OOV) lemmas in the candidate space for which there was no co-occurrence evidence.

4.3 Baselines

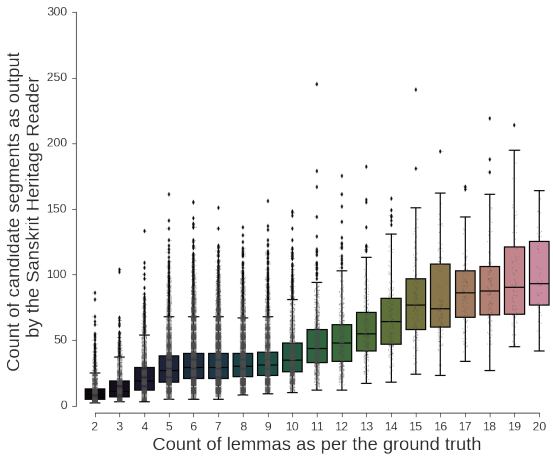
In Section 4.1 we showed the effectiveness of our model over the existing approaches. The systems described in Section 4.1, however, do not handle the context of an entire sentence. Additionally, the systems do not consider the morphological information in their training phases, and hence it will be unfair to use those systems for further analysis. We propose the following methods as our baselines. In all the systems we use the morphological analyser output as the input to predict the correct segmentation.
Longest Word selection (B1) - Inspired by the maximum matching approach from Chen and Liu (1992), we iteratively select the longest word available from the given set of possible segmentations. At each step of the iteration, we eliminate the candidates conflicting with the current winner. This method does not use any morphological information as such.
Greedy candidate selection approach (B2) - Here we consider the morphological analyser output to be a tree, and we perform a greedy selection that maximises the overall likelihood of the selection. In this approach, we combine the co-occurrence probabilities mentioned in paths 1,2 and 6 in Table 1.
Unsupervised RWR - General Relations (B3) - In this approach, we form the graph G for the sentence, similar to as mentioned in Section 3. However, we only use path-types 1,2 and 6 of Table 1. We perform Random Walks with Restarts (RWR) over the graph G and then average the scores from different random walk runs to obtain a winner node.
Supervised PCRW (S) - This is our proposed system, as defined in Section 3.

4.4 Results

We consider 100,000 sentences of varying length (not used for training in Section 3.5) from the DCS corpus. From the 100,000 sentences, 90,000 of the sentences, henceforth to be referred to as "DCS 90K", were used for calculating the co-occurrence values and we keep the remaining 10,000 sentences as held-out data which was not used at any point in our framework. Figure 3b shows the box plot for the number of sentences based on the frequency distribution of lemmas in ground truth against the count of possible candidates as given by the Sanskrit Heritage Reader for all the 100,000 files. From the plot, it is evident that the count of candidates increases manifold with the increase in lemmas. We eliminate all the sentences with exactly one lemma from the dataset which amounts to less than 1% of the total dataset. Table 3 compares the performance of each of the system in terms of precision, recall and F-Score. Our final system S performs the best with an increase of 13.79% and 10.60% in precision and recall respectively from our strong baseline B3 for the DCS 90K dataset. For the held-out dataset, the results remain more or less consistent with those obtained over DCS 90K for all the systems. Our



(a) Recall Vs. GT Frequency distribution



(b) Candidate Vs. GT Frequency distribution

Figure 3: Analysis of systems based on frequency distribution of ground truth (GT) lemmas

system S has an F-Score of 76.58% which is slightly less than the F-score attained on the original dataset, which was 79.15%. Our system could predict all the correct segmentations for as many as 38.64% of the sentences against the 19.57% fully correct predictions of B3.

Variation in Recall based on length of lemmas - The line graphs in the Figure 3a illustrate the recall of each system over the frequency distribution on the 'DCS 90K' based on the number of lemmas per sentence in the ground truth. The x-axis represents the count of lemmas in a sentence, and the y-axis represents average recall. The bar chart represents the proportion of sentences with respect to the count of lemmas in ground truth. The average length of sentences in our dataset is 6.87. It can be observed that our system works better for the sentences of all the lengths.

5 Discussion

A close inspection of the Tables 2 and 3, reveals that for all the systems, precision is higher than recall for the first dataset, and vice-versa in the DCS. On our manual inspection of the dataset, it is found that the DCS being manually tagged, relies on the context to decide whether to decompose a compound into its components. For example the word '*daśaratha*' is a compound word, so is *kṛṣṇārjunanakulasahadevāḥ*. In DCS, '*daśaratha*' being an exo-centric compound, it will not be decomposed into its components. But '*kṛṣṇārjunanakulasahadevāḥ*' is a copulative compound containing names of four different people. DCS decomposes the compound into its components. But, our morphological analyzer decomposes all the compounds into its components as it only considers the syntactic conditions. Unlike the case with first dataset, DCS does not contain segmented word-forms but only segmented lemmas. Hence joining of components is sometimes not possible as grammar cases for the nouns is not available in the system to perform *sandhi*. Due to this issue we find that multiple components which are predicted are treated as separate lemmas. This results in lower precision for the system, especially when 75% of the vocabulary in DCS consists of compounds.

Another intriguing challenge is with the case of compounds where it can be split with different interpretations such that, two possible segmentations result into being the exact negation of each other. Even from the semantic context, it will be difficult to consider whether the negation is required or not. There are a considerable number of sentences in which such issues occur. For example, the string *kurvannāpnoti kilbiṣam* may be decomposed in two different ways '*kurvan āpnoti kilbiṣam*' (While doing, you will accumulate sin) and '*kurvan na āpnoti kilbiṣam*' (While doing, you will not accumulate any sin) Both of them have completely opposite meanings.

6 Related Work

During the last couple of decades, computational analysis of Sanskrit and its grammar has gained considerable attention from the computational linguistics community. Hyman (2008) observes that the Pāṇinian *sūtras* for external *sandhi* can be modeled using a finite state grammar. Huet (2009) developed a tool for segmentation in Sanskrit by building an efficient Finite State Automata (FSA). With efforts from Huet, an automated analyser for exhaustive syntactically valid analysis of a Sanskrit sentence, called as the 'Sanskrit Heritage Reader' is available (Goyal et al., 2012; Goyal and Huet, 2013). However, the system provides all the possible syntactically valid segmentation and it requires human assistance to choose the relevant segmentations so as to form the semantically correct sentence. We use the 'Sanskrit Heritage Reader' in our pipeline to obtain the set of all possible segmentations.

Mittal (2010) has proposed an approach for automated Sanskrit segmentations by relying on the maximum a posteriori estimate from all possible *sandhi* splits for a given string. Natarajan and Charniak (2011) has proposed 'S3 - Statistical Sandhi Splitter', a Bayesian word segmentation approach which can handle *sandhi* formations. Hellwig (2015) proposed a neural network based approach that jointly solves the problem of compound splitting and sandhi resolution. Mochihashi et al. (2009), Johnson et al. (2006) developed non-parametric Bayesian approaches which have been applied on word segmentation tasks. The work uses a Dirichlet process model inspired from the model of Goldwater et al. (2006). Kuncham et al. (2015) have developed a language independent sandhi splitter for agglutinative languages using a strucutred prediction approach.

Word Segmentation challenges for in languages like Arabic, Japanese, Korean and Chinese are extensively studied. Though, in theses languages, identifying the proper word boundary is a challenge, none of these have to deal with the 'Sandhi' as such. In Chinese word segmentation, Xue (2003) presented a character tagging approach. Wang et al. (2014) proposed a method based on dual decomposition (Rush et al., 2010) to combine word-based and character based approaches in an efficient framework. Yao and Huang (2016) proposed a bi-directional RNN with long short-term memory (LSTM) units which makes use of character embeddings. In our approach, we use the PCRW framework proposed by Lao and Cohen (2010). Gao et al. (2013) tackled the problem of query expansion using PCRW for web-log queries. PCRW has been widely used in heterogeneous information networks. The framework has been effectively put to use in Never Ending Language Learning (NELL) (Mitchell et al., 2015; Lao et al., 2011).

7 Conclusion

In this paper we presented an approach to Sanskrit word segmentation using supervised PCRW. We treated the problem as a query expansion problem. In our approach, the input sentence is treated as a graph. The candidate segments form the nodes and there exists an edge between every pair of nodes except the conflicting ones. In Sanskrit, since there is no guarantee of proximity to be maintained between words, we presume that treating the input as a sequence as followed in models like linear-chain Conditional Random Fields (CRF) and Hidden Markov Models might be a serious limitation for the task. Also, with the rich feature space that we employ, inference in CRF Models might be a bottleneck for the model's performance (Doppa et al., 2014). We find that the inclusion of morphological information by means of linguistically motivated ILP path-types greatly increases the performance of the system as much as by 12.30% in F-Score. Our system also outperforms the existing best approach by 28.82% in F-Score. We also showed the effectiveness of our system in a dataset with a collection of both prose and poetry. Our approach is scalable and can be applied to larger datasets as well.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. The authors are grateful to Dr. Oliver Hellwig for providing the DCS Corpus and Dr. Gérard Huet for providing the Sanskrit Heritage Engine, to be installed at local systems, and helping with other queries related to Sanskrit Heritage Reader. They are also grateful to Mr. Unnikrishnan T A for his suggestions on the implementation of the framework.

References

Galen Andrew and Jianfeng Gao. 2007. Scalable training of l 1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning*, pages 33–40. ACM.

Keh-Jiann Chen and Shing-Huan Liu. 1992. Word identification for mandarin chinese sentences. In *Proceedings of the 14th conference on Computational linguistics-Volume 1*, pages 101–107. Association for Computational Linguistics.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. 2014. Structured prediction via output space search. *Journal of Machine Learning Research*, 15:1317–1350.

Jianfeng Gao, Gu Xu, and Jinxi Xu. 2013. Query expansion using path-constrained random walks. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 563–572. ACM.

Sharon Goldwater, Thomas L Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 673–680. Association for Computational Linguistics.

Pawan Goyal and Gérard Huet. 2013. Completeness analysis of a sanskrit reader. In *Proceedings, 5th International Symposium on Sanskrit Computational Linguistics. DK Printworld (P) Ltd*, pages 130–171.

Pawan Goyal, Gérard P Huet, Amba P Kulkarni, Peter M Scharf, and Ralph Bunker. 2012. A distributed platform for sanskrit processing. In *COLING*, pages 1011–1028.

Oliver Hellwig. 2009. Sanskrittagger: A stochastic lexical and POS tagger for sanskrit. In *Sanskrit Computational Linguistics*, pages 266–277. Springer.

Oliver Hellwig. 2015. Using recurrent neural networks for joint compound splitting and sandhi resolution in sanskrit. In *4th Biennial Workshop on Less-Resourced Languages*.

Gérard Huet and Pawan Goyal. 2013. Design of a lean interface for Sanskrit corpus annotation. In *Proceedings of ICON 2013, the 10th International Conference on NLP*, pages 177–186.

Gérard Huet. 2009. Sanskrit Segmentation, South Asian Languages Analysis Roundtable xxviii, Denton, Texas. South Asian Languages Analysis Roundtable XXVIII.

Malcolm D. Hyman. 2008. From Paninian Sandhi to Finite State Calculus. In *Sanskrit Computational Linguistics*, pages 253–265.

Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In *Advances in neural information processing systems*, pages 641–648.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Amba Kulkarni, Preethi shukla, Pavankumar Satuluri, and Devanand Shukla. 2015. *How Free is free Word Order in Sanskrit*. The Sanskrit Library, USA.

Prathyusha Kuncham, Kovida Nelakuditi, Sneha Nallani, and Radhika Mamidi. 2015. Statistical sandhi splitter for agglutinative languages. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 164–172. Springer.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 529–539, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*.

Vipul Mittal. 2010. Automatic sanskrit segmentizer using finite state transducers. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 85–90. Association for Computational Linguistics.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested pitman-yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 100–108. Association for Computational Linguistics.

Abhiram Natarajan and Eugene Charniak. 2011. S3-statistical saṃdhi splitting. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 301–308. Association for Computational Linguistics.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11. Association for Computational Linguistics.

Mengqiu Wang, Rob Voigt, and Christopher D. Manning. 2014. Two knives cut better than one: Chinese word segmentation with dual decomposition. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), Baltimore, MD*.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Yushi Yao and Zheng Huang. 2016. Bi-directional lstm recurrent neural network for chinese word segmentation. *arXiv preprint arXiv:1602.04874*.

Mongolian Named Entity Recognition System with Rich Features

Weihua Wang, Feilong Bao*, Guanglai Gao
College of Computer Science, Inner Mongolia University
Huhhot, China, 010021
Email: *wangweihuacs@163.com*;{csfeilong,csggl}@imu.edu.cn

Abstract

In this paper, we first build a manually annotated named entity corpus of Mongolian. Then, we propose three morphological processing methods and study comprehensive features, including syllable features, lexical features, context features, morphological features and semantic features in Mongolian named entity recognition. Moreover, we also evaluate the influence of word cluster features on the system and combine all features together eventually. The experimental result shows that segmenting each suffix into an individual token achieves better results than deleting suffixes or using the suffixes as feature. The system based on segmenting suffixes with all proposed features yields benchmark result of F-measure=84.65 on this corpus.

1 Introduction

Named Entity Recognition (NER) is a natural language processing (NLP) task that consists of finding names in an open domain text and classifying them among several predefined categories such as person, organization and location. It is an important tool in almost all NLP application areas, such as Question Answering, Machine Translation (Chen et al., 2013), Social Media Analysis, Semantic Search or Automatic Summarization.

Since the MUC (Sundheim, 1995) and CoNLL (Sang, 2002) conferences, NER has drawn more and more attention in NLP community. Many NER systems have been developed for English and other language (Ratinov and Roth, 2009; Benajiba et al., 2010; Kravalová and Žabokrtskỳ, 2009). Machine learning based approach have been the predominant in these systems to achieve state-of-the-art results (Radford et al., 2015). As one of them, Conditional Random Fields (CRF) (Lafferty et al., 2001) was proved to an efficient classifier for NER.

Recently, there are more and more concern about how to incorporate more latent sematic features into the NER system (Konkol et al., 2015). Therefore, word cluster IDs function as a non local feature to improve the performance of NER system (Turian et al., 2010; Zirikly and Diab, 2015).

Mongolian is a widely spread language in the world. It is called classical Mongolian in China and called Cyrillic Mongolian in Mongolia and Russia. The classical Mongolian uses Uighur-script, while Cyrillic Mongolian uses Cyrillic-script. In this paper, we address the problem of NER for classical Mongolian.

Compared with other languages, the research on Mongolian NER is still at its initial stage and many issues in Mongolian NER remain unsolved. As far as we know, there has been very little work in the area of NER in Mongolian. Tong (2013) only investigated Mongolian person name recognition. There is still no work publicly reported on recognition of Mongolian location and organization name. Moreover, there are no public available resources and tools for Mongolian NER.

However, proper identification and classification of named entities are very crucial in Mongolian information processing. Therefore, we propose a framework to develop resources and several methods for

*Corresponding author

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 505–512, Osaka, Japan, December 11-17 2016.

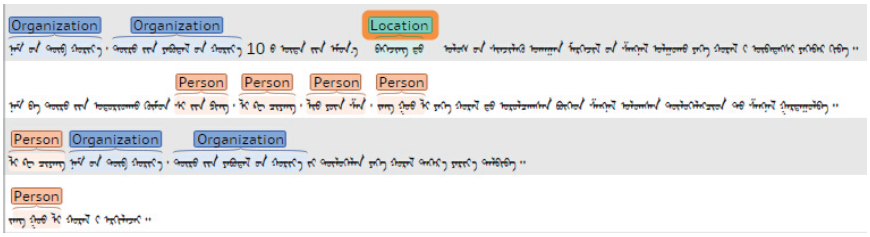


Figure 1: The annotation platform for Mongolian NER.

Mongolian named entity recognition. This paper introduces the work on Mongolian NER that is still in progress.

As one of agglutinative languages, Mongolian has complex morphological structures. We explore different morphological processing methods to alleviate data spareness. Different from the work (Şeker and şen Eryiğit, 2012), we separates Mongolian suffixes from the words and even delete the suffixes. We also propose rich features by exploiting Mongolian orthographic feature, morphological feature, syntactic feature and semantic feature.

The remainder of this paper is organized as follows: Section 2 introduces the construction of Mongolian NER resources; Section 3 presents three morphological processing methods; Section 4 introduces the language independent and language specific features we used; Section 5 describes the results of experiments; Section 6 concludes the paper and summaries some future work.

2 Construction of Mongolian NER resources

2.1 Characteristics of Mongolian

Mongolian writes from top to bottom, and the same letter has different presentation forms decided by position in the word. The NER task of Mongolian was difficult due to the following reasons:

-*Mongolian has large scale vocabulary*: Mongolian has complex morphological structures that each root can be followed by several suffixes to formulate new words. So the larger vocabulary decreased the performance of Mongolian NER system.

-*Absence of capital letters in the orthography*: In English and other Latin language, the proper names always appear with capitalized letter, but there is no concept of capitalization in Mongolian.

-*Multi-category word is very common to named entities*: many common nouns, adjectives and verbs can act as person names or location names, such as an adjective word "" (means "clever") is a very common person name in Mongolian.

-*Subject-Object-Verb word order*: boundaries between named entities are easy to confuse when the subject and object are both proper names.

2.2 Corpus

Nowadays, there is no public annotated corpus about Mongolian named entities. In this paper, we firstly created a new corpus gathered from several Mongolian news web site. We extract mainly content for every web page by analysing the character of each web page html tags. The content of this corpus includes political news, economic news, cultural news and daily news.

This corpus contains 33209 sentences, 59562 named entities and 119M tokens. It annotated manually with person, location and organization by a Mongolian native speaker under the open source platform "Brat" (Stenetorp et al., 2012). The interface of this platform shows in Figure 1. The annotation task cost about three months. At beginning, we discuss almost every sentence to guarantee the quality of annotation. It is time consuming but worth to be done. After twenty days then the annotation become more quick and more unambiguous.

This corpus was converted into BIO2 (Kudo and Matsumoto, 2001) label format. A token is labeled as "B-label" if the token is the beginning of a named entity, and labeled as "I-label" if it is inside a named entity but not the first token within the named entity, others will "O". So there are seven types, that is

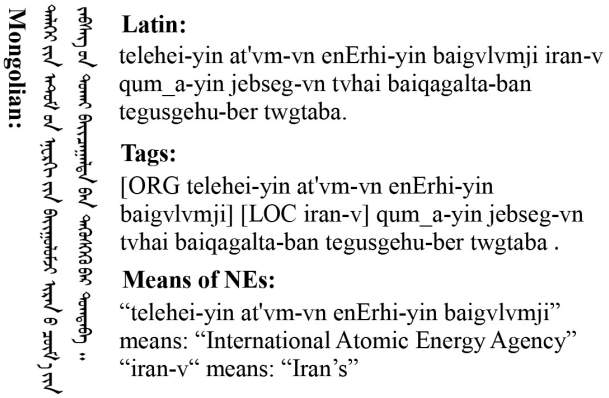


Figure 2: Example of Mongolian NNBS suffixes and named entity tags

"B-PER", "I-PER", "B-LOC", "I-LOC", "B-ORG", "I-ORG" and "O", will be classified by learning algorithm.

The average length of named entities is 2.87 words in our corpus. The person, location and organization entities account for 20.74%, 47.62% and 31.64%, respectively. There are 55% organization entities length are above three words. Mongolian person name always express in one word. However, when transliterating Chinese person into Mongolian, the person name length unchanged. So about 39% person names are three words and 33% person names are only one word.

3 Approach

3.1 Model

CRF is a probabilistic framework that suitable for labeling input sequence data (Lafferty et al., 2001). For an input sequence $X = x_1, x_2...x_n$, CRF model aims to find the best named entity label sequence $Y = y_1, y_2...y_n$ that maximizes the conditional probability $p(y|x)$ among all possible tag sequences. The $p(y|x)$ can be expressed as:

$$p(y|x) = Z(x)^{-1} exp(\sum_t \sum_k \lambda_k f_k(y_{t-1}, y_t, x) \quad (1)$$

where λ_i represents the weight assigned to different features and $Z(x)$ is the normalizing function, it can be defined as:

$$Z(x) = \sum_{y \epsilon Y} exp(\sum_t \sum_k \lambda_k f_k(y_{t-1}, y_t, x) \quad (2)$$

$f_k(y_{t-1}, y_t, x)$ is the binary feature function, such as

$$f_k(y_{t-1}, y_t, x) = 1(y_{t-1} = y', y_t = y, x_t = x) \quad (3)$$

3.2 Morphological processing

For the morphological structure of Mongolian, a Mongolian words can be decomposed into roots, derivational suffixes and inflectional suffixes. As for nouns, the inflectional suffixes contain case suffixes, reflexive suffixes and plural suffixes. All the case suffixes, reflexive suffixes and partly plural suffixes connected to stem through a Narrow Non-Break Space (NNBS) (U+202F, Latin:"-"), so we called them NNBS suffixes. For example, in Figure 2, there are 8 NNBS suffixes in one sentence, and the suffixes appeared inside or beside the named entities.

The NNBS suffixes are used very flexible that each stem can add several NNBS suffixes to change the word form. What's more, the NNBS suffixes in Mongolian can be located unambiguously, while other suffixes segmented may lead to some letters insertion, lost and substitution. Therefore, we proposed three methods to process NNBS suffixes.

RE:Remove all NNBS suffixes in text. After this processing, the sentence in Figure 2 will be "*telehei at'vm enErhi baigvlvmji iran qum_a jebseg tvhai baiqagalta tegusgehu twgtab*."

FE:Take NNBS suffixes as a new feature and replace word with stem. After this processing, sentence length remain unchanged but the feature dimension will add one.

SE:Segment NNBS suffixes as a new token. After then, the sentence will longer, for example, the sentence in Figure 2 will be turned into "*telehei -yin at'vm -vn enErhi -yin baigvlvmji iran -v qum_a -yin jebseg -vn tvhai baiqagalta -ban tegusgehu -ber twgtaba*".

If the suffixes are the last tokens in the entity, we will remove the suffixes from the entity. Because this kind of suffixes only add the syntax function for previous stem. For example, the tag of "iran-v" will change to "*[LOC iran] -v*".

4 Features

Supervised NER is sensitive to the selection of features, we consider the following feature sets for Mongolian. In the following experiment, we fixed all features window at [-1,1], that means take the previous feature, current feature and next feature into consideration, except the contextual feature.

Contextual Feature (CXT): this feature was automatically generated, and mean to combine the current and previous output tokens.

Orthographic Feature (ORT): this feature defined the lexical orthographic nature of the tokens in the text, which means the n-gram of tokens. If the suffixes were split, the n-gram tokens will include suffixes directly.

Syllable Feature (SYN): this feature contained syllable count, first and end syllable of the current token.

Syllable count: we concluded 28 rules about counting Mongolian syllables for the first time, according to Mongolian grammar. In general, too many syllables might not be names.

First and end syllable: some first syllables or end syllables occur frequently in many Mongolian person names.

Look up feature: defined as binary features and matched exactly with the lookup table.

Gazetteers (GAZ): this collected gazetteer consist of 8735 location names and 2731 person names. We extracted location names from Mongolian Chinese dictionary mainly contained Inner Mongolian location names manually. The person names list found in few Mongolian blogging web sites, and mainly contained Mongolian names.

Transliteration table (TRS): this table contained 564 Mongolian borrowed words from Chinese. For example, a very common surname in Chinese "王" ("wang").

Person title and job title list(TIT): this list contained 373 person title entries and 582 job title entries.

Morphological Feature: this feature explored rich morphological structure of Mongolian.

Part-of-speech (POS): we employed a rule and dictionary based POS tagger to produce this features. This top level POS marking set include 15 classes which according with (China Standard, 2011). When SE method applied, the POS feature of NNBS suffixes will be denoted by "F".

NNBS suffixes: used as feature only when the FE method applied. If a word contains NNBS suffixes, the suffixes themselves will be referred as features.

Word Clusters IDs: this feature gained from massive unlabeled corpus after the same SE method preprocessed. The corpus used also crawled from web sites in a more wider range. It contained 337M sentences and its token size and vocabulary showed in Table 1. From Table 1, we found the vocabulary decrease 27% while the token number growth 22%.

Word2vec clusters IDs (W2V): this feature achieved by performing K-means clustering on word2vec vectors in (Mikolov et al., 2013) and directly used the cluster IDs as features. The vectors' dimension in our experiments are 200, the minimum occurrences number of token is 3 and the context window fixed at 8. We retrained word vectors with negative sampling used skip-gram model. A new cluster number assigned to the test token without trained cluster ID.

LDA word classes (LDA): we followed the work in (Chrupala, 2011) to induce LDA to produce different word clusters with the minimum occurrences number of token is 3. If the test tokens are out of

Table 1: Vocabulary decrease after processing by SE method in cluster training corpus (mincount=3)

	Vocabulary	**Tokens number**
Word model	395511	72051575
SE model	285063	88021157

Table 2: Results of different morphological processing methods

Feats.	**Baseline**			**RE**			**FE**			**SE**		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
+ORT	84.50	77.05	80.65	84.57	79.50	81.96	84.57	79.50	81.96	84.61	80.07	82.28
+POS	84.70	78.71	81.59	85.09	81.10	83.05	85.11	81.25	83.13	85.11	81.67	**83.35**
+GAZ	84.88	79.49	82.10	85.20	81.95	83.55	85.20	82.28	83.62	85.21	82.35	83.75
+TRS	84.97	79.60	82.20	85.56	82.25	83.87	85.57	82.15	83.83	85.30	82.34	83.79
+SYN	85.02	81.04	82.98	85.11	82.63	83.81	85.18	82.63	83.88	85.07	83.16	84.10
+TIT	85.01	81.14	83.03	85.01	82.42	83.69	85.30	82.71	83.99	85.28	83.32	**84.29**

vocabulary of trained LDA word, we also assigned a new LDA classes number for them.

5 Experiment

In our experiments, we analyzed the impact of various morphological processing and various categories features under an CRF framework with the same parameters. All the experiments carried on 5-fold cross-validation, the proportion of train and test set is 80% , 20%. We evaluated the results by the CoNLL metrics of precision, recall and F-measure.

Precision, means the percentage of corrected named entities (NEs) found by the classifier. It can be expressed as:

$$precsion = \frac{Num(correct\ NEs\ predicted)}{Num(NEs\ predicted)} \tag{4}$$

Recall is the percentage of NEs existing in the corpus and which were found by the system. It can be expressed as:

$$recall = \frac{Num(correct\ NEs\ predicted)}{Num(all\ NEs)} \tag{5}$$

F_1 is the harmonic mean of precision and recall. It can be expressed as:

$$F_1 = \frac{2 * precision * recall}{precision + recall} \tag{6}$$

5.1 Impact of morphological processing

Firstly, we incrementally added features to the three methods mentioned above and each feature window fixed at [-1,1]. The results show in Table 2, the "Baseline" means without any morphological processing. From Table 2 we found that all the three methods can improve the overall performance. When incorporated all features, the SE method achieved the best and improved F-measure by 1.26. Because segmenting by NNBS can decrease the percentage of unknown word in sentences and do help to detect named entities. The features played more effect roles when the suffixes took apart.

Each feature improves the performance based on the former one. The F-measure improvement caused by POS feature is obvious. The contribution of GAZ feature lies in improving both the precision and recall of person names and location names. The TRS feature improves the F-measure because the corpus contains amount of Chinese person and location names. The SYN feature slightly indicates some cues for named entities, the first and end syllable act on prefixes or suffixes. The TIT feature benefits the person names recognition to improve overall F-measure.

Table 3: Results (in F-measure) of different semantic space of LDA word classes and word2vec clusters

Clusters numbers	LDA	Word2vec
50	83.16	83.27
100	83.20	**83.49**
200	**83.26**	83.26
500	83.24	–

Table 4: Results of different LDA word classes and word2vec clusters combination

Clusters IDs combination	F1
LDA200+W2V100	**83.53**
LDA500+W2V200	83.29
LDA100+LDA200	83.13
W2V100+W2V200	83.29
All Cluster IDs	83.24
SE+ORT	82.28

5.2 Impact of word cluster features

Secondly, we evaluate the impact of semantic features, that is, only adding LDA word cluster or word2vec cluster features onto the SE method, Table 3 shows the results. The F-measure varies with cluster number and cluster type, the more word classes does not always mean better performance. The best cluster number is 200 for LDA word cluster and 100 for word2vec cluster. Word2vec clusters outperform the LDA word cluster because that it can induce more context to cluster.

We then combined the best and second performance in Table 3 without other features to produce the best word cluster combination. Table 4 shows the results. In Table 4, SE+ORT means only using context feature with SE method, as baseline system. The F-measure reaches 83.53 when coupled with LDA200 and W2V100. This best F-measure even surpassed the performance of POS and ORT feature combination about F_1 is 83.35 under SE method in the same condition. However, the overall performance reduced when added all type clusters features to the feature set.

5.3 Final system

Finally, we integrate all features including traditional features and word cluster features in SE method, Table 5 shows the final system performance.

In Table 5, AFH represents all the handcraft features, including ORT, POS, GAZ, TRS, SYN and TIT. From Table 5, we find that the same word cluster feature works different when combine with traditional features. With only single word cluster, the effect is weak, but when we use the combination of AFH, LDA100 and LDA200, result reach the best. It outperforms the handcraft features 0.36 in F_1. As the results shown, combining more features does not mean higher performance.

6 Conclusion

In this paper, we built a Mongolian named entity recognition corpus and explored three morphological processing methods with different features combination under the CRF framework. This is the first corpus for Mongolian and we carry on experiment on this corpus. The experimental results show that the proposed methods can alleviate the sparseness of data and improve the performance of Mongolian NER system. In addition, the word cluster features represent the latent semantic of word can also benefit the system. Among the above three methods, treating NNBS suffixes as individual token perform best. It can reach F-measure at 84.65 when combined all features including handcraft features and word cluster features. This work can also provide benchmark system to promote the future Mongolian NER research community.

In the future, we will try our method to other agglutinative languages and expand the work on using word cluster feature. We will also try to use deep neural network to perform to Mongolian NER.

Table 5: Final performance combined all features

Features	F1
AHF	84.29
AHF+LDA50	84.21
AHF+LDA100	84.33
AHF+LDA200	84.36
AHF+W2V50	84.34
AHF+W2V100	84.48
AHF+W2V200	84.54
AHF+LDA100+LDA200	**84.65**
AHF+LDA200+W2V100	84.57
AHF+LDA200+W2V200	84.54
AHF+LDA100+LDA200+W2V100	84.57
AHF+LDA100+LDA200+W2V100+W2V100	84.36

Acknowledgements

This research is partially supported by the China National Nature Science Foundation (No.61263037, No.61303165 and No.61563040), Inner Mongolia Nature Science Foundation (No.2014BS0604 and No.2016ZD06) and the program of high-level talents of Inner Mongolia University. Finally, we thank the anonymous reviews for their many helpful comments.

References

Yassine Benajiba, Imed Zitouni, Mona Diab, and Paolo Rosso. 2010. Arabic named entity recognition: using features extracted from noisy data. In *Proceedings of the ACL 2010 conference short papers*, pages 281–285. Association for Computational Linguistics.

Yufeng Chen, Chengqing Zong, and Keh-Yih Su. 2013. A joint model to identify and align bilingual named entities. *Computational Linguistics*, 39(2):229–266.

GB26235-2010 China Standard. 2011. *GB26235-2010 Information technology-Mongolian word and expression marks for information processing*. China National Standardization Technical Committee.

Grzegorz Chrupala. 2011. Efficient induction of probabilistic word classes with LDA. In *Fifth International Joint Conference on Natural Language Processing, IJCNLP 2011, Chiang Mai, Thailand, November 8-13, 2011*, pages 363–372.

Gökhan Akın Şeker and Gül şen Eryiğit. 2012. Initial explorations on using crfs for turkish named entity recognition. In *Proceedings of the 24th International Conference on Computational Linguistics, COLING 2012.*, Mumbai, India, 8-15 December.

Michal Konkol, Tomas Brychcin, and Miloslav Konopík. 2015. Latent semantics in named entity recognition. *Expert Syst. Appl.*, 42(7):3470–3479.

Jana Kravalová and Zdeněk Žabokrtskỳ. 2009. Czech named entity corpus and svm-based recognizer. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration*, pages 194–201. Association for Computational Linguistics.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of the 2001 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 746–751.

Will Radford, Xavier Carreras, and James Henderson. 2015. Named entity recognition with document-specific KB tag gazetteers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 512–517.

Lev-Arie Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning, CoNLL 2009, Boulder, Colorado, USA, June 4-5, 2009*, pages 147–155.

Erik F. Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning, CoNLL 2002, Held in cooperation with COLING 2002, Taipei, Taiwan, 2002*.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics.

Beth Sundheim. 1995. Overview of results of the MUC-6 evaluation. In *Proceedings of the 6th Conference on Message Understanding, MUC 1995, Columbia, Maryland, USA, November 6-8, 1995*, pages 13–31.

Gala Tong. 2013. *Automatic Recognition of Mongolian Names Based on Corpus*. Ph.D. thesis, Ming Zu University of China.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 384–394.

Ayah Zirikly and Mona Diab. 2015. Named entity recognition for arabic social media. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 176–185, Denver, Colorado, June. Association for Computational Linguistics.

Appraising UMLS Coverage for Summarizing Medical Evidence

Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong, Fang Chen
University of New South Wales, Sydney, Australia
Data61 CSIRO, Australia
{elahehs,mohammade,wong,fang}@cse.unsw.edu.au

Abstract

When making clinical decisions, practitioners need to rely on the most relevant evidence available. However, accessing a vast body of medical evidence and confronting the issue of information overload, can be challenging and time consuming. This paper proposes an effective summarizer for medical evidence by utilizing both UMLS and WordNet. Given a clinical query and a set of relevant abstracts, we aim to generate a fluent, well-organized, and compact summary that answers the query. Analysis via ROUGE metrics shows that using WordNet as a general-purpose lexicon helps to capture the concepts not covered by the UMLS Metathesaurus, and hence significantly increases the summarization performance. The effectiveness of our proposed approach is demonstrated by conducting a set of experiments over a specialized evidence-based medicine (EBM) corpus - which has been gathered and annotated for the purpose of biomedical text summarization.

1 Introduction

Over the past two decades, clinical guidelines urged practitioners to move towards evidence-based medicine, which is formally defined as *conscientious and judicious use of current best evidence in making decisions about the care of individual patients* (Sackett et al., 1996). Evidence-based medical practice heavily relies on research evidence, rather than intuition, unsystematic clinical experience, or pathologic rationale (Group and Others, 1992). However, searching through and evaluating primary medical literature is extremely time consuming (Sarker et al., 2015). Even targeted searches tend to return a large set of relevant documents, and not summaries or answers to the queries. Hence, the explosive growth of content of medical evidence requires development of techniques to present information to physicians and researchers in an effective way. Automatic text summarization has been introduced as a natural language processing technique to address this problem (Reeve et al., 2007).

Even though the problem of information overload and the advantages of summarization are critical in the biomedical domain, the majority of summarizers are designed to be general-purpose. They usually work with a simple representation of the summary comprising of information that can be directly extracted from the document, such as terms, phrases, or sentences (Mihalcea and Tarau, 2004). However, recent studies (e.g. (Fiszman et al., 2004)) have demonstrated the benefits of summarization based on richer representations that make use of domain-specific knowledge sources. These approaches represent the documents using concepts instead of words, and may also be enriched by using semantic associations among concepts (e.g. synonymy, hypernymy, etc.) (Plaza et al., 2011).

While a query is asked in the field of biomedicine, one of the main challenges is to understand the underlying semantic relatedness of the query and document sentences, and consequently extract the most non-redundant, query-relevant parts from the documents. Documents in biomedicine are very different from documents in other fields, and include very different document types (e.g. patient records, web documents, scientific papers, etc.) (Plaza et al., 2011). Therefore, particular characteristics and the type of

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 513–524, Osaka, Japan, December 11-17 2016.

biomedical documents are required to be exploited by the summarization systems. To this end, promising domain specific NLP techniques have been efficiently employed to release a repository of biomedical vocabularies named the Unified Medical Language System (UMLS[1]) (Bodenreider, 2004). UMLS is a very rich source of information in medical and biological domain. Therefore, most existing biomedical summarizers utilize UMLS as a large lexical and semantic medical ontology. However, UMLS does not provide a full coverage of non-medical concepts, terms, and relations included in general-purpose thesauri such as WordNet[2] (Huang et al., 2009). Moreover, utilizing WordNet to complement the UMLS coverage is challenging due to their different structures, natures, terms, and sizes.

This challenge has motivated us to provide a deeper analysis of biomedical texts by keeping an eye on the biomedical peculiarities. Given a clinical query and a set of relevant medical evidence, our aim is to generate a fluent, well-organized, and compact summary that answers the query. The quality of biomedical summaries is also enhanced by appraising the applicability of both general-purpose (WordNet), and domain-specific (UMLS) knowledge sources for concept discrimination. In details, our approach comprises different components: capturing underlying sentence-to-query and sentence-to-sentence semantic similarities using WordNet and UMLS; ranking and filtering sentences considering their similarity scores to the clinical query; clustering sentences by their relevance to each other; generating new summary sentences through a word graph representation by ensuring their importance and syntactic structure.

The rest of the paper is organized as follows. Section 2 summarizes the background. Utilized data and the preprocessing steps are discussed in Section 3. We demonstrate the proposed approach in Section 4. Section 5 reports the evaluation metrics and the performed experiments. Finally, Section 6 concludes the paper.

2 Background

Text summarization is the process of automatically creating a compressed version of a given text. A summary can either be query-focused (biased to a user query), or generic (conveying the document gist). In traditional query-focused summarization systems, lexical similarity measures are used to select content that are similar to the question. Such approaches also have to ensure that redundant information is minimized. Some recent researches have addressed query-focused text summarization from the question answering perspective (Yu and Cao, 2008), and some others have modeled summarization as a sentence classification problem (Cao et al., 2011). A machine learning classifier trained on a small dataset is employed in another study (Demner-Fushman and Lin, 2007) to select the summary sentences. Another summarization system (Cao et al., 2011) utilizes category of an input question to generate paragraph level summaries. They suggest that the generated summary should be customized with respect to the type of the question. More advanced summarization techniques such as LexRank (Erkan and Radev, 2004) incorporate graph-based methods. LexRank assumes a fully connected and undirected graph for the set of documents to be summarized.

Among the researches performed in the text summarization area, many studies (e.g. (Coumou and Meijman, 2006)) have also explored the obstacles associated with evidence-based medicine practice in the absence of pre-existing systematic reviews. When primary care physicians seek answers to clinical problems, the time required to search, evaluate, and synthesize evidence has been known as a critical factor (Sarker et al., 2016). Literature review and analysis may take a long time (e.g. it takes more than 30 minutes on average for a practitioner to find and extract evidence (Hersh et al., 2002)). Numerous IR approaches have already been proposed to address the search-related needs of practitioners (Hanbury, 2012). However, post-retrieval techniques (e.g. (Sarker et al., 2016)) to perform query-oriented summarization are still scarce. The complicated nature of biomedical texts and the limited amount of suitable annotated data for the task of summarization are the main reasons that raise various difficulties in progress (Athenikos and Han, 2010; Sarker et al., 2016).

To overcome the lack of incorporation of domain-specific information, UMLS came to play, and has proved to be a useful knowledge source for summarization in biomedical domain (Reeve et al., 2007).

[1]Developed by the U.S. National Library of Medicine (available at http://www.nlm.nih.gov/research/umls/)

[2]http://wordnet.princeton.edu

However, a decline is found in the performance of the summarizers which only utilize UMLS as their source of knowledge. The reason is that UMLS is less likely to cover all concepts included in the source text (Plaza et al., 2011). To compensate this deficiency, a question-oriented extractive system for biomedical multi-document summarization (i.e. (Shi et al., 2007)), utilized WordNet as a general-purpose lexicon to capture the concepts not covered by UMLS. They constructed a graph containing ontological concepts (general ones from WordNet, and specific ones from UMLS), name entities, and noun phrases. Our work differs in intent, and explores the utility of graph representation of both domain-independent (WordNet) and domain-specific (UMLS) lexicons for incorporating underlying textual semantic similarities as the main basis of an efficient biomedical summarizer. Next, we discuss the utilized data and the preprocessing steps.

3 Data and Preprocessing

To develop, test, and evaluate our approach, we employed the evidence-based medicine (EBM) corpus[3] gathered and annotated by (Mollá et al., 2015), which is the only available corpus for the task of EBM text summarization. This corpus is sourced from the Clinical Inquiries section of the Journal of Family Practice[4], and consists of 456 clinical queries, with 1396 bottom-line multi-document summaries (i.e. evidence-based answers). The total number of associated single-document evidence-based summaries is 3036, which are generated from 2908 unique articles. Table 1 lists the properties of this corpus. The bottom-line answers are used as the reference (gold) summaries. The question and all the abstracts associated with the bottom-line summary are also considered as the source texts.

total #clinical queries	456
#bottom-line multi-document summaries	1396
#single-document evidence summaries	3036
total #unique articles	2908

Table 1: Information about the EBM Corpus

The specific nature of the biomedical terminology makes it difficult to automatically process biomedical information (Nadkarni, 2000). One of these difficulties is caused by abbreviations (e.g. the use of *OCP* instead of *oral contraceptive pills*). In our approach, if the abstract includes abbreviations, their expansions are used to replace these shortened forms in the abstract body. If the abstract contains abbreviations and acronyms, but without any definition, the software[5] for abbreviation recognition and definition presented in (Hearst, 2003) is used. To remove the stopwords, we utilized the stopword list included in nltk[6] extended with the PubMed stopwords[7]. We also employed OpenNLP[8] to detect and split the sentences, and Stanford POS tagger (Toutanova et al., 2003) for tokenizing and part of speech tagging of each sentence.

4 The Proposed Approach

4.1 Measuring Semantic Similarity using WordNet and UMLS

Automatic summarization approaches rely on similarity comparison of sentences. For general English text, research on measuring relatedness has relied on WordNet, and for clinical and biomedical vocabularies, they are compiled into UMLS. Quantifying semantic relationships between linguistic terms lies at the core of many NLP applications (Pilehvar and Navigli, 2015). However, hard matching between words has long been an obstacle in identifying the relatedness of two sentences (ShafieiBavani et al., 2016b). We tackle this issue by dealing with concepts instead of terms, and with semantic relations

[3] Available at: http://sourceforge.net/projects/ebmsumcorpus
[4] http://www.jfponline.com/articles/clinical-inquiries.html
[5] Available at http://biotext.berkeley.edu/software.html
[6] http://nltk.org/
[7] http://www.ncbi.nlm.nih.gov/books/NBK3827/table/pubmedhelp.T.stopwords/
[8] http://opennlp.sourceforge.net/

instead of lexical or syntactical ones. In our approach, the main requirement for computing semantic similarities on WordNet and UMLS is Semantic Signature, which is firstly introduced as a multinomial distribution generated from repeated random walks on WordNet (Pilehvar and Navigli, 2015). We utilize this concept to capture the semantic similarities on both WordNet and UMLS. Note that in our work, a query is treated as a long single sentence.

Semantic Signature on WordNet To construct each semantic signature on WordNet, we make use of WordNet 3.0 (Fellbaum, 1998) repository. We also employ an alignment-based sense disambiguation algorithm presented in (Pilehvar and Navigli, 2015) to disambiguate each word. This algorithm leverages the content of the paired sentence in order to disambiguate each element. Afterwards, an iterative method for calculating Personalized PageRank has been used. The key assumption is that repeated random walks beginning at a sense (node) or a set of senses (seed nodes) in the WordNet network can provide a frequency or multinomial distribution over all the senses in WordNet. A higher probability will then be assigned to senses that are frequently visited from the seeds.

Consider an adjacency matrix *M* for the WordNet network, where edges connect senses according to the relations defined in WordNet (e.g. hypernymy and meronymy). The probability distribution for the starting location of the random walker in the network is denoted by $\vec{w}^{(0)}$. Given the set of senses *S* in a sentence, the probability mass of $\vec{w}^{(0)}$ is uniformly distributed across the senses $s_i \in S$, with the mass for all $s_i \notin S$ set to zero. The PageRank vector is then computed using Equation 1.

$$\vec{w}^{(t)} = (1-\alpha)M\vec{w}^{(t-1)} + \alpha\vec{w}^{(0)} \tag{1}$$

where at each iteration, the random walker may jump to any node $s_i \in S$ with probability $\alpha/|S|$. Following the standard convention, the value of α is set to 0.15. The number of iterations is also set to 30, which is sufficient for the distribution to converge. The resulting probability vector $\vec{w}^{(t)}$ is the semantic signature of the sentence, as it has aggregated its senses similarities over the entire graph. The UKB[9] implementation of Personalized PageRank has been used in this step.

Semantic Signature on UMLS Each semantic signature on UMLS is constructed by performing iterative random walks over the graph representation of version 2015AB of the UMLS Metathesaurus. This algorithm has previously been utilized for query expansion (Martinez et al., 2014). Metathesaurus, Semantic Network and SPECIALIST Lexicon are three major components of UMLS. Our approach focuses on the UMLS Metathesaurus, which contains a wide range of information about the relations between terms in the form of database tables. Among them, MRREL table lists different relations between concepts (i.e. *parent*, *can be qualified by*, and *related and possibly synonymous*). We consider the UMLS concepts as nodes (seeds), and the relations listed in MRREL table as directed edges.

Besides, we employ version 2016 of the MetaMap[10] program to map each sentence to concepts from the UMLS Metathesaurus and semantic types from the UMLS Semantic Network. Using the built-in WSD module, MetaMap allows to disambiguate terms, and returns directly the relevant concept. A broad range of concepts from very generic UMLS semantic types, that have already been considered in capturing WordNet-based semantic similarities, are discarded in this step. These semantic types are defined as *quantitative concept, qualitative concept, temporal concept, functional concept, idea or concept, intellectual product, mental process, spatial concept,* and *language* (Plaza et al., 2011). Thus, only concepts of the rest of semantic types are considered for constructing the semantic signature. Table 2 provides an example of mapping a sentence by MetaMap. Same as WordNet-based semantic signature, the UKB implementation of Personalized PageRank is utilized, but on UMLS.

Let *N* be an adjacency matrix for the UMLS graph with all relations in MRREL. The random walker starts in any of the concepts included in the sentence, and randomly follows one of the relations to another concept. With certain probability, the random walker would restart in any of the concepts, and continue its walk. Finally, the number of visits to each concept in the graph would give an indication of how related that concept is to the sentence terms. The result is a probability distribution over UMLS

[9] http://ixa2.si.ehu.es/ukb/

[10] Developed by the U.S. National Library of Medicine (available at https://metamap.nlm.nih.gov)

Score	Concept	Semantic Type	Considered
862	No evidence of	Qualitative Concept	✘
593	Increase	Functional Concept	✘
593	Risk	Idea or Concept	✘
578	Major	Qualitative Concept	✘
744	Hemorrhage	Pathologic Function	✔
578	Result	Functional Concept	✘
578	Accidental Falls	Injury or Poisoning	✔
1000	Hospitalized Patients	Patient or Disabled Group	✔
966	Take	Health Care Activity	✔
1000	Warfarin	Pharmacologic Substance	✔

Table 2: MetaMap mapping for the sentence "*There is no evidence of increased risk for major bleeding as a result of falls in hospitalized patients taking warfarin.*"

concepts. The higher the probability for a concept, the more related it is to the given sentence. The probability distribution for the starting location of the random walker in the network is denoted by $\vec{u}^{(0)}$. Having the set of MetaMap concepts *C* in a sentence, the probability mass of $\vec{u}^{(0)}$ is uniformly distributed across the concepts $c_i \in C$, with the mass for all $c_i \notin C$ set to zero. The PageRank vector is then computed using Equation 2.

$$\vec{u}^{(t)} = (1-\beta)N\vec{u}^{(t-1)} + \beta\vec{u}^{(0)} \tag{2}$$

where at each iteration, the random walker may jump to any node $c_i \in C$ with probability $\beta/|C|$. Following the standard convention, the value of β is set to 0.15. The number of iterations is also set to 30, which is sufficient for the distribution to converge. The resulting probability vector $\vec{u}^{(t)}$ is the semantic signature of the sentence on UMLS, as it has aggregated its concepts similarities over the entire graph.

Semantic Similarities at the Sentence Level For comparing pairs of semantic signatures at the sentence level, we use Weighted Overlap (WO) algorithm proposed by (Pilehvar and Navigli, 2015). This algorithm first sorts the two signatures according to their values and then harmonically weights the overlaps between them. Using the knowledge source N (i.e. WordNet or UMLS), WO calculates the semantic similarity (Sim_N) of two sentence signatures S_{N1} and S_{N2} as:

$$Sim_N(S_{N1}, S_{N2}) = \frac{\sum_{h \in H} (r_h(S_{N1}) + r_h(S_{N2}))^{-1}}{\sum_{i=1}^{|H|} (2i)^{-1}} \tag{3}$$

where *H* denotes the intersection of all senses/concepts with non-zero probability (dimension) in both signatures, and $r_h(S_{Nj})$ denotes the rank of the dimension *h* in the sorted signature S_{Nj}, where rank 1 denotes the highest rank. The denominator is also used as a normalization factor that guarantees a maximum value of one. The minimum value is zero and occurs when there is no overlap between the two signatures, i.e. $|H| = 0$.

To estimate the final semantic similarity score between two sentences, we conducted a set of experiments using the WordNet-based semantic similarities (Sim_W), and/or UMLS-based semantic similarities (Sim_U), and obtained the best result while using both scores with different weights according to Equation 4.

$$Sim_{final}(S_1, S_2) = \mu \times Sim_U(S_{U1}, S_{U2}) + (1-\mu) \times Sim_W(S_{W1}, S_{W2}) \tag{4}$$

where $Sim_U(S_{U1}, S_{U2})$ denotes the semantic similarity score between two sentence signatures on UMLS. The semantic similarity score between two sentence signatures on the WordNet is also shown by $Sim_W(S_{W1}, S_{W2})$. The scaling factor μ was optimized on development data in our experiments and set to 0.6 to reach the best result (Section 5).

4.2 Constructing Similarity Graph

To filter out less query relevant information, sentences are modeled as a Similarity Graph - a weighted undirected graph on which each node represents a sentence and the edge weight carries the similarity

of two sentences (ShafieiBavani et al., 2016b). For more clarity, let $S = \{s_1, s_2, ..., s_n\}$, be a set of sentences, and $(S_{ij})_{i,j=1,...,N}$ be the similarity matrix in which each element indicates the similarity $S_{ij} \geq 0$ between two sentences S_i and S_j (pairwise similarity scores are already achieved in Section 4.1). Hence, the input query and the abstract sentences are considered as nodes on the graph, where we consider two kinds of edge for each node: (1) sentence-to-query similarity edge; (2) sentence-to-sentence similarity edge. The achieved similarity weight for each sentence-to-query and sentence-to-sentence relation is assigned to its corresponding edge in our similarity graph. Considering the combination of sentence-to-query and sentence-to-sentence similarities, our model decides which sentences are relevant to the query, and should be kept for the further clustering step. To this end, we employ a combination model (Chali et al., 2011):

$$C(S_i|Q) = \gamma \times \frac{Sim_{final}(S_i, Q)}{\sum_{S_j \in A} Sim_{final}(S_j, Q)} + (1 - \gamma) \times \sum_{S_k \in A} \frac{Sim_{final}(S_i, S_k)}{\sum_{S_j \in A} Sim_{final}(S_j, S_k)} \times C(S_k|Q) \tag{5}$$

where $C(S_i|Q)$ denotes the score of a sentence S_i given a query Q. A contains all sentences in the abstract set. The weighting parameter $0 \leq \gamma \leq 1$ is used to specify the relative contribution of two similarities: the similarity of a sentence to the query and similarity to the other sentences in the abstract set. Previous experiments (Chali et al., 2011) lead us to choose 0.4 as the best value of γ. The denominators in both terms are for normalization. $Sim_{final}(S_i, S_k)$ is the weight of the edge between two sentence nodes S_i and S_k. Likewise, $Sim_{final}(S_i, Q)$ is the weight of the edge connecting the sentence node S_i to the query node Q. Finally, sentences with $C \geq \delta$ with the best empirical value of 0.5 for δ are picked among the set of sentences. This step results in a subgraph comprising a set of the most query-relevant sentences to be clustered in the next step.

4.3 Clustering Relevant Sentences

In this step, we use Chinese Whispers (CW) which is a graph-based clustering algorithm proposed by (Biemann, 2006). CW is a basic - yet effective - parameter-free algorithm to partition the nodes of graphs in a bottom-up fashion. This algorithm is also a special case of Markov-Chain-Clustering, but time-linear in the number of edges. So, the power of CW lies in its capability of handling very large graphs in reasonable time. First, a distinct class is assigned to each node , and a clustering C containing the singleton clusters c_i is created. Then, a series of iterations is performed to merge the clusters. Specifically, at each iteration the algorithm analyzes each node s in random order and assigns it to the majority class among those associated with its neighbors. In other words, it assigns each node s to the class c that maximizes the sum of the weights of the edges s_i, s_j incident on s_j such that c is the class of s_i (Equation 6).

$$class(s_j) = \underset{c}{argmax} \sum_{\substack{\{s_i, s_j\} \in E(G) \\ s.t. class(s_i) = c}} Sim(s_i, s_j) \tag{6}$$

As soon as an iteration produces no change in the clustering, the algorithm stops and outputs the final clustering. The result of CW is a hard partitioning of the given graph into a number of clusters. Although it is possible to obtain a soft partitioning in CW, we prefer hard partitioning to keep the redundancy low.

4.4 Word Graph-based Summarization of EBM

For each obtained cluster, we build a word graph by iteratively adding sentences to it. This graph is an ordered pair $G = (V, E)$ comprising of a set of vertices (nodes) V, together with a set of directed edges E, which shows the adjacency between corresponding nodes. The graph is first constructed by the first sentence and displays words in a sentence as a sequence of connected nodes. The first word is the start node and the last one is the end node. Words are added to the graph in three steps of the following order: (1) non-stopwords for which no candidate exists in the graph; or for which an unambiguous mapping is possible; (2) non-stopwords for which there are either several possible candidates in the graph; or for which they occur more than once in the sentence; (3) stopwords. As mentioned in Section 3, for the last group, we use the stopword list included in nltk extended with the PubMed stopwords.

Where mapping in the graph is ambiguous (i.e. there are two or more nodes in the graph that refer to the same word/POS pair), we follow the instructions stated by (Filippova, 2010): the immediate context (the preceding and following words in the sentence, and the neighbouring nodes in the graph) or the frequency (i.e. the node which has words mapped to it) is used to select the candidate node. A new node is created only if there are no suitable candidates to be mapped to, in the graph. Conducting this step not only removes the redundancy, but also makes use of redundant parts to indicate the salient path (Figure 1 (a)). Edge weights are calculated using the weighting function defined in Equation 7 (Filippova, 2010).

$$W(e_{i,j}) = \frac{(freq(i) + freq(j))/\sum_{s \in S} diff(s,i,j)^{-1}}{freq(i) \times freq(j)} \tag{7}$$

where $freq(i)$ is the number of words mapped to the node i. The function $diff(s,i,j)$ refers to the distance between the offset positions of words i and j in sentence s.

Utilizing Synonymy To reduce the redundancy caused by existing synonym words in the sentences, we use the synsets in WordNet to identify synonym representative candidate if available. For example, consider n different sentences containing words *biliary*, *bilious*, *tumor*, *tumour*, and *neoplasm*. The first two words, and the latter three ones are synonyms of each other. Assume each sentence contains one of these possible combinations (i.e. biliary tumor, biliary neoplasm, biliary tumour, bilious tumor, bilious neoplasm, bilious tumour). Without an appropriate synonym mapping based on a notion of synonymy, several synonym nodes will be added to the word graph as separate nodes. We consider their frequency to pick one of them as the representative of its synonyms from the other sentences. The weight of the obtained node is computed by summing the frequency scores from the other nodes (Figure 1 (b)).

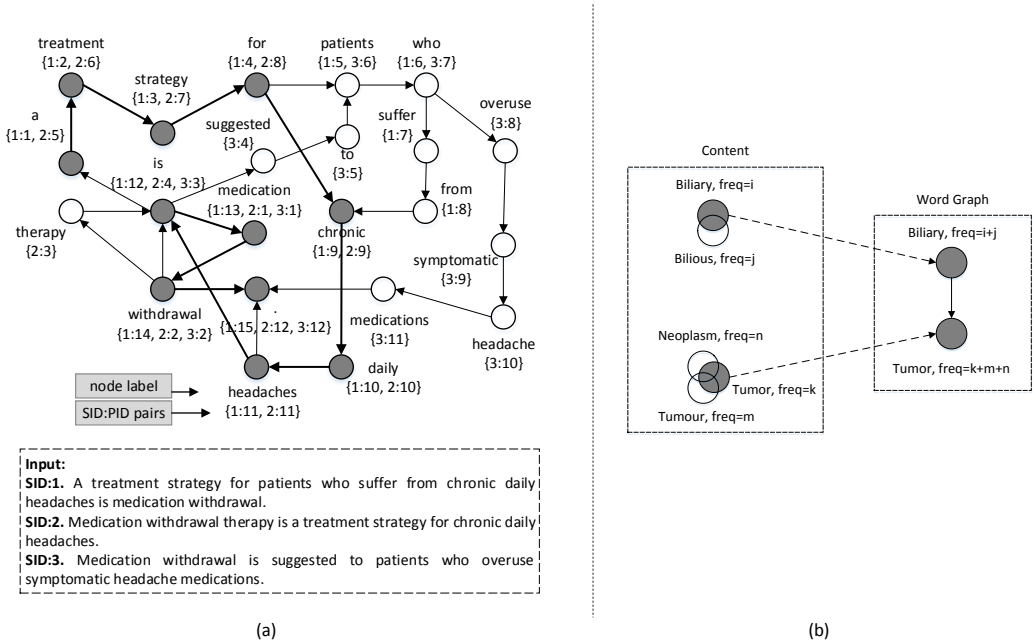


Figure 1: (a) An example of the Constructed Word Graph. Thick edges indicate salient paths. (b) An example of Biomedical Synonym Mapping

Ensuring Information Richness To re-rank the summary candidates based on the information richness, important key-phrases have been exploited using the TextRank algorithm (Mihalcea and Tarau, 2004). Hence, a word recommends other co-occurring words, and the strength of the recommendation is recursively computed based on the importance of the words making the recommendation. The score of a key-phrase *k* is computed by summing the salience of the words it contains, normalized with its $length + 1$ to favor longer n-grams. The paths are then re-ranked based on their key-phrases and the score of a summary candidate c is given by Equation 8.

$$Score_{Key}(c) = \frac{\sum_{i,j \in path(c)} W(e_{i,j})}{length(c) \times \sum_{k \in c} (\frac{\sum_{w \in k} TextRank(W)}{length(k)+1})} \quad (8)$$

The heuristic algorithm discussed in (Boudin and Morin, 2013) is then used to find the k-shortest paths ($k = 50$ throughout our experiments) from start to end node in the graph. Paths shorter than eight words or do not contain a verb are filtered before re-ranking. The remaining paths are re-ranked and the path that has the lightest average edge weight is eventually considered as the richest summary sentence.

Ensuring Syntactic Structure Since our word graph generates new summary sentences, we need to ensure the grammatical structure of these newly constructed sentences. So, we build a part-of-speech based language model (POS-LM) to re-rank the paths in our word graph (ShafieiBavani et al., 2016a). The POS-LM assigns a score to each generated summary in terms of grammatical structure, and helps in identifying the most grammatical sentence among the k-richest sentences. It estimates the probability of string of m POS tags by $p(t_1^m) \propto \prod_{i-1}^{m} p(t_i|t_{i-n+1}^{i-1})$ (Monz, 2011), where *n* is order of the language model, and t_i^j refers to the sub-sequence of tags from position *i* to *j*.

To build a POS-LM, we make use of Stanford POS tagger to annotate a large part (∼100 M-words) of the BioMed Central full-text corpus for text mining research[11]. Then, we remove all words from the pairs of words/POS in the POS annotated corpus. Finally, the SRILM toolkit (Stolcke and others, 2002) is employed to collect n-gram statistics. The candidate sentences also need to be annotated with POS tags, and the score of each summary is estimated by the 7-gram language modeling, based on its sequence of POS tags. To re-rank the obtained paths, POS-LM gives the perplexity score ($Score_{LM}$), which is the geometric average of $1/probability$ of each sentence, normalized by the number of words. So, $Score_{LM}$ for each sequence of POS in the k-richest sentences is computed by Equation 9.

$$Score_{LM}(c) = 10^{\frac{\log prob(c)}{|word|}} \quad (9)$$

where $prob(c)$ is the probability of summary candidate C including $|word|$ number of words, computed by the 7-gram POS-LM. A unity-based normalization is then used to bring the values of $Score_{Key}(c)$ in Equation 8, and the score of POS-LM into the range [0, 1]. The score of each summary is finally given by Equation 10.

$$Score_{final}(c) = \eta \times Score_{Key}(c) + (1 - \eta) \times Score_{LM}(c) \quad (10)$$

The scaling factor η was optimized on development data in our experiments and set to 0.4 (Section 5). Hence, the most grammatical candidate among the candidates that contain the most important phrases, has been selected as the summary for each cluster. All automatic summaries were generated by selecting sentences until the summary is 30% of the original document size (Plaza et al., 2011). This choice of the summary size is based on the well-accepted heuristic that a summary should be between 15% and 35% of the size of the source text. Considering this convention, we pick a number of three summary sentences (based on their sentence-to-query similarity scores) to answer the corresponding clinical query.

5 Experiments

In our work, the generated summaries are assessed automatically through version 2.0 [12] of ROUGE (Lin, 2004) over the released EBM corpus by (Mollá et al., 2015). ROUGE measures the summary quality by counting the overlapping units between system-generated summaries and human-written reference/gold summaries. We used ROUGE F-measure for unigram, bigrams, and SU4 (skip-bigram with maximum gap length 4) to evaluate the generated summaries. The bottom-line answers in the EBM corpus have also been used as the reference summaries.

To investigate the effectiveness of our approach, we compare our summarizer with *FastSum* (Schilder and Kondadadi, 2008), and a research prototype *LexRank* (Erkan and Radev, 2004). FastSum is a fast query-focused multi-document summarization system based only on word frequency features of topics,

[11] http://old.biomedcentral.com/about/datamining
[12] http://kavita-ganesan.com/content/rouge-2.0

documents, and clusters. Each sentence is ranked based on a linear function of scores using a variety of frequency measures. A support vector machine regression is also used to learn weights of the features. Comparing our approach with FastSum would let us evaluate the superiority of our approach over the word frequency-based approaches on the task of query-focused multi-document summarization. LexRank is a topic-oriented generic summarizer that focuses on multi-document extractive text summarization, and extracts the information in the text that is related to the user specified topic. This prototype outperformed both centroid-based methods and other systems participating in DUC in most of the cases (Erkan and Radev, 2004). Comparison with LexRank will allow us to evaluate whether semantic information provides benefits over merely lexical information in graph-based summarization approaches. Table 3 shows an example of a summary generated by human (Gold), our proposed approach (Proposed), and LexRank.

Question: Are major bleeding events from falls more likely in patients on warfarin?
Gold Summary: There is no evidence of increased risk for major bleeding as a result of falls in hospitalized patients taking warfarin. [*PubMed IDs: 7668955, 15638939*]
Proposed Summary One study found no difference in major bleeding complications between patients taking anticoagulation therapy with not taking. Criteria for taking warfarin were not reported. Prescribing warfarin for patients judged less likely to fall.
LexRank Summary No major hemorrhagic complications were seen following 131 falls in the anticoagulation group (93 patients) and 269 falls in the group not on anticoagulation (175 patients). The study was limited because most falls were from a seated position or partially controlled by an attendant. Major hemorrhage was defined as bruising or cuts requiring immediate attention from a physician.

Table 3: An example of Gold summary, Proposed summary, and LexRank summary

Three different baselines for sentence selection have also been used, each aiming to construct a different type of summary according to the type of information in various parts of the source. In details, we pick the first and last third sentences of each set of abstracts related to a clinical query, so called (*first part*, and *last part*). We also consider all sentences included in the abstracts related to a clinical query as *whole part*. Afterwards, included sentences of each of these three parts are considered as the input bag of sentences for the following baselines:

- Head Baseline: This baseline is used in a variety of summarization applications, specifically in the news summarization area. In our work, this baseline generates summaries by unintentionally selecting three sentences from the *first part*.
- Random Baseline: Randomly selects three sentences from the *whole part*.
- Tail Baseline: The last sentences in the medical abstracts usually provide conclusions. Hence, this has been used as a baseline for summarization of biomedical texts (Demner-Fushman and Lin, 2007). In our work, this baseline generates summaries by selecting three sentences at random from the *last part*.

The average performance of the baseline systems and the proposed approach in terms of ROUGE scores are provided in Table 4.

System	ROUGE-1	ROUGE-2	ROUGE-SU4
Head Baseline	0.2710	0.1723	0.1593
Random Baseline	0.2623	0.1801	0.1509
Tail Baseline	0.2866	0.1834	0.1607
FastSum	0.3382	0.2081	0.188
LexRank	0.3407	0.2069	0.1938
Proposed	**0.3985**	**0.2450**	**0.2259**

Table 4: Average scores by ROUGE metrics over the EBM corpus

The statistics point out the effectiveness of our summarizer over the compared systems on all evaluation metrics. Besides, considering the results obtained by *Tail Baseline*, it has been realized that the last part of each abstract is more likely to be included in the summary.

Standard Deviation of ROUGE Scores Since Table 4 demonstrates the average results, an important research question that immediately arises is how much the ROUGE scores differ across the abstracts. Hence, the standard deviation of different ROUGE scores for the summaries generated by the proposed approach are shown in Table 5.

Metric	ROUGE-1	ROUGE-2	ROUGE-SU4
Standard Deviation	0.02104	0.03250	0.03079

Table 5: Standard deviation of ROUGE scores for the summaries generated by the proposed approach

Exploring Scaling Factors In our work, two free parameters are defined: (1) μ for measuring semantic similarities using WordNet and UMLS; (2) η for final re-ranking score of each generated summary sentence. We randomly selected 30% of the EBM corpus as our development set to tune these parameters. Figure 2 shows the results obtained by ROUGE-1 F-Measure, using different values for μ and η. The best results are obtained using $\mu = 0.6$, and $\eta = 0.4$. Performance deteriorates when the UMLS portion in measuring semantic similarities is less or more than 0.6. On the other hand, when contribution of *TextRank* score is whatever except 0.4, the performance gradually decreases. The lowest performance is obtained when *TextRank* score is ignored in re-ranking the generated summary sentences, and also when the UMLS semantic signature occupies 0.9 of whole 1.0 value of final semantic similarity measure. This demonstrates the importance of using both WordNet and UMLS to capture the semantic similarities.

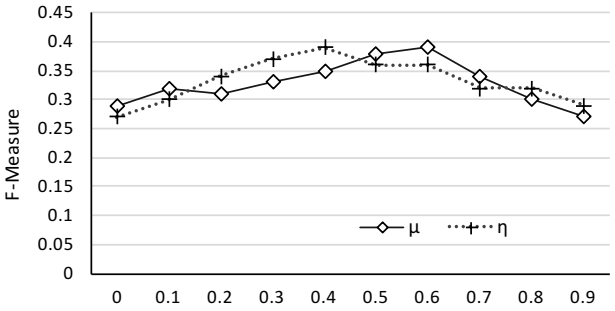


Figure 2: Exploring scaling factors μ and η on the development set

6 Conclusions

We have presented an effective approach for summarizing biomedical texts. Given a clinical query, our approach generates a well-organized, informative summary from a set of related biomedical abstracts through: (1) repetitive random walks on WordNet and UMLS to capture semantic similarities between sentences and the input query; (2) filtering out less query-relevant sentences; (3) clustering the remaining relevant sentences; (4) summarizing the clusters through a word graph-based approach, which considers the important key-phrases along with the syntactic structure of the generated summaries. Based on an automatic evaluation (via ROUGE metrics) using an evidence-based medicine corpus, our approach outperforms the two competitive systems. It has also been found that the last part of each abstract is more likely to be included in the summary. We have tackled the main issue faced by state-of-the-art biomedical summarizers (i.e. decline in summarization efficiency due to the poor UMLS coverage of general concepts in the documents to be summarized) (Plaza et al., 2011). This issue is addressed by using WordNet to represent the layman knowledge, and UMLS to represent the professional knowledge. We believe that this approach can bridge the knowledge and language gaps in biomedical summarizers.

Acknowledgements

We thank the anonymous reviewers for their insightful comments and valuable suggestions.

References

Sofia J Athenikos and Hyoil Han. 2010. Biomedical question answering: A survey. *Computer Methods and Programs in Biomedicine*, 99(1):1–24.

Chris Biemann. 2006. Chinese whispers: An efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the first Workshop on Graph Based Methods for Natural Language Processing*, pages 73–80. Association for Computational Linguistics.

Olivier Bodenreider. 2004. The unified medical language system (umls): Integrating biomedical terminology. *Nucleic Acids Research*, 32(suppl 1):D267–D270.

Florian Boudin and Emmanuel Morin. 2013. Keyphrase extraction for n-best reranking in multi-sentence compression. In *North American Chapter of the Association for Computational Linguistics*.

Yonggang Cao, Feifan Liu, Pippa Simpson, Lamont Antieau, Andrew Bennett, James J Cimino, John Ely, and Hong Yu. 2011. Askhermes: An online question answering system for complex clinical questions. *Journal of Biomedical Informatics*, 44(2):277–288.

Yllias Chali, Sadid A Hasan, and Shafiq R Joty. 2011. Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels. *Information Processing & Management*, 47(6):843–855.

Herma CH Coumou and Frans J Meijman. 2006. How do primary care physicians seek answers to clinical questions? a literature review. *Journal of the Medical Library Association*, 94(1):55.

Dina Demner-Fushman and Jimmy Lin. 2007. Answering clinical questions with knowledge-based and statistical techniques. *Computational Linguistics*, 33(1):63–103.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, pages 457–479.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Katja Filippova. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 322–330. Association for Computational Linguistics.

Marcelo Fiszman, Thomas C Rindflesch, and Halil Kilicoglu. 2004. Abstraction summarization or managing the biomedical research literature. In *Proceedings of the HLT-NAACL Workshop on Computational Lexical Semantics*, pages 76–83. Association for Computational Linguistics.

Evidence-Based Medicine Working Group and Others. 1992. Evidence-based medicine. a new approach to teaching the practice of medicine. *The Journal of the American Medical Association*, 268(17):2420.

Allan Hanbury. 2012. Medical information retrieval: An instance of domain-specific search. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1191–1192. Association for Computing Machinery.

MAAS Schwartz Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text.

William R Hersh, M Katherine Crabtree, David H Hickam, Lynetta Sacherek, Charles P Friedman, Patricia Tidmarsh, Craig Mosbaek, and Dale Kraemer. 2002. Factors associated with success in searching medline and applying evidence to answer clinical questions. *Journal of the American Medical Informatics Association*, 9(3):283–293.

Kuo-Chuan Huang, James Geller, Michael Halper, Yehoshua Perl, and Junchuan Xu. 2009. Using wordnet synonym substitution to enhance umls source integration. *Artificial Intelligence in Medicine*, 46(2):97–109.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8.

David Martinez, Arantxa Otegi, Aitor Soroa, and Eneko Agirre. 2014. Improving search over electronic health records using umls-based query expansion through random walks. *Journal of Biomedical Informatics*, 51:100–106.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. Association for Computational Linguistics.

Diego Mollá, María Elena Santiago-Martínez, Abeed Sarker, and Cécile Paris. 2015. A corpus for research in text processing for evidence based medicine. *Language Resources and Evaluation*, pages 1–23.

Christof Monz. 2011. Statistical machine translation with local language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 869–879. Association for Computational Linguistics.

PM Nadkarni. 2000. E-medicine-information retrieval in medicine: Overview and applications.

Mohammad Taher Pilehvar and Roberto Navigli. 2015. From senses to texts: An all-in-one graph-based approach for measuring semantic similarity. *Artificial Intelligence*, 228:95–128.

Laura Plaza, Alberto Díaz, and Pablo Gervás. 2011. A semantic graph-based approach to biomedical summarisation. *Artificial Intelligence in Medicine*, 53(1):1–14.

Lawrence H Reeve, Hyoil Han, and Ari D Brooks. 2007. The use of domain-specific concepts in biomedical text summarization. *Information Processing & Management*, 43(6):1765–1776.

David L Sackett, William MC Rosenberg, JA Muir Gray, R Brian Haynes, and W Scott Richardson. 1996. Evidence based medicine: what it is and what it isn't. *British Medical Journal*, 312(7023):71–72.

Abeed Sarker, Diego Mollá, and Cécile Paris. 2015. Automatic evidence quality prediction to support evidence-based decision making. *Artificial Intelligence in Medicine*, 64(2):89–103.

Abeed Sarker, Diego Mollá, and Cecile Paris. 2016. Query-oriented evidence extraction to support evidence-based medicine practice. *Journal of Biomedical Informatics*, 59:169–184.

Frank Schilder and Ravikumar Kondadadi. 2008. Fastsum: fast and accurate query-based multi-document summarization. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 205–208. Association for Computational Linguistics.

Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong, and Fang Chen. 2016a. An efficient approach for multi-sentence compression. In *JMLR: Workshop and Conference Proceedings*.

Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong, and Fang Chen. 2016b. A query-based summarization service from multiple news sources. In *Services Computing (SCC), 2016 IEEE International Conference on*, pages 42–49. IEEE.

Zhongmin Shi, Gabor Melli, Yang Wang, Yudong Liu, Baohua Gu, Mehdi M Kashani, Anoop Sarkar, and Fred Popowich. 2007. Question answering summarization of multiple biomedical documents. In *Advances in Artificial Intelligence*, pages 284–295. Springer.

Andreas Stolcke et al. 2002. Srilm-an extensible language modeling toolkit. In *INTERSPEECH*, volume 2002, page 2002.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Hong Yu and YongGang Cao. 2008. Automatically extracting information needs from ad hoc clinical questions. In *American Medical Informatics Association*.

Large-scale Multi-class and Hierarchical Product Categorization for an E-commerce Giant

Ali Cevahir
RIT Tokyo, Rakuten Inc.,
Rakuten Crimson House,
1-14-1 Tamagawa, Setagaya-ku, Tokyo
ali.cevahir@rakuten.com

Koji Murakami
RIT NY, Rakuten USA,
215 Park Avenue South,
New York NY 10003
koji.murakami@rakuten.com

Abstract

In order to organize the large number of products listed in e-commerce sites, each product is usually assigned to one of the multi-level categories in the taxonomy tree. It is a time-consuming and difficult task for merchants to select proper categories within thousands of options for the products they sell. In this work, we propose an automatic classification tool to predict the matching category for a given product title and description. We used a combination of two different neural models, i.e., deep belief nets and deep autoencoders, for both titles and descriptions. We implemented a selective reconstruction approach for the input layer during the training of the deep neural networks, in order to scale-out for large-sized sparse feature vectors. GPUs are utilized in order to train neural networks in a reasonable time. We have trained our models for around 150 million products with a taxonomy tree with at most 5 levels that contains 28,338 leaf categories. Tests with millions of products show that our first predictions matches 81% of merchants' assignments, when "others" categories are excluded.

1 Introduction

E-commerce has grown rapidly in recent years. Giant e-commerce companies like Amazon, e-Bay, Taobao and Rakuten list millions of products on their sites sold by thousand of merchants. As of May 2016, Japan's largest e-commerce site Rakuten Ichiba[1] hosted 186 million active products sold by 43,363 different merchants. In order to organize products so that customers can navigate and search them easily, products are categorized into multi-level categories. "Women's Fashion > Tops > Sweaters > Long-sleeved knit > Crew neck" is an example for such categories. Rakuten Ichiba contains around 30 thousand categories of up to 5 levels. Merchants need to manually assign each product to one of those categories, which is a tedious task and prone to error. Moreover, merchants may not be accurate while assigning products, the cateogry assignment of the same product listed by different merchants may not be consistent. Automatic category recommendation for given product information helps to solving these problems.

Product classification is a text classification with a large hierarchical product taxonomy, and a lot of research have been conducted with various methodologies (Gupta et al., 2016; Shen et al., 2012; Kozareva, 2015; Qiu et al., 2011). Product classification includes the following challenges: 1) the products sparsely distributed in a large number of categories and data distribution is quite skewed, 2) the length of product titles and descriptions is diverse, 3) even though there are tons of product data, it is not guaranteed that pairs of current product title and assigned category are correct.

In this paper we propose a large-scale classification method for e-commerce products to classify them into thousands of multi-level categories. We use 172 million product title and description data for making predictions on products from Rakuten Ichiba. Please note that our techniques can be applied to other languages as well. Titles and descriptions are preprocessed to extract words, model numbers, sizes and

[1] http://www.rakuten.co.jp

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 525–535, Osaka, Japan, December 11-17 2016.

counts. Deep belief nets (DBN) and deep autoencoders (DAE) are used to build models for given data sources. Combining results from different models and different data sources, final predictions are made. Input feature vectors are very high dimensional and sparse for the deep models we train, which makes it impractical to train in usual way. We apply the selective reconstruction idea by Dauphin et al. (2011) for training DBN and DAE. DBN and DAE with selective reconstruction are implemented on GPUs in order to process a large product base within a reasonable amount of time. Conventional methods like multinomial Naive Bayes are not practical to be used in that scale. We compared our results with passive-aggressive learning, and confirmed large accuracy improvement.

We can summarize our contributions as follows:

- We propose a large-scale classification method for e-commerce products to classify them into 28,338 categories organized in 5 level.
- E-commerce-specific features, like product models, sizes, counts are extracted from a corpus which mainly contains Japanese text.
- DBN and DAE with selective input reconstruction are implemented on GPUs.
- We conducted experiments with 172 million product titles.
- Our comparisons with merchants' assignments suggest 81% match, when "others" categories are excluded. We observed that our predictions can sometimes be more accurate than human labeling.

The remainder of this paper is organized as follows. Section 2 introduces related work. Section 3 and Section 4 overview exploited deep models and explain our proposed framework. In Section 5 we explain tokenization and feature extraction from product data. Section 6 presents experimental results of product classification, with varying settings. Section 7 concludes the paper.

2 Related Work

2.1 Text classification in large taxonomies

Xue et al. (2008) worked on deep classification in large-scale text taxonomy. They proposed two stage algorithm consisting of a search stage and classification stage. A language model was trained with over 1 million web documents and 130,000 documents was classified into over 130,000 categories. Qiu et al. (2011) proposed a variant Passive-Aggressive (PA) algorithm with latent concepts and evaluated the method with LSHTC dataset[2], which include over 13,000 categories and over 100,000 documents. Kosmopoulos (2015) proposed an extended hierarchical classification to predict the correct leaf by estimating the probability of each root-to-leaf path. LSHTC dataset has also been used to evaluate the method. Ha-Thuc et al.(2011) exploited classification approach without labeled data. In their algorithm, ontological knowledge was used to define the meaning of categories instead of relying on human-labelled documents. A typology consisting of 1,131 categories was used in the evaluation.

2.2 Product classification

There are various works devoted to multi-level category predictions for e-commerce products. Chen and Warren (2013) used multi-class-SVM with cost-sensitive function. They used 1,073 categories from UNSPSC taxonomy [3], which includes over 17,000 categories, and over 1 million products. Gupta et. al (2016) used word clustering and idf values to obtain document vector from product description and showed this document representation worked well in their product classification. Kozareva (2015) has worked on product classification with Yahoo! product data. They compared several classifiers and 5 kinds of features, and showed neural network embedding representation outperformed in product classification with over 300 categories in their category taxonomy.

Shen et. al (2011; 2012) proposed hierarchical classification, which decomposes into a coarse level and a fine level task, and used graph algorithm to discover automatically groups of highly similar classes as product category instead of relying on human-defined hierarchy. The model was trained with 83 million products from eBay.

[2] http://lshtc.iit.demokritos.gr
[3] http://www.unspsc.org/

2.3 Text classification with deep learning

In the last couple of years, deep learning algorithms have been exploited to text classification.

Zhang et al. (2015) showed that character-level convolutional network is an effective method in the text classification, and compared various models, including BoW (Bag of Words), word embedding, word-based ConvNet and long-short term memory with several kinds of large-scale dataset. Lai et al. (2015) have also worked on recurrent convolutional neural network (RCNN). A recurrent structure was used to capture contextual information as far as possible when learning word representations. They compared various models with 4 kinds of datasets, including small number of classes and middle size of instances. Kim (2014) reported sentence classification with Convolutional Neural Network. A simple improvement was considered to the convolutional architecture that two input channels are used to allow the employment of task-specific and static word embeddings simultaneously. Evaluation has been conducted with 6 kinds of datasets, including a few classes and small number of instances. Wang et al. (2015) worked on semantic clustering and convolutional neural network for short text classification and used 2 kinds of datasets, which consists of small number of classes and instances.

Ha et. al (2016) used deep learning-based product classification method, which consists of multiple recurrent neural networks (RNNs). They evaluated the method with more than 94 million products with approximately 4,100 leaf categories from NAVER shopping.

3 Deep Models Exploited

Recently, neural models gained attention for classification and semantic compression tasks. Deep belief nets are multiple layer neural networks used for classification tasks. They contain all-to-all connections between layers. Top layer of a DBN represents class probabilities of a given input vector. Hinton et al. (2006) proposed greedy layer-wise pre-training for DBN. Each layer (Restricted Boltzmann Machines) is trained by constructive divergence, using 1-step Gibbs sampling. Given class labels for top layer, the network is fine-tuned after greedy layer-wise pre-training is completed.

Deep autoencoders are used for finding compressed representations, i.e., semantic hashes (Salakhutdinov and Hinton, 2009; Hinton and Salakhutdinov, 2011) of given data, so that related input items have closer hash values (Hinton and Salakhutdinov, 2006). Similar to DBN, DAE contains multiple layers of restricted Boltzmann machines (RBM) which are stacked on top of each other. Each RBM is trained one after another in a greedy way and the overall network is fine-tuned after greedy pre-training is completed. Unlike DBN, DAE does not contain any class layer on top; DAE training is fully unsupervised.

Matrix multiplication is the core operation for training deep networks. Weight for each layer corresponds to a matrix of size $v \times h$, where v is the input layer length and h is the output layer length. For large and sparse inputs, sparse operations can be used to construct output layer of an autoencoder. However, input layer should also be reconstructed from dense output vectors for which sparse operations cannot be used in the original algorithm. This is prohibitively time-consuming for large-dimensional input data. In order to train with very high dimensional and sparse inputs, Dauphin et al. (2011) applied reconstruction sampling for stacked denoising autoencoders. In this work, we apply reconstruction sampling for DBN and DAE, both for pre-training of the first layer autoencoder and during fine-tuning by stochastic gradient descent.

Let us give an example for reconstruction sampling. Let us consider a network with visible layer of size v and first hidden layer of size h. Then, size of the weight matrix W for the first autoencoder becomes $v \times h$. For the forward pass for a minibatch of 3 input vectors, $3 \times v$ matrix X is multiplied with W to generate $3 \times h$ batch output matrix Y. After Y goes some non-linear operations, during the backward pass input $\bar{X}$ is reconstructed from $\hat{Y}$ by $\bar{X} = \hat{Y} \times W^T$ (we ignore bias parameters for the sake of simplicity). Say $X[1,a] = X[1,b] = X[2,b] = X[2,c] = X[2,d] = X[3,a] = X[3,e] = 1$ and others are all 0. Then, for this minibatch, we use only rows a, b, c, d, e of W during training. This means, while reconstructing $\bar{X}$ only $h \times 5$ matrix $W[a,b,c,d,e]^T$ is multiplied with $\hat{Y}$. For each minibatch, only a portion of the weight matrix is trained. After all training batches are processed, training is completed for all rows of W.

Classify(title, descr, category-ids, trained-models)
// *trained-models:* DBN-{title,descr}-*, DAE-{title,descr}, hashed-training-{titles, descrs}
title-word-vec ← **feature-extractor**(title) // *Regex for product codes etc. and Kuromoji Analyzer*
descr-word-vec ← **feature-extractor**(descr)

// *Nearest points (found by kNN over semantic hashes) are used for both steps*
title-semantic-hash ← **compute-DAE**(DAE-title, title-word-vec)
descr-semantic-hash ← **compute-DAE**(DAE-descr, descr-word-vec)
title-nearest-points ← **nearest-points**(title-word-vec, hashed-training-titles)
descr-nearest-points ← **nearest-points**(descr-word-vec, hashed-training-descrs)

// *Classifier outputs for first step*
title-l1-kNN-probs ← **kNN-classify**(title-nearest-points, level1-category-ids)
descr-l1-kNN-probs ← **kNN-classify**(descr-nearest-points, level1-category-ids)
title-l1-DBN-probs ← **compute-DBN**(DBN-title-l1, title-word-vec)
descr-l1-DBN-probs ← **compute-DBN**(DBN-descr-l1, descr-word-vec)
l1-probs ← **average**(title-l1-kNN-probs, descr-l1-kNN-probs, title-l1-DBN-probs, descr-l1-DBN-probs)
N ← **argmax**(l1-probs) // Predicted level-1 category id

// *Step 2*
title-kNN-probs ← **kNN-classify**(title-nearest-points, leaf-category-ids)
descr-kNN-probs ← **kNN-classify**(descr-nearest-points, leaf-category-ids)
title-N-DBN-probs ← **compute-DBN**(DBN-title-N, title-word-vec)
descr-N-DBN-probs ← **compute-DBN**(DBN-title-N, descr-word-vec)
probs ← **average**(title-kNN-probs, descr-kNN-probs, title-N-DBN-probs, descr-N-DBN-probs)
final-prediction-id ← **argmax**(probs)

Figure 1: 2-step classification for a given product title with trained models.

4 Classification Models

Classification is implemented in two steps. In first step, first level categories are predicted. There are 35 first level categories in Rakuten Ichiba. In the second step, leaf categories are predicted.

In each step, two classifiers, DBN and kNN (k-Nearest Neighbors) are used for each of two different data sources (titles and descriptions). Category predictions are made by averaging probability distribution scores of four different classifiers. DBN accepts 0-1 word vectors of high dimensions, and returns class probabilities as output. DAE is used to find semantic hash values for 0-1 word vectors. These semantic hashes are then used for k-nearest-neighbor classification.

For each data source, one DBN model is trained for first level classification and 35 different models are trained for sub-categories under each first level category. In total, we have $2 \times (1 + 35) = 72$ DBN models trained. However, we have only 2 DAE models for semantic hashing of titles and descriptions. In the first step, first level category IDs are used for kNN classification and in the second step, leaf level category IDs are used for kNN classification using the same semantic hashes. Two step classification using trained models is summarized in Figure 1.

kNN classification using high dimensional semantic hash values and hundreds of millions of points (training items) is not practical if all points are traversed during the neighbor search. Therefore, points are first clustered using hierarchical k-means (Böcker et al., 2004). Nearest neighbor search is implemented in the cluster whose center is the closest to the search point.

In the second step for classification, DBN classifiers return results from the same level-1 category tree. On the other hand, kNN classifiers can return any category. It is possible to build different DAE models and different kNN classifiers for each level-1 category, just like we do for DBN classification models.

However, doing so increases the number of models twice and it requires 4 kNN searches to classify a product, instead of 2. Besides decreasing run-times for training and classification, having a global model for kNN helps decreasing the error propagation ratio. If level-1 category is mispredicted in step 1, there is no way DBN classification can find the correct category. But, mixing with global kNN result may correct mispredictions made in step 1.

5 Tokenization and Feature Extraction

We use product titles and descriptions for classification, which are mainly in Japanese. Merchants in Rakuten Ichiba are free to assign their own titles and descriptions for the products they are listing. We assume no dictionary is available for meta-information, like manufacturer name. However, we extract model numbers, counts and sizes from the text. Remaining text is tokenized using Kuromoji[4] Japanese morphological analyzer to extract words.

We first normalize text by converting all Japanese characters to full-width and all non-Japanese characters to lower cases. All HTML tags are cleaned from descriptions. Products' model numbers are predicted using regular expressions, by checking alphabet/number combinations, possibly with spaces and dashes. For example, the words "iPhone 4s" is normalized to "iphone4s". Sizes are estimated by checking whether there exists quantity keywords following numbers. For example, "12.3 cm" is normalized to "12.3cm" and "12cm x 3 cm" is converted to "12cmx3cm". There may be some conflicts with model number extractor and size extractor. In this case, we keep both model numbers predictions and sizes, like "iPhone 4Gb" to be normalized as "iphone4gb 4gb". Japanese have counters for different type of objects. For example, "本" is used to count long and cylindirical objects and "枚" is used to count flat objects. Counters following numbers are not tokenized and kept together, like "3枚".

Japanese words are not split by spaces, so it is not straight-forward to split words. We use Kuromoji in search mode for tokenization of Japanese text and take the base form as word features. Stop words defined in Kuromoji plug-in for Lucene[5] and punctuation marks are excluded from the dictionary.

As a result of above feature extraction process, we have around 26 million words for titles and 47 million words for descriptions. These numbers are much bigger than dictionary sizes of natural languages, because of the nature of e-commerce data. Pre-processing for model number, size and count extraction, as well as misspelled words and failing to properly tokenize Japanese text increase the number of words. It is especially difficult to tokenize Katakana words which are used for writing imported words from other languages and prone to misspelling. We choose a frequency threshold, so that words appearing only a few times in the corpus are not selected as features to be used. Although less-frequent words are more expressive for products, by eliminating them we eliminate most of the noisy information as well. Eliminating less-frequent words also helps making the classifier practical and more robust for classification of new products. Sparse word vectors of the dictionary size after elimination of less-frequent words are accepted as input features for deep network classifiers, which we explain the details in the following sections.

6 Experimental Results

In this section, we discuss details of experiments in terms of prediction matching ratios with merchants' assignments and run-time discussions for the methods and implementations we explained in the paper.

6.1 Dataset and Model Properties

We used Rakuten product dataset, which is available under Rakuten Data Release program[6]. We processed 280 million (active and inactive) products listed by over 40,000 merchants. Products are assigned to 28,338 active categories. There are many products sharing the same title which we remove beforehand. After deduplication by titles, around 40% of products were eliminated, remaining 172 million titles. 90% of those products were randomly selected as being training data, and remaining 10% as test.

[4] http://www.atilika.org/
[5] http://lucene.apache.org/
[6] http://rit.rakuten.co.jp/opendata.html

We extracted features from training data as explained in Section 5. As a result, 26 million words were extracted for titles and 47 million words were extracted for descriptions. Words appearing more than 50 times in the training data were selected as features for training first-step DBN and DAE networks. As a result, the number of word features (dictionary size) was 968,471 for titles and 1,461,625 for descriptions. We built 4-layer step 1 DBN for titles with layer sizes $\{968471, 1000, 2000, 35\}$, and for descriptions $\{1461625, 650, 2000, 35\}$. 35 is the number of level-1 categories. For DAE, last layer size is 64, meaning we built 64-D semantic hash values for each input word vector.

Input layer is still very large for training deep networks, but the word-vectors are very sparse which makes it possible for deep networks to be trained in reasonable time using the selective reconstruction method explained in Section 3, using latest generation GPUs. Average number of words for a title is 12.9 and average number of words for a description is 98.7. Hence, for a input vector batch of size 100, at largest 1290×1000 parameters were trained for titles and at most 9870×650 for descriptions.

DBNs in step 2 for each level-1 category used different feature sets. Dictionary sizes for level-1 categories are much smaller when compared with that of whole corpus. Most frequent 500,000 words were taken as features for second step DBNs. By doing so, if there are more words than 500,000 in a level-1 category, words appearing only once or twice were eliminated. Therefore DBN models for titles and descriptions in step 2 is of size $\{\min(v, 500000), 1800, 2000, n\}$, where v is the dictionary size and n is the number of leaf categories in the corresponding level-1 category.

For kNN classification, after training DAEs (where we selected k to be 10), semantic hash values were calculated for training data. Those vectors then goes into hierarchical k-means clustering, where the set of hash vectors are clustered into 64 in each level until the number of points within a cluster falls below 10,000.

6.2 Hardware and Software Setup

We used a Ubuntu 14.04 Linux server with 4 Nvidia TitanX GPUs for our experiments. Each GPU has 12GB memory and 3072 processing cores. The server has two 12-core Intel CPUs running at 2.4GHz. The system has 96 GB main memory.

Extraction of word features was implemented on CPU, using regular expressions and Kuromoji analyzer. It took around 8 hours to extract features from 280 million titles and descriptions.

DBN and DAE were implemented using CUDA library with C++. Earlier work by Raina et al. (2009) shows that GPU implementation of deep network training can be more than two order of magnitudes faster when compared with single-core CPU implementations. In our implementation, besides the kernels we have written for original operations like selective reconstruction, we exploited CUDAMat (Mnih, 2009), cuBLAS and cuSPARSE libraries (NVIDIA, 2015) for efficient matrix operations. During training iterations, model weights were kept in GPU memory and input word vectors were stored in main memory in sparse format. For even larger models, it is possible to store model weights in main memory and communicate working parameters in each batch with GPUs, and/or stream input features from disk drive. However, these choices considerably slows down GPU training times. During greedy layer-wise pre-training, upper layers were constructed on memory using on-memory sparse input word vectors and trained lower-layer weights. This is a more practical solution than storing each layer output and streaming it for training upper layers, because intermediate layers are not sparse, requiring huge amount of storage for intermediate layers for large number of training samples.

Hierarchical k-means clustering and kNN search using hierarchical k-means tree were also implemented using GPUs, using the guidelines explained in Cevahir and Torii's work (2013). With our settings explained above, it took several days to train deep models and k-means tree using 4 GPUs.

6.3 Prediction Recalls

We compared prediction results with merchants' assignments in our test data. Merchants' assignments in Rakuten Ichiba are quite noisy, having the same products by different merchants distributed through different categories and more than 40% of products are in "Others" leaf categories (such as "Women's Fashion > Tops > Sweaters > Long-sleeved knit > Others"). However, it was not easy to eliminate noise

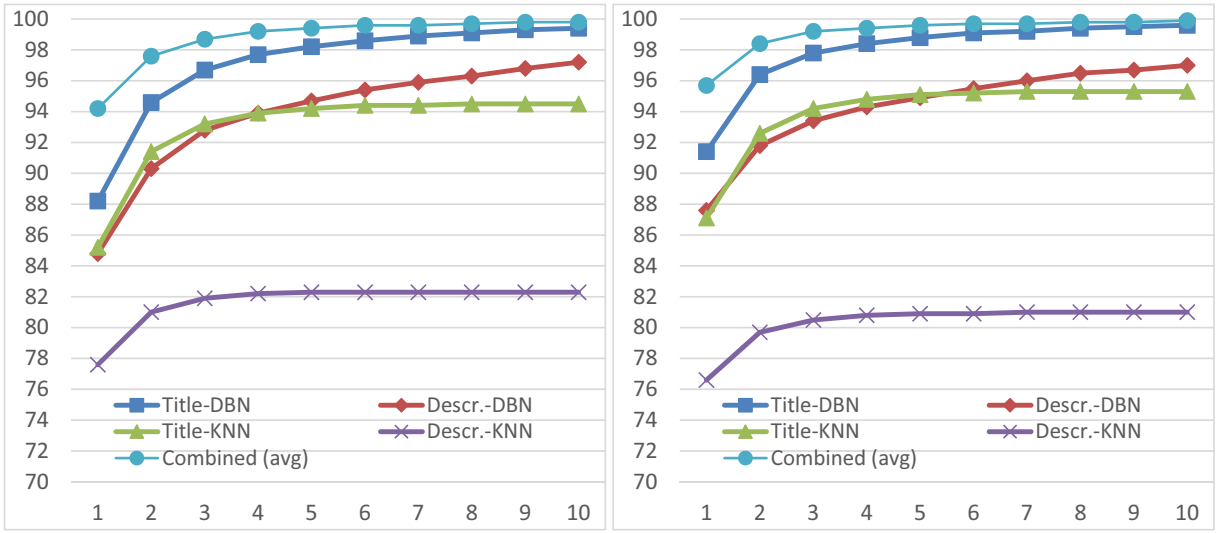


Figure 2: Top level category matching percent recalls (first step matching for 35 level-1 categories). **(a)** All categories including leaf-level categories named "Others", **(b)** excluding leaf-level categories named "Others".

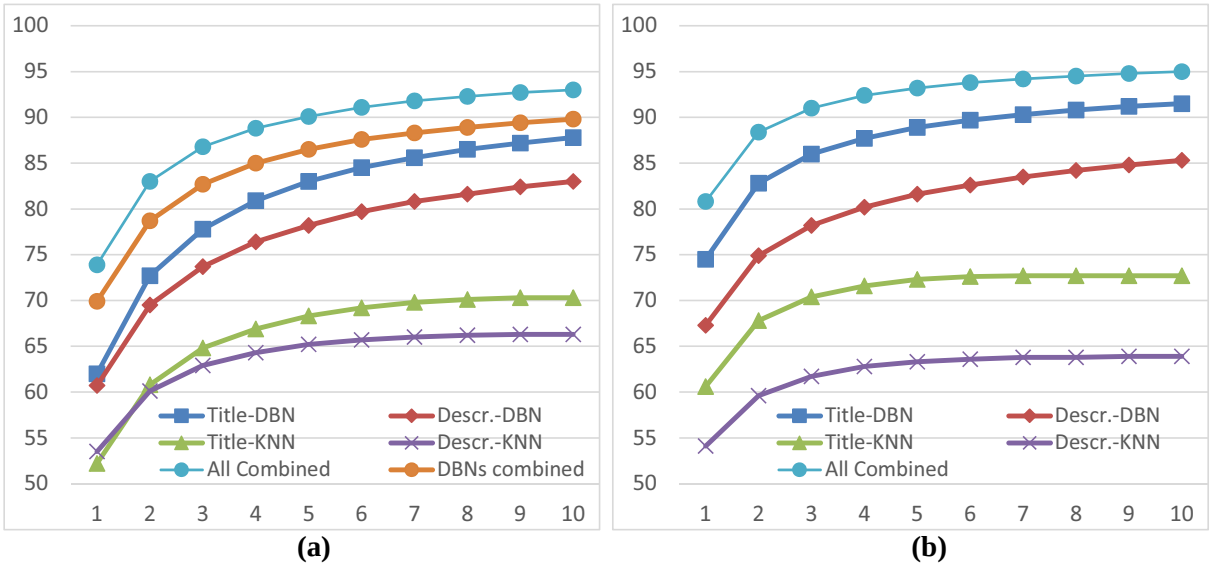


Figure 3: 2-step category matching percent recalls. **(a)** All 28,338 categories including leaf-level categories named "Others", **(b)** excluding leaf-level categories named "Others".

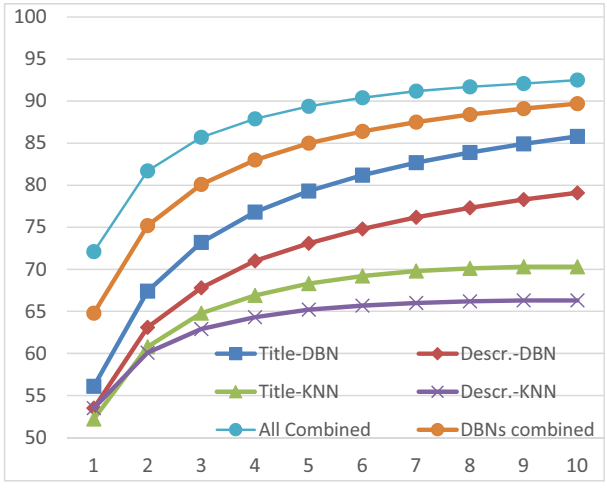


Figure 4: 1-step category matching percent recalls for all 28,338 categories including "Others".

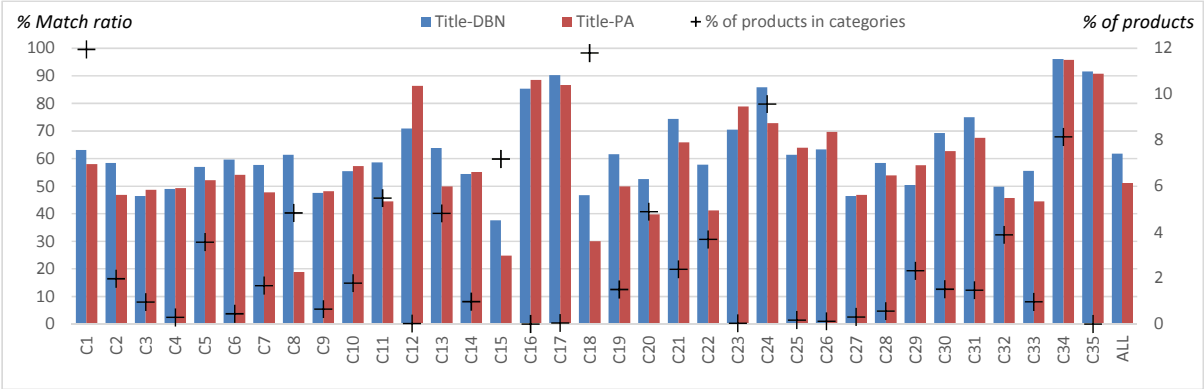


Figure 5: Comparison of category matching results grouped by first-level categories for title-DBN with a Passive-Aggressive algorithm with kernel slicing (Yoshinaga and Kitsuregawa, 2010).

for millions of test data in 28,338 categories by hand, hence we present prediction recalls considering the merchants' assignments as the basis.

Figure 2**a** depicts level-1 categorization matching percent ratios up to top-10 predictions in first step and Figure 3**a** depicts final categorization matching ratios after step 2. We excluded items from "Others" categories in evaluation results depicted in Figure 2**b** and Figure 3**b** as many products in those categories were miscategorized by merchants. Titles usually better define products, so title-based approaches yield better matching ratios than description-based approaches. DBN matching ratios were better than kNN classification, but the overall combination of methods by score averaging yielded much better matching ratios than each individual method.

Both using different data sources and different models affected the increase in matching ratios. The *DBNs combined* line in Figure 3**a** shows the matching ratios when only DBN models for titles and descriptions are combined. First prediction matches by 70% with the merchant assignments, which is better than individual model results, but combining kNN models as well increases the overall matching ratio for the first prediction 4% more.

The effect of making categorization in 2-steps, instead of using big 1-step models, can be observed by comparing results from Figure 3**a** and Figure 4. In Figure 4, categorization results are presented by using direct classification in one step with features used in the first step of 2-step approach. Although 2-step approach suffered from error propagation, number of features used in the second step is much larger, hence the overall matching ratios are better. Matching ratios of individual approaches had large gaps, but the gap reduces to 1.8% if combined models were compared for first predictions.

In order to confirm the effectiveness of deep network training for classification, we compared our results with a passive-aggressive (PA) learning algorithm (Crammer et al., 2006) with kernel slicing, using OPAL tool (Yoshinaga and Kitsuregawa, 2010). Figure 5 depicts the comparison between second step DBN classification and PA classification matching results using title data only for products grouped by level-1 categories. Please note that, we also tried one-step direct classification to 28,338 categories with PA, but it was not able to be trained on single server with 96 GB main memory, because of the memory overflow problem. It can be confirmed from the figure that DBN works better for 23 categories out of 35. Categories that PA works better were usually very small. The overall performance difference was between title-DBN and title-PA is more than 10%.

Although overall performance was worse for the passive-aggressive approach, one may think that it can still be used to increase accuracy by combining with the models which we have explained above. However, the scores given by the algorithm was not suitable for combining by score averaging.

6.4 Sample Results

Although we provide results for matching between our predictions and merchant categories, the actual correct prediction ratios are different. There are four possibilities in case of a mismatch between a prediction and the corresponding merchant: merchant correct / prediction incorrect, merchant incorrect /

	**Product title:** Sweet Mother - Isaac Andrews **Merchant Cat.:** Books, Magazines & Comics > Western Books > Books For Kids **Predicted Cat.:** Books, Magazines & Comics > Western Books > Fiction & Literature
	**Product title:** トヨトミ ［KS-67H］ 電子火式流型石油ストブ KS67H **Merchant Cat.:** Flowers, Garden & DIY/DIY & Tools > Others **Predicted Cat.:** Consumer electronics > Seasonal home appliances > Heating appliance > Oilstove > 14+ tatami (wooden) , 19+ tatami (rebar)
	**Product title:** レンタル【RG87】はかまフルセット/大学生/小学生/高校生/中学生 **Merchant Cat.:** Women's Fashion > Japanese style > Kimono > Hakama **Predicted Cat.:** Women's Fashion > Kimono > Rental
	**Product title:** 夜咄用具 スキヤろうそく 大 **Merchant Cat.:** Kitchenware, tableware & cookware > Japanese tableware > Tea utensils > Other **Predicted Cat.:** Kitchenware, tableware & cookware > Japanese tableware > Small bowl

Table 1: Sample results. Omitted description information. Images are for reference, not used for classification.

prediction correct, both correct, both incorrect. See Table 1 for sample results for those 4 cases.

In case merchant is correct and prediction is incorrect, confidence scores for predictions are lower. In the first example in Table 1, although it is predicted as a Western book, the type of the book is mispredicted. Second example is a typical mismatch case where merchant is incorrect. Products in "Others" categories, which account for 40% of all products, have high probability of being misplaced. It is possible to predict detailed correct categories for such products. In the third example, both merchant and our prediction can be considered as correct, as the product is a hakama style kimono, but it is rental. The candle in the last example is used for tea ceramonies, but it is not a tea utensil. It is predicted as small bowl because of the explanations about candle stand in the product description, but the prediction confidence score was quite low, 0.045.

7 Conclusion and Future Work

In this work, we have presented a categorization system for large scale e-commerce data using product titles and descriptions. We have trained our system with a dataset having hundreds of millions of products and tens of thousands of categories. Our tests confirm high matching ratios of predictions with merchant-assigned categories. We used different data sources and different algorithms for classification, where the final scores are calculated by averaging of scores of different models' results. Exploration and evaluation of different combination techniques of results are left as future work.

Although we have utilized all textual content about products in this work, we ignored image content as it takes a lot of time to process images. Our initial evaluations utilizing image data, which we have not discussed in this work, suggest that it is possible to increase the system performance by several percent. We leave full evaluation of the prediction system with image data as a future work.

References

A. Böcker, S. Derksen, E. Schmidt, and G. Schneider, 2004. *Hierarchical K-means Clustering*. Modlab, Universit at Frankfurt AM MAIN.

Ali Cevahir and Junji Torii. 2013. High performance online image search with gpus on large image databases. *International Journal of Multimedia Data Engineering and Management (IJMDEM)*, 4(3):24–41.

Jianfu Chen and David Warren. 2013. Cost-sensitive learning for large-scale hierarchical classification of commercial products. In *Proc. of 22nd Conference on Information and Knowledge Management (CIKM 2012)*, pages 1351–1360.

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

Yann N. Dauphin, Xavier Glorot, and Yoshua Bengio. 2011. Large-scale learning of embeddings with reconstruction sampling. In *Proc. of 28th International Conference on Machine Learning*, pages 945–952.

Vivek Gupta, Harish Karnick, Ashendra Bansal, and Pradhuman Jhala. 2016. Product classification in e-commerce using distributional semantics. *arXiv preprint arXiv:1606.06083*.

Jung-Woo Ha, Hyuna Pyo, and Jeonghee Kim. 2016. Large-scale item categorization in e-commerce using multiple recurrent neural networks. In *Proc. of 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Viet Ha-Thuc and Jean-Michel Renders. 2011. Large-scale hierarchical text classification without labelled data. In *Proc. of the fourth ACM international conference on Web search and data mining (WSDM 2011)*, pages 685–694.

Geoffrey E. Hinton and Ruslan R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Geoffrey Hinton and Ruslan Salakhutdinov. 2011. Discovering binary codes for documents by learning deep generative models. *Topics in Cognitive Science*, 3(1):74–91.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computing*, 18(7):1527–1554.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. of Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1746–1751.

Aris Kosmopoulos, Georgios Paliouras, and Ion Androutsopoulos. 2015. Probabilistic cascading for large scale hierarchical classification. *arXiv preprint arXiv:1505.02251*.

Zornitsa Kozareva. 2015. Everyone likes shopping! multi-class product categorization for e-commerce. In *Proc. of Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 1329–1333.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2267–2273.

Volodymyr Mnih. 2009. Cudamat: a cuda-based matrix class for python. Technical report, Tech. Rep. UTML TR 2009–004, Department of Computer Science, University of Toronto.

NVIDIA. 2015. Nvidia cusparse and cublas libraries. `http://docs.nvidia.com/cuda/`.

Xipeng Qiu, Xuanjing Huang, Zhao Liu, and Jinlong Zhou. 2011. Hierarchical text classification with latent concepts. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*, pages 598–602.

Rajat Raina, Anand Madhavan, and Andrew Y. Ng. 2009. Large-scale deep unsupervised learning using graphics processors. In *Proc. of the 26th Annual International Conference on Machine Learning (ICML 2009)*, pages 873–880.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978.

Dan Shen, Jean David Ruvini, Manas Somaiya, and Neel Sundaresan. 2011. Item categorization in the e-commerce domain. In *In Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM 2011)*, pages 1921–1924.

Dan Shen, Jean-David Ruvini, and Badrul Sarwar. 2012. Large-scale item categorization for e-commerce. In *In Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM 2012)*, pages 595–604.

Peng Wang, Jiaming Xu, Bo Xu, Cheng-Lin Liu, Heng Zhang, Fangyuan Wang, and Hongwei Hao. 2015. Semantic clustering and convolutional neural network for short text categorization. In *Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing 2015*, pages 26–31.

Gui-Rong Xue, Dikan Xing, Qiang Yang, and Yong Yu. 2008. Deep classification in large-scale text hierarchies. In *In Proc. of the 31th Annual International Conference on Research and Development in Information Retrieval (SIGIR 2008)*, pages 619–626.

Naoki Yoshinaga and Masaru Kitsuregawa. 2010. Kernel slicing: Scalable online training with conjunctive features. In *Proc. of the 23th International Conference on Computational Linguistics (COLING 2010)*, pages 1245–1253.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *In Proc. of Advances in Neural Information Processing System 28 (NIPS)*, pages 649–657.

Product Classification in E-Commerce using Distributional Semantics

Vivek Gupta , Harish Karnick
Indian Institute of Technology , Kanpur
{vgupta,hk}@cse.iitk.ac.in
Ashendra Bansal , Pradhuman Jhala
Flipkart Internet Pvt. Ltd., Bangalore
{ashendra.bansal,pradhuman.jhala}@flipkart.com

Abstract

Product classification is the task of automatically predicting a taxonomy path for a product in a predefined taxonomy hierarchy given a textual product description or title. For efficient product classification we require a suitable representation for a document (the textual description of a product) feature vector and efficient and fast algorithms for prediction. To address the above challenges, we propose a new distributional semantics representation for document vector formation. We also develop a new two-level ensemble approach utilizing (with respect to the taxonomy tree) path-wise, node-wise and depth-wise classifiers to reduce error in the final product classification task. Our experiments show the effectiveness of the distributional representation and the ensemble approach on data sets from a leading e-commerce platform and achieve improved results on various evaluation metrics compared to earlier approaches.

1 Introduction

Existing e-commerce platforms have evolved into large B2C and/or C2C marketplaces having large inventories with millions of products. Products in ecommerce are generally organized into a hierarchical taxonomy of multilevel hierarchical categories. Product classification is an important task in catalog formation and plays a vital role in customer oriented services like search and recommendation and seller oriented services like seller utilities on a seller platform. Product classification is a hierarchical classification problem and presents the following challenges: a) a large number of categories have data that is extremely sparse with a skewed long tailed distribution, b) a hierarchical taxonomy imposes constraints on activation of labels. If a child label is active then it is necessary for a parent label to be active, c) for practical use the prediction should happen in real time - ideally within few milli-seconds.

Traditionally, documents have been represented as a weighted bag-of-words (BoW) or tf-idf feature vector, which contains weighted information about the presence or absence of words in a document by using a fixed length vector. Words that define the semantic content of a document are expected to be given higher weight. While tf-idf and BoW representations perform well for simple multi-class classification tasks, they generally do not do as well for more complex tasks because the BoW representation ignores word ordering and polysemy, is extremely sparse and high dimensional and does not encode word meaning. Such disadvantages have motivated continuous, low-dimensional, non-sparse distributional representations. A word is encoded as a vector in a low dimension vector space typically $\mathcal{R}^{100}$ to $\mathcal{R}^{300}$. The vector encodes local context and therefore is sensitive to local word order and captures word meaning to some extent. It relies on the 'Distributional Hypothesis'(Harris, 1954) i.e. *Similar words occur in similar contexts*. Similarity between two words can be calculated via cosine distance between their vector representations.

Le and Mikolov (Le and Mikolov, 2014) proposed paragraph vectors, which use global context together with local context to represent documents. But paragraph vectors suffer from the following problems: a) current techniques embed paragraph vectors in the same space (dimension) as word vectors

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 536–546, Osaka, Japan, December 11-17 2016.

although a paragraph can consist of words belonging to multiple topics (senses), b) current techniques also ignore the importance and distinctiveness of words across documents. They assume all words contribute equally both quantitatively (weight) and qualitatively (meaning).

In this paper we describe a new compositional technique for formation of document vectors from semantically enriched word vectors to address the above problems. Further, to capture importance, weight and distinctiveness of words across documents we use a graded weights approach, inspired by the work of Mukerjee et al. (Pranjal Singh, 2015), for our compositional model. We also propose a new two-level approach for product classification which uses an ensemble of classifiers for label paths, node labels and depth-wise labels (with respect to the taxonomy) to decrease classification error. Our new ensemble technique efficiently exploits the catalog hierarchy and achieves improved results in top K taxonomy path prediction. We show the effectiveness of the new representation and classification approach for product classification of two e-commerce data-sets containing book and non-book descriptions.

2 Related Work

2.1 Distributional Semantic Word Representation

The distributional word embedding method was first introduced by Bengio et al. as the Neural Probabilistic Language Model (Bengio et al., 2003). Later, Mikolov et al. (Mikolov et al., 2013) proposed a simple log-linear model which considerably reduced training time - Word2Vec Continuous Bag-of-Words (CBoW) model and Skip-Gram with Negative Sampling (SGNS) model. Figure 1 shows the architecture for CBoW (Left) and Skip-Gram (Right).

Later Glove (Jeffrey Pennington, 2014) a log-bilinear model with a weighted least-squares objective was proposed which uses the statistical ratio of global word-word co-occurrences in the corpus for training word vectors. The word vectors learned using the skip-gram model are known to encode many linear linguistic regularities and patterns (Levy and Goldberg, 2014b).

While the above methods look very different they implicitly factorize a shifted positive point-wise mutual information matrix (PPMI) with tuned hyper parameters as shown by Levy and Goldberg (Levy and Goldberg, 2014c). Some methods also describe use of non-linear dependency based context (Levy and Goldberg, 2014a). Some variants incorporate ordering information in context words to capture syntactic information by replacing summation of context word vectors with concatenation during training (Wang Ling, 2015) of CBoW and SGNS models.

2.2 Distributional Paragraph Representation

Most models for learning distributed representations for long text such as phrases, sentences or documents that try to capture semantic composition do not go beyond simple weighted average of word vectors. This approach is analogous to a bag-of-words approach and neglects word order while representing documents. Socher et al. (Socher et al., 2013) propose a recursive tensor neural network where the dependency parse-tree of the sentence is used to compose word vectors in a bottom-up approach to represent sentences or phrases. This approach considers syntactic dependencies but cannot go beyond sentences as it depends on parsing.

Mikolov proposed a distributional paragraph vector framework called paragraph vectors which are trained in a manner similar to word vectors. He proposed two types of models called *Distributed Memory Model Paragraph Vectors (PV-DM)* (Le and Mikolov, 2014) and *Distributed BoWs paragraph vectors (PV-DBoW)* (Le and Mikolov, 2014). In PV-DM the model is trained to predict the center word using context words in a small window and the paragraph vector (Le and Mikolov, 2014). Here context words to be predicted are represented by w_{t-k},....,w_{t+k} and the document vector is represented by D_i. In PV-DBoW the paragraph vector is trained to predict context words directly. Figure 2 shows the network architecture for PV-DM(Left) and PV-DBoW(Right).

The paragraph vector presumably represents the global semantic meaning of the paragraph and also incorporates properties of word vectors i.e. meanings of the words used. A paragraph vector exhibits close resemblance to an n-gram model with a large n. This property is crucial because the n-gram model preserves a lot of information in a sentence (and the paragraph) and is sensitive to word order.

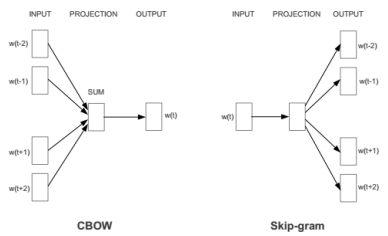


Figure 1

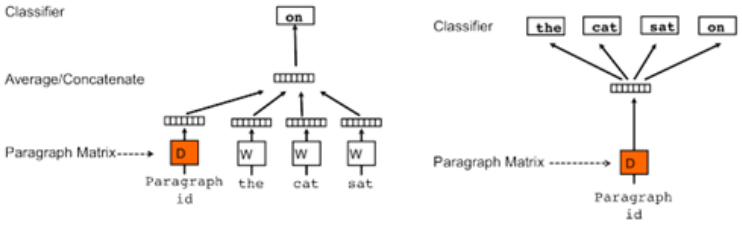


Figure 2

This model mostly performs better than the BoW models which usually create a very high-dimensional representation leading to poorer generalization.

2.3 Problem with Paragraph Vectors

Paragraph vectors obtained from PV-DM and PV-DBoW are shared across context words generated from the same paragraph but not across paragraphs. On the other hand a word is shared across paragraphs. Paragraph vectors are also represented in the same space (dimension) as word vectors though a paragraph can contain words belonging to multiple topics (senses). The formulation for paragraph vectors ignores the importance and distinctiveness of a word across documents i.e. assumes all words contribute equally both quantitatively (weight wise) and qualitatively (meaning wise). Quantitatively, only binary weights i.e. 0 weight for stop-words and non-zero weight for others are used. Intuitively, one would expect the paragraph vector to be embedded in a larger and enriched space.

2.4 Hierarchical Product Categorization

Most methods for hierarchical classification follow a *gates-and-experts* method which have a two level classifier. The high-level classifier serves as a "gate" to a lower level classifier called the "expert" (Shen et al., 2011). The basic idea is to decompose the problem into two models, the first model is simple and does coarse-grained classification while the second model is more complex and does more fine-grained classification. The coarse-grained classification deals with a huge number of examples while the fine-grained distinction is learned within a subtree under every top level category with better feature generation and classification algorithms and deals with fewer categories. Later, Xue et al. (Xue et al., 2008) suggested an interesting two stage strategy called *"deep classification"*. The first stage (search) groups documents in the training set that are similar to a given document. In the second stage (classification) a classifier is trained on these classes and used to classify the document. In this approach a specific classifier is trained for each document making the algorithm computationally inefficient.

For large scale classification Bengio et al. (Bengio et al., 2010) use the confusion matrix for estimating class similarity instead of clustering data samples. Two classes are assumed to be similar if they are often confused by a classifier. Spectral clustering, where the edges of the similarity graph are weighted by class confusion probabilities, is used to group similar classes together.

Shen and Ruvini (Shen et al., 2012) (Shen et al., 2011) extend the previous approach by using a mixture of simple and complex classifiers for separating confused classes rather then spectral clustering methods which has faster training times. They approximate the similarity of two classes by the probability that the classifier incorrectly predicts one of the categories when the correct label is the other category. Graph algorithms are used to generate connected groups from estimated confusion probabilities. They represent the relationship among classes using an undirected graph $G = (V, E)$, where the set of vertices V is the set of all classes and E is the set of all edges. Two vertices's are connected by an edge if the confusion probability $Conf(c_1, c_2)$ is greater than a given threshold α (Shen et al., 2012).

Other simple approaches like flat classification and top down classification are intractable due to the large number of classes and give poor results due to error propagation as described in (Shen et al., 2012).

3 Graded Weighted Bag of Word Vectors

We propose a new method to form a composite document vector using word vectors i.e. distributional meaning and tf-idf and call it a Graded Weighted Bag of Words Vector (gwBoWV). gwBoWV is inspired from the computer vision literature where we use a Bag of Visual words to form feature vectors. gwBoWV is calculated as follows:

1. Each document is represented in a lower dimensional space $D = K * d + K$, where K represents number of semantic clusters and d is the dimension of the word-vectors.

2. Each document is also concatenated with inverse cluster frequency(icf) values which is calculated using idf values of words present in the document.

Idf values from the training corpus are directly used for the test corpus for weighting. Word vectors are first separated into a pre-defined number of semantic clusters using a suitable clustering algorithm (e.g. k-means). For each document we add the word-vectors of each word in the document belonging to a cluster to form a cluster vector. We finally concatenate the cluster vector and the icf for each of the K clusters to obtain the document vector. Algorithm 1 describes this in more detail.

Algorithm 1: Graded Weighted Bag of Word Vectors

Data: Documents D_n, n = 1 . . . N
Result: Document vectors $gw\vec{BoWV}_{D_n}$, n = 1 . . . N

Train SGNS model to obtain word vector representation (wv_n) using all document D_n, $n = 1..N$;
Calculate idf values for all words: $idf(w_j)$, $j = 1..|V|$; `/* |V| is vocabulary size */`
Use K-means algorithm for clustering all words in V using their word-vectors into K clusters;
for $i \in (1..N)$ **do**
    Initialize cluster vector $c\vec{v}_k = \vec{0}$, $k = 1..K$;
    Initialize cluster frequency $icf_k = 0$, $k = 1..K$;
    while *not at end of document* D_i **do**
        read current word w_j and obtain wordvec $w\vec{v}_j$;
        obtain cluster index $k = idx(w\vec{v}_j)$ for wordvec $w\vec{v}_j$;
        update cluster vector $c\vec{v}_k += w\vec{v}_j$;
        update cluster frequency $icf_k += idf(w_j)$;
    end
    obtain $gw\vec{BoWV}_{D_i} = \bigoplus_{k=1}^{K} c\vec{v}_k \oplus icf_k$; `/* ⊕ is concatenation */`
end

Since semantically different vectors are in separate clusters we avoid averaging of semantically different words during Bag of Words Vector formation. Incorporation of idf values captures the weight of each cluster vector which tries to model the importance and distinctiveness of words across documents.

4 Ensemble of Multitype Predictors

We propose a two level ensemble technique to combine multiple classifiers predicting product paths, node labels and depth-wise labels respectively. We construct an ensemble of multi-type features for categorization inspired by the recent work of Zornitsa et. al. from Yahoo Labs (Kozareva, 2015). Below are the details of each classifier used at level one:

- *Path-Wise Prediction Classifier*: We take each possible *path* in the catalog taxonomy tree, from leaf node to root node, as a possible class label and train a classifier (PP) using these labels.

- *Node-Wise Prediction Classifier*: We take each possible *node* in the catalog taxonomy tree as a possible prediction class and train a classifier (NP) using these class labels.

- *Depth-Wise Node Prediction Classifiers*: We train multiple classifiers (DNP_i) one for each depth level of the taxonomy tree. Each possible *node* in the catalog taxonomy tree at that depth is a possible class label. We take all data samples which have a potential node at depth k for training. For samples of data point whose path ended before depth k we use special None Label and use 10% of such data points for training.

We use the output probabilities of these classifiers at level one (PP, NP, DNP_i) as a feature vector and train a classifier (level two) after some dimensionality reduction.

The increase in training time can be reduced by training all level one classifiers in parallel. The algorithm for training the ensemble is described in Algorithm 2. The testing algorithm is similar to training Algorithm 3

Algorithm 2: Training Two Level Boosting Approach

Data: Catalog Taxonomy Tree (T) of depth M and training data D = (d, p_d) where d is the product description and p_d is the taxonomy path label.

Result: Set of level one Classifiers C = $\{PP, NP, DNP_1, \ldots, DNP_M\}$ and level two classifier FPP.

1 Obtain $gw\vec{BoW}V_d$ features for each product description d ;

2 Train Path-Wise Prediction Classifier (PP) with possible classes as product taxonomy paths (p_d);

3 Train Node-Wise Prediction Classifier (NP) with possible classes as nodes in taxonomy path i.e. (n_d). Here each description will have multiple node labels.

4 **for** $m \in (1 \ldots M)$ **do**

5 Train Depth-Wise Node Classifier for depth m (DNP_m) with labels as nodes at depth m i.e. (n_m)

6 **end**

7 Obtain output probabilities $\vec{P}_X$ over all classes for each level one classifier X i.e. $\vec{P}_{PP}$, $\vec{P}_{NP}$ and $\vec{P}_{DNP_m}$, $m = 1..M$.;

8 Obtain feature vector $\vec{FV}_d$ for each description as:

$$\vec{FV}_d = gw\vec{BoW}V_d \oplus \vec{P}_{PP} \oplus \vec{P}_{NP} \bigoplus_{m=1}^{M} \vec{P}_{DNP_m} \quad (1)$$

/* $\bigoplus$ is the concatenation operation */

9 Reduce feature dimension ($R\vec{FV}_d$) using suitable supervised feature selection technique based on mutual information criteria;

10 Train Final Path-Wise Prediction Classifier ($F\vec{PP}_d$) using RFV_d as feature vector and possible class labels as product taxonomy paths (p_d)

5 Dataset

We use seller product descriptions and title samples from a leading e-commerce site for experimentation[1]. The data set had two product taxonomies:*non-book* and *book*. Non-book data is more discriminative with average description + title length of around 10 to 15 words, whereas book descriptions have an average length greater than 200 words. To give more importance to the title compared to the description, we repeated words in title three times (i.e. weighting trice). The distribution of items over leaf categories (verticals) exhibits high skewness and heavy tailed nature and suffers from sparseness as shown in Figure 3. We use random forest and k-nearest neighbors as base classifiers as they are less affected by data skewness.

[1]This data is proprietary to the e-commerce Company. Major part of the work was done when the first author was a research ex-tern at the E-Commerce Company Flipkart

Algorithm 3: Testing Two Level Boosting Approach

Data: Catalog Taxonomy Tree (T) of depth M and testing data D = (d,p_d) where d is product description p_d is taxonomy of paths. Set of level one Classifiers C = $\{PP$,NP,$DNP_1 \ldots DNP_M\}$ and final level two classifier FPP

Result: top 6 prediction path P_d for training description d

1 Obtain $gw\vec{BoW}V_d$ features for each product description d in test data;

2 Get Prediction Probabilities from all level one classifiers to obtain level two feature vector ($\vec{FV}_d$) using Equation 1;

3 Obtain ($\vec{RFV}_d$) reduced feature vector;

4 Output top m paths from final prediction using output probabilities from level two classifier FFP for description d.

Level	#Categories	%Data Samples
1	21	34.9%
2	278	22.64%
3	1163	25.7%
4	970	12.9%
5	425	3.85%
6	18	0.10%

Table 1: Percentage of Book Data ending at each depth level of the book taxonomy hierarchy which had a maximum depth of 6.

We have removed data samples with multiple paths to simplify the problem to single path prediction. Overall, we have 0.16 million training and 0.11 million testing samples for book data and 0.5 million training and 0.25 million testing samples for non-book data. Since the taxonomy evolved over time all category nodes are not semantically mutually exclusive. Some ambiguous leaf categories are even meta categories. We handle this by giving a unique id to every node in the category tree of book-data. Furthermore, there are also category paths with different categories at the top and similar categories at the leaf nodes i.e. reduplication of the same path with synonymous labels. Below are examples of such synonymous path labels.

1. *Household → Lights and Lamps → Bulbs → LED Bulbs*

2. *Home Decor → Lights and Lamps → Bulbs → LED Bulbs*

Another Example

1. *Home Furnishing → Living → Cushion Pillow Covers*

2. *Home Furnishing → Bed → Pillows and Pillow Covers → Pillow Covers*

The quality of the descriptions and titles also varies a lot. There are titles and descriptions that do not contain enough information to decide an unique appropriate category. There were labels like *Others* and *General* at various depths in the taxonomy tree which carry no specific semantic meaning. Also, descriptions with the special label 'wrong procurement' are removed manually for consistency.

6 Results

The classification system is evaluated using the usual precision metric defined as fraction of products from test data for which the classifier predicts correct taxonomy paths. Since there are multiple similar paths in the data set predicting a single path is not appropriate. One solution is to predict more than one path or better a ranked list of 3 to 6 paths with predicted label coverage matching labels in the true path.

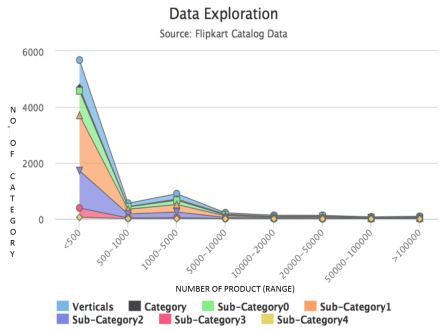


Figure: 3

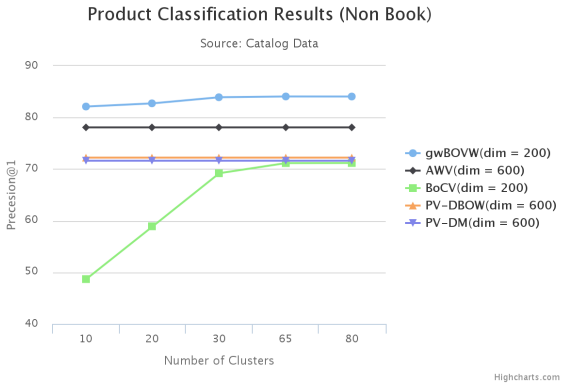


Figure: 4

The ranking is obtained using the confidence score of the predictor. We also calculate the confidence score of the correct prediction path by using the k (3 to 6) confidence scores of the individual predicted paths. For the purpose of measuring accuracy when more than one path is predicted, the classifier result is counted as correct when the correct class (i.e. path assigned by seller) is one of the returned class (paths). Thus we calculated Top 1, Top 3 and Top 6 prediction accuracy when 1, 3 and 6 paths are predicted respectively.

6.1 Non-Book Data Result

We also compare our results with document vectors formed by averaging word-vectors of words in the document i.e. Average Word Vectors (AWV), Distributed Bag of Words version of Paragraph Vector by Mikolov (PV-DBoW), Frequency Histogram of word distribution in Word-Clusters i.e. Bag of Cluster Vector (BoCV). We keep the classifier (random forest with 20 trees) common for all document vector representations. We compare performance with respect to number of clusters, word-vector dimension, document vector dimension and vocabulary dimension (tf-idf) for various models.

Figure 4 shows results for a random forest (20 trees) on various classifiers trained by various methods on 0.2 million training and 0.2 million testing samples with 3589 classes. It compares our approach gwBoWV with PV-DBoW and PV-DM models with varying word vector dimension and number of clusters. The dimension of word vector for gwBoVW and BoCV is 200. Note AWV , PV-DM and PV-DBoW are independent of cluster number and have dimension 600. Clearly gwBoWV performs much better than other methods especially PV-DBoW and PV-DM. Table 2 Shows the effect of varying cluster numbers on accuracy for Non Book Data for 0.2 million training and testing using 200 dimension word vector for Top 1 prediction using gwBoWV.

# Cluster	Precision@1
10	81.35%
20	82.29%
50	83.66%
65	83.85%
80	83.91%
100	84.40%

Table 2: *Result of classification on varying Cluster Numbers for fixed word vector size 200 for Non Book Data for Precision@1 on #Train Sample = 0.2 million , #Test Sample = 0.2 million*

We use the notation given below to define our evaluation metrics for Top K path prediction :

- τ^* represents the true path for a product description.
- τ_i represents the i^{th} predicted path by our algorithm , where $i \in \{1, 2 \dots K\}$.

#Clus, #Dim	%PP	%CP	%LR	%LC
40, 50	82.07	96.43	98.27	34.50
40, 100	**83.18**	96.67	98.39	**34.91**
100, 50	82.05	96.40	98.26	34.41
100,100	83.13	**96.75**	**98.42**	34.88

Table 3: *Result for top 6 paths predicted for multiple Bag of Word Vectors with varying dimension and number of clusters with weighting on Non-Book Data with #Training Samples = 0.50 million, #Test Samples = 0.35 million.*

#Dim	%PP	%CP	%LR	%LC
2000	81.10	94.04	96.85	35.37
4000	**82.74**	**94.78**	**97.33**	**35.61**

Table 4: *Result of top 6 paths prediction for tf-idf with varying dimension on Non Book Data #Training Samples = 0.50 million, #Test Samples = 0.35 million.*

- $\mathbb{T}^*$ represent the nodes in true path τ^*.
- $\mathbb{T}^i$ represents the nodes in i^{th} predicted path τ_i, where $i \in \{1, 2 \dots K\}$.
- p(τ^*) represents the probability predicted by our algorithm for the true path τ^*. p(τ^*) = 0 if $\tau^* \notin \{\tau_1, \tau_2 \dots \tau_K\}$
- p(τ_i) represents the probability of i^{th} predicted path by the algorithm, here $i \in \{1, 2 \dots K\}$.

We use four evaluation metrics to measure performance for the top k predictions as described below:

1. Prob Precision @ K : $PP@K$ = p(τ^*) / (p(τ_1) + p(τ_2) + . . . + p(τ_K)).
2. Count Precision @ K : $CP@k$ = 1 if $\tau^* \in \{\tau_1, \tau_2 \dots \tau_K\}$ else $CP@K$ = 0.
3. Label Recall @ K : $LR@k$ = $\|\mathbb{T}^* \cap (\cup_1^K \mathbb{T}^i)\| / \|\mathbb{T}^*\|$. Here $\|$S$\|$ represent number of elements in set S.
4. Label Correlation @ K : $LC@k$ = $\| \cap_1^K \mathbb{T}^i \| / \| \cup_1^K \mathbb{T}^i \|$. Here $\|$S$\|$ represent number of elements in set S.

Table 3 shows the results on all evaluation metrics with varying word-vec dimension and clusters on Non Book Data. Table 4 shows results of top 6 paths prediction for tf-idf baseline with varying dimension for Non Book Data.

6.2 Book Data Result

Book data is harder to classify as it has larger text ($>$ 200 words) with more common words. There are more cases of improper paths and labels in the taxonomy and hence we had to do a lot of pre-processing. Around 51% of the books did not have labels at all and 15% books were given extremely ambiguous labels like 'general' and 'others'. To maintain consistency we prune the above 66% data samples and work with the remaining 44% i.e. 0.37 million samples.

To handle improper labels and ambiguity in the taxonomy we use multiple classifiers one predicting path (or leaf) label, another predicting node labels and multiple classifiers, one at each depth level of the taxonomy tree, that predict node labels at that level. In depth-wise node classification we also introduce the 'none' label to denote missing labels at a particular level i.e. for paths that end at earlier levels. However we only take a random strata sample for this 'none' label.

Ensemble	Dim_n	Feature	%PP	%CP	%LR	%LC
No	4100	gwBoWV	39.86	74.17	86.37	22.19
No	8080	gwBoWV	41.08	74.83	86.60	22.19
Yes	8000	gwBoWV	45.64	**77.26**	**88.86**	24.57
Yes	6000	gwBoWV	**46.68**	75.74	87.67	**25.08**

Table 5: *Results from various approaches for Top 6 predictions for Book Data*

Accuracy	Precision	Recall	F-score
81.6%	81.1%	81.1%	80.9%

Table 6: *Preliminary Result of gwBoWV with 60 clusters on word-vector dimension of 200 (min word count 20, context window 10), 83% features selection with Anova reduction, C is 2.1 with Linear SVM (one vs rest) on 20newsGroup using weighted averaging for Precision, F-Score and Recall.*

6.3 Ensemble Classification

We use the ensemble of multi-type predictors as described in Section 4 for final classification on Book Data. Results on Non-Book data was already very good and as discussed earlier books were harder to classify. For dimensionality reduction we use feature selection methods based on mutual information criteria (ANOVA F-value i.e. analysis of variance). We obtain improved results for all four evaluation metrics with the new ensemble technique as shown in Table 5 for Book Data for Top 6 Prediction.

6.4 Real Examples from Book Data

Description : *harpercollins continues with its commitment to reissue maurice sendaks most beloved works in hardcover by making available again this 1964 reprinting of an original fairytale by frank r stockton as illustrated by the incomparable maurice sendak in the ancient country of orn there lived an old man who was called the beeman because his whole time was spent in the company of bees one day a junior sorcerer stopped at the hut of the beeman the junior sorcerer told the beeman that he has been transformed if you will find out what you have been transformed from i will see that you are made all right again said the sorcerer could it have been a giant or a powerful prince or some gorgeous being whom the magicians or the fairies wish to punish the beeman sets out to discover his original form. the beeman of orn. the beeman of orn. the beeman of orn.*
Actual Class : books-tree → children → knowledge and learning → animals books → reptiles and amphibians
Predictions, **Probability Score**
books-tree → children → knowledge and learning → animals books → reptiles and amphibians , 0.28
books-tree → children → fun and humor, 0.72

Description : *behavioral economist and new york times bestselling author of predictably irrational dan ariely returns to offer a much needed take on the irrational decisions that influence our dating lives our workplace experiences and our general behaviour up close and personal in the upside of irrationality behavioral economist dan ariely will explore the many ways in which our behaviour often leads us astray in terms of our romantic relationships our experiences in the workplace and our temptations to cheat blending everyday experience with groundbreaking research ariely explains how expectations emotions social norms and other invisible seemingly illogical forces skew our reasoning abilities among the topics dan explores are what we think will make us happy and what really makes us happy why learning more about people make us like them less how we fall in love with our ideas what motivates us to cheat dan will emphasize the important role that irrationality plays in our daytoday decision making not just in our financial marketplace but in the most hidden aspects of our livesabout the author an ariely is the new york times bestselling author of predictably irrational over the years he has won numerous scientific awards and his work has been featured in leading scholarly journals in psychology economics neuroscience and in a variety of popular media outlets including the new york times the wall street journal the washington*

post the new yorker scientific american and science. the upside of irrationality. the upside of irrationality. the upside of irrationality. the upside of irrationality
Actual Class : books-tree → business, investing and management → business → economics
Predictions, **Probability Score**
books-tree → business, investing and management → business → economics 0.15
books-tree → philosophy → logic, 0.175
books-tree → self-help → personal growth, 0.21
books-tree → academic texts → mathematics, 0.465

6.5 Quality of WordVec Clusters

Below are examples of words contained in some clusters formed by clustering of word vectors and their possible cluster topic meaning for book data.

1. Cluster #0 basically talks about crime and punishment related terms like *accused, arrest, assault, attempted, beaten, attorney,brutal,confessions, convicted cops, corrupt, custody, dealer, gang, investigative, gangster, guns, hated, jails, judge, mob, undercover, trail, police, prison, lawyer, torture, witness etc*

2. Cluster #10 talks about scientific experiments and related terms like *yield, valid, variance, alternatives, analyses, calculating, comparing, assumptions, criteria, determining, descriptive, evaluation, formulation, experiments, measures model, parameters, inference, hypothesis etc*

Similarly, Cluster #13 is talking about dating and marriage, Cluster #11 about tools and tutorials and Cluster #15 about persons. Similarity of words within a cluster leads to an efficient distributional semantic representation of word vectors.

7 Conclusions

We presented a novel compositional technique using embedded word vectors to form appropriate document vectors. Further, to capture importance, weight and distinctiveness of words across documents we used a graded weighting approach. Our document vectors are embedded in a vector space that is different from and has higher dimension than the word embedding vector space. This higher dimensional document vector space tries to encode the intuition that a document has more topics than a word.

We also developed a new technique which uses an ensemble of multiple classifiers that predicts label paths, node labels and depth-wise labels to decrease classification error. We tested our method on data sets from a leading e-commerce platform We gain nearly 4% PP@K , 2% CP@K , 2% LR@K and 3% LC@K using gwBoWV with the ensemble classifier compared to other competing methods without the ensemble classifier - see top 6 path prediction on Book Data in Table 5. On Non Book Data we gain nearly 2% in CP@K using gwBoWV without ensemble refer Table 3 and Table 4.

8 Future Work

The number of clusters K is a hyper-parameter we would like to learn this from the data set. We intend to extend the gwBOVW approach to incorporate the path class label in some fashion during the embedding. However, most results are shown on proprietary e-commerce datsets. We experimented gwBoWV with the 20newsgroup data-set and obtain state of art results(Liu et al., 2015) refer Table 6. We are currently applying the approach on large a scale public hierarchy like DMOZ and Wikipedia.

9 Acknowledgement

The authors are grateful to Mr. Dheeraj Mekala (B-Tech, CSE Department, IIT Kanpur) for carrying out preliminary experiments on the 20newsgroup data set to incorporate reviewer feedback. The authors also thank Dr. Muthaswamy Chelliah (Director, Academic Engagement, Flipkart) for facilitating collaboration between IIT Kanpur and Flipkart (E-Commerce) and Dr. Nagarajan Natarajan (PhD, UT-Austin) for encouraging feedback.

References

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

Samy Bengio, Jason Weston, and David Grangier. 2010. Label embedding trees for large multi-class tasks. In *Advances in Neural Information Processing Systems 23*, pages 163–171. Curran Associates, Inc.

Zellig Harris. 1954. Distributional structure. *Word*, 10:146–162.

Christopher D. Manning Jeffrey Pennington, Richard Socher. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. ACL.

Zornitsa Kozareva. 2015. Everyone likes shopping! multi-class product categorization for e-commerce. *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 1329–1333.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Omer Levy and Yoav Goldberg. 2014a. Dependencybased word embeddings. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2:302–308.

Omer Levy and Yoav Goldberg. 2014b. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014c. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27*, pages 2177–2185.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *AAAI Conference on Artificial Intelligence*.

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

Amitabha Mukerjee Pranjal Singh. 2015. Words are not equal: Graded weighting model for building composite document vectors. In *Proceedings of the twelfth International Conference on Natural Language Processing (ICON-2015)*. BSP Books Pvt. Ltd.

Dan Shen, Jean David Ruvini, Manas Somaiya, and Neel Sundaresan. 2011. Item categorization in the e-commerce domain. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1921–1924.

Dan Shen, Jean-David Ruvini, and Badrul Sarwar. 2012. Large-scale item categorization for e-commerce. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 595–604.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Chris Dyer Wang Ling. 2015. Two/too simple adaptations of wordvec for syntax problems. In *Proceedings of the 50th Annual Meeting of the North American Association for Computational Linguistics*. North American Association for Computational Linguistics.

Gui-Rong Xue, Dikan Xing, Qiang Yang, and Yong Yu. 2008. Deep classification in large-scale text hierarchies. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 619–626.

AttSum: Joint Learning of Focusing and Summarization with Neural Attention

Ziqiang Cao[1] **Wenjie Li**[1] **Sujian Li**[2] **Furu Wei**[3] **Yanran Li**[1]
[1]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[2]Key Laboratory of Computational Linguistics, Peking University, MOE, China
[3]Microsoft Research, Beijing, China
{cszqcao, cswjli, csyli}@comp.polyu.edu.hk
lisujian@pku.edu.cn
furu@microsoft.com

Abstract

Query relevance ranking and sentence saliency ranking are the two main tasks in extractive query-focused summarization. Previous supervised summarization systems often perform the two tasks in isolation. However, since reference summaries are the trade-off between relevance and saliency, using them as supervision, neither of the two rankers could be trained well. This paper proposes a novel summarization system called AttSum, which tackles the two tasks jointly. It automatically learns distributed representations for sentences as well as the document cluster. Meanwhile, it applies the attention mechanism to simulate the attentive reading of human behavior when a query is given. Extensive experiments are conducted on DUC query-focused summarization benchmark datasets. Without using any hand-crafted features, AttSum achieves competitive performance. We also observe that the sentences recognized to focus on the query indeed meet the query need.

1 Introduction

Query-focused summarization (Dang, 2005) aims to create a brief, well-organized and fluent summary that answers the need of the query. It is useful in many scenarios like news services and search engines, etc. Nowadays, most summarization systems are under the extractive framework which directly selects existing sentences to form the summary. Basically, there are two major tasks in extractive query-focused summarization, i.e., to measure the saliency of a sentence and its relevance to a user's query.

After a long period of research, learning-based models like Logistic Regression (Li et al., 2013) etc. have become growingly popular in this area. However, most current supervised summarization systems often perform the two tasks in isolation. Usually, they design query-dependent features (e.g., query word overlap) to learn the relevance ranking, and query-independent features (e.g., term frequency) to learn the saliency ranking. Then, the two types of features are combined to train an overall ranking model. Note that the only supervision available is the reference summaries. Humans write summaries with the trade-off between relevance and saliency. Some salient content may not appear in reference summaries if it fails to respond to the query. Likewise, the content relevant to the query but not representative of documents will be excluded either. As a result, in an isolated model, weights for neither query-dependent nor query-independent features could be learned well from reference summaries.

In addition, when measuring the query relevance, most summarization systems merely make use of surface features like the TF-IDF cosine similarity between a sentence and the query (Wan and Xiao, 2009). However, relevance is not similarity. Take the document cluster "d360f" in DUC[1] 2005 as an example. It has the following query: *What are the benefits of drug legalization?* Here, "Drug legalization" are the key words with high TF-IDF scores. And yet the main intent of the query is to look for "benefit", which is a very general word and does not present in the source text at all. It is not surprising that when measured by the TF-IDF cosine similarity, the sentences with top scores all contain the words "drug" or "legalization". Nevertheless, none of them provides advantages of drug legalization. See Section 4.6

[1]http://www-nlpir.nist.gov/projects/duc/

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 547–556, Osaka, Japan, December 11-17 2016.

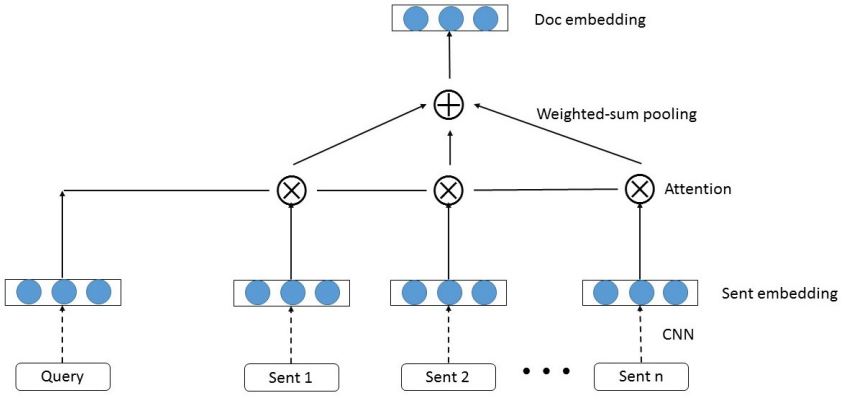


Figure 1: Generation of sentence and document cluster embeddings. "⊕" stands for a pooling operation, while "⊗" represents a relevance measurement function.

for reference. Apparently, even if a sentence is exactly the same as the query, it is still totally useless in the summary because it is unable to answer the query need. Therefore, the surface features are inadequate to measure the query relevance, which further augments the error of the whole summarization system. This drawback partially explains why it might achieve acceptable performance to adopt generic summarization models in the query-focused summarization task (e.g., (Gillick and Favre, 2009)).

Intuitively, the isolation problem can be solved with a joint model. Meanwhile, neural networks have shown to generate better representations than surface features in the summarization task (Cao et al., 2015b; Yin and Pei, 2015). Thus, a joint neural network model should be a nice solution to extractive query-focused summarization. To this end, we propose a novel summarization system called AttSum, which joints query relevance ranking and sentence saliency ranking with a neural attention model. The attention mechanism has been successfully applied to learn alignment between various modalities (Chorowski et al., 2014; Xu et al., 2015; Bahdanau et al., 2014). In addition, the work of (Kobayashi et al., 2015) demonstrates that it is reasonably good to use the similarity between the sentence embedding and document embedding for saliency measurement, where the document embedding is derived from the sum pooling of sentence embeddings. In order to consider the relevance and saliency simultaneously, we introduce the weighted-sum pooling over sentence embeddings to represent the document, where the weight is the automatically learned query relevance of a sentence. In this way, the document representation will be biased to the sentence embeddings which match the meaning of both query and documents. The working mechanism of AttSum is consistent with the way how humans read when having a particular query in their minds. Naturally, they pay more attention to the sentences that meet the query need. It is noted that, unlike most previous summarization systems, our model is totally data-driven, i.e., all the features are learned automatically.

We verify AttSum on the widely-used DUC 2005 $\sim$ 2007 query-focused summarization benchmark datasets. AttSum outperforms widely-used summarization systems which rely on rich hand-crafted features. We also conduct qualitative analysis for those sentences with large relevance scores to the query. The result reveals that AttSum indeed focuses on highly query relevant content.

The contributions of our work are as follows:

- We apply the attention mechanism that tries to simulate human attentive reading behavior for query-focused summarization;
- We propose a joint neural network model to learn query relevance ranking and sentence saliency ranking simultaneously.

2 Query-Focused Sentence Ranking

For generic summarization, people read the text with almost equal attention. However, given a query, people will naturally pay more attention to the query relevant sentences and summarize the main ideas

from them. Similar to human attentive reading behavior, AttSum, the system to be illustrated in this section, ranks the sentences with its focus on the query. The overall framework is shown in Fig. 1. From the bottom to up, AttSum is composed of three major layers.

CNN Layer Use Convolutional Neural Networks to project the sentences and queries onto the embeddings.

Pooling Layer With the attention mechanism, combine the sentence embeddings to form the document embedding in the same latent space.

Ranking Layer Rank a sentence according to the similarity between its embedding and the embedding of the document cluster.

The rest of this section describes the details of the three layers.

2.1 CNN Layer

Convolutional Neural Networks (CNNs) have been widely used in various Natural Language Processing (NLP) areas including summarization (Cao et al., 2015b; Yin and Pei, 2015). They are able to learn the compressed representations of n-grams effectively and tackle the sentences with variable lengths naturally. We use CNNs to project both sentences and the query onto distributed representations, i.e.,

$$\mathbf{v}(s) = \text{CNN}(s)$$
$$\mathbf{v}(q) = \text{CNN}(q)$$

A basic CNN contains a convolution operation on the top of word embeddings, which is followed by a pooling operation. Let $\mathbf{v}(w_i) \in \mathbb{R}^k$ refer to the k-dimensional word embedding corresponding to the i_{th} word in the sentence. Assume $\mathbf{v}(w_i : w_{i+j})$ to be the concatenation of word embeddings $[\mathbf{v}(w_i), \cdots, \mathbf{v}(w_{i+j})]$. A convolution operation involves a filter $\mathbf{W}_t^h \in \mathbb{R}^{l \times hk}$, which is applied to a window of h words to produce the abstract features $\mathbf{c}_i^h \in \mathbb{R}^l$:

$$\mathbf{c}_i^h = f(\mathbf{W}_t^h \times \mathbf{v}(w_i : w_{i+j})), \tag{1}$$

where $f(\cdot)$ is a non-linear function and the use of $tanh$ is the common practice. To simplify, the bias term is left out. This filter is applied to each possible window of words in the sentence to produce a feature map. Subsequently, a pooling operation is applied over the feature map to obtain the final features $\hat{\mathbf{c}}^h \in \mathbb{R}^l$ of the filter. Here we use the max-over-time pooling (Collobert et al., 2011).

$$\hat{\mathbf{c}}^h = \max\{\mathbf{c}_1^h, \mathbf{c}_2^h, \cdots\} \tag{2}$$

The idea behind it is to capture the most important features in a feature map. $\hat{\mathbf{c}}^h$ is the output of CNN Layer, i.e., the embeddings of sentences and queries.

2.2 Pooling Layer

With the attention mechanism, AttSum uses the weighted-sum pooling over the sentence embeddings to represent the document cluster. To achieve this aim, AttSum firstly learns the query relevance of a sentence automatically:

$$r(s, q) = \sigma(\mathbf{v}(s)\mathbf{M}\mathbf{v}(q)^T), \tag{3}$$

where $\mathbf{v}(s)\mathbf{M}\mathbf{v}(q)^T, \mathbf{M} \in \mathbb{R}^{l \times l}$ is a tensor function, and σ stands for the sigmoid function. The tensor function has the power to measure the interaction between any two elements of sentence and query embeddings. Therefore, two identical embeddings will have a low score. This characteristic is exactly what we need. To reiterate, relevance is not equivalent to similarity. Then with $r(s, q)$ as weights, we introduce the weighted-sum pooling to calculate the document embedding $\mathbf{v}(d|q)$:

$$\mathbf{v}(d|q) = \sum\nolimits_{s \in d} r(s, q)\mathbf{v}(s) \tag{4}$$

Notably, a sentence embedding plays two roles, both the pooling item and the pooling weight. On the one hand, if a sentence is highly related to the query, its pooling weight is large. On the other hand, if a sentence is salient in the document cluster, its embedding should be representative. As a result, the weighted-sum pooling generates the document representation which is automatically biased to embeddings of sentences match both documents and the query.

AttSum simulates human attentive reading behavior, and the attention mechanism in it has actual meaning. The experiments to be presented in Section 4.6 will demonstrate its strong ability to catch query relevant sentences. Actually, the attention mechanism has been applied in one-sentence summary generation before (Rush et al., 2015; Hu et al., 2015). The success of these works, however, heavily depends on the hand-crafted features. We believe that the attention mechanism may not be able to play its anticipated role if it is not used appropriately.

2.3 Ranking Layer

Since the semantics directly lies in sentence and document embeddings, we rank a sentence according to its embedding similarity to the document cluster, following the work of (Kobayashi et al., 2015). Here we adopt cosine similarity:

$$\cos(d, s|q) = \frac{\mathbf{v}(s) \bullet \mathbf{v}(d|q)^T}{||\mathbf{v}(s)|| \bullet ||\mathbf{v}(d|q)||} \tag{5}$$

Compared with Euclidean distance, one advantage of cosine similarity is that it is automatically scaled. According to (Kågebäck et al., 2014), cosine similarity is the best metrics to measure the embedding similarity for summarization.

In the training process, we apply the pairwise ranking strategy (Collobert et al., 2011) to tune model parameters. Specifically, we calculate the ROUGE-2 scores (Lin, 2004) of all the sentences in the training dataset. Those sentences with high ROUGE-2 scores are regarded as positive samples, and the rest as negative samples. Afterwards, we randomly choose a pair of positive and negative sentences which are denoted as s^+ and s^-, respectively. Through the CNN Layer and Pooling Layer, we generate the embeddings of $\mathbf{v}(s^+)$, $\mathbf{v}(s^-)$ and $\mathbf{v}(d|q)$. We can then obtain the ranking scores of s^+ and s^- according to Eq. 5. With the pairwise ranking criterion, AttSum should give a positive sample a higher score in comparison with a negative sample. The cost function is defined as follows:

$$\begin{aligned} &\epsilon(d, s^+, s^-|q) \\ &= \max(0, \Omega - \cos(d, s^+|q) + \cos(d, s^-|q)), \end{aligned} \tag{6}$$

where Ω is a margin threshold. With this cost function, we can use the gradient descent algorithm to update model parameters. In this paper, we apply the diagonal variant of AdaGrad with mini-batches (Duchi et al., 2011). AdaGrad adapts the learning rate for different parameters at different steps. Thus it is less sensitive to initial parameters than the stochastic gradient descent.

3 Sentence Selection

A summary is obliged to offer both informative and non-redundant content. While AttSum focuses on sentence ranking, it employs a simple greedy algorithm, similar to the MMR strategy (Carbonell and Goldstein, 1998), to select summary sentences. At first, we discard sentences less than 8 words like the work of (Erkan and Radev, 2004). Then we sort the rest in descending order according to the derived ranking scores. Finally, we iteratively dequeue the top-ranked sentence, and append it to the current summary if it is non-redundant. A sentence is considered non-redundant if it contains significantly new bi-grams compared with the current summary content. We empirically set the cut-off of the new bi-gram ratio to 0.5.

4 Experiments

4.1 Dataset

In this work, we focus on the query-focused multi-document summarization task. The experiments are conducted on the DUC 2005 $\sim$ 2007 datasets. All the documents are from news websites and grouped

into various thematic clusters. In each cluster, there are four reference summaries created by NIST assessors. We use Stanford CoreNLP[2] to process the datasets, including sentence splitting and tokenization. Our summarization model compiles the documents in a cluster into a single document. Table 1 shows the basic information of the three datasets. We can find that the data sizes of DUC are quite different. The sentence number of DUC 2007 is only about a half of DUC 2005's. For each cluster, a summarization system is requested to generate a summary with the length limit of 250 words. We conduct a 3-fold cross-validation on DUC datasets, with two years of data as the training set and one year of data as the test set.

Year	Clusters	Sentences	Data Source
2005	50	45931	TREC
2006	59	34560	AQUAINT
2007	30	24282	AQUAINT

Table 1: Statistics of the DUC datasets.

4.2 Model Setting

For the CNN layer, we introduce a word embedding set which is trained on a large English news corpus (10^{10} tokens) with the word2vec model (Mikolov et al., 2013). The dimension of word embeddings is set to 50, like many previous work (e.g., (Collobert et al., 2011)). Since the summarization dataset is quite limited, we do not update these word embeddings in the training process, which greatly reduces the model parameters to be learned. There are two hyper-parameters in our model, i.e., the word window size h and the CNN layer dimension l. We set $h = 2$, which is consistent with the ROUGE-2 evaluation. As for l, we explore the change of model performance with $l \in [5, 100]$. Finally, we choose $l = 50$ for all the rest experiments. It is the same dimension as the word embeddings. During the training of pairwise ranking, we set the margin $\Omega = 0.5$. The initial learning rate is 0.1 and batch size is 100.

4.3 Evaluation Metric

For evaluation, we adopt the widely-used automatic evaluation metric ROUGE (Lin, 2004) [3]. It measures the summary quality by counting the overlapping units such as the n-grams, word sequences and word pairs between the peer summary and reference summaries. We take ROUGE-2 as the main measures due to its high capability of evaluating automatic summarization systems (Owczarzak et al., 2012). During the training data of pairwise ranking, we also rank the sentences according to ROUGE-2 scores.

4.4 Baselines

To evaluate the summarization performance of AttSum, we implement rich extractive summarization methods. Above all, we introduce two common baselines. The first one just selects the leading sentences to form a summary. It is often used as an official baseline of DUC, and we name it "LEAD". The other system is called "QUERY_SIM", which directly ranks sentences according to its TF-IDF cosine similarity to the query. In addition, we implement two popular extractive query-focused summarization methods, called MultiMR (Wan and Xiao, 2009) and SVR (Ouyang et al., 2011). MultiMR is a graph-based manifold ranking method which makes uniform use of the sentence-to-sentence relationships and the sentence-to-query relationships. SVR extracts both query-dependent and query-independent features and applies Support Vector Regression to learn feature weights. Note that MultiMR is unsupervised while SVR is supervised. Since our model is totally data-driven, we introduce a recent summarization system DocEmb (Kobayashi et al., 2015) that also just use deep neural network features to rank sentences. It initially works for generic summarization and we supplement the query information to compute the document representation.

[2] `http://stanfordnlp.github.io/CoreNLP/`

[3] ROUGE-1.5.5 with options: -n 2 -m -u -c 95 -l 250 -x -r 1000 -f A -p 0.5 -t 0

To verify the effectiveness of the joint model, we design a baseline called ISOLATION, which performs saliency ranking and relevance ranking in isolation. Specifically, it directly uses the sum pooling over sentence embeddings to represent the document cluster. Therefore, the embedding similarity between a sentence and the document cluster could only measure the sentence saliency. To include the query information, we supplement the common hand-crafted feature TF-IDF cosine similarity to the query. This query-dependent feature, together with the embedding similarity, are used in sentence ranking. ISOLATION removes the attention mechanism, and mixtures hand-crafted and automatically learned features. All these methods adopt the same sentence selection process illustrated in Section 3 for a fair comparison.

4.5 Summarization Performance

The ROUGE scores of the different summarization methods are presented in Table 2. We consider ROUGE-2 as the main evaluation metrics, and also provide the ROUGE-1 results as the common practice. As can be seen, AttSum always enjoys a reasonable increase over ISOLATION, indicating that the joint model indeed takes effects. With respect to other methods, AttSum largely outperforms two baselines (LEAD and QUERY_SIM) and the unsupervised neural network model DocEmb. Although AttSum is totally data-driven, its performance is better than the widely-used summarization systems MultiMR and SVR. It is noted that SVR heavily depends on hand-crafted features. Nevertheless, AttSum almost outperforms SVR all the time. The only exception is DUC 2005 where AttSum is slightly inferior to SVR in terms of ROUGE-2. Over-fitting is a possible reason. Table 1 demonstrates the data size of DUC 2005 is highly larger than the other two. As a result, when using the 3-fold cross-validation, the number of training data for DUC 2005 is the smallest among the three years. The lack of training data impedes the learning of sentence and document embeddings.

It is interesting that ISOLATION achieves competitive performance but DocEmb works terribly. The pre-trained word embeddings seem not to be able to measure the sentence saliency directly. In comparison, our model can learn the sentence saliency well.

Year	Model	ROUGE-1	ROUGE-2
2005	LEAD	29.71	4.69
	QUERY_SIM	32.95	5.91
	SVR	36.91	**7.04**
	MultiMR	35.58	6.81
	DocEmb	30.59	4.69
	ISOLATION	35.72	6.79
	AttSum	**37.01**	6.99
2006	LEAD	32.61	5.71
	QUERY_SIM	35.52	7.10
	SVR	39.24	8.87
	MultiMR	38.57	7.75
	DocEmb	32.77	5.61
	ISOLATION	40.58	8.96
	AttSum	**40.90**	**9.40**
2007	LEAD	36.14	8.12
	QUERY_SIM	36.32	7.94
	SVR	43.42	11.10
	MultiMR	41.59	9.34
	DocEmb	33.88	6.46
	ISOLATION	42.76	10.79
	AttSum	**43.92**	**11.55**

Table 2: ROUGE scores (%) of different models. We draw a line to distinguish models with or without hand-crafted features.

4.6 Query Relevance Performance

We check the feature weights in SVR and find the query-dependent features hold extremely small weights. Without these features, the performance of SVR only drops 1%. Therefore, SVR fails to learn query relevance well. The comparison of AttSum and ISOLATION has shown that our method

AttSum	It acknowledges that illegal drugs cannot be kept out of the country by tougher border control and interdiction measures.
	Much greater resources, derived from taxation of the drugs that are now illegal and untaxed and from the billions saved by not wasting money on more criminal- justice measures, must be devoted to drug treatment and drug prevention.
	As is the case with tobacco, legalizing marijuana, cocaine and heroin would not signify an endorsement of their use.
	The consumption and production of marijuana in the United States is on the decrease, and that criminalization costs society more in terms of increased law-enforcement-related costs and deprived revenues from taxes on pot than legalization would.
TF-IDF	Drug prices have soared.
	Drug addicts are not welcome.
	How refreshing to have so much discourse on drugs and legalization.
	The only solution now is a controlled policy of drug legalization.
Query	What are the benefits of drug legalization?
AttSum	Boparai also said that wetlands in many developing countries were vital to the sustenance of human beings, not just flora and fauna.
	EPA says that all water conservation projects, and agriculture and forestry development along China's major rivers must be assessed in accordance with environmental protection standards, and that no projects will be allowed if they pose a threat to the environment.
	Finland has agreed to help central China's Hunan Province improve biodiversity protection, environmental education, subtropical forestry and wetlands protection, according to provincial officials.
	The EPA had sought as early 1993 to subject all development on wetlands to strict environmental review, but that approach was rejected by the courts, which ruled in favor of arguments made by developers and by the National Mining Association.
TF-IDF	Statistics on wetlands loss vary widely.
	Mitigation of any impact on wetlands by creating or enhancing other wetlands.
	The new regulations would cover about one-fourth of all wetlands.
	Now more and more people have recognized wetlands' great ecological and economic potential and the conservation and utilization of wetlands has become an urgent task.
Query	Why are wetlands important? Where are they threatened? What steps are being taken to preserve them? What frustrations and setbacks have there been?

Table 3: Sentences recognized to focus on the query.

can learn better query relevance than hand-crafted features. In this section, we perform the qualitative analysis to inspect what AttSum actually catches according to the learned query relevance. We randomly choose some queries in the test datasets and calculate the relevance scores of sentences according to Eq. 3. We then extract the top ranked sentences and check whether they are able to meet the query need. Examples for both one-sentence queries and multiple-sentence queries are shown in Table 3. We also give the sentences with top TF-IDF cosine similarity to the query for comparison.

With manual inspection, we find that most query-focused sentences in AttSum can answer the query to a large extent. For instance, when asked to tell the advantages of drug legalization, AttSum catches the sentences about drug trafficking prevention, the control of marijuana use, and the economic effectiveness, etc. All these aspects are mentioned in reference summaries. The sentences with the high TF-IDF similarity, however, are usually short and simply repeat the key words in the query. The advantage of AttSum over TF-IDF similarity is apparent in query relevance ranking.

When there are multiple sentences in a query, AttSum may only focus on a part of them. Take the second query in Table 3 as an example. Although the responses to all the four query sentences are involved more or less, we can see that AttSum tends to describe the steps of wetland preservation more. Actually, by inspection, the reference summaries do not treat the query sentences equally either. For this query, they only tell a little about frustrations during wetland preservation. Since AttSum projects a query onto a single embedding, it may augment the bias in reference summaries. It seems to be hard even for humans to read attentively when there are a number of needs in a query. Because only a small part of DUC datasets contains such a kind of complex queries, we do not purposely design a special model to handle them in our current work.

5 Related Work

5.1 Extractive Summarization

Work on extractive summarization spans a large range of approaches. Starting with unsupervised methods, one of the widely known approaches is Maximum Marginal Relevance (MMR) (Carbonell and Goldstein, 1998). It used a greedy approach to select sentences and considered the trade-off between saliency and redundancy. Good results could be achieved by reformulating this as an Integer Linear Programming (ILP) problem which was able to find the optimal solution (McDonald, 2007; Gillick and Favre, 2009). Graph-based models played a leading role in the extractive summarization area, due to its ability to reflect various sentence relationships. For example, (Wan and Xiao, 2009) adopted manifold ranking to make use of the within-document sentence relationships, the cross-document sentence relationships and the sentence-to-query relationships. In contrast to these unsupervised approaches, there are also various learning-based summarization systems. Different classifiers have been explored, e.g., conditional random field (CRF) (Galley, 2006), Support Vector Regression (SVR) (Ouyang et al., 2011), and Logistic Regression (Li et al., 2013), etc.

Many query-focused summarizers are heuristic extensions of generic summarization methods by incorporating the information of the given query. A variety of query-dependent features were defined to measure the relevance, including TF-IDF cosine similarity (Wan and Xiao, 2009), WordNet similarity (Ouyang et al., 2011), and word co-occurrence (Prasad Pingali and Varma, 2007), etc. However, these features usually reward sentences similar to the query, which fail to meet the query need.

5.2 Deep Learning in Summarization

In the summarization area, the application of deep learning techniques has attracted more and more interest. (Genest et al., 2011) used unsupervised auto-encoders to represent both manual and system summaries for the task of summary evaluation. Their method, however, did not surpass ROUGE. Recently, some works (Cao et al., 2015a; Cao et al., 2015b) have tried to use neural networks to complement sentence ranking features. Although these models achieved the state-of-the-art performance, they still heavily relied on hand-crafted features. A few researches explored to directly measure similarity based on distributed representations. (Yin and Pei, 2015) trained a language model based on convolutional neural networks to project sentences onto distributed representations. (Cheng and Lapata, 2016) treated single document summarization as a sequence labeling task and modeled it by the recurrent neural networks. Others like (Kobayashi et al., 2015; Kågebäck et al., 2014) just used the sum of trained word embeddings to represent sentences or documents.

In addition to extractive summarization, deep learning technologies have also been applied to compressive and abstractive summarization. (Filippova et al., 2015) used word embeddings and Long Short Term Memory models (LSTMs) to output readable and informative sentence compressions. (Rush et al., 2015; Hu et al., 2015) leveraged the neural attention model (Bahdanau et al., 2014) in the machine translation area to generate one-sentence summaries. We have described these methods in Section 2.2.

6 Conclusion and Future Work

This paper proposes a novel query-focused summarization system called AttSum which jointly handles saliency ranking and relevance ranking. It automatically generates distributed representations for sentences as well as the document cluster. Meanwhile, it applies the attention mechanism that tries to simulate human attentive reading behavior when a query is given. We conduct extensive experiments on DUC query-focused summarization datasets. Using no hand-crafted features, AttSum achieves competitive performance. It is also observed that the sentences recognized to focus on the query indeed meet the query need.

Since we have obtained the semantic representations for the document cluster, we believe our system can be easily extended into abstractive summarization. The only additional step is to integrate a neural language model after document embeddings. We leave this as our future work.

Acknowledgements

The work described in this paper was supported by Research Grants Council of Hong Kong (PolyU 152094/14E, PolyU 152248/16E), National Natural Science Foundation of China (61272291 and 61672445) and The Hong Kong Polytechnic University (4-BCB5, B-Q46C and G-YBJP). The correspondence authors of this paper are Wenjie Li and Sujian Li.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015a. Ranking with recursive neural networks and its application to multi-document summarization. In *Proceedings of AAAI*.

Ziqiang Cao, Furu Wei, Sujian Li, Wenjie Li, Ming Zhou, and Houfeng Wang. 2015b. Learning summary prior representation for extractive summarization. *Proceedings of ACL: Short Papers*, pages 829–833.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of SIGIR*, pages 335–336.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. End-to-end continuous speech recognition using attention-based recurrent nn: First results. *arXiv preprint arXiv:1412.1602*.

Ronan Collobert, Jason Weston, Lon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Hoa Trang Dang. 2005. Overview of duc 2005. In *Proceedings of DUC*, pages 1–12.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *JAIR*, 22(1):457–479.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of EMNLP*, pages 360–368.

Michel Galley. 2006. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proceedings of EMNLP*, pages 364–372.

Pierre-Etienne Genest, Fabrizio Gotti, and Yoshua Bengio. 2011. Deep learning for automatic summary scoring. In *Proceedings of the Workshop on Automatic Text Summarization*, pages 17–28.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on ILP for NLP*, pages 10–18.

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. Lcsts: A large scale chinese short text summarization dataset. In *Proceedings of EMNLP*, pages 1967–1972.

Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. 2014. Extractive summarization using continuous vector space models. In *Proceedings of EACL Workshop*, pages 31–39.

Hayato Kobayashi, Masaki Noguchi, and Taichi Yatsuka. 2015. Summarization based on embedding distributions. In *Proceedings of EMNLP*, pages 1984–1989.

Chen Li, Xian Qian, and Yang Liu. 2013. Using supervised bigram-based ilp for extractive summarization. In *Proceedings of ACL*, pages 1004–1013.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL Workshop*, pages 74–81.

Ryan McDonald. 2007. *A study of global inference algorithms in multi-document summarization*. Springer.

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

You Ouyang, Wenjie Li, Sujian Li, and Qin Lu. 2011. Applying regression models to query-focused multi-document summarization. *Information Processing & Management*, 47(2):227–237.

Karolina Owczarzak, John M Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9.

Rahul K Prasad Pingali and Vasudeva Varma. 2007. Iiit hyderabad at duc 2007. *Proceedings of DUC 2007*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*, pages 379–389.

Xiaojun Wan and Jianguo Xiao. 2009. Graph-based multi-modality learning for topic-focused multi-document summarization. In *IJCAI*, pages 1586–1591.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

Wenpeng Yin and Yulong Pei. 2015. Optimizing sentence modeling and selection for document summarization. In *Proceedings of IJCAI*, pages 1383–1389.

Using Relevant Public Posts to Enhance News Article Summarization

Chen Li[1], **Zhongyu Wei**[2]*, **Yang Liu**[3], **Yang Jin**[3], **Fei Huang**[4]
[1] Microsoft, Bellevue, WA, USA
[2] School of Data Science, Fudan University, Shanghai, P.R.China
[3] Computer Science Department, The University of Texas at Dallas
[4] Facebook, 770 Broadway, New York, NY, USA
chei@microsoft.com zywei@fudan.edu.cn
{yangl,yangjin@hlt.utdallas.edu} feihuang@fb.com

Abstract

A news article summary usually consists of 2-3 key sentences that reflect the gist of that news article. In this paper we explore using public posts following a new article to improve automatic summary generation for the news article. We propose different approaches to incorporate information from public posts, including using frequency information from the posts to re-estimate bigram weights in the ILP-based summarization model and to re-weight a dependency tree edge's importance for sentence compression, directly selecting sentences from posts as the final summary, and finally a strategy to combine the summarization results generated from news articles and posts. Our experiments on data collected from Facebook show that relevant public posts provide useful information and can be effectively leveraged to improve news article summarization results.

1 Introduction

Nowadays people are often overwhelmed by their daily exposure to large amount of online information. To make information easier to digest, news press like CNN, USAToday or news disseminator like Yahoo often provide 'summaries' for their news articles, so that readers can get the gist of a story quickly. Typically this kind of short summaries is manually generated. Obviously, it is very time consuming to manually produce high quality summaries for many popular topics. Therefore, automatic summarization for related news articles is essential to alleviate the manual work. With the popularity of social media, online news providers or disseminators are moving towards offering more interactions with news readers, for example, via comments on the news provide sites or post service like Twitter or Facebook public posts. When a news is published, we have access to not only the related news articles, but also the related public comments and posts. Our task in this paper is thus to explore how to use relevant public posts to improve summarization of a single news article. In particular, we use Facebook public posts related to a news article to help summarize a popular topic. This work is also motivated by the following observations of the data (see Sec 3 for the data we use). First, the posts under a news article are closely related to and very indicative for the topic of that news story. Second, the sentences from some posts whose accounts are maintained by news agencies are well written, so they may be directly used as the units of extractive summarization. In addition, the sentences in posts are often shorter than those from the news, thus again they may be more suitable to be used as summary sentences in sentence-based extractive summarization.

Our contributions in this paper are as follows: (1) We propose an integer linear programming (ILP) based news summarization approach using relevant Facebook public posts. It involves generating extractive and abstractive summaries. (2) We explore various ways of using post information to boost summarization performance. There are three general strategies: one is to leverage the lexical frequency information in the post to help estimate a word's importance in the news article and thus choose better summary sentences; another one is to extract sentences from the posts to form the summary; and the last one is to combine summarization results generated from the news articles and the posts. (3) To evaluate our method, we collect 190 popular news topics from Facebook. Each one has a news article, a human

*Corresponding Author

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 557–566, Osaka, Japan, December 11-17 2016.

generated summary and hundreds to thousands of related public posts. To our knowledge, this is the first data set of this kind.

2 Related Work

Our work is closely related to the following aspects: ILP based summarization method, dependency tree based sentence compression by considering extra information, and mining social media for document summarization.

Recently optimization methods have been widely used in extractive summarization. McDonald (2007) first introduced sentence level ILP for summarization. Later Gillick et al. (2009) revised it to concept-based ILP, which is similar to the Budgeted Maximal Coverage problem in (Khuller et al., 1999). Then other optimization methods have been used in summarization (Lin and Bilmes, 2010; Davis et al., 2012; Li et al., 2015b; Li et al., 2015a). In the concept-based ILP summarization methods, how to determine the concepts and measure their weights are the two key factors impacting the system performance. Woodsend and Lapata (2012) utilized ILP to jointly optimize different aspects including content selection, surface realization, and rewrite rules in summarization. Galanis et al. (2012) used ILP to jointly maximize the importance of the sentences and their diversity in the summary. In this work, we leverage the unsupervised ILP framework from Gillick et al. (2009) as our summarization system and incorporate post information to help boost summarization performance.

Sentence compression techniques are widely used in summarization in order to generate abstractive summaries. Previous research has shown the effectiveness of sentence compression for automatic document summarization (Knight and Marcu, 2000; Zajic et al., 2007; Chali and Hasan, 2012; Wang et al., 2013). The compressed summaries can be generated through a pipeline approach that combines a generic sentence compression model with a summary sentence pre-selection or post-selection step. In addition, joint summarization and sentence compression method attracts lots of attention these years. (Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011; Li et al., 2014) are typical work in this area. Their focus is to leverage the ILP technique to jointly select and compress sentences for multi-document summarization. In our work, we consider posts as summary related information and then use them for joint sentence compression and summarization.

Although there is little work about generating summaries by considering extra information on Facebook data, there is some similar work done on Twitter or other resources. Unsupervised method was tried for summarization by (Wong et al., 2008). (Phelan et al., 2011) used tweets to recommend news articles based on user preferences. (Gao et al., 2012) produced cross-media news summaries by capturing the complementary information from both sides. Kothari et al. (2013) and Štajner et al. (2013) investigated detecting news comments from Twitter for extending news information provided. Wei and Gao (2014) derived external features based on a collection of relevant tweets to assist the ranking of the original sentences for highlight generation. In addition to tweets, Svore et al. (2007) leveraged Wikipedia and query log of search engines to help document summarization. Tsukamoto et al. (2015) proposed a method for efficiently collecting posts that are only implicitly related to an announcement post, taking into account retweets on Twitter in particular. Our work involves the two aspects when using post information: one is that we utilize post information to help choose sentences from new articles and compress them to form a summary, and the other is that we directly use sentences from the posts as the summary.

3 Corpus Construction

For our work, we manually collected popular news stories and related data from a personal Facebook account during the period of Oct 20, 2015 to Nov 10, 2015. During that time, we collected the top 10 popular news stories every day (each story includes a human generated summary, a related news article and all the following public posts). The topics of these stories may come from politics, science and sport categories. An example of such a news summary and corresponding posts is shown in Fig1.

Due to the space limit, we only show one public post following the new story on the right side of the picture. In order to better evaluate the impact of the relevant posts, we ignore the popular news stories

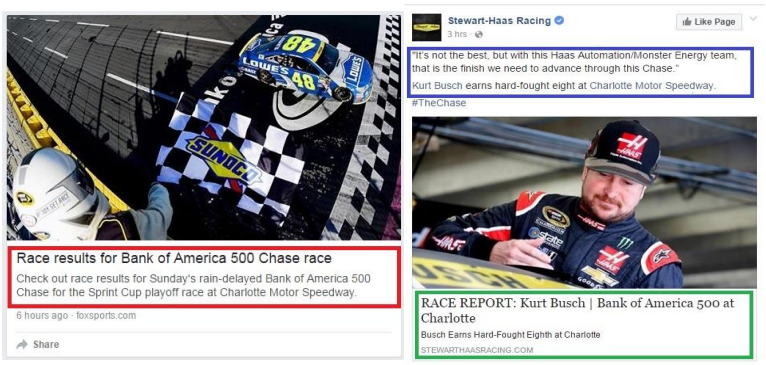


Figure 1: An example of a news story in our data set. The short manual summary is marked in red rectangle. The blue rectangle shows a post from a user. In the green rectangle, it is a link of a related news story. Some posts may only include comments, reactions, etc. without the link to the related news stories.

with less than 50 public posts. In total, we collected 190 popular news topics and their public posts[1].

The statistics of this corpus are given in Table 1. As shown in the table, the number of relevant posts for a popular topic varies a lot, with a mean of about 217 and standard deviation of 188. The high variance is because some of the topics are much more popular than others. We expect that the large number of relevant posts to a news story can provide useful information to guide the summary generation model. We can see from the table that the average sentence length from posts (12.85 tokens) is much shorten than that from the news (21.67 tokens). The average summary length for each topic is 43 words. This means that a summary can only contain on average two sentences from the news, or sometime just one long sentence. But usually such one or two sentences can not represent all the important information in the summary, therefore we may need to compress the long sentences in the news, or extract shorter sentences from the posts that contain similar information as the long sentences in the news.

	All	Politics	Science	Sports
# of Popular Topic	190	49	71	70
	News			
# of sent/news	22.83±13.27	27.71±15.91	19.94±12.13	22.35±11.23
# of token/sent	20.76±11.00	20.69±11.35	20.96±10.59	20.65±11.05
	Posts			
# of post/topic	216.45±187.56	299.04±262.56	236.58±166.81	138.23±87.77
# of sent/topic	454.28±468.54	459.01±280.48	725.08±753.52	259.93±171.71
# of token/sent	12.85±11.14	11.66±10.61	14.45±11.89	11.87±10.16
	Summary			
# of token/topic	43.34±4.76	43.68±4.37	42.22±4.39	44.22±5.14
# of token/sent	21.67±8.98	21.84±8.47	21.11±8.96	22.11±9.3

Table 1: Overview statistics on the corpus (mean and standard deviation)

4 Extractive Summarization Methods and Results

4.1 Background: ILP-based Document Summarization

The core idea of using ILP for summarization is to select the summary sentences by maximizing the sum of the weights of the language concepts that appear in the summary. Gillick et al. (2009) showed that using bigrams as concepts gave consistently better performance than unigrams or trigrams for a

[1]The data is available at http://www.hlt.utdallas.edu/∼chenli/summarization

variety of ROUGE measures. The association between the language concepts and sentences serves as the constraints. This ILP method is formally represented as below:

$$
\begin{aligned}
max \quad & \textstyle\sum_i w_i b_i && (1)\\
s.t. \quad & s_j Occ_{ij} \leq b_i && (2)\\
& \textstyle\sum_j s_j Occ_{ij} \geq b_i && (3)\\
& \textstyle\sum_j l_j s_j \leq L && (4)\\
& b_i \in \{0,1\}\ \forall i,\ \ s_j \in \{0,1\}\ \forall j && (5)
\end{aligned}
$$

b_i and s_j are binary variables that indicate the presence of a bigram and a sentence respectively. l_j is the sentence length and L is maximum length of the generated summary. w_i is a bigram's weight and Occ_{ij} means the occurrence of concept i in sentence j. Inequalities (2) and (3) associate the sentences and concepts. They ensure that selecting a sentence leads to the selection of all the concepts it contains, and selecting a concept only happens when it is present in at least one of the selected sentences.

4.2 Our Extractive Summarization Methods

The following describes all the extractive methods we use.

4.2.1 Generating summaries from news article

In this setup we extract sentences from news articles using the ILP based summarization framework. Our main goal is to investigate if we can use the relevant posts to better determine the bigrams and their weights in the ILP model described above. We compare the following three ways for the selection and weight of bigrams.

- **Bigram and Weight from News Article**: we use the bigram in the news article and its augmented term frequency as its weight: $w_i = 0.5 + \frac{f_{i,d}}{max\{f_{i,d}:i \in d\}}$ ($f_{i,d}$ is the raw frequency of bigram i in document d).

- **Bigram from News and Posts**: among the bigram candidates extracted from the news article, we use the subset that also appear in the posts, and the same weight as above (that is, the weight information is just based on the news article).

- **Bigram and Weight both from News and Post**: using the common bigrams from both the news article and posts (same as the previous setup), we further update the bigram weight by adding a bigram's post frequency in the relevant posts. In the following equation , pf_i is the number of posts that contain bigram i: $w_i^{'} = 0.5 + \frac{f_{i,d}}{max\{f_{i,d}:i \in d\}} + pf_i$.

4.2.2 Generating summary from posts only

In this setup, we evaluate whether sentences from posts are good candidates for a summary. Here each post can be seen as an individual document and we can treat this as a 'multi-document' summarization task and easily apply the ILP module on all the posts to choose a set of sentences as the final summary. In this process, the input sentences and bigrams are only from the posts, and the bigram weight is post frequency: $w_i^{''} = pf_i$ (Number of posts in which the bigram has appeared.).

4.2.3 Generating summary from news and posts

Here we use all the sentences from the news and posts as the input for summarization. This is again a multi-document summarization task, where we consider each post and the new article as a document. The bigram weight is document frequency. This method combines the news article and posts together to form a document collection for summarization. In the following we call it document level combination.

4.2.4 Combination of summarization results from news article and posts

In contrast to the above combination method, we can also build summarization systems using the news article and the relevant posts separately, and then combine the generated summaries. This kind of summary result level combination allows us to develop individual models tailored for different input sources, and may produce better combined final results. In this combination method, we have two summarization results, generated from the sentences in the news article and the posts respectively. Our aim is to decide which of the two summaries is better and use that as the final result. Since we do not have enough data to train supervised models, we propose to use heuristic rules to select which summary to use. The combination rules are based on the following parameters.

- **Sentence number**: $n_{sentNum}$. This represents how many sentences a summary result consists of. We observe that often when a summary contains just one sentence, that sentence is the news highlight and contains the most important information.

- **Bigram weight**: w_i in Section 4.1. Sentences containing bigrams with high weights are often good summary sentence candidates. We further define $w_{maxInTopic}$ as the maximum weight of the bigram in a topic, and $w_{maxInRes}$ as the maximum weight of the bigram in the summary result.

- **Bigram exist ratio**: R_{Bigram}, which represents the percentage of bigrams in a sentence that are used as variables in the ILP formula.We define this ratio since we prefer sentences that contain more bigrams that are used in the ILP model.

Then our rule-based classifier works by going through the following rules one by one. If a decision can be made at any point, the procedure will stop.

- **Rule 1**: If $n_{sentNum}$ from a summary result equals to one and the length of that sentence is longer than 40 words, choose that result. If both or neither equals to one, go to Rule 2.

- **Rule 2**: If $w_{maxInRes}$ from the post summary equals to $w_{maxInTopic}$, but if it is not true for the summary from news, choose the result from posts as the final summary. Otherwise, go to Rule3.

- **Rule 3**: If the maximum R_{Bigram} from a sentence in post result is larger than a threshold value[2], use the post result as the final summary; otherwise use the news result as the final summary. If the maximum R_{Bigram} from post and news results are the same, go to Rule 4.

- **Rule 4**: Choose the result with higher average R_{Bigram}. If the average R_{Bigram} is the same, go to Rule 5.

- **Rule 5**: Choose news result as the final result.

4.3 Experimental Setup and Results

The summary length is set as 45 words maximum (because the average length of human summary is 43 words in each topic). Note that a sentence in the post may be exactly the same as a sentence in the reference summary. One possible reason for this is that a user may simply copy the summary and then post it. In order to minimize this effect, in our data set we only consider the posts whose cosine similarity with the corresponding reference summary is less than 65%. We use the ROUGE evaluation metrics (Lin, 2004), with R-1 and R-2 measuring the unigram and bigram overlap between the system and reference summaries, and R-SU4 measuring the skip-bigram with the maximum gap length of 4.

We compare the following summarization methods:

(a) Summary sentences from news article I: bigrams are from news, and weight is their augmented term frequency from news.

[2]This value is empirically set as 0.85 in our experiments.

(b) Summary sentences from news article II: bigrams are from both news and posts, and weight is their augmented term frequency from news.

(c) Summary sentences from news article III: bigrams are from news, and weight is the combination of their augmented term frequency from news and their raw post frequency.

(d) Summary sentences from news article IV: bigrams are from news and posts, and weight is the combination of their augmented term frequency from news and their raw post frequency.

(e) Summary sentences from posts: bigrams are from posts, and weight is their post frequency.

(f) Document level combination: sentences are from news or posts, and bigram weight is 'document' frequency.

(g) Summary result level combination: given two summaries with sentences extracted from either news or posts, decide which one to use as the final result.

Table 2 presents the recall performance of these systems in ROUGE-1, ROUGE-2 and ROUGE-SU4 along with the corresponding 95% confidence intervals. We determine the statistical significance by comparing the 95% confidence intervals, that is, significant differences are those where the confidence intervals for the estimates of the means for the two systems either do not overlap, or where the two intervals overlap but neither contains the best estimate for the mean of the other.

From the results we find that systems using only information from the news (e.g., 'a') performs the worst. This also shows that this kind of single document summarization is not a trivial task. After adding information from posts, such as requiring the bigrams to also appear in posts (system 'b') or computing bigram weights using post related frequency (system 'd'), the results (system 'd' compared with 'a' and 'b') improved significantly. It is consistent with our expectation that post information can help enhance summarization of news topics.

System	ROUGE-1	ROUGE-2	ROUGE-SU4
a	0.30650 (0.29449 - 0.31896)	0.08621 (0.07620 - 0.09627)	0.10776 (0.09996 - 0.11737)
b	0.35453 (0.34173 - 0.36710)	0.12304 (0.11172 - 0.13474)	0.13940 (0.12948 - 0.14956)
c	0.37459 (0.36327 - 0.38507)	0.13655 (0.12698 - 0.14593)	0.14746 (0.13935 - 0.15554)
d	0.37943 (0.36838 - 0.39157)	0.14359 (0.13328 - 0.15548)	0.15391 (0.14503 - 0.16425)
d oracle	0.42377 (0.41130 - 0.43573)	0.21249 (0.20051 - 0.22445)	0.19915 (0.18825 - 0.21047)
e	0.39787 (0.38695 - 0.40930)	0.16292 (0.15314 - 0.17323)	0.16596 (0.15778 - 0.17464)
e oracle	0.54269 (0.53003 - 0.55503)	0.34810 (0.33195 - 0.36409)	0.31372 (0.29901 - 0.32948)
f	0.39182 (0.38048 - 0.40369)	0.15504 (0.14436 - 0.16643)	0.16359 (0.15489 - 0.17349)
g	0.40651 (0.39526 - 0.41793)	0.17254 (0.16178 - 0.18408)	0.17499 (0.16566 - 0.18532)

Table 2: ROUGE-N recall results for different extractive summarization systems.

One important finding from Table 2 is that system 'e' (using post sentences in extraction) performs even better than that from news article sentences. To better understand this, we conducted an oracle experiment when extracting sentences from the news article and posts respectively: we use the bigrams from the reference summary as the bigram concepts in the ILP method, and the weight is the bigram's term frequency in the reference summary. This oracle experiment can reflect the possible best result of the ILP extractive summarization system when extracting sentences from news or posts. The results are also included in Table 2. We can see that the possible best summaries from posts are also significantly better than that from news. By analyzing the results of this oracle experiment, we find that the average length of the generated summary is 38.15 tokens when using news, and is 41.25 when using posts. This means that the summary generated from posts may contain more information. Looking at this from another aspect, the news-based summary contains 2.1 sentences on average, in contrast to 2.5 sentences for the post-based summary. As mentioned earlier, the sentences from posts are often shorter than those from news. Therefore when the target summary has a short length limit (for example 45 tokens, usually

fewer than 3 sentences), one informative long sentence could use up all the length budget, while shorter sentences have more flexibility, allowing different information to be incorporated (sentence compression will be discussed in Section 5). Similar patterns are also found in the results of system d and e. The average length of the summary from system d is 41.8, and there are 2.3 sentences on average, comparing to the length of 43.4 words and 2.7 sentences for the summary from system e.

Even though overall system 'e' has the best performance, our analysis of results shows that only for 110 topics, the summary results from the posts are better than that from the news article, and for the remaining 80 topics, the results based on the news sentences are better. This also justifies why we expect combining results from the news and the posts based summaries may improve system performance. From the results in Table 2, we find that document level combination (system 'f') is not very effective. It is similar to the results using just the posts. A better bigram selection and weighting strategy may be needed when combining the posts and news at the input level. However, summary result level combination (system 'g') significantly outperforms each individual system, suggesting we can build each individual system, and then effectively choose one as the final output. The oracle result combination (i.e., comparing to the reference summary and picking the one with better scores as the system prediction) has a ROUGE-2 Recall score of 0.1922 (0.18085 - 0.20394). Our rule based combination method is quite close to the oracle combination result, indicating our rules can measure the goodness of a system generated summary.

5 Abstractive Summarization Method and Results

5.1 Dependency Tree Based Compression

We have mentioned that sentences from the news are generally long. Intuitively compressing the sentences in the news will give us room to incorporate more information. In fact, as discussed above, the summaries generated from the news sentences are on average shorter than that from the posts. This is due to the long sentences and the summary length constraint. Therefore next we investigate abstractive summarization by applying sentence compression when extracting sentences from news to improve summarization performance. Again the core idea of our proposed compression method is using the information from relevant posts to guide compression. Our compression framework is inspired by the work in (Filippova and Strube, 2008), where they use extra resources to guide the unsupervised dependency tree based sentence compression module.

The sentence compression task can be defined as follows: given a sentence s, consisting of words $w_1, w_2, ..., w_m$, identify a subset of the words of s, such that it is grammatical and preserves essential information of s. In the framework of (Filippova and Strube, 2008), a dependency graph for the original sentence is first generated and then compression is done by deleting edges of the dependency graph. The goal is to find a subtree with the highest score:

$$max \sum_{e_i \in E} a_{e_i} * w_{info}(e_i) * w_{syn}(e_i) \tag{6}$$

where a_{e_i} is a binary variable, indicating whether a directed dependency edge e_i is kept (a_{e_i} is 1) or removed (a_{e_i} is 0), and E is the set of edges in the dependency graph. The weighting of edge e considers both its syntactic importance ($w_{syn}(e_i)$) and the informativeness ($w_{info}(e_i)$). Suppose edge e_i is pointed from head h to node n with dependency label l, we use two methods to calculate the two weights in Formula 6.

The first one uses a bigram news corpus with the corresponding summaries: $w_{info}(e_i) = \frac{P_{summary}(n)}{P_{news}(n)}$ and $w_{syn}(e_i) = P(l|h)$, $P_{summary}(n)$ and $P_{news}(n)$ are the unigram probabilities of word n in the language models trained on human generated summaries and the original news articles respectively. $P(l|h)$ is the conditional probability of label l given head h. We used the New York Times Annotated Corpus (LDC Catalog No: LDC2008T19) as the extra background corpus. It has both the original news articles and human generated summaries.

In the second method, we explore using relevant posts as background information for compression. Here, $w_{info}(e_i) = \frac{pf(n)}{\#Post}$ and $w_{syn}(e_i) = \frac{pf(h,n)}{\#Post}$, $pf(n)$ is the number of posts including word n and

$pf(h, n)$ is the number of posts where n and head h appear together. If h and n appear together in two sentences in one post, it is counted as one. #Post represents the total number of posts in a topic.

5.2 Joint model for summarization and sentence compression

We propose a joint model for sentence selection and compression at the same time under the ILP framework, in order to avoid the problem with pre-compression (error propagation due to imperfect compression, important information may be missing) or post-compression (after compression it is hard to add new sentences to use the new available space). In the joint model, we combine the objectives in Section 4.1 and Formula 6, and thus the goal is to find a set of sentences with the highest score:

$$max \sum_{e_{jk} \in E} \lambda * a_{e_{jk}} * w_{info}(e_{jk}) * w_{syn}(e_{jk}) + \sum_i w_i b_i, \quad \forall i, j, k \tag{7}$$

e_{jk} means the k^{th} edge in the j^{th} sentence in this news article. λ is used to balance the contribution from the edge importance and bigram weights. After we add edges into our ILP-based summarization model, we need to adjust the previous constraints and also design more constraints to represent relationships between sentences and edges, and bigrams and edges in order to produce valid results.

First, the length constraint in Section 4.1 should be expressed in the form of edges rather than sentences.

$$\sum_{j,k} a_{e_{jk}} \leq L - 1, \ \forall j, k \tag{8}$$

Second, we want to avoid picking just a few words from many sentences as the summary, which typically leads to ungrammatical summaries. Hence it is more desirable to obtain a solution with only a few sentences extracted and compressed. To do so, we create the relationship between edges and sentences like following: if sentence j is selected, there are at least $\rho * L_j$ words extracted. L_j is the length of sentence j. This constraint is shown in the first inequality in Formula 9. Together with the second inequality there, they make sure that if sentence j is selected, at least $\rho * L_j$ words will be chosen; if sentence j is not selected, none of the edges from this sentence will be selected.

$$\sum_{j,k} a_{e_{jk}} \geq \rho * L_j * s_j, \quad a_{e_{jk}} \leq s_j, \ \forall j, k \tag{9}$$

Third, one bigram has two tokens, meaning it involves at least one edge and at most two edges. Therefore we build the relationship between bigrams and edges as follows:

$$b_i \geq a_{e_{jk}}, \qquad b_i \leq \sum a_{e_{jk}} \tag{10}$$

where e_{jk} represents all the edges whose head h or node n is one element of bigram i.

Forth, in the dependency tree, if an edge $e_{j,k}$ is removed, all the edges whose head node is $e_{j,k}$'s node n need to be removed as well.

$$a_{e^l_{jk}} \geq a_{e^{l+1}_{jk'}} \tag{11}$$

in which edge $e^{l+1}_{jk'}$ is at level $l+1$ and its head node is the node n of e^l_{jk} at level l. Please note we do not include the vice verse constraints. This means even if all the edge $e^{l+1}_{jk'}$ are removed, we can still keep edge e^l_{jk}.

In addition to all the constraints from Formula 7 to 11, we require that b_i, s_j and $a_{e_{jk}}$ are all binary variables. This gives the ILP setup for the joint summarization and sentence compression model.

5.3 Results

The abstractive summarization experiments are based on the setup of System 'd', that is, we extract sentences from the news articles, but the bigrams and their weight information come from both the news and the posts. We use the joint summarization and compression method described above, with extra background information to help guide compression. λ in Formula 7 and ρ in Formula 9 are empirically set as 20 and 0.85 respectively in our experiment.

Results are shown in Table 3. For System 'd', we present results using two different resources for compression: the generic NY Times Corpus and the relevant posts for each topic. We find adding compression improves summarization performance over the extractive summarization baseline. Using posts as extra information outperforms that using the general news. This improvement is also statistically significant. In the table we also include the result using the System 'g' configuration. For this method, once the combination rules determine to use the extractive summary from the news as the final system output, we apply abstractive summarization (i.e., joint compression and summarization) to this topic to regenerate the summary. We can see applying compression on these topics gave additional improvement over the original combination result.

Compression System		ROUGE-1	ROUGE-2	ROUGE-SU4
Based on	Extra Resource			
Sys d	NYT corpus	0.40437 (0.39326 - 0.41586)	0.15059 (0.14095 - 0.15985)	0.16311 (0.15484 - 0.17167)
	Post	0.41111 (0.40025 - 0.42282)	0.15567 (0.14561 - 0.16637)	0.17100 (0.16231 - 0.18051)
Sys g	Post	0.41232 (0.40133 - 0.42329)	0.17495 (0.16421 - 0.18653)	0.17871 (0.16983 - 0.18879)
Extractive System (d)		0.37943 (0.36838 - 0.39157)	0.14359 (0.13328 - 0.15548)	0.15391 (0.14503 - 0.16425)

Table 3: Recall of ROUGE-N results on abstractive summary.

6 Conclusion and Future Work

In this paper we explore utilizing relevant Facebook public posts in addition to news articles to generate a summary of popular news. We adopt the ILP based summarization method and propose different ways using information from posts, including weighting the bigrams using frequency information from the posts, compressing news sentences by estimating importance of dependence tree edges based on occurrence information in the posts, selecting sentences from posts as final summary, and finally combining the results generated from news articles and posts. Our experiments show that post information is useful for improving the performance.

We plan to pursue a number of directions in our future work. First, we plan to use a statistical classifier to choose a better summary for system combination. Second, we will perform more fine grained combination by choosing individual sentences from different results. Third, we will conduct human evaluation for our system results. Finally, it is worthwhile to investigate multi-document summarization once we can collect multiple news articles for a popular topic.

Acknowledgements

References

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of ACL*.

Yllias Chali and Sadid A. Hasan. 2012. On the effectiveness of using sentence compression models for query-focused multi-document summarization. In *Proceedings of COLING*.

Sashka T. Davis, John M. Conroy, and Judith D. Schlesinger. 2012. Occams - an optimal combinatorial covering algorithm for multi-document summarization. In *Proceedings of ICDM*.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*.

Dimitrios Galanis, Gerasimos Lampouras, and Ion Androutsopoulos. 2012. Extractive multi-document summarization with integer linear programming and support vector regression. In *Proceedings of the COLING*.

Wei Gao, Peng Li, and Kareem Darwish. 2012. Joint topic modeling for event summarization across news and social media streams. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. ACM.

Dan Gillick, Benoit Favre, Dilek Hakkani-Tur, Berndt Bohnet, Yang Liu, and Shasha Xie. 2009. The ICSI/UTD summarization system at tac 2009. In *Proceedings of TAC*.

Samir Khuller, Anna Moss, and Joseph Seffi Naor. 1999. The budgeted maximum coverage problem. *Information Processing Letters*.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *AAAI*.

Alok Kothari, Walid Magdy, Ahmed Mourad Kareem Darwish, and Ahmed Taei. 2013. Detecting comments on news articles in microblogs. In *Proceedings of ICWSM*.

Chen Li, Yang Liu, Fei Liu, Lin Zhao, and Fuliang Weng. 2014. Improving multi-documents summarization by sentence compression based on expanded constituent parse trees. In *Proceedings of EMNLP*.

Chen Li, Yang Liu, and Lin Zhao. 2015a. Improving update summarization via supervised ilp and sentence reranking. In *Proceedings of the NAACL*.

Chen Li, Yang Liu, and Lin Zhao. 2015b. Using external resources and joint learning for bigram weighting in ilp-based multi-document summarization. In *Proceedings of NAACL*.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of NAACL*.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Proceedings of ACL*.

Andre F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the ACL Workshop on Integer Linear Programming for Natural Language Processing*.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of ECIR*.

Owen Phelan, Kevin McCarthy, Mike Bennett, and Barry Smyth. 2011. Terms of a feather: Content-based news recommendation and discovery using twitter. In *Advances in Information Retrieval*. Springer.

Tadej Štajner, Bart Thomee, Ana-Maria Popescu, Marco Pennacchiotti, and Alejandro Jaimes. 2013. Automatic selection of social media responses to news. In *Proceedings of the 19th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*. ACM.

Krysta Marie Svore, Lucy Vanderwende, and Christopher JC Burges. 2007. Enhancing single-document summarization by combining ranknet and third-party sources. In *Proceedings of EMNLP-CoNLL*.

Yuma Tsukamoto, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2015. Collecting microblog posts implicitly related to announcement post. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *Proceedings of ACL*.

Zhongyu Wei and Wei Gao. 2014. Utilizing microblogs for automatic news highlights extraction. COLING.

Kam-Fai Wong, Mingli Wu, and Wenjie Li. 2008. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of COLING*.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proceedings of EMNLP-CoNLL*.

David Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. In *Information Processing and Management*.

A Proposition-Based Abstractive Summariser

Yimai Fang, Haoyue Zhu, Ewa Muszyńska, Alexander Kuhnle and Simone Teufel
University of Cambridge Computer Laboratory
15 JJ Thomson Avenue
Cambridge CB3 0FD, United Kingdom
{yf261,hz315,emm68,aok25,sht25}@cam.ac.uk

Abstract

Abstractive summarisation is not yet common amongst today's deployed and research systems. Most existing systems either extract sentences or compress individual sentences. In this paper, we present a summariser that works by a different paradigm. It is a further development of an existing summariser that has an incremental, proposition-based content selection process but lacks a natural language (NL) generator for the final output. Using an NL generator, we can now produce the summary text to directly reflect the selected propositions. Our evaluation compares textual quality of our system to the earlier preliminary output method, and also uses ROUGE to compare to various summarisers that use the traditional method of sentence extraction, followed by compression. Our results suggest that cutting out the middle-man of sentence extraction can lead to better abstractive summaries.

1 Introduction

Abstraction, rather than extraction, is generally seen as the more desirable method of performing automatic summarisation. It implies generating an entirely new text from the information contained in the input text. Amongst its advantages is the promise of better information packaging: conveying more information using less output text. But end-to-end abstractive summarisation is still uncommon in current systems, mostly due to the problems with NLP text analysis and knowledge representation that would plague any "deeper" method. Current research into abstractive summarisation therefore mainly concerns compressing individual sentences or merging sentences of similar content. Sentence compression using machine learning and/or constraint-solving methods is an active area of research. Common methods use syntactic, lexical and discourse-based features to determine which words should be dropped or paraphrased (Knight and Marcu, 2000; McDonald, 2006; Clarke and Lapata, 2008; Cohn and Lapata, 2007, 2008; Yoshikawa et al., 2012). More recently, neural language model has also been applied to the problem (Rush et al., 2015). For multi-document summarisation, sentence fusion (Barzilay and McKeown, 2005) allows the synthesis of common information across documents in one sentence, using grammatical dependencies as a substitute for a semantic representation. These methods have the ability to produce high-quality output, but their sentence generation is a local step in the summarisation pipeline, i.e. isolated from the content selection, which is essentially global. The only exceptions we know are Martins and Smith's (2009) system, and Nishikawa's (2014) system for Japanese text, both of which optimise sentence selection jointly with sentence compression.

A different paradigm for abstractive summarisation would be to avoid the "middle man" of sentence extraction altogether, and to operate over propositions instead. Such a system would first transform the text into sub-sentential semantic units, then perform summarisation operations over these (possibly performing inference and creating new semantic units), and finally use NL generation techniques to verbalise the semantic units that make up the final summary. Kintsch and van Dijk (1978, henceforth KvD) present one such summarisation technique. They use a tree of connected propositions to simulate the human memory. While the text is processed incrementally, new propositions are attached to the tree

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,
pages 567–578, Osaka, Japan, December 11-17 2016.

by argument overlap, and old propositions are pruned according to simulated memory limitations. The final summary is based on the best-connected propositions. We have presented in previous work (Fang and Teufel, 2016) an implementation of KvD's core ideas. However, it did not have the ability to generate new text from the propositions, and we were thus forced to use sentence extraction for the creation of a compromise summary output. We were nevertheless able to show that the content selection of our summariser outperforms state-of-the-art extractive summarisers.

The contribution in the current paper is the combination of our summariser with an existing NL generator that can verbalise the summary propositions. The propositions we use are created by aggregating grammatical dependencies gained from a syntactic parse with the Stanford Parser (Klein and Manning, 2003), in an attempt to create more meaningful semantic units similar to KvD's original concept. This close-to-surface representation allowed us ease of operation and robustness, but the grammatical relations do not contain enough semantic information to regenerate from them. During our research, we decided to experiment with the ACE processor[1], allowing parsing and generation from Minimal Recursion Semantics (MRS) representations (Copestake et al., 2005) and related formalisms such as Dependency Minimal Recursion Semantics (DMRS) (Copestake, 2009). For sentences containing summary propositions, our generation module aligns Stanford grammatical dependencies with their corresponding DMRS components (called Elementary Predicates or EPs). New text is then generated by selecting the minimal set of EPs such that it fulfils two conditions: 1. it covers all the information in the desired proposition from our summariser and 2. it contains all the EPs necessary for successful generation with ACE. The second condition ensures grammaticality.

How should one assess the quality of a summary sentence that minimally expresses the information in a proposition? Our first evaluation aims at measuring the grammaticality and truth preservation of the generated sentences using a human experiment. It is clear that producing grammatical, semantically and pragmatically interpretable text is a necessary goal for any abstractive technology. Truth-preservation is a property that also needs to be evaluated carefully whenever sentential material is manipulated. It is easy to accidentally introduce effects that would distort the meaning of the new sentence, be it by inappropriate referring expressions, by dropping restrictive relative clauses and thus changing the truth-conditional conditions of the sentence, or by many other subtle unintentional changes. In our evaluation, we ask humans to interpret the shortened sentence together with its original context in order to detect possible truth distortions.

Our second evaluation uses the ROUGE metric (Lin, 2004) to evaluate the overall content selection of our abstractive end-to-end method. We cannot directly compare our output to that of an isolated sentence compressor; we first need to select the sentences. We therefore build a pipeline system that uses our summariser as the sentence extractor, and then compresses the summary sentences with two competitive sentence compression methods (Clarke and Lapata, 2008; Cohn and Lapata, 2007). This gives the extraction–compression route a fair chance because our system as a sentence extractor was found to be the best system evaluated in our previous work (Fang and Teufel, 2016). As a further comparison system, we additionally use the next best "generation" algorithm that can express the information in a proposition: one that simply extracts all words that gave rise to the proposition (cf. section 4.2). This is the output mechanism we used in (Fang and Teufel, 2014).

Our results show that our system produces sentences of a good textual quality, and that the overall content selection (as measured in ROUGE) rivals current sentence-selection, while achieving much stronger compression than traditional sentence compressors do. We see two possible reasons why leaving out the middle-man of sentence selection is advantageous in abstractive summarisation. General sentence compression systems are trained to recognise material that tends to be dropped in a "general case". But without information about global discourse effects in the overall text, such a compressor cannot choose particular information selectively inside the sentence. The sentence extractor, which is run independently of it, would typically have selected sentences based on different criteria. In contrast, our system has access to global information about what connects the best propositions, which, as we would argue, enables it to perform better content selection. Secondly, the fact that our system works on propositions, i.e.,

[1]The Answer Constraint Engine, `http://sweaglesw.org/linguistics/ace/`

smaller semantic units, also affords it better information packaging abilities. This is shown by the fact that our abstractive summariser compresses sentences more heavily than traditional systems, because it uses an input sentence only as "raw material" to realise the small piece of information which it has determined to be essential.

In what follows, we will review previous work including the operation of the underlying proposition-based summariser and the NL generator we use (section 2). We will then explain our generation algorithm (section 3), describe our evaluations (section 4), and conclude with a discussion of our results.

2 Background

We now introduce the two modules we use: the summariser that chooses the summary propositions, and the grammar on which the generation operates.

2.1 Proposition-based summarisation

Our proposition-based summariser was introduced in Fang and Teufel (2014) and improved in Fang and Teufel (2016). The summariser maintains a coherence tree, which simulates human working memory. It processes the text sentence by sentence, incrementally adds new propositions to the tree, and prunes old propositions off the tree. Propositions are attached to existing propositions in the tree with which they share arguments (semantic participants). They are "forgotten" if the summariser predicts that no new propositions are likely to be attached to them, using a "leading-edge strategy" that prefers keeping recent nodes that are high-level (i.e. of small depth) on the tree. The idea of applying a graph algorithm on a semantic representation is not new to unsupervised summarisation (Vanderwende et al., 2004), but we have shown that the incremental processing of our model is superior to computing centrality on a single graph that represents the full text (Fang and Teufel, 2016).

The summariser is inspired by KvD's theory of text comprehension. The computational model we present is a simplification of the theory in two ways: 1. It does not generalise propositions or create meta-statements (macro-propositions) for specific domains. (These operations could in principle be performed if NLP technologies such as robust inference and entailment recognition were in place.) Instead it defines the summary-worthiness of a proposition in the simplest possible way, as the number of cycles during which it is retained in memory. 2. Rather than using "concepts" as arguments in the propositions (which would require human intelligence), we create propositions based only on a syntactic parse, and only containing textual tokens, not intelligent "concepts". Our propositions are comparable to the subject-verb-object triples in Genest and Lapalme (2011), but we allow more kinds of predication such as adjectival and prepositional modifications to be propositions in their own right, which means these pieces of information can be selected independently. For example, "*The discovery of the element revolutionised fire-lighting*" gives rise to the propositions *revolutionised (the discovery, fire-lighting)* and *of (the discovery, the element)*. We use coreference resolution and models of semantic relatedness (such as lexical chains and cosine similarity in a vector space) to determine the overlap between these arguments.

Starting from a list of summary propositions found to be important by the summariser, how should we generate a textual summary from it? Doing so is not only important for the usability of the summariser, but also for automated evaluation – ROUGE evaluates the quality of summaries by comparing them to human-written gold standard summaries. It cannot evaluate abstract propositions without surface forms. To solve this problem, we previously made our summariser extract full sentences that contain the summary propositions. However, this does not fully demonstrate the fine granularity of the content selection of our proposition-based summariser, and the advantages this granularity brings. We also ideally want grammatical sentences that *only* realise the information from the propositions selected by the summariser. We do not want the summary to be diluted by any other content that happened to co-occur with it in the same input sentence. In this paper we present a solution, namely NL generation from propositions.

2.2 ACE and The English Resource Grammar

ACE is one of the processors designed to work with DELPH-IN HPSG wide-coverage, linguistically motivated grammars such as the English Resource Grammar (ERG) (Flickinger, 2000; Flickinger et al.,

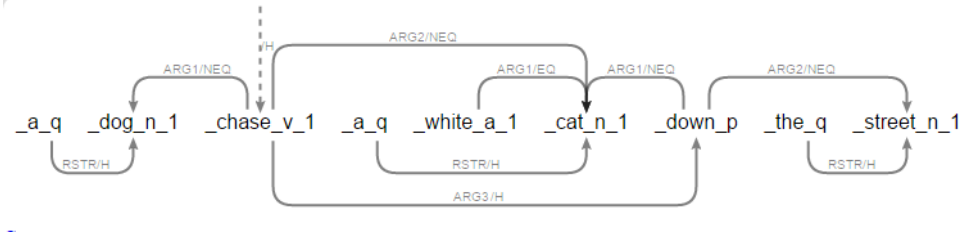


Figure 1: A DMRS graph for the sentence "*A dog chased a white cat down the street.*"

2014), a broad-coverage, symbolic, bidirectional grammar of English. It was developed as part of the DELPH-IN initiative[2] and LinGO[3] project. The ERG uses Minimal Recursion Semantics (MRS) (Copestake et al., 2005) as its semantic representation. In contrast to this, the kind of generation that is traditionally used in summarisation to realise proposition-like units is much shallower, e.g. SimpleNLG as used by Genest and Lapalme (2011, 2013). The MRS format can be transformed into a more readable Dependency Minimal Recursion Semantics (DMRS) graph (Copestake, 2009), which represents its dependency structure. An example of a DMRS graph is shown in Figure 1. The nodes correspond to predicates (EPs); edges (links), represent relations between them.

DMRS graphs can be manipulated using two existing Python libraries. The `pyDelphin` library[4] is a more general MRS-dedicated library. It allows conversions between MRS and DMRS representations but internally performs operations on MRS objects. The `pydmrs` library[5] (Copestake et al., 2016) is dedicated solely to DMRS manipulations.

3 NL generation from propositions

To generate, a deep semantic representation for generation is needed – as well as the propositions, which represent the content we want to include in the summary. But for historic reasons, the front-end of our summariser is the Stanford parser. At the time, propositions based on dependency relations seemed to provide the best fit to KvD's notion of propositions. After realising the shortcomings of shallow generation by extraction, and after the ACE generator became available, we started experimenting with deep generation of summaries using DMRS, as described in this paper. As a downside of this development, we now have to run two parsers on each sentence that is involved in the final summary.

We iteratively generate summary text for each proposition in the ranked list of propositions output by our summariser described in section 2.1, until the summary length limit is reached. For each proposition to be verbalised, we use the original sentence that contains it as the source, parse it using ACE, and then manipulate its DMRS representation so as to generate a (much shortened and) grammatical summary sentence, which should then express the meaning of the proposition in a human-digestible form.

Because propositions from a sentence are ranked independently, as we go down the ranked list, it is possible that we encounter a proposition that shares the source sentence with one or more propositions that were verbalised a few iterations ago. In this case, we restart the generation on that source sentence, and require the new formulation to include the information of both the new and the old propositions. The previously generated summary sentence is then replaced by the new formulation.

3.1 Initial nodes selection

The first step of generating a sentence is to select the initial set of nodes (EPs) in the DMRS representation of the source sentence. This initial set corresponds to the one or more propositions we want to verbalise from the source sentence. A proposition contains information of the index of the source sentence, as well as the textual tokens of its functor and arguments. The character offset of a token is useful because a token in the dependency parse (and therefore in the proposition) may not necessarily align with an EP of

[2]Deep Linguistic Processing with HPSG, `www.delph-in.net`
[3]Linguistic Grammars Online, `lingo.stanford.edu`
[4]`https://github.com/delph-in/pydelphin`
[5]`https://github.com/delph-in/pydmrs`

ERG. For example, the Stanford parser analyses "*didn't*" as two tokens `did` and `n't`, while ERG treats it as one grammar predicate called `neg`. Referring to the character span of `neg`, we are able to select it if the propositions contain either token.

We also need a strategy for unknown words, as ACE does not allow generation from a DMRS if any of its EPs contains an unknown word. We handle this by a temporary replacement mechanism where dummy words stand in for unknown words in the input text.

3.2 Node set expansion for grammaticality

After obtaining the initial node set, our algorithm appends additional nodes and removes existing nodes, based on syntactic information from the ERG parse. The most essential general rule is to always add EPs which are syntactic arguments of already selected EPs. These arguments are indicated by NEQ and HEQ links in the ERG. The ERG has more sophisticated mechanisms than shallower parsers for dealing with subcategorisation in general, which should improve the grammaticality of our results. Our rules include adding compulsory prepositional complements of verbs, adding quantifiers, and dismantling sentence coordination.

We have devised specialised rules for cases when the syntactic parse is more superficial than the ERG's analysis. For instance, the ERG can handle complicated cases of quantification, e.g. "*most of the students*", by using additional meaning-bearing EPs which express semantic relationships such as a part-of relationship or a quantity relationship. In the Stanford dependencies, however, the quantifier "*most*" is the head of the noun phrase, modified by a prepositional phrase "*of the students*". We therefore prevent the undesirable outcome where "*most*" is selected on its own for the summary, by including the semantic head "*the students*", which is connected to the predicate `part_of` via an NEQ link.

As an example, consider a source sentence encountered in our evaluation:

> *In Britain, for example, the dull weather of winter drastically cuts down the amount of sunlight that is experienced which strongly affects some people.*

Extracting the textual tokens that gave rise to the selected summary propositions produces an uninformative and ungrammatical sentence for the summary:

> *In Britain, for example, the weather cuts down the amount strongly affects.*

But our ERG-based generation preserves the semantic head of the quantity relationship, and the obligatory object of the transitive verb:[6]

> *In Britain, the weather cuts down the amount of sunlight, which affects some people, strongly.*

We have also written rules for abstract DMRS predicates such as compound, implicit conjunction, and apposition. We use about ten specialised rules to treat certain constructions, such as "*in order to*", and phrasal verbs such as "*keep from*". The system could be strengthened in the future by the systematic addition of more such rules.

3.3 Node set expansion for graph connectivity

Generation with ACE requires a connected DMRS graph. In this step we ensure connectivity of the subgraph using an undirected graph algorithm. But the subgraph consisting of the EPs selected so far (henceforth "the subgraph"), which we will use for generation, may or may not be connected: If the initial set of EPs was already connected, the expanded set created by the algorithm in section 3.2 will certainly remain connected. If the initial set of EPs was not connected, however, the algorithm for grammaticality can sometimes make the set connected, but there is no guarantee.

Making a disconnected subgraph connected by including the minimum set of additional nodes corresponds to the NP-hard Steiner tree problem (Hwang et al., 1992). Our greedy approach to this problem is to iteratively grow the largest connected component. An iteration starts with the largest connected

[6]Please note that the adverbial modifier "*strongly*" appears in a different place from the source sentence, but is still grammatical. This is an effect of the generator's degree of freedom when verbalising a semantic representation, discussed in section 3.5.

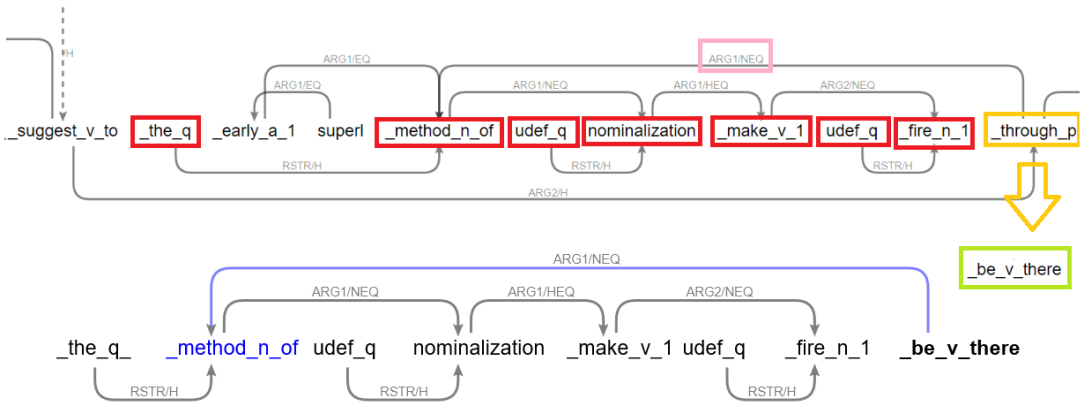


Figure 2: A "*there be*" generation.

component in the subgraph. We want to include one additional node into the subgraph in each iteration, and we select such a node among the non-selected nodes that are adjacent to this connected component. First, we prefer including a node if that node can join the maximum number of other connected components to this component. Second, in case of a tie, we prefer a node that is positioned in the EP sequence (i.e. in textual order) between this connected component and an other connected component. This serves as a heuristic that guides us towards the nodes that are adjacent to the other components. Third, as an additional tiebreaker, we prefer a node that is connected to this connected component via an NEQ link over one connected via an EQ link. After adding the additional node to the subgraph, the next iteration starts with the currently largest connected component in the subgraph. The process continues until there is only one connected component in the subgraph, i.e. the subgraph is connected.

3.4 Node modification

We always want to generate full, grammatical sentences. For propositions that are expressed as noun phrases in the original text, the best solution would be to reformulate the phrase in a verbal predicate. But this is hard, so instead we devised a graph-based procedure that generates "*there be*" sentences for them. This algorithm works by finding the local top node for the set of selected nodes (called `s_n`), and changing this local top node to `_be_v_there_`.

Figure 2 shows a part of the subgraph of the DMRS representation of the sentence

> *Studies of primitive societies suggest that the earliest method of making fire was through friction.*

After expanding the node set from the initial set indicated by our selected proposition, we arrive at the selected node set as shown in the red boxes, namely {`_the_q`, `_method_n_of`, `udef_q`, `nominalization`, `_make_v_1`, `udef_q`, `_fire_n_1`}. The local top node for the node set is `_through_p` (shown in the yellow box); it is connected to the noun predicate `_method_n_of` with an outgoing NEQ link (shown in the pink box), and it is the only link connected with the set. Therefore, we replace the local top with the predicate `_be_v_there_`, and pass the resulting DMRS subgraph to the generator, resulting in the summary sentence

> *There is the method of making fire.*

While this sentence is not perfectly natural, it verbalises the information about fire-making methods reasonably fluently, without introducing unnecessary information from the source sentence that was not selected for the summary.

3.5 Selection among generations

The final step of our method concerns the selection of the best surface sentence generated by ACE. Given a DMRS, ACE generates a list of sentence variants covering the DMRS's semantics, which can

differ by active vs. passive tense, verb alternations, order variations and various other effects. ACE ranks the variants using a statistical model according to naturalness. If a sentence is parsed into EPs, the regenerated sentence from these EPs may therefore be different from the original (e.g. "*Kim gave a book to Sandy*" can turn into "*Kim gave Sandy a book*"). But sentence reformulation is undesirable for us, because we want the shortest sentence that is also closest in meaning and external form to the original sentence – and not necessarily the "most likely" sentence. We therefore select, from the top five generated sentences, the one that shares the longest common subsequence (LCS) with the original sentence.[7]

Of course, Stanford and ACE do not always give the same sentence an equivalent analysis, and not all ACE parses and generations succeed. ACE parsing could fail for different reasons, mostly in our case due to very complex and long sentences, unforeseen syntactic constructions and unusual punctuation. We used our corpus of 108 documents described below to measure its coverage, which is 86.5%.[8] Our backoff strategy in case of parsing or generation failure is to produce the sequence of all tokens which are involved in the proposition, as we did in Fang and Teufel (2014). This primitive generation technique, also a baseline system we use in the following section, is quite likely to produce ungrammatical output, but it is generally applicable to any proposition.

4 Evaluation

For both evaluations we perform, we use the IELTS summary corpus, which we introduced in Fang and Teufel (2016). The IELTS is a standardised test of English proficiency for non-native speakers. The texts we use are taken from the academic reading sections of the official practice tests. We use all 108 such documents available in *Cambridge IELTS 1–9*,[9] of which we randomly sampled 31. For each of the 31 texts, we commissioned four 100-word summaries by 15 native, near-native, or highly proficient speakers of English. We chose this corpus over the commonly-used news corpora for evaluation, because the IELTS texts are not in the journalistic genre, where the lead sentences are usually abstract-like, yet they are naturally occurring, generally understandable and of high editorial standard. They have the additional advantage that they do not need specialised world knowledge in the areas of politics or economics, as news articles (and particularly the frequently used Wall Street Journal) often do. This is beneficial if one wants to study text understanding-based forms of summarisation.

We use human judgement to evaluate the textual quality of individual generated sentences, and use the automatic ROUGE metric to evaluate the content selection ability on entire summaries.

4.1 Systems

We test four kinds of generation mechanisms for summarisation here (Figure 3): 1. OurAbs, our KvD-based summariser with NL generation, which realises the summary propositions directly as new text; 2. various sentence extraction systems (without sentence compression); 3. various combinations of sentence extraction systems followed by sentence compression systems; and 4. OurTok, our KvD summariser with the primitive generation technique, which only outputs the word tokens involved in summary propositions in textual order, without any attempt at generating fully grammatical sentences.

Both OurAbs (category 1) and OurTok (category 4) are based on first deriving a ranked list of summary propositions, where the importance (weight) of a proposition is determined by the KvD-style simulation of text understanding (cf. section 2.1). In principle, a summariser in categories 2 and 3 could also use the same information to extract sentences (followed by a compression stage for category 3). For instance, an extractor can choose sentences such that the sum of the weights of the propositions from these sentences is maximised. We have in fact implemented such a sentence extractor and formally

[7] If there is a tie, we choose the sentence with fewer words. If candidates of the maximum LCS are equally long, we choose the top-ranked generation.

[8] We also implemented a simple sentence simplification method that splits unparsed sentences at top-level S coordination nodes, which increased coverage to 90%. But we do not further report on this method here because its ROUGE results suffered slightly.

[9] For example, *Cambridge IELTS 9: authentic examination papers from Cambridge ESOL*, Cambridge University Press, ISBN: 9781107615502.

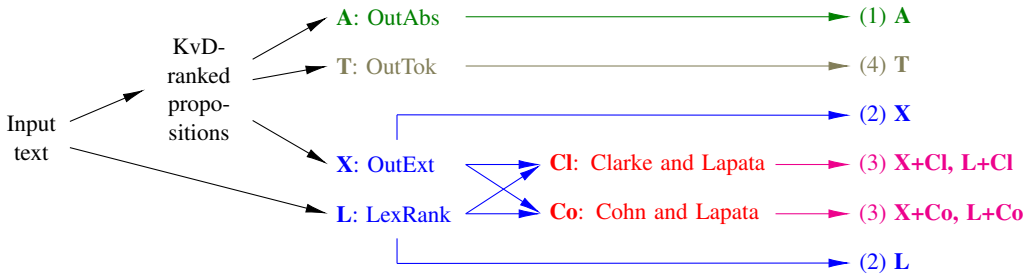


Figure 3: The 8 summarisation systems in 4 categories.

	Grammatical			Truthful		
	OurAbs	OurTok	Tie	OurAbs	OurTok	Tie
Average	20	5	15	25.3	5.3	9.3
Total	120	30	90	152	32	56

Table 1: Text quality evaluation: number of "higher" judgements given.

evaluated its performance in Fang and Teufel (2016); we call it OurExt here.

Alternatively, the sentence extraction module could be a shallower summariser that is not based on propositions. For instance, LexRank (Erkan and Radev, 2004) is a summariser that internally represents the input document as a graph of sentences. The inter-sentential similarity dictates the edges and their weights, and the centrality algorithm computes the value of a sentence by the probability of a random walk arriving at it. It does not build the graph incrementally, and it only measures the importance of sentences but not meaning units.

We use two sentence compression methods in combination with OurExt and LexRank. Clarke and Lapata (2008) use integer linear programming (ILP) to find the optimal compression per sentence within linguistic constraints. We choose its unsupervised version,[10] which requires a language model. We trained a trigram language model on the 100M-word British National Corpus (BNC) with interpolated Kneser-Ney smoothing and the "unknown word" token. Cohn and Lapata (2007) present a supervised tree-to-tree transduction method for sentence compression.[11] It is a variation of the Noisy Channel model for sentence compression first introduced by Knight and Marcu (2000). Instead of the Broadcast News Corpus they used, we trained the tree rewriting model on the full Written News Corpus, which only became available later (Clarke and Lapata, 2008). Following them, we aligned the words between the original and the compressed sentences using GIZA++ (Och et al., 1999), with additional pairs added where each word in the lexicon is aligned with itself, and symmetrised the alignment using the intersection heuristic (Koehn et al., 2003). We used the Stanford PCFG Parser (Klein and Manning, 2003) to obtain the sentence parses, and the above-mentioned BNC language model for training and decoding.

Both compressors were run on all input sentences.[12] The sentence extractor (OurExt or LexRank) computes the summary-worthiness of the uncompressed sentences, and then the corresponding compressed sentences are output.

4.2 Textual quality evaluation

We performed a human experiment to evaluate the textual quality of the sentences generated by our new system OurAbs, in comparison to OurTok. This evaluation is performed on individual sentences. We only tested sentences where OurAbs's generation succeeded, as OurAbs's fall-back strategy is identical to OurTok's normal operation.

We recruited six participants for this study. They were provided with 40 sentence triples, each consisting of a sentence from the input document, and both outputs by OurTok and OurAbs based on the same

[10] We used the implementation from `https://github.com/cnap/sentence-compression`

[11] We obtained the implementation from the authors.

[12] The Cohn and Lapata compressor failed to process 8 sentences. If any of them is chosen for summary, the uncompressed sentence is used instead.

	R-1	R-2	R-L	R-SU4
Abstract generation from propositions				
OurAbs (A)	0.364	0.088	0.340	0.131
Sentence extraction with compression				
X + Cl	0.361	0.090	0.335	0.132
X + Co	0.340	0.074	0.321	0.113
L + Cl	0.356	0.077	0.325	0.126
L + Co	0.336	0.067	0.314	0.110
Sentence extraction				
OurExt (X)	0.376	0.122	0.345	0.154
LexRank (L)	0.349	0.087	0.316	0.129
Token extraction for propositions				
OurTok (T)	0.356	0.088	0.336	0.130

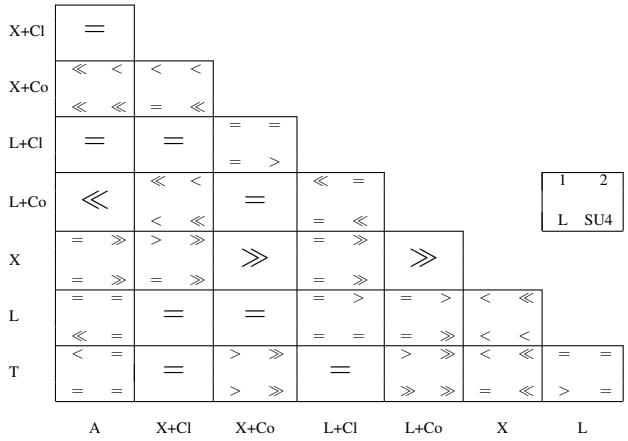


Table 2: ROUGE F-scores and statistical significance of the differences. The four positions in the significance table correspond to ROUGE-1, 2, L and SU4, respectively. "$\gg$" means row statistically outperforms column at $p < 0.01$ significance level; "$>$" at $p < 0.05$ significance level, and "$=$" means no statistical difference detected.

underlying summary proposition(s), in randomised order across triples. They were asked to rank the two summaries for two properties: 1. Which sentence is more grammatical? 2. Which sentence is more truthful to the meaning of the original sentence? They were allowed to judge two summaries as equally good (i.e. tie). The total number of judgements in this evaluation is 240 (40×6) for each question.

The results are given in Table 1. In 50% of cases (120/240), OurAbs was judged as strictly more grammatical than OurTok; as opposed to 12% of cases where OurTok was judged as strictly more grammatical. In terms of truth preservation, OurAbs was judged as distorting the truth less in 63% of cases (152 out of 240), whereas OurTok was seen as more truth-preserving in 13% of cases (32 out of 240). All differences are statistically significant as determined with the sign test at $p = 0.05$. Seeing that the new system improved or did not decrease grammaticality in 87% of cases where generation was successful, and improved or did not decrease truth-preservation in 86% of cases, we believe that the new generation technique has increased text quality overall considerably over our previous primitive method, while achieving a very strong compression of 33%.

4.3 Content selection evaluation

We evaluate the quality of content selection using four ROUGE metrics (ROUGE-1, 2, L, and SU4) on the IELTS corpus, and compute the statistical significance using the paired Wilcoxon test (Table 2). Although OurExt is the best system in all metrics, OurAbs performs comparably to it on ROUGE-1 and ROUGE-L. It performs at least as well as the four pipeline systems. This is an achievement because the goal of OurAbs is to generate only the summary propositions, and is therefore only directly comparable to OurTok. It is an improvement over OurTok, which is significantly worse than OurExt in all metrics except ROUGE-L.

To a limited degree, we can compare OurAbs to the pipeline systems that combine sentence extraction with compression, as long as we bear in mind that OurAbs faces a more challenging generation task: it is required to capture specific information with a sentence, instead of what is generally important in that sentence. It performs this task very well, which is shown by a very low compression rate (length of generation divided by length of original, in words) of 33%, and easily beats either sentence extractor combined with Cohn and Lapata's compressor, which compresses to 43%. Under ROUGE, when the compression rate decreases, the shorter sentences give rise to much fewer n-grams, so that the generation imperfections become disproportionately penalised by ROUGE's numerical scores. We see the extreme case in OurExt, which has the highest ROUGE scores but does not compress at all. Even under these adverse conditions, OurAbs still performs indistinguishably from any sentence extractor with Clarke and Lapata's compressor, which, at a compression rate of 67%, benefits under ROUGE from being closer to full-sentence extractions.

5 Conclusions

What we have presented here are the first steps in abstractive summarisation via propositions. What we hope we have shown is that there is promise in using propositions as a functional semantic representation, which works for both content selection and verbalisation of the output. Relying only on text coherence for the memory operations, this content selection process has the flexibility to work across domains.

Our new KvD-based abstractive summariser uses propositions not only for content selection, but also for generation. For this we use a deep generator based on the DMRS formalism. Others have used proposition-like units for selection and generation, but they use a much shallower NL generator and they limit the units either to certain syntactic constructions (Genest and Lapalme, 2011) or to certain domains (Genest and Lapalme, 2013). The addition of the NL generator is of course an important step forward for the usability of our summariser, but it is also an improvement for evaluation, because it allows us to measure the effect of granularity in content selection, without being tied to sentence extraction as our generation method, as we were before. In the future, we also plan to concentrate our efforts on developing a fully automatic proposition-based evaluation.

Our current system produces summary output of a higher quality than our previous primitive output method. For a more extensive comparison, we have constructed the combinations of sentence extractors (both ours and LexRank) with two well-cited sentence compressors. Sentence compression presupposes a prior step of sentence extraction; it is the only other way how abstractive summaries can be constructed. However, this previous step is rarely talked about in the literature. In this work, we have implemented four versions of this extraction–compression model, and found that our system performs at least as well as these systems. But its biggest advantage is that it has better information packaging abilities, as demonstrated by the lower compression rate. This is arguably an effect of our use of propositions.

There are also many ways how the generation component could be made more robust. Despite ERG's high quality, parsing and generation sometimes still produces ungrammatical or awkward text, or fails altogether, in which case our system has to fall back to the crude token-based output method. The main reasons for this is that our system is still not very deep, as there are many cases when neither the KvD step nor the generation step has enough information to select the correct information.

We see the fact that our summariser is not based on sentence extraction for content selection as one of its advantages. However, the generation step itself currently has the limitation that the textual output representing a proposition can only come from a single sentence. This leads to an abundance of bullet point-like short sentences in our automatic summaries, which also makes our abstractive summaries disadvantaged in the ROUGE-based evaluation. Removing this limitation would allow a truly flexible summariser. This could be achieved by knowing the identity of concepts in a document, and aggregating attributes of an entity (i.e. propositions about an entity). As a subproblem of this, we would need to generate appropriate referring expressions, or create discourse-new elements as needed.

The KvD summariser itself is also constantly being developed further. For instance, keeping a distinction between identity and bridging links between arguments is important to the above-mentioned task. On top of this, the recognition of identical propositions and generalised propositions that already exist in text can also be achieved, which will be an initial step towards inference.

References

Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Trevor Cohn and Mirella Lapata. 2007. Large margin synchronous generation and its application to sentence compression. In *EMNLP-CoNLL*, pages 73–82.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 137–144. Association for Computational Linguistics.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(2):281–332.

Ann Copestake, Guy Emerson, Michael Wayne Goodman, Matic Horvat, Alexander Kuhnle, and Ewa Muszyńska. 2016. Resources for building applications with Dependency Minimal Recursion Semantics. In *Proceedings of the Tenth Language Resources and Evaluation Conference (LREC '16)*.

Ann Copestake. 2009. Slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 1–9, Athens, Greece, March. Association for Computational Linguistics.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Yimai Fang and Simone Teufel. 2014. A summariser based on human memory limitations and lexical competition. In *EACL*, pages 732–741.

Yimai Fang and Simone Teufel. 2016. Improving argument overlap for proposition-based summarisation. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 479.

Dan Flickinger, Emily M. Bender, and Stephan Oepen. 2014. Towards an encyclopedia of compositional semantics. Documenting the interface of the English Resource Grammar. In *Proceedings of the Ninth Language Resources and Evaluation Conference (LREC '14)*, pages 875–881, Reykjavik, Iceland.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 64–73. Association for Computational Linguistics.

Pierre-Etienne Genest and Guy Lapalme. 2013. Absum: a knowledge-based abstractive summarizer. *Génération de résumés par abstraction*, 25.

Frank K Hwang, Dana S Richards, and Pawel Winter. 1992. *The Steiner tree problem*, volume 53. Elsevier.

Walter Kintsch and Teun A. van Dijk. 1978. Toward a model of text comprehension and production. *Psychological review*, 85(5):363–394.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization – step one: Sentence compression. *AAAI/IAAI*, 2000:703–710.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

André FT Martins and Noah A Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 1–9. Association for Computational Linguistics.

Ryan T McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *EACL*.

Hitoshi Nishikawa, Kazuho Arita, Katsumi Tanaka, Tsutomu Hirao, Toshiro Makino, and Yoshihiro Matsuo. 2014. Learning to generate coherent summary with discriminative hidden semi-markov model. In *COLING*, pages 1648–1659.

Franz Josef Och, Christoph Tillmann, Hermann Ney, et al. 1999. Improved alignment models for statistical machine translation. In *Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Lucy Vanderwende, Michele Banko, and Arul Menezes. 2004. Event-centric summary generation. *Working notes of DUC*, pages 127–132.

Katsumasa Yoshikawa, Tsutomu Hirao, Ryu Iida, and Manabu Okumura. 2012. Sentence compression with semantic role constraints. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 349–353, Stroudsburg, PA, USA. Association for Computational Linguistics.

Cross-lingual Learning of an Open-domain Semantic Parser

Kilian Evang
University of Groningen
The Netherlands
k.evang@rug.nl

Johan Bos
University of Groningen
The Netherlands
johan.bos@rug.nl

Abstract

We propose a method for learning semantic CCG parsers by projecting annotations via a parallel corpus. The method opens an avenue towards cheaply creating multilingual semantic parsers mapping open-domain text to formal meaning representations. A first cross-lingually learned Dutch (from English) semantic parser obtains f-scores ranging from 42.99% to 69.22% depending on the level of label informativity taken into account, compared to 58.40% to 78.88% for the underlying source-language system. These are promising numbers compared to state-of-the-art semantic parsing in open domains.

1 Introduction

Scarceness of manually annotated corpora for training dependency parsers has led researchers to explore more indirect forms of supervision, such as cross-lingual learning, where annotations in one language are used in training a system for another language. Semantic parsers, which map sentences directly to logically interpretable meaning representations, equally suffer from a lack of annotated training corpora, despite recent efforts like the Groningen Meaning Bank (Basile et al., 2012) or AMR Bank (Banarescu et al., 2013). The lack is especially pronounced for languages other than English.

This paper aims to show that cross-lingual learning can help create semantic parsers for new languages with little knowledge about those languages and minimal human intervention. We present a method that takes an existing (source-language) semantic parser and parallel data and learns a semantic parser for the target language. Our method is in principle applicable to all parsers producing interpreted derivations (i.e., parse trees) of Combinatory Categorial Grammar (CCG; Steedman 2001) . It is independent of the concrete meaning representation formalism used, as long as meaning representations are assembled in the standard CCG way using the lambda calculus. We evaluate our method by applying it to English as source language, Dutch as target language and Discourse Representation Theory (DRT; Kamp and Reyle 1993) as meaning representation formalism, and measuring the performance of the obtained Dutch semantic parser.

2 Related Work

Semantic parsing has been tackled from a wide variety of angles. Systems that add a semantic interpretation component to an existing supervised syntactic parser (Curran et al., 2007; Le and Zuidema, 2012; Lewis and Steedman, 2013) have wide coverage but require much syntactically annotated training data. Other approaches are restricted to relatively narrow linguistic domains but manage to do without strong syntactic supervision. Forms of supervision used include sentence/meaning representation pairs (Wong and Mooney, 2007; Zettlemoyer and Collins, 2007) and even weaker forms of supervision (Clarke et al., 2010; Liang et al., 2011; Kwiatkowski et al., 2013; Goldwasser and Roth, 2011; Chen and Mooney, 2011; Krishnamurthy and Mitchell, 2012; Reddy et al., 2014; Artzi and Zettlemoyer, 2011; Poon, 2013). Only recently have approaches not relying on explicit syntactic supervision successfully been applied to

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 579–588, Osaka, Japan, December 11-17 2016.

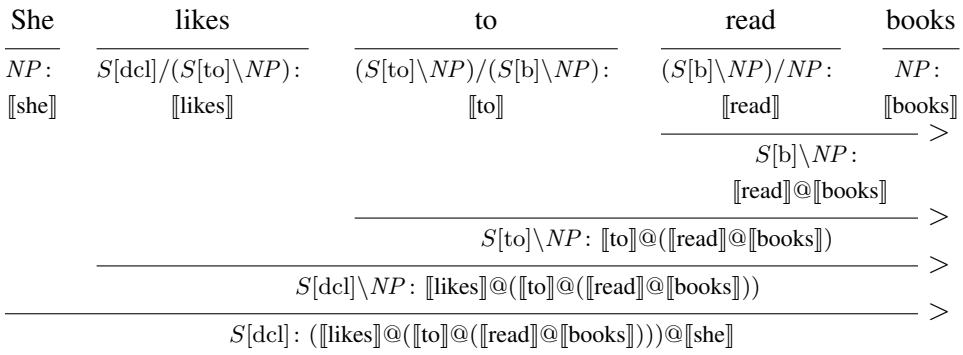


Figure 1: Example CCG derivation.

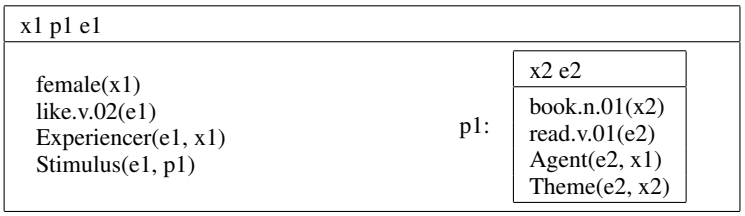


Figure 2: Example DRS for the sentence in Figure 1.

more open-domain sentences (Vanderwende et al., 2015; Artzi et al., 2015). Ours is, to the best of our knowledge, the first such work using cross-lingual learning.

Cross-lingual learning has previously been applied to different NLP tasks, notably part-of-speech tagging and dependency parsing. For dependency parsing, the task most similar to ours, three families of approaches can be distinguished. In **annotation projection**, existing annotations of source-language text are automatically projected to target-language translations in a parallel corpus; the result is used to train a target-language system (Hwa et al., 2005; Tiedemann, 2014; Rasooli and Collins, 2015; Johannsen et al., 2016; Agić et al., 2016). In **model transfer**, parsers for different languages share some of their model parameters, thereby using information from annotations in multiple languages at the same time. (Zeman and Resnik, 2008; Ganchev et al., 2009; McDonald et al., 2011; Naseem et al., 2012; Täckström et al., 2013). The **translation approach** pioneered by Tiedemann et al. (2014) is similar to annotation projection, but instead of relying on existing translations, it automatically translates the data and synchronously projects annotations to the translation result. Our approach falls within the annotation projection family, with the new challenge that entire CCG derivations with logical interpretations need to be transferred.

3 Combinatory Categorial Grammar

Combinatory Categorial Grammar (Steedman, 2001) is a grammar formalism widely used for semantic parsing due to its suitability to statistical parsing (Clark and Curran, 2007) and its transparent syntax-semantics interface. Every constituent has a *category*—either a basic one (S for sentence, N for noun, NP for noun phrase, PP for prepositional argument) or a functional one such as $S\backslash NP$ for verb phrase, indicating that a constituent can combine with a noun phrase to its left to yield a sentence. Smaller constituents are combined into larger ones according to a handful of combinatorial rules such as application and composition. Every constituent also has a semantics, its *interpretation*, which is a term of the lambda calculus. Crucially, the combinatorial rules specify precisely how the interpretation of each non-lexical constituent is computed from the interpretations of the constituents that combine to form it. An example derivation (CCG parse tree) for an English sentence is shown in Figure 1.

The lambda calculus and thus CCG is applicable to any kind of meaning representation, as long as it can be constructed compositionally. In this paper, we use a flavor of Discourse Representation Theory (DRT; Kamp and Reyle 1993) and a corresponding semantic lexicon introduced by Bos (2009) which

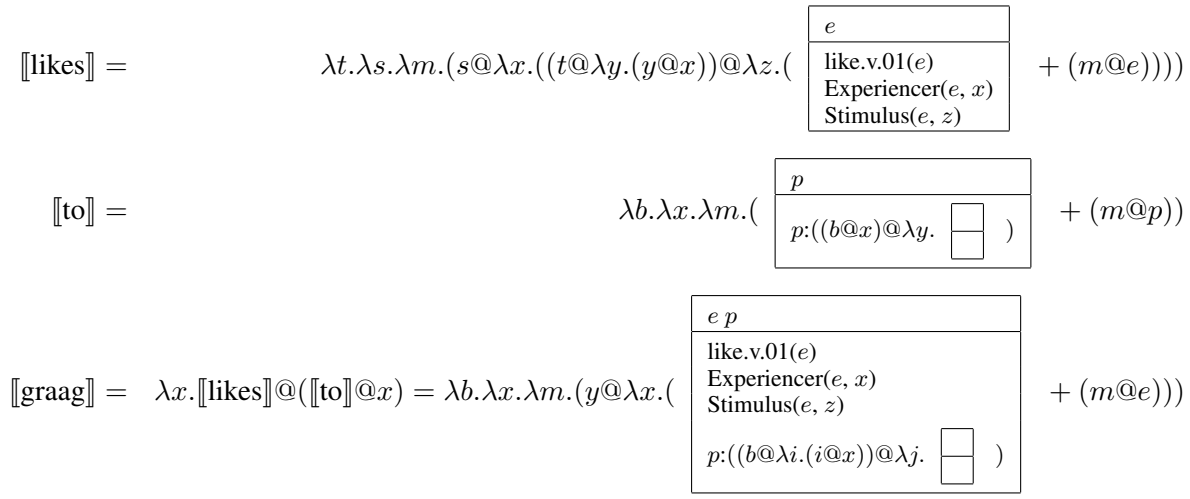


Figure 3: Examples of some lexical interpretations (two English, one Dutch).

uses a neo-Davidsonian event semantics with VerbNet/LIRICS semantic roles (Bonial et al., 2011) and WordNet 3.0 senses (Fellbaum, 1998). The meaning representation (discourse representation structure; DRS) for our example sentence is shown in Figure 2. Lexical interpretations of some words are shown in Figure 3.

4 Method

We start with a parallel corpus of sentence pairs whose source-language part has been annotated with semantic CCG derivations as in Figure 1 by the source-language system. We use this annotation in two ways: first, to induce a target-language lexicon in a first step called **category projection**. Secondly, we use it as a form of indirect supervision: we assume that the source-language system works mostly correctly, and that if two sentences are translations of each other, they should have the same interpretation—thus we can train the target-language parser to produce the same interpretations as the source-language parser. To this end, we try to find target-language derivations resulting in the same interpretations as the source-language ones, based on the target-language candidate lexical items found in category projection. We call this second step **derivation projection**. The derivations thus found are used to train a statistical parsing model for the target language. We call this third step **parser learning**.

All three steps make use of a shift-reduce CCG parser similar to that of Zhang and Clark (2011). Parse actions are SHIFT-C-I, COMBINE-C, UNARY-C (where C is the category placed on top of the stack by shifting or applying a binary/unary rule and I is the interpretation of the (multi)word placed on top of the stack by shifting), SKIP for skipping words as semantically empty, FINISH for marking a parse as complete and IDLE for keeping complete parses on the agenda while others are still incomplete (Zhu et al., 2013).

We now describe the three steps in more detail.

4.1 Step 1: Category Projection

Category projection assigns candidate categories and interpretations to target-language (multi)words in the training data. It thereby also induces the target-language lexicon that we use in subsequent steps. It serves as a cross-lingual alternative to the two traditional main strategies of inducing CCG lexicons for semantic parsing, namely hand-written, language-specific lexical templates (Zettlemoyer and Collins, 2005) and higher-order unification constrained by search heuristics (Kwiatkowksi et al., 2010).

Category projection first word-aligns the training corpus—we use the n best alignments found by GIZA++ (Och and Ney, 2003) with default settings. The result is a large, noisy set of translation units. From each contiguous translation unit, we try to induce a candidate lexical item. Figure 4 shows an

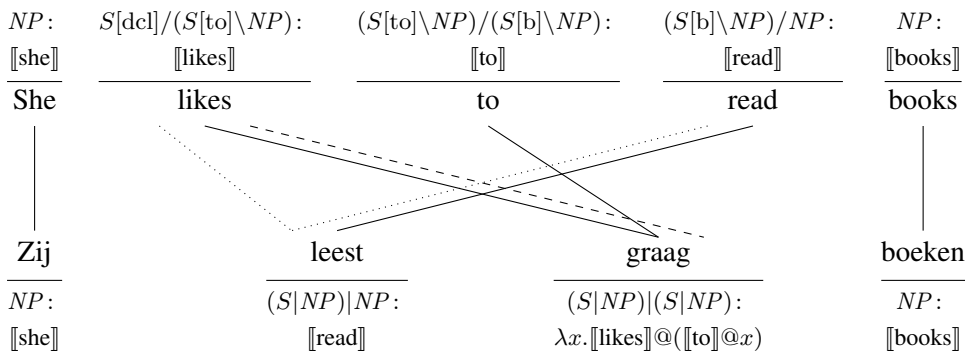


Figure 4: Category projection: word alignments induce candidate categories and interpretations for target-language words.

example sentence pair: at the top there are the lexical items from the English derivation in Figure 1, each with a CCG category and interpretation. The interpretations are here represented using the $[\![\cdot]\!]$ notation but are actually a complex λ-terms with DRT-based meaning representations (for short: a λ-DRS) as exemplified in Figure 3. The lines in the center represent possible word alignments, with correct translation units drawn as solid lines and incorrect ones as dashed or dotted ones. At the bottom there is the Dutch sentence with induced candidate lexical items. For the sake of the example, we only show one candidate lexical item per word, those induced from the correct translation units.

Inducing a candidate lexical item from a translation unit works as follows: if the translation unit contains only one source-language word, it provides the category and interpretation for the (multi)word on the target side, as is the case for *She*/*Zij*, *read*/*leest* and *books*/*boeken*. Since slash directions are language-specific, we change all categories to have vertical slashes, which can apply in either direction. We also remove English-specific category features such as $[b]$ and $[dcl]$, distinguishing bare and declarative verb phrases. For example, $(S[\mathrm{b}]\backslash NP)/NP$ becomes $(S|NP)|NP$.

If the translation unit contains more than one source-language word, this string is parsed using CCG combinatory rules, and if successful, the resulting category and interpretation are used as a lexical item for the word on the target side. For example, the verb *likes* and the particle *to* combine via forward composition into one category and interpretation for the Dutch adverb *graag*, which expresses the same meaning in a syntactically different way. The full resulting λ-DRS for *graag* is shown in Figure 3.

Unaligned target words get a special category $skip$.

The final target-language lexicon $\mathcal{L}$ contains the lexical items thus induced, but only those that are at least a cutoff factor c times as frequent as the most frequent candidate for this target-language word/part-of-speech combination.

4.2 Step 2: Derivation Projection

The lexical items assigned to target-language words in category projection give rise to a space of possible CCG derivations. The space is large and noisy, partly because of the pervasive syntactic ambiguity of natural language, partly because we use more than one word alignment in category projection. In derivation projection, the task is to filter out only the "correct" derivations so we can then train on these. We regard as "correct" any derivation that results in the same interpretation for the whole sentence as the source-language derivation.

For finding the "correct" derivations, we use the method of Zhao and Huang (2015) of running the parser in forced decoding mode: we use a beam of unlimited width but prune away parse items where, based on their interpretations, we can rule out that they could lead to a "correct" derivation. For instance, in our example, an item with interpretation ⟦read⟧@⟦she⟧ would be pruned because it cannot be part of (⟦likes⟧@(⟦to⟧@(⟦read⟧@⟦books⟧)))@⟦she⟧. To make this check tractable, we treat English lexical interpretations such as ⟦read⟧ as atomic.

The forced decoding uses a local lexicon, using only lexical items induced from the same sentence pair.

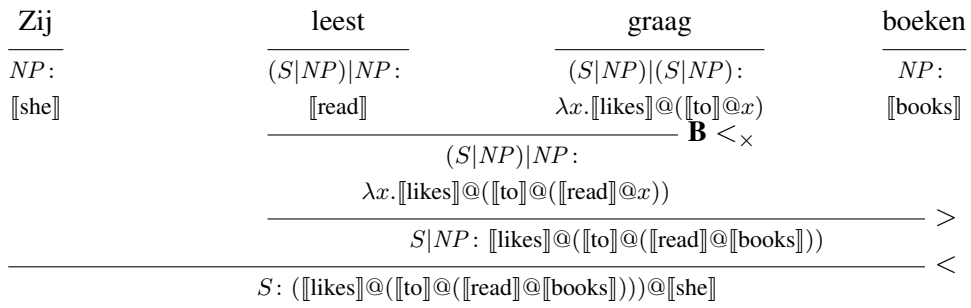


Figure 5: Derivation projection: combinatory rules are applied to find a derivation with the same interpretation as the source-language sentence.

The combinatory rule instances used are extracted from all English training derivations, but to allow for different word orders, we verticalize all slashes and for binary rule instances add mirror-image versions, e.g., the backward application instance $NP\ S\backslash NP \Rightarrow S$ generates $NP\ S|NP \Rightarrow S$ and $S|NP\ NP \Rightarrow S$.

If we cannot find any "correct" derivation, this means we did not get the word alignments inducing the lexical items needed to find one. This can be due to translations being idiomatic, loose or informative (Bos, 2014). In such cases, our assumption that the interpretation for source and target sentence should be the same breaks down, and we would not want to use this training example anyway. In this sense, derivation projection also has the function of cleaning the training set.

Despite the pruning, for some sentences the search space is prohibitively large, so we restrict the size of the parser agenda to 256, a number that still allows us to run this step in reasonable time. If this limit is exceeded or if we do not find a complete derivation with the target interpretation, we discard the sentence. If we do find one or more—such as the one in Figure 5—the sentence becomes part of the training data for the following step.

4.3 Step 3: Parser Learning

For statistical parsing, we use an averaged perceptron with a hash kernel (Bohnet, 2010) and the same feature templates as Zhang and Clark (2011), plus, for shift actions, a feature uniquely identifying a lexical item including the (multi)word, its part(s) of speech and the chosen category and interpretation. The parser uses the full global lexicon $\mathcal{L}$. The same grammar as in derivation projection is used.

The parser uses beam search. If at some point during training on one example there is no item on the beam anymore that could lead to one of the "correct" derivations found in derivation projection, the parser aborts training on this example and performs an early perceptron update (Collins and Roark, 2004).

5 Experimental Setup

To ensure that derivation projection can find a large number of high-quality derivations, we need training data with a large proportion of "literally" translated sentences. By this we do not mean that the translation has to be syntactically isomorphic—our projection approach can actually deal with a wide range of such syntactic divergences (cf. Dorr, 1993), such as the *likes to*/*graag* example. But translations should not be informative or loose, as this changes their meaning. More literal translations than in freely occurring text can be found in resources aimed at human language learners (who are faced with a similar task as our system: learning to understand an unknown language, helped by example sentences translated into a familiar one). One such resource is `tatoeba.org`, based on the Tanaka corpus (Tanaka, 2001). We used 13,122 English-Dutch sentence pairs from Tatoeba as training data, 1,639 for development and 1,641 as final test set, of which a random sample of 150 sentences was manually annotated to serve as a gold standard.

In preliminary experiments, we tried out different values for n where we use the n-best alignments per direction to extract candidate lexical item from. Too low and derivation projection may not find a derivation for many target-language sentences for lack of needed candidate lexical entries. Too high and the agenda gets polluted with spurious parse items, and derivation projection aborts due to the agenda limit. We found that for our training set, the percentage of examples for which we find at least one target-language derivation peaked at $n = 3$ with 57.7%.

The source language system whose output we use for supervision is the C&C/Boxer system (Curran et al., 2007), which takes English sentences and produces discourse representation structures. We use SVN repository version 2444, giving the options `--modal true --nn true --roles verbnet` to Boxer and making some minor modifications to its code to better match our annotation scheme for adjectives, adverbs, semantic roles and modals.

For statistical parsing, we initialize all model weights to 0 and use a beam width of 16. The Dutch sentences are POS-tagged using TreeTagger (Schmid, 1995). For decoding, the input is only POS-tagged Dutch development/test sentences. We use the same lexicon as for training, but to deal with unseen content words, an abstract version of each lexical entry is created where the synset ID in its λ-DRS is replaced by the $\_\_UNKNOWN\_\_$ symbol. The parser then selects between all categories occurring with the POS tag, with the most common abstract interpretation for each category. The output is a CCG derivation—or, since the parser can fall back to fragmentary output, a sequence thereof—each of which has a DRS interpretation.

6 Evaluation Setup

For evaluation, we follow the approach proposed by Allen et al. (2008): meaning representations are converted to graphs and an alignment between system output and gold graph vertices is found that maximizes the number of labeled edges in a maximum common subgraph. An instantiation of this evaluation metric for Abstract Meaning Representations, SMATCH (Cai and Knight, 2013), is now commonly used. We use the instantiation for DRSs that was first introduced by Le and Zuidema (2012).

We first measure how closely the output of our system for Dutch resembles that of C&C/Boxer for English on the development/testing portion of our parallel corpus. This gives an idea of how well our system has learned to imitate the existing system, but has two problems: first, it does not say much about the quality of the output because that of C&C/Boxer is not free from errors, it is not a gold standard. Secondly, the data contains idiomatic, informative and loose translations, in which case we *want* the outputs of both systems to differ.

Therefore, we also measure how closely the outputs of C&C/Boxer and our system resemble a gold standard of 150 sentence/DRS pairs from the testing portion, for their respective input languages. Since DRSs are complex structures not easily created in completely manual annotation, we resorted to hand-correcting automatically produced ones to obtain the gold standard. This was done as follows: Two annotators independently corrected 50 DRSs produced by C&C/Boxer so that the DRSs represented the meaning of the Dutch annotations. Inter-annotator agreement at this point as measured by the evaluation metric was 67%. Instances of disagreement were identified, with 29% related to WordNet senses, 22% to semantic roles, 16% to other relations such as prepositional ones, 13% to the rendering of Dutch idioms using English WordNet senses, 9% to modal and logical operators such as implication and negation, and 11% to other structural issues such as nested DRSs. In an adjudication phase, both annotators resolved the differences together and agreed on a common gold standard. A single annotator then corrected another 100 Boxer DRSs, which were then checked by the other annotator, and differences were again resolved through discussion. One annotator finally created an adapted version of all 150 DRSs where in case of non-literal translations, the annotation matches the English rather than Dutch sentence.

No comparable systems for Dutch as input language and DRS as meaning representation language exist yet. To demonstrate the effect of learning the parsing model, we picked a simple baseline that assigns each target-language word the semantic representation most frequently associated with aligned English words and outputs the resulting, very fragmented graph.

Table 1: Development set (non-gold-standard) f-score depending on lexical cutoff factor c and training iterations T (i.e., number of passes over the entire training data).

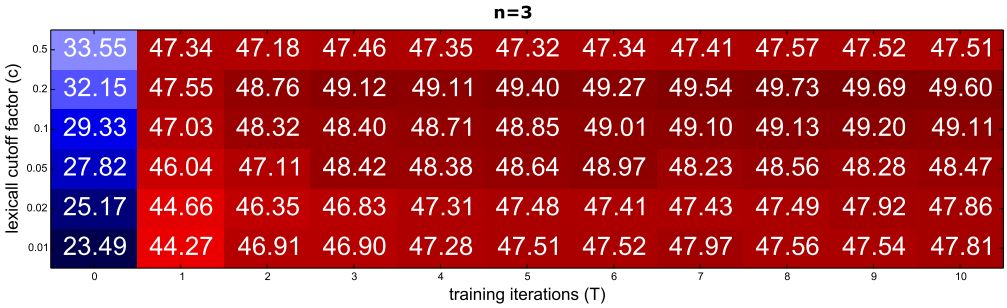


Table 2: Gold-standard match f-score for Boxer, our baseline and our best cross-lingually trained model.

Language	English	Dutch	
System	C&C/Boxer	Baseline	Our system
Full	58.40	26.71	42.99
Ignoring WordNet senses	69.06	36.67	60.23
Ignoring VerbNet/LIRICS roles	64.51	27.57	47.82
Ignoring other relation labels	59.18	27.57	43.39
Ignoring all	78.88	39.04	69.22

7 Results and Discussion

Table 1 shows how the f-score of our system on the (non-gold-stard because automatically annotated) development corpus varies as a function of the lexical cutoff factor c and number of training iterations T. We used the model with the highest score ($c = 0.1$, $T = 10$) for final testing. Table 2 shows the results, comparing the performance of our cross-lingually learned system on Dutch against the baseline and against C&C/Boxer's performance on the English versions of the same sentences.

C&C/Boxer obtains an f-score of 58.40% on the gold standard. Although the data and the formalism are not directly comparable, we note that this f-score is close to those of current state-of-the-art open-domain semantic parsers for English, e.g. those that participated in the recent Abstract Meaning Representation shared task (May, 2016). A large part of the errors comes from misidentifying word senses and semantic roles. "Sloppy" evaluations in which we treat all word senses, all roles and/or other (e.g. prepositional) relation labels as equal give markedly higher f-scores of up to 78.88%.

Our system for Dutch scores around 15% lower than the source-language system under the strict evaluation, at 42.99%. The gap narrows to around 10% under the sloppy evaluation, scoring 69.22%. The gap is expected for a number of reasons. For one, the English system has the advantage of a strong syntactic parser which was trained on a far larger number of sentences, which also had explicit syntactic annotation. The especially large gap under the strict evaluation can partially be explained by many unseen words in the test data, with the training data insufficiently large to learn a wide-coverage lexicon, while the system for English has access to the full WordNet lexicon.

For languages like Dutch, available resources could be exploited to address these problems. For example, one could improve a cross-lingually bootstrapped CCG parser by training it to recover the dependencies in a dependency treebank, e.g. Universal Dependencies (Nivre et al., 2016). Multilingual lexical databases like Open Multilingual Wordnet (Bond and Foster, 2013) could be exploited to attack unseen words. For truly low-resource languages where such resources are not available, parallel data could be mined in order to extend the target-language lexicon. This could work even with data that is currently too loosely translated or too syntactically complex to work well with our projection approach.

Although we optimized the hyperparameter $n = 3$ for the number of successfully projected derivations, Dutch derivations were found for only 58.35% of our training sentences, considerably reducing the amount of training data that is available for parser learning. To what extent this is due to non-literal translations being weeded out (cf. Section **??**), and to what extent failing derivation projections could be avoided (e.g. by considering other combinatory rules than those extracted from the English data) is an important question for future work.

8 Conclusions

Semantic parsing for open domains is a young and very dynamic research area that may shortly enable computers to make use of natural language on a new and significantly deeper level. With a field notoriously focused on English, how can other languages keep up with the developments?

In this paper, we have shown a possible avenue. We draw upon CCG's flexible notion of constituency and the language-independent nature of its combinatory rules to develop a lexicon induction technique that overcomes certain translation divergences between languages. We have then used cross-lingual supervision to train a semantic parser for Dutch at a far lower cost than the original English one, considering the cost of manually creating explicit syntactic annotation and a semantic lexicon.

Bridging the gap between source and target language does come at an additional cost in performance. However, there are a number of possible ways to attack this gap in future work, including using target-language lexical resources if available, unsupervised mining of large amounts of parallel data for lexical entries, and also improving the parsing model itself with recent advances in CCG semantic parsing.

Dutch and English are relatively close cousins; in ongoing work we are investigating the applicability of our method to a number of Germanic and Romance languages (e.g., German and Italian) and so far have found no theoretical obstacles. To what extent applying it to less closely related language pairs than English/Dutch is harder empirically remains to be investigated. In any case, we are confident that the techniques presented in this paper can help develop multilingual semantic parsers without starting from scratch, software-wise and data-wise, for every new language.

Acknowledgments

We would like to thank Phong Le for sharing his evaluation code, Johannes Bjerva, Hessel Haagsma, Dieke Oele and Rob van der Goot for feedback on earlier version of this paper, and the three anonymous reviewers for helpful comments. This work was partially funded by the NWO-VICI grant "Lost in Translation—Found in Meaning" (288-89-003).

References

Ž. Agić, A. Johannsen, B. Plank, H. Martínez Alonso, N. Schluter, and A. Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *TACL*, 4:301–312.

J.F. Allen, M. Swift, and W. de Beaumont. 2008. Deep Semantic Analysis of Text. In J. Bos and R. Delmonte, editors, *Semantics in Text Processing*. College Publications.

Y. Artzi and L. Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of EMNLP*.

Y. Artzi, K. Lee, and L. Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of EMNLP*.

L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of LAW VII & ID*.

V. Basile, J. Bos, K. Evang, and N. J. Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proceedings of LREC*.

B. Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of Coling*.

F. Bond and R. Foster. 2013. Linking and extending and open multilingual wordnet. In *Proceedings of ACL*.

C. Bonial, W. J. Corvey, M. Palmer, V. Petukhova, and H. Bunt. 2011. A hierarchical unification of LIRICS and VerbNet semantic roles. In *Proceedings of ICSC*.

J. Bos. 2009. Towards a large-scale formal semantic lexicon for text processing. In *Proceedings of GSCL*.

J. Bos. 2014. Semantic annotation issues in parallel meaning banking. In *Proceedings of ISA-10*.

S. Cai and K. Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria, August. Association for Computational Linguistics.

D. L. Chen and R. J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of AAAI*.

S. Clark and J. R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of CoNLL*.

M. Collins and B. Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*.

J. Curran, S. Clark, and J. Bos. 2007. Linguistically Motivated Large-Scale NLP with C&C and Boxer. In *Proceedings of ACL: Demo and Poster Sessions*.

B. J. Dorr. 1993. *Machine Translation: A View from the Lexicon*. The MIT Press.

C. Fellbaum, editor. 1998. *WordNet. An Electronic Lexical Database*. The MIT Press.

K. Ganchev, J. Gillenwater, and B. Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of ACL-IJCNLP*.

D. Goldwasser and D. Roth. 2011. Learning from natural instructions. In *Proceedings of IJCAI*.

R. Hwa, P. Resnik, A. Weinberg, C. Cabezas, and O. Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(03):311–325.

A. Johannsen, . Agi, and A. Sgaard. 2016. Joint part-of-speech and dependency projection from multiple sources. In *Proceedings of ACL*.

H. Kamp and U. Reyle. 1993. *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Kluwer.

J. Krishnamurthy and T. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of EMNLP-CoNLL*.

T. Kwiatkowksi, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233. Association for Computational Linguistics.

T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of EMNLP*.

P. Le and W. Zuidema. 2012. Learning compositional semantics for open domain semantic parsing. In *Proceedings of COLING*.

M. Lewis and M. Steedman. 2013. Combined distributional and logical semantics. *TACL*, 1:179–192.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of ACL*.

J. May. 2016. Semeval-2016 task 8: Meaning representation parsing. In *Proceedings of SemEval*, pages 1063–1073.

R. McDonald, S. Petrov, and K. Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of EMNLP*.

T. Naseem, R. Barzilay, and A.l Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of ACL*.

J. Nivre, M. de Marneffe, F. Ginter, Y. Goldberg, J. Haji, C.D. Manning, R. McDonald, S. Petrov, S. Pyysalo, N. Silveira, R. Tsarfaty, and D. Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of LREC*.

F.J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

H. Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of ACL*.

M.S. Rasooli and M. Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of EMNLP*.

S. Reddy, M. Lapata, and M. Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *TACL*, 2(1):377–392.

H. Schmid. 1995. Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT-Workshop*.

M. Steedman. 2001. *The Syntactic Process*. The MIT Press.

O. Täckström, R. McDonald, and J. Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of NAACL-HLT*.

Y. Tanaka. 2001. Compilation of a multilingual parallel corpus. In *Proceedings of PACLING*.

J. Tiedemann, Ž. Agić, and J. Nivre. 2014. Treebank translation for cross-lingual parser induction. In *Proceedings of CoNLL*.

J. Tiedemann. 2014. Rediscovering annotation projection for cross-lingual parser induction. In *Proceedings of COLING: Technical Papers*.

L. Vanderwende, A. Menezes, and C. Quirk. 2015. An AMR parser for English, French, German, Spanish and Japanese and a new AMR-annotated corpus. In *Proceedings of NAACL-HLT*.

W. Wong and R. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of ACL*.

D. Zeman and P. Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP Workshop on NLP for Less Privileged Languages*.

L. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 658–666, Arlington, Virginia. AUAI Press.

L. S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of EMNLP-CoNLL*.

Y. Zhang and S. Clark. 2011. Shift-reduce CCG parsing. In *Proceedings of ACL*.

K. Zhao and L. Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *Proceedings of NAACL-HLT*.

M. Zhu, Y. Zhang, W. Chen, M. Zhang, and J. Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of ACL*.

A subtree-based factorization of dependency parsing

Qiuye Zhao[1] and **Qun Liu**[2,1]
1. Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences
2. ADAPT Centre, School of Computing, Dublin City University

Abstract

We propose a dependency parsing pipeline, in which the parsing of long-distance projections and localized dependencies are explicitly decomposed at the input level. A chosen baseline dependency parsing model performs only on 'carved' sequences at the second stage, which are transformed from coarse constituent parsing outputs at the first stage. When k-best constituent parsing outputs are kept, a third-stage is required to search for an optimal combination of the overlapped dependency subtrees. In this sense, our dependency model is subtree-factored. We explore alternative approaches for scoring subtrees, including feature-based models as well as continuous representations. The search for optimal subset to combine is formulated as an ILP problem. This framework especially benefits the models poor on long sentences, generally improving baselines by 0.75-1.28 (UAS) on English, achieving comparable performance with high-order models but faster. For Chinese, the most notable increase is as high as 3.63 (UAS) when the proposed framework is applied to first-order parsing models.

1 Introduction

Incorporating 'non-local' features into syntactic parsing has been well-studied in literature. For exact parsing, we have seen cubic-time decoders for first and second-order models (Eisner, 1996; McDonald and Pereira, 2006), quadratic-time decoders for third-order models (Koo and Collins, 2010) and etc. In transition-based parsers, e.g. (Nivre et al., 2006), so-called non-local features can be easily extracted from parsing history. With a global inference framework, e.g. approximate Linear Programming (Martins et al., 2011; Koo et al., 2010), arbitrary structural constraints can be imposed. There are two main research goals underlying these works, one is to localize long-distance dependencies, which can be achieved by assuming the optimal substructure property or introducing parsing actions like reductions. The other goal is to appropriately factor tree score, over parts like arcs as well as over parsing actions.

In this work, we propose a dependency parsing pipeline, so that long-distance dependencies are explicitly localized at the input level. With the proposed factorization, dependency tree score sums over its subtrees. More specifically, we address a distinction between long-distance projections and localized dependencies, which can be characterized by word categories of their lexical heads. The long-distance projections can be captured by a coarse constituent parser, which only sees peripheral and head words of such long-distance projections. So as to reduce error propagation, we transform k-best constituent parsing outputs to 'carved' sequences for the following dependency parsing, which could be overlapped. Therefore, a third stage is required to search for the optimal subset of all candidate subtrees to combine.

Given previous work on parsing, e.g. (Charniak and Johnson, 2005),(McDonald et al., 2005),(Martins et al., 2013) and so on, reliable constituent-based parsers and dependency parsers are available. Our main implementation challenge is to select an subset of those subtrees over overlapped carved sequences of the original input to cover the original sentence. We propose to formulate this as an Integer Linear Programming (ILP) problem, which is similar to a set cover problem. Note that, since the search space at this stage is highly constrained by the input of the previous stages, an exact decoding is efficient enough.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 589–598, Osaka, Japan, December 11-17 2016.

Example 2.2. :

"The SEC will probably vote"
"vote on early next year"
"on the proposal"
"will , he said ."

Example 2.3. :

(VBDC (MDC (DT The) (NNP SEC)
(MD will) (RB probably)
(VBC (VB vote)
(INC (IN on) (DT the) (NN proposal))
(RB early) (JJ next) (NN year)))
(, ,) (PRP he) (VBD said) (. .))

Figure 1: Example 2.2 demonstrates 'carved sequences'. Example 2.3 demonstrates the coarse constituent parsing.

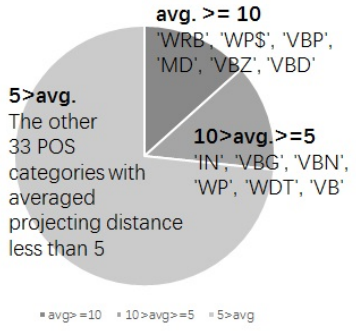


Figure 2: The averaged length of projections headed by each POS category, as computed from Penn WSJ treebank.

We experiment with two alternative approaches for scoring subtrees. A feature-based perceptron classifier, and Dependency-based Convolutional Neural Networks (DCNN) (Ma et al., 2015). Previous use of continuous representation in parsing mainly models parsing actions, phrases, words, or features. When dependency chains are modeled, they were only used for reranking of k-best lists of whole dependency trees, e.g. (Le and Zuidema, 2014; Zhu et al., 2015). Since transition-based parsers are known to act worse on long-distance dependencies, they benefit from this pipeline the most. With no algorithmic change to transition-based or first-order models, an increase of 1.28/1.21 (UAS) can be achieved solely due to the proposed factorization of input, thus achieving comparable performance with high-order models but faster. For Chinese, the most notable increase is as high as 3.63 (UAS), when the proposed framework is applied to first-order parsing models.

2 System Design

2.1 Motivation

Consider a sentence as follows:

Example 2.1. *The SEC will probably vote on the proposal early next year , he said .*

Suppose, magically, we could determine oracle 'carved' sequences of this input, such that all words in a sequence are dependent of some word in the same sequence, except for one head word, e.g. Example 2.2 in Figure 1. Further dependency parsing over these short sequences could be much easier than parsing the original input. This kind of localization is implicitly realized during a dynamic programming parsing process, assuming the optimal substructure property. We propose to explicitly capture this kind of localization by decomposing input space, especially motivated by the following two observations:

- There is a notable distinction in word categories with respect to their averaged projection scope.
- The input to a first-stage constituent-based parsing for long projections can be pruned to contain relevant peripheral and head words only (otherwise, the pipeline is of no practical interest at all.)

First of all, we acquire the distinction in word categories and define long-distance projecting POS categories according to statistics of the training corpus. Based on this statistically verifiable distinction we propose the following parsing pipeline:

(1) Constituent-based parsing for phrases headed by long-distance projecting word categories.
(2) Dependency parsing over carved sequences of input, which are transformed from the coarse constituent parsing outputs of the first stage.
(3) When k-best constituent-based parsing is employed at the first stage, search for an optimal subset of subtrees to combine into a whole dependency tree over the original input.

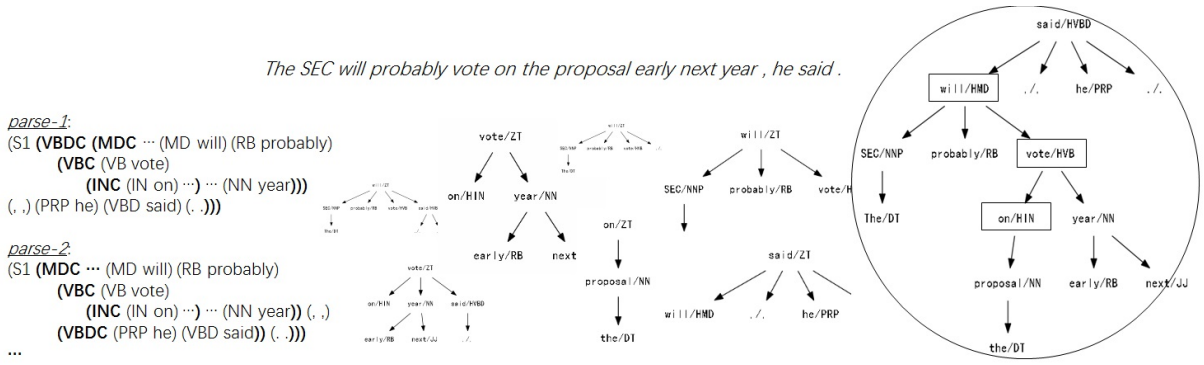


Figure a. First stage: K-best outputs of the coarse constituent-based parsing.

Figure b. Second stage: parse carved sequences into dependency trees. Those vague treelets at background demonstrate redundant subtrees that are not selected to the optimal subset for combination

Figure c. Third stage: combine subtrees by substitution.

Figure 3: A parsing framework of three stages.

Given previous work on parsing, e.g. (Charniak and Johnson, 2005),(McDonald et al., 2005),(Martins et al., 2013) and so on, reliable constituent-based parsers and dependency parsers are available. In other words, other than selecting the set of POS tags that are considered as long-distance projecting categories, there are few implementation details for the first and second stages. Our main implementation challenge seems lies in the third stage. However, since the search space at this stage is highly constrained by the input of the previous stages, we propose to straightforwardly model this search as an ILP problem, which can be efficiently solved by a general ILP decoder, e.g. cplex.

2.2 The first constituent parsing stage

In a projective dependency tree, we consider the projection of a head as the range of input covering all of its descendants. As shown in Figure 2, the POS categories that project longer distance in average is in accordance with our 'linguistic instinct'. Nominal categories, adverbs, adjectives and most closed-class categories, except for subordinating conjunction words and *wh-* holders, tend to project over local words only. And for other categories, we do not stipulate, but leave it as a variant in our experiments.

Given a projective tree, projections of heads can be expressed in brackets, e.g. Example 2.3 in Figure 1. At this stage, we are only interested in projections that are headed by words that fall in long-distance projecting POS categories. Given such a head word of tag 'X', for the sake of expressive convenience, we tag the corresponding bracket covering all its descendants as 'XC'. This simple rule could transform a dependency treebank to brackets of long-distance projections with no ambiguities. Any constituent-based parser that doesn't impose fixed head rules, can be trained over this type of structures, instead of a conventional Treebank. However, if this stage is as slow as a full constituent parsing process, the whole pipeline is of no practical use. As will be shown in Section 5.3, with pruned input and coarse grammar, the constituent parsing stage is no longer a painful bottleneck for parsing speed.

2.3 The second dependency parsing stage

For a bracket, e.g. *(VBC (VB vote) (INC (IN on) (DT the) (NN proposal))(RB early) (JJ next) (NN year))* in Example 2.3 of Figure 1 , we carve all its embedded brackets out, but leaving head words to hold the space. This transformation gives 'carved' sequences such as *"vote on early next year"*. As long as long-distance dependencies do not cross, this simple rule can transform long-distance constituent parsing outputs to 'carved' sequences of the original input without ambiguities. For each oracle sequence, all of its words are guaranteed to be dependant of some word in the same range except for one head word. For example, in Figure 1 brackets in Example 2.3 are transformed to sequences in Example 2.2 by this rule.

Dependency parsing over these carved sequences is much easier, in the sense that both performance and efficiency are dramatically improved. If the first-stage constituent parsing offers reliable 1-best output, we can combine subtrees over these sequences by substituting each word, r, with the subtree rooted at r, if any. For example, in Figure 3-c, a squared node indicates that a subtree is substituted here. We borrow the term 'substitution' from TAG formalism (Joshi and Schabes, 1997). However, when k-

best constituent parsing outputs are kept for reducing error propagation, further search strategy for an optimal subset of these redundant subtrees will be discussed in Section 3.

3 Combination of subtrees

At the first k-best constituent parsing stage, we generate "carved" sequences to be parsed at the second dependency parsing stage. At the third stage, dependency parsing is reduced to the search of an optimal subset to form a legal tree over the original whole input. In this section, we show how to formulate this search as an Integer linear Programming problem, and alternative approaches for scoring subtrees.

3.1 Formulated as an ILP problem

Given a set of subtrees, searching for its optimal subset over the original input can be straightforwardly formulated as an Integer Linear Programming (ILP) problem, which is similar to a set cover problem. The only practical concern is whether it can be solved efficiently by a general ILP decoder. As will be shown with experiments in Section 5.3, the search space of this stage is well-constrained by outputs from the previous stages, therefore this succinct idea nicely works out in the proposed pipeline.

Because subtree outputs from the second stage already satisfy tree requirements, we only need to impose the globally single-root and single-headed constraints on a compatible subset, but not to worry about the non-cyclic requirement. We introduce a designated root \$ for the final dependency tree. For each candidate subtree, add a dummy subtree that includes only a single arc from the designated root \$ to the subtree's root node. Suppose there are n words in the original input sentence, $N = 1...n$, not including the designated root node. Let $\mathbf{T}$ be a set of subtrees, $\mathbf{T} = t_1...t_s$. Our goal is to find an optimal solution $\mathbf{x} = x_1...x_s$ that maximizes $\sum_{i=1}^{s} score(t_i) \cdot x_i$, and subjects to

$$(1) \sum_{i:n \in t_i \text{as non-root node}} x_i = 1,\ \forall n \in N; \qquad (2)\ x_i \in \{0,1\}, 1 \leq i \leq s$$

The boolean value of x_i, as guaranteed by constraint (2), indicates whether t_i is selected in the combination solution or not. Constraint (1) requires every node has one and only one head, and the introduction of the dummy root trees guarantees single-root.

3.2 Scoring of subtrees

A dependency tree y is commonly factored into smaller *parts*, which can be scored in isolation. At the third stage of our parsing pipeline, an overall dependency tree is computed by the combination of a compatible set of subtrees, thus tree score sums over subtree scores. In this sense, our parsing model is *subtree-factored*. We consider two alternatives for scoring subtrees:

(1) further factor subtrees into smaller parts; or

(2) Dependency-based Convolutional Neural Networks to model dependency chains in trees.

3.3 Factor subtrees to smaller parts

A 'first-order' parser decomposes dependency trees into arcs; and for second-order and third-order factorizations, parts of consecutive siblings, grandchild, grand-sibling and tri-sibling, have been widely studied in literature, e.g. (Eisner, 2000; McDonald and Pereira, 2006; Koo and Collins, 2010). Since our parsing model is subtree-factored, there is no limitation on high-order features for us to consider. However, in practice, we follow the second-order and third-order notations.

3.4 DCNN for scoring subtrees

Dependency-based Convolutional Neural Networks were originally proposed for sentence embedding (Ma et al., 2015) and evaluated for sentiment analysis and question classification tasks. We adopt this model for scoring subtrees since it captures dependency chains in trees including ancestor paths and siblings. Given parsed subtrees from the second stage, an oracle combination computes subtree scores as the number of gold arcs it contains. For any input sentence, if a subtree is selected into the oracle

combination, it is considered as a 'legal' subtree otherwise 'illegal'. A subtree classifier, e.g. DCNN, is trained to tell legal subtrees apart from illegal subtrees. This is not a structured learning schema as we are familiar with for parsing tasks, since DCNN is not designed to capture specific structural factorization of dependency trees, such as arcs or RCNN units as defined in (Zhu et al., 2015).

4 Related work

Klein and Manning (2003) proposed to factor parsing model into a phrase-structure tree and a dependency tree, even before dependency parsing was well-studied later in literature. The idea of combining the merits of constituents and dependency parsing is not new, but our proposed factorization is novel.

Since k-best constituent parsing outputs are used, this work resembles the re-ranking works. However, we do not perform k-best list re-ranking, e.g. (Charniak and Johnson, 2005; Hall, 2007; Qian and Liu, 2015). Neither forest reranking (cube pruning), e.g. (Huang, 2008; Zhang and McDonald, 2012; Hayashi et al., 2011). Our search space is factored into pieces of subtrees, thus represents more combinatory candidates than a k-best list of whole dependency trees. This decomposition distinguishes our work from (Le and Zuidema, 2014) and (Zhu et al., 2015) which use NN-based models to re-rank whole dependency trees. These models could also be evaluated in our parsing pipeline. Furthermore, our work concerns very different aspects with (Ren et al., 2013), which uses dependency model for forest reranking of constituent-based parsing. Our dependency parsing process deals with local dependencies only, a complementary task to the constituent parsing process. The third combination stage performs a global search, neither as a simple reranking. It is more accurate to describe our pipeline as imposing constituent-based structural constraints to dependency parsing. Even though imposing constraints practically performs similarly as pruning, it provokes interesting work, e.g. recent work on approximate Linear Programming decoders for parsing, (Koo et al., 2010; Martins et al., 2011). In this work, we also formulate the search of optimal combination of subtrees as an ILP problem, and due to our efforts on constraining the search space in previous stages, it can be efficiently solved by exact decoding.

Furthermore, works on Vine parsing, e.g. (Dreyer et al., 2006; Rush and Petrov, 2012), especially relate to ours. This line of work pays special attention to lengths of arcs and consider it as an import factor to constrain the search. Instead of pruning arcs by distance like Vine parsing, we decompose parsing of long-distance projections and localized dependencies, which is characterized by pattern in word categories but not absolute distance. There is also a more recent related work, (Fernandez-Gonzalez et al., 2016), that also pays attention to length of arcs. In their work, they use simple criteria based on the length and position of dependency arcs to determine how to combine the outputs of an left-to right transition-based parser and its "mirrored" version.

5 Experiments

Our main experiments are performed on dependency trees extracted from English WSJ Treebank (Marcus et al., 1993). We use Yamada and Matsumoto (2003)'s head rules to convert phrase structures to dependency structures, considering the productivity assumption. Following the conventional split, we use sections 02-21 for training, section 22 for development and section 23 for testing. Dependency parsing is evaluated by unlabeled attachment score (UAS), which is the percentage of words that correctly identified their heads. Chinese experiments are performed on CTB5 with the conventional splits described in (Zhu et al., 2015).

For both English and Chinese experiments, the BLLIP parser (Charniak and Johnson, 2005) is used for the first-stage constituent parsing. Because the BLLIP parser doesn't require POS tag input, we do not impose gold POS tags or use an automatic POS tagger. We use 10-fold cross-validation training data at the third combination stage, since the subtree classifier needs to cope with noisy input from the previous stages. More specifically, we divide training corpus into 10 folds and train a coarse constituent-based parser for each divide, then combine all parsing outputs from each divide into a whole training corpus containing k-best constituent parsing outputs.

Oracle parsing is performed by the oracle combination of oracle-parsed subtrees transformed from k-best constituent parsing outputs. We have defined oracle combination in Section 3.4. Given a carved

	constituents recall 200-best	dependency UAS gold subtrees	oracle combination
avg.>= 5	**98.34**	**96.21**	**95.24**
avg.>= 10	97.97	92.19	91.75

Table 1: The threshold of averaged length for long-distance projecting POS categories affects constituent parsing and dependency parsing.

avg. >= 5	constituents k-best recall	avg. number of subtrees	oracle parsing
k= 200	98.34%	59.7	99.46
k= 500	98.41%	72.75	99.51
k= 1000	98.43%	80.75	99.52

Table 2: The choice of k in k-best constituent parsing.

sequence of input, each word's oracle head is either its gold head, or the root of this sequence when the word's gold head is not seen in the same sequence. Oracle-parsed subtrees are not used for training, but only for tuning parameters on development sets.

5.1 The first-stage constituent parsing

As discussed in section 2.2, there are two distinct jumps in the distribution over averaged lengths of projections headed by each POS category. We select the set of POS tags that are considered as long-distance projecting by tuning a threshold on the development set. As shown in Table 1, a proper setting of this threshold matters for all parsing stages. For following experiments on English (also Chinese and German), we consider a POS category as long-distance projecting, if the averaged distance of its projections is more than or equal to 5. For English, this setting gives us all verbal categories, *wh*-holders, and prepositions. For Chinese, it gives long-distance projecting categories of 'VA', 'DEC', 'P', 'CS', 'SP', 'VV', 'VE', 'LB', 'BA' and 'VC', whose meanings are referred to (Xia et al., 2000).

The choice of k defines k-best constituent parsing outputs that will be transformed to carved sequences in the following dependency parsing stage. As shown in Table 2, a larger k doesn't introduces sharply more subtrees per sentence, because most brackets in the top k-best constituent parsing outputs are the same. This observation shows a great advantage of our factorization and distinguishes our pipeline from k-best list reranking of whole trees. We can make use of a relatively large k to achieve higher recall, which is set to be 500 in the following experiments on English and Chinese.

5.2 Dependency Parsing Results

baseline parsing models	directly parse whole trees to compare	parsing subtrees in pipeline	oracle combination of parsed subtrees	combination by subtree scores of order-1 ptron	combination by subtree scores of order-2 ptron	combination by subtree scores of DCNN
NNDep	91.59	97.05	96.13	92.52	92.87(**+1.28**)	92.79
Mst-1	91.39	96.93	95.90	92.38	92.60 (+1.21)	92.45
Mst-2	92.13	97.05	96.05	92.51	92.88 (+0.75)	92.57
Turbo-standard	93.11	97.34	96.31	92.85	93.17 (+0.06)	92.88
Turbo-full	93.43	97.35	96.36	92.91	93.24 (-0.19)	92.93
merged	N/A	N/A	97.31	93.25	**93.57**	93.26

Table 3: Performance (UAS) of parsing by the combination of subtrees. The order-1/2 perceptron-based subtree classifier and DCNN are three alternative combination strategies. The highest increase of UAS for each baseline model is given in parentheses. A merge of the subtrees from all baseline models can be used to improve the second dependency parsing stage.

We could pick any dependency model for subtree dependency parsing, especially the following:

- An NN-based transition-based model, NNDep, (Chen and Manning, 2014).
- A first- (second-) order graph-based model, MST-1(2), (McDonald and Pereira, 2006).
- A third-order(and beyond) approximate model, Turbo-standard(full), (Martins et al., 2011).

For scoring subtrees, we experiment with a feature-based perceptron classifier of first/second- order MST features, as well as the DCNN model described in Section 3.4. Given the well-known over-fitting problem for re-ranking models, we adopt the same mixture strategies as both (Le and Zuidema, 2014)

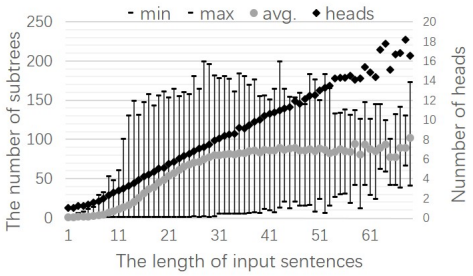


Figure 4: The range (min, max, average) of the number of subtrees (primary y-axis) and the averaged number of long-distance projecting heads (secondary y-axis) with respect to the length of input sentences as computed from 500-best constituent parsing outputs.

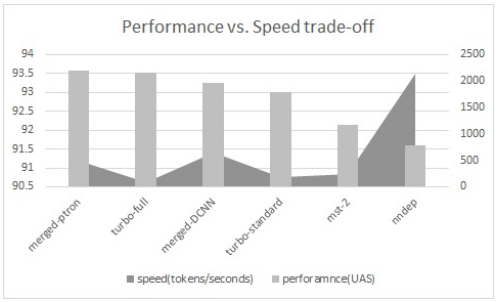


Figure 5: The trade-off between performance (UAS on the promary y-axis) and speed (tokens/seconds on the secondary y-axis).

and (Zhu et al., 2015) with arc scores by a standard Turbo parser. The classifiers are trained with merged dependency parsing outputs, since to do 10-fold cross-validation training for every baseline model is overloading. For a given set of carved sequences transformed from constituent parsing outputs, the merged dependency parsing model covers the subtrees from all baseline models, thus increasing the oracle-combination score by 0.95 even compared to Turbo-full, the best baseline model. More specifically, the feature-based classifiers are trained with online perceptron-based learning (Collins, 2002). DCNN is trained off-line with the same settings as (Ma et al., 2015). And oracle combination as described in Section 3.4 is used to guide training.

As shown in Table 3, with the merged dependency parsing model, our parsing system performs better than Turbo-full, by 0.14 (UAS). However, this is not what we push for in this work. We are more encouraged by the results that, with no algorithmic change to transition-based or first-order models, an increase of 1.28/1.21 (UAS) can be achieved solely due to a factorization of input. It is worth to notice that, on full dependency parsing, Turbo-full performs better than the transition-based and first-order models by nearly 2 (UAS). However, all baseline models achieve the same level of performance for subtree dependency parsing, no bigger difference than 0.5 (UAS). This is enough to show that the proposed factorization works. The use of DCNN doesn't introduce special gain in performance, however, we observe the same advantage as described in (Chen and Manning, 2014), i.e. the NN-based model is nearly 40 times faster than the feature-based discriminative classifier.

5.3 Efficiency

The efficiency of solving the ILP problem at the third combination stage depends on the number of possible subtrees generated at the second stage. As shown in Figure 4, the averaged number of long-distance projecting heads is linearly dependent on the input length, which suggests that the number of subtrees increases at most linearly. Furthermore, since we only keep k-best constituent parsing outputs, the averaged number of subtrees stays below a constant. When k is set as large as 500, the averaged number of subtrees still stays below 100, as shown in Figure 4. Therefore a general ILP decoder, e.g. cplex, is efficient enough. This efficient combination clearly shows the strength of the proposed factorization.

As shown in Table 3, all baseline models achieve comparable performance for subtree dependency parsing. In this sense, we can now make use of the fastest dependency parsing model for subtree parsing, without concerns in performance loss.

It turns out that the k-best constituent parsing stage is the efficiency bottleneck. Therefore, we propose to prune the input to this stage. As a first try, we tag all head words with 'H', any beginning or end words of some bracket with 'B', and other words with 'I'. Then those words of tag 'I' can be pruned for constituent parsing. However, this classification pattern cannot be acquired. This failure is worth special notice, it reminds us that the success of a classification task not only depends on powerful machine learning techniques but also crucially on appropriate representations. We then keep all punctuations, all words before head words and all words before punctuational bracket endings , i.e. tag them with label 'B'. This tagging task can be performed with a precision above 93%, with a perceptron tagger. However,

	grammar size	input avg./max	words per sec.	oracle parsing
full	69630	24/99	681	99.51
pruned	57256	18/76	912	99.50

Table 4: Compare full and pruned input to constituent parsing. Grammar size is the number of unlexicalized rewrite rules.

(UAS)	baseline	subtree	oracle combination	order-2 ptron
MST-1	80.48	96.61	88.54	84.11 (**+3.63**)
Turbo-full	84.74	97.08	89.07	84.35 (-0.39)

Table 5: Results on CTB5, the same terms of Table 3.

high recall of 'B' and 'H' is more crucial for our task, i.e. relevant input are not wrongly pruned. For decoding, we set $b = 20$ for tag 'B' and 'H', and 0 for tag 'I', achieving 98.4% recall of tag 'B' and 'H', still pruning 27.5% of the original input. As shown in Table 4, by feeding pruned input to the coarse constituent parsing, we can process about 25% words more per second at this stage.

In Figure 5, we show that the proposed pipeline provides a better trade-off between parsing performance and parsing efficiency. Merged-ptron achieves comparable performance (93.57 UAS) with Turbo-full (93.24 UAS) but 5 times faster. Even though NNDep (92.87 UAS) is about 6 times faster than merged-ptron, there is a loss of 1.3 (UAS) in performance.

6 Experiments on Chinese

It is well-known that, Chinese parsing performance is much lower than English, for example, with MST-1 and Turbo-full, it is only 80.48 and 84.74 (UAS) respectively for full dependency parsing. Thus it is a surprise for us to see in Table 5 that, for subtree dependency parsing, MST-1 and Turbo-full achieve the same level of parsing performance on Chinese as on English, 96.61 and 97.08 (UAS) respectively. It is the first constituent parsing stage that reveals the difficulty in parsing Chinese. With 500-best c-parsing over full input, the recall is merely 88.63%, and even with the proposed pruned input, the recall is improved by 1% only. However, if we prune the input for constituent parsing with oracle 'H', 'B', 'I' tags, a recall as high as 97% can be achieved. This big gap in results leaves a huge space for future work to explore. The proposed factorization intriguingly discovers the bottleneck in Chinese parsing.

For non-projective parsing, we take the German case as an example. There are about 3 percent of arcs are crossing in the TIGER corpus (Brants et al., 2002). Recall that the projective assumption is only essential to the first constituent parsing stage. If these crossing arcs are local, our pipeline still works. However, 98% of the crossing arcs involve long-distance projecting heads. We can employ techniques such as super-tagging instead of constituent parsing to deal with non-projective long-distance dependencies .

7 Conclusion and Discussion

We propose a novel factorization of parsing task that explicitly utilizes the distinction between long-distance projections and localized dependencies. This intuitive idea works out given that this factorization can be characterized by POS categories of each projection's head word. The first and second stages, which cope with long-distance projections and localized dependencies separately, are fed with complementary input instead of full input, thus taking great advantage of the constrained search space to perform better and faster.

In this work, the proposed factorization is realized in a parsing pipeline of three stages. At the second stage, any dependency parsing model can be used for subtree parsing, so 'baseline' models in our work are not only used for comparison, but to show how much increase in parsing performance can be introduced solely by the proposed factorization, with no change in parsing algorithms or learning strategies to baseline models. On the final stage, dependency parsing is subtree-factored, thus any high-order feature-based model and even continuous representation of dependency chains can be used for subtree scoring. Moreover, dependency parsing over carved sequences of the original input especially suits for parallel computing. Therefore, the proposed factorization provides a better trade-off in parsing speed and performance. In future work, we would like to experiment with end-to-end learning framework for future reducing error propagation.

Acknowledgements

We sincerely thank the anonymous reviewers for their thorough reviewing and valuable suggestions. Qun Liu's work is partially supported by Science Foundation Ireland at ADAPT Centre for Digital Content Platform Research (Grant 13/RC/2106).

References

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

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

Christopher D. Manning Dan Klein. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15: Proceedings of the 2002 Conference*, volume 15, page 3. MIT Press.

Markus Dreyer, David A. Smith, and Noah A. Smith. 2006. Vine parsing and minimum risk reranking for speed and precision. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 201–205, New York City, June. Association for Computational Linguistics.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 340–345. Association for Computational Linguistics.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in probabilistic and other parsing technologies*, pages 29–61. Springer.

Daniel Fernandez-Gonzalez, Carlos Gomez-Rodriguez, and David Vilares. 2016. Improving the arc-eager model with reverse parsing. *Computing & Informatics*, 35(3).

Keith Hall. 2007. K-best spanning tree parsing. In *Annual Meeting-Association for Computational Linguistics*, volume 45, page 392.

Katsuhiko Hayashi, Taro Watanabe, Masayuki Asahara, and Yuji Matsumoto. 2011. Third-order variational reranking on packed-shared dependency forests. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1479–1488. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio, June. Association for Computational Linguistics.

Aravind K Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In *Handbook of formal languages*, pages 69–123. Springer.

T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–11, Uppsala.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA, October. Association for Computational Linguistics.

Phong Le and Willem Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 729–739, Doha, Qatar, October. Association for Computational Linguistics.

Mingbo Ma, Liang Huang, Bowen Zhou, and Bing Xiang. 2015. Dependency-based convolutional neural networks for sentence embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 174–179, Beijing, China, July. Association for Computational Linguistics.

Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

André FT Martins, Noah A Smith, Pedro MQ Aguiar, and Mário AT Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 238–249. Association for Computational Linguistics.

Andre Martins, Miguel Almeida, and A. Noah Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 617–622. Association for Computational Linguistics.

Ryan T McDonald and Fernando CN Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Association for Computational Linguistics (ACL)*.

J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 221–225.

Xian Qian and Yang Liu. 2015. Feature selection in kernel space: A case study on dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1180–1190, Beijing, China, July. Association for Computational Linguistics.

Xiaona Ren, Xiao Chen, Chunyu Kit, Tower D SOGOU-INC, and Tsinghua Tongfang High-tech Plaza. 2013. Combine constituent and dependency parsing via reranking. In *IJCAI*.

Alexander Rush and Slav Petrov. 2012. Vine pruning for efficient multi-pass dependency parsing. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Montréal, Canada, June. Association for Computational Linguistics.

Fei Xia, Martha Palmer, Nianwen Xue, Mary Ellen Okurowski, John Kovarik, Fu dong Chiou, Shizhe Huang, Tony Kroch, and Mitch Marcus. 2000. Developing guidelines and ensuring consistency for chinese text annotation. In *In Proceedings of the Second Language Resources and Evaluation Conference*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *8th International Workshop on Parsing Technologies*.

Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 320–331, Jeju Island, Korea, July. Association for Computational Linguistics.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015. A re-ranking model for dependency parser with recursive convolutional neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1159–1168, Beijing, China, July. Association for Computational Linguistics.

K-SRL: Instance-based Learning for Semantic Role Labeling

Alan Akbik
IBM Research Almaden
650 Harry Road, San Jose
CA 95120, USA
akbika@us.ibm.com

Yunyao Li
IBM Research Almaden
650 Harry Road, San Jose
CA 95120, USA
yunyaoli@us.ibm.com

Abstract

Semantic role labeling (SRL) is the task of identifying and labeling predicate-argument structures in sentences with semantic frame and role labels. A known challenge in SRL is the large number of low-frequency exceptions in training data, which are highly context-specific and difficult to generalize. To overcome this challenge, we propose the use of *instance-based learning* that performs no explicit generalization, but rather extrapolates predictions from the most similar instances in the training data. We present a variant of *k-nearest neighbors* (kNN) classification with composite features to identify nearest neighbors for SRL. We show that high-quality predictions can be derived from a very small number of similar instances. In a comparative evaluation we experimentally demonstrate that our instance-based learning approach significantly outperforms current state-of-the-art systems on both in-domain and out-of-domain data, reaching F_1-scores of 89,28% and 79.91% respectively.

1 Introduction

Semantic role labeling (SRL) is the task of annotating predicate-argument structures in sentences with shallow semantic information. One prominent labeling scheme for the English language is the Proposition Bank (Palmer et al., 2005), which annotates predicates with *frame* labels and arguments with *role* labels (see Figure 1 for examples). Frame labels disambiguate the predicate meaning in the context of the sentence. Role labels roughly correspond to simple questions (*who, when, how, why, with whom*) with regards to the disambiguated predicate. SRL has been found useful for a wide range of applications such as information extraction (Fader et al., 2011), question answering (Shen and Lapata, 2007; Maqsud et al., 2014) and machine translation (Lo et al., 2013).

Current state-of-the-art SRL approaches train classifiers with bags of features (Johansson and Nugues, 2008; Björkelund et al., 2009; Choi and Palmer, 2011) to predict semantic labels for each constituent in a sentence. These approaches typically employ classifiers such as logistic regression or SVM that learn for each feature a measure of impact on the classification decision and abstract away from local contexts in specific training examples.

Local bias. We argue that such approaches are not ideal for SRL due to a strong local bias of features within specific contexts. Low-frequency examples in SRL are often not noise to be abstracted away, but rather correspond to exceptions that require explicit handling.

For example, consider the task of argument labeling: Arguments that are syntactically realized as passive subjects are typically labeled **A1**[1]. However, there exist numerous low-frequency exceptions to this rule. For instance, passive subjects of certain frames (such as the frame TELL.01) are most commonly labeled **A2**. See Figure 1 for an example. Other examples of local bias include different types of diathesis alternation which affect specific frames and argument types (Kipper et al., 2008), the syntactic realization of higher order roles (A2 to A5) which is highly irregular among frames (Palmer et al., 2005), and the realization of roles in non-agentive frames. These phenomena are observed only in specific and often low-frequency contexts of composite features, but are highly relevant to SRL.

[1]This corresponds to the linguistic intuition that active subjects are commonly thematic *agents*, while direct objects and passive subjects are most commonly the thematic *patient* or *theme* of a frame (van der Plas et al., 2014)

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 599–608, Osaka, Japan, December 11-17 2016.

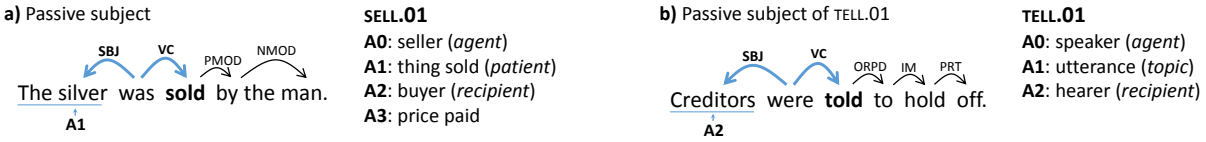


Figure 1: Example sentences with a passive subject (underlined). Passive subjects are typically labeled **A1** (e.g. Sentence a), but there are exceptions to this rule (e.g. frame TELL.01 in Sentence b).

	STATEMENT	COMPOSITE	SUPPORT
(1)	57% of all subjects are labeled **A0**	**P**	17,788 instances
(2)	33% of all subjects are labeled **A1**	**P**	17,788 instances
(3)	74% of *active* subjects are labeled **A0**	**P+V**	13,737 instances
(4)	86% of *passive* subjects are labeled **A1**	**P+V**	4,051 instances
(5)	100% of passive subjects of SELL.01 are labeled **A1**	**P+V+F**	137 instances
(6)	88% of passive subjects of TELL.01 are labeled **A2**	**P+V+F**	53 instances

Table 1: Observations based on CoNLL09 training data: The more atomic features we include in a composite, the more discriminative (and descriptive) it becomes, but generally with lower support.

Feature contexts. We propose to explicitly capture local bias with feature contexts by constructing composites of standard SRL features. Refer to Table 1 for a list of observations over the CoNLL09 shared task gold data (Hajič et al., 2009) for different numbers of features combined into composites: Statements 1 and 2 involve only the syntactic path feature P that models the syntactic function of an argument. Statements 3 and 4 use a composite feature of P and V, the predicate voice feature (either *active* or *passive*). Finally, statements 5 and 6 use a composite feature of P, V and F, the specific frame context.

We make three observations in Table 1: First, the more atomic features we include in a composite feature, the more discriminative it becomes and the more explicitly it captures local bias. For instance, statement 6 is a composite of three atomic features and explicitly captures the exception for passive subjects of TELL.01 discussed earlier. Second, higher order composite features tend to have lower support (i.e. number of training examples that share this combination of atomic features). The use of composite features can therefore aggravate sparsity issues in training data. Finally, since composite features make the interplay of features explicit, they can be rendered as human readable statements. Classification decisions using such features can be easily interpretable for error analysis and extension.

Instance-based Learning for SRL Based on these observations, we propose to use *instance-based learning* (Aha et al., 1991; Daelemans and Van den Bosch, 2005) for SRL. Such learning does not abstract away from specific feature contexts, but rather considers the overall similarity of a test instance to instances in the training data. It has been shown to be applicable to a range of NLP tasks such as PoS tagging (Daelemans et al., 1999), dependency parsing (Nivre et al., 2004) and word sense disambiguation (Veenstra et al., 2000). The arguably most well-known approach of this kind is *k-nearest neighbors* classification (kNN) in which the class label is determined as the majority label of the k most similar training examples (Cover and Hart, 1967).

We propose to identify nearest neighbors using composite features, i.e. instances that share the most similar combination of atomic features. We use a function to assign to each composite of atomic features a discrete distance value, effectively rendering the search for nearest neighbors as a search within a *Parzen* window (Parzen, 1962). The variable k represents the minimum support within this window that we require.

Figure 2 illustrates our proposed approach: To classify the underlined argument in the sample sentence, we search for nearest neighbors using composite features. A distance 1 composite feature consists of P+V+F and AL, the lemma of the argument head. Nearest neighbors at this distance are therefore all training instances in which "creditor" is a passive subject of TELL.01. As the diagram on the right hand side in Figure 2 shows, this finds only one match, labeled **A2**, which is below the minimum support k that we require. We therefore increase the window to distance 2, which relaxes the argument head lemma

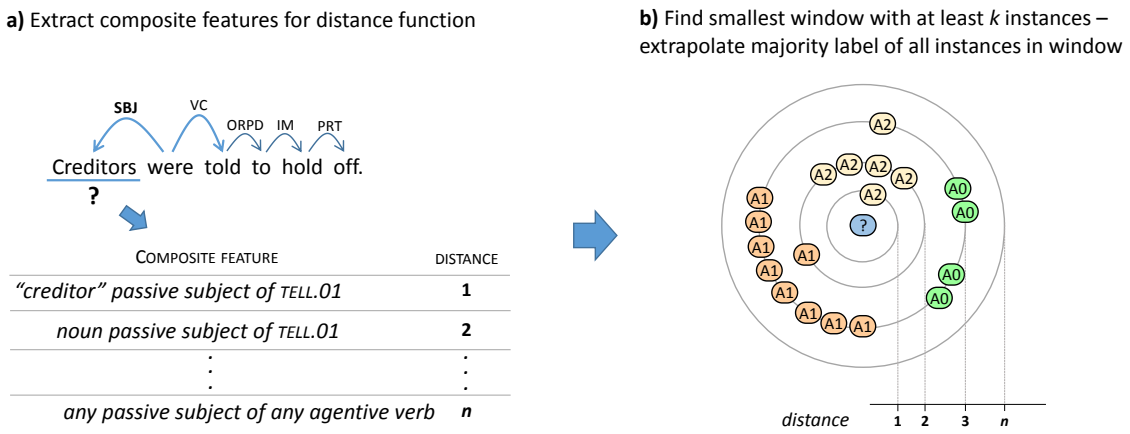


Figure 2: Example of argument labeling with nearest neighbor classification and a composite feature distance function.

restriction on composites features. Nearest neighbors at this distance are all passive subjects of TELL.01 in the training data. As the diagram shows, there are six nearest neighbors within a distance 2 window. From this neighborhood, we extrapolate the label **A2** as prediction.

Contributions We propose a simple and highly effective instance-based learning model for semantic role labeling[2]. We develop a *nearest k-window* variant of kNN in which we use a composite feature distance function to explicitly capture local contexts in sparse data. We give a full description of our SRL system, dubbed K-SRL, motivate and illustrate the atomic and composite features we use, and describe an easy-first algorithm for modeling global argument labeling constraints (Section 2). We present a detailed experimental evaluation that shows that our proposed approach significantly outperforms existing state-of-the-art systems (Section 3).

2 Instance-based Learning for SRL

We use instance-based learning for SRL as a sequence of two classification tasks: First, joint predicate identification and classification (referred to as *predicate labeling*), followed by joint argument identification and classification (referred to as *argument labeling*). See Figure 3 for an illustration. In this section, we describe the proposed nearest *k*-window classifier (Section 2.1) and discuss the specific features used (Sections 2.3 and 2.2), before presenting how we include global constraints into argument labeling (Section 2.4).

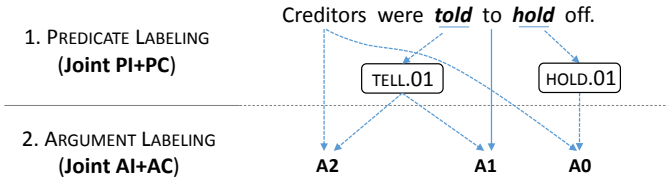


Figure 3: K-SRL system outline.

2.1 Nearest *k*-Window Classification

Algorithm 1 outlines our nearest *k*-window classification algorithm. It takes as input unlabeled instances and performs feature extraction. A distance function assigns each feature an integer distance value, with 1 as the closest distance. The search for nearest neighbors begins with a window of distance 1. The algorithm retrieves for each unlabeled sample all training examples whose distance from the current sample lie within this window. If the window contains fewer than *k* training instances, we increase the window size until it passes a threshold. Among the instances in the window, the algorithm determines

[2]In this work, we focus on verbal predicates since they are comprehensively and consistently labeled in available PropBank releases. We aim to revisit SRL for other types of predicates once current efforts to consistently annotate noun predicates, light verb constructions and adjectives (Bonial et al., 2014) are completed.

the relative frequencies of each label. For instance, if the window contains 10 similar instances of which 9 are labeled **A0** and one is labeled **A1**, the relative frequencies of **A0** and **A1** are 90% and 10% respectively. We interpret this relative frequency as a measure of *confidence*. The label with the highest relative frequency is returned as the most confident prediction. A label is returned only if its associated confidence is higher than a threshold (denoted by θ). If either insufficient examples in the nearest neighborhood are found, or the associated confidence is below the threshold, we return a fallback label. For the subtask of predicate labeling, the fallback label is the most common sense of a verb. For argument labeling, the fallback is to not label the word as an argument.

Algorithm 1 Nearest k-Window Classification

```
ws ← 1
if ws ≤ maximal distance then
  for each x ∈ Unknown Sample do
    Add to I all y ∈ Training Set, where distance(x, y) ≤ ws
    if |I| ≥ k then
      Determine majority class label I
      if the relative frequency of the label for x ≥ θ then
        Return I, the label and its relative frequency for x
      end if
    else
      ws = ws + 1
    end if
  end for
end if
```

2.2 Features for Instance-based Predicate Labeling

Atomic Features The Proposition Bank distinguishes different frame options for a verb based on syntactic subcategorization and coarse-grained polysemy (Palmer et al., 2005). For instance, the verb *return* may evoke the frames RETURN.01 (*return to a place*, as in *John returned to Boulder*) and RETURN.02 (*return an item to someone*, as in *John returned the stapler to Mary*). The key difference of the two in subcategorization is that RETURN.02 may take a syntactic object while RETURN.01 may not. Besides objects, other differentiators in subcategorization involve particles, complements and prepositional objects mediated by different prepositions.

We use each component of a subcategorization frame as one feature: The subject lemma S, the verb lemma VB, the verb particle VP, the object lemma O and the prepositional object PP . Each of these features may also be empty if unobserved. In addition to such lexical features, we define a set of binary features that indicate whether a subcategorization frame component is observed or not: S? for subjects, O? for objects and PP? for prepositional objects. Finally, we use the verb voice V since some frames are more commonly observed in passive voice.

Composite Feature The set of atomic features described above constructs one single composite feature, denoted as F_x for a given instance x. The distance between x_{test} (test instance) and x_{train} (training instance) corresponds to the total number of non-empty features that the two instances do not share, defined as $d(x, y) = |F_{x_{test}} \cup F_{x_{train}}| - |F_{x_{test}} \cap F_{x_{train}}|$. This distance function in essence corresponds to Jaccard distance, without normalizing to a value between 0 and 1.

2.3 Features for Instance-based Argument Labeling

Atomic Features Table 2 includes a list of atomic features. Besides four well-established features from previous work (i.e. the predicate frame F, the predicate voice V, the argument head lemma AL and the argument head PoS tag AP), we define the following two novel atomic features:
(1) *Syntactic-Semantic Path* P: A variant of the syntactic path that, instead of traversing only the syntactic tree, traverses semantic arcs from preceding classifications whenever possible. It is designed for the

FEATURE	SHORTHAND
Predicate frame	F
Predicate frame class	FC
Predicate voice	V
Syntactic-semantic path	P
Argument head lemma	AL
Argument head pos	AP

Table 2: Argument labeling features, with novel features in bold.

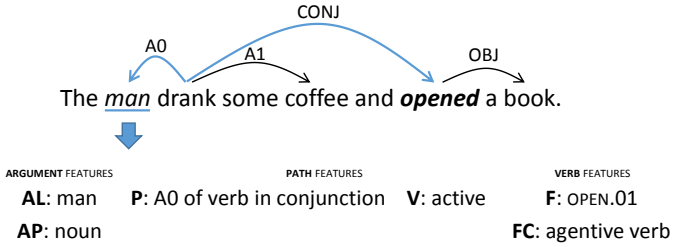


Figure 4: Features extracted for the underlined word (*man*) with regards to the predicate OPEN.01.

COMPOSITE	DISTANCE	EXAMPLE
AL+V+P+F	1	"man" ∧ A0 of verb in conjunction ∧ active ∧ OPEN.01
AP+V+P+F	2	any noun ∧ A0 of verb in conjunction ∧ active ∧ OPEN.01
V+P+F	3	any word ∧ A0 of verb in conjunction ∧ active ∧ OPEN.01
AL+V+P+FC	4	"man" ∧ A0 of verb in conjunction ∧ active ∧ agentive verb
AL+V+P+FC	5	any noun ∧ A0 of verb in conjunction ∧ active ∧ agentive verb
V+P+FC	6	any word ∧ A0 of verb in conjunction ∧ active ∧ agentive verb

Table 3: Distance value assigned to each composite feature, with an example for features extracted in Figure 4.

phenomenon of *raised* arguments, defined as syntactic constituents of a preceding verb. For instance, in Figure 4, *man* is a constituent of the verb *drank* as well as a raised argument for the verb *open*. In this example, we build the path from *man* to *open* by first traversing the semantic arc (A0) from *man* to *drank* and then the syntactic arc (CONJ) from *drank* to *open*. The resulting syntactic-semantic path is verbalized as "A0 of verb in conjunction".

(2) *Frame Class* FC: This feature categorizes frames based on whether they may take a thematic *agent* as argument. Some frames, such as FESTER.01 and HOVER.01, cannot and are therefore considered non-agentive. Non-agentive frames define no **A0** and therefore realize semantic roles differently[3].

Composite Features Table 3 lists all composite features along with their associated distances, including features extracted for the sample sentence in Figure 4. As described in Section 1, the relevant context for argument labeling includes: 1) The syntactic-semantic relationship between predicate and argument (P+V), 2) The frame-specific context of this syntactic-semantic relationship (F or FC), and 3) The argument-specific context (AL or AP). We require each of the three components to be represented in each composite feature in order to capture argument contexts. We define a distance function that assigns the closest distance 1 to the most discriminative of these composite features (i.e. AL+P+V+F). The function assigns higher distances to composites with fewer or less specific features (AP is a less specific representation of the argument context than Al).

This approach draws inspiration from *backoff* modeling, a well known method for addressing sparsity in language modeling with n-grams (Katz, 1987): If insufficient training data exists, such models commonly backoff to lower histories (for instance, a 3-gram model may back off to a 2-gram language model). The six distance values assigned to composite features in Table 3 may be interpreted in a similar spirit since our approach broadens the search to nearest neighbors with less specific composite features if insufficient training data exists.

2.4 Easy-First Argument Labeling

While argument labeling decisions are made locally, each core semantic role (labels **A0** through **A5**) may only be assigned once per predicate (Che et al., 2009). To include this global constraint, we use a greedy approach in which already assigned core labels are removed from consideration for the remaining predictions. Our approach follows an easy-first philosophy (Goldberg and Elhadad, 2010) where clas-

[3]For example, while active subjects are most commonly labeled **A0** (agent) of a verb ("the dog ate", "the bird sang"), they are typically the **A1** (theme) of non-agentive frames ("the wound festered").

Algorithm 2 Easy-First Argument Labeling

```
for each predicate p ∈ Unknown Sample do
  A ← ∅
  C ← Candidate arguments of p, their labels and confidence value in sorted order by confidence
  for c ∈ C with the highest confidence value in C do
    Remove c from C
    if Label of c ∉ Set of labels in A then
      Add c to A
    end if
  end for
end for
```

sifications for all predicate arguments are ordered by confidence and highest confidence predictions are made first. Algorithm 2 outlines this approach.

3 Experiments

In this section we evaluate K-SRL, our proposed instance-based labeling approach for SRL. We conduct a comparison study to evaluate its performance against previously published state-of-the-art systems. We also examine how different parameters of K-SRL impact its performance, including the minimum support variable *k*, different components in composite features, and our interpretation of relative label frequencies in the nearest neighborhood as an indication of confidence for assigning labels.

3.1 Experimental Setup

We use the benchmark data sets from the CoNLL-2009 shared task (Hajič et al., 2009) and compare our results against the top two scoring systems of the CoNLL-2009 shared task as well as two recent state-of-the-art systems: 1) CHEN (Zhao et al., 2009), which uses maximum entropy classifiers. 2) CHE (Che et al., 2009), which uses SVM classifiers. 3) MATEPLUS (Roth and Woodsend, 2014a), a state-of-the-art extension of a previous system (Björkelund et al., 2009) that uses logistic regression classifiers and word embeddings. 4) PATHLSTM (Roth and Lapata, 2016), the current state-of-the-art which uses logistic regression classifiers for predicates and neural network models with word embeddings for arguments. Our default settings for K-SRL are $k = 3$ and confidence threshold $\theta = 0$, both determined through experimentation. For both settings, we present parameter sweep experiments.

We compute the precision, recall and F_1 to measure the quality of the systems. In our study, we focus on verbal predicates and their roles, which we evaluate using the scoring metric of the CoNLL-2009 shared task. We recomputed the measures for the previous state-of-art systems using their published results to focus on verbal predicates and their roles[4].

3.2 Evaluation Results

Tables 4 and 5 summarize the results for our comparison study on in-domain and out-of-domain data respectively. As can be seen, K-SRL outperforms all previous approaches by a significant margin on both data sets. In the in-domain setting, K-SRL achieves 89.28% F_1-score, outperforming PATHLSTM, the currently published state-of-the-art approach, by 1.1 percentage points. In the out-domain setting, K-SRL achieves 79.91% F_1-score, outperforming MATEPLUS, the best previous system on out-of-domain data in our evaluation, by over 3 percentage points, and PATHLSTM by an even larger margin.

3.3 Additional Experiments

We conduct a detailed empirical examination to evaluate different aspects of our approach and make the following observations:

[4] As a result, the numbers for previous work reported here are slightly different from the published numbers.

SYSTEM	PRECISION	RECALL	F_1
CHE	87.43	83.92	85.64
CHEN	88.45	84.22	86.28
MATEPLUS	89.59	86.07	87.79
PATHLSTM	90.24	86.24	88.19
K-SRL	**91.21**	**87.42**	**89.28**
K-SRL$_{local}$	90.19	87.15	88.64
K-SRL$_{(-AL)}$	90.33	86.55	88.4
K-SRL$_{(-F)}$	88.74	85.11	86.89
K-SRL$_{(-FC)}$	91.17	87.53	89.31

Table 4: Evaluation results on in-domain data.

SYSTEM	PRECISION	RECALL	F_1
CHE	76.25	71.24	73.66
CHEN	78.1	71.64	74.73
MATEPLUS	79.46	74.21	76.74
PATHLSTM	79.92	73.31	76.47
K-SRL	**82.09**	**77.84**	**79.91**
K-SRL$_{local}$	80.38	77.78	79.06
K-SRL$_{(-AL)}$	81.69	77.28	79.42
K-SRL$_{(-F)}$	80.96	76.88	78.86
K-SRL$_{(-FC)}$	81.69	77.71	79.65

Table 5: Evaluation results on out-of-domain data.

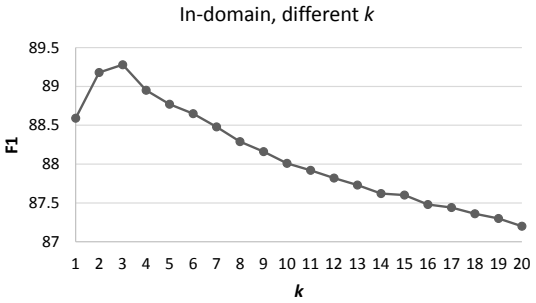


Figure 5: Parameter sweep over k on in-domain data.

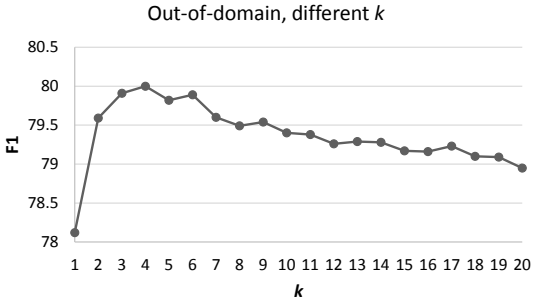


Figure 6: Parameter sweep over k on out-of-domain data.

Highest quality predictions from small neighborhoods. To determine the best setting for *k*, we conduct a parameter sweep experiment. Figures 5 and 6 summarize results of SRL for *k* from 1 to 20 on both in-domain and out-domain data. The results show that the best F_1 scores are achieved at relatively low settings for *k*, with best results obtained with $k = 3$ for in-domain and $k = 4$ for out-of-domain data. At higher *k*, F_1-score drops gradually, indicating that this approach loses its ability to capture local bias. At lower *k*, the approach overfits, decreasing F_1-score especially in the out-of-domain scenario (↓1.8 pp). These observations confirm our initial conjecture that SRL is affected by a strong local bias and that a small nearest neighborhood suffices to make high quality predictions.

Global constraints improve argument labeling. We run an ablation test in which we make only local predictions without modeling global constraints as described in Section 2.4, which reduces the F_1-score by .6 and .8 percentage points respectively on in-domain and out-of-domain data (see K-SRL$_{local}$ in Tables 4 and 5). These results are in line with previous evaluations on the impact of modeling global argument constraints (Toutanova et al., 2008; Roth and Lapata, 2016).

Frame and argument contexts are important. To assess the importance of individual features in their contexts, we run ablation tests in which we remove individual atomic features from composites, as summarized in Tables 4 and 5. Specifically, removing the frame feature F from argument labeling (K-SRL$_{(-F)}$)), which causes all argument labeling predictions to be made without frame-specific contests, leads to the most significant decrease on F_1 scores (↓2.5 pp and ↓1 pp) in our ablation tests. Omitting argument head lemma feature AL (K-SRL$_{(-AL)}$), the only feature capturing argument-level selectional preference (Resnik, 1997) in our approach, results in evident reduction on F_1 scores (↓0.8 pp and ↓0.5 pp). Meanwhile, the removal of frame class feature (K-SRL$-FC$) impacts only the out-of-domain scenario slightly (↓0.3 pp). This observation indicates that small neighborhoods with the frame feature often suffice to capture exceptions for non-agentive verbs.

Relative frequencies measures confidence. We assess our interpretation of relative frequencies in the nearest neighborhood as a measure of confidence by running a parameter sweep over θ. The results are depicted in Figures 7 and 8. As can be seen, precision improves at higher θ, while recall decreases, indicating that the quality of label prediction positively correlates with the associated confidence. We measure the best F_1-scores at $\theta = 0.5$ and $\theta = 0.6$ respectively, but F_1-score remains relatively stable between $\theta = 0.0$ and $\theta = 0.7$, indicating a balanced trade-off within these parameters. These observa-

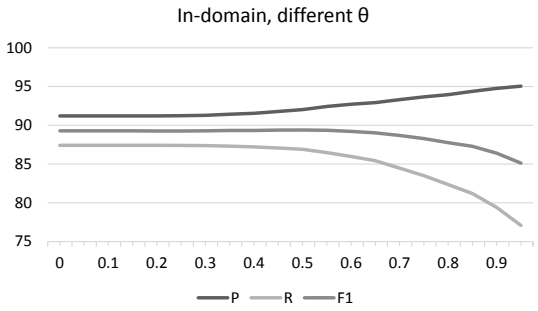


Figure 7: Parameter sweep over θ on in-domain data.

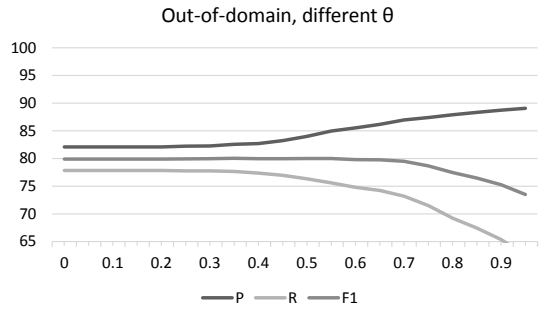


Figure 8: Parameter sweep over θ on out-of-domain data.

tions indicate that relative frequencies can serve as a good measure of confidence and be used to influence the precision-recall trade-off.

3.4 Discussion

Our instance-based learning approach is designed to capture the strong local bias of SRL. We note that even in out-of-domain evaluation scenarios, a very small nearest neighborhood suffices to make high quality predictions. Our experimental results demonstrate the effectiveness of this approach compared to previous state-of-the-arts for both in-domain and out-of-domain scenarios.

The state-of-the-art results are also remarkable in light of the relatively simple feature set we used. While previous work has investigated the use of word clusters (Choi and Palmer, 2011), word embeddings (Roth and Woodsend, 2014b; Roth and Lapata, 2016) and explicit learning of selectional preference (Zapirain et al., 2013) for better generalization across the training data, such features are absent in our current approach. Instead, for predicate labeling we use only the subcategorization frame and for argument labeling a simple set of 6 basic atomic features. This is in stark contrast to previous works that often employ dozens of different features classes (Johansson and Nugues, 2008; Björkelund et al., 2009; Choi and Palmer, 2011).

Interpretability of classification decisions. Our approach has the advantage of interpretability since each classification is determined through a specific composite feature that can be translated into a human readable statement (as illustrated in Table 3). This enables us to easily understand classification results and debug misclassifications, and thus facilitates the process of defining atomic features and composites for SRL. At the same time, we note that explicitly modeling composites does add another layer of complexity in feature engineering to this task. We plan to further investigate this in future work.

4 Conclusion and Outlook

We introduced an instance-based learning approach for semantic role labeling that is designed to address the large number of low-frequency exceptions in training data. We proposed to construct composites based on a few existing well-known features to identify similar instances. Our experimental results indicates that our model built on top of this approach significantly outperform existing systems, leading to new state-of-the-art results in SRL for verbal predicates and their roles.

We intend to focus more specifically on feature engineering for instance-based SRL. In particular, we plan to explore automatic feature selection methods especially in the context of composite features. We also plan to evaluate generalization features such as word clusters or word embeddings in the context of instance-based SRL.

Finally, we plan to extend our system to different types of predicates including nouns and complex predicates (Bonial et al., 2014), as well as evaluate its applicability to SRL in different languages (Xue and Palmer, 2005).

References

David W Aha, Dennis Kibler, and Marc K Albert. 1991. Instance-based learning algorithms. *Machine learning*, 6(1):37–66.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 43–48. Association for Computational Linguistics.

Claire Bonial, Julia Bonn, Kathryn Conger, Jena D Hwang, and Martha Palmer. 2014. Propbank: Semantics of new predicate types. In *LREC*, pages 3013–3019.

Wanxiang Che, Zhenghua Li, Yongqiang Li, Yuhang Guo, Bing Qin, and Ting Liu. 2009. Multilingual dependency-based syntactic and semantic parsing. In *Proceedings of the thirteenth conference on computational natural language learning: shared task*, pages 49–54. Association for Computational Linguistics.

Jinho D Choi and Martha Palmer. 2011. Transition-based semantic role labeling using predicate argument clustering. In *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, pages 37–45. Association for Computational Linguistics.

Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

Walter Daelemans and Antal Van den Bosch. 2005. *Memory-based language processing*. Cambridge University Press.

Walter Daelemans, Sabine Buchholz, and Jorn Veenstra. 1999. Memory-based shallow parsing. In *Proceedings of the Third Conference on Computational Natural Language Learning*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

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

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18. Association for Computational Linguistics.

Richard Johansson and Pierre Nugues. 2008. Dependency-based semantic role labeling of propbank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 69–78. Association for Computational Linguistics.

Slava Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of english verbs. *Language Resources and Evaluation*, 42(1):21–40.

Chi-kiu Lo, Meriem Beloucif, and Dekai Wu. 2013. Improving machine translation into chinese by tuning against chinese meant. In *Proceedings of the Tenth International Workshop on Spoken Language Translation (IWSLT 2013)*.

Umar Maqsud, Sebastian Arnold, Michael Hülfenhaus, and Alan Akbik. 2014. Nerdle: Topic-specific question answering using wikia seeds. In *COLING (Demos)*, pages 81–85.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of the Eighth Conference on Computational Natural Language Learning*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Emanuel Parzen. 1962. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076.

Philip Resnik. 1997. Selectional preference and sense disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How*, pages 52–57. Washington, DC.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany. To appear.

Michael Roth and Kristian Woodsend. 2014a. Composition of word representations improves semantic role labelling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 407–413, Doha, Qatar, 25–29 October.

Michael Roth and Kristian Woodsend. 2014b. Composition of word representations improves semantic role labelling. In *EMNLP*, pages 407–413. ACL.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *EMNLP-CoNLL*, pages 12–21.

Kristina Toutanova, Aria Haghighi, and Christopher D Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

Lonneke van der Plas, Marianna Apidianaki, Rue John von Neumann, and Chenhua Chen. 2014. Global methods for cross-lingual semantic role and predicate labelling.

Jorn Veenstra, Antal Van den Bosch, Sabine Buchholz, Walter Daelemans, et al. 2000. Memory-based word sense disambiguation. *Computers and the Humanities*, 34(1-2):171–177.

Nianwen Xue and Martha Palmer. 2005. Automatic semantic role labeling for chinese verbs. In *IJCAI*, volume 5, pages 1160–1165. Citeseer.

Benat Zapirain, Eneko Agirre, Lluis Marquez, and Mihai Surdeanu. 2013. Selectional preferences for semantic role classification. *Computational Linguistics*, 39(3):631–663.

Hai Zhao, Wenliang Chen, Chunyu Kit, and Guodong Zhou. 2009. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 55–60. Association for Computational Linguistics.

Keystroke dynamics as signal for shallow syntactic parsing

Barbara Plank
University of Groningen
The Netherlands
b.plank@rug.nl

Abstract

Keystroke dynamics have been extensively used in psycholinguistic and writing research to gain insights into cognitive processing. But do keystroke logs contain actual *signal* that can be used to *learn* better natural language processing models?

We postulate that keystroke dynamics contain information about syntactic structure that can inform shallow syntactic parsing. To test this hypothesis, we explore labels derived from keystroke logs as auxiliary task in a multi-task bidirectional Long Short-Term Memory (bi-LSTM). Our results show promising results on two shallow syntactic parsing tasks, chunking and CCG supertagging. Our model is simple, has the advantage that data can come from distinct sources, and produces models that are significantly better than models trained on the text annotations alone.

1 Introduction

As people produce text, they unconsciously produce loads of cognitive side benefit such as keystroke logs, brain activations or gaze patterns. However, natural language processing (NLP) hitherto almost exclusively relied on the written text itself. We argue that cognitive processing data contains potentially useful information beyond the linguistic signal and propose a novel source of information for shallow syntactic parsing, keystroke logs.

Keystroke dynamics concerns a user's typing pattern. When a person types, the latencies between successive keystrokes and their duration reflect the unique typing behavior of a person. Keystroke logs, the recordings of a user's typing dynamics, are studied mostly in cognitive writing and translation process research to gain insights into the cognitive load involved in the writing process. However, until now this source has not yet been explored to inform NLP models.

Very recent work has shown that cognitive processing data carries valuable signal for NLP. For instance, eye tracking data can inform sentence compression (Klerke et al., 2016) and gaze is predictive for part-of-speech (Barrett and Søgaard, 2015; Barrett et al., 2016).

Keystroke logs have the distinct advantage over other cognitive modalities like eye tracking or brain scanning, that they are readily available and can be harvested easily, because they do not rely on any special equipment beyond a keyboard. Moreover, they are non-intrusive, inexpensive, and have the potential to offer continuous adaptation to specific users. Imagine integrating keystroke logging into (online) text processing tools.

We hypothesize that keystroke logs carry syntactic signal. Writing time between words can be seen as proxy of the planning process involved in writing, and thus represent structural information between words. To test our hypothesis, we evaluate a multi-task bidirectional Long-Short Term Memory (bi-LSTM) model that is—to the best of our knowledge—the first to exploit keystroke logs to improve NLP models. We test our model on two shallow syntactic tasks, chunking and CCG supertagging. The choice of tasks is motivated by the fact that writing research analyzes so-called *bursts* of writing (i.e., consecutive spans of text, cf. Section 2.2), which are related to shallow syntactic annotation.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 609–619, Osaka, Japan, December 11-17 2016.

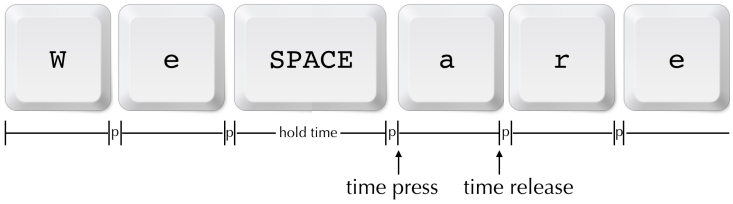


Figure 1: Keystroke logging. p are pauses between keystrokes.

To exploit the keystroke log information we model it as auxiliary task in a multi-task setup (cf. Section 3). This setup has the advantage that the syntactic data and keystroke information can come from *distinct* sources, thus we are not restricted to the requirement of jointly labeled data (a corpus with both annotations). Our exploratory evaluation shows that little keystroke data suffices to improve a syntactic chunker on out-of-domain data, and that keystrokes also aid CCG tagging.

Contributions We are the first to use keystroke logs as signal to improve NLP models. In particular, the contributions of this paper are the following: i) we present a novel bi-LSTM model that exploits keystroke logs as auxiliary task for syntactic sequence prediction tasks; ii) we show that our model works well for two tasks, syntactic chunking and CCG supertagging, and iii) we make the code available at: `https://github.com/bplank/coling2016ks`.

2 Keystroke dynamics

We see keystroke dynamics as providing a complementary view on the data beyond the linguistic signal, which can be harvested easily and is particularly attractive to build robust models for out-of-domain setups. Keystroke logging data can be seen as an instance of *fortuitous data* (Plank, 2016); it is side benefit of behavior that we want to exploit here. However, keystroke log is raw data, thus first needs to be *refined* before it can be used. Our idea is to treat the duration of pauses before words as a simple sequence labeling problem.

We first describe the process of obtaining auto-labeled data from raw keystroke logs, and then provide background and motivation for this choice. Section 3 then describes our model, i.e., by solving the keystroke sequence labeling problem jointly with shallow syntactic parsing tasks (chunking and CCG supertagging) we want to aid shallow parsing.

2.1 From keystroke logs to auxiliary labels

While keystroke dynamics considers a number of timing metrics, such as *holding time* and *time press* and *time release* between every keystroke (p in Figure 1), in this study we are only concerned with the pause preceding a word (i.e., the third p in Figure 1).[1] We here use a simple tokenization scheme. Whitespace delimits tokens, punctuation delimits sentence boundaries.

An example of pre-word pauses (in the remainder simply called *pauses*) calculated from our actual keylog data is shown in Table 1. If we take an arbitrary threshold of 500ms, the chunks indicated by the brackets are derived. This affirms that pre-word pauses carry constituency-like information.

However, typing behavior of users differs, as illustrated in Figure 2. Hence, rather than finding a global metric we rely on per-user calculated aggregate statistics and discretize them to obtain auto-derived labels, as explained next.

We calculate p, the pause duration before a token, and bin it into the following categories, using BIO encoding, where $median$ is the per-user median and mad the median absolute deviation. In this way, we automatically gather labels from keystrokes representing pause durations.

In particular, we use the following discretization, i.e., a label for a token is calculated by:

[1] Figure inspired by the figure in (Goodkind and Rosenberg, 2015).

Token:	[Coefficient	of	determination]	[is	a]	[measure	used	in]	[statisitcal	model]	[analysis]
Pause (ms):	0	96	496	30769	96	2144	96	80	2975	240	680

Table 1: Example keystroke log for user 33 (including typo). If we segment the data using an arbitrary 500ms pre-word pause the chunks indicated by the brackets are obtained. To normalize over idiosyncrasies of users we use per-user average statistics to obtain segments with auto-derived labels, see Section 2.1.

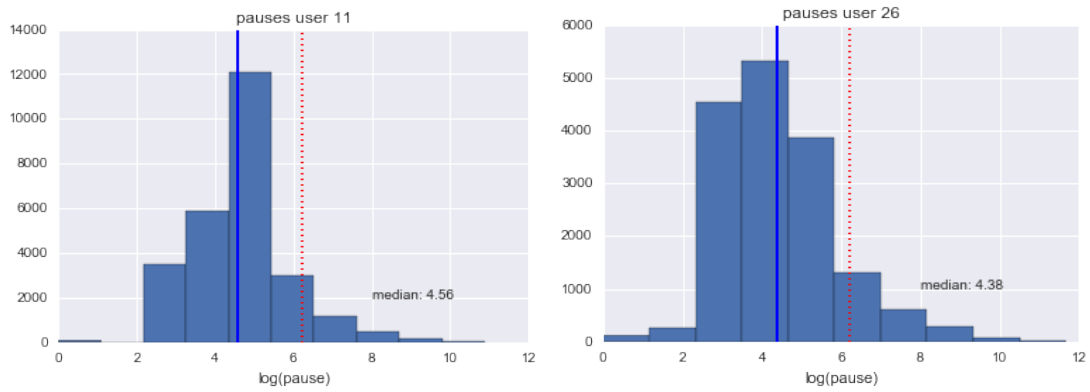


Figure 2: Distribution of pauses for two users (plotted in log space). Red solid line: per-user median pause. Dotted line: arbitrary 500ms threshold. As can be seen from the plots, the users' typing dynamics differs.

$$
\begin{array}{lll}
label = & \texttt{<m} & \text{if } p < median; \\
& \texttt{<m+.5} & \text{if } p < median + 0.5 * mad; \\
& \texttt{<m+1} & \text{if } p < median + mad; \\
& \texttt{>m1} & \text{else;} \\
& \texttt{O} & \text{for punctuation symbols.}
\end{array}
$$

The label is further enriched with a prefix in BIO encoding style, motivated by the fact that we want to model spans of information. Punctuation symbols are treated as `O`, because due to their location at boundary positions the pause information varies highly. We leave treating punctuation separately as future work. Klerke et al. (2016) use a related encoding scheme to discretize fixation durations obtained from eye tracking data, however, in contrast to them we here use median-based measures which are better suited for such highly skewed data (Leys et al., 2013). An actual example of automatically labeled keystroke data is given in Table 2.

B-<m	B-<m+1	B-<m	I-<m	B-<m+.5	I-<m+.5	B->m+1
the	closer	the	number	is	to	1

Table 2: Example auto-derived keystroke annotation.

2.2 Background

The major scientific interest in keystroke dynamics is that it provides a non-intrusive method for studying cognitive processes involved in writing. Keystroke logging has developed to a promising tool in writing research (Sullivan et al., 2006; Nottbusch et al., 2007; Wengelin, 2006; Van Waes et al., 2009; Baaijen et al., 2012), where time measurements—pauses, bursts and revisions (described below)—are studied as traces of the recursive nature of the writing process.

In its raw form, keystroke logs contain information on which key was pressed for how long (key, time press, time release). This data is then used to calculate between keystroke pause durations, such as pre-word pauses. It has been shown that pauses reflect the planning of the unit of text itself (Baaijen

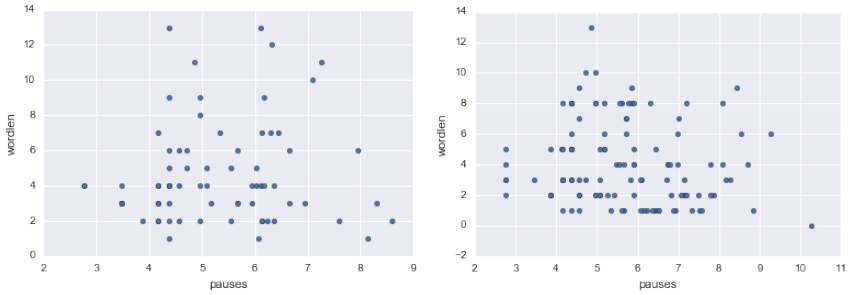


Figure 3: left: Word pauses length vs word length (left: user 7, right: user 3; Pearson $\rho = 0.08$, and $\rho = -0.12$)

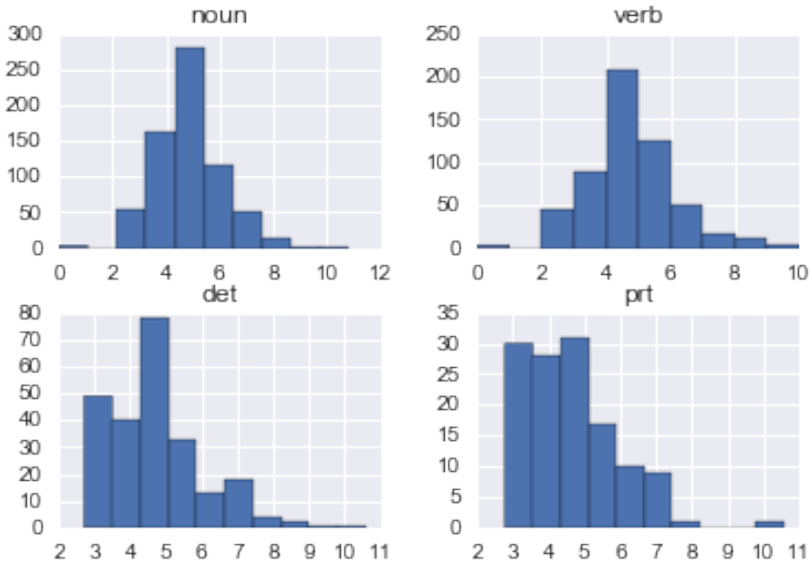


Figure 4: Word pause length distribution per part-of-speech (user 5).

et al., 2012) and that they correlate with clause and sentence boundaries (Spelman Miller and Sullivan, 2006). Writing research is interested in *bursts* of writing, defined as consecutive chunks of text produced and defined by a 2000ms time of inactivity (Wengelin, 2006), or revisions. Such a cutoff is rather arbitrary (Baaijen et al., 2012), and from our own experience results in long chunks. Taking writing research as a starting point, we postulate that keystrokes contain further fine-grained information that help identify syntactic chunks. We aim at a finer-grained representation, and transform user-based average statistics into automatically derived labels (cf. above).

We notice that the literature defines different ways to define a pause. Goodkind and Rosenberg (2015), coming from a stylometry background, use the difference between release time of the previous key and the timepress of the current key to calculate pre-word pause duration.[2] In contrast, writing research (Wengelin, 2006; Van Waes et al., 2009; Baaijen et al., 2012) defines pauses as the start time of a keystroke until the start time of the next keystroke. We experimented with both types of pause definitions, and found the former slightly more robust, hence we use that to calculate pauses throughout this paper.

In order to get a better feel of word pause durations, we examine various properties of them. First, do we need to normalize pauses for word length? Goodkind and Rosenberg (2015) found a linear relationship between pre-word pauses and word length in their dataset. We calculated the correlation between word length and pauses in our dataset, but could not observe such a relation in our data (cf.

[2]Goodkind sets negative pause durations (which can arise in this setup) to 0 (personal communication).

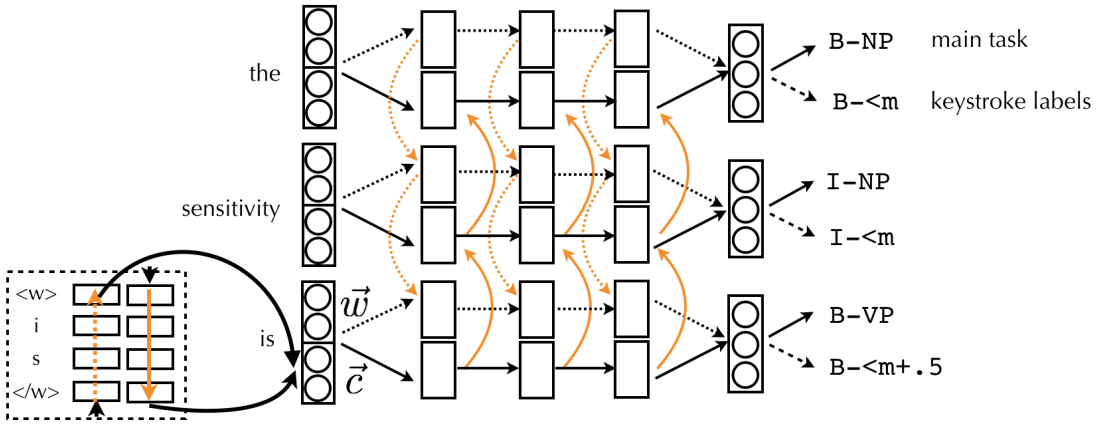


Figure 5: Hierarchical Bi-LSTM with 3 stacked layers using word $\vec{w}$ and characters $\vec{c}$ embeddings.

Figure 3; plots for the other participants looks similar). Even if we break the data down by POS and calculate per-POS wise correlations we found no relation between pause duration and word length.[3] Hence we do not normalize word pause durations. In addition, Figure 4 plots pauses for various part-of-speech, showing that function POS (determiner, particles) are preceded by shorter pauses than content POS (we obtain similar plots for other participants).

Second, keystroke logs are presumably idiosyncratic, can we still use it? In fact, user keystroke biometrics are successfully used for author stylometry and verification in computer security research (Stewart et al., 2011; Monaco et al., 2013; Locklear et al., 2014). However, also eye tracking data like scanpaths (the resulting series of fixations and saccades in eye tracking) are known to be idiosyncratic (Kanan et al., 2015). Nevertheless it has been shown that gaze patterns help to inform NLP (Barrett and Søgaard, 2015; Klerke et al., 2016). We believe this is also the case for biometric keystroke logging data.

3 Tagging with bi-LSTMs

We draw on the recent success of bi-directional recurrent neural network (bi-RNNs) (Graves and Schmidhuber, 2005), in particular Long Short-Term Memory (LSTM) models (Hochreiter and Schmidhuber, 1997). They read the input sequences twice, in both directions. Bi-LSTM have recently successfully been used for a variety of tasks (Collobert et al., 2011; Ling et al., 2015; Wang et al., 2015; Huang et al., 2015; Dyer et al., 2015; Ballesteros et al., 2015; Kiperwasser and Goldberg, 2016; Liu et al., 2015). For further details, see Goldberg (2015) and Cho (2015).

3.1 Bidirectional Long-Short Term Memory Models

Our model is a a hierarchical bi-LSTM as illustrated in Figure 5. It takes as input word embeddings $\vec{w}$ concatenated with character embeddings obtained from the last two states (forward, backward) of running a lower-level bi-LSTM on the characters. Adding character representations as additional information has been shown to be effective for a number of tasks, including parsing and tagging (Ballesteros et al., 2015; Gillick et al., 2015; Plank et al., 2016).

In more detail, our model is a context bi-LSTM taking as input word embeddings $\vec{w}$. Character embeddings $\vec{c}$ are incorporated via a hierarchical bi-LSTM using a sequence bi-LSTM at the lower level (Ballesteros et al., 2015; Plank et al., 2016). The character representation is concatenated with the (learned) word embeddings $\vec{w}$ to form the input to the context bi-LSTM at the upper layers.

For the hidden layers, we use stacked LSTMs with h=3 layers. The 3-layer bi-LSTM and lower-level character bi-LSTM represents the shared structure between tasks. From the topmost (h=3) layer labels for the different tasks (e.g., chunking, pauses) are predicted using a softmax. In Figure 5, the main

[3]POS annotations were obtained by looking up the possible tag of a token in English wiktionary (Li et al., 2012).

task (chunking or CCG tagging) is represented by the solid arrow, the auxiliary task (keystroke logs) is indicated by the dashed arrow.

During training, we randomly sample a task and instance, and backpropagate the loss of the current instance through the shared deep network. In this way, we learn a joint model from distinct sources. Note that we also experimented with predicting the pause durations at lower levels (h=1), motivated by having lower-level tasks at lower layers in the network (Søgaard and Goldberg, 2016), however, we found the setup with both tasks at the outer layer more robust. Predicting all tasks at the outermost layer is the most commonly used form of multi-task learning in neural networks (Caruana, 1998; Collobert et al., 2011).

4 Experiments

We implement our model in `CNN/pycnn`.[4] For all experiments, we use the same hyperparameters, set on a held-out portion of the CoNLL 2000 data, i.e., SGD with cross-entropy loss, no mini-batches, 30 epochs, default learning rate (0.1), 64 dimensions for word embeddings, 100 for character embeddings, random initialization for all embeddings, 100 hidden states, $h = 3$ stacked layers, Gaussian noise with σ=0.2. As training is stochastic, we use a fixed seed throughout (chosen and fixed upfront). No further unlabeled data is considered.

Datasets An overview of the syntactic datasets considered in this paper is given in Table 3. For chunking, we use the original CoNLL data (Tjong Kim Sang and Buchholz, 2000) from WSJ (WSJ sections 15-18 as training data and section 20 as test data, containing 8936 and 2012 sentences, respectively).[5] For testing we take out-of-domain data whenever available, to test the adaptability of the method to noisy out-of-domain data. For chunking we use Twitter data from Ritter (2011) (all, 2364 tweets) and Foster et al. (2011) (250 sentences), converted to chunks (Plank et al., 2014).

The CCG supertagging data also comes from WSJ (39604 training and 2407 test sentences). We unfortunately do not have access to out-of-domain test data, hence use the CCG tagging test set.

sentences	TRAIN	DEV	TEST
CONLL 2000	8936	–	2012
FOSTER	–	269	250
RITTER	–	–	2364
CCG	39604	1913	2407

Table 3: Statistics on the data sets

The keystroke logging data stems from students taking an actual test on spreadsheet modeling in a university course (Stewart et al., 2011; Monaco et al., 2013). The advantage of this dataset is that it contains free-text input.[6] We used data from 38 users,[7] which produced on average 250 sentences. The data totals to 7699 sentences.

To evaluate our models we use standard evaluation measures computed with `conlleval.pl` with default parameters, i.e., we report F1 on chunks and accuracy on CCG tags. Statistical significance is computed using the approximate randomization test (Noreen, 1989) using $i = 1000$ iterations and p-values are reported (Søgaard et al., 2014).

4.1 Results

Baseline model Both or baseline models are comparable to prior work, while being simpler. The results are summarized in Table 4. Our chunking baseline achieves an $F1$ of 93.21 on CoNLL, compared to the $F1$ of 93.88 of Suzuki and Isozaki (2008), who use a CRF and gold POS tags. We do not use any POS information. A similar bi-LSTM achieves 93.64 (Huang et al., 2015), however, additionally uses

[4] `https://github.com/clab/cnn`

[5] `http://www.cnts.ua.ac.be/conll2000/chunking/`

[6] In contrast to `http://www.casmacat.eu/` data that logs revisions from MT post-editing.

[7] Disregarding users due to issues with logging (Stewart et al., 2011).

Chunking	F1	CCG tagging	Accuracy
Our model	93.21	Our model	92.41
Suzuki and Isozaki (2008)	93.88	Xu et al. (2015)	93.00

Table 4: Baseline model, comparison to existing systems

POS embeddings. Our baseline CCG supertagging model achieves 92.41, compared to the more complex model by Xu et al. (2015) achieving an accuracy of 93.00. Very recently even higher accuracies were reported, e.g. (Vaswani et al., 2016), however, in this exploratory paper we are interested in examining whether we find signal in keystroke data, and are not interested in beating the latest state-of-the-art.

	FOSTER.DEV	FOSTER.TEST	RITTER	CCG
Baseline	73.93	73.61	66.65	92.41
+PAUSE	**74.63**†	**74.32**†	**66.91**†	**92.62**†
p-values	$<$0.01	$<$0.01	$<$0.01	$<$0.048

Table 5: Chunking results (F1, +Pause is average over 38 participants) and CCG accuracy (using all pause data at once). Results marked with † are significantly better than the corresponding baseline using a randomization test with $i = 1000$ iterations; p-values provided in row below.

Keystroke pauses The aim of our experiments is to gauge whether through joint learning of shallow syntax and pause duration the system learns to generalize over the pause information and thus aids the syntactic signal.

The results in Table 5 support our hypothesis that keystroke dynamics contains useful information for chunking. We here report the average over models trained on a per-user basis, i.e., 38 participants. The results show that overall F1 chunking score improves over all datasets. For instance on the Ritter data, for 25/38 participants using their keystroke information as auxiliary task helps to improve overall chunking performance. However, if we combine all data and train a single model, performance degrades on chunking. We attribute this effect to the fact that the chunking data is relatively small, and higher amounts of keystroke data show signs of overfitting. In fact, similar effects have been shown in a multi-task machine translation and parsing setup (Luong et al., 2016), where mixing coefficients were used to downplay the importance of the auxiliary parsing data that otherwise swamped the main task data. We leave examining task-specific weights for the loss for future work.

In contrast in CCG tagging, where we have more training data, we see a positive effect of using keystroke data when training a model that uses all keystroke data at once (concatenation of all keystroke data from all users), see last column in Table 5. Note that all results in Table 5 are significant.

		FOSTER.DEV	FOSTER.TEST	RITTER
Baseline	NP	72.18	71.41	61.76
	VP	70.25	73.44	75.13
	PP	93.25	91.85	89.05
+PAUSE	NP	**73.99**	**72.77**	**62.60**
	VP	69.88	**74.93**	75.05
	PP	93.24	90.82	88.87

Table 6: Chunking results per label.

LABEL	BASELINE	+PAUSE
N	97.20	**97.27**
N/N	96.38	**96.62**
NP[nb]/N	99.02	**99.03**
(NP\NP)/NP	88.41	**88.95**
NP	97.06	**97.63**
PP/NP	72.60	**73.64**
((S\NP)\(S\NP))/NP	72.83	**74.07**
conj	98.62	**98.67**

Table 7: CCG tagging results per label (most frequent non-punctuation labels shown).

5 Discussion

To gain better insights into what the model has learned, Table 6 provides the per-label breakdown for chunking, Table 7 for CCG tagging. Most of the improvements come from noun phrases (NP) chunks. From manual inspection we determine that the model improves particularly on non-conventional spelling and fragmented noun phrases typical for Twitter, see examples given in Table 8.

As Table 6 shows, keystroke data also helps for verb phrases on one dataset. The current encoding is not so beneficial for PPs. Pauses before prepositions are short, as illustrated in Figure 4, and pauses often fall within segments in the auxiliary annotation, while prepositions constitute separate tokens in chunking. Hence, it is unsurprising that the model fares worse on PPs.

We believe that our pause encoding mainly captures structural information between words, less morphosyntactic information itself, i.e., that pauses are more informative of syntactic structure than of part-of-speech. This intuition is in fact confirmed by initial experiments on POS tagging (UD English), which are less promising. We observe small improvements for low amounts of auxiliary data, however, they are not significant. Thus keystrokes seem to capture mostly structural shallow syntactic information, as confirmed in our experimental evaluation. However, this is only a first exploration, with one way of using keystroke logging data, but given our promising results, further experiments are warranted.

TOKEN	GOLD	BASELINE	MODEL
Auburn	B-NP	I-NP	B-NP
party	I-NP	I-NP	I-NP
at	B-PP	B-PP	B-PP
Spurs	B-NP	B-NP	B-NP
v	B-NP	I-NP	B-VP
Man	B-NP	I-NP	B-NP
utd	I-NP	B-VP	I-NP
sounds	B-VP	B-VP	B-VP
bithcy	B-ADJP	B-VP	B-ADJP

Table 8: Selected examples from the Twitter datasets.

6 Related Work

Keystroke logging has developed into a promising tool for research into writing (Wengelin, 2006; Van Waes et al., 2009; Baaijen et al., 2012), as time measurements can give insights into cognitive processes involved in writing (Nottbusch et al., 2007) or translation studies. In fact, most prior work that uses keystroke logs focuses on experimental research. For example, Hanoulle et al. (2015) study whether a bilingual glossary reduces the working time of professional translators. They consider pause

durations before terms extracted from keystroke logs and find that a bilingual glossary in the translation process of documentaries reduces the translators' workload. Other translation research has combined eye-tracking data with keystroke logs to study the translation process (Carl et al., 2016). An analysis of users' typing behavior was studied by Baba and Suzuki (2012). They collect keystroke logs of online users describing images to measure spelling difficulty. They analyzed corrected and uncorrected spelling mistakes in Japanese and English and found that spelling errors related to phonetic problems remain mostly unnoticed.

Goodkind and Rosenberg (2015) is the only study prior to us that use keystroke loggings in NLP. In particular, they investigate the relationship between pre-word pauses and multi-word expressions and found within MWE pauses vary depending on cognitive task. We take a novel approach and *learn* keystroke patterns and use them to inform shallow syntactic parsing.

A recent related line of work explores eye tracking data to inform sentence compression (Klerke et al., 2016) and induce part-of-speech (Barrett and Søgaard, 2015). Similarly, there are recent studies that predict fMRI activation from reading (Wehbe et al., 2014) or use fMRI data for POS induction (Bingel et al., 2016). The distinct advantage of keystroke dynamics is that it is easy to get, non-expensive and non-intrusive.

7 Conclusions

Keystroke dynamics contain useful information for shallow syntactic parsing. Our model, a bi-LSTM, integrates keystroke data as auxiliary task, and outperforms models trained on the linguistic signal alone. We obtain promising results for two syntactic tasks, chunking and CCG supertagging. This warrants many directions for future research, e.g., using information from the non-linear writing process, which we here disregarded (e.g., revisions), evaluating on other languages and going to the full parsing task.

Acknowledgements

I would like to thank Veerle Baaijen for insightful discussions, and the three anonymous reviewers as well as Héctor Martínez Alonso, Maria Barrett and Zeljko Agić for comments on earlier drafts of this paper. I acknowledge the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high performance computing (HPC) cluster, as well as NVIDIA corporation for supporting my research.

References

Veerle M Baaijen, David Galbraith, and Kees de Glopper. 2012. Keystroke analysis: Reflections on procedures and measures. *Written Communication*, page 0741088312451108.

Yukino Baba and Hisami Suzuki. 2012. How are spelling errors generated and corrected?: a study of corrected and uncorrected spelling errors using keystroke logs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *EMNLP*.

Maria Barrett and Anders Søgaard. 2015. Using reading behavior to predict grammatical functions. In *Workshop on Cognitive Aspects of Computational Language Learning*.

Maria Barrett, Joachim Bingel, and Anders Søgaard. 2016. Part-of-speech induction from eye-tracking data. In *ACL*.

Joachim Bingel, Maria Barrett, and Anders Søgaard. 2016. Part-of-speech induction from fmri. In *ACL*.

Michael Carl, Isabel Lacruz, Masaru Yamada, and Akiko Aizawa. 2016. Measuring the translation process. In *The 22nd Annual Meeting of the Association for Natural Language Processing. NLP 2016*.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Kyunghyun Cho. 2015. Natural language understanding with distributed representation. *ArXiv*, abs/1511.07916.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *ACL*.

Jennifer Foster, Ozlem Cetinoglu, Joachim Wagner, Josef Le Roux, Joakim Nivre, Deirde Hogan, and Josef van Genabith. 2011. From news to comments: Resources and benchmarks for parsing the language of Web 2.0. In *IJCNLP*.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2015. Multilingual language processing from bytes. *arXiv*.

Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *ArXiv*, abs/1510.00726.

Adam Goodkind and Andrew Rosenberg. 2015. Muddying the multiword expression waters: How cognitive demand affects multiword expression production. In *Proceedings of MWE 2015*.

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

Sabien Hanoulle, Véronique Hoste, and Aline Remael. 2015. The translation of documentaries: Can domain-specific, bilingual glossaries reduce the translators' workload? an experiment involving professional translators. *New Voices in Translation Studies*, 23(13).

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Christopher Kanan, Dina NF Bseiso, Nicholas A Ray, Janet H Hsiao, and Garrison W Cottrell. 2015. Humans have idiosyncratic and task-specific scanpaths for judging faces. *Vision research*, 108:67–76.

E. Kiperwasser and Y. Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *ArXiv e-prints*.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *NAACL*.

Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. 2013. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766.

Shen Li, João Graça, and Ben Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *EMNLP*.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *EMNLP*.

Hilbert Locklear, Sathya Govindarajan, Zdenka Sitova, Adam Goodkind, David Guy Brizan, Andrew Rosenberg, Vir V Phoha, Paolo Gasti, and Kiran S Balagani. 2014. Continuous authentication with cognition-centric text production and revision features. In *Biometrics (IJCB), 2014 IEEE International Joint Conference on*.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR*.

John V Monaco, John C Stewart, Sung-Hyuk Cha, and Charles C Tappert. 2013. Behavioral biometric verification of student identity in online course assessment and authentication of authors in literary works. In *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, pages 1–8. IEEE.

Eric Noreen. 1989. *Computer-intensive methods for testing hypotheses: an introduction*. Wiley.

Guido Nottbusch, Rdiger Weingarten, and Said Sahel. 2007. From written word to written sentence production. *Writing and cognition: Research and applications.*, pages 31–54.

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014. Learning part-of-speech taggers with inter-annotator agreement loss. In *EACL*.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *ACL*.

Barbara Plank. 2016. What to do about non-standard (or non-canonical) language in NLP. In *KONVENS*.

Alan Ritter. 2011. *Extracting Knowledge from Twitter and The Web*. Ph.D. thesis, University of Washington.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*.

Anders Søgaard, Anders Johannsen, Barbara Plank, Dirk Hovy, and Héctor Martínez Alonso. 2014. What's in a p-value in nlp? In *CoNLL*.

Kristyan Spelman Miller and Kirk PH Sullivan. 2006. *Keystroke logging: an introduction*. Elsevier.

John C Stewart, John V Monaco, Sung-Hyuk Cha, and Charles C Tappert. 2011. An investigation of keystroke and stylometry traits for authenticating online test takers. In *Biometrics (IJCB), 2011 International Joint Conference on*, pages 1–7. IEEE.

Kirk PH Sullivan, Eva Lindgren, et al. 2006. *Computer keystroke logging and writing: Methods and applications*. Elsevier.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *ACL*.

Erik Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *CoNLL*.

Luuk Van Waes, Mariëlle Leijten, and Daphne Van Weijen. 2009. Keystroke logging in writing research: Observing writing processes with inputlog. *GFL-German as a foreign language*, 2(3):41–64.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging with lstms. In *NAACL*.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *pre-print*, abs/1510.06168.

Leila Wehbe, Ashish Vaswani, Kevin Knight, and Tom Mitchell. 2014. Aligning context-based statistical models of language with brain activity during reading. In *EMNLP*.

Åsa Wengelin. 2006. Examining pauses in writing: Theory, methods and empirical data. *Computer key-stroke logging and writing: methods and applications (Studies in Writing)*, 18:107–130.

Wenduan Xu, Michael Auli, and Stephen Clark. 2015. Ccg supertagging with a recurrent neural network. In *ACL*.

A Bayesian model for joint word alignment and part-of-speech transfer

Robert Östling
Department of Modern Languages, University of Helsinki
Department of Linguistics, Stockholm University
robert.ostling@helsinki.fi, robert@ling.su.se

Abstract

Current methods for word alignment require considerable amounts of parallel text to deliver accurate results, a requirement which is met only for a small minority of the world's approximately 7,000 languages. We show that by jointly performing word alignment and annotation transfer in a novel Bayesian model, alignment accuracy can be improved for language pairs where annotations are available for only one of the languages—a finding which could facilitate the study and processing of a vast number of low-resource languages. We also present an evaluation where our method is used to perform single-source and multi-source part-of-speech transfer with 22 translations of the same text in four different languages. This allows us to quantify the considerable variation in accuracy depending on the specific source text(s) used, even with different translations into the same language.

1 Introduction

Word alignment is the problem of identifying translationally equivalent words across the languages of a parallel text. It has found widespread use for enabling applications such as statistical machine translation (Brown et al., 1993; Koehn et al., 2003), annotation transfer (Yarowsky et al., 2001), word sense disambiguation (Diab and Resnik, 2002) and lexicon extraction (Wu and Xia, 1994).

Although many types of algorithms have been explored, the main line of research through the last couple of decades has been based on the generative IBM models introduced by Brown et al. (1993). What these models have in common is that they are unsupervised, asymmetric models, assuming one of the languages in a bitext (the *source language*) generates the corresponding text in the other language (the *target language*), word by word.

Most often, a variant of the Expectation-Maximization algorithm (Dempster et al., 1977) has been used for inference in these models, but recently there has been some work using Bayesian alignment models using Gibbs sampling for inference (DeNero et al., 2008; Mermer and Saraçlar, 2011; Gal and Blunsom, 2013). The incorporation of Bayesian priors into these models has been shown to improve accuracy, since they provide a flexible way of biasing the model towards empirical observations about language, most importantly that a given word type tends to have a very limited number of translations.

While the basic word alignment models use only lexical co-occurrence and word order, lexical data tends to be sparse and a number of authors have explored the usefulness of other information sources. Toutanova et al. (2002) showed that Part of Speech (PoS) tags can be integrated into the IBM models to improve word alignment accuracy, and others have reported similar results for dependency (Cherry and Lin, 2003; Wang and Zong, 2013) and phrase-structure (Yamada and Knight, 2001) parse trees, and for lemmatized texts (Bojar and Prokopov, 2006).

In addition to the studies just mentioned that showed how various types of linguistic annotation can be used to guide word alignment, there has been research showing that the reverse also holds: word-aligned parallel texts can be used to transfer annotations and models from languages where those resources exist to languages where they do not. Pioneering work by Yarowsky et al. (2001) explored tasks such as PoS

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 620–629, Osaka, Japan, December 11-17 2016.

tagging, shallow parsing and lemmatization, which was followed by e.g. dependency parsing (Hwa et al., 2005).

The present work combines these previous lines of work by exploring joint models of word alignment and annotation transfer (of PoS tags), within a Bayesian framework. The source code of our implementation is available at `http://www.ling.su.se/spacos`.

2 Methods

This section first discusses Bayesian word alignment using IBM-based models along with extensions to these, and finally describes our model of joint PoS transfer and word alignment.

2.1 Bayesian IBM models

The IBM 1 alignment model can be extended with sparse Dirichlet priors, and efficient inference is possible using Gibbs sampling (Mermer and Saraçlar, 2011; Mermer et al., 2013; Gal and Blunsom, 2013) or Variational Bayesian techniques (Riley and Gildea, 2012).

IBM model 1 assumes each target word $t_j = f$ of a sentence is generated by one source word $s_{a_j} = e$ through the alignment variable a_j, and that all words are generated independently and do not depend on the sentence positions i and j. The probability of a target sentence $\boldsymbol{t}$ (of length J) and an alignment $\boldsymbol{a}$ given a source sentence $\boldsymbol{s}$ (of length I) then becomes

$$P(\boldsymbol{t}, \boldsymbol{a}|\boldsymbol{s}) \propto p(J|I) \prod_{j=1}^{J} P(t_j|s_{a_j}) \tag{1}$$

One drawback of this model (apart from the extreme independence assumptions addressed by later IBM models) is that there is no penalty for having very flat distributions $P(f|e)$ for target words conditioned on a source word, a fact that causes the so-called *garbage collection effect* where rare source words are incorrectly linked to a large number of target words. By using priors on the translation distributions that discourage such solutions, it is possible to improve alignment accuracy. Mermer and Saraçlar (2011) introduced the use of Dirichlet priors for this task. If the Dirichlet parameter α is 1, this reduces to the uniform distribution, but it turns out that by using much smaller values of α, below about 10^{-2} (Riley and Gildea, 2012, Figure 1), the model better reflects the empirical observation that words tend to have very few possible translations.

2.2 Inference

For the standard IBM models, the EM algorithm is normally used for inference. In the Bayesian version with Dirichlet priors, we mentioned above that two main options have been investigated: Variational Bayes and Gibbs sampling. While both methods have been shown to improve word alignment accuracy for IBM model 1, the computational complexity of Gibbs sampling is lower with more complex models (Östling and Tiedemann, 2016, Section 3.2). For this reason, Gibbs sampling is used in the present work and will be discussed in further detail.

Gibbs sampling (Gelfand and Smith, 1991) is a specific instance of the more general Markov Chain Monte Carlo algorithm, which is used to draw samples from a model M which defines a probability function $p_M(\boldsymbol{x})$ over the variable space $\boldsymbol{x}$. This is done by constructing a Markov chain with p_M as its stationary distribution and performing a sufficiently long random walk in it. A Gibbs sampler achieves this by specifying for each variable x_i in $\boldsymbol{x}$ a sampling distribution $P(x_i = a|\boldsymbol{x}_{-i})$ for x_i conditioned on $\boldsymbol{x}_{-i}$, which denotes all variables in $\boldsymbol{x}$ except x_i. For IBM model 1, this gives the following sampling equation, which we also use, and for more complex models extend with additional factors given Equation (4):

$$P(a_j = i) = \frac{n_{\boldsymbol{a}_{-j}, s_i, t_j} + \alpha_{t_j}}{\sum_f (n_{\boldsymbol{a}_{-j}, s_i, f} + \alpha_f)} \tag{2}$$

Here, $n_{\boldsymbol{a}_{-j}, e, f}$ is a count vector representing the number of times each source type e is aligned to each target type f under the alignment $\boldsymbol{a}$, not counting the alignment at position j. In the end, we are interested

in computing the expectations $\mathbb{E}\left[\delta_{a_j,i}\right]$ under the alignment model, where δ is the Kronecker delta. Given a series of samples of $\boldsymbol{a}^{(t)}$ for $t \in 1 \ldots T$, we approximate this using

$$\mathbb{E}\left[\delta_{a_j,i}\right] \approx \frac{1}{T} \sum_{t=1}^{T} P(a_j = i | \boldsymbol{a}_{-j}^{(t)}, \boldsymbol{s}, \boldsymbol{t}) \tag{3}$$

The initial alignments $\boldsymbol{a}^{(0)}$ are sampled from a uniform distribution, and in order to reduce initialization bias we average the marginals from eight independently initialized samplers. For details on the tradeoffs involved in choosing the number of samplers and sampling iterations, we refer to Table 2 of Östling and Tiedemann (2016).

2.3 Word order and fertility

Even with good prior parameters, IBM model 1 is a poor model of word alignment because it ignores two important characteristics of parallel text: word order and morpheme counts. While different distortion models have been used to model word order, we use the HMM-based model of Vogel et al. (1996), which has been demonstrated to deliver better performance than either no distortion model (like IBM model 1) or models based on absolute sentence positions (IBM models 2 and 3). This introduces a distribution $P(a_j - a_{j-1}|I)$ of the "jump" $a_j - a_{j-1}$ in the source sentence when moving one step in the target sentence.

As a way of modeling the relative number of morphemes in a word for a pair of languages, the *fertility* of a source word e is defined as the number of target words it is aligned to in a particular context. This is modeled using a distribution $P(\phi_i = k|s_i = e)$, where the fertility ϕ_i at position i is conditioned on the word e at that position. This is particularly important when the languages have large differences in word formation strategies and the general level of morphological complexity.

Zhao and Gildea (2010) explored a model with a word order and fertility model as described above, but based their work on the EM algorithm, using Gibbs sampling only for approximating the expectations. An important conclusion from their work is that a simple HMM with fertility model is competitive with the more complex IBM model 4, and we follow them in using this model as our baseline. Our full baseline model is given by

$$\begin{aligned} &P\big(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{a}, \boldsymbol{\theta}, \boldsymbol{\psi}, \boldsymbol{\pi}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}\big) \\ &\quad \propto \left(\prod_{k=1}^{K} \prod_{j=1}^{J^{(k)}} \theta_{s_{a_j^{(k)}}^{(k)}, t_j^{(k)}} \right) \cdot \left(\prod_{e=1}^{E} \prod_{f=1}^{F} \theta_{e,f}^{\alpha_f - 1} \right) \\ &\quad \cdot \left(\prod_{k=1}^{K} \prod_{j=1}^{J^{(k)}+1} \psi_{a_j^{(k)} - a_{j-1}^{(k)}} \right) \cdot \left(\prod_{I=I_{min}}^{I_{max}} \prod_{m=m_{min}}^{m_{max}} \psi_m^{\beta_{I,m}-1} \right) \\ &\quad \cdot \left(\prod_{k=1}^{K} \prod_{i=1}^{I^{(k)}} \pi_{s_i^{(k)}, \phi_i} \right) \cdot \left(\prod_{e=1}^{E} \prod_{n=0}^{n_{max}} \pi_{e,n}^{\gamma_n - 1} \right) \end{aligned} \tag{4}$$

where K is the number of parallel sentences, $\boldsymbol{\theta} \sim \mathrm{Dir}\big(\boldsymbol{\alpha}\big)$ are the lexical translation parameters, $\boldsymbol{\psi} \sim \mathrm{Dir}\big(\boldsymbol{\beta}\big)$ are the categorical distribution parameters for the word order model $P\big(a_j - a_{j-1} = m|I\big)$, and $\boldsymbol{\pi}_e \sim \mathrm{Dir}\big(\boldsymbol{\gamma}\big)$ for the fertility model $P(\phi_i = k|s_i = e)$.

2.4 PoS-guided word alignment

This work follows Toutanova et al. (2002) in adding another factor to the model, akin to the lexical translation probability $P(f|e)$ but using the PoS tags of the respective words, $P(\boldsymbol{T}_f^t|\boldsymbol{T}_e^s)$. The main difference is that in their work PoS tags for both source and target languages were assumed, whereas here only one of the languages is assumed to be PoS-annotated. For the other language, the PoS tags are sampled using the method described below.

Algorithm 1 Alternating alignment-annotation.

▷ Align a single sentence pair $\boldsymbol{s}, \boldsymbol{t}$. The extension to multiple sentences is straightforward.
function AAA($\boldsymbol{s}, \boldsymbol{t}$)
  ▷ initialize alignments using the baseline HMM + fertility model
  ▷ the forward direction uses alignment vector $\boldsymbol{a}$
  $\boldsymbol{a} \leftarrow$ Baseline($\boldsymbol{s}, \boldsymbol{t}$)
  ▷ the backward direction uses alignment vector $\boldsymbol{b}$
  $\boldsymbol{b} \leftarrow$ Baseline($\boldsymbol{t}, \boldsymbol{s}$)
  while sampling **do**
    $M \leftarrow$ estimate bigram HMM model using $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{T}^s$
    ▷ set the target sentence tags $\boldsymbol{T}^t$ using the Viterbi algorithm
    $\boldsymbol{T}^t \leftarrow \arg\max_T P(T|M)$
    ▷ sample alignment variables $\boldsymbol{a}$, $\boldsymbol{b}$
    for all $j \leftarrow 1 \ldots J$ **do**
      $a_j \sim P(a_j = i|\boldsymbol{a}_{-j}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{T}^s, \boldsymbol{T}^t)$
    end for
    for all $i \leftarrow 1 \ldots I$ **do**
      $b_i \sim P(b_i = j|\boldsymbol{b}_{-i}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{T}^s, \boldsymbol{T}^t)$
    end for
  end while
  ▷ return expected values for PoS tags alignments, as in Equation (3)
  return $\mathbb{E}[\boldsymbol{a}], \mathbb{E}[\boldsymbol{b}], \mathbb{E}[\boldsymbol{T}^t]$
end function

2.5 PoS transfer

We focus on applying PoS transfer as a way of obtaining better word alignment accuracy, rather than improving PoS transfer as such. These are largely complementary goals, as our evaluation in Section 3 shows that small changes in PoS tagging accuracy do not seem to influence alignment accuracy. For this reason, and because of our focus on low-resource languages precludes using data-intensive approaches like that of Das and Petrov (2011), we choose the simple method of Yarowsky and Ngai (2001) as a starting point for the PoS transfer part of our model. The basis of this method is to use heuristics to estimate a robust first-order HMM tagger from (noisy) projected tags, and to re-tag the data using this tagger. Furthermore we extended the tagger using the affix-tree method of Schmid (1994) for rare words, in order to be able to handle morphologically complex languages better.

While it would have been preferable for reasons of theoretical elegance to use a simpler PoS transfer model, matching the alignment model, such attempts by Östling et al. (2015) resulted in very modest improvements for their sign language data set, and their model gave no improvement at all for our data sets.

2.6 Alternating alignment-annotation (AAA)

Algorithm 1 summarizes our method, which can be viewed as a modified Gibbs sampler of the latent alignment variables $\boldsymbol{a}$ and $\boldsymbol{b}$ (in the forward and backward alignment direction) as well as the target-side PoS tags $\boldsymbol{T}^t$. While the PoS transfer part is not stochastic,[1] it operates on samples of the alignment variables $\boldsymbol{a}$ and $\boldsymbol{b}$ and can be seamlessly integrated into the sampler.

3 Evaluation

The empirical evaluation aims at investigating whether the alternating alignment-annotation (AAA) algorithm improves word alignment and/or PoS transfer accuracy, compared to the corresponding PoS-

[1] We also tried sampling $\boldsymbol{T}^t$ using marginal distributions computed by the forward-backward algorithm, but found no effect on the accuracy of the algorithm.

Corpus	Sentences	$\|S\|$	$\|P\|$
English-French	1 130 588	4 038	17 438
Romanian-English	48 641	5 034	5 034
English-Inuktitut	333 185	293	1 972
English-Hindi	3 556	1 409	1 409
English-Swedish	692 662	3 340	4 577

Table 1: Total corpus sizes (in sentences) and number of (S)ure and (P)ossible alignment links in their respective evaluation sets.

unaware Bayesian IBM model with an HMM word order model and fertility.

3.1 Data

In order to assess the general usefulness of the method presented, a number of parallel corpora representing a diverse set of languages and genres are used: the English-French Hansards corpus in the version presented by Mihalcea and Pedersen (2003), the Romanian-English, English-Inuktitut and English-Hindi corpora from Martin et al. (2005), as well as parts of the Swedish-English Europarl corpus (Koehn, 2005) with the evaluation set of Holmqvist and Ahrenberg (2011). In addition, a set of translations of the New Testament will be used to investigate the quality of the transfered PoS tags. Some properties of these corpora are summarized in Table 1.

Silver-standard PoS annotations were provided for English, French and German by the Stanford Tagger (Toutanova et al., 2003) and for Swedish by Stagger (Östling, 2013). The native tagsets were mapped to the Universal PoS Tagset of Petrov et al. (2012).

3.2 Experimental setup

The main intended use case is a fairly short parallel text, with two very different languages of which only one has an accurate PoS tagger available. This excludes the possibility of extensive per-language tuning (unlike some of the previous results cited), and in this evaluation the different language pairs use identical parameters to the largest possible extent.[2] We fixed the hyperparameters in Equation (4) to $\alpha = 10^{-5}$, $\beta = \gamma = 1$.

The experiments use eight individually initialized samplers, each of which used a pipelined approach where initially a lexical-only model equivalent to that of Mermer and Saraçlar (2011) was used, then a word order term using the HMM model was added, then the fertility term, and finally (when applicable) the PoS translation probability. No burn-in period was used during sampling, since the initial value of the last pipeline step is already quite good.

Since the model is asymmetric, the alignments are run in both directions and symmetrized. A soft variant of the intersection heuristic is used, where the final set of links L is defined as $L = \{(i, j) \mid P(a_j = i)P(b_i = j) > t\}$. for a threshold value t, in these evaluations fixed to 0.25. This gives a fairly conservative set of links, favoring precision before recall. Note however that the model does not use NULL words, so this conservatism is not as severe as in models with NULL words.[3] Heuristics based on the union on the contrary tend to over-generate links under these conditions.

3.3 Results

The systems used as baselines in the evaluation are mainly from the Workshop on Parallel Text shared tasks (Mihalcea and Pedersen, 2003; Martin et al., 2005), where most of the data sets used were intro-

[2]The main exception is that some of the language pairs (Romanian-English and English-Hindi), following previous work, use a poor man's stemming trick where only the first four letters of each token is used. The only other exception is that the English-French evaluation did not use the fertility parameter, since it showed no further improvement beyond the plain HMM model.

[3]Later experiments with a related model (Östling and Tiedemann, 2016, compare their Table 2 with our Table 2) show that for some language pairs in particular, using NULL with standard symmetrization heuristics gives considerably worse AER scores.

duced: ISI2 (Fraser and Marcu, 2005), JHU (Schafer and Drábek, 2005), UMIACS2 (Lopez and Resnik, 2005) and XRCE (Dejean et al., 2003). The Swedish-English figures, LIU, are from Holmqvist and Ahrenberg (2011). Finally, we have run GIZA++ (Och and Ney, 2003) on the corpora as an additional baseline.[4] Results from previous work using more data than a bitext plus PoS tags for one of the languages are not included, although some such systems have obtained better results on some of the corpora used, using e.g. semi-supervised discriminative training (Liu et al., 2010).

Table 2 summarizes the main results of the evaluation. In all cases, the alternating alignment-annotation method surpasses the baseline model that does not use PoS tags. The model is generally competitive compared to previous work, in particular for the smaller corpora and where the languages are substantially different. The improvement compared to the non-Bayesian baselines is particularly good for the English-Inuktitut corpus, which could be due to the fact that the morpheme/word ratio of Inuktitut is very high, resulting in very many low-frequency words that tend to function as garbage collectors in non-Bayesian models. The situation is similar for the English-Hindi corpus, although in this case the cause for the many rare words is rather the short bitext than the languages themselves.

It is interesting to compare the corresponding **AAA** and **Supervised** figures in Table 2, where the only difference is that **AAA** uses a supervised PoS tag on one language (English) and annotation transfer to the other, whereas **Supervised** uses supervised PoS tags on both languages. The overall accuracy figures are nearly identical, even though the accuracy of the transfered tags is lower. This indicates that the word alignment algorithm is not very sensitive to PoS tagging accuracy, so that the relatively simple PoS transfer method used is sufficient for the purpose of increasing word alignment accuracy.

There are multiple translations of the New Testament into each of English, French, German and Swedish, which can be exploited for multi-source transfer. In our model, multi-source transfer can be done trivially by averaging the expectations returned by Algorithm 1. Table 3 shows that this overall has a large positive effect on PoS accuracy, with an average error reduction of one fourth compared to the median single-source result, and one tenth compared to the best (out of 22) single-source result. Using many translations in each language allows us to see how widely the accuracy varies, even when using the same source (or target) language. This is due to many factors, including the large time span (hundreds of years) between the different translations. In contrast, the multi-source results are, as could be expected, much more robust. This is an encouraging result, given that the New Testament is perhaps the most widely translated text of significant length, and offers a great possibility to transfer linguistic annotations to languages where little other data is available.

4 Conclusions and future work

We have presented a model for joint word alignment and PoS annotation transfer, and shown empirically that it leads to improved word alignment accuracy, in particular for low-resource languages. Using automatically transfered PoS tags led to improvements that were as big as the improvements seen when using PoS tags from supervised taggers on both sides of a bitext.

In addition, we took the opportunity to perform an evaluation investigating what kind of variation can be expected depending on which translation(s) are used as source texts in PoS annotation transfer, and found that this variation can be great, even among translations into the same language. Using multi-source transfer reduces this variation considerably and typically gives better accuracy than even the best single-source transfer among many.

In this study, only PoS annotations were considered, but there are other types of annotation such as parse trees, named entities and word senses which potentially could be transfered jointly with word alignment. This is left to future work, as are improvements to the baseline alignment model.

Acknowledgments

Thanks to the anonymous reviewers, Mats Wirén, Jörg Tiedemann and Joakim Nivre for advice.

[4] In order to provide a competitive but fair baseline, the same general approach was used with GIZA++ as with the new system presented, using default parameters and no language-specific tuning. The specific alignment pipeline used was $1^3h^53^34^{10}$, and the symmetrization that provides the best alignment on the test set is chosen. This gives GIZA++ some advantage, but ensures that any claimed improvements by our algorithm over GIZA++ are not simply due to symmetrization.

Model	$\|A\|$	$\|A \cap S\|$	$\|A \cap P\|$	P	R	F	AER
English-French ($\|S\| = 4038$, $\|P\| = 17438$. 1 130 588 sentences)							
Baseline	5359	3717	5134	95.8	92.1	93.9	5.8
AAA	5505	3751	5254	95.4	92.9	94.1	5.6
Supervised	5542	3778	5263	95.0	93.6	94.3	5.6
GIZA++	4831	3531	4715	97.6	87.4	92.2	7.0
XRCE				90.1	93.8	91.9	8.5
Romanian-English ($\|S\| = \|P\| = 6201$. 48 641 sentences)							
Baseline	3374	3070	3070	91.0	61.0	73.0	27.0
AAA	3447	3120	3120	90.5	62.0	73.6	26.4
GIZA++	3730	3161	3161	84.7	62.8	72.1	27.9
ISI2				87.9	63.1	73.5	26.6
RACAI				76.8	71.2	73.9	26.1
English-Inuktitut ($\|S\| = 293$, $\|P\| = 1972$. 333 185 sentences)							
Baseline	598	267	559	93.5	91.1	92.3	7.3
AAA	630	273	595	94.4	93.2	93.8	6.0
GIZA++	342	170	306	89.5	58.0	70.4	25.0
JHU				96.7	76.8	85.6	9.5
JHU				84.4	92.2	88.1	14.3
English-Hindi ($\|S\| = \|P\| = 1409$. 3 556 sentences)							
Baseline	712	606	606	85.1	43.0	57.1	42.9
AAA	817	677	677	82.9	48.0	60.8	39.2
GIZA++	984	615	615	62.5	43.6	51.4	48.6
UMIACS2				43.7	56.1	49.1	50.9
English-Swedish ($\|S\| = 3340$, $\|P\| = 4577$. 692 662 sentences)							
Baseline	3183	2742	2933	92.1	82.1	86.8	13.0
AAA	3125	2774	2961	94.8	83.1	88.5	11.3
Supervised	3262	2823	3034	93.0	84.5	88.6	11.3
GIZA++	3436	2890	3136	91.3	86.5	88.8	11.1
LIU				85.3	–	–	12.6

Table 2: Results from the empirical evaluation, including the Bayesian model without PoS tags (Baseline), the alternating alignment-annotation algorithm (AAA), the corresponding method but with supervised PoS taggers for both languages (Supervised), and comparable previous results on the same data. The number of alignment links $|A|$, of which $|A \cap S|$ are considered (S)ure, and $|A \cap P|$ (P)ossible, are reported. For convenience, precision (P), recall (R), F_1 score (F) and Alignment Error Rate (AER) are also given.

Target	Source texts															Multi
	deu (8)			eng (5)			fra (5)			swe (4)			All (22)			
deu_1				79.0	80.1	80.9	80.1	81.1	81.4	75.0	83.7	85.1	75.0	81.0	85.1	**86.8**
deu_2				79.3	80.8	81.4	79.5	80.5	80.7	78.3	83.5	85.2	78.3	80.7	85.2	**85.8**
deu_3				79.8	80.6	81.7	81.1	81.9	82.0	77.1	84.4	85.9	77.1	81.7	85.9	**88.2**
deu_4				80.6	81.1	82.4	81.0	81.8	81.8	75.3	83.2	85.4	75.3	81.6	85.4	**87.3**
deu_5				80.2	80.7	81.7	81.3	81.6	82.2	76.5	84.9	85.9	76.5	81.5	85.9	**86.8**
deu_6				79.4	81.3	81.9	80.0	80.7	81.3	80.7	85.0	**86.0**	79.4	81.0	**86.0**	85.4
deu_7				79.9	81.8	82.3	81.1	81.2	81.9	76.6	85.6	86.3	76.6	81.7	86.3	**86.4**
deu_8				80.0	81.4	82.3	81.0	81.4	82.0	76.3	84.8	85.7	76.3	81.6	85.7	**86.5**
eng_1	74.2	76.2	79.4				76.2	77.0	77.8	76.6	81.5	81.7	74.2	76.7	81.7	**83.8**
eng_2	79.2	81.8	84.2				80.9	81.4	82.0	79.8	85.8	**86.2**	79.2	81.8	**86.2**	85.5
eng_3	80.1	81.7	83.7				80.6	81.0	81.6	80.7	85.7	**86.3**	80.1	81.6	**86.3**	85.4
eng_4	79.5	81.4	84.3				80.3	80.7	81.3	78.8	85.8	**86.5**	78.8	81.3	**86.5**	85.1
eng_5	80.3	81.5	84.1				80.7	81.0	81.8	80.3	86.0	**86.7**	80.3	81.5	**86.7**	84.7
fra_1	80.2	82.9	83.5	80.1	81.3	81.5				78.0	83.8	84.5	78.0	82.5	84.5	**85.8**
fra_2	80.3	83.5	84.5	80.7	80.9	81.1				77.0	83.6	84.7	77.0	83.0	84.7	**86.0**
fra_3	80.5	83.1	83.5	79.9	81.2	81.9				77.6	84.3	85.1	77.6	82.7	85.1	**85.3**
fra_4	80.3	83.5	83.8	80.0	81.1	81.2				77.4	84.1	84.6	77.4	82.7	84.6	**85.3**
fra_5	80.5	83.2	83.8	80.1	81.2	81.4				77.2	84.0	84.8	77.2	82.9	84.8	**86.1**
swe_1	80.4	81.2	81.8	82.8	84.0	85.4	80.2	81.0	81.9				80.2	81.2	85.4	**90.3**
swe_2	76.3	77.5	79.4	80.9	81.7	82.4	76.9	77.9	79.4				76.3	78.3	82.4	**85.7**
swe_3	81.3	82.1	82.5	83.4	85.4	86.4	81.1	82.5	82.8				81.1	82.5	86.4	**90.6**
swe_4	81.8	82.3	82.7	82.7	84.8	85.4	81.8	82.7	83.3				81.8	82.7	85.4	**90.7**
Avg.	79.6	81.6	82.9	80.5	81.7	82.4	80.2	80.9	81.5	77.7	84.4	85.4	77.9	81.5	85.3	**86.5**

Table 3: PoS transfer accuracy (in percent) using single-source (first five columns) and multi-source (rightmost column) transfer in the New Testament corpus. Rows are target texts and columns are source languages. For each language (with number of translations), the worst/median/best results are given for the different translations. The **All** columns summarize the results over all the source texts from the preceding columns. Finally, **Multi** is the result of multi-source transfer using the sums of tag marginal distributions. The best result on each row is bold-faced.

References

Ondej Bojar and Magdalena Prokopov. 2006. Czech-English word alignment. In *LREC 2006*, pages 1236–1239, Genova, Italy. ELRA.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, June.

Colin Cherry and Dekang Lin. 2003. A probability model to improve word alignment. In *ACL 2003*, pages 88–95, Sapporo, Japan, July. Association for Computational Linguistics.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *ACL 2011*, HLT '11, pages 600–609, Stroudsburg, PA, USA. Association for Computational Linguistics.

Herve Dejean, Eric Gaussier, Cyril Goutte, and Kenji Yamada. 2003. Reducing parameter space for word alignment. In *HLT-NAACL-PARALLEL '03*, pages 23–26, Stroudsburg, PA, USA. Association for Computational Linguistics.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

John DeNero, Alexandre Bouchard-Côté, and Dan Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *EMNLP 2008*, pages 314–323, Honolulu, Hawaii, October. Association for Computational Linguistics.

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *ACL 2002*, pages 255–262, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alexander Fraser and Daniel Marcu. 2005. ISI's participation in the Romanian-English alignment task. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 91–94, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yarin Gal and Phil Blunsom. 2013. A systematic Bayesian treatment of the IBM alignment models. In *NAACL 2013*, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alan E. Gelfand and Adrian F. M. Smith. 1991. Gibbs sampling for marginal posterior expectations. Technical report, Department of Statistics, Stanford University.

Maria Holmqvist and Lars Ahrenberg. 2011. A gold standard for English-Swedish word alignment. In *Proceedings of the 18th Nordic Conference of Computational Linguistics (NODALIDA 2011)*, number 11 in NEALT Proceedings Series, pages 106–113.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(3):311–325, September.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *The Tenth Machine Translation Summit.*, Phuket, Thailand.

Yang Liu, Qun Liu, and Shouxun Lin. 2010. Discriminative word alignment by linear modeling. *Computational Linguistics*, 36(3):303–339, September.

Adam Lopez and Philip Resnik. 2005. Improved HMM alignment models for languages with scarce resources. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, ParaText '05, pages 83–86, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joel Martin, Rada Mihalcea, and Ted Pedersen. 2005. Word alignment for languages with scarce resources. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, ParaText '05, pages 65–74, Stroudsburg, PA, USA. Association for Computational Linguistics.

Coşkun Mermer and Murat Saraçlar. 2011. Bayesian word alignment for statistical machine translation. In *ACL 2011*, pages 182–187, Stroudsburg, PA, USA. Association for Computational Linguistics.

Coşkun Mermer, Murat Saraçlar, and Ruhi Sarikaya. 2013. Improving statistical machine translation using Bayesian word alignment and Gibbs sampling. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):1090–1101, May.

Rada Mihalcea and Ted Pedersen. 2003. An evaluation exercise for word alignment. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond - Volume 3*, HLT-NAACL-PARALLEL '03, pages 1–10, Stroudsburg, PA, USA. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, March.

Robert Östling and Jörg Tiedemann. 2016. Efficient word alignment with Markov Chain Monte Carlo. *Prague Bulletin of Mathematical Linguistics*, 106:125–146, October.

Robert Östling, Carl Börstell, and Lars Wallin. 2015. Enriching the Swedish Sign Language Corpus with part of speech tags using joint Bayesian word alignment and annotation transfer. In *Proceedings of the 20th Nordic Conference on Computational Linguistics (NODALIDA 2015)*, volume 23 of *NEALT Proceedings Series*, pages 263–268, Vilnius, Lithuania, May.

Robert Östling. 2013. Stagger: An open-source part of speech tagger for Swedish. *North European Journal of Language Technology*, 3:1–18.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *LREC 2012*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Darcey Riley and Daniel Gildea. 2012. Improving the IBM alignment models using variational Bayes. In *ACL 2012*, pages 306–310, Stroudsburg, PA, USA. Association for Computational Linguistics.

Charles Schafer and Elliott Franco Drábek. 2005. Models for Inuktitut-English word alignment. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, ParaText '05, pages 79–82, Stroudsburg, PA, USA. Association for Computational Linguistics.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK.

Kristina Toutanova, H. Tolga Ilhan, and Christopher Manning. 2002. Extensions to HMM-based statistical word alignment models. In *EMNLP 2002*, pages 87–94.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL 2003*, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *COLING 1996*, pages 836–841, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhiguo Wang and Chengqing Zong. 2013. Large-scale word alignment using soft dependency cohesion constraints. *Transactions of the Association for Computational Linguistics*, 1:291–300.

Dekai Wu and Xuanyin Xia. 1994. Learning an English-Chinese lexicon from a parallel corpus. In *Proceedings of the First Conference of the Association for Machine Translation in the Americas*, pages 206–213.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530, Toulouse, France, July. Association for Computational Linguistics.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *NAACL 2001*, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *HLT 2001*, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shaojun Zhao and Daniel Gildea. 2010. A fast fertility Hidden Markov Model for word alignment using MCMC. In *EMNLP 2010*, pages 596–605, Cambridge, MA, USA, October. Association for Computational Linguistics.

Splitting compounds with ngrams

Naomi Tachikawa Shapiro
Department of Linguistics
Stanford University
Stanford, CA 94305, USA
tsnaomi@stanford.edu

Abstract

Compound words with unmarked word boundaries are problematic for many tasks in NLP and computational linguistics, including information extraction, machine translation, and syllabification. This paper introduces a simple, proof-of-concept language modeling approach to automatic compound segmentation, demonstrated with Finnish. The approach utilizes an off-the-shelf morphological analyzer to split training words into their constituent morphemes. A language model is subsequently trained on ngrams composed of morphemes, morpheme boundaries, and word boundaries. Finally, linguistic constraints are used to weed out phonotactically ill-formed segmentations, thereby allowing the language model to select the best grammatical segmentation. This approach achieves an accuracy of ∼97%.

1 Introduction

Compound segmentation—the automatic splitting of compounds into their constituent words—is essential to many language processing tasks, including machine translation, information extraction, semantic parsing, spell checking, and syllabification. To that end, we propose a simple, supervised approach to compound segmentation that integrates existing morphological analysis software with language modeling and optional linguistic constraints.

Specifically, the proposed segmenter seeks to identify word boundaries in *closed compounds*, compounds in which word boundaries go undelimited by spaces, hyphens, or other markers (e.g., *bookworm*). These are distinct from *open compounds*, where word boundaries are clearly indicated (e.g., *rain gutter*) and thus less of an issue for applications that require their whereabouts. (Note that all compounds are considered *complex*, while words that are not compounds are considered *simplex*—e.g., *book*, *rain*).

In contrast to previous approaches, which have primarily sought to identify the dictionary forms of compound constituents, this approach aims to identify the exact location of word boundaries in compounds. This sort of identification is highly relevant to the domain of computational phonology.

For instance, the location of word boundaries is crucial for syllabification. Closed compounds pose a problem for automatic syllabification because word boundaries form a proper subset of syllable boundaries; a syllable break will always fall on a word boundary (or, so common phonological theory tells us). Without insight into the location of these boundaries, a rule-based syllabifier might fail to recognize compound-internal word boundaries. For example, if *bookworm* is mistaken for a simplex word, English phonotactics would syllabify it as **boo.kworm*. Instead, we expect the syllabification *book.worm*, where the syllable break falls on the unmarked word boundary between *book* and *worm*.

Hence, we have developed an approach to compound segmentation that specifically identifies word *boundaries* in compounds, versus lemmatized constituents. In Section 2, we describe previous approaches to compound segmentation in the areas of machine translation and information extraction. We then give a broad sketch of our approach in Section 3, introducing morpheme-based language modeling. Finally, in Section 4, we describe and evaluate an implementation of our approach on Finnish data. This

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 630–640, Osaka, Japan, December 11-17 2016.

implementation uses the morphological analyzer Morfessor 2.0 (Virpioja et al., 2013) and a trigram language model with Stupid Backoff smoothing (Brants et al., 2007). We also compare our approach to the popular frequency-based method by Koehn and Knight (2003).

2 Related work

Research in compound segmentation has varied in its motivations. While some has focused on improving information retrieval (e.g., Alfonseca et al., 2008) and spell checking (e.g., Huyssteen and Zaanen, 2004) systems, the majority of this work has been tailored to machine translation (e.g., Brown, 2002; Koehn and Knight, 2003; Macherey et al., 2011).

Motivated by a need for translatable constituents, many of these approaches involve multiple lexica and corpora. For instance, Brown (2002) utilizes parallel corpora and a translation lexicon to establish cognates between source and target languages. This allows decompounding into cognate constituents. Brown also proposes an extension to this approach that does not rely on cognate relationships.

Perhaps best known is Koehn and Knight's (2003) benchmark work on compound segmentation. Though the paper proposes several splitting methods, it is most cited for its frequency-driven approach, which scores candidate segmentations according to the geometric means of their constituents' corpus frequencies:

$$\hat{c} = \arg\max_{c \in C} \left(\prod_{p \in C} \text{count}(p)\right)^{\frac{1}{n}} \quad (1)$$

Above, $\hat{c}$ is the highest-scoring candidate in candidate set C, where each candidate c is composed of n number of constituent parts p. Candidate sets include all splits into known words, according to a training corpus. Splitting options are further confined to constituents of a minimum length three, and can assume hand-specified letters either inserted or dropped between constituent words, mirroring morphological operations at word joints. This monolingual approach is considered state-of-the-art and serves as a comparison in subsequent work (e.g., Alfonseca et al., 2008; Clouet and Daille, 2014), as well as in Section 4 of this paper.

Also in the domain of machine translation, Macherey et al. (2011) pitch an unsupervised, language-independent approach to compound segmentation. The approach uses phrase translation tables to learn the morphological operations that facilitate compounding, such as the insertion of linking morphemes. It then references a monolingual corpus to assess candidate segmentations. These elements are brought together in a complex dynamic programming algorithm.

Tackling compound segmentation from an information retrieval perspective, Alfonseca et al. (2008) leverage 900 million web documents to determine whether proposed compound constituents exist in anchor texts pointing to the same document. This approach builds on the semantic relationship between compound constituents: "If two words can create a compound word in a language, we can assume that there is some kind of semantic relationships [sic] between them and therefore we would expect to be able to find them near each other in other situations" (134). Alfonseca et al. combine this mutual information feature with lexical, frequency, and probability-based features in a Support Vector Machine classifier.

Efforts in compound segmentation vary widely in terms of the resources they require to get off the ground: monolingual and bilingual corpora, hand-written linguistic rules, web documents, and POS and frequency information. In addition, they have focused largely on the identification of lemmatized constituent words, and not on the identification of compound-medial word boundaries.

The segmenter presented in this paper is a supervised, monolingual approach that uses existing software and incorporates hand-written linguistic constraints. Instead of using multiple corpora, it draws from a single word list annotated for word boundaries. In addition, its linguistic rules are to ensure grammatical constituents, and do not mirror morphological operations to restore constituents to their pre-compound or lemmatized form (as in Koehn and Knight, 2003; Clouet and Daille, 2014; Owczarzak et al., 2014).

3 Compound splitting method

The method of compound segmentation introduced here begins with annotated training data, a set of word forms, both simplex and complex, hand-annotated with any unseen word boundaries. For instance, an annotator would likely mark up the Finnish closed compound *Suomenmaassa* 'in the Finnish country' as *Suomen=maassa* (*suomen* 'Finland-NOM', *maassa*-INE 'country'). Affix boundaries are not annotated unless they also constitute word boundaries.

An off-the-shelf morphological analyzer is trained on the annotated data and used to segment words into their constituent morphemes (e.g., one might split *Suomenmaassa* into the morphemes *suome*, *n*, *maa*, *ssa*). This morphological analyzer is then used to generate candidate compound segmentations and to train a language model.

3.1 Candidate generation

For every morpheme m asserted by the morphological analyzer, there are the four possible bigrams below, where # denotes a word boundary and X denotes a boundary between two morphemes, or $\neg$#.

$$\begin{array}{c} \# \; m \\ \mathrm{X} \; m \\ m \; \# \\ m \; \mathrm{X} \end{array}$$

Using this decomposition, candidate segmentations are generated for a word by proposing a word boundary along each of its alleged morpheme boundaries. Suppose that the morphological analyzer takes in the input w_i and outputs the morphemes $\{m_1,\ m_2,\ m_3\}$. The list of candidate segmentations for w_i would then be every possible combination of internal morpheme boundaries, as shown below. (A word assigned n morphemes will have 2^{n-1} candidate segmentations.)

$$\begin{array}{ll} \# \; m_1 \; \# \; m_2 \; \# \; m_3 \; \# & \text{(the most segmented)} \\ \# \; m_1 \; \# \; m_2 \; \mathrm{X} \; m_3 \; \# & \\ \# \; m_1 \; \mathrm{X} \; m_2 \; \# \; m_3 \; \# & \\ \# \; m_1 \; \mathrm{X} \; m_2 \; \mathrm{X} \; m_3 \; \# & \text{(the least segmented)} \end{array}$$

To give a real world example, if *Suomenmaassa* is split into the constituents {*suomen*, *maa*, *ssa*},[1] it would yield the following four candidate segmentations:

$$\begin{array}{ll} \# \; suomen \; \# \; maa \; \# \; ssa \; \# & (\textit{suomen=maa=ssa}) \\ \# \; suomen \; \# \; maa \; \mathrm{X} \; ssa \; \# & (\textit{suomen=maassa}) \\ \# \; suomen \; \mathrm{X} \; maa \; \# \; ssa \; \# & (\textit{suomenmaa=ssa}) \\ \# \; suomen \; \mathrm{X} \; maa \; \mathrm{X} \; ssa \; \# & (\textit{suomenmaassa}) \end{array}$$

One benefit of this approach to candidate generation is that it takes advantage of the linguistic insight that a word boundary will only occur where there is also a morpheme boundary. This reduces the size of the candidate set, compared to approaches that generate candidates by proposing a word boundary in between each letter of a word (e.g., Macherey, 2011), or by recursively proposing all two-part segmentations of some minimum length (e.g., Clouet and Daille, 2014).

Yet, a drawback of this approach is that candidate generation is at the mercy of the performance of the morphological analyzer. If the analyzer fails to identify a morpheme boundary between two morphemes, that boundary will not be considered as a potential word boundary site during compound segmentation.

3.2 Language modeling with morphemes

The intuition behind this segmenter is that the left and right edges of a morpheme each have a certain likelihood of appearing with a word boundary: For some morpheme m_i, the bigram '# m_i' might be more likely than the bigram 'X m_i' (or vice versa), and likewise for 'm_i #' and 'm_i X'.

Consider the English simplex word *lighting*, composed of the root *light* and affix *-ing* ('# $light$ X ing #'). English speakers know the bigram 'X ing' to be extremely common, or, at least, far more common than its counterpart '# ing'. This intuition can be captured with language modeling.

[1] The correct morphological analysis for *Suomenmaassa* is *Suome-n=maa-ssa* 'Finland-NOM=country-INE'.

Language models are used to assign probabilities to sequences of natural language tokens, typically words and punctuation. A language model sets out to determine the conditional probability of some token w_i given its history h (i.e., all of the tokens that precede w_i in a training corpus). We approximate the posterior $P(w|h)$ using the Markov assumption that w_i's history can be estimated through the few tokens that precede w_i:

$$P(w_i|h) \approx P(w_i|w_{i-n+1}^{i-1}) \tag{2}$$

Above, w_i^j is a string of tokens $w_i, ..., w_j$ and n is the order of ngram (i.e., $n = 2$ for bigrams, $n = 3$ for trigrams, etc.). We can calculate $P(w_i|w_{i-n+1}^{i-1})$ by dividing the frequency of w_{i-n+1}^{i} by the frequency of w_{i-n+1}^{i-1} in a training corpus. This gives us a *maximum likelihood estimate* (MLE) of $P(w_i|h)$:

$$P(w_i|w_{i-n+1}^{i-1}) = \frac{\text{count}(w_{i-n+1}^{i})}{\text{count}(w_{i-n+1}^{i-1})} \tag{3}$$

The probability of some sequence of tokens w_1^L is then estimated by taking the product of the MLE for each ngram that appears in the sequence:

$$P(w_1^L) = \prod_{i=1}^{L} \frac{\text{count}(w_{i-n+1}^{i})}{\text{count}(w_{i-n+1}^{i-1})} \tag{4}$$

Just as language modeling is used to estimate the likelihood of a sentence, it can be used to predict the likelihood of a proposed compound. Instead of training ngram probabilities on sentences (i.e., sequences of words), we can train them on words, wherein each word is a sequence of *morphemes* and *morpheme boundaries*.

Thus, to continue with the English example *lighting*, the bigram-MLE of the candidate segmentation **light=ing* would be calculated as follows:

$$P(\#light\#ing\#) = P(light|\#) \times P(\#|light) \times P(ing|\#) \times (\#|ing) \tag{5}$$

Since the bigram 'X ing' is far more frequent than '# ing', $P(\#light\text{X}ing\#)$ will receive a higher MLE than $P(\#light\#ing\#)$. Simply put, $P(lighting) > P(light{=}ing)$.

As the core component of this segmenter is a language model trained with morphemes, we consider this a *morpheme-driven* approach to compound segmentation. This is in contrast to word-driven approaches, such as the approaches outlined in the previous section. Word-driven approaches can struggle with accounting for compounds composed of constituent words that were unseen in their training lexicon (see Owczarzak et al., 2014, which addresses this issue). Our segmenter faces this same challenge, but attempts to overcome it by exploiting how, for any set of words, there is generally more information about the distribution of morphemes in the language than the distribution of its words.

3.3 Phonotactic constraints

Using morpheme-based language modeling alone, this approach is vulnerable to predications that are phonotactically ungrammatical, in that it can assert impossible word-internal sequences of sounds. For example, analyzing *bookworm* as **bookw=orm* produces the unpronounceable constituent **bookw*.

To compensate for these ungrammaticalities, linguistic constraints can be posited to prune unsound candidates from the candidate sets. Imagine that we imposed the constraint that all constituent words have at least one vowel. This would discard the candidate compound **boobwo=rm* because its constituent **rm* violates the constraint.

With a whittled-down candidate set, the segmenter selects the remaining candidate with the highest language model score.

4 Experiment

We trained and evaluated this approach on an annotated subset of the Aamulehti-1999 Finnish daily newspaper corpus (Aamulehti, 1999).

Data set	*Total*	*Simplex*	*Complex*		
			Total	*Open*	*Closed*
training set	18,079	14,489	3,590	312	3,278
test set	2,001	1,606	395	41	354

Table 1: Training-test set split.

4.1 The data

Aamulehti-1999 contains 16,608,843 words across 61,529 news articles. We extracted a subset composed of 20,080 unique word types, including any word that appears one hundred or more times in the corpus.

A single annotator hand-annotated unmarked word boundaries in this subset. The annotator identified 3,632 closed compounds among the 20,080 word types, giving the data the distribution in Table 1. This gold standard underwent a 90-10 train-test split: We used 90% of the annotated data to train both a morphological analyzer and morpheme-based language model. We then evaluated the segmenter on the held-out 10%.

4.2 The segmenter

Morphological analyzer

We used a baseline Morfessor 2.0 model (Virpioja et al., 2013) to segment words into morphemes. The Morfessor model was trained on raw text consisting of space-delimited data, in which all marked and unmarked word boundaries were indicated with spaces. (E.g., '`...runoja raaka-aineen suomen=maassa...`' would have been represented as '`...runoja raaka aineen suomen maassa...`'.) As needed for candidate generation and the language model, the model's `viterbi_segment` method was used to segment words along their predicted morpheme boundaries.

Language model

To avoid zero-probabilities (i.e., where $P(sequence) = 0$), we used a simple trigram language model with Stupid Backoff smoothing (Brants et al., 2007). The model ultimately backed off to a Laplace-smoothed unigram count, assigning a score S to a morpheme/boundary m_i accordingly:[2]

$$S(m_i|m_{i-2}^i) = \begin{cases} \dfrac{\text{count}(m_{i-2}^i)}{\text{count}(m_{i-2}^{i-1})}, & \text{if count}(m_{i-2}^i) > 0 \\ \alpha \times \dfrac{\text{count}(m_{i-1}^i)}{\text{count}(m_{i-1})}, & \text{if count}(m_{i-1}^i) > 0 \\ \alpha^2 \times \dfrac{\text{count}(m_i) + \delta}{N + \delta|V|}, & \text{otherwise} \end{cases} \tag{6}$$

Above, N is the total number of unigrams seen during training, V is the vocabulary size (i.e., the number of unique morphemes and boundaries), and the Laplace discount δ is 1.0. As in Brants et al. (2007), the Stupid Backoff discounting parameter α is 0.4 — it was not tuned to the corpus.

The language model score (S_{LM}) for a candidate c_i is the product of the score for each morpheme and boundary that appeared in c_i; this sequence is defined as m_1^L:

$$S_{LM}(c_i) = S_{LM}(m_1^L) = \prod_{i=1}^{L} S(m_i|m_{i-2}^i) \tag{7}$$

[2]Stupid Backoff dispenses with real probabilities, assigning a score $S(x)$ instead of a probability $P(x)$. Some, however, attribute this S to stand for $Stupid$.

To predict the best segmentation $\hat{c}$ for a word given its candidate set C, the segmenter selected the candidate with the highest language model score:

$$\hat{c} = \arg\max_{c \in C} S_{LM}(c) \tag{8}$$

Phonotactic constraints

We formulated four phonotactic constraints that are, in effect, unviolable in Finnish:

- *Minimal Word* (MINWRD): A candidate incurs a MINWRD violation for each constituent word that contains fewer than two vowels. (Cf. Suomi et al., 2008). E.g., *a=*asian* ∼ *aasian*, *jun=*tunen* ∼ *juntunen*, *kä=*väisi* ∼ *käväisi*, *n=*uotio* ∼ *nuotio*, *mat=*ala=vaahtoisen* ∼ *matala=vaahtoisen*, and *ta=*lutus=nuorasta* ∼ *talutus=nuorasta*.

- *Sonority Sequencing* (SONSEQ): A candidate incurs a SONSEQ violation for each constituent word that begins in a consonant cluster not in */pl*, *pr*, *tr*, *kl*, *kr*, *sp*, *st*, *sk*, *spr*, *str*/ or that ends in any consonant cluster. (Cf. Sulkala and Karjalainen, 1992; Suomi et al., 2008). E.g., *luonn=*ehti* ∼ *luonnehti*, *ehtoisa=mpaa* ∼ *ehtoisampaa*, and *jukola=ntupien* ∼ *jukolan=tupien*.

- *Vowel Harmony* (V-HARMONY): A candidate incurs a V-HARMONY violation for each constituent word that contains both front vowels */ä*, *ö*, *y*/ and back vowels */a*, *o*, *u*/. (Cf. Sulkala and Karjalainen, 1992; Suomi et al., 2008; Ringen and Heinamaki, 1999; Karlsson, 2015). E.g., *kesäillan* ∼ *kesä=illan*, *taaksepäin* ∼ *taakse=päin*, and *muutostöitä* ∼ *muutos=töitä*.

- *Word-Final Consonants* (WRDFINAL): A candidate incurs a WRDFINAL violation for each constituent word that ends in a consonant that is not */t*, *s*, *n*, *l*, *r*/. (Cf. Sulkala and Karjalainen, 1992; Suomi et al., 2008). E.g., *pitem=pään* ∼ *pitempään*, *sulok=kuutta* ∼ *sulokkuutta*, and *hyp=pää* ∼ *hyp=pää*.

Since these constraints are largely unviolable in Finnish, we designed the segmenter to discard candidates that violate the constraints. Given a set of candidates for an input, for each phonotactic constraint, the segmenter discarded any candidates that violated the constraint, unless it was the case that *every* candidate violated the constraint. If every candidate violated it, the number of shared violations was subtracted from the number of violations incurred by each candidate. Any candidates that still violated the constraint were then discarded. This is the notion of wiping out shared violations found in Optimality Theory (Prince and Smolensky, 1993/2004).

After using the constraints to pare down the candidate set, the segmenter selected the remaining candidate with the highest language model score. If no candidates remained, it defaulted to the simplex candidate;[3] this meant that all of the candidates violated *some* phonotactic constraint, but not the same constraint equally.

4.3 Evaluation

We evaluated our approach by comparing the segmentations produced by the segmenter to the held-out gold standard. This was done by tallying true and false positives and negatives according to the definitions below. These metrics mirror those used by Koehn and Knight (2003), Alfonseca et al. (2008), Aussems et al. (2013), and Clouet and Daille (2014).

- **True positives** (TP): Closed compounds that are correctly segmented.

- **False positives** (FP): Simplex words that are mistakenly segmented.

[3]Motivations to default to the simplex candidate were twofold. First, the dataset's high simplex rate renders a word more likely to be simplex than complex. (However, it is important to recall that the dataset is skewed towards more frequent words, and that compound frequency likely has an inverse correlation with word frequency.) In addition, as we learned from a development set, it is generally only with loanwords that *every* candidate violates a constraint. Since the language model is trained predominantly on core Finnish, it stands to reason that it will make poorer predictions with loanwords, resulting in a greater number of false positives with loanwords. Hence, in these cases, we had the segmenter default to the simplex candidate.

- **True negatives** (TN): Simplex words or open compounds that are appropriately left unsegmented.
- **False negatives** (FN): Closed compounds that the segmenter fails to segment altogether.
- **Bad segmentations** (Bad): Closed compounds that the segmenter segments, but improperly so.

Using these classifications, precision, recall, and accuracy were calculated, with both precision and recall penalized for bad segmentations:

- **Precision** = $\frac{TP}{TP + FP + Bad}$
- **Recall** = $\frac{TP}{TP + FN + Bad}$
- **Accuracy** = $\frac{TP + TN}{TP + TN + FP + FN + Bad}$

As a probabilistic morphological analyzer, Morfessor predicts slightly different segmentations for the same set of data each time a model is trained. These segmentations consequently impact the ngrams stored in the language model, as well as the constraint interactions further down the pipeline. Due to this variation, we trained a Morfessor model and subsequent language model 50 times on the training set. This was done to ascertain the average performance of each segmenter given the training set and variation from Morfessor. Table 2 portrays the mean precision, recall, and accuracy for using language modeling alone and language modeling with constraints to segment compounds.

Segmenter	*P*	*R*	*Acc.*
Baseline (no segmentation)	-	0.0	0.8230
Language model	0.7493	0.8970	0.9373
Language model + constraints	0.8855	0.9201	0.9690

Table 2: Mean performance from training the Morfessor and language models for 50 iterations.

On average, language modeling alone achieved an accuracy of ∼94%. In contrast, a language model coupled with linguistic constraints achieved a much higher accuracy, hovering around 97%. Both methods substantially surpassed a baseline of leaving all inputs unsegmented.

Error analysis

To examine specific errors from a representative iteration, we found the iteration that produced accuracies most similar to the mean accuracies depicted in Table 2. The results from this average iteration are shown in Table 3.

Segmenter	*TP*	*TN*	*FP*	*FN*	*Bad*	*P*	*R*	*Acc.*
Baseline (no segmentation)	0	1,647	0	354	0	-	0.0	0.8230
Language model	322	1,553	92	16	18	0.7454	0.9045	0.9370
Language model + constraints	328	1,611	36	18	8	0.8817	0.9266	0.9690

Table 3: Performance from the average iteration.

The 62 errors made by the constraint-based (i.e., "language model + constraints") segmenter fell under one of three types: constraint errors, Morfessor errors, and language modeling errors. The distribution of these errors is summarized in Table 4.

Constraint errors. A constraint error occurred when one of the phonotactic constraints was the cause of the correct segmentation losing. The constraint-based segmenter encountered only three such errors (4.8%), the false negatives **mäkicup* 'ski jump cup' and **bruttokansantuote* 'gross domestic product', and the false positive **yksin=omaan* 'only'.

Error type	*FP*	*FN*	*Bad*	*Total*
Constraint errors	1	2	0	3 (4.8%)
Morfessor errors	0	4	3	7 (11.3%)
Language modeling errors	35	12	5	52 (83.9%)
All errors	36	18	8	62

Table 4: Distribution of errors from the "language model + constraints" segmenter.

The correct segmentation *mäki=cup* was ruled out due to the loanword *cup* violating MINWRD. And, *brutto=kansan=tuote* was eliminated because the */br/* onset of the Swedish stem *brutto* 'gross' violated SONSEQ. In these two cases, all of the candidates violated some constraint and the segmenter consequently defaulted to the simplex candidate. Had the correct segmentation not violated a phonotactic constraint, it would have been uniquely identified as the winner.

In the third case, the simplex candidate *yksinomaan* violated V-HARMONY, as it contains the front vowel */y/* and back vowels */o*, *a/*.[4] This led **yksin=omaan* to be selected, as it violated none of the phonotactic constraints.

The presence of constraint errors cautions us that a segmentation approach that uses phonotactic constraints is sensitive to loanwords. However, it was only with loanwords where each candidate violated a constraint. This, to some extent, offers loanword detection that falls naturally out of the architecture of the segmenter. This might render it possible to make special considerations for core-periphery structure in the future.

Morfessor errors. The most serious error this segmenter faced was one that stemmed from the morphological analyzer. If the morphological analyzer failed to segment a word into its correct constituent morphemes and, in doing so, did not insert a morpheme boundary that also constituted a true word boundary, that word's candidate set did not include the correct segmentation. 11.3% (7/62) of the errors made were Morfessor errors. For example, the segmenter was unable to predict the compound *oheis=kräsän* 'extraneous junk', since Morfessor split the input *oheiskräsän* into the constituents {*o*, *hei*, *sk*, *rä*, *ä*, *sä*, *n*}. Here, the constituent *sk* subsumes the compound break.

The non-trivial frequency of these errors emphasizes the importance of having a well-trained morphological analyzer.[5] One way to possibly minimize these errors would be to train the analyzer on words annotated for morpheme boundaries instead of word boundaries. (However, morpheme annotation would require more specialized knowledge than compound annotation.) As always, training the morphological analyzer on more data would also likely lead to some improvement.

Language modeling errors. By far the most rampant errors were language modeling errors, totaling 83.9% of the errors (52/62). Language modeling errors arose when, given a refined set of grammatical candidates, the language model favored the incorrect segmentation. Their prevalence is telling about the compound segmentation problem: It highlights the difference between predicting possible *nonword* compounds and predicting *actual* compounds. It also indicates that the language model is the paramount site for improvement with this approach.

Comparison to frequency-based approaches

We also implemented two versions of Koehn and Knight's (2003) frequency-based segmenter. Both implementations scored candidates solely according to the geometric means of their constituents' frequencies, as in (1). Frequency information and part-of-speech (POS) tags were provided by Aamulehti-1999.

[4] Although *yksinomaan* is left simplex in the gold standard, *yksin=omaan* is quite arguably the correct segmentation. *Yksinomaan* is composed of the stems *yksin* 'alone' and *oma-an* 'own-ILL'. While the word is semantically noncompositional, evidence from syllabification suggests that it is a compound. Were *yksinomaan* truly simplex, its syllabification would be **yk.si.no.maan*. However, it syllabifies as if it were a compound, with a syllable boundary falling in between the two stems: *yk.sin.o.maan*.

[5] As an aside, we used Morfessor's own built-in evaluation suite to evaluate this iteration's Morfessor model. Like Virpioja et al. (2013), we used the morpheme-annotated Finnish gold standard from Morpho Challenge 2010 (Kurimo et al., 2010). Our Morfessor model received 0.548 and 0.614 on precision and recall, respectively. We find it promising that language modeling alone achieved a 0.937 accuracy, despite our Morfessor model's performance.

Segmenter	*TP*	*TN*	*FP*	*FN*	*Bad*	*P*	*R*	*Acc.*
Baseline (no segmentation)	0	1,647	0	354	0	-	0.0	0.8230
Frequency-based	252	1.389	251	34	75	0.4360	0.6981	0.8201
Frequency-based + POS	287	1,554	92	46	22	0.7157	0.8085	0.9200
Language model	322	1,553	92	16	18	0.7454	0.9045	0.9370
Language model + constraints	328	1,611	36	18	8	0.8817	0.9266	0.9690

Table 5: Performance of the frequency-based and language model segmenters.

The two implementations differed with respect to their candidate sets. The first implementation allowed splits into *any* words found in the corpus; the second implementation employed POS-filtering, only permitting splits into *content* words (but not proper nouns). Candidate sets for both implementations were restricted to constituents of at least three characters in length.

Table 5 displays the results from evaluating the frequency-based segmenters on the test set; the language modeling results from the average iteration are repeated for easy comparison. As the table shows, the pure frequency-based approached received 43.60% on precision and 69.81% on recall, culminating in an accuracy comparable to the baseline's accuracy (∼82%). Both the baseline and frequency-based segmenters were surpassed by the POS-filtered approach, which achieved an accuracy of 92.00%. The language modeling approach achieved a slightly higher accuracy of 93.70%. And, overall, the constraint-based approach achieved the highest precision (88.17%), recall (92.66%), and accuracy (96.95%).

Most notably, the language modeling segmenters earned far fewer false negatives (i.e., higher recall) than the frequency-based segmenters. This returns us to the issue mentioned in Section 3.2. As word-driven segmenters, the frequency-based approaches struggled with capturing compounds whose constituents did not appear on their own in the corpus. Out of the 46 false negatives produced by the POS-filtered segmenter, 32 occurred because, in each case, one or more of the correct segmentation's constituents did not appear in Aamulehti-1999 (or did not appear as a content word), precluding it from the candidate set.

On the other hand, as morpheme-driven approaches, the language modeling segmenters largely avoided these errors. (They produced only 3 of the aforementioned 32 errors.) For instance, the frequency-based approaches failed to segment the compound *yli=määräistä* 'extra' (i.e., *yli=määrä-is-tä* 'over=amount-Adj.-PAR'), since the corpus did not contain *määräistä* as a standalone word. In contrast, the language modeling approaches were able to insert a word boundary in between *yli* and *määräistä*, since the bigram 'yli #' surfaced 58 times in the training set, and 'yli X' only 17 times.

5 Conclusion

We have proposed a language modeling and constraint-based approach to compound segmentation. This approach was demonstrated with Finnish, a highly agglutinative language. We showed that, by using a morphological analyzer to split words annotated for compound-medial word boundaries into constituent morphemes, we can train a language model that scores different configurations of morphemes, morpheme boundaries, and word boundaries.

Our implementation of this approach used the off-the-shelf morphological analyzer Morfessor 2.0 (Virpioja et al., 2013) and a simple trigram language model with Stupid Backoff smoothing (Brants et al., 2007). This achieved a segmentation accuracy of ∼94%. Then, by layering linguistic constraints on top of the language model, we rooted out phonotactically ill-formed segmentations, allowing the language model to select only grammatical segmentations. This boosted the segmentation accuracy to ∼97%.

In sum, using imperfect morphological analysis with language modeling can achieve impressive results in compound segmentation. This suggests that using a better trained morphological analyzer in conjunction with a more sophisticated language model will lead to further traction in the compound segmentation problem. Furthermore, the better the language model performs, the more the need for phonotactic constraints is obviated.

Lastly, while this approach was specifically designed to identify word boundary sites in the realm of computational phonology, perhaps it can be adapted for machine translation and information retrieval. For instance, a LEMMA constraint could be added: A candidate segmentation could incur a LEMMA violation for each constituent whose lemmatized form cannot be found in a dictionary or translation lexicon.

Acknowledgements

I am grateful to Arto Anttila for his mentorship throughout this project and to Christopher Manning and the anonymous reviewers for their constructive feedback. In addition, many thanks to Kati Kiiskinen for providing swift and thoughtful annotations of Finnish compounds.

References

Aamulehti. 1999. An electronic Finnish newspaper containing 16,608,843 words. Gatherers: Research Institute for Languages of Finland, the Department of General Linguistics at the University of Helsinki, and the Finnish IT Center for Sciences.

Enrique Alfonseca, Slaven Bilac, and Stefan Paries. 2008. German decompounding in a difficult corpus. In *Lecture Notes in Computer Science: Proceedings of the 9th International Conference on Intelligent Text Processing and Computational Linguistics*, volume 4149, pages 128–139.

Suzanne Aussems, Bas Goris, Vincent Lichtenberg, Nanne van Noord, Rick Smetsers, and Menno van Zaanen. 2013. Unsupervised identification of compounds. In *Proceedings of the 22nd Annual Belgian-Dutch Conference on Machine Learning*, pages 18–25, Nijmegen, Netherlands.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 858—-867, Prague, Czech Republic.

Ralf D. Brown. 2002. Corpus-driven splitting of compound words. In *Proceedings of the 9th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 12–21, Keihanna, Japan.

Elizaveta Clouet and Béatrice Daille. 2014. Splitting of compound terms in non-prototypical compounding languages. In *Proceedings of the First Workshop on Computational Approaches to Compound Analysis*, pages 11–19, Dublin, Ireland.

Gerhard B. Van Huyssteen and Menno M. Van Zaanen. 2004. Learning compound boundaries for Afrikaans spell checking. In *Proceedings of First Workshop on International Proofing Tools and Language Technologies*, pages 101–108, Patras, Greece.

Fred Karlsson. 2015. *Finnish: An essential grammar*. Routledge, London, United Kingdom, 3 edition.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of the 10th conference on European chapter of the Association for Computational Linguistics*, pages 187–193, Budapest, Hungary.

Mikko Kurimo, Sami Virpioja, and Ville T. Turunen. 2010. Overview and results of Morpho Challenge 2010. In *Proceedings of the Morpho Challenge 2010 Workshop*, pages 7–24, Espoo, Finland. Aalto University School of Science and Technology, Department of Information and Computer Science. Technical Report TKK-ICS-R37.

Klaus Macherey, Andrew M. Dai, David Talbot, Ashok C. Popat, and Franz Och. 2011. Language-independent compound splitting with morphological operations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 1395–1404, Portland, Oregon.

Karolina Owczarzak, Ferdinand de Haan, George Krupka, and Don Hindle. 2014. Wordsyoudontknow: Evaluation of lexicon-based decompounding with unknown handling. In *Proceedings of the First Workshop on Computational Approaches to Compound Analysis*, pages 63–71, Dublin, Ireland.

Alan Prince and Paul Smolensky. 1993/2004. Optimality Theory: Constraint interaction in generative grammar. Technical report, Rutgers University and University of Colorado at Boulder, 1993. Revised version published by Blackwell, 2004.

Catherine O. Ringen and Orvokki Heinämäki. 1999. Variation in Finnish vowel harmony: An OT account. *Natural Language & Linguistic Theory*, 17(2):303–337.

Helena Sulkala and Merja Karjalainen. 1992. *Finnish*. Routledge, London, United Kingdom.

Kari Suomi, Juhani Toivanen, and Riikka Ylitalo. 2008. *Finnish sound structure: Phonetics, phonology, phonotactics and prosody*. Oulu University Press, Oulu, Finland.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. 2013. Morfessor 2.0: Python implementation and extensions for Morfessor baseline. Technical report, Department of Signal Processing and Acoustics, Aalto University.

GAKE: Graph Aware Knowledge Embedding

Jun Feng[⋆]**, Minlie Huang**[⋆]**, Yang Yang**[†]**, and Xiaoyan Zhu**[⋆]
[⋆] State Key Lab. on Intelligent Technology and Systems,
Tsinghua National Lab. for Information Science and Technology,
Dept. of Computer Science and Technology, Tsinghua University
[†] College of Computer Science and Technology, Zhejiang University
feng-j13@mails.tsinghua.edu.cn, aihuang@tsinghua.edu.cn
yangya@zju.edu.cn, zxy-dcs@tsinghua.edu.cn

Abstract

Knowledge embedding, which projects triples in a given knowledge base to d-dimensional vectors, has attracted considerable research efforts recently. Most existing approaches treat the given knowledge base as a set of triplets, each of whose representation is then learned separately. However, as a fact, triples are connected and depend on each other. In this paper, we propose a graph aware knowledge embedding method (GAKE), which formulates knowledge base as a directed graph, and learns representations for any vertices or edges by leveraging the graph's structural information. We introduce three types of graph context for embedding: neighbor context, path context, and edge context, each reflects properties of knowledge from different perspectives. We also design an attention mechanism to learn representative power of different vertices or edges. To validate our method, we conduct several experiments on two tasks. Experimental results suggest that our method outperforms several state-of-art knowledge embedding models.

1 Introduction

Knowledge bases, such as DBpedia, YAGO, and Freebase, are important resources to store complex structured facts about the real world in the form of triplets as (*head entity*, *relation*, *tail entity*). These knowledge bases have benefited many applications, such as web search and question answer. In the meanwhile, knowledge base embedding, which aims to learn a D-dimensional vector for each *subject* (i.e., an entity or a relation) in a given knowledge base, has attracted considerable research efforts recently (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015b; Ji et al., 2015). For instance, TransE method (Bordes et al., 2013) regards the relation in a triplet as a translation between the embedding of the two entities. In other words, TransE learns a preference of $\mathbf{h} + \mathbf{r} = \mathbf{t}$ for each triple, where $\mathbf{h}$, $\mathbf{r}$, and $\mathbf{t}$ are the representation vector of head entity, relation, and tail entity respectively. Similar ideas are also proposed in TransH (Wang et al., 2014), TransR (Lin et al., 2015b), TransSparse (Ji et al., 2016), etc.

Despite the success of above methods in learning knowledge representations, most of them mainly consider knowledge base as a set of triples and models each triple separately and independently. However, in reality, triples are connected to each other and the whole knowledge base could be regarded as a directed *graph* consisting of vertices (i.e., entities) and directed edges (i.e., relations). In this way, we see that most of existing methods only consider "one hop" information about directed linked entities while miss more global information, such as multiple-steps paths, K-degree neighbors of a given vertex, etc. We call these different structural information as *graph context* inspired by *textural context* utilized in learning a given word's representation (Tomas Mikolov, 2013).

In this paper, we present a novel method to learn the representations of knowledge by utilizing graph context. Figure 1 gives an example to further explain the motivation of our work. In Figure 1(a), we are given a knowledge base organized as a directed graph which shores the facts about the singer Taylor Swift and president Barack Obama. We then demonstrate three kinds of graph context utilized to encode "Taylor_Swift" and "Barack_Obama".

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 641–651, Osaka, Japan, December 11-17 2016.

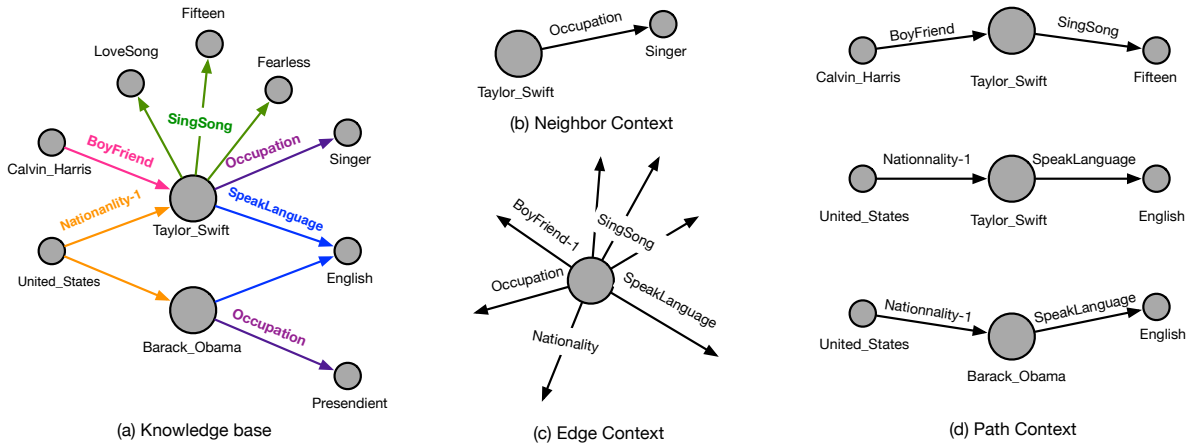


Figure 1: An illustration of three types of graph context, given by a knowledge base.

Neighbor context, as shown in Figure 1(b), consists of the target entity (e.g., "Taylor_Swift") and its directed linked entities (e.g., "Singer") along with their relations (e.g., "Occupation"). It is the most common context and is used in all knowledge base embedding methods.

Edge context, which is shown in Figure 1(c), indicates all kinds of relations relevant to the target entity, such as "SingSong", "BoyFriend^{-1}"(a reverse relation of 'BoyFriend"), "Nationality", and "Occupation" relations of "Taylor_Swift". The relations together would be helpful identify the target entity. For example, "SingSong" and several "BoyFriend^{-1}" relations represent the fact that Taylor, as a singer, has quite a few boy friends in reality. Please notice that different relations has different representation power. For instance, "SingSong" is a very unique relation and is very helpful to identify a singer. Meanwhile, "Nationality" occurs with every human being, so that gains less value. We will introduce how to handle this issue by utilizing an *attention mechanism* in our proposed method later.

Path context is defined as paths in the given graph containing the target entity. Figure 1(d) gives an example of several 3-step paths containing "Taylor_Swift" or "Barack_Obama". The two paths $United\_States \xrightarrow{Nationality^{-1}} Taylor\_Swift/Barack\_Obama \xrightarrow{SpeakLanguage} English$ represent that the two target entities are similar in terms of nationality and language and suggests their embedding vectors should be somehow similar from this perspective.

There are several challenges when learning knowledge representation by graph context. First, there are quite a few different types of graph context while each has unique structural properties. How to propose a general framework that is able to handle all kinds of graph context is one of the challenges in this work. Second, as we have mentioned previously, in the same type of graph context, different entities/relations have different representation power. For example, in edge context, the "SingSong" relation is more powerful than the "occupation" relation as the former one is less frequent and more unique for singers. How to learn the representation power of each entity/relation is the second challenge we meet. Third, how to estimate model parameters by utilizing real data is also a challenge.

Our contributions in this work include: (1) We treat a given knowledge base as a directed graph instead of a set of independent triples, and extract different types of graph context to study the representation of knowledge. (2) We propose a novel and general representation learning approach, GAKE (Graph Aware Knowledge Embedding), which can be easily extended to consider any type of graph context. (3) We propose an attention mechanism in our approach to learn representation power of different entities and relations.

The rest of this paper are organized as follows. In Section 2, we introduce some related works. In Section 3, we detail the proposed method of graph aware knowledge embedding. Section 4 describes the data and presents experimental results to validate our method. Section 5 concludes the paper.

Table 1: A summary of different knowledge embedding methods.

Method	Triple	Path	Edge
NTN(Socher et al., 2013)	✓	×	×
TransE(Bordes et al., 2013)	✓	×	×
TransH(Wang et al., 2014)	✓	×	×
TransR(Lin et al., 2015b)	✓	×	×
TransD(Ji et al., 2015)	✓	×	×
TranSparse(Ji et al., 2016)	✓	×	×
PTransE(Lin et al., 2015a)	✓	✓	×
Traversing(Gu et al., 2015)	✓	✓	×
GAKE(ours)	✓	✓	✓

2 Related Work

In this section, we review some existing work relevant to our paper. Generally, our work is closely related to the following two topics: (1) knowledge base embedding (2) Graph embedding.

2.1 Knowledge Base Embedding

A variety of approaches have been explored for knowledge base embedding, such as general linear based models, such as SE (Bordes et al., 2011), bilinear based models, like LFM (Jenatton et al., 2012; Sutskever et al., 2009), neural network based models, like SLM (Socher et al., 2013), NTN (Socher et al., 2013), and translation based models (Bordes et al., 2013; Wang et al., 2014; Xiao et al., 2015). The mainstream models for knowledge base embedding are translation based models including TransE (Bordes et al., 2013) and its variant models.

Translation-based models all share quite similar principle $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$, where $\mathbf{h}, \mathbf{r}$ and $\mathbf{t}$ are the embedding vectors of a triple (h, r, t), though these models differ in score functions. The score function of the translation based models is designed as: $f_r(h, t) = \mathbf{h}_r + \mathbf{r} - \mathbf{t}_r$, where $\mathbf{h}_r$ and $\mathbf{t}_r$ are the embedding vectors of head and tail entities which projected into the relation-specific space.

In TransE (Bordes et al., 2013), the entity and relation embedding vectors are in the same space, say $\mathbf{h}_r = \mathbf{h}, \mathbf{t}_r = \mathbf{t}$. In TransH (Wang et al., 2014), entity embedding vectors are projected into a relation-specific hyperplane $\mathbf{w}_r$, say $\mathbf{h}_r = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r, \mathbf{t}_r = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t}\mathbf{w}_r$. In TransR (Lin et al., 2015b), $\mathbf{h}_r = \mathbf{h}\mathbf{M}_r, \mathbf{t}_r = \mathbf{t}\mathbf{M}_r$, where entities are projected from the entity space to the relation space by $\mathbf{M}_r$. In TransD (Ji et al., 2015), $\mathbf{h}_r = \mathbf{M}_{rh}\mathbf{h}$, $\mathbf{t}_r = \mathbf{M}_{rt}t$, where the mapping matrices $\mathbf{M}_{rh}$ and $\mathbf{M}_{rt}$ are both related to the entity and relation. In TransSparse (Ji et al., 2016), $\mathbf{h}_r = \mathbf{M}_r(\theta_r)\mathbf{h}$, $\mathbf{t}_r = \mathbf{M}_r(\theta_r)\mathbf{t}$, where $\mathbf{M}_r$ is an adaptive sparse matrix, whose sparse degrees are determined by the number of entities linked by the relations.

In addition, there are still some works(Xiao et al., 2016b; Xiao et al., 2016a) follow the principle $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$, although they do not share the same form of score function. Particularly, (Xiao et al., 2016b) proposes to use a generative model to deal with multiple semantic meanings of a relation. To accommondate more flexible knowledge embedding, (Xiao et al., 2016a) proposes a manifold principle instead of a point-wise estimation of entity and relation embeddings. There are some other works incorporate additional information, such as text(Toutanova and Chen, 2015; Toutanova et al., 2015) and entity types(Guo et al., 2015).

Above knowledge base embedding models all treat the knowledge base as a set of triples. However, in fact, knowledge base is a graph with its graph structure which can be used to better embed the entities and relations in knowledge base. Although (Gu et al., 2015) and PTransE(Lin et al., 2015a) introduce the relation path instead of only considering the direct relations between entities, they just treat the relation path as a new relation and the path length is limited to the model complexity.

However, (Feng et al., 2016) claims the principle $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ is too strict to model the complex and diverse entities and relations and propose a novel principle Flexible Translation to address these issues without increasing the model complexity.

Table 1 compares different knowledge representation learning methods by types of information each method considers. In the table, *Triple* means using each fact as the context when embedding an entity

(or a relation); *Path* stands for treating multiple steps of (undirected) linked entities as the context; *Edge* indicates using all relations that connect to the target entity as the context.

2.2 Graph Embedding

A growing literature has been studying the embedding of graph structure. For example, DeepWalk (Perozzi et al., 2014) uses local information obtained from truncated random walks to learn latent representations by treating walks as the equivalent of sentences. Line (Tang et al., 2015) is a network embedding method that preserves both the local and global network structures.

Although the graph embedding models use the network structures to learn the latent representations, the proposed models are still not suit for us to learn the embeddings of knowledge base. The first reason is that, the knowledge base embedding should learn the representations of both entities(vertices) and relations(edges), but network embedding models only learn the representations for vertices. Second, the assumptions which is the foundation of their models do not hold in knowledge base. For instance, in Line (Tang et al., 2015), it assumes that two vertices which are connected through a strong tie should be similar and be placed closely. But, in knowledge base the head entity and tail entity of a triple may be totally different, such as in triple *(Barack_Obama, Gender,Male)*, entity "Barack_Obama" and "Male" are not the same at all.

In this paper, we propose a novel approach to learn the representations of entities and relations by formulating a given knowledge base as a directed graph and leveraging the graph's structural information.

3 Our Approach

In the following, we present our approach, GAKE (Graph Aware Knowledge Embedding), for learning representations of a given knowledge graph. We describe our approach in steps, adding complexity, and start with necessary notations and definitions.

3.1 Preliminaries

A traditional knowledge graph is a set of triples, each describes a fact, as (*Barack_Obama*, *SpeakLanguage*, *English*). In this work, we use a directed graph to represent these facts by treating head/tail entities as vertices and relations as directed edges. More formally, we have

Definition 1 (Knowledge Graph) *A knowledge graph $G = (V, E)$ is a directed graph, where V is the set of vertices (i.e., entities), and E is the set of edges, where each directed edge $e = (v_i, v_j)$ represents the relation from the entity v_i to the entity v_j $(v_i, v_j \in V)$.*

The way to build a knowledge graph as we defined from given facts (or triples) is as follows: for each fact (h, t, r), where h and t are two terms to represent head entity and tail entity respectively, we first create two corresponding vertices v_i and v_j in the graph G, where i and j are unique index of h and t respectively. After that, we create a directed edge e, which represents the relation r, from v_i to v_j, along with a reverse relation r^{-1} from v_j to v_i. This is a common trick, which is similar to "back translation" in machine translation, to allow us to fully utilize the structural information of knowledge graph and improve the performance. The above process keeps running until all facts are included in the graph G.

Moreover, we use $s = (t, k)$ to represent a *subject* (i.e., a vertex or an edge) of the knowledge graph G, where t indicates subject type, and k is the index of the corresponding vertex or edge. Specifically, we let $t = 0$ to denote a vertex and let $t = 1$ to denote an edge. We use a set $S = \{s_i\}$ to represent all subjects in G.

Given a subject s_i, we define its *context* as a set of other subjects to indicates vertices or edges relevant to s_i:

Definition 2 (Graph Context) *Given a subject s_i, its graph context $c(s_i)$ is a set of other subjects relevant to s_i: $\{s_w | s_w \in S, s_w$ relevant to $s_i\}$.*

Different types of graph context defines the "relevance" between subjects differently. In this work, we use three types of graph context as examples, which will be introduced in detail later.

The objective of GAKE is to learn the representation of each subject in a given knowledge graph G according to its graph context. More formally, we target the problem of *Knowledge Graph Embedding* as

Problem 1 (Knowledge Graph Embedding) *Given a knowledge graph $G = (V, E)$, the problem of knowledge graph embedding aims to represent each vertex $v \in V$ and each edge $r \in E$ by a d-dimensional vector with real numbers.*

Then, we introduce te notations used in GAKE. In detail, s is a subject (i.e., a vertex or an edge); $C(s)$ means graph context of the subject s; $\phi(s)$ is embedding vector of the subject s; $\pi(C(s))$ is translation of subject s's context; $a(s)$ means attention model of a given subject s; θ is parameters used in the attention model.

3.2 Framework

We then introduce our approach in detail. Generally, the learning objective of GAKE is to predict missing subjects given by their context. (e.g., given two vertices, predicting whether there is a missing link from one to another). More formally, we define the probability of s_i given one of its contexts $c(s_i)$:

$$P(s_i|c(s_i)) = \frac{exp(\phi(s_i)^\top \pi(c(s_i))}{\sum_{j=1}^{|S|} exp(\phi(s_j)^\top \pi(c(s_i)))} \tag{1}$$

where $\phi : s_i \in S \longmapsto \mathbb{R}^{|S| \times D}$ is the embedding vector of a given subject s_i, and $\pi(\cdot)$ is a function that represents the translation of a graph context. In this work, we define $\pi(\cdot)$ as follows:

$$\pi(c(s_i)) = \frac{1}{|c(s_i)|} \sum_{s_j \in c(s_i)} \phi(s_j) \tag{2}$$

We then introduce how to construct different types of graph context. Specifically, to take advantage of the graph structure, given a subject s_i, we consider three types of context: *neighbor context* $C_N(s_i)$, *path context* $C_P(s_i)$, and *edge context* $C_E(s_i)$. Please notice that we take these context as examples while our approach is flexible and could easily be extended to other types of graph context.

Neighbor context. Given a subject s_i, taking an entity as an example, we regard each of its out-neighbors, along with their relations, as the *neighbor context*. Formally, when s_i is an entity, its neighbor context $c_N(s_i)$ is a pair of subjects (e, v), where v is an another vertex in G and e is a directed edge links s_i and v. One thing worth to notice is that neighbor context is equivalent to using triplets relevant to the given subject s_i.

The objective function of taking neighbor context into consideration is to maximize the log-likelihood of all subjects given by their neighbor contexts. Based on Eq. 1, we have

$$O_N = \sum_{s_i \in S} \sum_{c_N(s_i) \in C_N(s_i)} \log p(s_i|c_N(s_i)) \tag{3}$$

where $C_N(s_i)$ is the set of neighbor context of subject s_i.

Path context. A path in a given knowledge graph reflects both direct and indirect relations between entities. For example, the path $v_1 \xrightarrow{BornInCity} v_2 \xrightarrow{CityInState} v_3 \xrightarrow{StateInCountry} v_4$ indicates the relation "Nationality" between v_1 and v_4.

In this work, given a subject s_i, we use *random walk* to collect several paths starting from s_i. For more details, we first sample a integer L uniformly to indicates the length of the path (i.e., number of edges) we aim to generate. After that, at each step, the random walk will choose a neighbor randomly and will terminate once L edges have been collected. We define the path context $c_P(s_i)$ as a set of vertices and edges that are contained in a generated path. Similar methods are also used in (Spielman and Teng, 2004) and (Perozzi et al., 2014).

We then aim to maximize the probability of a subject s_i given by all paths starting from s_i:

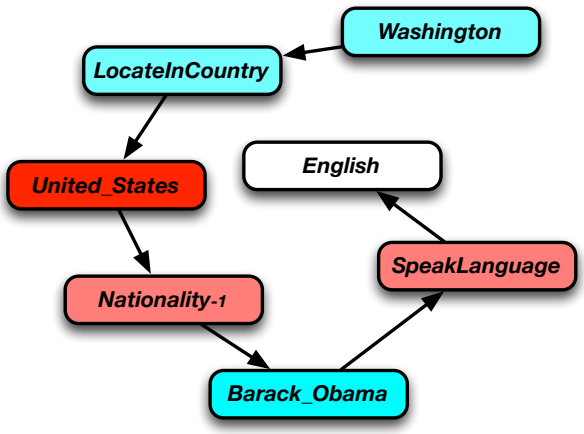


Figure 2: Illustration of the attention for a path context when predicting the entity "English". Darker cells indicate greater attentions.

$$O_P = \sum_{s_i \in S} \sum_{c_P(s_i) \in C_P(s_i)} \log p(s_i | c_P(s_i)) \tag{4}$$

Edge context. All relations connecting a given entity are representative to that entity. For example, the entity connected with "SpeakLanguage" are most likely to be a kind of languages. We define the edge context $c_E(s_i)$ of a subject s_i as all other subjects directly linked with s_i. When s_i is a vertex, $c_E(s_i)$ is a set of edges of s_i. Similar with other two types of graph context, we define the objective function of learning knowledge representation when considering edge context as follows:

$$O_E = \sum_{s_i \in S} \log p(s_i | c_E(s_i)) \tag{5}$$

Context extension. To utilize other types of graph context, one could first define $c(s_i)$ and the algorithm used to extract the context from the given knowledge graph G. After that, the remaining steps for knowledge representation learning would be exactly the same with other types of graph context. Thus, our framework is general and flexible to extend different types of graph context easily.

3.3 Attention Mechanism

So far, the translation of a graph context, $\pi(\cdot)$, takes the embedding results of each subject contained in the context equally. However, in reality, different subjects may have different power of influence to represent the target subject. As an example shown in Figure 1, in edge context, "SingSong" relation is more unique and preventative than "Nationality" as only few people like singers will connect with this "SingSong" while everyone has "Nationality". In this work, we model representative powers of different subjects in graph context by an *attention mechanism* (Ling et al., 2015; Hermann et al., 2015).

The basic idea of the attention mechanism is using an *attention model* $a(s_i)$ to represent how subject s_i selectively focuses on representing another subject s_j when s_i is a part of s_j's context (Kelvin Xu, 2015). In this work, we define the attention model $a(s_i)$ as

$$a(s_i) = \frac{\exp(\theta_i)}{\sum_{s_j \in C(s_i)} \exp(\theta_j)} \tag{6}$$

where θ is the parameters we aim to estimate. Figure 2 illustrates the attention for a path context when predicting the entity "English", where darker color indicates a greater attention. We see that entities like "Washington" and relations like "LocateInCountry" have less attentions, while the entity "UnitedStates" and the relation "SpeakLanguage" have greater attentions on representing "English".

We then re-define the translation of a given graph context, taking the embedding vector of each subject with different weights by further considering attention mechanism. Specifically, we have

$$\pi(c(s_i)) = \sum_{s_j \in c(s_i)} a(s_j)\phi(s_j) \tag{7}$$

3.4 Model Learning

To utilize these three types of context, we combine them by jointly maximizing the objective functions:

$$O = \lambda_N O_N + \lambda_P O_P + \lambda_E O_E \tag{8}$$

We define λ_T, λ_P and λ_N to represent the prestige of neighbor context, path context and edge context separately. We then use a Stochastic gradient descent (SGD) algorithm to estimate model parameters by optimizing Eq. 8. The derivatives are calculated using the back-propagation algorithm. The learning rate for SGD is initially set to 0.1 at first and decreased linearly with the number of training instances. Furthermore, to speed up the training process, we use Hierarchical Softmax (Bengio et al., 2006; Mikolov et al., 2013) to reduce the time complexity of normalization.

4 Experiments

We evaluate our proposed approach with two experiments: (1) triple classification (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015b), which determines whether a given triple is correct or not, and (2) link prediction (Wang et al., 2014; Xiao et al., 2016b), which aims to predict missing entities. For the data, we adopt dataset from Freebase (Bollacker et al., 2008): FB15K (Bordes et al., 2013). We then demonstrate the effectiveness of GAKE in the two tasks respectively. In all experiments, we set the dimension of embedding vectors to 100, $\lambda_T = 1$, , $\lambda_P = 0.1$ and $\lambda_E = 0.1$. The code and data used in this work are publicly available[1].

4.1 Triple classification.

Setup. In this task, given a knowledge base and a triple (h, r, t), we aim to determine whether it is correct (i.e., existing in the given knowledge base) or not. This task is also constructed in several previous work (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015b) and is widely used in many NLP scenarios such as question answering. For example, the result of triple classification can be directly applied to answer questions like "Does Taylor Swift publish the song Fifteen". We use the data set FB15K (Lin et al., 2015b), which contains 1,345 relations among 14,951 entities. We use 483,142 triples as training data to learn embeddings of all subjects. We then use 50,000 triples as validation data and 59,071 triples as test data.

We compare the proposed GAKE method with several state-of-art knowledge base embedding baselines, which includes NTN (Socher et al., 2013), TransE (Bordes et al., 2013), TransH (Wang et al., 2014), TransR (Lin et al., 2015b) and TransD (Ji et al., 2015). For each baseline method, we first learn representations of all entities and relations. For a query (h, r, t), we define a relation-specific threshold ρ_r by maximizing the classification accuracy on validation set. After that, we calculate the conditional probability $P(t|h, r)$ by regarding h and r as the context of t, while in GAKE, we construct the neighbor context with h and r's corresponding subjects. At last, we say (h, r, t) is positive (correct) if $P(t|h, r) \geq \rho_r$, where ρ_r is estimated according to the validation data.

Results. We show the evaluation results on triple classification in Figure 3. As the figure shows, it is clear that our approach outperforms others by 11.04% in terms of accuracy on average, as the graph context brings more information especially indirect relations between entities when learning the knowledge representations.

[1] https://github.com/JuneFeng/GAKE

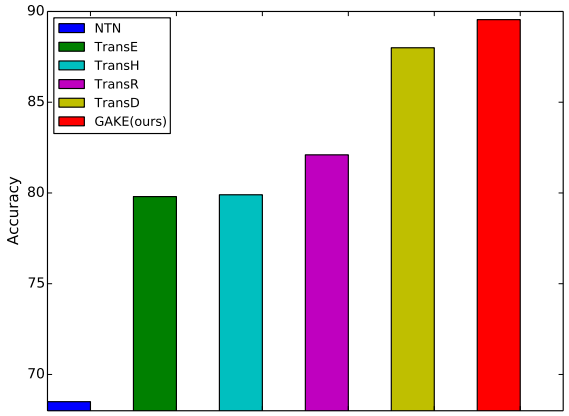


Figure 3: Evaluation results of triple classification.

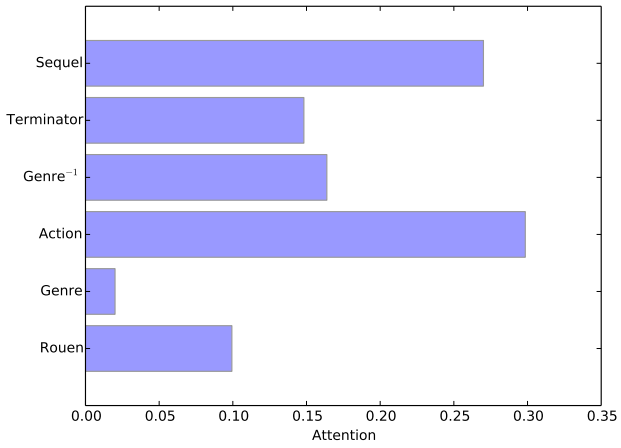


Figure 4: Attentions of subjects as the path context of the entity "Terminate2:JudgementDay".

Furthermore, to better understand the attention mechanism in our approach, we demonstrate attentions of 6 different subjects when they are regarded as the path context of the entity "Terminate2:JudgementDay", which indicates a movie. Figure 4 shows the results. From the figure, we see that two entities, "Action" and "Sequel", have the largest attention to represent the target entity, as "Action" reflects the type of the movie while only some of the movies have sequels. Meanwhile, the relation "Genre" has the least attention as every movie entity connects with "Genre".

4.2 Link Prediction.

Setup. As reported in (Bordes et al., 2011; Bordes et al., 2013), link prediction is to predict the missing h or t given (h, r) or (r, t) respectively. In this task, we conduct the evaluation by ranking the set of candidate entities in knowledge graph, instead of offering a best matching entity. This experiment is conducted on FB15K.

For the baseline methods, we compare our model models with the baselines which include Unstructured (Bordes et al., 2014), RESCAl (Nickel et al., 2011), SE (Bordes et al., 2011), SME(linear/bilinear) (Bordes et al., 2014), LFM (Jenatton et al., 2012) and TransE (Bordes et al., 2013).

Following the protocol in TransE (Bordes et al., 2013), for each test triple (h, r, t), we replace the head entity h by every entity in the knowledge graph, and rank these corrupted triples in descending order by the similarity score which is given by f_r. Similarly, we repeat this procedure by replacing the

Table 2: Experimental results on link prediction.

Data Sets	FB15K			
Metric	Mean Rank		Hits@10(%)	
	Raw	Filter	Raw	Filter
Unstructured (Bordes et al., 2014)	1,074	979	4.5	6.3
RESCAl (Nickel et al., 2011)	828	683	28.4	44.1
SE (Bordes et al., 2011)	273	162	28.8	39.8
SME (linear) (Bordes et al., 2014)	274	154	30.7	40.8
SME (bilinear) (Bordes et al., 2014)	284	158	31.3	41.3
LFM (Jenatton et al., 2012)	283	164	26.0	33.1
TransE (Bordes et al., 2013)	243	125	34.9	47.1
GAKE (ours)	**228**	**119**	**44.5**	**64.8**

tail entity t. After collecting all these triples, we use two evaluation metrics: the mean rank of the correct entities (denotes as *Mean Rank*); the proportion of correct entity ranks within 10 (denotes as *Hits@10*). We expect lower *Mean Rank* and higher *Hits@10* for a better predictor. However, some corrupted triples should be considered as correct ones, since they actually exist in knowledge graph. Ranking such triples ahead of the original correct one should not be counted as an error. To eliminate such cases, we filter out those corrupted triples which appear either in the training, validation or test datasets. We term the former evaluation setting as "Raw" and the latter as "Filter".

Results. Table 2 lists the results on link prediction. It shows that our method GAKE, gets better experiment results than other baselines including Untructured, RESCAL, SE, SME (linear/bilinear), LFM and TransE models. The result demonstrates the superiority of the idea that fully utilizes the graph information to learn representations for entities and relations.

5 Conclusion

In this paper, we propose a graph aware knowledge embedding model to address graph-level contexts. Most existing methods regard knowledge graph as a set of independent triples, and ignore the indirect dependency between subjects (i.e., entities or relations). To deal with this issue, we propose a novel method, GAKE, for learning the representation of a given knowledge graph by formulating a given knowledge base as a directed graph and leveraging graph context, which includes path context, neighbor context, and edge context. We further design an attention mechanism to learn representative power of different subjects. To validate our model, we conduct extensive experiments on benchmark datasets for two tasks, i.e., triple classification and link classification. Experimental results show that GAKE outperforms several state-of-art knowledge embedding methods.

Learning knowledge graph representations is an interesting and new research direction, and there are many potential future directions for this work. For instance, it will be interesting to incorporate the power of explicit knowledge (Wang et al., 2015) into our method to further improve the performance. In addition, the framework of this model is flexible to handle sundry information except the graph context. In other words, we can also build text context by using descriptions of entities or additional text information from other sources like Wikipedia.

6 Acknowledgements

This work was partly supported by the National Basic Research Program (973 Program) under grant No. 2013CB329403, the National Science Foundation of China under grant No.61272227/61332007 and an another 973 Program under grant No. 2015CB352300.

References

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *AAAI*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Jun Feng, Minlie Huang, Mingdong Wang, Mantong Zhou, Yu Hao, and Xiaoyan Zhu. 2016. Knowledge graph embedding by flexible translation. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016.*, pages 557–560.

Kelvin Gu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. *EMNLP*.

Shu Guo, Quan Wang, Bin Wang, Lihong Wang, and Li Guo. 2015. Semantically smooth knowledge graph embedding. In *Proceedings of ACL*, pages 84–94.

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

Rodolphe Jenatton, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski. 2012. A latent factor model for highly multi-relational data. In *Advances in Neural Information Processing Systems*, pages 3167–3175.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of ACL*, pages 687–696.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *2016 AAAI Spring Symposium Series*.

Ryan Kiros Kyunghyun Cho Aaron C. Courville Ruslan Salakhutdinov Richard S. Zemel Yoshua Bengio Kelvin Xu, Jimmy Ba. 2015. Show, attend and tell: Neural image caption generation with visual attention. *ICML'15*.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2015a. Modeling relation paths for representation learning of knowledge bases.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion.

Wang Ling, Lin Chu-Cheng, Yulia Tsvetkov, and Silvio Amir. 2015. Not all contexts are created equal: Better word representations with variable attention. In *ACL*. Citeseer.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Daniel A Spielman and Shang-Hua Teng. 2004. Nearly-linear time algorithms for graph partitioning, graph ification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90. ACM.

Ilya Sutskever, Joshua B Tenenbaum, and Ruslan Salakhutdinov. 2009. Modelling relational data using bayesian clustered tensor factorization. In *Advances in neural information processing systems*, pages 1821–1828.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee.

Kai Chen Greg Corrado Jeffrey Dean Tomas Mikolov, Ilya Sutskever. 2013. Distributed representations of words and phrases and their compositionality. *NIPS'13*, pages 3111–3119.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. *ACL Association for Computational Linguistics*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Chenguang Wang, Yangqiu Song, Ahmed El-Kishky, Dan Roth, Ming Zhang, and Jiawei Han. 2015. Incorporating world knowledge to document clustering via heterogeneous information networks. In *KDD*, pages 1215–1224.

Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu. 2015. TransA: An adaptive approach for knowledge graph embedding. *arXiv preprint arXiv:1509.05490*.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016a. From one point to a manifold: Knowledge graph embedding for precise link prediction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1315–1321.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016b. TransG : A generative model for knowledge graph embedding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Ranking Responses Oriented to Conversational Relevance in Chat-bots

Bowen Wu[1], **Baoxun Wang**[1], **Hui Xue**[2]
[1]Microsoft AI and Research Group, Beijing, China
[2]Microsoft Research, Beijing, China
{bowenwu, baoxwang, xuehui}@microsoft.com

Abstract

For automatic chatting systems, it is indeed a great challenge to reply the given query considering the conversation history, rather than based on the query only. This paper proposes a deep neural network to address the context-aware response ranking problem by end-to-end learning, so as to help to select conversationally relevant candidate. By combining the multi-column convolutional layer and the recurrent layer, our model is able to model the semantics of the utterance sequence by grasping the semantic clue within the conversation, on the basis of the effective representation for each sentence. Especially, the network utilizes attention pooling to further emphasis the importance of essential words in conversations, thus the representations of contexts tend to be more meaningful and the performance of candidate ranking is notably improved. Meanwhile, due to the adoption of attention pooling, it is possible to visualize the semantic clues. The experimental results on the large amount of conversation data from social media have shown that our approach is promising for quantifying the conversational relevance of responses, and indicated its good potential for building practical IR based chat-bots.

1 Introduction

There exist two query intentions in Intelligent Agents: the task completion oriented intention and the open-domain chat intention. As the applications of dialog systems, task completion oriented agents are designed to accomplish users' requirements in a few rounds of conversations. This kind of intentions reflect users' basic needs on the agents, thus studies on dialog systems have a longer history and achieved great process (Weizenbaum, 1966; Ferguson et al., 1996; Shawar and Atwell, 2007; Williams, 2010).

The open-domain chat intention, by contrast, represents users' communicating needs. Apparently, automatic chatting systems with good using experience will significantly attract people's interests, even make people form the habits of communicating with agents, hence it is possible to be a new platform for any task-oriented services to plug in (see Duer[1]). One challenge directly brought by open-domain Chat-bots is, indeed, user queries can be related to any topic in any possible forms, that is, it's NOT wise to transform chatting queries into slot-value sequences to further trace users' intentions within the conversation, as task-oriented dialog systems do.

The even more essential challenge chat-bots have to face is to guarantee the semantic and logic continuity of conversations, that is, a response from bots should be relevant with both the adjacent query and the corresponding short conversation history. Actually, such "context-aware" chatting ability is the critical feature of a human-like chat-bot, thus much attention has been paid on this task. The basic requirement for chat-bots is to semantically understand conversations like humans, which is abstracted as the conversation modeling problem. This paper discusses the approaches to addressing the context-aware chatting problem, by investigating and simulating the inner mechanism of human conversations.

Basically, two methodologies can be utilized to provide responses according to the given query, or more complicated, conversation history as discussed in this paper. The first method is to directly generate

[1]http://duer.baidu.com/

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 652–662, Osaka, Japan, December 11-17 2016.

Q0: 今天*天气好差*，天气**转凉**的**跨度好大**!
A0: 是不是*很恐怖*，今天已经**结冰了**。
Q1: 那你最近要怎么度过？
A1: 那就冬眠吧。
A2: 我一会去上班。

(a) Raw case in Chinese.

Q0: The *weather is so bad* today, the **temperature drops so much**!
A0: *Terrible*, isn't it? The temperature **goes down to freezing**.
Q1: Then, what are you going to do?
A1: Let's go to hibernate then.
A2: I'll go to work later.

(b) Translated to English.

Figure 1: A conversation example.

responses for a given query and its context (Ritter et al., 2011; Vinyals and Le, 2015; Shang et al., 2015), which provides an end-to-end solution for chat-bots. Despite its meaningful integrated architecture, it seems still a great challenge for generation based approaches to give responses with good readability and diversity. This problem can be directly addressed by another option, that is, finding proper methods to rank candidate responses selected from large amount of human dialog utterances by information retrieval (IR) method (Ji et al., 2014; Xian et al., 2016). Such Candidate Re-Ranking based solutions (Lowe et al., 2015; Sordoni et al., 2015; Hu et al., 2014) are of great value for building the practical chat-agents like Duer and XiaoIce[2], for which readability and diversity of responses are critical metrics.

For both generation and re-ranking approaches, indeed, their very basis is capturing the semantic clues within conversations, so as to select proper responses or generate them directly, and this procedure is generally named as "conversation modeling". As denoted by Grosz and Sidner (1986), the sequential utterances' structure, purposes and the state of focus of attention are the key components in a discourse. Correspondingly, to provide a conversationally reasonable response for a given session, the following abilities are needed for conversation modeling approaches: a) achieving the semantic representations of short sentences with the oral style; b) obtaining the focus of the entire dialog session; and c) selecting or generating responses based on the modeling of utterance sequences.

This paper aims to explore an integrated model framework to achieve the above goals, so as to find the context-aware responses from the candidates given by IR modules. Especially, our model will pay much attention to obtaining dialog focuses, that is, the model is designed to capture the semantic clues implicitly existing in human conversations. Such clues are always composed of phrases scattered in the utterances of conversations, and play a significant role in determining whether a given candidate is context-aware or not. Take the session in Figure 1 for example, some implicit clues flowing throughout the context (marked in **bold**) can be observed. Some of them like "temperature drops", "goes down to freezing", are more about the conversational topic, meanwhile, the words "weather", "temperature" stand for the key ingredient of the sentences. But, the keywords marked in *italic* maybe not helpful for judging A1 as a better response than A2. By contrary, we should task focus on some relatively meaningless phrases, such as "so much". So the clues detection is related with the end-to-end modeling task, and it is difficult and useless to treated this as a pre-processing.

This paper presents a convolution neural network (CNN) with attention pooling strategy to capture semantic clues within conversations by performing the sequential learning, so as to pick out context-aware replies based on the corresponding dialog history. According to the analysis on semantic relationships of the historical contexts, present-posted query and candidate responses, we propose to employ attention mechanism to model sentence upon convolutional layers, so as to provide meaningful semantic representations of contexts. After that, a Gated Recurrent Unit (GRU) based layer accomplishes the sequence modeling for response selection. Experiments with various structures of sentence and sequence modeling are conducted on the dataset from a Chinese Social Network Service (SNS), which has shown the good potential of our approach. Especially, due to the utilization of attention pooling, the obtained conversational clues can be visualized, which is very useful for the conversational state tracing and the interpretability of ranking results.

[2] http://www.msxiaoice.com/

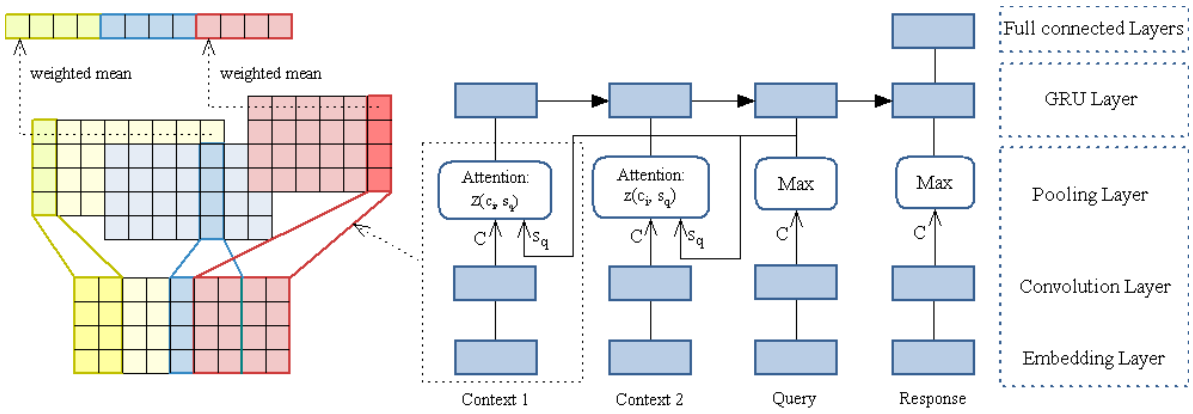


Figure 2: Architecture for modeling the conversation.

2 CNN with Attention Pooling for Quantifying Conversational Relevance

As mentioned in Section 1, the three abilities are needed for modeling open domain conversations, and our motivation is to design an integrated neural network framework to provide these abilities for response selection. This section will detail our approach that mainly takes Chinese characters as basic input elements[3], as well as the specially designed pooling strategy.

2.1 Model Architecture

As illustrated by Figure 2, the architecture of our model is composed of the following three modules:

Sentence Representing: As the essential part of our deep learning architecture, the Sentence Representing module aims to basically map short sentences into the real-valued semantic space. Moreover, the sentence modeling part in our work has to be able to locate the conversationally essential phrases of utterances, which can be jointly absorbed by the upper layer as the semantic clue for modeling conversations. For this purpose, we take the multi-width convolutional function and special designed pooling functions performing on the character-embedding layer, to build the sentence representing part.

CNN based sentence models have achieved success in some NLP tasks (Collobert et al., 2011; Kalchbrenner et al., 2014; Hu et al., 2014), especially, the recent character-level CNN (Kim et al., 2015; Zhang and LeCun, 2015) has even got some state-of-the-art results. Our sentence representing module continues this series of work, employing an one-dimensional valid convolutional layer over char embeddings.

Suppose there are n characters in a sentence, and let $x_i \in \mathbb{R}^k$ be the k-dimensional char vector corresponding to the i-th character, X present a $n \times k$-dimensional matrix made up of x_i, and $w \in \mathbb{R}^{1*m}$ is the weight of a convolutional filter (we'll use $m \in \{2, 3, 5\}$ to represent the bi-gram, tri-gram and 5-gram level abstraction). C stands for the output of this feature map, and represents the meaning of sub-phrases, each element vector c_i is computed by:

$$c_i = f(x_{i::i+m-1} \cdot w + b) \tag{1}$$

Where $b \in \mathbb{R}$ indicates the bias term and f stands for a non-linear function, e.g., the rectifier. Various potential features of the words or phrases are generated by multiple filters. For each of these candidate features will be screened by higher level pooling layers.

At present, the max pooling and average pooling are widely applied. In image modeling scenarios, max pooling can depict the texture better, while average pooling results represent more information about background (Boureau et al., 2010). The heuristics can be applied to NLP tasks similarly, that is, the whole meaning of sentences can be obtained by average strategy; on the other hand, the max pooling concentrates on the significant points relied on established tasks, and this is the reason for max pooling being utilized for solving many challenging text classification problems (Collobert et al., 2011; Kalchbrenner et al., 2014; Zhang and LeCun, 2015).

[3]Different from the European languages such as English, each Chinese character may keep special semantic independently.

In the conversation scenario, context-aware response selection basically relies on two major aspects: a) the relevance between the present query and the candidate response; and b) the additional background information provided by the dialog history. For quantifying the semantic relevance between the present query and candidate, the key phrases within them are playing the great role, and the irrelevant words should be ignored. By contrast, the phrases in a history utterance tend to act as a whole background, to supply and maintain a topic for the conversation. Consequently, as illustrated by Figure 2, we employ the max pooling for extracting the key points in the present query and the candidate for relevance judgment, and for the context sentences, the average pooling strategy is taken to introduce complete background.

Indeed, for each word in the dialog history (previous two sentences as shown by Figure 2), its contribution varies for selecting context-aware candidates, thus it is reasonable to give different weights to the words in the dialog history. For this purpose, in the sentence modeling module demonstrated in the left colorized part of Figure 2, we further present a new attention pooling strategy to learn the weights of each word, according to its contribution for determining whether a candidate is conversationally relevant. This attention pooling strategy will be detailed in Section 2.2.

Conversation modeling: Generated by the sentence modeling layer, the sentence vectors are adopted by a GRU layer for sequentially modeling the entire conversation. The motivation for selecting GRU is to naturally utilize its internal memory to process sequential inputs, and its performance is comparable with LSTM by controlling the gradient vanishing/exploding problems of ordinary RNNs (Bengio et al., 1994; Chung et al., 2014). To further investigate the balance between computational complexity and modeling ability, we also explored the RNN with identity initialized weights (iRNN for short) in practice as Mikolov et al. (2014) did.

Candidate Ranking: Given a sequence representation from conversation modeling, the candidate ranking module takes the full-connected layer to quantify the relevance of candidate responses. We employ the cross entropy as the point-wise ranking loss, and various ranking objective functions can be used to learn the parameters of the whole model.

2.2 Attention Pooling

As mentioned in the previous section, we wish to enhance the modeling of the context utterances for better understanding of the whole conversation, by employing the attention mechanism to learn the weights of words reasonably, meanwhile, it is possible to visualize the semantic clues in conversations according to the learnt weights. The attention strategies have been widely used in machine translation (MT) (Bahdanau et al., 2014; Meng et al., 2015; Li et al., 2015) and question answering (Weston et al., 2014; Hermann et al., 2015; Kumar et al., 2015). Especially for the Encoder-Decoder framework, the attention mechanism may introduce weighting functions of the encoding state and current decoding hidden state, so as to determine the elements that should be focused on.

This paper proposes a new pooling function with attention mechanism to model conversational contexts. Noticing that a posted utterance mainly pays attention to some specific points of the previous utterances in the same session, our pooling approach aims to emphasize such points while obtaining the whole meaning of sentences in the context. As mentioned in Section 2.1, the average pooling can cover the overall information, and our attention pooling tries to assign weights to the words and perform weighted averaging, to find the more important words or phrases in contexts.

The given c_i is the i-th combination of chars as defined previously, and $s_q \in \mathbb{R}^{(k(n-m+1))*1}$ presents the sentence embedding of the posted query upon the max pooling layer. The feature set $z(c_i, s_q)$ for weighting and the attended result a_i can be computed following:

$$z(c_i, s_q) = [c_i, s_q, c_i^T W^{(a)} s_q], \quad a_i = \frac{e^{(z(c_i,s_q)\cdot W^{(1)}+b^{(1)})}}{\sum_{j=0}^{n-m+1} e^{(z(c_j,s_q)\cdot W^{(1)}+b^{(1)})}} \tag{2}$$

Where $W^{(a)} \in \mathbb{R}^{(k(n-m+1))*k}$, $W^{(1)} \in \mathbb{R}^{(k(n-m+2)+1)*1}$, and $b^{(1)}$ is the bias term. The sentence vector

$s_j^{context}$ by the j-th convolutional filter is the weighted average of each c_i instead of ordinary mean:

$$s_j^{context} = \sum_{l=0}^{n-m+1} c_l^j \circ a_l^j \tag{3}$$

The $\circ$ indicates element-wise dot, c_l^j and a_l^j are computed by the j-th filter.

For similar purposes, there are some works take the attention strategy on RNN based dialogue generation (Yao et al., 2015; Shang et al., 2015), different from these works, the model described in this paper aims to apply the attention pooling upper the character level convolutional layer. Besides, Yin et al. (2015) applied an attention method upper convolutional layers to reflect the sub-phrases' interaction between sentence pairs, while we mainly focus on the sentence modeling about contexts in the conversations, and we proposed different attention function to model the relationship between context and query, other than matching units in two feature maps.

3 Experiments

Our proposed model is utilized on the response selecting task, that is, our goal is to distinguish the conversationally relevant responses from the irrelevant ones.

3.1 Dataset & Metrics

The dataset contains totally 1,025,000 sessions collected from the threads in a popular Chinese SNS, each session is composed of 4 utterances including a one-turn context, a present query and a context-aware response. Each sentence's character count varies from 3 to 50, with 10 as average. All the examples used in this paper are included in the dataset. For each conversation, we replace the response with another one randomly sampled from the corpus as the negative sample like (Hu et al., 2014; Lowe et al., 2015; Al-Rfou et al., 2016). This operation repeats 4 times, and we duplicate the real conversations 4 times as positive samples. For all the experiments, we split our dataset into training, validation and test sets, with 8,000,000, 100,000 and 100,000 conversations respectively.

Except evaluating the classified performance by accuracy, we introduce **1 in t P@k** to evaluate the ranking ability with $t-1$ negative cases, where P@k denotes the precision at top k.

3.2 Competitor Models & Parameter settings

The baseline approaches taken by our work can be basically categorized into two groups: the classic methods include the Logistic Regression (LR) models trained on Tri-Gram based TF-IDF features or LDA (Blei et al., 2003) based distributed representations of sentences. Besides, several neural network combinations of different components' implementation are adopted in our experiments, whose general frameworks are the same with the one illustrated in Figure 2, and their details are given as follows:

- GRU+MLP: This model takes GRU to model sentences, which is different from our CNN based sentence modeling layers. Above that, the multi-layer perceptron (MLP) is used to model the conversations without consideration about the sequential characteristic of conversations;

- GRU+iRNN/GRU: In these models, iRNN or GRU takes the position of conversation modeling module, and the sentences modeling part still employs GRU;

- Attention GRU+iRNN: Employing GRU with attention for modeling the sentences in contexts and iRNN for sequence modeling;

- CNN+iRNN: This model takes iRNN for conversation modeling, and in the CNN based sentence representation module, several pooling strategies are tried, including max pooling, mean pooling, and their mixture, to replace our attentional average pooling in the architecture in Figure 2;

Group	Model	Accuracy	1 in 2 P@1	1 in 5 P@1	1 in 5 P@2
#1	Random	50.0%	50.0%	20.0%	40.0%
	TF-IDF+LR	53.2%	58.4%	24.2%	43.0%
	LDA+LR	59.7%	66.7%	35.6%	52.8%
#2	GRU+MLP	65.8%	72.0%	43.4%	67.6%
#3	GRU+iRNN	72.5%	80.3%	54.7%	78.5%
	bi-GRU+iRNN	**73.6%**	**81.7%**	**55.7%**	**80.2%**
	GRU+GRU	**75.8%**	**84.5%**	**63.1%**	**83.3%**
	Attention GRU+iRNN	70.3%	78.0%	51.5%	75.0%
#4	Max CNN+iRNN	73.1%	81.0%	55.1%	80.0%
	Mean CNN+iRNN	73.5%	81.7%	55.8%	80.8%
	Mix CNN+iRNN	74.2%	82.9%	56.9%	81.9%
	Attention CNN+iRNN	**75.7%**	**84.3%**	**60.5%**	**83.5%**
	Attention CNN+GRU	**78.6%**	**87.0%**	**65.1%**	**86.1%**

Table 1: Comparison of different approaches on the context-aware candidate selection task.

The implementation with online learning for LDA (Hoffman et al., 2010) is used for our experiments, with α and β fixed at 0.01 and the number of topics $K = 400$. In all the neural network based experiments, we initialize the learning rate with 0.005, and the network is trained with the Adam update rule (Kingma and Ba, 2014). Early stopping (Giles, 2001) and Dropout (Hinton et al., 2012) are taken to prevent overfitting. As recommend by Krizhevsky et al. (2012), we utilize ReLU as the non-linear active function of convolutional and full-connected layers, and $tanh$ is used for the hidden states of GRU. The dimension of character embedding is 100 for all the NN models. For CNN based sentence modeling layers, the widths of the filter windows are set to 2, 3 and 5 in parallel, and the pooling window covers all the element after convolutional function. The GRU based sentence modeling module holds a 100-unit hidden layer. For conversation modeling layers, the size of the hidden states of iRNN and GRU is set to 300. Finally, the size of the hidden layer of MLP is 50.

3.3 Results & Analysis

Table 1 details the results, and groups them into four categories for the following analysis perspectives:
(a) traditional methods vs. neural networks based ones for modeling short sentences with oral style;
(b) GRU vs. CNN on sentence representation;
(c) aligning sentence embeddings vs. modeling sentence sequences for conversation understanding;
(d) separately modeling sentences vs. sentence representation with attention mechanism;
(e) GRU with attention vs. CNN with attention for sentence representation.

From the results in Table 1, it can be observed that all the models adopting neural network components have notably outperformed TFIDF-LR and LDA-LR. This phenomenon reflects the difficulty of modeling the short sentences with oral style, since the information introduced by pure lexical features introduce is very limited for such text samples. By contrast, both GRU and CNN have the ability of catching the richer semantic information in short texts, according to the layer-by-layer learning upon the distributed character embeddings. Thus, the comparison of aspect (a) shows NN based sentence models are more suitable for conversation utterances.

Further, by comparing GRU+iRNN with CNN+iRNN, aspect (b) tries to figure out which deep learning architecture works better as the sentence modeling module, and our observation is CNN outperforms GRU on the whole task. We ascribe this result to the information bias of sentence embeddings generated by GRU, that is, GRU tends to pay more attention to the words in the end of a sentence. However, for the task discussed by this paper, complete semantics provide more help to context-aware candidate selection as discussed in Section 2.1. The limited improvement of CNN with max pooling also supports our inference. This problem has been partially solved by introducing bi-direction GRU (see bi-GRU+iRNN), it can also be seen that the special defined mixture pooling strategy (Mix CNN+iRNN) can achieve

context	The weather is so bad today, the temperature dips so much! Terrifying, isn't it? The temperature goes down to freezing.				
query	Then, what are you going to do?				
Mix CNN+iRNN	Attention CNN+iRNN	Label	Rank Label	No.	Response
0.782	**0.826**	1	1	#1_1	Let's go to hibernate then.
0.573	0.422	0	2	#1_2	I'll go to work later.
0.150	0.029	0	3	#1_3	How to prove it?
context	What's the point of keeping my phone if it can't connect to Wifi anymore? So what are we waiting for? Buy a new one!				
query	You sponsor me.				
0.808	**0.940**	1	1	#2_1	No phone, no money!
0.840	0.824	1	2	#2_2	How to sponsor?
0.384	0.226	0	3	#2_3	Curiously, this's across both 3G and Wifi.

Table 2: Samples of the response selection. Sentences are translated to English for better understanding.

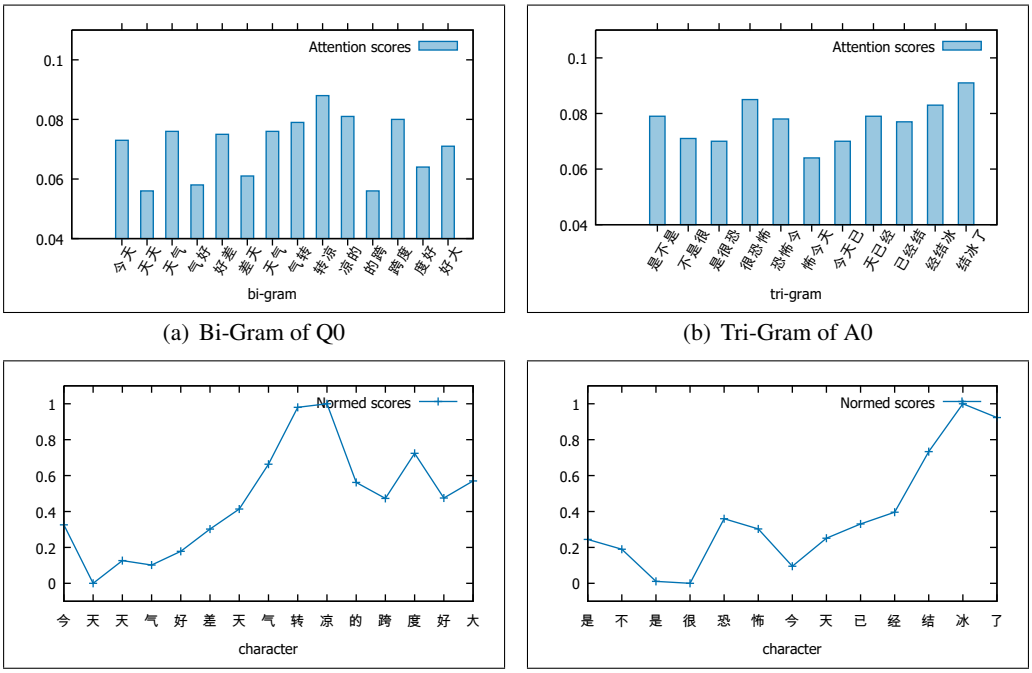
more competitive results, because the advantages of different pooling methods have been integrated for complete semantic representation.

We can easily observe the huge gap between the performances of models in group #2 and group #3. All of these methods take GRU as the sentence modeling layer, but the ones in group 3 adopt RNN based layers for conversation modeling. Since the conversation modeling task is naturally a sequential modeling problem, it is reasonable that models with GRU components achieve better results, which is the motivation of aspect (c). Another observation is iRNN performs fairly well as the conversation modeling layer, with good potential for practical usage. Besides, the comparisons suggest that GRU is indeed more powerful for modeling conversations.

As shown in the results of group #3 and #4, the attention pooling is helpful to improve the precisions, especially on 1 in 5 P@1, which meets the expectation of aspect (d). Nevertheless, when considering aspect(e), it is noticed that GRU with attention (attention GRU+iRNN) gets unsatisfying performance comparing with the ones without attention function. This observation is different from the general impression, since quite a number of studies adopting attention mechanism have good results (Bahdanau et al., 2014; Yao et al., 2015; Hermann et al., 2015; Kumar et al., 2015). We attribute the performance gap to the character-level inputs. In detail, since the attention function is applied on each hidden state, which mainly contains the information of the current input, despite a small amount of previous information involved. Meanwhile, a single Chinese character keeps very limited semantic information, thus the information obtained by the attention function of GRU is incomplete, reflecting some single characters in fact. By contrast, the convolution layer can extract the combinations of characters indicating words or even phrases, and the attention function performed upon such combinations is possible to figure out the important segments with complete semantics, for the upper layer to understand the whole sequence. This is the main reason for our proposed "Attention CNN+GRU" model finally get the best results.

3.4 Case study

To get a better intuition for what the model and attention pooling is learning, we give some cases to illustrate the details. Table 2 gives two groups of query-response pairs, with the predicted scores by Mix CNN+iRNN (MCNN) and Attention CNN+iRNN (ACNN) models. **Label=1** indicates the suitable response to the given context, and the **Rank Label** reflects the candidates' recommendation degrees. It can be seen that scores of both models are aligned with the overall ranking trend, which reflects the models having the ability to quantify the conversational relevance reasonably. It should be noted that the scores given by ACNN are more closed to the labels. More specifically, all the predictions of ACNN are correct, while MCNN makes some incorrect decisions on #1_2 and #2_2. Different from other candidates, the sentences (#1_2 and #2_2) are very sensitive to contexts, in other words, they are natural to answer the corresponding query without considering the contexts. #1_2 has wandered off topic, by contrary, #2_2 can also be a suitable response even it seems #2_1 have better maintenance of the information in the conversation flow. The results of ACNN reflect these phenomena, as the ranking scores express the right disposition and offer higher scores to the more appropriate responses. Besides, the gaps within the



(a) Bi-Gram of Q0

(b) Tri-Gram of A0

(c) Relative attention on chars of Q0

(d) Relative attention on chars of A0

Figure 3: Attention values on the Bi-Gram of Q0 and Tri-Gram of A0 are detailed in sub-figure (a) and (b), where y-axis is the attention value. Sub-figure (c) and (d) give the focus on each char.

groups of ACNN are larger than those scores by MCNN. All these enhancement of the model's ability can be ascribed to the capability of differentiating the matching degree not only correspond to the posted query but also corresponding with to the whole session, obtained by attention pooling to lay particular emphasis on the phrases that the previous conversation focused on.

In order to further illustrate the effect of attention pooling, Figure 3 details the distribution of probabilities given by the attention function to the phrases of the case in Figure 1. Actually, the extra advantage of our framework is that we can locate the key information for candidate selecting, by visualizing the attention weights of phrases and performing proper transforming on them. In detail, we firstly visualize the pooling weights for each character combination in each convolutional kernel as shown by Figure 3(a)-(b), then we assign the weights averaged by the frequencies of the characters in each kernel, and get the curves in Figure 3(c)-(d). According to this operation, we can clearly see which positions in the context considered to be more important when given a query. While it can be seen that, obvious higher weights appeared along the positions of significant words and phrases (marked in **bold**) in Figure 1. Another observation from the histograms is the overall scores of the essential words and phrases are higher than the other char-combinations, which indicates the sentence embedding is mostly draw from the meaningful words. This group of results shows the attention pooling, rather than simple mean pooling, are effective to draw focus on the words and phrases composing the semantic clue in a given conversation.

4 Related Work

Before open-domain chat agents, the task-completion oriented dialog system has been a subject of study for a long time, and most of these studies pay attention to particular vertical domains. Such as *ELIZA* based on simple text parsing rules (Weizenbaum, 1966). Ferguson et al. (1996) built a rule-based system to solve problems in transportation domain; Shawar and Atwell (2007) leverage answer template in generating the *ALICE*; Williams (2010) focus on tracing pre-defined dialogue state. These systems rely on pre-designed rules or templates, which can be hardly generalized on open domain chat-style robots.

Until recently, some works such as (Ritter et al., 2011) demonstrate that the sentences can be generated corresponding to a given post or context using MT techniques. Since the encoder-decoder based Recurrent Neural Networks (RNN) outperforms other methods on MT tasks in the past two years (Bahdanau

et al., 2014; Sutskever et al., 2014), several approaches are directly applied on the conversation modeling task by concatenating the context that modeled by one recurrent encoder (Vinyals and Le, 2015; Shang et al., 2015). Yao et al. (2015) and Serban et al. (2015) model the sentences of context separately by an encoder, and address the sequential embeddings by cumulative hidden units. The main drawback of these approaches is they can't guarantee the readability and variety of generated sentences.

By contrast, the response ranking strategies can avoid the problems caused by direct generation, since this methodology tries to pick up reasonable responses from the human-generated sentences. Ji et al. (2014) proposed an IR approach to generate candidates, and rank them with many kinds of features such as MT, keywords, similarity, etc; Sordoni et al. (2015) and Luan et al. (2016) directly utilize the generating loss of the response for ranking, with the adjusted RNN based encoder-decoder framework. Generally, the architectures introducing CNN or RNN to learn representations of sentences and modeling the relevance of context and candidate response on hyper layers, tend to achieve state-of-the-art performance (Hu et al., 2014; Lowe et al., 2015).

5 Conclusion

In this paper, we have presented a deep learning architecture to quantify the conversational relevance of responses for candidate ranking. The contributions of this paper can be summarized as follows: a) According to the investigation on the role of contexts in conversations, this paper proposes the attention pooling to provide more reasonable context representations, by taking the phrases' different contributions to the semantic clue into consideration. b) We have combined the multi-column convolutional layer and the GRU based layer to build the candidate ranking model, so as to take the advantages of both CNN on sentence modeling and RNN on sequence modeling. c) The proposed model enables the visualization of the achieved essential phrases, and our analysis on them shows the importance of capturing semantic clues for finding conversationally relevant responses.

Acknowledgements

We thank the anonymous reviews, along with Deyuan Zhang and Zhen Xu for their valuable comments and suggestions that helped to improve the quality of this paper.

References

Rami Al-Rfou, Marc Pickett, Javier Snaider, Yun-hsuan Sung, Brian Strope, and Ray Kurzweil. 2016. Conversational contextual cues: The case of personalization and history for response ranking. *arXiv preprint arXiv:1606.00372*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. 2010. Learning mid-level features for recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2559–2566. IEEE.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

George Ferguson, James F Allen, Bradford W Miller, et al. 1996. Trains-95: Towards a mixed-initiative planning assistant. In *AIPS*, pages 70–77.

Rich Caruana Steve Lawrence Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, volume 13, page 402. MIT Press.

Barbara J Grosz and Candace L Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational linguistics*, 12(3):175–204.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Matthew Hoffman, Francis R Bach, and David M Blei. 2010. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, June.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Yi Luan, Yangfeng Ji, and Mari Ostendorf. 2016. Lstm based conversation models. *arXiv preprint arXiv:1603.09457*.

Fandong Meng, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, and Qun Liu. 2015. Encoding source language with convolutional neural network for machine translation. *arXiv preprint arXiv:1503.01838*.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. 2014. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*, pages 583–593. Association for Computational Linguistics.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Building end-to-end dialogue systems using generative hierarchical neural network models. *arXiv preprint arXiv:1507.04808*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Bayan Abu Shawar and Eric Atwell. 2007. Chatbots: are they really useful? In *LDV Forum*, volume 22, pages 29–49.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Joseph Weizenbaum. 1966. Elizaa computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Jason D Williams. 2010. Incremental partition recombination for efficient tracking of multiple dialog states. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 5382–5385. IEEE.

Li Xian, Mou Lili, Yan Rui, and Zhang Ming. 2016. Stalematebreaker: A proactive content-introducing approach to automatic human-computer conversation. *arXiv preprint arXiv:1604.04358*.

Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. 2015. Attention with intention for a neural network conversation model. *arXiv preprint arXiv:1510.08565*.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Probabilistic Prototype Model for Serendipitous Property Mining

Taesung Lee[*] **Seung-won Hwang** **Zhongyuan Wang**[†]
IBM T. J. Watson Research Center Yonsei University Microsoft Research
taesung.lee@ibm.com seungwonh@yonsei.ac.kr wzhy@outlook.com

Abstract

Besides providing the relevant information, amusing users has been an important role of the web. Many web sites provide serendipitous (unexpected but relevant) information to draw user traffic. In this paper, we study the representative scenario of mining an amusing quiz. An existing approach leverages a knowledge base to mine an unexpected property then find quiz questions on such property, based on prototype theory in cognitive science. However, existing *deterministic model* is vulnerable to noise in the knowledge base. Therefore, we instead propose to leverage probabilistic approach to build a prototype that can overcome noise. Our extensive empirical study shows that our approach not only significantly outperforms baselines by 0.06 in accuracy, and 0.11 in serendipity but also shows higher relevance than the traditional relevance-pursuing baseline using TF-IDF.

1 Introduction

Unlike the traditional purpose of the web providing relevant information or answers to user questions, conversely, recent web services ask users unexpected trivia questions to amuse them. Bing provides a set of interesting quizzes with an image of the day on the front page. Figure 1 describes an amusing quiz question generation on a long 'neck' of `giraffe`, which we use as a motivating scenario throughout this paper.

Inspired by Figure 1, we study the problem of finding a "serendipitous" property a such as 'neck' for any given entity e. Table 1 categorizes existing automatic quiz generation efforts pursuing relevance and unexpectedness, respectively. Inspired by Jeopardy!, Seyler et al. (2015) focus on relevance to the domain and a certain difficulty level. Inspired by Bing questions, Lee et al. (2016) seek unexpected entity-property pair (e, a).

In clear contrast, we complement the solution space by pursuing the intersection of finding unexpected but still relevant properties, which is often named as *serendipitous* (or Case B). As the existing *deterministic* model (DM, Lee et al. (2016)) fails to distinguish Case B and C, we propose a new *probabilistic* model (PM).

- DM: Unexpectedness of 'neck' can be found by building a deterministic prototype using the average of (normalized) property frequencies of all MAMMALS. As the frequency of 'neck' for a `giraffe` is far higher than the average, this can be found.

- PM: DM is effective when the underlying data (*i.e.*, knowledge base) to derive probabilities are not noisy, but an automatically harvested knowledge base inevitably contains noise. For example, Probase, mining textual patterns such as "<property> of <category>," may contain 'some part' as a property of a `giraffe`, which can be recognized as a desirable unexpected property by DM. We propose a probabilistic model that overcomes this problem.

[*] This work was done at Yonsei University and Microsoft Research.

[†] Zhongyuan Wang is now with Facebook Inc.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 663–673, Osaka, Japan, December 11-17 2016.

Our extensive evaluation results using real-life Flickr data and the Probase knowledge base validate that our approach not only significantly outperforms baselines by 0.06 in accuracy and 0.11 in serendipity, but also shows higher relevance than the traditional relevance-pursuing baseline using TF-IDF.

Figure 1: Bing quiz scenario used by DM (Lee et al., 2016).

	Expected	Unexpected (Lee et al., 2016)
Relevant (Seyler et al., 2015)	Case A	Case B
Irrelevant	-	Case C

Table 1: Dimensions of a mining problem.

2 Related work

This paper studies the serendipitous property mining problem of finding relevant yet unexpected properties for a given entity based on a knowledge base. Therefore, our work is closely related to knowledge acquisition. Also, mining unexpected part of knowledge can be considered as serendipitous mining or outlier detection.

2.1 Knowledge acquisition

Our serendipitous property mining system leverages automatically harvested knowledge on the web. Particularly, measuring unexpectedness requires knowing the expectation, which requires worldly knowledge. The basis of such knowledge acquisition works is taxonomies that contain categories and their entities (Carlson et al., 2010; Suchanek et al., 2007; Wu et al., 2012). Among them, Probase (Wu et al., 2012) provides a conditional probability of an entity for a category, and also that of a property given an entity, from which we probabilistically model the expectation (*i.e.*, prototype in our approach).

Besides the automatically harvested knowledge bases, other types of knowledge such as a traditional DB (Merzbacher, 2002) or a linked open data (Marie et al., 2013) are manually generated and often do not cover new entities, like new idols possibly attracting click-through. Therefore, we rely on an automatically harvested knowledge base.

2.2 Serendipitous mining

The primary metric for recommender system is prediction accuracy. However, focusing solely on this metric is reported to limit user satisfaction by always recommending predictable items, such as a new comedy for a comedy fan who can discover it without recommendation. To amuse users, serendipitous mining is studied in recommendation and search. Several approaches (Onuma et al., 2009; Nakatsuji et al., 2010) focus on finding serendipitous items such as funny zombie movie, which is both relevant and unexpected, from user-item matrices. Another direction is pursuing serendipity in search: The existing approaches propose to consider emotional expressions (Hauff and Houben, 2012; Bordino et al., 2013) or presentations such as bold font (O'Brien, 2011), to detect surprises and apply that in search results. These efforts cannot be applied to our problem as a user-item matrix or the text format is unavailable, but our knowledge-based signals are orthogonal and thus can be straightforwardly applied to improve both lines of the work.

2.3 Outlier detection

The unexpected property mining can be considered as an outlier detection problem. Outlier detection has been extensively studied with different approaches. Deviation-based approaches find observations whose removal greatly reduces the sample variance (Arning et al., 1996). Distance-based approaches define a distance measure and consider observations whose distance to others is above a threshold as outliers (Knorr and Ng, 1997; Ramaswamy et al., 2000; Fan et al., 2006). Density-based approaches

consider observations which have little or no neighbors as outliers (Jin et al., 2006). Most of these approaches are designed, trained, or tuned for a target domain using labeled data or using user specified parameters, which is not suitable for our scenario targeting diverse categories. Moreover, these solutions categorize observations into two: normal observations and outliers. Lee et al. (2016), mining quiz questions given images, also leverage the similar notion of classifying the properties into the two, which we use as a baseline and show their ineffectiveness in our experiments, due to their inability to distinguish noise and serendipitous properties.

3 Knowledge base

We leverage an automatically harvested knowledge base for human-like unexpected and relevant property mining. In particular, we leverage Probase *'is-a'* knowledge and its property data to mine probabilities, which is essentially equivalent to deriving probabilities from a huge corpus. Probase is a large knowledge base containing millions of categories and their information including *entities*, *properties*, and their *typicalities* which we describe below.

Is-A knowledge

The backbone of Probase is probabilistic knowledge of huge amount of *'is-a'* relations between entities and categories (Wu et al., 2012)[1]. For example, Probase contains a relation: "`giraffe` *is-a* MAMMAL" where `giraffe` is an entity and MAMMAL is a category. A category may contain many entities (*e.g.*, MAMMAL contains `giraffe` and `platypus`), and an entity may also belong to several categories (*e.g.*, `giraffe` belongs to both MAMMAL and ANIMAL). Such relationships are mined from a huge amount of web documents using patterns (Wu et al., 2012).

Property knowledge

Properties are words representing a certain aspect of an entity. For example, 'neck' is a property of the entities in category MAMMAL. Properties of an entity are obtained by several approaches including pattern-based extraction methods (Lee et al., 2013).

Typicality

Based on the number of pattern occurrences, we can compute the conditional probability of a certain element given a condition, which we also call *typicality*. For example, given category MAMMAL, people would usually think of typical mammals such as `dog`. In particular, Probase has diverse types of typicalities including the *entity typicality for a category*, and *property typicality for an entity*.

$P(E|C)$ is a conditional probability of entity E given category C. For example, we can obtain the probability $P(E = \text{giraffe}|C = \text{mammal})$, representing how typical entity `giraffe` is for category MAMMAL. Such probability can be obtained using all occurrences of MAMMAL, and the co-occurrences of MAMMAL and `giraffe`:

$$P(E = \text{giraffe}|C = \text{mammal}) = \frac{\text{Freq}(\text{mammal}, \text{giraffe})}{\text{Freq}(\text{mammal})} \quad (1)$$

where $\text{Freq}(x, y)$ represents the co-occurrence of x and y, and $\text{Freq}(x)$ indicates the occurrence of x in Probase. We can similarly compute other probabilities including $P(C)$, $P(E)$, $P(C|E)$. Also, we can obtain $P(A|E)$ where A is a property and E is an entity.

We can consider a typicality as the amount of people's interest in the topic since it is derived from how frequently we discuss the topic. We usually have general topics of interest for entities in a certain category: 'lifespan', 'diet', 'size' and so on for mammals. But, if the heart of a giraffe is often discussed in comparison to other mammals, it can be an unexpected topic about a giraffe.

[1]Probase knowledge base is publicly available online at https://concept.research.microsoft.com/.

4 Methods

In this section, we describe our approach to model prototype and mine serendipitous properties. We follow the framework presented in (Lee et al., 2016) that leverages the category of a given entity, and propose a method for unexpected property mining. Therefore, we assume that we have the given topic entity e, and its category c.

4.1 Modeling a prototype

Identifying an unexpected property of an entity requires comparison to what we consider expected. If we expect most mammals have a heart with the size in a specific range, the extra large heart of a `giraffe` can be unexpected. Therefore, defining the prototype of a category—which represents the human expectation for entities in the category— is the key step to find serendipitous properties. We may consider selecting a typical entity among the existing entities in the category (such as `dog` for MAMMAL). However, typical properties such as 'lifespan' can be missing with `dog` due to data sparsity. In this case, 'lifespan' of any entity can be rather considered unexpected.

DM (Lee et al., 2016) models a deterministic prototype model $\mathcal{R}_c^{\text{DM}}$ as a *hypothetical* entity using the average typicalities in the category. Note that, instead of the values of properties, their typicalities are leveraged to directly capture the human interest on the properties for entities and the category. That is, high $P(\text{height}|\text{giraffe})$ means that 'height' draws human interest so that it is discussed frequently with `giraffe` on the web. If some property is frequently mentioned with most entities in a category, it can be considered the representative property of the category. Then, we can consider some property of an entity is unexpected if the property is particularly more frequently mentioned with the entity than with other entities in the category.

Thus, using the average of typicalities allows us to compute the representativeness of the property for the category (*e.g.*, MAMMAL). Formally, $\mathcal{R}_c^{\text{DM}}$ is defined as an ordered set of the average of property typicalities in the category c as follows:

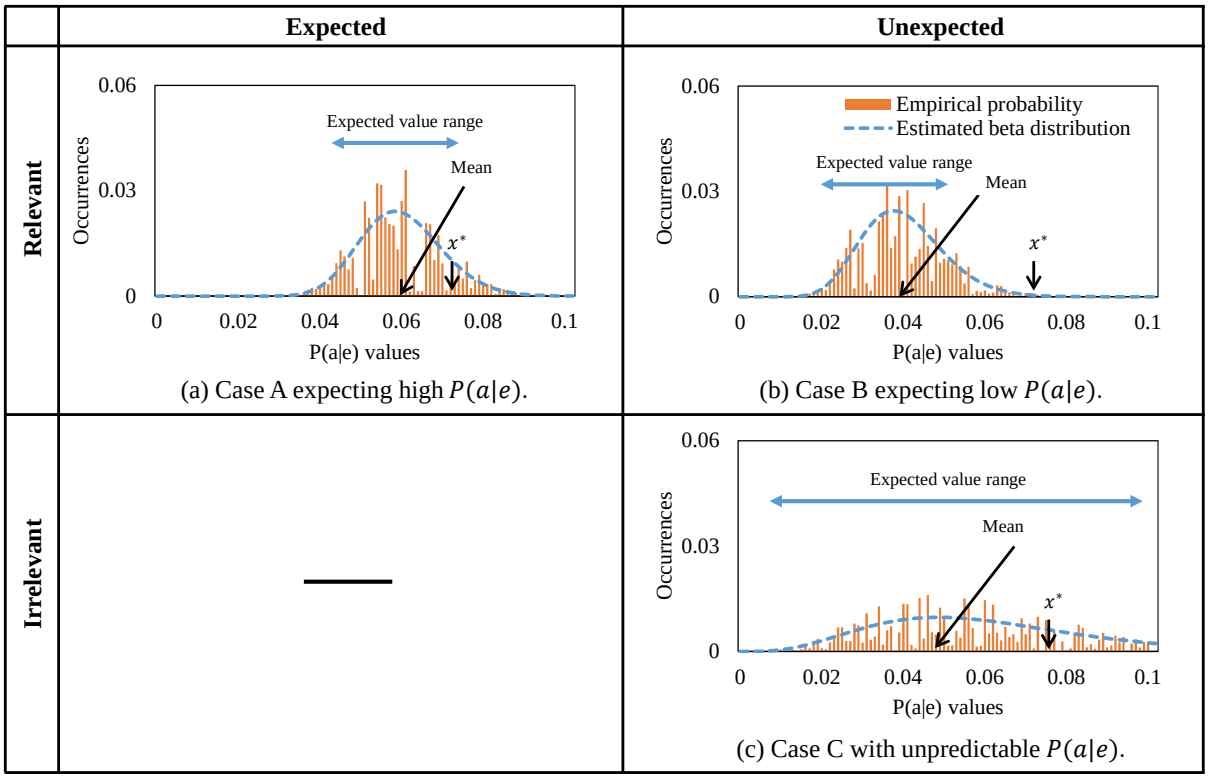
$$\mathcal{R}_c^{\text{DM}} = \{\text{avg}_{e \in c}\, P(a|e)P(e|c) \mid a \text{ is a property}\} \tag{2}$$

where $P(a|e)$ is the property typicality given entity, and $P(e|c)$ is the entity typicality given category (Section 3). This hypothetical prototype does not suffer from missing properties caused by data sparsity. However, this approach is vulnerable to noise of another cause: 'some part', wrongfully identified as a property, would be considered unexpected due to its infrequency.

Instead of this deterministic approach, we leverage a probabilistic method similar to (Eskin, 2000), which is originally designed for intrusion detection, to model the prototype. Unlike its category-agnostic approach using a Markov chain, we leverage the category information to model the prototype with beta distributions. In particular, we define a prototype of category c hypothetically as an ordered set of random variables $\mathcal{R}_c = \{X_{a,c}|a \text{ is a property}\}$ (an example distribution of $X_{a,c}$ is shown as the blue dashed line in Figure 2(b)). That is, unlike DM using averages to produce an ordered set of expected typicalities, we build probability distributions.

Given this model, we can consider that the properties of an entity in the category are realizations of the random variables. To illustrate, suppose we have $\mathcal{R}_c = \{X_{lifespan,mammal}, X_{tail,mammal}, X_{heart,mammal}\}$, and typicalities of `dog` $P(\text{lifespan}|\text{dog})$, $P(\text{tail}|\text{dog})$, and $P(\text{heart}|\text{dog})$ are 0.3, 0.6, and 0.1 respectively. Then, we consider 0.3, 0.6 and 0.1 are realizations of $X_{\text{lifespan,mammal}}$, $X_{\text{tail,mammal}}$ and $X_{\text{heart,mammal}}$ with the corresponding probabilities $P(X_{\text{lifespan,mammal}} = 0.3)$, $P(X_{\text{lifespan,mammal}} = 0.6)$, and $P(X_{\text{lifespan,mammal}} = 0.1)$. By measuring the likelihood of this event (having 0.3, 0.6, and 0.1) using these probabilities, we can measure how unexpected this entity or its properties are. Later in Section 4.2, we will argue how this model distinguishes Case B and C in Figure 2 and explain how we measure the unexpectedness.

Formally, we consider an entity of the category as an ordered set of properties represented as $\{P(a|e)\}$ (*e.g.*, $\{0.3, 0.6, 0.1\}$) each of which is drawn from the corresponding random variable in $\mathcal{R}_c$. The co-occurrences of properties and entities can be modeled to be drawn from a multinomial distribution. Then,



(a) Case A expecting high $P(a|e)$.

(b) Case B expecting low $P(a|e)$.

(c) Case C with unpredictable $P(a|e)$.

Figure 2: Distribution of $X_{a,c}$.

the typicality $P(a|e)$ for each a constitutes parameter probabilities $p_1, \ldots, p_k$ of the multinomial distribution $Mult(n|p_1, \ldots, p_k)$. We exploit a Dirichlet distribution of a single dimension, which is a beta distribution, to model $P(a|e)$ since a Dirichlet distribution is a conjugate prior of a multinomial distribution. That is, each random variable $X_{a,c}$ in prototype $\mathcal{R}_c$ is modeled as a beta distribution corresponding to a property of entities in category c: $X_{a,c} \sim Beta(x; \alpha_{a,c}, \beta_{a,c})$.

Now we have to learn the parameters specifying the beta distributions $Beta(x; \alpha_{a,c}, \beta_{a,c})$ of the prototype $\mathcal{R}_c$. In particular, we use a set $O_{a,c}$ of property typicality $P(a|e)$ for each entity e in the identified category c as samples together with its occurrence probability $P(e|c)$, that we obtain from Probase. That is, we give a larger weight to a more typical entity in the model. Let $X_{a,c}$ be a random variable modeled by a beta distribution $Beta(x; \alpha_{a,c}, \beta_{a,c})$. We find parameters $\alpha_{a,c}$ and $\beta_{a,c}$ so that $X_{a,c}$ generates the observation $O_{a,c} = \{x^e_{a,c} = P(a|e) | e \in c\}$ with their associated occurrences $P(e|c)$. Then, we have $P(x_1 \leq X_{a,c} < x_2) = \sum_{e \text{ s.t. } x_1 \leq P(a|e) < x_2} P(e|c)$. Thus, the mean μ of the beta distribution $Beta(x; \alpha_{a,c}, \beta_{a,c})$ is $\sum_{e \in c} P(a|e)P(e|c)$, and the variance σ^2 is $\sum_{e \in c} (P(a|e) - \mu)^2 P(e|c)$. The parameters $\alpha_{a,c}$ and $\beta_{a,c}$ can be fitted by the widely adopted method of moments using the mean and the variance, of exploiting the first and second moments of $Beta(x; \alpha_{a,c}, \beta_{a,c})$.

4.2 Mining unexpected properties

With the obtained probabilistic prototype $\mathcal{R}_c = \{X_{a,c} | a \text{ is a property}\}$ for category c, we discover serendipitous properties of the given entity, and show why noisy properties are not mined. We can consider that entities in the category are created by drawing each property typicality $P(a|e)$ from the beta distribution of the prototype. If a property typicality of the given entity is unlikely based on the prototype, we can consider the property is serendipitous.

We show how a property of the topic entity can be compared with that of the prototype. We divide the comparison into three cases that we have described in the introduction (Table 1): relevant and expected, unexpected but relevant (serendipitous), and noisy. Figure 2 shows examples of the component random variables $X_{a,c}$ of the model for Case A, B, and C in Table 1. Then, we consider how likely the given entity to be generated.

Case A: Property is relevant and expected

First, Figure 2(a) depicts a distribution skewed to high values. For example, 'lifespan' is a common property for the most of entities in MAMMAL. Therefore, high $P(\text{lifespan}|\text{giraffe})$ does not mean `giraffe` has an unexpected property.

Case B: Property is unexpected, but relevant (serendipitous)

In contrast, Figure 2(b) shows a distribution skewed to low $P(a|e)$. In this case, we expect a relatively low value, and thus high $P(a|e)$ of the topic entity implies a serendipitous event. For example, although the most entities in MAMMAL have a property 'neck,' it is not frequently mentioned, and hence we expect to have low $P(\text{neck}|e)$ value. Therefore, the distribution of $X_{\text{neck,mammal}}$ is skewed to low values. Then, the high value of $P(\text{neck}|\text{giraffe})$ indicates that `giraffe` has an serendipitous property 'neck.'

Case C: Property is noisy

Unlike Case A and B modeling relevant properties, the prototype may include a random variable representing a noisy property. Specifically, when a noisy property is modeled, it looks like Figure 2(c). A noisy property does not show a general tendency, and it is rather randomly distributed as often modeled by a uniform distribution. That is, since a noisy property is modeled like a uniform-esque distribution as shown in Figure 2(c), the most value of $P(a|e)$ is expected to happen, and would not be considered serendipitous.

Computing unexpected relevance

Based on these observations, we compute our measure of serendipitous properties. Formally, as exploited in (Eskin, 2000), we compute the log-likelihood of an entity-property probability $P(X_{a,c} = x^e_{a,c})$ of the given topic entity using the obtained distribution for $X_{a,c}$ of $\mathcal{R}_c$. In addition, we use a minus sign to obtain a measure indicating a more serendipitous property with a greater value.

$$\begin{aligned} I(x^e_{a,c}) &= -\log(P(X_{a,c} = x^e_{a,c})) \\ &= -\log Beta(x; \alpha_{a,c}, \beta_{a,c}) \end{aligned} \tag{3}$$

Upon this value, we also consider that a serendipitous property of a well-known or more typical entity can more easily draw user attention. Thus, we quantify the degree of being serendipitous for property a of entity e that belongs to category c as follows.

$$H(c, e, a) = I(x^e_{a,c})P(e|c) \tag{4}$$

We measure this value for each property of the given topic entity, and consider one with the highest score the most interesting, so that we present it to users.

5 Evaluation

In this section, we evaluate our systems using various measures evaluating several facets of the proposed work. Throughout the evaluation, we compare our method PM with DM (Lee et al., 2016) in the same setting of using Flickr and Yahoo! Answers.

5.1 Serendipitous property discovery

In this section, we analyze and evaluate serendipitous property mining. Table 2 first shows examples of the top serendipitous entity-property pairs for diverse categories returned by PM and DM. We can observe that DM is prone to pick long phrasal nouns or noisy properties, which are less commonly mentioned. DM considers (possum, plural) and (intel, fourth piece) unexpected because the conditional probabilities $P(\text{plural}|\text{mammal})$ and $P(\text{fourth piece}|\text{company})$ are very low while $P(\text{plural}|\text{possum})$ and $P(\text{fourth piece}|\text{intel})$ are not. On the other hand, our approach does not propose noisy properties, by modeling such noises together. Instead, our approach suggests entity-property pairs such as (marsupial, embryo), (microsoft, founder), or (china, population) that would lead to interesting information. For

Table 2: Mined unexpected entity-property pairs for diverse categories.

Category	PM	DM
mammal	(marsupial, embryo), (giraffe, heart), (chimpanzee, brain), ...	(wolf, strength), (muskrat, best part), (otter, presence), ...
company	(microsoft, founder), (facebook, founder), (amazon.com, success), ...	(intel, fourth piece), (dell, model number), (coca-cola, original color), ...
country	(china, great wall), (china, population), (india, population), ...	(china, great wall), (china, choices), (india, reserve bank), ...
metal	(copper, resistivity), (gold, price), (gold, purity), (lead, density), ...	(copper, discovery), (lead, presence), (iron, presence), (aluminum, presence), ...
drug	(cocaine, price), (marijuana, legalization), (marijuana, odor), ...	(marijuana, 80 pounds), (cocaine, freebase form), (cocaine, last use), ...

Table 3: Unexpectedness of discovered properties

Method	SS@5
PM	0.31
DM	0.25

example, unlike other mammals, a marsupial has a pouch on her stomach to carry her babies. China is known to have the largest population in the world.

We also quantitatively evaluate the accuracy, or how well the discovered serendipitous properties are aligned with human judges. The serendipitous property (Case B in Table 1) should be relevant to the given entity and category pair, but peculiar in the category. In addition, the property is the most useful in our scenario drawing users if the property leads to a novel information (less known). We build a labeled dataset by annotating the mined property with the serendipitous score on a scale of 0, 1/3, 2/3 and 3/3. In particular, 3/3 point is given if the property is relevant, peculiar, and novel; 2/3 point is given if the property is relevant, and peculiar; 1/3 point is given if the property is relevant; and 0 point is given if the property is irrelevant. For example, the population of China is extraordinarily large which is serendipitous, but this property is well known so that we give 2/3. We have built a gold standard of size 386 independently labeled by three assessors from which we observe sufficient agreement (0.64 pairwise cosine similarity), and thus use the average of the scores assigned by the assessors. We leverage *SS@5*, the average serendipitous scores of the top-5 properties for each entity.

Table 3 shows the evaluation result of serendipitous property discovery methods. We see that ours show the highest score. While our approach might find a typical property for a typical entity (*e.g.*, 'size' for `dog`), the discovered properties are mostly relevant, and often peculiar. On the other hand, DM frequently present irrelevant ones such as noisy properties (*e.g.*, 2 cups, oldddddddd guns), and thus show the lower scores.

5.2 Trivia question mining evaluation

We plug in our method into the framework of (Lee et al., 2016) that seeks trivia quizzes from image tags. We compare the trivia quiz mining results using our serendipitous property mining module and that of (Lee et al., 2016). Our evaluation procedure is based on an serendipitous document mining work (Bordino et al., 2013). Bordino et al. (2013) use both rank-agnostic accuracy of top-N results (Section 5.2.1) and rank correlation of treating its rank differently (Section 5.2.2).

5.2.1 Relevance and serendipity

We measure both relevance, and *serendipity* (*i.e.*, pleasant surprise) using the evaluation procedure introduced by (Ge et al., 2010). to evaluate a recommender system or a serendipitous web search method. It shares the same spirit as our goal that we amuse users with serendipitous properties.

Both for relevance and serendipity, we label question relevance into *relevant* (1) and *irrelevant* (0). We randomly show a query image and one of the top N result questions from any methods so that an assessor is not biased to a certain method. Note that no other information is shown to the assessor.

Then, we first measure the average relevance of the results for all images in J for each method. Formally, we compute the relevance as follows.

$$Average\ relevance@N = \frac{\sum_j \sum_k^N rel_{j,k}}{N|J|} \tag{5}$$

where $rel_{j,k}$ is the relevance (1 or 0) of the rank k question for $j \in J$.

Measurements of serendipity have been established in the context of evaluating recommendation systems (Ge et al., 2010; Shani and Gunawardana, 2011). Specifically, (Ge et al., 2010) evaluates serendipity as the ratio of unexpected but relevant results based on a *benchmark model* that generates expected recommendations, which are relevance-pursuing results. That is, from the result questions of each method, we remove the expected ones that are retrieved by the benchmark model. Then the remainder is considered unexpected. By checking the relevance of the remainder, we measure the relevance of the unexpected results.

As a benchmark model, we might consider Yahoo! Answers search results or TF-IDF-based results using the host page keywords. Unfortunately, Yahoo! Answers returns very few or no results when there are more than two keywords, which is often much less than the number of keywords an image has (*e.g.*, "giraffe zoo savanna" gives no result on Yahoo! Answers). To make the matters worse, obtaining top 100 results for subsets of keywords on Yahoo! Answers and joining the results also gives few or no intersection. Therefore, we use TF-IDF to obtain the benchmark results.

Formally, suppose that $BM@N$ is the top N questions retrieved by the benchmark model, and $RS@N$ is the top N questions retrieved by a method we want to test. Then, we calculate the unexpected recommendation set as follows.

$$UNEXP@N = RS@N - BM@N \tag{6}$$

These unexpected recommendations may or may not be relevant to the query, but we want unexpected but still relevant ones. Therefore, we measure the serendipity based on the relevance of each item in $UNEXP$. Based on relevance labels $rel_{j,k}$ of k-th result for image j, serendipity is defined as follows.

$$SRDP(RS)@N = \frac{\sum_{(j,k)\in UNEXP@N} rel_{j,k}}{N} \tag{7}$$

Table 4: Average question relevance and serendipity at N.

Method	*Rel.*@5	*Rel.*@10	*SRDP@5*	*SRDP@10*
PM	**0.6689**	**0.6607**	**0.6662**	**0.6275**
DM	0.5672	0.5820	0.5562	0.5819
TF-IDF	0.6190	0.6048	-	-

Table 4 shows the average relevance and the serendipity of each method. As a reference, we also include TF-IDF, which does not consider unexpectedness, to show the results of a typical relevance-pursuing model. Note that TF-IDF is the benchmark model and hence it does not provide unexpected questions for serendipity.

We may anticipate that the methods pursuing serendipity may have lower relevance, since such methods would avoid highly relevant results, which are often expected. Thus, DM shows lower relevance

Table 5: Average Kendall coefficient for the questions.

Methods	Q1.	Q2.	Q3.	Q4.
PM	**0.4960**	**0.3980**	**0.2486**	**0.3538**
DM	-0.0558	-0.0764	-0.2567	-0.2026

than PM because it gives lower score to those close to the average, but it may instead acquire and leverage noisy properties as unexpected. For example, DM may consider 'fourth piece' of `Intel`, which is extracted due to ambiguity and hence rare in the category, as unexpected. Therefore, it leverages such a property and in turn mines irrelevant trivia quizzes. We can also see that our approach shows higher serendipity than the baseline regardless of N. As in relevance, the baseline shows low serendipity because they highlight noisy properties as we have already seen in Section 5.1. Note that we can see PM show comparable or better relevances with TF-IDF as our approach pursuing serendipitous properties distinguishes noises.

5.2.2 Kendall's tau-b rank correlation coefficient

We evaluate the results considering their ranks using *Kendall's tau-b rank correlation coefficient* (Kendall, 1938). Kendall's coefficient ranges from -1 (perfect ranking disagreement) to +1 (perfect ranking agreement).

We build a reference ranking according to the several evaluation dimensions since "pleasant" in serendipity (pleasant surprise) can be interpreted in different ways. Therefore, we use several criteria such as 'more relevant questions rank higher' based on (Bordino et al., 2013) and evaluate individual dimensions. Specifically, we generate a task with a photograph and two randomly chosen result trivia quizzes out of those returned by all tested methods. A result trivia quiz that is more suitable for each of the following criteria is labeled as 'better.'

- Q1. The trivia quiz is relevant to the image.
  (*i.e.*, 'pleasant' means 'relevant')
- Q2. If someone is interested in the image, they would also be interested in the trivia quiz.
  (*i.e.*, 'pleasant' means 'interesting given interest to the image')
- Q3. Even if you are not interested in the image, the trivia quiz is interesting to you personally.
  (*i.e.*, 'pleasant' means 'interesting regardlessly to the image')
- Q4. You learn something new about the image from the trivia quiz.
  (*i.e.*, 'pleasant' means 'novel')

Note that Q2 and Q3 try to evaluate both the image or topic-centric view of interestingness (Q2), and the intrinsic interestingness of the trivia quiz (Q3) as in (Bordino et al., 2013). For example, if a method shows high performance on Q3 but not on Q2, its trivia quiz tends to deviate much from the image/topic. However, the method is anyway presenting interesting trivia quiz.

A human assessor has to label each of criteria for a given task. As used in (Bordino et al., 2013), the reference ranking for each criterion is built by a simple voting-based approach, by ranking items with the greater number of 'better' votes higher. Since this evaluation may heavily depend on human assessors, we validate the gold standard by measuring the agreement. Thirty tasks are randomly sampled and evaluated by four human assessors. We measure Fleiss' kappa (Fleiss, 1971) to obtain $\kappa = 0.57$, which shows moderate agreement (Landis and Koch, 1977). In addition, considering only confident answers by removing those tasks with any 'not sure' vote, we obtain higher agreement $\kappa = 0.71$, which shows substantial agreement.

The evaluation result using Kendall coefficient is shown in Table 5. We can see that our method outperforms the baseline. In particular, our approach retains a high positive correlation with the reference rankings based on user perception. On the other hand, the baseline method shows low or negative correlation with all reference rankings.

6 Conclusions and future work

This paper studies the serendipitous property mining problem of finding relevant yet unexpected properties for a given entity. Although the serendipitous information mining is important for industry to increase website traffic, it has not been studied actively on general knowledge due to lack of knowledge or having noisy knowledge. Such noisy knowledge is challenging as noisy properties can be evaluated as unexpected. We empirically show that probabilistic modeling of a prototype for each category alleviates the noise problem, while the existing approach is prone to pick up the noisy properties that may significantly detract the user experience of the applications like trivia questions. Our evaluation results suggest that our approach shows not only higher serendipity than the baselines, but also higher relevance than a traditional baseline using TF-IDF optimized for relevance.

We expect many research topics can be stemmed from our work. One possibility of modeling a prototype would be neural embedding. Comparing its performance with our probabilistic model will be a good research direction. Meanwhile, our approach is limited to properties, and the proposed framework evaluates one property of an entity at each time. But taking relations into account, or considering more than one properties/relations may give even more interesting questions and facts. Also, we expect a similar approach can be exploited to mine outstanding issues from social network data which have a considerable amount of noise.

7 Acknowledgments

This work was supported by the Yonsei University New Faculty Research Seed Funding Grant of 2015, the Yonsei University Research Fund (Post Doc. Researcher Supporting Program) of 2015 (project no.: 2015-12-0211), and Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.B0101-16-0307, Basic Software Research in Human-level Lifelong Machine Learning (Machine Learning Center)).

References

Andreas Arning, Rakesh Agrawal, and Prabhakar Raghavan. 1996. A linear method for deviation detection in large databases. In *Proc. 2nd International Conference on Computational Linguistics (KDD 1996)*, pages 164–169. ACM.

Ilaria Bordino, Yelena Mejova, and Mounia Lalmas. 2013. Penguins in sweaters, or serendipitous entity search on user-generated content. In *Proc. 20th ACM International Conference on Information and Knowledge Management (CIKM 2013)*. ACM.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proc. 24th Conference on Artificial Intelligence (AAAI 2010)*. AAAI Press.

Eleazar Eskin. 2000. Anomaly detection over noisy data using learned probability distributions. In *Proc. 17th International Conference on Machine Learning (ICML 2000)*, pages 255–262. Morgan Kaufmann.

Hongqin Fan, Osmar R Zaïane, Andrew Foss, and Junfeng Wu. 2006. A nonparametric outlier detection for effectively discovering top-n outliers from engineering data. In *Proc. Pacific-Asia conference on knowledge discovery and data mining (PAKDD 2006)*, pages 557–566. Springer.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. 2010. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In *Proc. 4th Conference on Recommender Systems (RECSYS 2010)*, pages 257–260. ACM.

Claudia Hauff and Geert-Jan Houben. 2012. Serendipitous browsing: Stumbling through wikipedia. In *Proc. 34th European Conference on IR Research (ECIR 2012)*.

Wen Jin, Anthony KH Tung, Jiawei Han, and Wei Wang. 2006. Ranking outliers using symmetric neighborhood relationship. In *Proc. Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2006)*, pages 577–593. Springer.

M. G. Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93.

Edwin M. Knorr and Raymond T. Ng. 1997. A unified approach for mining outliers. In *Proc. 2rd International Conference on Computational Linguistics (KDD 1997)*, pages 219–222. ACM.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174.

Taesung Lee, Zhongyuan Wang, Haixun Wang, and Seung-won Hwang. 2013. Attribute extraction and scoring: A probabilistic approach. In *Proc. 29th International Conference on Data Engineering (ICDE 2013)*, pages 194–205. IEEE.

Taesung Lee, Seung-won Hwang, and Zhongyuan Wang. 2016. Trivia quiz mining using probabilistic knowledge. In *Proc. 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2016)*.

Nicolas Marie, Fabien Gandon, Myriam Ribière, and Florentin Rodio. 2013. Discovery hub: On-the-fly linked data exploratory search. In *Proc. 9th International Conference on Semantic Systems (I-SEMANTICS 2013)*, pages 17–24.

Matthew Merzbacher. 2002. Automatic generation of trivia questions. In *Proc. 13th International Symposium on Foundations of Intelligent Systems (ISMIS 2002)*, pages 123–130.

Makoto Nakatsuji, Yasuhiro Fujiwara, Akimichi Tanaka, Toshio Uchiyama, Ko Fujimura, and Toru Ishida. 2010. Classical music for rock fans?: Novel recommendations for expanding user interests. In *Proc. 19th ACM International Conference on Information and Knowledge Management (CIKM 2010)*, pages 949–958. ACM.

Heather L. O'Brien. 2011. Exploring user engagement in online news interactions. In *Proc. Annual Meeting on Association for Information Science and Technology (ASIST 2011)*, pages 1–10. Wiley Subscription Services, Inc., A Wiley Company.

Kensuke Onuma, Hanghang Tong, and Christos Faloutsos. 2009. Tangent: A novel, 'surprise me', recommendation algorithm. In *Proc. 15th International Conference on Computational Linguistics (KDD 2009)*, pages 657–666. ACM.

Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient algorithms for mining outliers from large data sets. In *Proc. SIGMOD International Conference on Management of Data (SIGMOD 2000)*, pages 427–438. ACM.

D Seyler, M Yahya, and K Berberich. 2015. Generating quiz questions from knowledge base. In *Proc. 24st International World Wide Web Conference (WWW 2015)*. ACM.

Guy Shani and Asela Gunawardana. 2011. Evaluating recommendation systems. In *Recommender Systems Handbook*, pages 257–297. Springer US.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *Proc. 16th International World Wide Web Conference (WWW 2007)*. ACM.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q. Zhu. 2012. Probase: a probabilistic taxonomy for text understanding. In *Proc. SIGMOD International Conference on Management of Data (SIGMOD 2012)*. ACM.

Identifying Cross-Cultural Differences in Word Usage

Aparna Garimella and **Rada Mihalcea**
University of Michigan
{gaparna,mihalcea}@umich.edu

James Pennebaker
University of Texas at Austin
pennebaker@utexas.edu

Abstract

Personal writings have inspired researchers in the fields of linguistics and psychology to study the relationship between language and culture to better understand the psychology of people across different cultures. In this paper, we explore this relation by developing cross-cultural word models to identify words with cultural bias – i.e., words that are used in significantly different ways by speakers from different cultures. Focusing specifically on two cultures: United States and Australia, we identify a set of words with significant usage differences, and further investigate these words through feature analysis and topic modeling, shedding light on the attributes of language that contribute to these differences.

1 Introduction

According to Shweder et al. (1998), "to be a member of a group is to think and act in a certain way, in the light of particular goals, values, pictures of the world; and to think and act so is to belong to a group."

Culture can be defined as any characteristic of a group of people, which can affect and shape their beliefs and behaviors (e.g., nationality, region, state, gender, or religion). It reflects itself in people's everyday thoughts, beliefs, ideas, and actions, and understanding what people say or write in their daily lives can help us understand and differentiate cultures. In this work, we use very large corpora of personal writings in the form of blogs from multiple cultures[1] to understand cultural differences in word usage.

We find inspiration in a line of research in psychology that poses that people from different cultural backgrounds and/or speaking different languages perceive the world around them differently, which is reflected in their perception of time and space (Kern, 2003; Boroditsky, 2001), body shapes (Furnham and Alibhai, 1983), or surrounding objects (Boroditsky et al., 2003). As an example, consider the study described by Boroditsky et al. (2003), which showed how the perception of objects in different languages can be affected by their gender differences. For instance, one of the words used in their study is the word "bridge," which is masculine in Spanish and feminine in German: when asked about the descriptive properties of a bridge, Spanish speakers described bridges as being *big*, *dangerous*, *long*, *strong*, *sturdy*, and *towering*, while German speakers said they are *beautiful*, *elegant*, *fragile*, *peaceful*, *pretty*, and *slender*.

While this previous research has the benefit of careful in-lab studies that explore differences in world view for one dimension (e.g., time, space) or word (e.g., bridge, sun) at a time, it also has limitations in terms of the number of experiments that can be run when subjects are being brought to the lab for every new question being asked. We aim to address this shortcoming by using the power of large-scale computational linguistics, which allows us to identify cultural differences in word usage in a data-driven bottom-up fashion.

We hypothesize that we can use computational models to identify differences in word usage between cultures, regarded as an approximation of their differences in world view. Rather than starting with predetermined hypotheses (e.g., that Spanish and German speakers would have a different way of talking about bridges), we can use computational linguistics to run experiments on hundreds of words, and

[1] Throughout this paper, we use the term culture to represent the nationality (country) of a group of people.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 674–683, Osaka, Japan, December 11-17 2016.

consequently identify those words where usage differences exist between two cultures. We explore this hypothesis by seeking answers to two main research questions.

First, given a word W, are there significant differences in how this word is being used by two selected cultures? We build cross-cultural word models in which we use classifiers based on several classes of linguistic features and attempt to differentiate between usages of the given word W in different cultures. By applying them to a large number of words, these models are used to identify those words for which there exist significant usage differences between the two cultures of interest.

Second, if such significant differences in the usage of a word are identified, can we use feature analysis to understand the nature of these differences? We perform several analyses: (1) Feature ablation that highlights the linguistic features contributing to these differences; (2) Topic modeling applied to the words with significant differences, used to identify the dominant topic for each culture and to measure the correlation between the topic distributions in the two cultures; and (3) One-versus-all cross-cultural classification models, where we attempt to isolate the idiosyncrasies in word usage for one culture at a time.

2 Data

We base our work on personal writings collected from blogs, and specifically target word usage differences between Australia and United States. These two countries are selected for two main reasons: (1) they both use English as a native language, and therefore we can avoid the noise that would otherwise be introduced by machine translation; and (2) they have a significant number of blogs contributed in recent years, which we can use to collect a large number of occurrences for a large set of words.

We obtain a large corpus of blog posts by crawling the blogger profiles and posts from Google Blogger. For each profile, we consider up to a maximum of 20 blogs, and for each blog, we consider up to 500 posts. Table 1 gives statistics of the data collected in this process. We process the blog posts by removing the HTML tags and tagging them with part-of-speech labels (Toutanova et al., 2003).

Country	Profiles	Blogs	Posts
Australia	469	1129	320316
United States	374	1267	471257

Table 1: Blog statistics for the two target cultures.

Next, we create our pool of candidate target words by identifying the top $1,500$ content words based on their frequency in the blog posts, additionally placing a constraint that they cover all open-class parts-of-speech: of the $1,500$ words, 500 are nouns, 500 verbs, 250 adjectives, and 250 adverbs. These numbers are chosen based on the number of examples that exist for the target words; e.g., most (> 490) of the 500 selected nouns have more than 300 examples; etc. We consider all possible inflections for these words, for instance for the verb *write* we also consider the forms *writes*, *wrote*, *written*, *writing*. The possible inflections for the target words are added manually, to ensure correct handling of grammatical exceptions.

To obtain usage examples for the two cultures for these words, we extract paragraphs from the blog posts that include the selected words with the given part-of-speech. Of these paragraphs, we discard those that contain less than ten words. We also truncate the long paragraphs so they include a maximum of 100 words to the left and right of the target word, disregarding sentence boundaries. The contexts of the target words are then parsed to get the dependency tags related to the target word (Klein and Manning, 2003). We also explicitly balance the data across time. Noting there could be cases where the number of blog posts published in a specific year is higher compared to that in other years due to certain events (e.g., an Olympiad, or a major weather related event), we draw samples for our dataset from several different time periods. Specifically, for each culture, we consider an equal number of instances from four different years (2011-2014). Table 2 shows the per-word average number of data instances obtained in this way for each part-of-speech for each culture for the years 2011-2014.

Note that we do not attempt to balance the data across topics (domains), as we regard potentially different topic distributions as a reflection of the culture (e.g., Australia may be naturally more interested

in water sports than United States is). We do instead explicitly balance our data over time, as described above, to avoid temporal topic peaks related to certain events.

Country	Noun	Verb	Adj	Adv
Australia	22461	18396	19206	19377
United States	15199	12347	12513	12952

Table 2: Average number of instances for the 1,500 target words for the years 2011-2014.

3 Finding Words with Cultural Bias

We start by addressing the first research question: given a word W, are there significant differences in how this word is being used by two different cultures? We formulate a classification task where the goal is to identify, for the given target word W, the culture of the writer of a certain occurrence of that word. If the accuracy of such a classifier exceeds that of a random baseline, this can be taken as an indication of word usage differences between the two cultures. We run classification experiments on each of the $1,500$ words described in the previous section, and consequently aim to identify those words with significant usage differences between Australia and United States.

3.1 Features

We implement and extract four types of features:

Local features. These features consist of the target word itself, its part-of-speech, three words and their parts-of-speech to the left and right of the target concept, nouns and verbs before and after the target concept. These features are used to capture the immediately surrounding language (e.g., descriptors, verbs) used by the writers while describing their views about the target word.

Contextual features. These features are determined from the global context, and represent the most frequently occurring open-class words in the contexts of the word W in each culture. We allow for at most ten such features for each culture, and impose a threshold of a minimum of five occurrences for a word to be selected as a contextual feature. Contextual features express the overall intention of the blogger while writing about the target word.

Socio-linguistic features. These features include (1) fractions of words that fall under each of the 70 Linguistic Inquiry and Word Count (LIWC) categories (Pennebaker et al., 2001); the 2001 version of LIWC includes about 2,200 words and word stems grouped into broad categories relevant to psychological processes (e.g., emotion, cognition); (2) fractions of words belonging to each of the five fine-grained polarity classes in OpinionFinder (Wilson et al., 2005), namely strongly negative, weakly negative, neutral, weakly positive, and strongly positive; (3) fractions of words belonging to each of five Morality classes (Ignatow and Mihalcea, 2012), i.e., authority, care, fairness, ingroup, sanctity; and (4) fractions of words belonging to each of the six Wordnet Affect classes (Strapparava et al., 2004), namely anger, disgust, fear, joy, sadness, and surprise. These features provide social and psychological insights into the perceptions bloggers have about the words they use.

Syntactic features. These features consist of parser dependencies (De Marneffe et al., 2006) obtained from the Stanford dependency parser (Klein and Manning, 2003) for the context of the target word. Among these, we select different dependencies for each part-of-speech: (1) nouns: root word of context (`root`), governor[2] if noun is nominal subject (`nsubj`), governor verb if noun is direct object (`dobj`), adjectival modifier (`amod`); (2) verbs: root, nominal subject (`nubj`), direct object (`dobj`), adjectival complement (`acomp`), adverb modifier (`advmod`); (3) adjectives: root, noun being modified (`amod`), verb being complemented (`acomp`), adverb modifier (`advmod`); (4) adverbs: root, adverb modifier (`advmod`). These features capture syntactic dependencies of the target word that are not always obtained using just its context.

[2]We follow the convention provided in `http://nlp.stanford.edu/software/dependencies_manual.pdf`

3.2 Cross-cultural Word Models

The features described above are integrated into an AdaBoost classifier.[3] This classifier was selected based on its performance on a development dataset, when compared to other learning algorithms. We compare the performance of the classifier with a random choice baseline, which is always 50%, given the equal distribution of data between the two cultures. This allows us to identify the words for which we can automatically identify the culture of the writers of those words, which is taken as an indication of word usage differences between the two cultures.

Throughout the paper, all the results reported are obtained using ten-fold cross-validation on the word data. When creating the folds, we explicitly ensure that posts authored by the same blogger are not shared between the folds, which in turn ensures no overlap between bloggers in training and test sets. This is important as repeating bloggers in both the train and the test splits could potentially overfit the model to the writing styles of individual bloggers rather than learning the underlying culture-based differences between the bloggers.

To summarize the cross-validation process: First, for each of the 1,500 target words, we collect an equal number of instances containing the given target word or its inflections from Australia and United States, from each of the selected years (2011-2014). We then divide the posts belonging to Australia and United States each into ten approximately equal groups, such that no two groups have bloggers in common. We finally combine the corresponding groups to form a total of ten bi-cultural groups that are approximately of equal size, which form our cross-validation splits. [4]

To compute the statistical significance of the results obtained, we perform a two-sample t-test over the correctness of predictions of the two systems namely, Adaboost and random chance classifiers. Disambiguation results that are significantly better ($p < 0.05$) than the random chance baseline of 50% are marked with $*$.

On average, the classifier leads to an accuracy of 58.36%*, which represents an absolute significant improvement of 8.36% over the baseline (a random chance of 50%). Table 3 shows the average classification results for each part-of-speech, as well as the number of words for which the AdaBoost classifier leads to an accuracy significantly larger than the baseline. These results suggest that there are indeed differences in the ways in which writers from Australia and United States use the target words with respect to all the parts-of-speech.

Part-of-speech	Average accuracy	Words with significant difference
Nouns	57.51*	393
Verbs	58.01*	395
Adjectives	59.25*	207
Adverbs	61.77*	215
Overall	58.36*	1210

Table 3: Average ten-fold cross-validation accuracies and number of words with an accuracy significantly higher than the baseline, for each part-of-speech, for United States vs. Australia.

4 Where is the Difference?

We now turn to our second research question: Once significant differences in the usage of a word are identified, can we use feature analysis to understand the nature of these differences?

4.1 Feature Ablation

We first study the role of the different linguistic features when separating between word usages in Australia and United States through ablation studies. For each of the feature sets specified in Section 3,

[3] We use the open source machine learning framework Weka (Hall et al., 2009) for all our experiments. We use the default base classifier for AdaBoost, i.e., a DecisionStump.

[4] This condition of equal data in each group is only approximate, as there will generally not be an exact division of bloggers with equal data. The average size of a cross-validation train split for a target word is 6246.57, while that for a test split is 819.88.

we retrain our concept models using just that feature set type, which helps us locate the features that contribute the most to the observed cultural differences.

The left side of Table 4 shows the ablation results averaged over the 1,500 target words for the four sets of features. We observe that the contextual and socio-linguistic features perform consistently well across all the parts-of-speech, and they alone can obtain an accuracy close to the all-feature performance.

Part-of-speech	Loc	Con	Soc	Syn	All	LIWC	OF	ML	WNA
Nouns	49.06	57.22*	56.52*	46.54	57.28*	56.62*	56.21*	54.17*	52.94
Verbs	47.97	57.65*	57.04*	47.71	57.90*	56.90*	56.28*	53.73*	53.25*
Adjectives	48.67	58.63*	57.90*	47.52	59.01*	58.03*	57.31*	55.42*	54.30*
Adverbs	50.72	61.09*	59.80*	46.85	60.81*	60.27*	59.80*	57.00*	56.57*
All	48.91	58.25*	57.47*	47.14	58.36*	57.55*	57.01*	54.70*	53.87*

Table 4: Feature ablation averaged over 1,500 target words. Loc: Local features, Con: Contextual features, Soc: Socio-linguistic features, Syn: Syntactic features, All: All feature types, LIWC: Linguistic Inquiry and Word Count, OF: OpinionFinder, ML: Morality Lexicon, WNA: WordNet Affect.

We also perform a feature ablation experiment to explore the role played by the various socio-linguistic features. The right side of Table 4 shows the classification accuracy obtained by using one socio-linguistic lexicon at a time: LIWC, OpinionFinder, Morality, and WordNet Affect. Among all these resources, LIWC and OpinionFinder appear to contribute the most to the classifier; while the morality lexicon and WordNet Affect also lead to an accuracy higher than the baseline, their performance is clearly smaller.

4.2 Topic Modeling

We next focus our analysis on the top 100 words (25 words for each part-of-speech) that have the most significant improvements over the random chance baseline, considered to be words with cultural bias in their use. The average accuracy of the classifier obtained on this set of words is 65.45%; the accuracy for each part-of-speech is shown in the second column of Table 7.

We model the different usages of the words in our set of 100 words by using topic modeling. Specifically, we use Latent Dirichlet Allocation (LDA)[5] (Blei et al., 2003) to find a set of topics for each word, and consequently identify the topics specific to either Australia or United States.

As typically done in topic modeling, we preprocess the data by removing a standard list of stop words, words with very high frequency ($> 0.25\% \times$datasize), and words that occur only once. To determine the number of topics that best describe the corpus for each of the 100 words, we use the average corpus likelihood over ten runs (Heinrich, 2005). Specifically, we choose that number of topics ($>= 2, <= 10$) for which the corpus likelihood is maximum.

For each data instance, we say that a topic dominates the other topics if its probability is higher than that of the remaining topics. For a given word, we then identify the *dominating topic* for each culture as the topic that dominates the other topics in a majority of data instances. We use this definition of dominating topic in all the analyses done in this section.

Quantitative Evaluation. To get an overall measure of how different cultures use the words that were found to have significant differences, we compute the Spearman's rank correlation between the topic distributions for the two cultures. For each topic, we get the number of data instances in which it dominates the other topics, in both cultures (Australia and United States). Subsequently, we measure the overall Spearman correlation coefficient between the dominating topic distributions for all 100 words. In other words, the distribution of topics is compared across cultures for each word. The Spearman coefficient is calculated as 0.63, which reflects a medium correlation between the usages of the words by the two cultures.

Qualitative Evaluation. For a qualitative evaluation, Table 5 shows five sample words for each part-of-speech, along with the identified number of topics and the dominating topic for Australia and United

[5]LDA has been shown to be effective in text-related tasks, such as document classification (Wei and Croft, 2006).

States. We associate labels to the hidden topics manually after looking at the corresponding top words falling under each of them.

As seen in this table, the number of topics that best describe each word can vary widely between two topics for words such as *start* and *economic*, up to ten topics (which is the maximum allowed number of topics) for words such as *color* or *support*. The dominating topics illustrate the biases that exist in each culture for these words; for instance, the word *teach* is dominantly used to describe academic teaching in Australia, whereas in United States it is majorly used to talk about general life teaching. Several additional examples of differences are shown in Table 5.

4.3 One-versus-all Classification

For additional insight into word usage differences between United States and Australia, we expand our study to develop word models to separate word usages in Australia (or United States) from a mix of ten different cultures. In other words, we conduct a one-versus-all classification using the same process as described in Section 3, but using Australia (United States) against a mix of other cultures to examine any features specific to Australia (United States).

In order to do this, we collect data from nine additional English speaking countries, as shown in the left side of Table 6. As before, the data for each country is balanced over time, and it includes an equal number of instances for four different years (2011-2014). The right side of Table 6 shows the average number of instances per word collected for each part-of-speech.

In this classification, for a given target word, one half of the data is collected from Australia (United States), and the other half is collected from the remaining countries, drawing 10% from each country. We run these classifiers for all the 100 words previously identified as having cultural bias in their use.

The average classifier accuracy for the Australia-versus-all classification, using ten-fold cross validation, is 64.23%, as shown in the third column of Table 7. We repeat the same one-versus-all classification for United States, with an average accuracy of 54.89%; the results of this experiment are listed in the last column of Table 7.

Overall, the performance improvement over the baseline is higher for Australia versus other countries (14.23% absolute improvement) than it is for United States versus others (4.89% absolute improvement). From this, we can infer that that the performance improvement over the baseline for the Australia versus United States task can be majorly attributed to the different word usages in Australia from the remaining countries. In other words, United States is more aligned with the "typical" (as measured over ten different countries) usage of these words than Australia is.

5 Related Work

Most of the previous cross-cultural research work has been undertaken in fields such as sociology, psychology, or antropology (De Secondat and others, 1748; Shweder, 1991; Cohen et al., 1996; Street, 1993). For instance, Shweder (1991) examined the cross-cultural similarities and differences in the perceptions, emotions, and ideologies of people belonging to different cultures, while Pennebaker et al. (1996) measured the emotional expressiveness among the northerners and southerners in their own countries, to test Montesquieu's geography hypothesis (De Secondat and others, 1748). More recently, the findings of Boroditsky et al. (2003) indicate that people's perception of certain inanimate objects (such as bridge, key, violin, etc.) is influenced by the grammatical genders assigned to these objects in their native languages.

To our knowledge, there is only limited work in computational linguistics that explored cross-cultural differences through language analysis. Our work is most closely related to that by Paul and Girju (2009), in which they identify cultural differences in people's experiences in various countries from the perspective of tourists and locals. Specifically, they analyzed forums and blogs written by tourists and locals about their experiences in three countries, namely Singapore, India, and United Kingdom, using an extension of LDA. One of their findings is that while topic modeling on tourist forums offered an unsupervised aggregation of factual data specific to each country that would be important to travelers (such as destination's climate, law, and language), topic modeling on blogs authored by locals showed cultural differences between the three countries with respect to several topics (e.g., fashion, pets, religion, health).

Word	Accuracy					NT	Dominating topic	
	All	Loc	Con	Soc	Syn		Australia	United States
Nouns								
store	67.14*	53.11*	67.15*	64.98*	50.79	10	ONLINE (blog, card, gifts, online, sale)	GROCERY (grocery, shopping, things)
support	66.05*	44.56	52.45	62.54*	51.29	10	EDUCATION (school, students, education)	LAW (public, police, law, social)
color	65.56*	50.74	67.35*	65.56*	50.27	10	BACKGROUND (pink, design, background)	PICTURE (paint, kit, picture, photo)
version	65.42*	47.75	65.47*	63.57*	47.28	4	RENDERING (story, thought, school)	ALBUM (song, music, album, classic)
phone	64.32*	53.93*	62.51*	58.19*	52.49	3	COMMUNICATION (calls, email, message)	FRIEND (night, friend, talking)
Verbs								
go	63.93*	47.29	64.15*	65.18*	42.77	2	TIME (time, day, night, love, today)	LIFE (life, world, work, god, children)
start	63.66*	46.80	63.46*	62.09*	54.08*	2	DAYBREAK (starting, day, love, morning)	SCHOOL (starting, school, softball)
know	63.51*	50.18	64.20*	62.40*	47.12	2	GOOD TIMES (love, good, things, life)	CHILDREN (children, school, year, book)
sing	63.54*	51.07	63.96*	58.19*	49.80	4	CHRISTMAS (christmas, happy, kids)	ROCK SINGER (band, guitar, rock, singer)
teach	62.66*	52.72	61.69*	59.88*	51.18	10	ACADEMICS (teachers, education, curriculum)	LIFE (time, young, life, thought, work, friends)
Adjectives								
various	65.64*	54.0*	64.70*	63.18*	48.40	10	POLITICS (political, party, war, revolution)	DIVERSITY (states, people, companies)
own	64.76*	56.89*	65.19*	62.71*	51.79	2	POWER (life, war, political, power)	LIFE (love, music, work, school)
economic	61.88*	44.82	42.11	59.16*	33.07	2	FINANCE (economy, financial, market, tax)	POLITICS (political, social, war, power)
old	66.45*	42.95	67.09*	64.74*	39.89	2	AGE (older, children, family, age)	PAST (back, school, days, love)
human	64.37*	52.78	60.92*	59.09*	48.14	9	RIGHTS (rights, law, freedom, civil)	LIFE (life, time, love, real)
Adverbs								
quite	73.56*	55.22*	70.98*	73.49*	51.50	2	EXTENT (time, back, good, love, thought)	EXTENT (time, back, good, love, thought)
else	67.35*	52.45	68.21*	64.66*	45.19	2	(make, find, world, invented, book, christ)	(time, life, night, love, work, home, god)
actually	65.62*	50.67	66.60*	65.51*	45.75	9	POLITICS (law war, people, government)	FAMILY (kids, fun, home, couple)
usually	68.37*	57.71*	70.16*	67.03*	44.35	2	SPORTS (softball, cricket, play)	ROUTINE (things, work, love, home)
certainly	64.64*	53.76*	63.25*	62.87*	43.03	2	SPORTS (game, series, softball, wrestling)	TIME (time, work, things, make)

Table 5: Five sample words per part-of-speech with significant usage difference in the Australian and American cultures. All: All the features, Loc: Local features, Con: Contextual features, Soc: Sociolinguistic features, Syn: Syntactic features, NT: Number of topics.

Country	Profiles	Blogs	Posts	Word instances			
				Noun	Verb	Adj	Adv
Barbados	440	830	32785	581	466	490	476
Canada	461	1097	397479	15015	12020	12965	12512
Ireland	473	978	231240	8936	7161	7919	7809
Jamaica	451	770	41495	1632	1318	1353	1297
New Zealand	450	1112	226900	8713	7313	7883	8284
Nigeria	464	908	223772	13719	9710	9796	7631
Pakistan	458	1404	135473	2861	2130	2243	1847
Singapore	406	803	208972	5623	5430	5447	6639
United Kingdom	473	934	282740	10887	9432	10021	11066

Table 6: Statistics for blog data collected for additional English speaking countries.

Part-of-speech	United States vs. Australia	Australia vs. all	United States vs. all
Nouns	65.54*	63.45*	57.07*
Verbs	64.20*	63.97*	53.87*
Adjectives	65.13*	64.36*	54.48*
Adverbs	66.92*	65.13*	54.13*
Overall	65.45*	64.23*	54.89*

Table 7: Ten-fold cross-validation accuracies averaged over the top 100 target words for United States vs. Australia; Australia vs. a mix of ten other countries; United States vs. a mix of ten other countries.

Yin et al. (2011) used topic models along with geographical configurations in Flickr to analyze cultural differences in the tags used for specific target image categories, such as cars, activities, festivals, or national parks. They performed a comparison over the topics across different geographical locations for each of the categories using three strategies of modeling geographical topics (location-driven, text-driven, and latent geographical topic analysis (LGTA) that combines location and text information), and found that the LGTA model worked well not only for finding regions of interest, but also for making effective comparisons of different topics across locations.

Ramirez et al. (2008) performed two studies to examine the expression of depression among English and Spanish speakers on the Internet. The first study used LIWC categories to process depression and breast cancer posts to identify linguistic style of depressed language. Significantly more first person singular pronouns were used in both English and Spanish posts, supporting the hypothesis that depressed people tend to focus on themselves and detach from others. The second study focused on discovering the actual topics of conversation in the posts using Meaning Extraction Method (Chung and Pennebaker, 2008). It was found that relational concerns (e.g., family, friends) were more likely expressed by depressed people writing in Spanish, while English people mostly mentioned medical concerns.

6 Conclusions

In this paper, we explored the problem of identifying word usage differences between people belonging to different cultures. Specifically, we studied differences between Australia and United States based on the words they used frequently in their online writings. Using a large number of examples for a set of $1,500$ words, covering different parts-of-speech, we showed that we can build classifiers based on linguistic features that can separate between the word usages from the two cultures with an accuracy higher than chance. We take this as an indication that there are significant differences in how these words are used in the two cultures, reflecting cultural bias in word use.

To better understand these differences, we performed several analyses. First, using feature ablation, we identified the contextual and socio-linguistic features as the ones playing the most important role in these word use differences. Second, focusing on the words with the most significant differences, we used topic modeling to find the main topics for each of these words, which allowed us to identify the dominant topic for a word in each culture, pointing to several interesting word use differences as outlined in Table

5. We also measured the correlation between the topic distributions for the top 100 words between the two cultures, and found a medium correlation of 0.63. Finally, we also performed a one-versus-all classification for these 100 words, where word use instances drawn from one of Australia or United States were compared against a mix of instances drawn from ten other cultures, which suggested that United States is a more "typical" culture when it comes to word use (with significantly smaller differences in these one-versus-all classifications than Australia).

In future work, we plan to extend this work to understand differences in word usages between a larger number of cultures, as well as for a larger variety of words (e.g., function words).

The cross-cultural word datasets used in the experiments reported in this paper are available at http://lit.eecs.umich.edu.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation (#1344257), the John Templeton Foundation (#48503), and the Michigan Institute for Data Science. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, the John Templeton Foundation, or the Michigan Institute for Data Science.

References

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Lera Boroditsky, Lauren A Schmidt, and Webb Phillips. 2003. Sex, syntax, and semantics. *Language in mind: Advances in the study of language and thought*, pages 61–79.

Lera Boroditsky. 2001. Does language shape thought?: Mandarin and english speakers' conceptions of time. *Cognitive psychology*, 43(1):1–22.

Cindy K Chung and James W Pennebaker. 2008. Revealing dimensions of thinking in open-ended self-descriptions: An automated meaning extraction method for natural language. *Journal of research in personality*, 42(1):96–132.

Dov Cohen, Richard E Nisbett, Brian F Bowdle, and Norbert Schwarz. 1996. Insult, aggression, and the southern culture of honor: An" experimental ethnography.". *Journal of personality and social psychology*, 70(5):945.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Charles-Louis De Secondat et al. 1748. *The Spirit of Laws*. Hayes Barton Press.

Adrian Furnham and Naznin Alibhai. 1983. Cross-cultural differences in the perception of female body shapes. *Psychological medicine*, 13(04):829–837.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Gregor Heinrich. 2005. Parameter estimation for text analysis. Technical report, Technical report.

Gabe Ignatow and Rada Mihalcea. 2012. Injustice frames in social media. Denver, CO.

Stephen Kern. 2003. *The culture of time and space, 1880-1918: with a new preface*. Harvard University Press.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Michael Paul and Roxana Girju. 2009. Cross-cultural analysis of blogs and forums with mixed-collection topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1408–1417. Association for Computational Linguistics.

James W Pennebaker, Bernard Rimé, and Virginia E Blankenship. 1996. Stereotypes of emotional expressiveness of northerners and southerners: a cross-cultural test of montesquieu's hypotheses. *Journal of personality and social psychology*, 70(2):372.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71:2001.

Nairan Ramirez-Esparza, Cindy K Chung, Ewa Kacewicz, and James W Pennebaker. 2008. The psychology of word use in depression forums in English and in Spanish: Texting two text analytic approaches. In *ICWSM*.

Richard A Shweder, Jacqueline J Goodnow, Giyoo Hatano, Robert A LeVine, Hazel R Markus, and Peggy J Miller. 1998. The cultural psychology of development: One mind, many mentalities. *Handbook of child psychology*.

Richard A Shweder. 1991. *Thinking through cultures: Expeditions in cultural psychology*. Harvard University Press.

Carlo Strapparava, Alessandro Valitutti, et al. 2004. Wordnet affect: an affective extension of wordnet. In *LREC*, volume 4, pages 1083–1086.

Brian Street. 1993. Culture is a verb: Anthropological aspects of language and cultural process. *Language and culture*, pages 23–43.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Xing Wei and W Bruce Croft. 2006. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185. ACM.

Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Opinionfinder: A system for subjectivity analysis. In *Proceedings of hlt/emnlp on interactive demonstrations*, pages 34–35. Association for Computational Linguistics.

Zhijun Yin, Liangliang Cao, Jiawei Han, Chengxiang Zhai, and Thomas Huang. 2011. Geographical topic discovery and comparison. In *Proceedings of the 20th international conference on World wide web*, pages 247–256. ACM.

Reading-Time Annotations
for *Balanced Corpus of Contemporary Written Japanese*

Masayuki Asahara
National Institute of Japanese Language and Linguistics, National Institute for the Humanities, Japan
masayu-a.ninjal.ac.jp

Hajime Ono
Faculty of Liberal Arts, Tsuda College.

Edson T. Miyamoto
Graduate School of Humanities and Social Science, University of Tsukuba

Abstract

The *Dundee Eyetracking Corpus* contains eyetracking data collected while native speakers of English and French read newspaper editorial articles. Similar resources for other languages are still rare, especially for languages in which words are not overtly delimited with spaces. This is a report on a project to build an eyetracking corpus for Japanese. Measurements were collected while 24 native speakers of Japanese read excerpts from the *Balanced Corpus of Contemporary Written Japanese* Texts were presented with or without segmentation (i.e. with or without space at the boundaries between *bunsetsu* segmentations) and with two types of methodologies (eyetracking and self-paced reading presentation). Readers' background information including vocabulary-size estimation and Japanese reading-span score were also collected. As an example of the possible uses for the corpus, we also report analyses investigating the phenomena of anti-locality.

1 Introduction

Corpora of naturally-produced texts such as newspapers and magazines marked with detailed morphological, syntactic and semantic tags, are often used in human language-production research. In contrast, texts created by psycholinguists exclusively for research purposes, are commonly used in language-comprehension research.

We introduce a reusable linguistic resource that can help bridge this gap by bringing together techniques from corpus linguistics and experimental psycholinguistics. More concretely, we have collected reading times for a subset of texts from the *Balanced Corpus of Contemporary Written Japanese* (BCCWJ) (Maekawa et al., 2014), which already contains syntactic and semantic types of annotations. The goal is to produce a resource comparable to the *Dundee Eyetracking Corpus* (Kennedy and Pynte, 2005), which contains reading times for English and French newspaper editorials from 10 native speakers for each language, recorded using eyetracking equipment. The English version of the *Dundee Eyetracking Corpus* is composed of 20 editorial articles with 51,501 words.

The *Dundee Eyetracking Corpus* does not target a specific set of linguistic phenomena; instead, it provides naturally occurring texts for the testing of diverse hypotheses. For example, Demberg and Keller (2008) used the corpus to test Gibson's Dependency Locality Theory (DLT), (Gibson, 2008), and Hale's surprisal theory (Hale, 2001). The corpus also allows for replications to be conducted, as in Roland et al. (2012), who concluded that previous analyses (Demberg and Keller, 2007) had been distorted by the presense of a few outlier data points.

Our goal is to produce a similar resource that can serve as a shared, available foundation for research in Japanese text processing. Once completed, the corpus will allow us to address two issues that are specific to Japanese. The first issue is related to two types of reading-time measurements commonly used, namely, eyetracking and self-paced reading. Although eyetracking provides detailed recordings of eye movements, it requires specialized equipment. Self-paced reading requires only a regular computer to collect button presses, which have been shown to be an effective alternative that correlates well with

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 684–694, Osaka, Japan, December 11-17 2016.

eyetracking data in English (Just et al., 1982). However, to date, no similar correlation analyses have been conducted for Japanese. A second issue related to Japanese is that its texts do not contain spaces to mark boundaries between words (or other linguistic units such as *bunsetsu*, in other words, a content word plus functional morphology), and the question arises as to the best way to show segments in self-paced reading presentations.

Here, we present specifications and basic statistics for the *BCCWJ Eyetracking Corpus*, which makes available reading times for BCCWJ texts that have been previously annotated with syntactic and semantic tags. This should allow for detailed analyses of human text processing having a diverse range of purposes (e.g., readability measurements, evaluations of stochastic language models, engineering applications).

The rest of the paper is organized as follows. Section 2 provides basic information about the reading-time annotations, participants (§2.1), articles (§2.2), apparatus/procedure (§2.3), and data format and basic statistics (§2.4). Section 3 presents an analysis investigating a phenomenon of anti-locality (Konieczny, 2000). These are followed by the conclusion and future directions.

2 Method

2.1 Participants

Twenty-four native speakers of Japanese who were 18 years of age or older at the time, participated in the experiment for financial compensation. The experiments were conducted from September to December 2015. Profile data collected included age (in five-year brackets), gender, educational background, eyesight (all participants had uncorrected vision or vision corrected with soft contact lenses or prescription glasses), geographical linguistic background (i.e. the prefecture within Japan where they lived until the age of 15), and parents' place of birth (See Table 1 for a summary).

Vocabulary size was measured using a Japanese language vocabulary evaluation test (Amano and Kondo, 1998). Participants indicated the words they knew from a list of 50 words and scores were calculated taking word-familiarity estimates into consideration.

As a measure of working-memory capacity, the Japanese version of the reading-span test was conducted (Osaka and Osaka, 1994). Each participant read sentences aloud, each of which contained an underlined content word. After each set of sentences, the participant recalled the underlined words. If all words were successfully recalled, the set size was increased by one sentence (sets of two to five sentences were used). The final score was the largest set for which all words were correctly recalled, with a half point added if half of the words were recalled in the last trial (See Table 2 for the scores in the vocabulary and working memory tests).

Table 1: Profile data for the participants

Age range (years)	Females	Males	Gender not given	Total
-20	1	1		2
21-25		2		2
26-30	2			2
31-35	3			3
36-40	9		1	10
41-45	3			3
46-50	1			1
51-		1		1
total	19	4	1	24

Table 2: Results for reading span test and vocabulary-size test

Vocab. size	Reading span test score 1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	Total
36,000 -		1	1						2
38,000 -		4		1					5
40,000 -	1	1							2
42,000 -		1							1
44,000 -						1			1
46,000 -									0
48,000 -			1						1
50,000 -			4	1	1		1		7
52,000 -			1					1	2
54,000 -	1								1
56,000 -									0
58,000 -			1						1
60,000 -		1							1
Total	2	8	8	2	1	1	1	1	24

2.2 Texts

Reading times were collected for a subset of the core data of the *Balanced Corpus of Contemporary Written Japanese* (BCCWJ) (Maekawa et al., 2014), consisting of newspaper articles (PN: published

newspaper) samples. Articles were chosen if they were annotated with information such as syntactic dependencies, predicative clausal structures, co-references, focus of negation, and similar details, following the list of articles that were given annotation priority in the BCCWJ.[1]

The 20 newspaper articles chosen were divided into four sets of data containing five articles each: sample sets A, B, C, and D. Table 3 shows the numbers of words, sentences, and screens (i.e. pages) for each set of data. Each article was presented starting on a new screen.

Articles were shown segmented or unsegmented, that is, with or without a half-width space to mark the boundary between segments. Segments conformed to the definition for *bunsetsu* units (a content word followed by functional morphology, e.g., a noun with a case marker) in the BCCWJ as prescribed by the National Institute for Japanese Language and Linguistics. Each participant was assigned to one of the eight groups of three participants each, one group for each of the eight experimental conditions with varying combinations of measurement methods and boundary marking for different data sets presented in different orders (see Table 4). The next section provides explanations for the two measurement methods (eyetracking and self-paced reading). Order of the tasks was fixed with eye movements collected in the first session, and keyboard presses recorded during a self-paced reading presentation in the second session. Each participant saw each text once, with task and segmentation for the texts counter-balanced across participants.

Table 3: Data set sizes

Data set	Segments	Sentences	Screens
A	470	66	19
B	455	67	21
C	355	44	16
D	363	41	15

Table 4: Experimental Design

Group	Eye tracking		Self-paced reading	
1	A unseg	B seg	C unseg	D seg
2	A seg	B unseg	C seg	D unseg
3	C unseg	D seg	A unseg	B seg
4	C seg	D unseg	A seg	B unseg
5	B unseg	A seg	D unseg	C seg
6	B seg	A unseg	D seg	C unseg
7	D unseg	C seg	B unseg	A seg
8	D seg	C unseg	B seg	A unseg

'seg' stands for with spaces, and 'unseg' stands for without spaces.

2.3 Apparatus and Procedure

Eye movements were recorded using a tower-mounted EyeLink 1000 (SR Research Ltd). View was binocular but data were collected from each participant's right eye using 1000-Hz resolution. Participants looked at the display by way of a half-mirror as their heads were fixed with their chins resting on a chin rest. Unlike self-paced reading, in eyetracking all segments are shown simultaneously thus allowing more natural reading as the participant can freely return and reread earlier parts of the text on the same screen (but, participants were not allowed to return to previous screens). Stimulus texts were shown in a fixed full-width font (MS Mincho 24 point), displayed horizontally as is customary with computer displays for Japanese, with five lines per screen on a 21.5-inch display.[2] In the segmented condition, a half-width space was used to indicate the boundary between segments. In order to improve vertical tracking accuracy, three empty lines intervened between lines of text. A line break was inserted at the end of sentence or when the maximum 53 full-width characters per line was reached. Moreover, line breaks were inserted at the same points in the segmented and unsegmented conditions to guarantee that the same number of non-space characters were shown in both conditions.

The same procedure was adopted for the self-paced reading presentation, except that the chin rest was not used and participants could move their heads freely while looking directly at the display. Doug Rohde's Linger program, Version 2.94[3] was used to record keyboard-press latencies while sentences were shown using a non-cumulative self-paced moving-window presentation, which had the best correlation with eyetracking data when different styles of presentation were compared for English (Just et al., 1982). Sentence segments were initially shown masked with dashes. Participants pressed the space key of the

[1] `https://github.com/masayu-a/BCCWJ-ANNOTATION-ORDER`

[2] EIZO FlexScan EV2116W (resolution 1920×1080), set 50 cm from the chin rest.

[3] `http://tedlab.mit.edu/~dr/Linger/`

keyboard to reveal each subsequent segment of the sentence while all other segments reverted to dashes. Participants were not allowed to return to reread earlier segments.

Figure 1 shows two types of segmentations in the self-paced reading setting. In order to illustrate the difference between full-width underscore and half-width underscore, their heights are slightly altered in the figure. In the original Linger software presentation, these are shown at the same height.

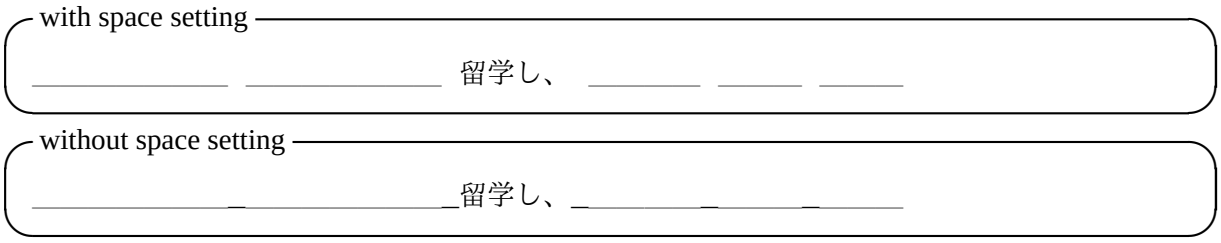


Figure 1: Types of segmentations in the self-paced reading experiment

2.4 Analysis

2.4.1 Reading-Times Tabulation

In the self-paced reading session, each segment was displayed separately, and participants could not return to reread earlier parts of the text. Therefore, the latencies for the button presses are straightforward measures of the time spent on each segment.

For the eyetracking data, five types of measurements are included, namely, First Fixation Time (FFT), First-Pass Time (FPT), Regression Path Time (RPT), Second-Pass Time (SPT), and Total Time (TOTAL), which will be explained using Figure 2.

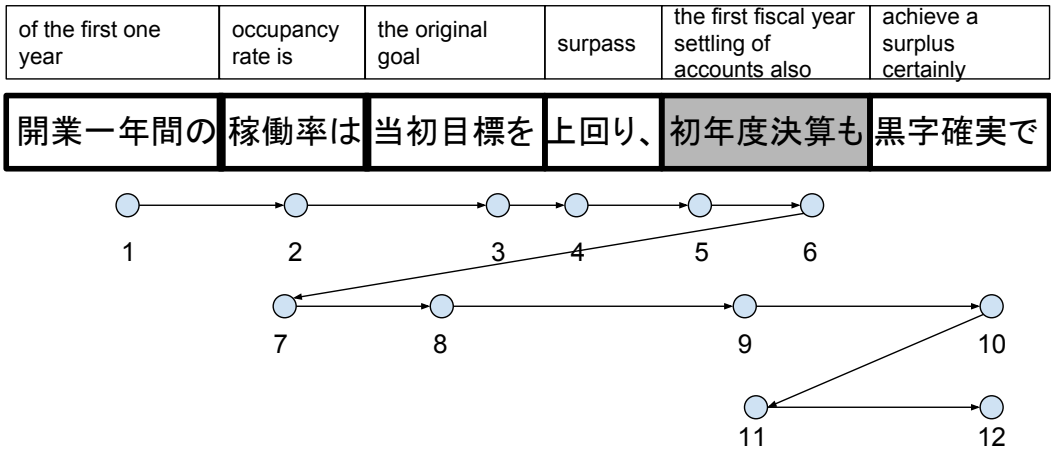


Figure 2: Example of fixations

First Fixation Time (FFT) is the fixation duration measured when the gaze first enters the area of interest. In Figure 2, the FFT for 'the first fiscal year settling of accounts also' (hereafter 'the area of interest') is the duration of fixation 5.

First-Pass Time (FPT) is the total duration of fixation from the moment the gaze first stops within the area of interest until it leaves the focus area by moving to the right or left of this area. In the figure, the FPT is the sum of the durations of fixations 5 and 6.

Regression Path Time (RPT) is the total span of from the moment the gaze enters the area of interest until it crosses the right boundary of this area for the first time. In the figure, the RPT is the sum of the

Table 5: Data format

Column name	Type	Description
surface	factor	Word surface form
time	int	Reading-time
measure	factor	Reading time types
sample	factor	Sample name
article	factor	Article information
metadata_orig	factor	Document structure tag
metadata	factor	Metadata
sessionN	int	Session order
articleN	int	Article display order
screenN	int	Screen display order
lineN	int	Line display order
segmentN	int	segment display order
sample_screen	factor	Screen identifier
length	int	Number of characters
space	factor	segment boundary with space or not
setorder	int	Segmentation-type order
subj	factor	Participant ID
rspan	num	Reading-span test score
voc	num	Vocabulary-test score
dependent	int	Number of dependents

durations for fixations 5, 6, 7, 8 and 9. The RPT can includes fixations to the left of the left boundary (e.g., 7 and 8) and durations of fixations when the gaze returns to the area of interest (e.g., 9).

Second-Pass Time (SPT) is the total span of time the gaze spend in the area of interest excluding the FPT. In the figure, the SPT is the sum of the durations of fixations for 9 and 11.

Total Time (TOTAL) is the total duration that the gaze spends within the area of interest. In other words, it is the sum of the SPT and the FPT. In the figure, TOTAL is the sum of the durations of fixations 5, 6, 9 and 11.

Only fixation times have been tabulated thus far. In the future, saccade information will also be made available.

2.4.2 Data Format and Basic Statistics

Data will be made available in tab-separated valuses (TSV) format for each of the reading-time measurements described in the previous section, along with information about the original articles and profiles of the participant. Table 5 summarizes the data format.

Word surface form (Surface: factor) refers to the text strings shown to the participants. These are organized according to the segment standards of the National Institute for Japanese Language and Linguistics, with full-width blank spaces removed.

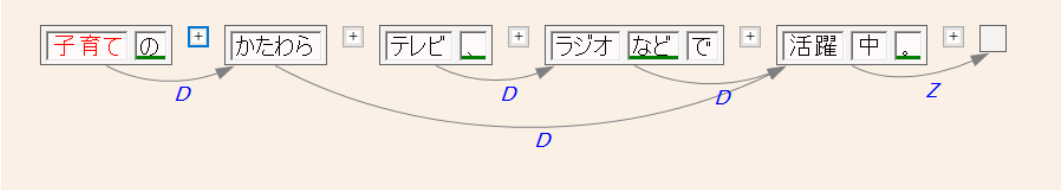
Reading time (time: int) is the `time` measurement expressed in milliseconds. For self-paced reading, this is the button-press latency for a single segment. For eyetracking, numbers are provided for each of the five measurements discussed in the previous section: First Fixation Time (FFT), First-Pass Time (FPT), Second-Pass Time (SPT), Regression Path Time (RPT) and Total Time (TOTAL). The *reading time types* (`measure`: factor) are defined as {'Self-Paced', 'EyeTrack: FFT', 'EyeTrack: FPT', 'EyeTrack: SPT', 'EyeTrack: RPT', 'EyeTrack: Total'}.

There are four types of information provided for the newspaper articles: `sample`, `article`, `metadata_orig` and `metadata`. The *sample name* (`sample`: factor) is derived from the data sets prepared for each session 'A', 'B', 'C', 'D'; each sample consists of five newspaper articles. *Article in-*

formation (`article`: factor) is a unique identifier for the individual articles, which is connected with an underscore to the BCCWJ annotation priority rankings, the BCCWJ internal sample IDs, and the article numbers. *Document structure tag* (`metadata_orig`: factor) is a BCCWJ internal document structure tag, which is connected with the tag information in the BCCWJ XML ancestor axis using a slash. *Metadata* (`metadata`: factor) is generated through the extraction of the properties of the article from the previously mentioned `metadata_orig`. It is set to one of {'authorsData', 'caption', 'listItem', 'profile', 'titleBlock', or 'undefined'}, and indicates manual revisions of mistakes or omissions in the BCCWJ internal document structure tag.

There are five types of information related to presentation order. *Session order* (`session`: int) indicates the session number (1 or 2). *Article display order* (`articleN`: int) indicates the article display sequence (1–5) within each session. *Screen display order* (`screenN`: int) indicates the screen's display sequence number within each article. *Line display order* (`lineN`: int) indicates the line number within each screen (1–5). Segment display order (`segmentN`: int) indicates the segment sequence number within each line.

Screen identifier (`sample_screen`: factor) is a unique identifier for the screens displayed to the participants. *Number of characters* (`length`: int) is the number of characters in the segment. *Segmented or unsegmented* (`space`: factor) indicates whether there is a half-width space between segment units ('1'), or not ('0'). *Segmentation-type order* (`setorder`: factor) is set to '0-1' if the participant saw unsegmented texts followed by segmented texts, and it is set to '1-0' otherwise. *Number of dependents* (`dependent`: int) is the number of segment units that are syntactically dependent on the current segment. Segment dependency relationships were annotated manually. Figure 3 shows an example of a dependency annotation on segments. Note, the dependency arcs are written from dependent to head following convention in Japanese annotations.



She raises her twins and is also active as a broadcaster of TV and radio programs.

Figure 3: Example of Dependency Annotation

There are three types of information assigned for each participant. *Experiment participant ID* (`subj`: factor) is a unique identifier for each participant, and is associated with two pieces of information. The first is the *reading span test score* (`rspan`: num), ranging from 1.5 to 5.0 in gradations of 0.5. The second is the *Vocabulary test score* (`voc`: num), which is the original result divided by 1,000 (37.1-61.8).

Table 6 shows means, standard deviations (SDs), and quartiles for each measurement. For eyetracking, the numbers shown exclude reading times of zero milliseconds (i.e. instances where segments were not fixated).

After each article, a simple yes-no question verified readers' comprehension. Overall accurary was 88.5% (ranging from 70%-100%). Accuracy was higher in eye-tracking (99.2%; 238/240) than in self-paced reading (77.9%; 187/240: $p < 0.001$). One possible factor favoring eye-tracking is that participants could reread texts freely. Another factor is that self-paced reading data was always collected in the second session, therefore participants may have been more tired.

3 Example Analysis

3.1 Anti-locality

As an example of the possible uses for the corpus, we conducted analyses investigating *anti-locality* phenomena, in which a head is read faster if it is preceded by more dependents as first reported for

Table 6: Mean reading times (in milliseconds)

	Mean	SD	Min.	1st Qu.	Median	3rd Qu.	Max.
Self-paced	699	506	62	415	550	798	9454
Eye tracking (excl. 0)							
First Fixation Time	235	142	12	162	219	292	1700
First-Pass Time	475	497	14	205	321	548	7340
Second-Pass Time	330	253	20	173	258	418	2553
Regression Path Time	698	1013	19	235	391	745	21577
Total Time	597	589	18	247	416	721	8397

German (Konieczny, 2000; Konieczny and Döring, 2003) and later replicated for Japanese (Uchida et al., 2014). Such speedu gains in head-final constructions are not easily explained by working-memory models, which either predict that attaching a large number of dependents should be costly (Gibson, 2008) or predict that the cost of an upcoming head should not be affected by the number of dependents preceding it (Nakatani and Gibson, 2010). Although the phenomenon is compatible with surprisal (Hale, 2001; Levy and Gibson, 2013), previous results were limited to reports that a ditransitive verb was read more quickly when preceded by two dependents (an accusative-marked argument and a dative-marked argument) than when preceded by just one dependent (the accusative argument). Therefore, these results do not necessarily support *anti-locality*, instead they may be related to ditransitive verbs being more natural when the dative noun phrase is expressed overtly. We report corpus analyses that show that *anti-locality* holds more generally.

3.2 Modeling Results

Linear mixed models were constructed for reading times to the main texts of the articles (i.e. excluding reading times that had the metadata field labelled as authorsData, caption, listItem, profile, or titleBlock). The first and last segments on a line may be exceptional as they may be affected by large eye movements going from the end of the line to the beginning of the following line, or backtracking to reread content at the end of the previous line. Therefore, factors were included in the analyses encoding whether the segment is the first (`is_first`), second to last (`is_second_last`) or last (`is_last`) on a line. We also excluded zero-millsecond data points from the eyetracking data. Because the models with maximal or close-to-maximal random structure did not converge, we performed forward model selections for each time setting and report the model with the smallest AIC that converged. After model-based trimming was used to eliminate points beyond three standard deviations, the model was rebuilt (Baayen, 2008). Tables 7, 8, 9, 10, 11, and 12 show the results of the smallest-AIC models for 'Self-Paced Reading (Self)', and eyetracking data for 'First Fixation Time (FFT)', 'First-Pass Time (FPT)', 'Second-Pass Time (SPT)', 'Regression Path Time (RPT)', and 'Total Times (Total)', respectively.

In the tables, the baseline (i.e. the intercept) encodes the False value for binary factors. Therefore, a factor name followed by '1' (e.g., `is_first1`) indicates what happens to the model prediction when the factor changes from FALSE to TRUE. For example, in Table 7, the intercept is the baseline (634.08 ms) which excludes reading times to the first, penultimate, and last segments (i.e. `is_first=FALSE`, `is_second_last=FALSE`; `is_last=FALSE`). Therefore, the row starting with `is_first1` (i.e. it is the first segment of the line) indicates that the first segment was 69.73 ms slower than the segments included in the baseline.

Table 13 shows the summary of the results from the linear mixed model. In this table, if the absolute *t*-value of the effect is larger than 1.96, we regard the factor as statistically significant and put the sign of the estimate. Otherwise, we put 0, indicating nonsignificant factors.

Texts presented that were segmented with a blank space had shorter first pass times, second pass times, total-reading times than unsegmented texts (factor `space`). These results are interesting because texts are usually unsegmented in Japanese writing, therefore the result is the opposite of what would be expected based on participants' reading habits. The result is also not compatible with previous results,

Table 7: Parameters of the linear mixed model for the self-paced reading data

	Estimate	Std. Err.	*t* value
Intercept	634.08	31.05	20.42
space1	3.04	4.03	0.75
sessionN	-46.50	27.10	-1.72
length	159.35	2.22	71.74
dependent	-27.00	2.26	-11.96
is_first1	69.73	6.56	10.63
is_last1	-37.93	6.61	5.73
is_second_last1	-8.22	5.88	-1.40
articleN	-40.84	14.45	-2.83
screenN	-42.56	2.74	-15.49
lineN	-19.36	2.14	-9.06
segmentN	-11.76	3.56	-3.31
space1:sessionN	2.34	54.04	0.04

316 data points (1.79%) were excluded in the 3-SD trimming. We choose the converged model with smaller AIC as follows: `lmer (time ~ space * sessionN + length + dependent + is_first + is_last + is_second_last + articleN + screenN + lineN + segmentN + (1 + articleN + segmentN | subj) + (1 + articleN | article)`

Table 8: Parameters of the linear mixed model for first fixation time

	Estimate	Std. Err.	*t* value
Intercept	227.00	7.39	30.86
space1	-3.01	1.70	-1.77
sessionN	-11.81	6.70	-1.76
length	-1.38	0.88	-1.57
dependent	-4.81	0.93	-5.15
is_first1	13.18	2.60	5.07
is_last1	-3.11	2.71	-1.15
is_second_last1	3.35	2.43	1.38
articleN	-0.57	1.81	-0.32
screenN	-1.15	1.09	-1.06
lineN	-5.24	0.87	-6.00
segmentN	3.436	1.67	2.06
space1:sessionN	22.32	13.29	1.68

170 data points (1.28%) were excluded in the 3-SD trimming. We choose the converged model with smaller AIC. `lmer (time ~ space * sessionN + length + dependent + is_first + is_last + is_second_last + articleN + screenN + lineN + segmentN + (1 + articleN + segmentN | subj) + (1 + articleN | article)`

Table 9: Parameters of the linear mixed model for first-pass time

	Estimate	Std. Err.	*t* value
Intercept	421.54	24.32	17.33
space1	-18.04	4.97	-3.73
sessionN	-24.51	17.31	-1.42
length	171.74	2.70	63.55
dependent	-32.16	2.71	-11.85
is_first1	83.53	7.62	10.96
is_last1	9.06	7.95	1.14
is_second_last1	23.14	7.12	3.25
articleN	-1.81	8.91	-0.20
screenN	-19.38	3.21	-6.15
lineN	-19.86	2.55	-7.81
segmentN	-4.26	5.58	-0.76
space1:sessionN	21.47	34.43	0.62

234 data points (1.76%) were excluded in the 3-SD trimming. We choose the converged model with smaller AIC as follows: `lmer (time ~ space * sessionN + length + dependent + is_first + is_last + is_second_last + articleN + screenN + lineN + segmentN + (1 + articleN + segmentN | subj) + (1 + articleN + segmentN | article)`

Table 10: Parameters of the linear mixed model for second-pass time

	Estimate	Std. Err.	*t* value
Intercept	317.14	12.07	26.28
space1	-26.73	5.86	-4.56
sessionN	-17.87	10.19	-1.75
length	16.72	3.02	5.54
dependent	-13.52	3.30	-4.09
is_first1	-21.05	8.34	-2.52
is_last1	-15.79	9.88	-1.60
is_second_last1	38.30	8.97	4.27
articleN	1.17	4.57	0.26
screenN	-9.29	3.54	-2.63
lineN	-12.30	2.97	-4.13
segmentN	-18.02	3.77	-4.78
space1:sessionN	28.49	19.88	1.43

77 data points (1.61%) were excluded in the 3-SD trimming. We choose the converged model with smaller AIC as follows: `lmer (time ~ space * sessionN + length + dependent + is_first + is_last + is_second_last + articleN + screenN + lineN + segmentN + (1 + articleN + segmentN | subj) + (1 + is_last + is_second_last + articleN | article)`

in which segmentation did not have a reliable effect in texts mixing *kanji* and *kana* characters (Sainio et al., 2007), but that may have been due to lack of statistical power or perhaps because the segmented texts were too short for participants to accommodate to this type of presentation and use segmentation information effectively.

An unsurprising finding is that longer segments (i.e. segments having more characters) took longer to read (factor `length`) except for the first fixation time. The result suggests that longer segments do not require longer first fixation, but nevertheless affect later measures as they may require further fixations.

Compared to the intermediate segments (i.e. second segment to the antepenultimate segments) on each line, longer reading times were observed for the first segment (`is_first=TRUE`; in self-paced reading, first fixation, first pass, and total reading time), for the penultimate segment

Table 11: Parameters of the linear mixed model for regression path time

	Estimate	Std. Err.	*t* value
Intercept	560.07	28.63	19.57
space1	-10.29	9.89	-1.04
sessionN	-57.59	23.14	-2.49
length	173.18	5.21	33.22
dependent	-10.86	5.49	-1.98
is_first1	8.37	15.13	0.55
is_last1	205.88	16.04	12.84
is_second_last1	7.38	14.27	0.52
articleN	2.35	13.61	0.17
screenN	-13.75	6.13	-2.24
lineN	21.90	5.09	4.31
segmentN	-34.21	11.22	-3.05
space1:sessionN	59.45	45.55	1.31

219 data points (1.65%) were excluded in the 3-SD trimming. We choose the converged model with smaller AIC as follows:

```
lmer (time ~ space * sessionN + length + dependent
+ is_first + is_last + is_second_last + articleN +
screenN + lineN + segmentN + (1 + articleN + segmentN
| subj) + (1 + articleN | article)
```

Table 12: Parameters of the linear mixed model for total time

	Estimate	Std. Err.	*t* value
Intercept	549.09	29.74	18.46
space1	-36.16	6.35	-5.70
sessionN	-24.76	20.96	-1.18
length	198.62	3.41	58.21
dependent	-41.04	3.45	-11.90
is_first1	-79.43	9.66	8.23
is_last1	-11.08	10.08	-1.10
is_second_last1	43.58	9.04	4.82
articleN	-7.90	12.87	-0.61
screenN	-31.50	4.13	-7.62
lineN	-23.60	3.24	-7.29
segmentN	-28.21	6.06	-4.65
space1:sessionN	6.68	42.65	0.16

232 data points (1.75%) were excluded in the 3-SD trimming. We choose the converged model with smaller AIC as follows:

```
lmer (time ~ space * sessionN + length + dependent
+ is_first + is_last + is_second_last + articleN +
screenN + lineN + segmentN + (1 + articleN + segmentN
| subj) + (1 + articleN | article)
```

Table 13: Summary of the results from the linear mixed models

	Self	FFT	FPT	SPT	RPT	Total
space=True	0	0	-	-	0	-
length	+	0	+	+	+	+
is_first=True	+	+	+	-	0	+
is_last=True	+	0	0	0	+	0
is_second_last=True	0	0	+	+	0	+
articleN	-	0	0	0	0	0
screenN	-	0	-	-	-	-
lineN	-	-	-	-	-	-
segmentN	-	+	0	-	-	-
dependent	-	-	-	-	-	-

(second_last_bunsetsu=TRUE; in first pass time, second pass time, and total time), and for the last segment (last_bunsetsu=TRUE; in self-paced reading and regression path time).

Within a session, reading times from self-paced reading became faster with each article (articleN); however, the effect was not reliable in any of the eye-tracking measures. Within an article, reading times got faster with each screen (screenN) in all measures except for first fixation time. In the vertical ordering within a screen, all reading times got faster with each line (lineN). In the horizontal ordering within a line, reading times except for first fixation time and first pass time became faster with each segment (segmentN). These speed gains are expected as readers gain speed as they process more information.

Apart from the effects described above related to the physical aspects of the presentation of the texts, we also observed a reliable *anti-locality* effect as words were read faster when more dependents preceded them (factor dependent). This generalizes previous findings (Konieczny, 2000; Konieczny and Döring, 2003; Uchida et al., 2014) and confirms that dependent phrases provide information that facilitates the processing of an upcoming head.

4 Conclusion

We created a data set with the reading times of 24 native speakers of Japanese. Preliminary analyses illustrate the uses of this type of data. First, although spaces are not commonly used to segment Japanese text, readers were nevertheless faster to read segmented texts. Second, we reported an analysis on *anti-locality* effects, which confirmed previous reports and generalized them to more natural texts.

The reading time data, excluding the original texts, will be licenced through Creative Commons Attribution-Noncommercial 4.0 (CC BY-NC 4.0: `https://creativecommons.org/licenses/by-nc/4.0/`). Apart from the data files described in Section 2.4.2, the original eye-tracking data can be obtained as EyeLink Data Viewer files, by contacting the first author. The original texts can be obtained by purchasing the BCCWJ DVD edition `http://pj.ninjal.ac.jp/corpus_center/bccwj/en/subscription.html`.

Future planned developments are as follows. First, we will extend the corpus with more participants and data. This initial data set was restricted to newspaper articles, and we are currently investigating the possibility of assigning reading times to other texts, such as books and magazines.

Other types of annotations will be added. Apart from information on the number of dependents already available in the current data, we are considering including other types of information such as dependency length, scope of coordinate structure. Other types of information that may be added in the future include morphological information such as word class, vocabulary classification table number, predicate clause structure (*ga*-case:subj, *o*-case:dobj, *ni*-case:iobj), co-reference information, clause boundary information, and information structure.[4]

Finally, we intend to examine possible applications for information processing. Participants were required to write a summary for each text they read. Contrast analysis of the reading times and the summaries may allow us to augment automatic summarization systems tailored to individual readers.

Acknowledgements

This research received assistance from the JSPS Grants-in-Aid for Scientific Research Fund (B) 25284083 (Reading Time Annotation for the Language Corpus and its Uses). We would like to thank Douglas Roland, Manabu Arai and four anonymous reviewers for their comments on the earlier draft.

[4] Parts of such annotation are already available at `https://bccwj-data.ninjal.ac.jp/mdl/`

References

S. Amano and T. Kondo. 1998. Estimation of mental lexicon size with word familiarity database. In *Proceedings of International Conference on Spoken Language Processing*, volume 5, pages 2119–2122.

R. H. Baayen. 2008. *Analyzing Linguistic Data: A practical Introduction to Statistics using R*. Cambridge University Press.

V. Demberg and F. Keller. 2007. Eye-tracking evidence for integration cost effects in corpus data. In *Proceedings of the 29th Meeting of the Cognitive Science Society (CogSci-07)*, pages 947–952.

V. Demberg and F. Keller. 2008. Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, 109(2):193–210.

E. Gibson. 2008. Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 68:1–76.

J. Hale. 2001. A probabilistic earley parser as a psycholinguistic model. In *Proceedings of the Second Conference of the North American Chapter of the Association for Computational Linguistics*, volume 2, pages 159–166.

M. A. Just, P. A. Carpenter, and J. D. Woolley. 1982. Paradigms and processes in reading comprehension. *Journal of Experimental Psychology: General*, 3:228–238.

A. Kennedy and J. Pynte. 2005. Parafoveal-on-foveal effects in normal reading. *Vision Research*, 45:153–168.

L. Konieczny and P. Döring. 2003. Anticipation of clause-final heads. evidence from eye-tracking and srns. In *Proceedings of the 4th International Conference on Cognitive Science*.

L. Konieczny. 2000. Locality and parsing complexity. *Journal of Psycholinguistic Research*, 29(6).

R. Levy and E. Gibson. 2013. Surprisal, the pdc, and the primary locus of processing difficulty in relative clauses. *Frontiers in Psychology*, 4(229).

K. Maekawa, M. Yamazaki, T. Ogiso, T. Maruyama, H. Ogura, W. Kashino, H. Koiso, M. Yamaguchi, M. Tanaka, and Y. Den. 2014. Balanced Corpus of Contemporary Written Japanese. *Language Resources and Evaluation*, 48:345–371.

K. Nakatani and E. Gibson. 2010. An on-line study of japanese nesting complexity. *Cognitive Science*, 34(1):94–112.

M. Osaka and N. Osaka. 1994. [working memory capacity related to reading: measurement with the japanese version of reading span test] (in japanese). *Shinrigaku Kenkyu: The Japanese Journal of Psychology*, 65(5):339–345.

D. Roland, G. Mauner, C. O'Meara, and H. Yun. 2012. Discourse expectations and relative clause processing. *Journal of Memory and Language*, 66(3):479–508.

M. Sainio, J. Hyöna, K. Bingushi, and R. Bertram. 2007. The role of inter spacing in reading japanese: An eye movement study. *Vision Research*, 47:2575–2584.

S. Uchida, E. T. Miyamoto, Y. Hirose, Y. Kobayashi, and T. Ito. 2014. An erp study of parsing and memory load in japanese sentence processing – a comparison between left-corner parsing and the dependency locality theory –. In *Proceedings of the Thought and Language/the Mental Architecture of Processing and Learning of Language 2014*.

"How Bullying is this Message?": A Psychometric Thermometer for Bullying

Parma Nand, Rivindu Perera, Abhijeet Kasture
School of Engineering, Computer and Mathematical Sciences
Auckland University of Technology
Auckland, New Zealand
{pnand,rperera,akasture}@aut.ac.nz

Abstract

Cyberbullying statistics are shocking, the number of affected young people is increasing dramatically with the affordability of mobile technology devices combined with a growing number of social networks. This paper proposes a framework to analyse Tweets with the goal to identify cyberharassment in social networks as an important step to protect people from cyberbullying. The proposed framework incorporates latent or hidden variables with supervised learning to determine potential bullying cases resembling short blogging type texts such as Tweets. It uses the LIWC2007 - tool that translates Tweet messages into 67 numeric values, representing 67 word categories. The output vectors are then used as features for four different classifiers implemented in Weka. Tests on all four classifiers delivered encouraging predictive capability of Tweet messages. Overall it was found that the use of numeric psychometric values outperformed the same algorithms using both filtered and unfiltered words as features. The best performing algorithms was Random Forest with an F1-value of 0.947 using psychometric features compared to a value of 0.847 for the same algorithm using words as features.

1 Introduction

Harassment and bullying as common forms of unacceptable behaviour have been topics in psychological research for a long time. Harassment aims to discriminate people on the basis of race, sex, disability etc. and bullying aims to intimidate people. The use of the terms "cyberharassment" and "cyberbullying" is relatively new. Both terms involve the application of modern technologies to support negative human behaviour. The ubiquitousness and affordability of modern technology makes it easy for organisations and individuals to easily communicate via e-mails, chat rooms, message boards and social media which generates a huge amount of public and private data containing information reflecting the pulse of the society. Unfortunately, this accessibility of the technology has also created a forum for the expression of negative human emotions, some of which also gives rise to tragic outcomes such as suicides, self harm and mental illnesses. The aim of our research is to create a framework for detecting cyberharassment from textual communication data, so that systems can be implemented to catch and eliminate harassing texts before it is able to inflict harm.

Bullying is a relationship issue that could result in significant emotional and psychological suffering with some even resulting in suicides (Boyd, 2007). Cyberbullying is even worse because it can follow the victim everywhere, happen at anytime and it is frequently anonymous. The number of victims in all age groups is growing worldwide. It is urgent to progress the research in this area in order to develop advanced methods for fast detection and prevention of cyberbullying cases. This involves analysis of a huge amount of social network textual data which is largely unstructured and chaotic.

A range of challenging tasks are associated with cyberharassment research. For example, bullying is to a great extent connected to the nature of the victim rather than solely on the language and topic of text. It generally involves the use of known weaknesses of victims that they cannot change. Texts can still contain profanities and include sensitive topics without being bullying or any psychological effect on the

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 695–706, Osaka, Japan, December 11-17 2016.

victim. In order determine whether if cyberbullying occurred, we would need to attain information on the victim's reaction and classify this reaction. This is a challenging task in itself and not part of our work. The context or background knowledge can also determine if a text is bullying or not. It is possible for individuals in close friendship to communicate using profanities and on sensitive topics, without causing a negative psychological effect. The actual context is normally not included in texts since it is difficult to determine, especially for short pieces which is typical of social media type communications.

To be able to computationally detect cyberbullying, one would need texts on time-line based conversational threads between individuals, which is difficult to obtain due to privacy and propriety implications. Although there has been an abundance of studies on cyberbullying from the social and psychological perspectives, good computational systems to identify cyberbullying are rare. The existing computational studies (Dinakar et al., 2011; Yin et al., 2009; Xu et al., 2012b; Van Royen et al., 2015; Cortis and Handschuh, 2015; Squicciarini et al., 2015; Han et al., 2015; Kansara and Shekokar, 2015) use predominantly keywords as features in classification algorithms. Those approaches deliver results which are good indicators of harassment and they provide text which can be processed further to find bullying cases. Identification of harassment in singleton blogging texts would be extremely useful on social media platforms as a monitoring tool. Once a blog has been found to be harassing, the corresponding conversations could then be monitored more closely over a certain time in order to identify the progress from cyberharassment to cyberbullying.

In this paper, we apply a well established theory from psychology integrated in a freely available software tool to develop a framework with the goal to classify pieces of texts as bullying. Instead of using a lexicon of profanities, the framework applies a much richer dictionary of words and weights, referred to as psychometric measurements in the psychology literature, to create a rich set of numeric features which are then used four different classification algorithms. We train four different classifiers using two types of filtered and unfiltered inputs (high correlation features and annotated text). All classifiers delivered reasonable results, however, the "Random Forest" classifier achieved the highest precision rate.

1.1 Psychometric Analysis

Harassment and bullying are manifestations of negative emotions which are described as *variables* in psychology (Browne, 2000). In many scientific experiments, we obtain exact measurements of well defined features and then we normally derive answers for correlated questions under investigation. It is not straightforward to find answers to questions that deal with aspects of human psychology such as bullying and harassment. Features related to psychological constructs are typically not clearly defined and cannot be directly measured. Additionally, indirect determinations of values are often subject to substantial measurement errors and varying degrees of correlation with the question under investigation. Psychologists declare those features as latent or hidden variables. *Latent variables* are those that are indirectly measured by deducing relations between manifest, or *observed variables* (Browne, 2000).

Psychometrics is a discipline that deals with the quantification and analysis of capacities and processes in human brains, which often manifest as latent variables. Psychometrics deals with the construction of procedures for measuring psychological constructs and the development of mathematical or statistical models for the analysis of the psychological data. Typically, psychological constructs are multi-variate consisting of a large proportion of latent variables. Researchers generally measure observed variables and then find correlations with the latent variables under investigation.

Psychometrics has been used in applications where measurements of some aspects of human psychology are required. For instance, it has been used to determine an individual's personality alignment to a given set of characteristics. In this study, the psychometric variables are measured to evaluate strengths and weaknesses of a person for specific tasks. The results have been mapped to their cognitive abilities and general social behavioural style as described by Kline (2013). Many companies perform psychometric evaluations on candidates to identify potential matches for specific job roles. It has also been used to identify individuals' suitability towards specific career paths using psychometric techniques on observed variables.

An individual's writing style is an example of an observed variable, hence its psychometric evaluation

would be able to provide insights into the individual's latent characteristics on a range of psychological processes. Further, psychometric evaluation performed on texts collectively generated by various people on a similar topic can provide insights into the psychological processes of this group as a whole. Twitter, for instance, provides the ability to use "hashtags" which allows multiple users to easily write on a designated topic. This makes it easy to collate Tweets on a topic for the purpose of carrying out psychometric evaluations such as opinion mining on the topic. For example, Poria et al. (2014) provide dependency based rules for sentiment analysis. Apart from binary classification of opinions, more advanced psychometric evaluations can even distinguish between different degrees of individual opinions on a topic.

1.2 Related Work

Detecting cyberharassment is a special case of text classification which is an older research area compared to research on cyberharassment. Text classification involves use of generic machine learning algorithms to classify texts into two or more categories using features extracted from the words in the text. In order to be able to extract features there maybe various types of preprocessing tasks such as Part-of-Speech tagging, stop word removal, named entity recognition, etc., applied which differentiates the various systems and adapts them better for the type of classification application at hand. Text classification techniques developed in the areas of sentiment analysis and opinion mining are especially applicable to cyberharassment detection as they also deal with human psyche expressed as online texts. For example, Bollen et al. (2011) report the results of a text classification framework applied to support business decision. This paper presents the results for analysing 10 million Tweets from three million users over a 10 month period to predict changes in the stock market. The authors aim to answer whether the analysed mood expressed in the Tweets from a cross section of three million people is related to the Down Jones Industrial Average(DJIA). The authors applied the following two mood tracking tools:

- OpinionFinder, a system that processes Tweets and separates them into positive versus negative sentiments
- Google-Profile of Mood States (GPOMS), a system that classifies Tweets based on the mood categories; happy, vital, alert, calm, sue, kind.

Granger Causality Analysis and fuzzy neural network classifiers were used for cross-validation of the resulting mood time series against large public events such as US presidential election and Thanksgiving in this paper. The paper reports a correlation of the changes in DJIA value with predicted values from the classifier with an accuracy of 87%.

Cortis and Handschuh (2015) report cyberbullying analysis in the backdrop of Tweets related to two top trending world events in 2014, namely Ebola virus outbreak in South Africa and shooting of Michael Brown in Ferguson, Missouri. These events generated bullying comments on Africa people with no connection to Ebola virus and racist comments regarding the shooting for each of the events respectively. The authors of this paper used only limited number of key terms (*whore, hoe, bitch, gay, fuck, ugly, fake, slut, youre, thanks*) to classify Tweets against ground truth classifications as annotated by trained curators. The combined precision values for key terms such as "Whore" ranged from 0.75 to extremely low values of 0.2 for key terms such as "youre" with an average precision of approximately 0.53.

Dadvar and de Jong (2012) also present a cyber bullying detection framework based on MySpace posts. This paper uses a large dataset of 2,200 posts annotated by students and use a SVM classifier. The interesting aspect of this work is that they used four types of features. All the profane words were treated as in one category and the ratio of foul words to the length of the post was used as the feature for the classifier.The other features used were second person pronouns, all other pronouns and TF-IDF all the words in each post. Additionally, the posts were also split by gender based on the hypothesis that bullying characteristics are between the genders. The reported precision value for male specific corpus was 0.44 and for female specific corpus was 0.40. However, the respective recall values were 0.21 and 0.05, which is rather low even after considering the data set size of 2,200 posts. The performance numbers from this

study in addition to similar numbers from others such as (Ptaszynski et al., 2010; Nandhini and Sheeba, 2015; Xu et al., 2012a) was the motivation for us to use the whole text instead of a keyword list based approach to detect cyberharassment in the previous works. The framework presented in this paper uses the psychometric analysis of all the work in the text and converts them to numeric values before using them as features in the classification algorithms. The next section describes this framework in detail.

2 Proposed Framework

In this paper, we describe a framework for processing Tweets to generate a model for the prediction of cyberbullying. Four main groups of experiments were done in order to determine the best performing classifier for the different forms of input data. The following variations of the Tweet data were tested after cleaning it for extra punctuations and repeated words. Various combinations of

1. All words.
2. Words with stopwords removed.
3. Words above a threshold correlation value with 2 types of filters, Correlation Feature Selection (CFS) and Infogain Attribute Evaluator (IAE).
4. Psychometric values for all words.
5. Psychometric values for stopwords removed.
6. Psychometric values for filtered words.

2.1 Architecture

Figure 1 shows a schematic representation of the framework. We divided our model into the following phases:

1. Extraction of Tweets with typical keywords for bullying using the TAGS tool.
2. Pre-processing of archived Twitter data.
3. Human annotation of Tweets to establish ground truth.
4. The LIWC2007 software generates a vector for each Tweet and the results are saved in two EXCEL files, corresponding to true positives and true negatives.
5. Vectors are filtered (using two different techniques) to reduce the number of attributes before classification.
6. Combined files (filtered or unfiltered) are used to train four machine learning classifiers in WEKA.
7. Evaluation of the classification results.

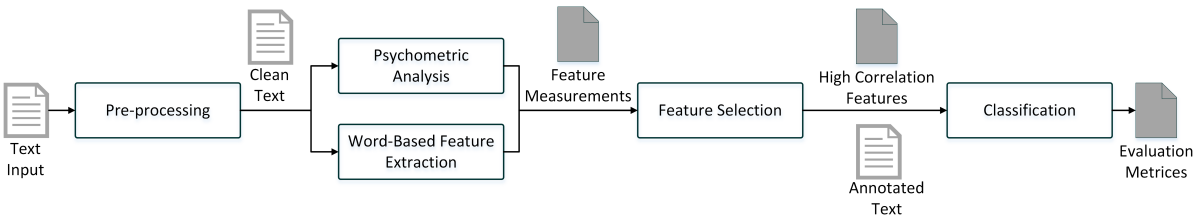


Figure 1: Schematic representation of the Architecture

2.2 Extraction of Tweets

We used the Twitter Archiving Google Spreadsheet[1] (TAGS) available for the public to explore opportunities for the exploitation of the huge amount of data generated everyday by the Twitter community.

2.3 Data preparation/Pre-processing

Tweets have a size restriction of 140 characters and typically originate from mobile devices, and due to small keypads they contain excessive amounts of noise. Apart from the "typo" errors, a range of other types of noise are usually contained in Tweet messages, such as, extra space and non-ASCII characters, due to the nature of the platform. In order to be able to extract the maximum possible semantics, we eliminated the following types of noise using techniques described by Nand et al. (2014).

Word Variations - e.g., tmro and 2moro replaced by tomorrow

Multi Word Abbreviation - e.g., lol replaced by laugh out loud

Slangs - e.g., gonna replaced by going to.

One of the major hurdles in researching online harassment is access to data, which, by the nature of the purpose, is usually private to an individual or to a closed group. Twitter, however, does have publicly available conversational data exhibiting harassment characteristics. For this research we downloaded a total of 2500 public Tweets using the TAGS archiving tool to generate Google Spreadsheets over a period of 2 months. The Tweets were retrieved by entering typical keywords for bullying as recommended in psychology literature (Ortony et al., 1987; Cortis and Handschuh, 2015; Squicciarini et al., 2015; Browne, 2000; Ybarra, 2010).

Keywords used to download Tweets: nerd, gay, loser, freak, emo, whale, pig, fat, wannabe, poser, whore, should, die, slept, caught, suck, slut, live, afraid, fight, pussy, cunt, kill, dick, bitch.

After removing the duplicates and ReTweets we had a sample of 1689 instances containing one or more of the keywords. In addition, @Usernames. #Hashtags and Hyperlinks were removed because the LIWC (Linguistic Inquiry and Word Count) - tool accepts only plain ASCII text files. The cleaned texts were then manually classified by a set of 3 trained annotators into either "cyberbullying" and "non-cyberbullying". The sample was divided into three lots containing 563 instances each. Each lot was assigned to two different annotators which meant that each Tweet was annotated twice by two different annotators. The annotators tagged the instances based on following guide lines:

Character Attacks: the reputation, integrity and morals of an individual are targeted with the purpose of defamation.

Competence attacks: bullies denigrate individuals ability to do something.

Malediction: an attack in which bullies curse and express a wish for some type of misfortune or pain to materialize in an individuals life.

Physical appearance: targeted on an individuals look and bodily structures. Typically, physical attributes of humans are found to shape and develop their personality and social behavioral relations. Due to the need of an individual to be socially accepted, these types of attacks make victims feel socially neglected making a long lasting negative impact on their self-esteem.

Insults: profanity is used as an attack wherein bullies use extremely offensive language that typically include foul, lewd, vulgar language in addition to swearing and cursing words.

Verbal abuse: includes false accusations or blames, extreme criticisms and judgements about an individual and/or statements that negatively define the victim.

[1] https://tags.hawksey.info/

Teasing: hurtful in nature and done as a spectacle for others to witness resulting in harassment and humiliation of the victim. It is perceived as a form of emotional abuse.

Threats: generally anonymous in cyberbullying. Due to this anonymity victims tend to live in constant fear that leads to long-lasting depression, low self-esteem and delinquent behaviours.

Name-calling: bullies use denigrating, abusive names and associate them to the victims leaving them humiliated in front of others.

Mockery: pass comments on victims making them feel worthless, disrespecting them and make fun of them in front of others. Escalated form of mockery leads to low self-esteem of the victims.

In order to ensure that all true positives were identified each of the three sets of Tweets were annotated twice by two different annotators with a Cohen's kappa of 0.833. This resulted in a total of 427 instances as harassing Tweets (true positives). In order to further validate the true positives, we did a second round of annotation by randomly dividing the 427 true positives into three lots, and had two annotators to again check each true positive instance. If a true positive instance was rejected by two annotators in the second round we rejected that instance and assigned it as non-cyberbullying or true negative. If a tweet was annotated as bullying by one annotator and non-bullying by the second, then it was looked at together and an agreement was reached. After the second round we had a total of 376 true positive instances which was taken as the ground truth.

2.4 LIWC2007, Psychometric Analysis

Linguistic Enquiry Word Count (LIWC) is a free text analysis application, originally designed to study emotions in text files. It started with counting words in psychology which were relevant to certain categories and it has been developed into a continuously improving software tool. At the beginning, human emotion words were categorized only into negative and positive words. The system architecture of the LIWC too is described in (Pennebaker et al., 2015) and the psychological categories can be found in (Tausczik and Pennebaker, 2010). The default dictionary of LIWC2007 is composed of 4,500 "dictionary words" grouped into four high level categories. The input text is a set of so-called "target words". LIWC2007 processes text files in one of many provided formats and writes the analysis in one output file. If a target word has been found in the dictionary, it will be assigned to one or more categories and the count of each of the categories to which the word belongs is incremented. For example, the word "cried" is part of five categories: sadness, negative emotion, overall affect, verb and past tense verb. The target words are the elements of the pre-processed Tweets in our study, where each Tweet is a unique instance. The output is a vector with 67 numeric values per Tweet corresponding to the number of counts for each of the word category divided by the total number of assignments. The average number of words per Tweet in all of the used, 1313 Tweet, was 18 and 89% of our target words have were found to be in the dictionary. The average number of words in a cyberbullying Tweet was 13 out of which an average of 11 words were assigned to the psychometric word categories. Mostly internet slang words or names of locations were not found in the dictionary.

3 Results and Evaluation

The graph in Fig. 2 shows the average psychometric measurement values for a subset of all 67 features generated by the LIWC tool for the Tweets manually identified as cyberbullying (blue or left line) and those that are not identified as cyberbullying (red or right line). The differences between the calculated values (length of the red and the blue line) for some word categories such as "you", "swear", "negemo", "anger", "bio" and "death" are relatively large and we can assume that the discriminatory power for true positives would be high for these features. But the results have shown that the classifier accuracies dropped by about 6% when the filters reduced the number of features to these features with high correlation. Another observation is that the value for the "sexual" category is almost equal for positive and negative instances because the downloaded Tweets have been selected based on a majority of words

belonging with the "sexual" category. From this we can infer that the words which are elements of the "sexual" category is not necessarily a good predictor of cyberbullying.

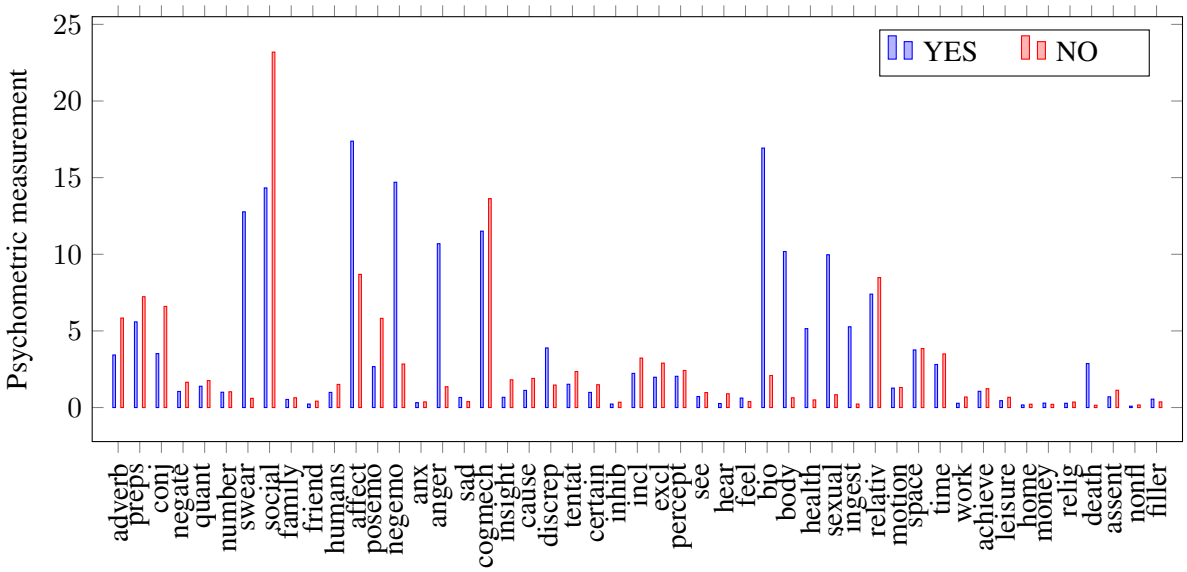


Figure 2: Graph showing the average psychometric measurements for positive and negative instances

The four classifiers have been tested in three lots. The first experiment used all 67 numerical values per instance generated by the LIWC software. Table 1 shows the numerical results for precision and recall for our selected classifiers. The second column in Table 1 provides the results of the cross validation of our trained model using the training data set and the third column shows the results for the test data. We used the usual split of available data, that means 66% for the randomly selected training set and the remaining 34% for the test set. The precision values for the Random Forest classifier are almost equal. SMO achieved the largest value for precision of the training set compared to the other classifiers, but the Random Forest classifier delivered a higher value for the test set. The Random Forest classifier for identifying cyberbullying Tweets had the best performance in this experiment.

Classifier	Cross-Val. 10 Training	Split (66-34) Testing	Results"YES"
Random Forest	0.984	**0.983**	Precision
	0.963	**0.935**	Recall
SMO	0.986	0.965	Precision
	0.912	0.902	Recall
Multilayer Perceptron	0.963	0.873	Precision
	0.912	0.894	Recall
J 48 Decision Tree	0.947	0.935	Precision
	0.941	0.935	Recall

Table 1: WEKA Classifier Outputs for Unfiltered Inputs

The second experiment used only the filtered attributes for classification. Table 2 shows the results for the two different filters for the training set (left two columns) and the test set (right two columns). Precision and recall values for the test data are significantly lower Random Forest, Multilayer Perception and J 48 Decision Tree. The SMO classifier performs best with respect to precision and the ING filter delivered slightly better results for the this classifier. Both filters result in equal results for precision of the training data set when the the Random Forest classifier has been selected, but ING performed slightly better for the test data. Only the J48 Decision Tree classifier delivers for the test data the same results for with respect to filtered and unfiltered inputs, for all other classifiers the achieved values (shown in Table 1 and

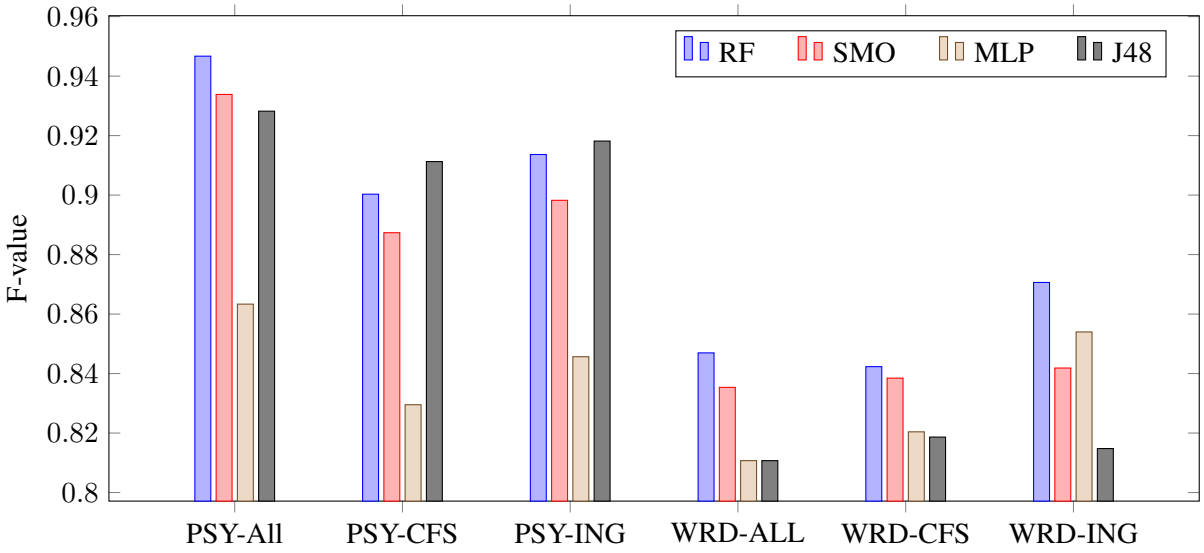


Figure 3: Graph showing the comparison of the F-values for Psychometric versus Words as Features in four classifiers.

Table 2) are equal or better for unfiltered input data.

	Cross-Val. 10		Split (66-34)		
Classifier	CFS	ING	CFS	ING	Results"YES"
Random Forest	0.978	0.978	0.967	0.951	Precision
	0.960	0.952	0.943	0.943	Recall
SMO	0.982	0.986	0.965	0.974	Precision
	0.894	0.910	0.886	0.902	Recall
Multilayer Perceptron	0.951	0.964	0.940	0.866	Precision
	0.939	0.918	0.894	0.894	Recall
J 48 Decision Tree	0.954	0.944	0.932	0.935	Precision
	0.931	0.941	0.894	0.935	Recall

Table 2: WEKA Classifier Outputs for Filtered Inputs

The third experiment was done to determine the effectiveness of the psychometric values generated by the LIWC tool, used as features for the prediction of cyberbullying. Table 3 provides the values for precision, recall and F-value for our test data, filtered and unfiltered applying psychometric features in the upper part of this table. The lower part of the table is a summery of values for the case when we use cleaned Tweets as inputs for the same filtering and classification procedures without applying the LIWC tool to compare the results. The integration of psychometric features produced for all classifiers (filtered or unfiltered attributes) better values. The outcome of our third experiment is a clear vote for transferring the clean text into the 67 LIWC attributes to identify cyberbullying Tweets.

The computed F-values in Table 3 are plotted as a graph in Fig. 3. Table 3 shows that the best performing algorithm was Random Forest with 0.98 precision and 0.91 recall using all 67 psychometric features. The filtered subsets of attributes delivered an approximately 5% drop in accuracy for the Random forest classifier and similar results for the other algorithms. Table 3 also shows that all algorithms performed better with psychometric features compared to word features. For example, the Random Forest precision is 0.983 using psychometric features compared to 0.869 using word features. The graph in Fig. 3 shows that all classifiers using psychometric features perform better than the same classifiers

Psychometric Value Features	**RF**	**SMO**	**MLP**	**J48**
All Features				
Precision	**0.983**	0.968	0.875	0.956
Recall	0.913	0.902	0.852	0.902
F-value	**0.947**	0.934	0.863	0.928
CFS				
Precision	0.926	0.912	0.859	0.926
Recall	0.876	0.864	0.802	0.897
F-value	0.900	0.887	0.830	0.911
ING				
Precision	0.932	0.925	0.863	0.936
Recall	0.896	0.873	0.829	0.901
F-value	0.914	0.898	0.846	0.918
Word Features	**RF**	**SMO**	**MLP**	**J48**
All Word Features				
Precision	0.869	0.859	0.826	0.826
Recall	0.826	0.813	0.796	0.796
F-value	0.847	0.835	0.811	0.811
CFS				
Precision	0.875	0.869	0.829	0.836
Recall	0.812	0.810	0.812	0.802
F-value	0.842	0.838	0.820	0.819
ING				
Precision	0.889	0.874	0.856	0.828
Recall	0.853	0.812	0.852	0.802
F-value	0.871	0.842	0.854	0.815

Table 3: Table showing the evaluation metrics for Psychometric and Words as features used on Random Forest, SMO, MLP and J48 classifiers

using words. There were two marked observations from the experiments; Firstly, All four of the chosen classifiers perform better with all of the features rather than a subset of higher correlation features. This was true for both the techniques used to filter the features and for both types of features, the psychometric values as well as words used as features. Secondly, the performance of the classifiers were higher for all the classifiers using psychometric values compared to use of raw words as features. Again, this was true for all the cases of using all features as well as subset of features using the two filtering algorithms.

Many text classification approaches have been performed with Support Vector Machines (SVM) (SMO is a special version of SVM) and the results have shown in many cases that SVM's are the best classifiers for text (Liu et al., 2005; Lee et al., 2012; Isa et al., 2008; Simanjuntak et al., 2010). In our case, the results show that the Random Forest classifier performed slightly better than SVM in both cases (psychometric numeric measurements as well as words used for features). Its noteworthy, to mention that the J48 algorithm also did comparatively well with an F-value over 0.9 for psychometric measurements used as features. While Random Forest, Support Vector Machine and J48's performance are quite close in most of the cases, the Multi Layer Perceptron performed is consistently lower with F-values around 0.8. The results have shown that the unsuitability of MLPs for cyberbullying, reaffirmed the suitability of SVMs for text classification. However, in addition, the results show that decision tree based algorithms, Random Forest and J48, can also perform well in short text classification, which is contrary to most of the previously reported results.

4 Discussion on Cyberharassment and Future

The issue of cyber harassment in this study as well as other studies have shown that it is complex, both in terms of definition of the problem as well as finding a solution for the problem. Almost all of the prior works (Cortis and Handschuh, 2015; Nandhini and Sheeba, 2015; Kansara and Shekokar, 2015; Xu et al., 2012a; Dinakar et al., 2011; Dadvar and De Jong, 2012; Han et al., 2015) on cyberbullying have worked on the generic problem of cyberharassment, however have referred to them as cyberbullying. However, Hosseinmardi et al. (2015) have analyzed the issue in detail and distinguished between targeted bullying, which has been referred to as cyberbullying in this paper, and general cyber aggression, referred to as cyberharassment in this paper. The broader definition of cyber aggression includes any form of digital media use to intentionally harm another person or persons. This includes targeting individuals by name calling, flaming, denigration, exclusion, etc. as well as indirect forms such as use of profanities, slangs and comments which are indirectly applicable to the individual's group, or the choice(s) made by the individual. Hosseinmardi et al. (2015), provide two additional criteria for cyberbullying as an imbalance of power between the aggressor and the victim. Their second criteria is the repetition of the act over time. The imbalance of power could be in several forms such as physical, psychological or social. For instance, an individual who is more popular bullying a less popular one or one who is physically stronger bullying a weaker one. This characteristic adds an additional level of complexity for automatic detection systems since all forms of the power imbalance is rarely known for members on social media platforms. The second criteria of repetition poses yet another level of complexity, which would require nested layers of data collection and analysis corresponding to threads of conversation rather than contents of individual posts. Research on cyberharassment will continue to be based on contents of individual posts unless the social media sites relax the data access policies and give access to thread based data. Nevertheless, research has continued to progress in detecting different forms of cyberbullying cases, especially in light of increased number of cyberbullying cases resulting in suicides and other forms of self harm. In addition to the text based platforms such as Twitter, there are several social media platforms, such as MySpace and Instagram which use a multi-modal communication model. Use of pictures and videos in addition to texts gives additional ammunition to potential bullies and presents an even greater challenge for bullying detection researchers. In spite of these challenges researchers (Han et al., 2015; Hosseinmardi et al., 2015) have attempted to work on systems to detect bullying on multi-modal social platforms by using a combination of pictures and text to detect true positives with reasonably good accuracy. For instance, Hosseinmardi et al. used a combination of text, pictures and user metadata to classify Intagrams with a recall of 76% and a precision of 62% using a MaxEnt classifier. The authors used linear SVM classifiers text based n-grams with stop word removal and used crowd sourced CrowdFlower website to establish ground truth. Although the use of the images in this study was at a basic level with categorization of images into categories such as Person, Drugs, Car, Nature and Celebrity as features for the classifier, this is a step in the right direction towards processing multimedia blogs. A lot more effort is required before it would be possible to build systems to detect harassment on multimedia social media platforms.

5 Conclusion

The results presented in this paper show that firstly it is possible to detect cyberharassment with fairly good precision. Secondly, we have shown that the use of psychometric measurements from the LIWC tool as features delivers slightly better results compared to the use of words for all of the algorithms tested. Further, out of the four classifiers tested, the Random Forest classifier proved to be the best performing both with words as classifiers as well as with psychometric measurements. The psychometric measurements uses the classification of all of the in the text which implies that cyberharassment is not only a function of profane words, but use of other category of words such as use of pronouns also determine whether a piece of text is harassing or not. As future work, we plan to use psychometric measurements with thread based social media texts instead of individual pieces of texts as this will be able to better determine the level of bullying rather than harassment as has been the case in this study.

References

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

Danah Boyd. 2007. Why youth (heart) social network sites: The role of networked publics in teenage social life. *MacArthur foundation series on digital learning–Youth, identity, and digital media volume*, pages 119–142.

Michael W Browne. 2000. Psychometrics. *Journal of the American Statistical Association*, 95(450):661–665.

Keith Cortis and Siegfried Handschuh. 2015. Analysis of cyberbullying tweets in trending world events. In *Proceedings of the 15th International Conference on Knowledge Technologies and Data-driven Business*, page 7. ACM.

Maral Dadvar and Franciska De Jong. 2012. Cyberbullying detection: a step toward a safer internet yard. In *Proceedings of the 21st International Conference on World Wide Web*, pages 121–126. ACM.

Karthik Dinakar, Roi Reichart, and Henry Lieberman. 2011. Modeling the detection of textual cyberbullying. *The Social Mobile Web*, 11:02.

Richard Han, Qin Lv, and Shivakant Mishra. 2015. Analyzing labeled cyberbullying incidents on the instagram social network. In *Social Informatics: 7th International Conference, SocInfo 2015, Beijing, China, December 9-12, 2015, Proceedings*, volume 9471, page 49. Springer.

Homa Hosseinmardi, Sabrina Arredondo Mattson, Rahat Ibn Rafiq, Richard Han, Qin Lv, and Shivakant Mishr. 2015. Prediction of cyberbullying incidents on the instagram social network. *arXiv preprint arXiv:1508.06257*.

Dino Isa, Lam H Lee, VP Kallimani, and Rajprasad Rajkumar. 2008. Text document preprocessing with the bayes formula for classification using the support vector machine. *IEEE Transactions on Knowledge and Data engineering*, 20(9):1264–1272.

Krishna B Kansara and Narendra M Shekokar. 2015. A framework for cyberbullying detection in social network. *International Journal of Current Engineering and Technology*, 5.

Paul Kline. 2013. *Handbook of psychological testing*. Routledge.

Lam Hong Lee, Chin Heng Wan, Rajprasad Rajkumar, and Dino Isa. 2012. An enhanced support vector machine classification framework by using euclidean distance function for text document categorization. *Applied Intelligence*, 37(1):80–99.

Tie-Yan Liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. 2005. Support vector machines classification with a very large-scale taxonomy. *ACM SIGKDD Explorations Newsletter*, 7(1):36–43.

Parma Nand, Ramesh Lal, and Rivindu Perera. 2014. A HMM POS Tagger for Micro-Blogging Type Texts. In *Proceedings of the 13th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2014)*.

B Sri Nandhini and JI Sheeba. 2015. Online social network bullying detection using intelligence techniques. *Procedia Computer Science*, 45:485–492.

Andrew Ortony, Gerald L Clore, and Mark A Foss. 1987. The referential structure of the affective lexicon. *Cognitive science*, 11(3):341–364.

James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of liwc2015. *UT Faculty/Researcher Works*.

Soujanya Poria, Erik Cambria, Gregoire Winterstein, and Guang-Bin Huang. 2014. Sentic patterns: Dependency-based rules for concept-level sentiment analysis. *Knowledge-Based Systems*, 69:45–63.

Michal Ptaszynski, Pawel Dybala, Tatsuaki Matsuba, Fumito Masui, Rafal Rzepka, and Kenji Araki. 2010. Machine learning and affect analysis against cyber-bullying. In *Proceedings of the 36th Annual Convention of the Society for the Study of Artificial Intelligence and the Simulation of Behaviour*, pages 7–16.

David Allister Simanjuntak, Heru Purnomo Ipung, Anto Satriyo Nugroho, et al. 2010. Text classification techniques used to faciliate cyber terrorism investigation. In *Advances in Computing, Control and Telecommunication Technologies (ACT), 2010 Second International Conference on*, pages 198–200. IEEE.

A Squicciarini, S Rajtmajer, Y Liu, and C Griffin. 2015. Identification and characterization of cyberbullying dynamics in an online social network. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 280–285. ACM.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of language and social psychology*, 29(1):24–54.

Kathleen Van Royen, Karolien Poels, Walter Daelemans, and Heidi Vandebosch. 2015. Automatic monitoring of cyberbullying on social networking sites: From technological feasibility to desirability. *Telematics and Informatics*, 32(1):89–97.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012a. Learning from bullying traces in social media. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 656–666. Association for Computational Linguistics.

Jun-Ming Xu, Xiaojin Zhu, and Amy Bellmore. 2012b. Fast learning for sentiment analysis on bullying. In *Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining*, page 10. ACM.

M Ybarra. 2010. Trends in technology-based sexual and non-sexual aggression over time and linkages to non-technology aggression. *National Summit on Interpersonal Violence and Abuse Across the Lifespan: Forging a Shared Agenda*.

Dawei Yin, Zhenzhen Xue, Liangjie Hong, Brian D Davison, April Kontostathis, and Lynne Edwards. 2009. Detection of harassment on web 2.0. *Proceedings of the Content Analysis in the WEB*, 2:1–7.

Learning grammatical categories using paradigmatic representations: Substitute words for language acquisition

Mehmet Ali Yatbaz[1]* **Volkan Cirik**[2]* **Aylin Küntay**[3] **Deniz Yuret**[3]
[1]Facebook Inc. 1 Hacker Way Menlo Park, CA, USA
[2]Carnegie Mellon University, Pittsburgh, PA, USA
[3]Koç University, İstanbul, Turkey
{myatbaz,vcirik,akuntay,dyuret}@ku.edu.tr

Abstract

Learning word categories is a fundamental task in language acquisition. Previous studies show that co-occurrence patterns of preceding and following words are essential to group words into categories. However, the neighboring words, or frames, are rarely repeated exactly in the data. This creates data sparsity and hampers learning for frame based models. In this work, we propose a paradigmatic representation of word context which uses probable substitutes instead of frames. Our experiments on child-directed speech show that models based on probable substitutes learn more accurate categories with fewer examples compared to models based on frames.

1 Introduction

Children abstract grammatical rules from individual words (e.g. baby, talk) and eventually apply them to word categories (e.g. noun, verb, adverb). A word category represents a group of words that can be substituted for one another without altering the grammatical appropriateness of a sentence. Learning word categories is an important step in language development.

The Distributional Hypothesis (Harris, 1954) suggests that words occurring in similar contexts tend to have similar meanings and grammatical properties. Studies on extraction of word categories have shown that distributional information of word co-occurrences is a reliable cue for the learning of word categories (Mintz, 2003; St Clair et al., 2010; Redington et al., 1998). Children need to extract word categories from incoming speech stream in order to be able to predict how words behave in various grammatical contexts and to produce words in appropriate grammatical constructions. Children tend to form word categories that group words used in similar contexts.

To judge how similar two contexts are, one can use syntagmatic or paradigmatic representations of the word context: A syntagmatic representation is based on the neighbors of the target word whereas a paradigmatic representation uses potential substitutes for the target word.

In this paper, we hypothesize that children judge context similarity using a paradigmatic representation: a context is similar to another if the same words can be substituted in both. The following two examples[1] illustrate the advantage of paradigmatic representations in uncovering latent similarities where a syntagmatic representation would fail to see any overt similarities. The word "you" from the first sentence and the word "I" from the second sentence have no common neighbors within the same sentence. The paradigmatic representation, shown below the sentences as substitute word probabilities, captures the similarity of these contexts by suggesting similar top substitutes for both:

(1) *"they fall out when* **you** *put it in the box ."*
you: you(.8188), I(.1027), they(.0408), we(.0146) . . .

(2) *"what have* **I** *got here ?"*
I: we(.8074), you(.1213), I(.0638), they(.0073) . . .

*Work was done when authors were at Koç University.

[1]These examples are extracted from the Anne corpus from CHILDES (MacWhinney, 2000). The computation of substitute word probabilities is described in Section 3.3.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 707–716, Osaka, Japan, December 11-17 2016.

The high probability substitutes reflect both semantic and grammatical properties of the context. The top substitutes for "I" and "you" are pronouns. As an additional example, the top substitutes for the word "fall" in the first sentence are other motion verbs: come(.7875), go(.0305), fall(.0232), . . .

These examples show that the paradigmatic representation can relate a pair of words according to the substitute word distribution of their contexts even when the surface forms of the contexts do not share any common neighbors. This makes the paradigmatic representations of word context more robust to the data sparsity compared to the syntagmatic representations, due to low re-occurrence frequency of large frames.

The rest of the paper is organized as follows. In the next section, we describe prior distributional approaches to word category acquisition. In Section 3, we provide a detailed explanation of our calculations of substitute words. In Section 4, first we introduce the experimental setup we used and give the details of the experiments to contrast the paradigmatic approach with the syntagmatic approach. The last section summarizes our contributions.

2 Related Work

In this section we review distributional approaches to word category acquisition and evaluate them based on two success criteria: accuracy and completeness. Accuracy measures how accurate the predictions were at grouping the words into the same word category together. Completeness, on the other hand, measures how well a given category is predicted.

Redington et al. (1998) define the context of a word as the previous and following words. They construct context vectors of target words for clustering. Using average link clustering, target words are separated into categories. Although the resulting categorizations are generally accurate, the method is weak in completeness because words that do not appear in frequent frames cannot be covered.

Mintz (2003) proposes top-N frequent frames surrounding a target word as a more fitting context to derive word categories from. A frequent frame consists of left and right neighbors that co-occur frequently. Experiments on child-directed speech reveal that frequent frames have the ability to assign word categories with high accuracy. However, this method also lacks satisfactory completeness. St Clair et al. (2010) combine the bigram's coverage power (Redington et al., 1998; Monaghan and Christiansen, 2008) and the accuracy of frequent frames (Mintz, 2003). Their experiments suggest that to match the performance of infants both bigram and trigram sources may need to be used.

Freudenthal et al. (2005) identify a complication of distributional methods for constructing word categories. Distributional methods suggest that words occurring in a similar context can be used interchangeably. They claim the evaluation methods used in studies like (Redington et al., 1998; Monaghan and Christiansen, 2008; Mintz, 2003) could be misleading. Specifically, if a word is substituted with another one in its category, the resulting sentences could be erroneous in a way that they are not observed in infants' speech. As a success criterion, they argue that the proposed categorization should generate plausible sentences. They introduce a chunking mechanism merging words that are seen frequently. The mechanism is successful in generating meaningful sentences, still, the proposed solution is computationally too complex as a learning mechanism in infants.

Alishahi and Chrupała (2012) propose an incremental learning scheme inducing soft word categories while learning the meaning of words. Thothathiri et al. (2012) examine the role of prosody in infants' distributional learning of syntactic categories and concludes that the prosody shows little influence. Reeder et al. (2013) discuss the use of distributional knowledge when the evidence in the possible context of a word is not enough. Furthermore, they explain how and when language users form new categories depending on the overlaps between the context words.

In the related part-of-speech induction literature[2], Schütze (1995) incorporates paradigmatic information by concatenating the left and the right co-occurrence vectors of the right and left neighbors respectively, and groups words that have similar vectors. The limitation of this model is that it uses only bi-gram information and suffers from sparsity as the context size gets larger. Yatbaz et al. (2012) calculate the most probable substitutes of a given context using a 4-gram statistical language model. Their

[2]See (Christodoulopoulos et al., 2010) for a comprehensive review of the part-of-speech induction literature.

model achieves the state-of-the-art result in the part-of-speech induction literature. Part-of-speech induction aims to induce word-categories from large amounts of unannotated text (mostly news corpora). Our paper evaluates the substitute-based context representation by Yatbaz et al. (2012) as a possible feature for classifying words in relatively small amounts of child-directed speech.

3 Method

In this section we explain the experimental methodology we used, including how the input corpora was processed, the language model was trained, the evaluation metrics, and the computational model to learn grammatical categories.

3.1 Input Corpora

To compare results with (St Clair et al., 2010) and (Mintz, 2003), we use the same six corpora of child-directed speech from the CHILDES[3] corpus (MacWhinney, 2000): Anne and Aran (Theakston et al., 2001), Eve (Crystal, 1974), Naomi (Sachs, 1983), Nina (Suppes, 1974), Peter (Bloom et al., 1974; Bloom et al., 1975). Following (Mintz, 2003) we only analyze the adult utterances in sessions where the target child is 2.6 years old or younger. Table 1 summarizes the number of target word tokens and types in each corpus.

Table 1: Summary of the total number of tokens, utterances and types in each child corpus together with the number of utterances and types that are observed as target word in a three word window aXb.

Corpus	Tokens	Utterances	Utterances Categorized		Types	Types Categorized	
			Count	%		Count	%
Anne	121726	93371	42789	45.82	2623	1846	70.37
Aran	129823	104997	54768	52.16	3256	2595	79.69
Eve	78778	59095	27315	46.22	2184	1465	67.07
Naomi	38302	28793	13002	45.15	1883	1194	63.40
Nina	89957	72879	39335	53.97	2036	1580	77.60
Peter	94521	72834	34997	48.05	2145	1472	68.62

The target grammatical categories of words in CHILDES are extracted by first applying the MOR parser (MacWhinney, 2000) and then using the POST disambiguator (Sagae et al., 2004). The accuracy of CHILDES grammatical categories is approximately 95% (Parisse and Le Normand, 2000) and is encoded in the MOR line of the CHILDES corpus.

3.2 Algorithm

We use supervised learning with a feed-forward connectionist model (a single hidden layer neural network) to compare the effect of distributional cues from various context representations on the word category learning. The input is a representation of the word context and the output is a word category. We evaluate two models with different input representations:

- $aX + Xb$ **model:** This is the flexible frames model of St Clair et al. (2010), the best performing syntagmatic model. Consider a five word window $a_1a_2Xb_1b_2$ where X is the target word. The input to the model consists of two one-hot vectors, one that correspond to the preceding bigram (a_1a_2) and one to the succeeding bigram (b_1b_2). Thus two input units are activated to represent the context of each target word.

- $a * b$ **model:** This is the paradigmatic model investigated in this paper. First, a small number of substitutes are sampled for the target word based on an n-gram language model as described in the next section. The input to the model consists of the counts of the substitutes in the sampled set. The length of the input vector is equal to the size of the substitute vocabulary.

[3]Specifically, CHILDES version 2.0.1 is used in experiments.

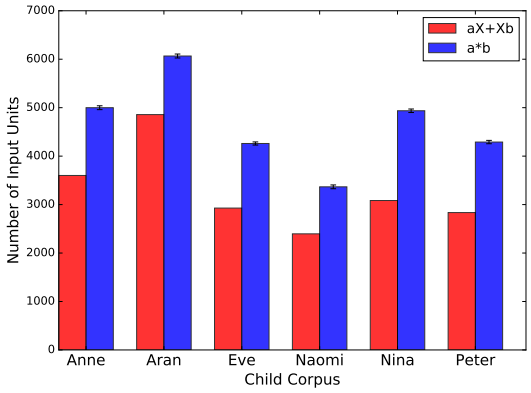


Figure 1: Number of input layer units of the flexible frame ($aX + Xb$) and the substitute based ($a * b$) models. ($a * b$) samples 16 substitutes per target word. Standard errors are reported with error bars.

Figure 1 gives the number of input layer units of syntagmatic ($aX + Xb$) and paradigmatic ($a * b$) models on each child corpus separately. The number of distinct frames is fixed for any given corpus while the number of distinct substitutes varies due to the random sampling. Both models have 10 output units due to the standard labeling (Mintz, 2003).

Unless stated otherwise, all connectionist models in this paper use the following parameters: (1) number of hidden units is set to 200 and initialized randomly for each model. (2) The non-linearity is sigmoid and the learning rate is 0.2.

3.3 Substitute Words

In the paradigmatic ($a * b$) model, we predict the word category of a word in a given context based on its most likely substitute words. We measure the likelihood of substitute words using an n-gram language model. Here, we first describe how substitute probabilities can be computed using an n-gram model and give details on training the n-gram model.

It is best to use both the left and the right context when estimating the probabilities for potential lexical substitutes. For example, in *"He lived in San Francisco suburbs."*, the token *San* would be difficult to guess from the left context but it is almost certainly determined looking at the right context.

We define the context c_w of a given word w as the $2n-1$ word window centered around the position of w : $w_{-n+1} \dots w \dots w_{n-1}$. The probability of a substitute word w in a given context c_w is estimated as:

$$
\begin{aligned}
P(w_0 = w|c_w) &\propto P(w_{-n+1} \dots w_0 \dots w_{n-1}) && (1)\\
&= P(w_{-n+1})P(w_{-n+2}|w_{-n+1}) \dots P(w_{n-1}|w_{-n+1}^{n-2}) && (2)\\
&\approx P(w_0|w_{-n+1}^{-1})P(w_1|w_{-n+2}^{0}) \dots P(w_{n-1}|w_0^{n-2}) && (3)
\end{aligned}
$$

where w_i^j represents the sequence of words $w_i w_{i+1} \dots w_j$. In Equation 1, $P(w_0 = w|c_w)$ is proportional to $P(w_{-n+1} \dots w_0 \dots w_{n+1})$ because the words of the context are fixed. Terms without w_0 are identical for each substitute in Equation 2 therefore they have been dropped in Equation 3. Finally, only the closest $n-1$ words are used in the experiments. Note that the substitute word distribution is a function of the context only and is indifferent to the target word.

Near the sentence boundaries the appropriate terms were truncated in Equation 3. Specifically, at the beginning of the sentence shorter n-gram contexts were used and at the end of the sentence terms beyond the end-of-sentence utterance were dropped.

To train the n-gram model, we first extracted training data of approximately 6.8 million tokens[4] of child-directed speech data from CHILDES. We trained a 4-gram language model on this data with Kneser-Ney discounting using SRILM (Stolcke, 2002). Words that were observed less than 2 times in the language model training data were replaced with an unknown word tag, which gave us a vocabulary size of 21734.

[4] Anne, Aran, Eve, Naomi, Nina, and Peter corpora are excluded.

3.4 Evaluation

To evaluate classification accuracy we use the standard labeling (Mintz, 2003)[5] that categorizes target words as: nouns (including pronouns), verbs (including copula and auxiliaries), prepositions, adjectives, adverbs, determiners, conjunctions, wh-words, negation (i.e., "not") and interjections. Following St Clair et al. (2010), we also report the asymmetric lambda value (Goodman and Kruskal, 1979) to compare the association among the classification of grammatical categories.

3.5 Training and Testing

We measure and compare the classification accuracy of models by applying 10-fold cross validation on the union of six child corpora. We randomly split each child corpus into 10 folds. The main advantage of the cross validation is that all sentences are eventually used both for testing and training.

To compare the effects of paradigmatic representation ($a * b$) with the syntagmatic one ($aX + Xb$) we train and test both models using the identical 10-fold cross validation split. Thus every model in this paper is exposed to the identical sequence of training and testing patterns. Unless stated otherwise, in the rest of this paper, we stopped the training phase of feed-forward connectionist model on each corpus after $100K$ input patterns, used the standard labeling to evaluate model accuracies, calculated substitute distributions as described in Section 3.3, and sampled 16 substitutes per target word in models using the paradigmatic representation.

Section 4 compares the classification accuracies of syntagmatic and paradigmatic representation based models. The effects of the number of substitutes and the language model n-gram order on the paradigmatic model performance are inspected in Section 5 and 6, respectively.

4 Experiment 1: Syntagmatic vs Paradigmatic

To compare the distributional information of syntagmatic and paradigmatic representations, we train separate feed-forward connectionist models for each child corpus based on these representations. St Clair et al. (2010) showed that flexible frames have richer distributional information than other frame types both in terms of classification accuracy and coverage. Thus we only report results of the models based on substitute words ($a * b$) and flexible frames ($aX + Xb$).

Method. All models are trained and evaluated according to steps summarized in Section 3.5. To see the effect of training data size, similar to the analysis in (St Clair et al., 2010), we split the training of each model into short and long training phases in which we stop and evaluate the models on the corresponding test sets after presenting identical 10K and 100K training patterns, respectively.

Results of Short Training Phase. Table 2 gives the overall classification accuracies of $aX + Xb$ and $a * b$ models on each child corpus. The accuracy of $a * b$ model significantly outperforms the $aX + Xb$ model on each child corpus even with a limited amount of training patterns. Lambdas of the $a * b$ model are significantly closer to the perfect association than lambdas of the $aX + Xb$ model. Lambdas of both models are significantly different from zero association.

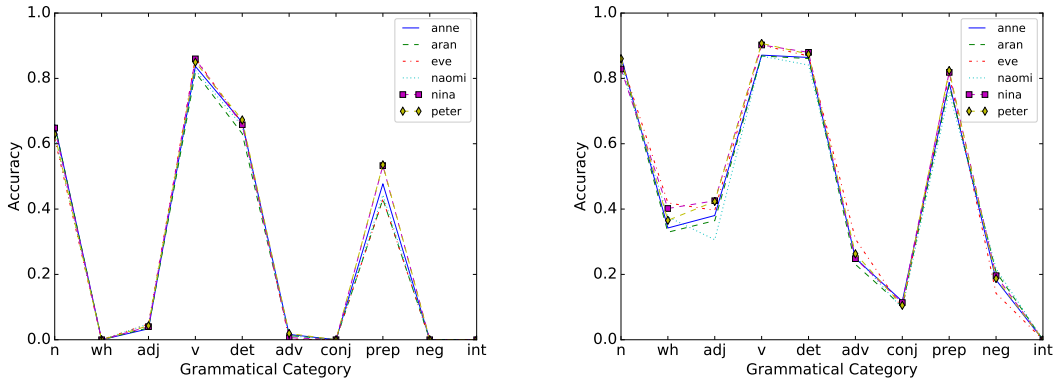
To further investigate the accuracy gap between $aX + Xb$ and $a * b$ models, we plot the classification accuracies of each grammatical category in the standard labeling for both models in Figure 2. Even after 10K training patterns both models are able to achieve relatively high accuracies on nouns (*n*), verbs (*v*), determiners (*det*) and prepositions (*prep*) than the rest of the word categories. The $a * b$ model is more successful than the $aX + Xb$ model in learning word categories such as wh-words (*wh*), adjectives (*adj*), adverbs (*adv*), conjunctions (*conj*), and negations (*neg*).

Finally, with limited amount of training patterns, the $a * b$ model is able to categorize nine out of ten grammatical categories in each child corpus with some accuracy. On the other hand, the $aX + Xb$ model performs poorly on *wh*, *conj*, *adv*, *neg* and *int* and can not correctly classify any members of these word groups in at least one of the child corpora.

[5] (Mintz, 2003) also defined an expanded labeling in which pronouns, auxiliaries and copula forms have their own categories.

Table 2: 10-fold cross-validation classification accuracies of models based on flexible frames ($aX + Xb$) and substitutes ($a * b$) on each child corpus after 10K training patterns are summarized. Standard errors are reported in parentheses. Lambdas of $aX + Xb$ and $a * b$ are both tested against each other and zero association by using two tailed z-test. All tests have $p < .001$.

Corpus	$aX + Xb$		$a * b$	
	Accuracy	λ	Accuracy	λ
Anne	.6252 (.0231)	.4323 (.0352)	.7970 (.0069)	.6925 (.0111)
Aran	.5968 (.0218)	.3908 (.0327)	.7783 (.0083)	.6653 (.0123)
Eve	.6193 (.0192)	.4248 (.0306)	.8091 (.0100)	.7116 (.0141)
Naomi	.6054 (.0236)	.3960 (.0395)	.7771 (.0100)	.6598 (.0178)
Nina	.6438 (.0216)	.4521 (.0362)	.8146 (.0096)	.7150 (.0162)
Peter	.6255 (.0246)	.4402 (.0372)	.8086 (.0088)	.7140 (.0130)



(a) $aX + Xb$ (b) $a * b$

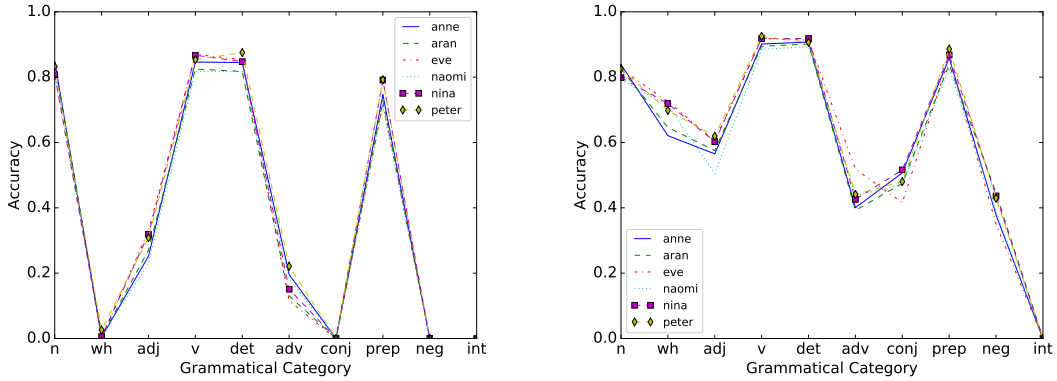
Figure 2: Individual tag accuracies of $aX + Xb$ and $a * b$ on each child corpus after $10K$ training patterns are presented.

Results of Long Training Phase. The previous section shows that the $a * b$ model is more accurate than the $aX + Xb$ model on learning word categories with limited amount of language exposure. In this section each model is trained with 100K input patterns to observe the effect of more extensive language exposure on learning.

Table 3 summarizes the overall classification accuracies of $aX + Xb$ and $a * b$ models on each child corpus. Although differences between corresponding accuracies and lambda values of $aX + Xb$ and $a * b$ models are less than 10K experiments, the $a * b$ model is still significantly more accurate than the $aX + Xb$ model on all child corpora. The $a * b$ model benefits less from the extensive training than the $aX + Xb$ model. One possible explanation for this behavior is that the number of input units of the $a * b$ model on each child corpus is significantly higher than the $aX + Xb$ (see Figure 1) while the number of hidden units is fixed to 200 for both models. Following St Clair et al. (2010), we experimented with the number of hidden units such that the ratio between the number of input units and the number of hidden units is the same for both models. We did not observe significant changes on the result.

Table 3: 10-fold cross-validation classification accuracies of models based on flexible frames ($aX + Xb$) and substitutes ($a * b$) on each child corpus after 100K training patterns are used. Standard errors are reported in parentheses. Lambdas of $aX + Xb$ and $a * b$ are both tested against each other and zero association by using two tailed z-test. All tests have $p < .001$.

Corpus	$aX + Xb$		$a * b$	
	Accuracy	λ	Accuracy	λ
Anne	.7628 (.0075)	.6407 (.0124)	.8311 (.0068)	.7442 (.0109)
Aran	.7337 (.0059)	.5977 (.0081)	.8139 (.0073)	.7189 (.0108)
Eve	.7580 (.0068)	.6351 (.0083)	.8396 (.0107)	.7576 (.0160)
Naomi	.7316 (.0086)	.5892 (.0113)	.8041 (.0090)	.7000 (.0169)
Nina	.7755 (.0040)	.6547 (.0075)	.8389 (.0097)	.7523 (.0165)
Peter	.7670 (.0071)	.6518 (.0088)	.8379 (.0073)	.7579 (.0112)



(a) $aX + Xb$ (b) $a * b$

Figure 3: Individual tag accuracies of $aX + Xb$ and $a * b$ on each child corpus after $100K$ training patterns are presented.

Similar to the 10K results, $aX + Xb$ model performs poorly on *wh*, *conj*, *neg*, and *int* as shown in Figure 3. Both models accurately learn the noun, verb, determiner, and preposition groups. However, the $a * b$ model is significantly more accurate on adjectives, conjunctions, and negation.

5 Experiment 2: Number of Substitutes

In this experiment we analyze the effect of the number of substitutes sampled per context on the classification accuracy.

Method. We used the same experimental settings except that the number of substitutes per target word is varied between 1 and 64. We did not observe any significant difference on model classification accuracies for the number of substitutes that are more than 64.

Results and discussion. Figure 4 plots the model classification accuracy of each child corpus versus the number of substitutes. The classification accuracy dramatically increases on each child corpus until the number of substitutes reaches 16. After 16 substitutes, increasing the number of substitutes does not significantly change the classification accuracy. Thus, the model is fairly robust to the number of substitutes as long as the model can observe at least 16 substitutes per target word.

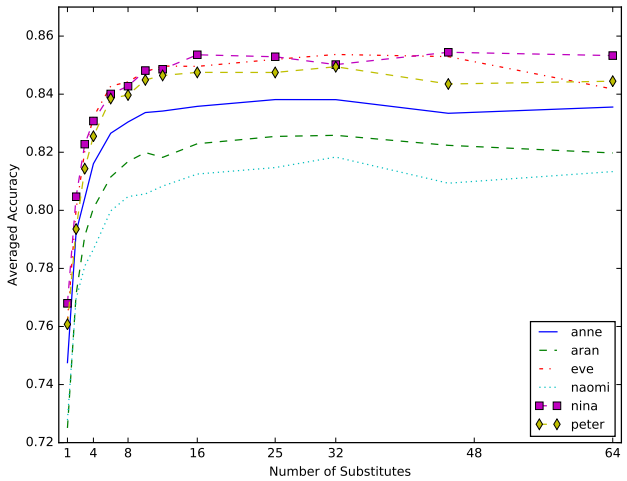


Figure 4: 10-fold cross validation accuracy of each child corpus for different number of substitutes.

6 Experiment 3: Language Model N-gram Order

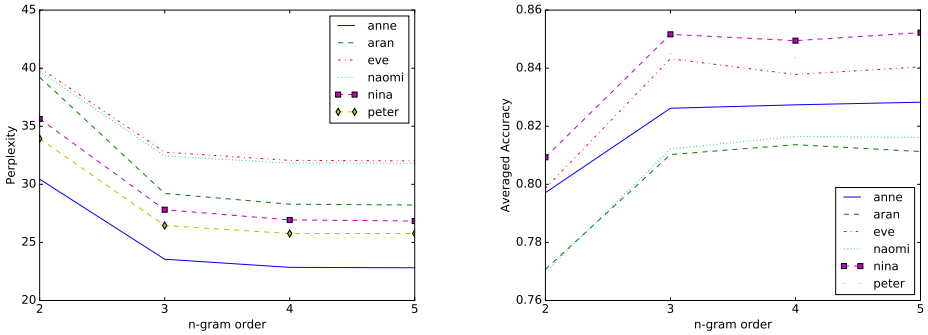
In this set of experiments, we test the paradigmatic model by changing the n-gram order of the language model that is used to sample substitutes. A language model defines probabilities for the sequences of

strings in a language. The n-gram order of language model determines the number of preceding items taken into account while determining the probability of the upcoming word. The previous studies show that young children are sensitive to statistical properties of language (Saffran et al., 1996) and are able to store 4-word sequences (Bannard and Matthews, 2008). Experiments with adults also suggest that the language users are sensitive to co-occurrence patterns beyond bigram (Arnon and Snider, 2010).

The perplexity of the language model is a measurement of the variation of words that can be observed in a given n-gram context window and is determined by n-gram order of the language model. Therefore, as the n-gram order increases the model assigns more relevant substitutes to the context[6].

Method. We used the same experimental settings except that the n-gram order of the language model that is used to sample substitutes is varied from 2 to 5.

Results and discussion. The perplexity of each child corpus is dramatically improved when the n-gram order of the language model is increased from 2 to 3 and varies slightly for orders higher than 3. Figure 5(a) plots the perplexity versus the n-gram order. Figure 5(b) plots the model classification accuracy versus the n-gram order on each child corpus which slightly improve for orders higher than 3 which is in fact parallel to the perplexity trends in Figure 5(a). Overall, the classification accuracy of paradigmatic model is highly correlated with the perplexity of the language model that is used to sample substitutes.



(a) (b)

Figure 5: Language Model perplexities on each child corpus for different n-gram orders are presented on the left figure while 10-fold cross validation accuracies calculated based on these models are presented on the right.

7 Conclusion

In this work, we proposed representing word context with substitute-based paradigmatic representations as opposed to neighbor-based syntagmatic representations for word category acquisition. The paradigmatic approach suggests using probable substitutes of word ($a * b$). On the other hand, the syntagmatic approach we used proposes using the preceding bigram and the succeeding bigram, whichever is fruitful ($aX + Xb$). Our experiments showed that paradigmatic representation of context is more accurate in learning word categories.

To contrast these two representations we replicated the experimental setup of St Clair et al. (2010). Experiments showed that when the models exposed to limited amount of training patterns the $a * b$ is significantly more accurate than $aX + Xb$. Results of the long training phase demonstrated the same pattern, however, the gap between these approaches decreased.

We investigated the dependency of the model to the number of substitutes sampled for each context. In this experimental setup the number of substitutes varies from 1 to 64. The results showed that the accuracy of the model dramatically increases up to 16. After 16 substitutes, no significant improvement in accuracy was observed. We conclude that the model is robust as long as 16 substitutes are sampled.

[6](Goodman, 2001) showed that the perplexity plateaued when the order is higher than 5 for datasets of about 10^8 words.

We explored the effect of the n-gram order of the language model to the accuracy of the model. While determining the probability of the next word in a sequence of words, n-gram order determines how many preceding words should be used. Our results demonstrated that the model's performance depends on the n-gram order of the language model up to order 3, larger contexts do not seem to improve the performance.

Acknowledgements

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) grants 114E628 and 215E201.

References

Afra Alishahi and Grzegorz Chrupała. 2012. Concurrent acquisition of word meaning and lexical categories. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 643–654. Association for Computational Linguistics.

Inbal Arnon and Neal Snider. 2010. More than words: Frequency effects for multi-word phrases. *Journal of Memory and Language*, 62(1):67–82.

Colin Bannard and Danielle Matthews. 2008. Stored word sequences in language learning the effect of familiarity on children's repetition of four-word combinations. *Psychological Science*, 19(3):241–248.

Lois Bloom, Lois Hood, and Patsy Lightbown. 1974. Imitation in language development: If, when, and why. *Cognitive Psychology*, 6(3):380 – 420.

Lois Bloom, Patsy Lightbown, Lois Hood, Melissa Bowerman, Michael Maratsos, and Michael P Maratsos. 1975. Structure and variation in child language. *Monographs of the society for Research in Child Development*, pages 1–97.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two Decades of Unsupervised POS Induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 575–584, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Crystal. 1974. Roger brown, a first language: the early stages. cambridge, mass.: Harvard university press, 1973. pp. xi + 437. *Journal of Child Language*, 1:289–307, 10.

Daniel Freudenthal, Julian M Pine, and Fernand Gobet. 2005. On the resolution of ambiguities in the extraction of syntactic categories through chunking. *Cognitive Systems Research*, 6(1):17–25.

Leo A Goodman and William H Kruskal. 1979. Measures of association for cross classifications. In *Measures of association for cross classifications*, pages 2–34. Springer.

Joshua T. Goodman. 2001. A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403 – 434.

Zellig Sabbetai Harris. 1954. Word. *Distributional Structure*, 10(23):146–162.

B. MacWhinney. 2000. *The CHILDES Project: Tools for Analyzing Talk, Volume II: The Database*, volume 2. Lawrence Erlbaum.

Toben H Mintz. 2003. Frequent frames as a cue for grammatical categories in child directed speech. *Cognition*, 90(1):91 – 117.

Padraic Monaghan and Morten H Christiansen. 2008. Integration of multiple probabilistic cues in syntax acquisition. *Corpora in language acquisition research: History, methods, perspectives*, pages 139–164.

Christophe Parisse and M. Le Normand. 2000. Automatic disambiguation of morphosyntax in spoken language corpora. *Behavior Research Methods, Instruments, & Computers*, 32(3):468–481.

Martin Redington, Nick Chater, and Steven Finch. 1998. Distributional information: A powerful cue for acquiring syntactic categories. *COGNITIVE SCIENCE*, 22(4):425–469.

Patricia A Reeder, Elissa L Newport, and Richard N Aslin. 2013. From shared contexts to syntactic categories: The role of distributional information in learning linguistic form-classes. *Cognitive psychology*, 66(1):30–54.

Jacqueline Sachs. 1983. Talking about the there and then: The emergence of displaced reference in parent-child discourse. *Children's language*, 4.

Jenny R Saffran, Richard N Aslin, and Elissa L Newport. 1996. Statistical learning by 8-month-old infants. *Science*, 274(5294):1926–1928.

Kenji Sagae, Brian MacWhinney, and Alon Lavie. 2004. Automatic parsing of parental verbal input. *Behavior Research Methods, Instruments, & Computers*, 36(1):113–126.

Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, EACL '95, pages 141–148, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Michelle C St Clair, Padraic Monaghan, and Morten H Christiansen. 2010. Learning grammatical categories from distributional cues: flexible frames for language acquisition. *Cognition*, 116(3):341–60.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286, November.

Patrick Suppes. 1974. The semantics of children's language. *American psychologist*, 29(2):103.

Anna L Theakston, Elena VM Lieven, Julian M Pine, and Caroline F Rowland. 2001. The role of performance limitations in the acquisition of verb-argument structure: An alternative account. *Journal of child language*, 28(1):127–152.

Malathi Thothathiri, Jesse Snedeker, and Erin Hannon. 2012. The effect of prosody on distributional learning in 12-to 13-month-old infants. *Infant and Child Development*, 21(2):135–145.

Mehmet Ali Yatbaz, Enis Sert, and Deniz Yuret. 2012. Learning syntactic categories using paradigmatic representations of word context. In *EMNLP-CoNLL*, pages 940–951.

Understanding the Lexical Simplification Needs of Non-Native Speakers of English

Gustavo Henrique Paetzold and **Lucia Specia**
Department of Computer Science
University of Sheffield, UK
{g.h.paetzold,l.specia}@sheffield.ac.uk

Abstract

We report three user studies in which the Lexical Simplification needs of non-native English speakers are investigated. Our analyses feature valuable new insight on the relationship between the non-natives' notion of complexity and various morphological, semantic and lexical word properties. Some of our findings contradict long-standing misconceptions about word simplicity. The data produced in our studies consists of 211,564 annotations made by 1,100 volunteers, which we hope will guide forthcoming research on Text Simplification for non-native speakers of English.

1 Introduction

Text Simplification is a useful application both to improve other Natural Language Processing tasks and to assist language-impaired readers (Chandrasekar et al., 1996). When a simplifier aims to help people, understanding their needs becomes very important. In Lexical Simplification – the task of replacing complex words and expressions with simpler alternatives – this has been shown to be the case (Rello et al., 2013b; Rello et al., 2013a; Rello et al., 2013c). They describe several user studies conducted with readers suffering from Dyslexia and outline the most recurring challenges faced by them, as well as the most effective ways to overcome these challenges.

Given the widespread availability of content in English, non-native speakers of English become an important group to focus on. Reves and Medgyes (1994) elaborate on the differences between native and non-native English teachers from an educational and behavioural standpoint. However, we were not able to not find user studies that investigate the needs of non-native speakers from the perspective of Text Simplification. In this paper, we introduce three user studies that aim to do so.

Each user study pertains to one of three steps in the usual Lexical Simplification pipeline (Figure 1): Complex Word Identification, Substitution Selection and Substitution Ranking. In the sections that follow, we describe the findings of each user study.

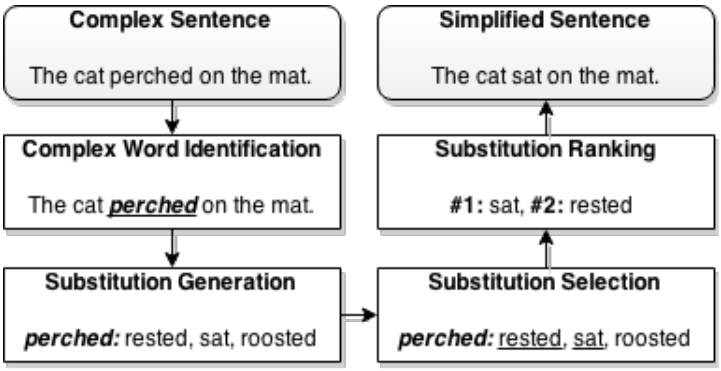


Figure 1: Common Lexical Simplification Pipeline

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 717–727, Osaka, Japan, December 11-17 2016.

2 Complex Word Identification

Complex Word Identification (CWI) is the task of deciding which words should be simplified in a text. Effective CWI strategies identify words which should not be simplified, and hence prevent LS systems from making inappropriate replacements (Paetzold, 2015). As shown in (Paetzold and Specia, 2013; Shardlow, 2014), ignoring CWI can considerably decrease the quality of the output produced by a simplifier. The goals of our user study on CWI are to:

- Provide a better understanding on features of words that challenge non-native English speakers, and
- Create a dataset that allows us to conceive models that automatically identify complex words.

2.1 Data Sources

We selected 9,200 sentences with 20-40 words in length, at random, from three sources:

- **CW Corpus** (Shardlow, 2013b): Composed of 731 sentences from the Simple English Wikipedia in which exactly one word has been simplified by editors from the standard English Wikipedia. 231 sentences that conformed to our criteria were extracted.
- **LexMTurk Corpus** (Horn et al., 2014): Composed of 500 sentences from the Simple English Wikipedia containing one word simplified from the standard English Wikipedia. 269 sentences were extracted.
- **Simple Wikipedia** (Kauchak, 2013): Composed of 167,689 sentences from the Simple English Wikipedia, each aligned to an equivalent sentence in the standard English Wikipedia. We selected a set of 8,700 sentences from the Simple Wikipedia version that were aligned to an identical sentence in Wikipedia.

2.2 Annotation Process

400 non-native English speakers participated in this study, all students and staff from various universities around the world. Volunteers provided information about their native language, age, education level and English proficiency level according to CEFR (Common European Framework of Reference for Languages). They were given a set of sentences and asked to judge whether or not they could understand the meaning of each content word and to annotate all words that they could not understand individually, even if they could comprehend the meaning of the sentence as a whole. The exact instructions given to annotators are as follows:

> *For each sentence, mark all the words you do not understand, even if you understand the sentence as a whole. If you understand all of them, just select the "I understand all words!" option.*

In order to offer some form of financial compensation for their work, all annotators who successfully completed the task were automatically included in a monetary prize draw (£50). This compensation method was used in all user studies described in this paper.

For agreement analysis purposes, 200 sentences were annotated by 20 volunteers each, while the remaining 9,000 sentences were annotated by only one volunteer (i.e. each volunteer annotating an average of 22 sentences from the 9,000).

3 Dataset Analysis

The resulting dataset contains 158,624 annotations. 3,854 distinct words (6,388 in total) were deemed complex by at least one annotator. In the following sections, we discuss details of the data collected.

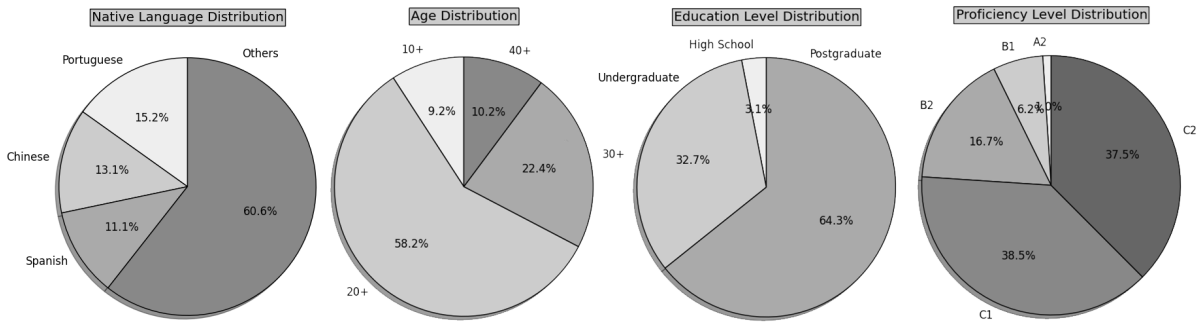
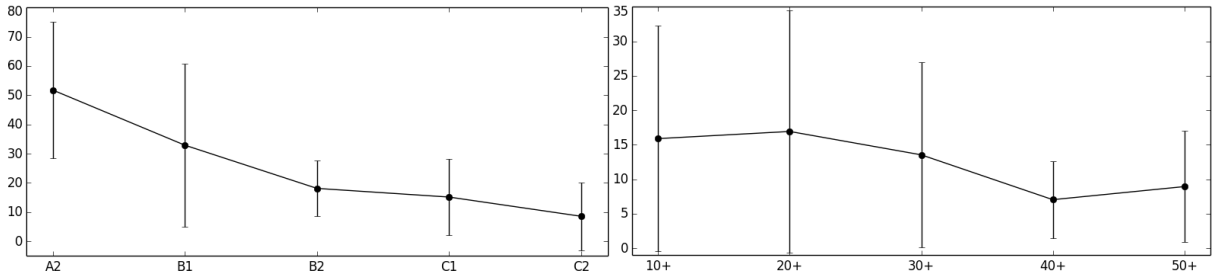


Figure 2: Annotators' backgrounds



(a) Proficiency level versus words deemed complex

(b) Age bands versus words deemed complex

Figure 3: Relationship between number of words deemed complex and the annotators' profiles

3.1 Profile of Annotators

Annotators spoke 45 different languages. The distributions with respect to native language, age, education and English proficiency levels are illustrated in Figure 2. As shown in Figures 3a and 3b, the data reveals interesting correlations between the number of complex words annotated and volunteers' age or English proficiency level.

Through F-tests, we found a significant difference ($p < 0.01$) between the band of 40+ years of age and the bands of 10+, 20+ and 30+ years of age. We also found significant differences between almost all English proficiency levels above A2, except between B2 and C1. Interestingly, B2 and C1 happen to have the same description in the London School Level Scale[1]: "*I speak and understand well but still make mistakes and fail to make myself understood occasionally*". We did not find significant differences among education levels.

3.2 Analysis of Data Sources

We found that the target words from the CW and LexMTurk datasets were deemed complex at least once by our annotators in only 51.9% and 40.8% of the instances, respectively. As for the remaining Simple Wikipedia instances, we discovered that at least one word in 27.3% of the instances was deemed complex by an annotator, which suggests that the simplified version of Wikipedia may still challenge non-native English speakers.

3.3 Features of Complex Words

We extracted and analysed 15 features that highlight the differences between simple words and those deemed complex by the annotators. We choose these features because they are the most widely used word complexity indicators in current work in Text Simplification. The features can be grouped in three types:

- **Morphological:** Word length and number of syllables, according to Morph Adorner (Burns, 2013).

[1] http://www.londonschool.com/level-scale

- **Semantic:** Number of senses, synonyms, hypernyms and hyponyms, according to WordNet (Fellbaum, 1998).

- **Lexical:** N-gram language model log-probabilities from the SubIMDB (Paetzold and Specia, 2016), Subtlex (Brysbaert and New, 2009) and Simple Wikipedia (Kauchak, 2013) corpora. We trained a specific language model using SRILM (Stolcke, 2002) from each of these corpora in order to estimate n-gram log-probabilities.

Table 1 shows the average feature values and standard deviations for all complex and simple words in the part of the dataset annotated by 20 volunteers. We define as complex any word which has been judged so by at least $n \in \{1, 5, 10\}$ annotators. The $[i, j]$ indicators present in the n-gram features of Table 1 refer to the number of tokens to the left (i) and right (j) of the words that was considered. Consequently, $[0, 0]$ refer to single-word frequencies. The column succeeding feature values indicate whether there was ($\bullet$) or not ($\circ$) a statistically significant difference between complex and simple words ($p < 0.01$), given the results of an F-test.

	$\mathbf{n=1}$			$\mathbf{n=5}$			$\mathbf{n=10}$		
Feature	**Complex**	**Simple**	p	**Complex**	**Simple**	p	**Complex**	**Simple**	p
Length	6.7 ± 2	6.6 ± 2	$\circ$	7.5 ± 2	5.9 ± 2	$\circ$	7.1 ± 2	6.1 ± 2	$\circ$
Syllables	2.1 ± 1	2.2 ± 1	$\circ$	2.3 ± 1	1.8 ± 1	$\circ$	2.2 ± 1	1.7 ± 1	$\circ$
Senses	6.6 ± 8	8.2 ± 8	$\bullet$	2.1 ± 2	9.1 ± 9	$\bullet$	1.1 ± 1	8.8 ± 9	$\bullet$
Synonyms	16.7 ± 21	20.0 ± 21	$\bullet$	5.3 ± 6	22.5 ± 23	$\bullet$	2.3 ± 3	22.7 ± 22	$\bullet$
Hypernyms	4.8 ± 6	5.5 ± 6	$\bullet$	1.7 ± 2	6.1 ± 8	$\bullet$	0.9 ± 1	5.9 ± 7	$\bullet$
Hyponyms	24.7 ± 48	32.3 ± 64	$\bullet$	4.0 ± 13	36.9 ± 52	$\bullet$	0.8 ± 2	32.8 ± 52	$\bullet$
Subimdb[0,0]	-5.3 ± 1	-4.8 ± 1	$\bullet$	-6.5 ± 1	-4.6 ± 1	$\bullet$	-6.6 ± 1	-4.5 ± 1	$\bullet$
Subtlex[0,0]	-10.4 ± 21	-5.0 ± 5	$\bullet$	-31.3 ± 41	-4.6 ± 1	$\bullet$	-51.3 ± 46	-4.4 ± 1	$\bullet$
Simple[0,0]	-5.9 ± 10	-4.3 ± 1	$\bullet$	-11.5 ± 22	-4.2 ± 1	$\bullet$	-8.4 ± 14	-4.2 ± 1	$\circ$
Subimdb[1,1]	-11.3 ± 3	-10.7 ± 3	$\bullet$	-12.9 ± 3	-11.0 ± 3	$\bullet$	-13.2 ± 3	-9.7 ± 3	$\bullet$
Subtlex[1,1]	-19.3 ± 28	-13.4 ± 18	$\bullet$	-40.0 ± 45	-16.4 ± 23	$\bullet$	-59.7 ± 52	-13.8 ± 21	$\bullet$
Simple[1,1]	-11.4 ± 18	-8.7 ± 9	$\bullet$	-16.7 ± 26	-7.9 ± 2	$\bullet$	-10.7 ± 15	-8.1 ± 2	$\circ$

Table 1: Average and standard deviation of features of words deemed complex or simple by at least n annotators. The $[i, j]$ indicators refer to the number of tokens to the left (i) and right (j) considered by n-grams. The p columns' values indicate the presence ($\bullet$) or not ($\circ$) of a statistically significant difference.

The results shed some light on word complexity for a non-native English speaker. They show that, unlike semantic and lexical features, length and number of syllables have little to do with complexity. Although it has been found that shorter words do promote understandability for readers suffering from Dyslexic (Rello et al., 2013b), our findings reveal that long words are not necessarily more difficult to understand for non-native English speakers.

When it comes to semantic properties, it can be noticed that, while both average and standard deviation values for complex words decrease as the number of complex judgements increase, the same does not happen for simple words. This phenomenon suggests a relationship between ambiguity and complexity, where complex words are more likely to be unambiguous. This finding is in line with those of (Shardlow, 2013a), who successfully modelled word complexity by exploring the hypothesis that complex words tend to be less ambiguous.

Interestingly, a somewhat similar relationship can be observed between lexical properties and word simplicity: while the n-gram log-probabilities of complex words are low on average and have high variance across all scenarios, the average log-probabilities of simple words are much higher, and they vary much less.

3.4 Agreement Analysis

Here we calculated Kappa's pairwise inter-annotator agreement coefficient (Carletta, 1996) for all pairs of annotators who were presented with sentences from the overlapping portion of the data. The values average to 0.616 ± 0.05, which is much higher than the agreement scores obtained in previous work in similar tasks. For example, the Lexical Substitution tasks of SemEval 2007 (McCarthy and Navigli,

2007) and 2010 (Mihalcea et al., 2010) obtain an agreement of 0.277 for their annotations, while the English Lexical Simplification task of 2012 obtains an agreement of 0.398 (Specia et al., 2012).

Perhaps more impressive is the agreement within certain classes of annotators. Although the agreement for annotators with the same proficiency level is lower (0.575 ± 0.07), the agreements within education levels and age bands are noticeably higher, reaching 0.638 ± 0.08 and 0.671 ± 0.08, respectively. The highest agreement is reached by annotators with the same native language: 0.718 ± 0.1. Inspecting the annotations, we found that the speakers of certain languages are sometimes challenged by words which, in most cases, are not considered complex by native speakers of any other languages. Table 2 illustrates the words with the highest percentage of variance (Brysbaert and New, 2009) between the number of times that they were deemed complex by the speakers of a specific native language, and the rest of the annotators.

Language	1	2	3	4	5	6
Arabic	fur	juvenile	apprenticed	city	serologic	link
Chinese	canton	inscribed	opium	referendum	thorax	contaminants
French	sewerage	subsequent	warships	escudo	dye	ridges
German	escape	strong	early	city	escudo	iconoclastic
Portuguese	rather	hurricane	undergo	southern	ruler	crude
Spanish	bailed	cryptanalysis	plaque	debris	demise	perm

Table 2: Words with highest percentage of complexity variance per native language. Indexes in the first row indicate the words' percentage of variance rank, from highest to lowest.

4 Substitution Selection

Substitution Selection (SS) is the task of deciding which candidate substitutions can replace a complex word in a given context. Its goal is to prevent a simplifier from performing replacements that compromise the sentence's grammaticality and/or meaning. The goals of this user study were to:

- Understand what makes a good candidate substitute for a complex word, and
- Create a dataset to build more effective substitution selectors.

Notice that we skip the step of Substitution Generation in our user studies. We do so because we believe that the extensive experiments of Horn et al. (2014) and De Belder and Moens (2012) already provide sufficient insight with respect to the relationship between Substitution Generation and the needs of non-native English speakers.

4.1 Data Sources

We first created a list of 1,471 complex words by filtering any numbers, names, colours and stop words from the ones obtained in the CWI study. We then produced an average of 50 candidate substitutions for each word by combining the output of all Substitution Generation systems in the LEXenstein framework (Paetzold and Specia, 2015). These systems exploit complex-to-simple parallel corpora (Horn et al., 2014), word embedding models (Glavaš and Štajner, 2015; Paetzold and Specia, 2016), WordNet (Devlin and Tait, 1998; Biran et al., 2011) and the Merriam Dictionary[2] (Kajiwara et al., 2013). Using the strategy described in (Paetzold and Specia, 2015), we selected the 10 candidates with the highest semantic similarity to each complex word. Using the Text Adorning module of LEXenstein, we ensured that all candidates have the same conjugation form as the complex word itself.

Finally, we extracted, according to availability, up to three sentences from Wikipedia in which each of these complex words appear (2,554 in total), and created 25,540 annotation instances by replacing the complex word in each sentence with one of the 10 candidate substitutions selected.

[2]http://www.merriam-webster.com

4.2 Annotation Process

400 fluent speakers of English participated in this study, all students and staff from different universities around the world. We recruited native English speakers for this task because it requires that the annotator understands the meaning of the complex word in question in order to make the necessary judgements.

Each annotator was presented with 80 annotation instances accompanied by the original complex word for reference. The exact instructions given to annotators are as follows:

> *Judge the following candidate substitutions of complex words with respect to their grammaticality and meaning preservation. When judging, please ignore any grammatical errors that are not caused by the substitution.*

For each instance, annotators were presented with two options:

- The substitution preserves the sentence's *grammaticality*, and
- The substitution preserves the original sentence's *meaning*.

Volunteers could select both, either or none of them. The resulting dataset contains 25,540 annotated instances. For agreement analysis purposes, 1,600 instances were annotated by 5 volunteers each, while the remaining 23,940 instances were annotated by a single volunteer. In total, 31,940 annotations were gathered. Notice that word simplicity is not taken into account in this user study, given that we wish to study how here readers interpret word replaceability only.

4.3 Dataset Analysis

We calculated several features to compare the grammatical and/or meaning preserving substitutions against the remaining substitutions. Table 3 illustrate the average and standard deviation feature values of candidates annotated positively by at least three out of five annotators (Good), and not (Bad), with respect to their grammaticality, meaning preservation, and both of them jointly. We chose six features, which can be grouped as:

- Language model probabilities of the sentence with the candidate in place of the target, given four 3-gram language models trained over the SubIMDB (Paetzold and Specia, 2016), Subtlex (Brysbaert and New, 2009) and Simple Wikipedia (Kauchak, 2013) corpora. Language model sentence probabilities have been used in previous work to create very effective Lexical Simplification systems (Horn et al., 2014; Glavaš and Štajner, 2015; Paetzold and Specia, 2016). Language models were trained with SRILM.

- The cosine word vector similarity between the candidate and the target (Target Sim.), as well as the average cosine similarity between the vector of the candidate and the vectors of content words in the sentence (Context Sim.). These features have been used in the creation of unsupervised lexical simplifier (Glavaš and Štajner, 2015). The embeddings model was trained with word2vec (Mikolov et al., 2013) with the CBOW architecture and 500 dimensions over a corpus of 7 billion words extracted from various sources (Paetzold, 2015; Brysbaert and New, 2009; Kauchak, 2013).

- The probability of the candidate receiving the same POS tag attributed to the target (POS Prob.). This feature has been shown a strong indicator of grammaticality (Aluisio and Gasperin, 2010; Nunes et al., 2013). The POS tag conditional probability models were trained over POS tags produced by the Stanford Parser (Klein and Manning, 2003) over the NewsCrawl corpus[3].

The column following the average values for Good and Bad candidates contain • for features for which we found a statistically significant difference between the averages through a F-test ($p < 0.01$), and ∘ for the remainder. The results suggest that, even though they are able to account for context, n-gram

[3]http://www.statmt.org/wmt11/translation-task.html

language model probabilities are much less effective in distinguishing good from bad candidates than the word vector distance between target and candidate words. Nonetheless, the same cannot be observed for the average similarity between candidate and context words.

Another interesting finding from our results that agree with previous contributions (Aluisio and Gasperin, 2010; Nunes et al., 2013) refers to the probability of the candidate receiving the POS tag of the target (POS Prob.), which does indeed show a strong relationship with grammaticality.

	Grammaticality			**Meaning**			**Joint (G/M)**		
Feature	**Good**	**Bad**	p	**Good**	**Bad**	p	**Good**	**Bad**	p
Prob. Subimdb	-0.9 ± 0.3	-1.0 ± 0.3	∘	-1.0 ± 0.3	-0.9 ± 0.3	∘	-0.9 ± 0.2	-1.0 ± 0.3	•
Prob. Subtlex	-3.1 ± 1.3	-3.2 ± 1.7	•	-3.2 ± 1.4	-3.2 ± 1.7	•	-3.1 ± 1.5	-3.4 ± 1.8	∘
Prob. Simple	-4.2 ± 1.6	-4.3 ± 1.9	•	-4.2 ± 1.8	-4.3 ± 1.9	•	-4.2 ± 1.7	-4.4 ± 2.0	∘
Target Sim.	0.41 ± 0.2	0.29 ± 0.2	•	0.39 ± 0.2	0.28 ± 0.2	•	0.34 ± 0.2	0.27 ± 0.2	•
Context Sim.	0.08 ± 0.1	0.06 ± 0.1	∘	0.08 ± 0.1	0.06 ± 0.1	∘	0.07 ± 0.1	0.06 ± 0.1	•
POS Prob.	0.62 ± 0.4	0.44 ± 0.4	•	0.53 ± 0.4	0.46 ± 0.4	∘	0.58 ± 0.4	0.32 ± 0.4	•

Table 3: Average and standard deviation of features of those words which were deemed grammatical, meaningful, or both by at least 3 annotators, and those that were not.

Out of the 356 candidates judged both grammatical and meaning preserving by at least three annotators, 171 (48%) are not listed in WordNet as either synonyms, hypernyms or hyponyms of the target word. This suggests that simplification strategies such as the ones of (Devlin and Tait, 1998) and (Biran et al., 2011), which extract candidate substitutions of complex words from WordNet, can suffer from low coverage.

4.4 Agreement Analysis

The average Kappa inter-annotator agreement scores for the data in this user study are 0.391 ± 0.16 for grammaticality, 0.424 ± 0.16 for meaning preservation, and 0.450 ± 0.16 for both of them jointly. Inspecting the data, we found that most disagreements resulted from situations in which the target word was part of a multi-word expression. Consider for example the target word *turn* in the sentence "*That in turn makes it difficult to affect policies to curb distracted driving*", which, in this case, is part of the multi-word expression *in turn*. Annotators were very much divided on whether or not candidate *reverse* preserved either grammaticality or meaning in this case: some judged it to be neither grammatical nor meaningful, while others claimed it to be grammatical or meaningful.

5 Substitution Ranking

In Substitution Ranking (SR), candidates are ranked according to their simplicity so that the complex word is replaced with the simplest candidate available. The goals of our user study are to:

- Discover which metrics best capture simplicity for non-native speakers, and
- Create a dataset for the evaluation and training of SR strategies.

5.1 Data Sources

We extracted 901 sentences from those collected in our Substitution Selection user study which had a minimum of two and maximum of four candidates annotated as both grammatical and meaning preserving by at least three annotators. In order to access the simplicity of these substitutes, we added the target complex word of each sentence to the set of candidates, and then replaced the target word in each sentence with a gap marker. Finally, we created an annotation instance for each pair of candidates, totalling 4,200 pairs (438 sentences * 3 pairs (3 candidates including complex word itself) + 436 * 6 pairs (4 candidates including complex word) + 27 * 10 pairs (5 candidates including complex word)).

5.2 Annotation Process

300 non-native English speakers participated in this study, all students and staff from different universities around the world. Volunteers provided anonymous information about their native language, age,

education level and English proficiency level. Each volunteer was presented with 70 annotation instances, each composed of a sentence with a gap and two candidates to fill it with.

For each instance, volunteers were asked to judge which candidate made the sentence easier to understand. They could also indicate that both candidates made the sentence equivalently complex/simple. The exact instructions given to annotators are as follows:

For each of the following instances, select which candidate makes the sentence easier to understand. If the words are equally complex/simple, select the "The words are equally simple" option. Please overlook any grammatical or spelling errors.

All instances were annotated by 5 volunteers. A total of 21,000 annotations were produced.

Once instances were annotated, we used the algorithm introduced by (Wauthier et al., 2013) to infer rankings from binary comparisons, and hence produce 901 instances composed of a sentence, a target complex word, and a set of candidate substitutions ranked according to their simplicity.

5.3 Dataset Analysis

For validation purposes, we computed the correlation between the simplicity rankings and the same 15 features used in the dataset analysis for our Complex Word Identification user study, described in Section 3.2. Notice that in our work we measure simplicity as the ease with which one can understand a given portion of text, which is the opposite of the definition of complexity used in our study on CWI. We use three evaluation metrics: Spearman correlation (r), Pearson correlation (ρ) and TRank. The TRank metric was introduced by (Specia et al., 2012) and measures the proportion of times in which the word ranked simplest by a given feature is also ranked simplest by the annotators.

The values in Table 4 reveal that word length and number of syllables correlate poorly with word complexity, while simpler words tend to be more ambiguous and occur more frequently in corpora. These findings reinforce the ones from our CWI user study. More importantly, our results show that correlation and TRank scores of [1,1] (one token to the left and right) n-grams consistently outperform the scores of single-word frequencies ([0,0]) according to all metrics used. These findings contradict a long-standing assumption that context is not an important factor in word simplicity estimation (Devlin and Tait, 1998; Carroll et al., 1999; Biran et al., 2011; Rello et al., 2013b; Shardlow, 2013a).

Feature	r	ρ	**TRank**
Length	0.172	0.179	0.386
Syllables	0.097	0.095	0.340
Senses	−0.345	−0.349	0.505
Synonyms	−0.288	−0.297	0.454
Hypernyms	−0.289	−0.297	0.472
Hyponyms	−0.309	−0.300	0.453
Subimdb[0,0]	−0.419	−0.436	0.539
Subtlex[0,0]	−0.465	−0.467	0.556
Simple[0,0]	−0.490	−0.468	0.578
Subimdb[1,1]	−0.463	−0.473	0.579
Subtlex[1,1]	−0.496	−0.496	0.590
Simple[1,1]	**−0.501**	**−0.475**	**0.593**

Table 4: Simplicity correlation analysis between features and annotations

5.4 Agreement Analysis

The average Kappa inter-annotator agreement scores for this user study resemble the ones reported in Section 3.4: although the agreement between all annotators is encouraging (0.454 ± 0.05), the scores are even higher for annotators with similar backgrounds. Annotators within the same education level,

age band and proficiency level reach agreement scores of 0.468 ± 0.01, 0.482 ± 0.02 and 0.486 ± 0.01, respectively. Like what was observed in our user study on Complex Word Identification, the highest agreement comes from annotators that speak the same native language (0.601 ± 0.15). This serves as further evidence that one's native language plays an important role on vocabulary acquisition.

6 Conclusions

We have described three user studies conducted with the goal of understanding the simplification needs of non-native speakers of English.

In our Complex Word Identification study we learned that words which are simpler to non-native English speakers have much higher probabilities according to language models, both alone and in context, while those which are more complex to them tend to have a smaller number of senses in WordNet. In contrast with what was reported by (Rello et al., 2013b) in experiments with readers who suffer from Dyslexia, we found no evidence of a relationship between the non-natives' perception of complexity and neither word length or number of syllables. Our experiments also showed that while a reader's English proficiency level indicates how many words will pose a challenge to them, their native language indicates which words these will be.

In our Substitution Selection study we found that, despite disregarding contextual information altogether, word vector distances between a complex word and a candidate substitution are more reliable than language model probabilities in capturing both grammaticality and meaning preservation. We also found that the conditional probability of a candidate with respect to the grammatical role of the complex word is a reliable indicator of grammaticality. From our agreement analysis, we found evidence that single-word replacements tend to compromise the understanding of multi-word expressions.

In our Substitution Ranking study, we found further evidence that, unlike ambiguity indicators and language model probabilities, length and number of syllables have little to do with word simplicity for non-native speakers of English. N-gram probabilities proved the most reliable simplicity indicators among the features evaluated, which contradicts the assumption often made in earlier work that context offers no important clues on a word's simplicity.

Table 5 summarises the data produced from each of these studies. Some examples of models and applications that could be built from these datasets are readability assessment tools, semantic analysers, text profilers and full lexical simplifiers. All of the datasets described herein can be downloaded from `http://ghpaetzold.github.io/data/User_Studies_NNS.zip`.

User Study	Sentences	Words	Word Pairs	Annotators	Annotations
Complex Word Identification	9,200	87,244	-	400	158,624
Substitution Selection	2,554	25,540	-	400	31,940
Substitution Ranking	901	3,193	4,200	300	21,000
Total	**12,655**	**115,977**	**4,200**	**1,100**	**211,564**

Table 5: Summary of annotated data produced: number of unique sentences, words and word pairs annotated, as well as the number of annotators who participated and annotations produced.

Acknowledgements

This work has been partially supported by the European Commission project SIMPATICO (H2020-EURO-6-2015, grant number 692819).

References

Sandra Aluisio and Caroline Gasperin. 2010. Fostering digital inclusion and accessibility: The porsimples project for simplification of portuguese texts. In *Proceedings of the 2010 NAACL Young Investigators Workshop on Computational Approaches to Languages of the Americas*, pages 46–53.

Or Biran, Samuel Brody, and Noémie Elhadad. 2011. Putting it simply: a context-aware approach to lexical simplification. In *Proceedings of the 49th ACL*, pages 496–501.

Marc Brysbaert and Boris New. 2009. Moving beyond kučera and francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english. *Behavior research methods*, 41:977–990.

Philip R Burns. 2013. Morphadorner v2: A java library for the morphological adornment of english language texts. *Northwestern University, Evanston, IL.*

Jean Carletta. 1996. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22:249–254.

John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying text for language-impaired readers. In *Proceedings of the 9th EACL*, pages 269–270.

Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the 16th Conference on Computational Linguistics*.

Jan De Belder and Marie-Francine Moens. 2012. A dataset for the evaluation of lexical simplification. In *Computational Linguistics and Intelligent Text Processing*, pages 426–437. Springer.

Siobhan Devlin and John Tait. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic Databases*, pages 161–173.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora? In *Proceedings of the 53rd ACL*, pages 63–69.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using wikipedia. In *Proceedings of the 52nd ACL*, pages 458–463.

Tomoyuki Kajiwara, Hiroshi Matsumoto, and Kazuhide Yamamoto. 2013. Selecting proper lexical paraphrase for children. In *Proceedings of the 25th Rocling*, pages 59–73.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st ACL*, pages 1537–1546.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st ACL*, pages 423–430.

Diana McCarthy and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the 4th SemEval*, pages 48–53.

Rada Mihalcea, Ravi Sinha, and Diana McCarthy. 2010. Semeval-2010 task 2: Cross-lingual lexical substitution. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 9–14.

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

Bernardo Pereira Nunes, Ricardo Kawase, Patrick Siehndel, Marco a. Casanova, and Stefan Dietze. 2013. As simple as it gets - a sentence simplifier for different learning levels and contexts. In *Proceedings of the 13th ICALT*, pages 128–132.

Gustavo H. Paetzold and Lucia Specia. 2013. Text simplification as tree transduction. In *Proceedings of the 9th STIL*, pages 116–125.

Gustavo Henrique Paetzold and Lucia Specia. 2015. Lexenstein: A framework for lexical simplification. In *Proceedings of The 53rd ACL*, pages 85–90.

Gustavo Henrique Paetzold and Lucia Specia. 2016. Unsupervised lexical simplification for non-native speakers. In *Proceedings of The 30th AAAI*.

Gustavo Henrique Paetzold. 2015. Reliable lexical simplification for non-native speakers. In *Proceedings of the 2015 NAACL Student Research Workshop*, pages 9–16.

Luz Rello, Ricardo Baeza-Yates, Stefan Bott, and Horacio Saggion. 2013a. Simplify or help?: text simplification strategies for people with dyslexia. In *Proceedings of the 10th W4A*, pages 1–10.

Luz Rello, Ricardo Baeza-Yates, Laura Dempere-Marco, and Horacio Saggion. 2013b. Frequent words improve readability and short words improve understandability for people with dyslexia. *Human-Computer Interaction*, pages 203–219.

Luz Rello, Susana Bautista, Ricardo Baeza-Yates, Pablo Gervás, Raquel Hervás, and Horacio Saggion. 2013c. One half or 50%? an eye-tracking study of number representation readability. *Human-Computer Interaction*.

Thea Reves and Peter Medgyes. 1994. The non-native english speaking efl/esl teacher's self-image: An international survey. *System*, 22(3):353–367.

Matthew Shardlow. 2013a. A comparison of techniques to automatically identify complex words. In *Proceedings of the 51st ACL Student Research Workshop*, pages 103–109.

Matthew Shardlow. 2013b. The cw corpus: A new resource for evaluating the identification of complex words. In *Proceedings of the 2nd Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pages 69–77.

Matthew Shardlow. 2014. Out in the open: Finding and categorising errors in the lexical simplification pipeline. In *Proceedings of the 9th LREC*, pages 1583–1590.

Lucia Specia, Sujay Kumar Jauhar, and Rada Mihalcea. 2012. Semeval-2012 task 1: English lexical simplification. In *Proceedings of the 1st SemEval*, pages 347–355.

Andreas Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Proceedings of the 2002 ICSLP*, pages 257–286.

Fabian Wauthier, Michael Jordan, and Nebojsa Jojic. 2013. Efficient ranking from pairwise comparisons. In *Proceedings of the 30th International Conference on Machine Learning*, pages 109–117.

How Interlocutors Coordinate with each other within Emotional Segments?

Firoj Alam, Shammur Absar Chowdhury, Morena Danieli, Giuseppe Riccardi
Department of Information Engineering and Computer Science,
University of Trento, Italy
{firoj.alam, shammur.chowdhury, morena.danieli, giuseppe.riccardi}@unitn.it

Abstract

In this paper, we aim to investigate the coordination of interlocutors behavior in different emotional segments. Conversational coordination between the interlocutors is the tendency of speakers to predict and adjust each other accordingly on an ongoing conversation. In order to find such a coordination, we investigated 1) lexical similarities between the speakers in each emotional segments, 2) correlation between the interlocutors using psycholinguistic features, such as linguistic styles, psychological process, personal concerns among others, and 3) relation of interlocutors turn-taking behaviors such as competitiveness. To study the degree of coordination in different emotional segments, we conducted our experiments using real dyadic conversations collected from call centers in which agent's emotional state include *empathy* and customer's emotional states include *anger* and *frustration*. Our findings suggest that the most coordination occurs between the interlocutors inside anger segments, where as, a little coordination was observed when the agent was empathic, even though an increase in the amount of non-competitive overlaps was observed. We found no significant difference between anger and frustration segment in terms of turn-taking behaviors. However, the length of pause significantly decreases in the preceding segment of anger where as it increases in the preceding segment of frustration.

1 Introduction

Behavioral and social signal processing are emerging interdisciplinary areas of research, which combine social science, psychology, and computer science. The aim of the research is to design computational models for processing human behavioral aspects, which can facilitate different domain experts while counseling, consulting and (or) providing services (Narayanan and Georgiou, 2013; Vinciarelli et al., 2009; Pantic et al., 2011; Vinciarelli et al., 2012; Stepanov et al., 2015). The idea is to analyze different overt and covert behavioral signals during social interactions and label them with some short and long term functional aspects (i.e., states and traits) in order to quantitatively measure them. The functional aspects include empathy, politeness, agreement, engagement, uncertainty, competitiveness and other typical, atypical, distressed and affective social behaviors. Using these short and long term states and traits, one can design an informative behavioral profile of an individual from the daily-life interactions. The measured behavioral profile can help to predict the next behavioral outcome/consequence and/or actions of an individual. This kind of behavioral profile can help domain experts in different application scenarios such as call center, health-care and teacher-student interactions.

In the field of social and psychological science, researchers have been trying to understand these functional aspects for a very long time, however, very recently there are attempts to design automatic computational models for real-world applications. Designing such automatic systems for measuring these behavioral and social functional aspects is still infancy due to many different challenges.

One of the important challenges is to understand how different behavioral cues are associated with one another and how we express them in different interaction scenarios. In this study, *we investigated, the coordination of interlocutors behavior in different emotional segments and how conversational turn-taking*

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 728–738, Osaka, Japan, December 11-17 2016.

dynamics are associated with emotional manifestations of the agent and customer. For the study, the conversational coordination between the interlocutors is defined as the tendency of speakers to predict and adjust each other accordingly on an ongoing conversation. We explored the coordination in terms of psycholinguistic features, lexical and turn-taking features using correlation analysis, cosine similarity, and regression analysis, respectively. For this study, we analyzed dyadic human-human spoken conversations, collected from the call centers in the domain of after-sale customer care, which has been annotated with turn-taking dynamics and emotional expressions. The turn-taking dynamics include competitiveness of overlaps, pauses, and lapses among others. Emotional expressions has been annotated for agent and customer separately with agent's emotional state include *empathy*, and customer's emotions include *anger* and *frustration*.

It has been a few decades to the study of automatically recognizing emotion in affective computing, which has been done in the lab as well as in real settings. The study includes classifying Ekman's six basic categorical emotions (Ekman, 1999) or dimensional levels of emotion such as valence and arousal (Russell, 1980). Still, there are challenges to make emotion recognition research in its practical use, which includes lack of publicly available realistic databases, issues of fusing multi-modal information, automatic segmentation, robustness in terms of generalizability across the domain, cross-corpus (Zeng et al., 2009; Schuller et al., 2011). A detailed overview of emotion recognition research in terms of theories, computation models, and relevant applications is provided in (Calvo and D'Mello, 2010).

The study of turn-taking dynamics such as speech overlap has also a long history. One of the first studies on speech overlap, as discussed in (Sacks et al., 1974), suggested that turn changes with overlap is a very rare case and occurs as a result of self-selection, which projects turn endings. Where as a recent study of (Heldner and Edlund, 2010) suggests that overlap is, in fact, a frequent phenomenon and is much more than just a turn-taking signal, which has also been discussed in (Chowdhury et al., 2015b).

There has been a very few study, which explores finding how different turn-taking features are associated with emotional states. The association of turn-management labels, such as grab, accept, back-channel, and emotional states have been studied in (Koutsombogera et al., 2015). The importance of turn-taking information for predicting user-satisfaction in terms of user manifested emotion have been studied in (Chowdhury et al., 2016). They discussed that turn-taking cues significantly helps in the automatic prediction of user-satisfaction. To the best of our knowledge, a very little study have been conducted to examine what actually happens within an emotional segment in terms of turn-taking. In our study, we present a call center conversation corpus (in Section 2) in which we have the manual annotation of emotional states and overlap discourse. Using which we explored the coordination of interlocutors behaviors as our preliminary study, presented in Section 3 and 4, which can shade a light in future for designing automated computational model.

2 Corpus and Annotation

2.1 Corpus Description

The data used for our research is a collection of Italian human-human spoken conversations, sampled from a large set of call center conversations providing after-sales customer care support in the energy sector. We randomly selected these conversations over six months, which were recorded on two separate channels at a sample rate of $8kHz$, 16bits. The average duration of these conversations was 406 seconds. The corpus has been annotated with emotional states such as empathy, anger and frustration, and overlap-discourse such as competitive and non-competitive.

2.2 Annotation of Emotional States

As mentioned earlier, we annotated *empathy* on the agent channel, and *anger* and *frustration* on the customer channel. In the literature, there is a lack of operational definition of empathy. Therefore, we adopted the *modal model* of emotion by Gross (1998) in order to define empathy and design annotation guidelines for the annotators. Gross's modal model is based on appraisal theory, which has been studied by many psychologists for the investigation of emotional states. Appraisal models of emotion suggest

that organisms appraise (i.e., evaluate, interpret, explain) events/situations based on the appraisal process in order to determine the nature of ensuing emotion as discussed by Scherer (2000).

According to the *modal model*, *"emotions involve person-situation transections that compel attention, have meaning to an individual in light of currently active goals, and give rise to coordinated yet flexible multisystem responses that modify the ongoing person-situation transection in crucial ways"* (Gross and Thompson, 2007; Gross, 2011). The key idea of the modal model is that emotional states unfold over time, and their response may change the environmental stimuli, and that may alter the subsequent instances of that and other emotional states. It is a useful framework for describing the dynamics of emotional states, which manifests over time, leads to the generation of an emotional sequence from the interlocutors' emotional manifestations. For example, the sequence of emotional states between an agent and a customer could be Frustration (C) → Empathy (A) → Satisfaction (C). A for the agent and C for the customer.

To design the annotation guideline, we have done an extensive analysis of one hundred conversations (more than 11 hours), and selected dialog turns where the speech signal showed the emergence of empathy, basic emotion, such as anger, and complex emotion such as frustration. In our qualitative analysis, we investigated the relevant emotional speech segments, which were often characterized by some perceivable variation in the speech signal. We observed that such variations could co-occur with emotionally connoted words, but also with functional parts of speech, such as adverbs and interjections, which could play the role of lexical supports for the variations in emotional states. We hypothesized that perceivable variations in the speech are a possible signal of an appraisal process. On the basis of those observations, we have designed annotation guidelines whose critical principle was to focus annotators' attention on their own perception of the variations in the speech signal as well as the variations in the linguistic content of the utterances. For example the annotation guidelines include the following recommendations for the annotators: 1) annotating the onset of the signal variations that supports the perception of the manifestation of emotions, 2) identifying the speech segments preceding and following the onset position, and 3) annotating the context (left of the onset) and target (right of the onset) segments with a label of an emotional state (e.g., frustration, empathy, etc.). In addition, the annotation guidelines include operational definitions of emotional states related to the given domain of application. For example, in this annotation task, the operational definition of empathy is defined as "an emotional state triggered by another's emotional state or situation, in which one feels what the other feels or would normally be expected to feel in his situation" (Hoffman, 2008).

The annotation task was performed by two expert annotators who worked on non-transcribed spoken conversations by following the annotation scheme reported above. In this task, the annotation unit is the speech segment. They annotated *Empathy* on the agent channel and *Frustration* and *Anger* on the customer channel. The annotators labeled *Neutral* on the segment that appeared before any emotional segment to define the context, as mentioned earlier. Finally, the annotated corpus includes 1894 customer-agent conversations (210 hours and 23 minutes in total). In order to evaluate the reliability of the annotation we measured inter-annotator agreement on the annotated segments, and obtained an average kappa 0.74. More details can be found in (Danieli et al., 2014; Danieli et al., 2015).

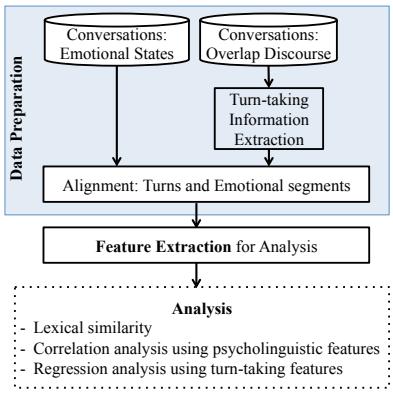
2.3 Annotation of Overlap Discourse

For the annotation of overlap discourse, we selected a subset of 565 conversations with approximately 62 hours of spoken content. Annotators manually segmented overlapping speech, then, categorized and labeled them with the competitive and non-competitive acts. The annotations were performed by two Italian native expert annotators by following the guideline described in (Chowdhury et al., 2015a). The guideline includes Competitive (Cmp) scenarios, in which the intervening speaker (overlapper) starts prior to the completion of the current speaker (overlappee), and both the speakers display interest in the turn for themselves, and also the speakers perceive the overlap as problematic. As for Non-Competitive (Ncm) scenarios, the overlapper starts in the middle of an ongoing turn. No evidence is shown by both the speakers to grab the turn for themselves. The overlapper used the overlap to signal the support for the current speaker's continuation of speech. Both of the speakers perceive the overlap as non-problematic.

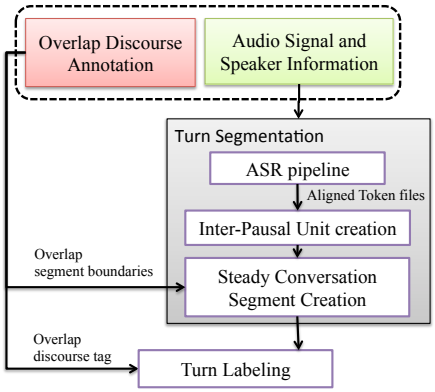
The inter-annotator agreement of the annotations is 0.70, which was measured using kappa statistics.

3 Methodology

In Figure 1, we present the experimental system of our study. In the data preparation phase, we selected a subset of conversations in which we have annotations of emotional states and overlap discourse. The turn-taking information extraction system utilized an Automatic Speech Recognition (ASR) system (Chowdhury et al., 2014) to create turn segments and extract turn information (see Section 3.1.1). Later, this information was aligned with the annotations of emotional segments to find the turn-taking information (more details can be found in Section 3.1.2). Using the aligned turn-taking information for an emotional segment, we extracted turn-taking features. We also used turn information to obtain lexical and psycholinguistic features per speaker from the segment. In the analysis phase of our experiment, we investigated lexical similarities and correlation of psycholinguistic features between speakers for different emotional segments. We also used multilevel logistic regression method to understand the association between turn-taking features and emotional segments, and how the association differs from one emotion to another.



(a) Experimental pipeline of this study.



(b) Turn-taking information extraction system.

Figure 1: System diagrams.

3.1 Data Preparation

For the analytical study, we selected a set of 523 conversations with the manual annotation of emotional states and overlap discourse. This set includes 310 conversations with emotional segments. Among the emotional segments around 11.28% of emotion are annotated as anger, 26.11% as frustration and 62.61% of annotated emotion in agent channel has empathy. From the rest of the 213 conversations, containing no emotional annotations, we selected segments and labeled them with no-emotion (NoEmo).

During the data preparation, we faced two important problems in order to define and align the emotional segment in association with turn-taking discourse: 1) emotional segment are very short in length, which made the task very difficult to get sufficient turn information, 2) an speaker respond to other speaker's emotion with a latency. To overcome these problems, we re-defined the following boundary of manual emotion segment with an impact window of length $2 * d$, where d is the length of the manual annotation of the emotional segment. Hence, the length of our emotional segment is $d+2*d = 3*d$. We also investigated preceding context of each customer's emotional segment and defined it as $Pre.Emo$ with a window of length $3 * d$. The $NoEmo$ segments have been selected from conversations where no emotion in both agent and customer side has been annotated. From the middle of each conversation, we selected and extracted two $NoEmo$ segments with a length of the average emotional segment, ($\approx$ 42 sec). We extracted the $NoEmo$ segments from both agent and customer channels. As mentioned earlier, empathy, Emp, has been annotated in the agent channel only. Thus the preceding context of agent's emotional segment is defined as $Pre.Emp$. Hence, the investigated emotional and non-emotional segments include Pre.Emp, Emp, Ang, Fru, Pre.Ang, Pre.Fru and NoEmo.

3.1.1 Turn-taking Information Extraction

The Turn-Taking Information Extraction System, described in Figure 1 (b), consists of a *turn segmentation and labeling system*. The system uses lexical and manual overlap discourse annotation information to segment and labels the turn types. The pipeline uses the time aligned ASR output as tokens to create Inter-Pausal Units (IPUs) for each input channel. IPUs are defined as the consecutive tokens with no less that 50 ms gaps in between. Using the start and end time information of inter-IPUs and intra-IPUs, we created a steady time line and binary representation (presence or absence of speech information) segments for both the channels. We then defined these segments as *steady conversation segments*. The labels of each segment were then defined by a set of rules. Labels of the segments are as follows:

- Turn (T): Maximal sequences of IPUs where one single speaker has the floor, and none of the IPUs from the interlocutor are present (Beňuš et al., 2011). T_A and T_C represent agent and customer's turns respectively.
- Pause (P): Gaps between the turns of the same speaker with no less than 0.5 sec. P_A and P_C represent agent and customer's pauses respectively.
- Overlap Types Ov= $\{Cmp, Ncm\}$: Overlapping turns between the two interlocutors with competitive or non-competitive intention (see section 2.3 for details).
- Lapse between speakers (L_B): Floor switches between the speakers with a silence duration of 2 sec or more.
- Lapse within speaker (L_W): Gaps between a speakers' turns with a silence duration of 2 sec or more.
- Switch (S): Floor switches between the speakers with silence less than 2 secs or with overlapping frames, not more than 20 ms.

3.1.2 Alignment: Turns and Emotional Segments

For the turn level analysis, it is important to align the turn sequences with the boundary of emotional segments. It is evident from manual annotation that an emotional segment consist of different turn types and not all the turns start inside the boundary. There are some cases where the start/end of emotional episode can be at the middle of a turn. We solved this problem using a rule-based approach. For example, if half of a mismatched turn fall inside an emotional segment we considered that as a part of emotional segment.

3.2 Feature Extraction

3.2.1 Lexical Features

We extracted lexical features from automatic transcriptions from an in-house developed Automatic Speech Recognition (ASR) System (Chowdhury et al., 2014). The word error rate of the system is 31.78% on the test set. To understand the utility of the automated transcriptions with such as error rate, in a different study we compared the performance between automatic and manual transcriptions for a automatic classification of emotions. The results show that performance differences are very low, only 1.2% drop with automated transcriptions (Alam et al., 2016). Therefore, we found that the use of automatic transcriptions are reasonable for the experiment given that manual transcriptions are not available in call cases. For the experiments, the transcriptions of each segment were converted into bag-of-words vectors weighted with logarithmic term frequencies (tf) multiplied with inverse document frequencies (idf). We also reduced the size of the dictionary by removing stop-words and lower frequent words.

3.2.2 Psycholinguistic Features

Psycholinguistic features were extracted from the transcriptions, using Linguistic Inquiry Word Count (LIWC) (Pennebaker et al., 2001). It has been used to study personality, the role of speakers in overlaps (Alam and Riccardi, 2014; Chowdhury et al., 2015b) among other social behaviors in order to understand the correlation between these attributes and word uses. The feature category includes linguistic (e.g., preposition, verb, word count), psychological (affect, positive, negative emotion, anxiety), personal concern (e.g., work, home, money), swear words, relativity among others. The LIWC is a knowledge-based system, which was designed using a set of dictionaries for different languages including Italian.

In the dictionary, each word was labeled with feature categories mentioned above. During the feature extraction process the word in the transcriptions was matched with the dictionary. Then, the matched category was computed as frequency or relative frequency. The Italian version of the dictionary contains 85 word categories (Alparone et al., 2004). We also extracted 5 general and 12 punctuation categories constituting a total of 102 features. We then removed LIWC features that are not observed in our training dataset.

3.2.3 Turn-Taking Features

The turn-taking features were generated using the turn sequence output of the Turn-Taking Information Extraction System, described in Section 3.1.1. The sequences were first aligned with each corresponding emotional segment (see Section 3.1.2). To understand the impact of the choice of turn-taking behavior, we divided the feature sets, at both segment and individual speaker levels, into two groups. A brief description of extracted features, in the segment, are as follows:

- General information about emotional segment (G1):
  - Participation equality, $P_{eq} = 1 - (\frac{\sum_i^N (T_i - T)^2 / T}{E})$ where T is the average speech duration of the speakers. T_i is the total speech duration for each speaker. E represents the total speech duration. $N = 2$, represents two speakers as agent and customer inside the emotional segment.
  - Percentage of overlaps.
  - Percentage of Cmp and Ncm on total overlap duration.
- Length of different turn types (G2):
  - Median duration of T_A, T_C, P_A, P_C, Cmp, Ncm, L_W and L_B, inside emotional segment normalized by the median of speaker's respective turn in the whole conversation.

4 Analysis and Results

For different feature sets, we investigated different experimental configurations. For the study of lexical similarities, our experimental conditions include: 1) lexical features from paired (i.e., agent and customer channel from same conversation) speakers' non-overlapping *vs* overlapping turns, 2) lexical features from non-paired (i.e., agent and customer channel extracted from unrelated conversation) speakers' non-overlapping *vs* overlapping turns. Where as for psycholinguistic features, we investigated features obtained from non-overlapping *vs* overlapping turns. For turn-taking features, we have not made any such distinctions. The non-overlapping turns include all the turns of the speakers excluding the overlaps. Where as the overlapping turns includes competitive (Cmp) and non-competitive (Ncm) overlaps.

4.1 Lexical Similarities

For the analysis, we computed cosine similarity of the agent and customer aligned segment representing different emotional states. For the lexical similarity we designed feature vector for agent $\overrightarrow{V_{S_A}}$ and customer $\overrightarrow{V_{S_C}}$ emotional segment using bag-of-word model and transformed them into tf-idf. Then, we computed cosine similarity, $sim(S_A, S_C) = \frac{\overrightarrow{V_{S_A}} \cdot \overrightarrow{V_{S_C}}}{\left|\overrightarrow{V_{S_A}}\right| \cdot \left|\overrightarrow{V_{S_C}}\right|}$ between the feature vector of the agent and customer's segment. For a pair-wise comparison of emotional states, then, we computed mean and standard deviation with statistical significance using t-test.

As mentioned earlier, we have four different experimental configurations for the analysis of lexical similarities. As a baseline, we computed the similarities between non-paired speakers using the lexical features from non-overlapping turns for different emotional segments. The results are presented in a form of similarity map in Figure 2. From the results, we observed that the interlocutors entrain each other in non-overlapping turns when the customer is expressing anger, and the value of similarity ($sim = 0.181$) is significantly ($p < 0.05$) higher than the similarities in any other emotional segment.

In the experiment with competitive overlapping turns, we observed the highest similarity of 0.035 and 0.031 in preceding-anger and anger segments, respectively. In the case of non-competitive overlapping

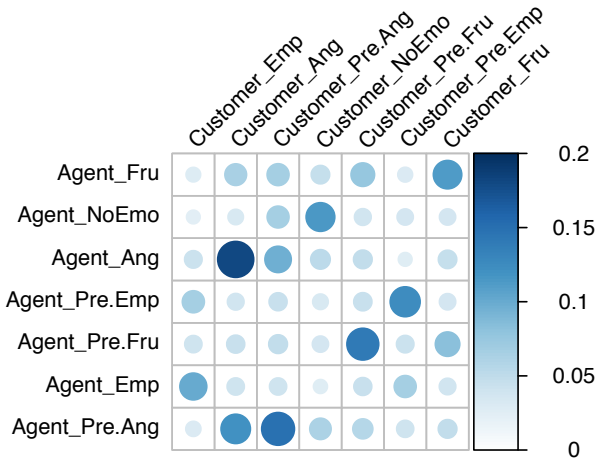


Figure 2: Lexical similarity between the emotional segment of the agent and the customer channel. Pre. represents preceding segments. Ang - anger, Fru - frustration, Emp - empathy, NoEmo - no-emotion

turns, a similarity of 0.034 was observed between the interlocutors in frustration segments. The results on overlapping turns are insignificant.

4.2 Psycholinguistic Features

We explored the degree of coordination using Pearson correlation coefficient (r) between the interlocutors' behaviors by correlating psycholinguistic features obtained from overlapping and non-overlapping turns, presented in Figure 3. For the sake of simplicity, the magnitude of r values are presented using colors where as '✕' symbol represent the corresponding r is not significant. These analyses are based on entire emotion segments from the agent and customer channels, irrespective of turns. The r is calculated for each psycholinguistic feature by correlating the agent and customer feature vectors of the conversations. We calculated the significance of the correlation coefficient r using t-test with a degree of freedom equal to $n - 2$, where n represent the total number of instances.

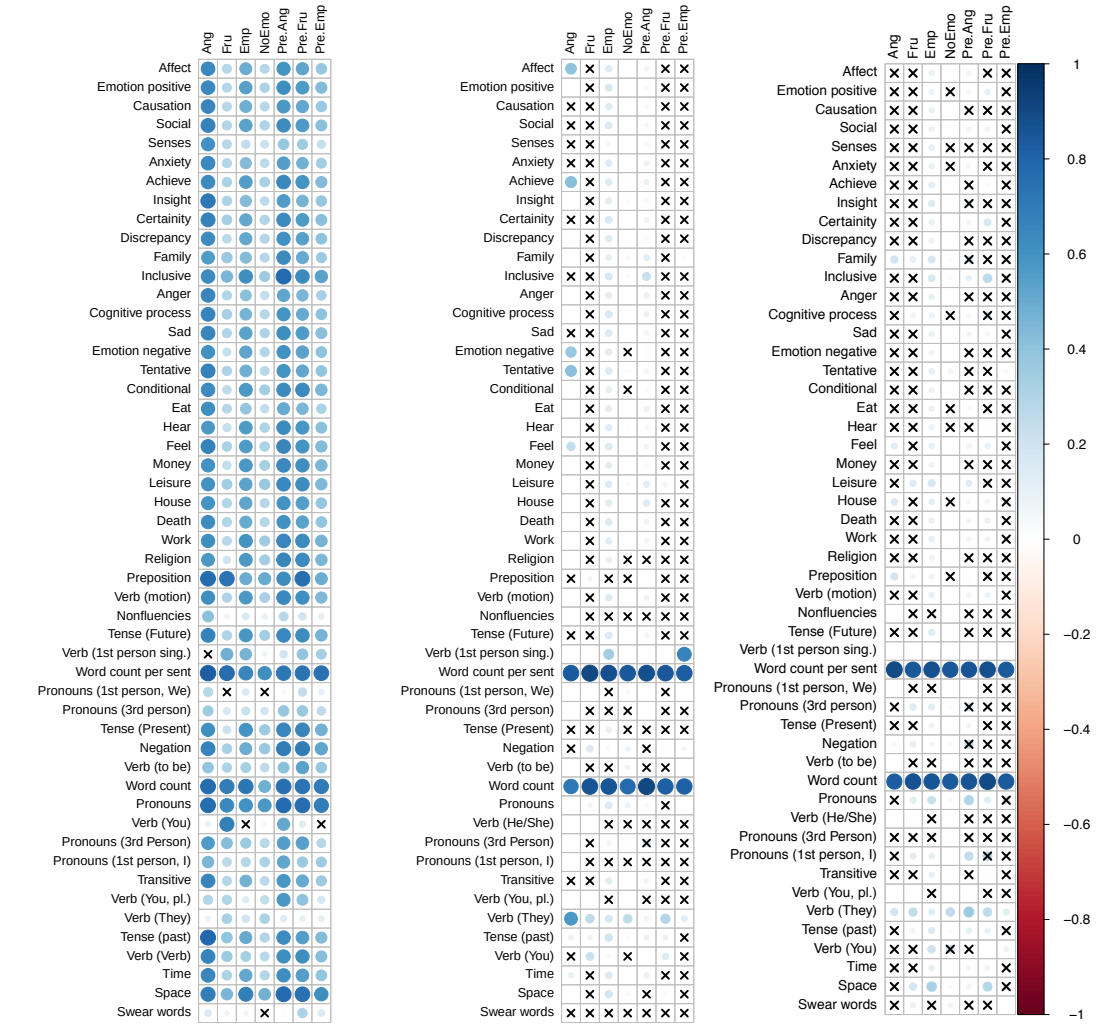
From the correlation plot, it is apparent that the non-overlapping turns of the interlocutors in anger (Ang) segments has high correlation values compared to other emotional segments non-overlapping turns and also compared to overlapping turns (Ncm and Cmp). Not surprisingly the magnitude of the correlation is significantly higher for psychological features like anxiety, affect, and sad between anger segments compared to frustration and empathy segments. Looking at the preceding-anger segments, we observed that the magnitude of r for personal concern along with psychological features are also stronger. It indicates that the cues of anger segment can be found in its preceding segments. The results also show that the uses of pronouns or negation words is directly proportional to the another speaker's usage. We also observed similar patterns in the uses of tenses. The magnitude of r is much higher for past-tense uses in anger compared to others emotional segment and preceding emotional context.

In the case of frustration, the strength of r decrease compared to the preceding segment of frustration. Unlike preceding-frustration segment, we observed that in frustration, there is less coordination between the interlocutors with an exception in preposition and word count features. Though a slight increase in r is observed in verb (they) feature. It is also observed that the interlocutors seem to be more coordinated in the use of swear words in preceding-frustration segments compared to all other segments.

In empathy segments, the coordination of the agent and customer improves compared to preceding-empathy, frustration, and no-emotion segments but the magnitude of coordination is not as impressive as anger segments.

In competitive and non-competitive overlapping turns, a very few significant coordination has been observed. The experiment with non-competitive turns shows that the interlocutors coordinate in anger segment with the features such as affect, achieve, negative emotion, tentative, and verb (they). In the case of competitive overlaps, we observed weak positive correlations between the interlocutors in preceding-

frustration segment with feature inclusive, preceding-anger segment with a verb (they), and in empathy segment with space feature.



(a) Non-overlapping segments (b) Ncm overlapping turns (c) Cmp overlapping turns

Figure 3: Correlation analysis at the non-overlapping segment, and overlapping segments, where '✕' symbol represents that the corresponding r is not significant. Pre. represents preceding segments. Ang - anger, Fru - frustration, Emp - empathy, NoEmo - no-emotion

4.3 Turn-taking Features

For the experiment with turn-taking features, we applied a multilevel logistic regression to understand the association of turn-taking features with emotional expressions and how they differ from one emotional state to another. The association of turn-taking features with emotional segments are presented in Table 1, in terms of regression coefficients. In Table 1 (a), the coefficients are reported with respect to the preceding segment of each emotion, where as in Table 1 (b), the coefficients represents the association of each turn-taking feature with the preceding emotion segment *vs.* no-emotion segments.

The results indicates that compared to the preceding context of empathy (Pre.Emp) and no emotion (NoEmo) segments, $participationEquality$, $MedianTurnC$ and $MedianPauseC$ has a negative effect on empathy (Emp) segment, where as $\%Overlap$ and length of non-competitive overlap ($MedianNcm$) has a significant positive effect. Thus indicating the importance of non-competitive overlap in the empathic segment (Emp). The results also hypothesize that during this emotional episode, agents tends to talk more allowing less participation equality between the agent and the customer. The duration of customer's turn and pause tends to be small.

The features $\%Overlap$, the length of overlaps ($MedianNcm$ and $MedianCmp$) has a positive effect

for anger segment compared to no-emotion segment. We also observed similar findings for $MedianCmp$ for preceding-anger w.r.t to no-emotion segment. It is observed that the length of non-competitive overlaps ($MedianNcm$) has a positive association where as the length of the lapse between the speakers ($MedianLb$) has a negative effect on anger with respect to preceding context (Pre.Ang). From the result of comparing preceding-anger w.r.t to no-emotion segments, we noticed that the positive association of the length of competitive overlap is present from the preceding context as an indication of anger.

The features $\%Overlap$, the length of overlaps ($MedianNcm$ and $MedianCmp$) has a positive effect for anger segment compared to the no-emotion segment. We also observed similar findings for $MedianCmp$ for preceding-anger w.r.t to the no-emotion segment. It is observed that the length of non-competitive overlaps ($MedianNcm$) has a positive association where as the length of the lapse between the speakers ($MedianLb$) has a negative effect on anger with respect to preceding context (Pre.Ang). From the result of comparing preceding-anger w.r.t to no-emotion segments, we noticed that the positive association of the length of competitive overlap is present from the preceding context as an indication of anger.

Apart from the results presented in Table 1, we also compared the association of turn-taking features with empathy, anger, and frustration with respect to each other. We found no significant difference between anger and frustration segments. However, the preceding context of anger and frustration shows that compared to the preceding-frustration, decrease of pause length is positively associated with preceding-anger segment, especially in agent's side. It is observed that an increase in the length of competitive overlap duration, $MedianCmp$, is positively associated with anger segments w.r.t empathy segments.

Figure 4: Duration distribution of competitive and non-competitive overlaps in different emotional segments.

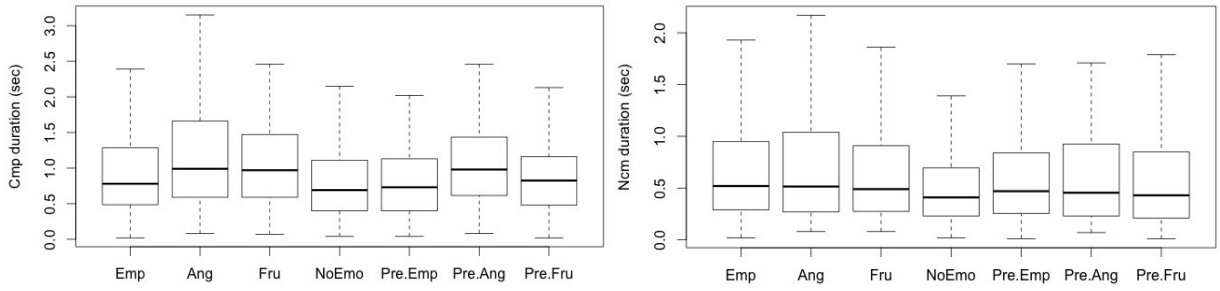


Table 1: Regression coefficient w.r.t preceding segment of each emotion and no-emotion segments.

Groups	Features	(a) Compared to preceding segment			(b) Compared to noemo segment					
		Emp	**Ang**	**Fru**	**Emp**	**Ang**	**Fru**	**Pre.Emp**	**Pre.Ang**	**Pre.Fru**
G1	participationEquality	**-0.948**	0.259	**-1.381**	-0.244	1.018	0.253	**0.823**	0.060	**1.669**
	% Overlap	**0.069**	0.059	**0.131**	**0.099**	**0.112**	**0.083**	0.035	0.046	-0.046
	% Cmp	0.002	0.008	-0.005	0.005	0.015	**0.011**	0.001	0.009	**0.015**
	% Ncm	0.007	0.000	-0.009	**0.010**	-0.002	0.000	0.002	0.000	**0.008**
G2	MedianTurnA	0.001	-0.002	-0.001	0.000	-0.001	-0.001	0.000	0.000	-0.001
	MedianTurnC	**-0.003**	0.001	**0.003**	-0.001	0.002	**0.003**	**0.002**	0.001	0.000
	MedianPauseA	0.001	0.001	-0.004	0.002	-0.005	-0.002	0.001	-0.004	**0.005**
	MedianPauseC	**-0.005**	-0.008	**-0.008**	**-0.003**	0.002	-0.002	0.002	0.003	**0.006**
	MedianCmp	0.001	0.002	0.001	**0.003**	**0.006**	**0.005**	0.002	**0.004**	**0.005**
	MedianNcm	**0.004**	**0.007**	0.002	**0.003**	**0.003**	0.002	0.001	0.001	0.000
	MedianLb	-0.002	**-0.011**	**-0.006**	**-0.005**	-0.004	-0.003	**-0.002**	0.000	0.000
	MedianLw	0.000	-0.002	-0.001	**-0.002**	-0.003	-0.001	-0.001	0.000	0.000

We also compared the duration of competitive and non-competitive overlap within different emotions and preceding emotional segments. In case of competitive, as shown in Figure 4, we observed that duration of mean competitive overlap in anger ($1.25s$) and frustration ($1.09s$) are significantly more compared to the empathy ($0.93s$), no-emotion ($0.80s$) while there is not significant difference between the duration of competitive in anger and frustration segment. In the case of preceding emotion segments, the duration

of competitive overlap in frustration is significantly higher than that of preceding-frustration ($0.91s$), where as preceding-anger ($1.12s$) and preceding-frustration is significantly higher than no-emotion. It is also observed that competitive duration in empathy segment is also longer ($p < 0.05$) than no-emotion segments. As for non-competitive duration, shown in Figure 4, there is no significant difference between anger ($0.72s$), frustration ($0.68s$) and empathy ($0.69s$) segment. But it is observed that empathy has significantly longer non-competitive overlap compared to no-emotion ($0.53s$) and preceding-empathy ($0.61s$) segment. Even, the preceding context of empathy (Pre.Emp) has significantly longer non-competitive overlap duration than the non-competitive overlap where there is no emotion. While in anger and frustration, the non-competitive overlap length is significantly higher than the no-emotion segment.

5 Conclusions

In this study, we explored the coordination of interlocutors in different emotional segments using lexical, psycholinguistic and turn-taking features. We investigated such feature sets in terms of regression coefficients, cosine similarity and correlation analysis, respectively. We observed that the interlocutors match each other turns, in terms of lexical similarity and psycholinguistic features, significantly more in anger segment compared to other emotional segments. We also observed that in preceding segment of anger the speakers shows significant correlation with each other in terms of psycholinguistic features. In terms of turn-taking features, no significant differences between anger and frustration have been noticed, apart from the difference in length of pauses in the preceding segment of the emotion. It indicates that preceding context of anger has shorter pause with respect to frustration. Unlike anger, we found less coordination in the segment where the agent is empathic even though an increase in the percentage of non-competitive overlaps has been observed. This is our preliminary study towards utilizing these feature sets for the classification of emotional states and turn-taking discourse, which we will investigate in future.

Acknowledgments

The research leading to these results has received funding from the European Union - Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 610916 - SENSEI - http://www.senseiconversation.eu/.

References

Alessandro Vinciarelli, Maja Pantic, and Hervé Bourlard. 2009. Social signal processing: Survey of an emerging domain. *Image and Vision Computing*, 27(12):1743–1759.

Alessandro Vinciarelli, Maja Pantic, Dirk Heylen, Catherine Pelachaud, Isabella Poggi, Francesca D'Errico, and Marc Schröder. 2012. Bridging the gap between social animal and unsocial machine: A survey of social signal processing. *IEEE Transactions on Affective Computing*, 3(1):69–87.

Björn Schuller, Anton Batliner, Stefan Steidl, and Dino Seppi. 2011. Recognising realistic emotions and affect in speech: State of the art and lessons learnt from the first challenge. *Speech Communication*, 53(9):1062–1087.

Evgeny A. Stepanov, Benoit Favre, Firoj Alam, Shammur Absar Chowdhury, Karan Singla, Jeremy Trione, Frederic Béchet, and Giuseppe Riccardi. 2015. Automatic summarization of call-center conversations. In *In Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU 2015)*.

Firoj Alam and Giuseppe Riccardi. 2014. Fusion of acoustic, linguistic and psycholinguistic features for speaker personality traits recognition. In *Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 955–959, May.

Firoj Alam, Morena Danieli, and Giuseppe Riccardi. 2016. Can we detect speakers' empathy?: A real-life case study. In *7th IEEE International Conference on Cognitive InfoCommunications*.

F. Alparone, S. Caso, A. Agosti, and A. Rellini. 2004. The italian liwc2001 dictionary. Technical report, LIWC.net, Austin, TX.

James J Gross and Ross A Thompson. 2007. Emotion regulation: Conceptual foundations. *Handbook of Emotion Regulation*, 3:24.

James J Gross. 1998. The emerging field of emotion regulation: An integrative review. *Review of General Psychology*, 2(3):271.

James J Gross. 2011. *Handbook of emotion regulation*. Guilford Press.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71.

James A Russell. 1980. A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161.

Klaus R Scherer. 2000. Psychological models of emotion. *The neuropsychology of emotion*, 137(3):137–162.

Maja Pantic, Roderick Cowie, Francesca D'Errico, Dirk Heylen, Marc Mehu, Catherine Pelachaud, Isabella Poggi, Marc Schroeder, and Alessandro Vinciarelli. 2011. Social signal processing: the research agenda. In *Visual analysis of humans*, pages 511–538. Springer.

Mattias Heldner and Jens Edlund. 2010. Pauses, gaps and overlaps in conversations. *Journal of Phonetics*, 38(4):555–568.

Martin L Hoffman. 2008. Empathy and prosocial behavior. *Handbook of Emotions*, 3:440–455.

Maria Koutsombogera, Dimitrios Galanis, Maria Teresa Riviello, Nikos Tseres, Sotiris Karabetsos, Anna Esposito, and Harris Papageorgiou. 2015. Conflict cues in call center interactions. In *Conflict and Multimodal Communication*, pages 431–447. Springer.

Morena Danieli, Giuseppe Riccardi, and Firoj Alam. 2014. Annotation of complex emotion in real-life dialogues. In Roberto Basili, Alessandro Lenci, and Bernardo Magnini, editors, *Proc. of 1st Italian Conf. on Computational Linguistics (CLiC-it) 2014*, volume 1.

Morena Danieli, Giuseppe Riccardi, and Firoj Alam. 2015. Emotion unfolding and affective scenes: A case study in spoken conversations. In *Proc. of Emotion Representations and Modelling for Companion Systems (ERM4CT) 2015,*. ICMI.

Paul Ekman. 1999. Basic emotions. *Handbook of cognition and emotion*, 98:45–60.

Rafael A Calvo and Sidney D'Mello. 2010. Affect detection: An interdisciplinary review of models, methods, and their applications. *Affective Computing, IEEE Transactions on*, 1(1):18–37.

Harvey Sacks, Emanuel A Schegloff, and Gail Jefferson. 1974. A simplest systematics for the organization of turn-taking for conversation. *Language*, pages 696–735.

Shammur Absar Chowdhury, Giuseppe Riccardi, and Firoj Alam. 2014. Unsupervised recognition and clustering of speech overlaps in spoken conversations. In *Proc. of Workshop on Speech, Language and Audio in Multimedia - SLAM2014*, pages 62–66.

Shammur Absar Chowdhury, Morena Danieli, and Giuseppe Riccardi. 2015a. Annotating and categorizing competition in overlap speech. In *Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Shammur Absar Chowdhury, Morena Danieli, and Giuseppe Riccardi. 2015b. The role of speakers and context in classifying competition in overlapping speech. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Shammur Absar Chowdhury, Evgeny A. Stepanov, and Giuseppe Riccardi. 2016. Predicting user satisfaction from turn-taking in spoken conversations. In *Proc. of INTERSPEECH*.

Shrikanth Narayanan and Panayiotis G Georgiou. 2013. Behavioral signal processing: Deriving human behavioral informatics from speech and language. *Proceedings of the IEEE*, 101(5):1203–1233.

Štefan Beňuš, Agustín Gravano, and Julia Hirschberg. 2011. Pragmatic aspects of temporal accommodation in turn-taking. *Journal of Pragmatics*, 43(12):3001–3027.

Zhihong Zeng, Maja Pantic, Glenn I Roisman, and Thomas S Huang. 2009. A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(1):39–58.

Advancing Linguistic Features and Insights by Label-informed Feature Grouping: An Exploration in the Context of Native Language Identification

Serhiy Bykh
Seminar für Sprachwissenschaft
Universität Tübingen
sbykh@sfs.uni-tuebingen.de

Detmar Meurers
Seminar für Sprachwissenschaft
Universität Tübingen
dm@sfs.uni-tuebingen.de

Abstract

We propose a hierarchical clustering approach designed to group linguistic features for supervised machine learning that is inspired by variationist linguistics. The method makes it possible to abstract away from the individual feature occurrences by grouping features together that behave alike with respect to the target class, thus providing a new, more general perspective on the data. On the one hand, it reduces data sparsity, leading to quantitative performance gains. On the other, it supports the formation and evaluation of hypotheses about individual choices of linguistic structures. We explore the method using features based on verb subcategorization information and evaluate the approach in the context of the Native Language Identification (NLI) task.

1 Introduction and related work

Native Language Identification (NLI) is the task of inferring the native language (L1) of writers from texts they wrote in another language. NLI started to attract attention in computational linguistics with the work of Koppel et al. (2005). Since then interest has steadily risen, leading to the First NLI Shared Task in 2013, with 29 participating teams (Tetreault et al., 2013).

NLI is usually considered as a text classification problem with the different L1s as labels. A range of features reaching from character and word n-grams to dependency- and constituency-based features have successfully been used in standard supervised machine learning setups, yielding accuracies of up to around 83% for the 11 classes in the First NLI Shared Task. Some more recent papers further advance the best result from that competition, namely 83.6% (Jarvis et al., 2013), reaching around 85% (Bykh and Meurers, 2014; Ionescu et al., 2014).

While pushing the quantitative side is one option of advancing the NLI work further, another avenue of research tries to improve our understanding of how the different feature types work and what conclusions one can draw from these observations for Second Language Acquisition (SLA) research. Swanson and Charniak (2013) utilized Tree Substitution Grammars as well as different measures of relevancy and redundancy to extract indicative linguistic patterns. Swanson and Charniak (2014) adopted the approach to dependencies. Malmasi and Dras (2014) proposed a technique to detect over- and underuse of certain patterns by writers with a particular L1-background using linear SVM weights derived from Adaptor grammar collocations or Stanford Dependencies. Meurers et al. (2014) employed verb subcategorization patterns as features and showed that there are differences in the usage patterns of verbs between native English writers and, e.g., writers with Chinese L1-background. Bykh and Meurers (2014) systematically explored constituency-based features and discussed the distinctive power of the different variants realizing lexical and phrasal categories. Malmasi and Cahill (2015) investigated the correlation between various features in a feature set commonly used in NLI.

Many of the current NLI approaches rely on large feature sets, which makes it difficult to qualitatively interpret the findings. We therefore want to explore grouping features together which behave alike to advance linguistic insight and improve classification. In place of zooming in on single features, with the potential danger of overfitting the training data, feature grouping can help make explicit underlying

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 739–749, Osaka, Japan, December 11-17 2016.

linguistic properties within a set of features. Thus, on the one hand, it can support the identification of linguistic generalizations, which is relevant for qualitative analysis and theoretical interpretation. On the other hand, we also expect quantitative benefits due to the potential reduction of data sparsity, especially in cases where the particular single feature realizations might be rare but the underlying structure captured by a group is more common.

One of the well-established techniques for building feature groups is hierarchical clustering (Park, 2013; Krier et al., 2007; Butterworth et al., 2005). It can be an effective method for capturing linguistic generalizations. For example, Pate and Meurers (2007) show in the context of PCFG parsing that contextually enriching categories followed by clustering the categories with similar distributions results in a performance improvement.

In this paper, we propose to employ hierarchical clustering for feature grouping in a way that is informed by the classification label – here the L1 of the writer. Adopting a variationist linguistic perspective that attempts to identify variants of an underlying variable (Tagliamonte, 2011), we illustrate and evaluate the feature grouping technique in detail for features encoding the different subcategorization options realized by a given verb – a feature type that is well-motivated in related SLA research (Tono, 2004; Callies and Szczesniak, 2008; Stringer, 2008). Using the technique, we first test the hypothesis, whether writers with different L1-backgrounds prefer certain subcategorization patterns when realizing particular verbs. Then we show how the technique can be used to investigate specific hypotheses about L1-transfer suggested in the SLA research; we focus on the subject in the subcategorization pattern and explore some of its realization options in L1 Chinese following Wang (2009). The results presented below confirm that feature grouping can indeed provide theoretical and practical benefits.

2 Feature grouping

We propose a label-informed feature grouping technique that can be used to group structured linguistic features in line with a variationist perspective. We first introduce the variationist perspective and the nature of variationist features, before we turn to our implementation of the technique.

Variationist sociolinguistics In variationist sociolinguistics, the focus is on the possible linguistic choices made by a speaker. This makes it possible to connect the choices in the language with extralinguistic variables, such as the gender or the age of a speaker. For example, in Labov's seminal study "The Social Stratification of (r) in New York City Department Stores" (Labov, 1972), he found that the presence or absence of the consonant [r] in postvocalic position (e.g., *fourth*) correlates with the ranking of people in status (social stratification). Hence, under a variationist perspective, one observes which of the possible *variants* of a *variable* is chosen by a particular speaker (Tagliamonte, 2011). Recent research in the language learning context argues that a preference for particular variants can also be indicative of individual characteristics such as proficiency or L1-background (Lüdeling, 2011; Callies and Zaytseva, 2011; Meurers et al., 2014; Bykh and Meurers, 2014).

Variationist features To obtain *variationist features* one has to implement the logic described in the previous paragraph. It requires choosing some language *variables* that can be realized by a particular set of *variants*. For our first explorations of the proposed technique, we chose verb lemmas as variables and the different subcategorization (subcat) patterns of that lemma as variants.

Grouping technique As motivated above, we want to explore whether writers with a given L1 prefer certain subcat variants when realizing a particular verb lemma variable.

In the training corpus, we can record the relative frequencies for the different subcat variants used to realize a lemma. Individual lemmas occur quite rarely, though, so we want to group together all those lemmas that behave alike with respect to their subcat variants.

Can we also take the classification label into account when clustering the variables by the frequency proportions of their variants? In other words, how can we group those lemmas together that for a given L1 have similar proportions of subcat variant realizations? In order to incorporate the classification label into the grouping procedure, we do not generate a single vector of variants for a variable, but k vectors, where k is the number of L1 labels in the training data. Each of the k vectors contains the proportions of

the variants for a given variable, calculated using the subset of the training data for a particular L1. Then the k vectors for each variable are concatenated in order to get an instance for clustering. Hierarchical clustering then groups variables together that for writers of a specific L1 realize a similar set of variants in similar proportions, i.e., it groups lemmas together that for a specific L1 label pattern alike with respect to the realized subcat variants. Since the feature set for clustering is informed by the L1 labels, we refer to this technique as *label-informed feature grouping*. Viewed from the variationist perspective, the method is designed to group those variables together that in terms of their variants behave alike with respect to the classification label.

Let us spell this out in an example. Assume that writers with Spanish L1 prefer the subcat variant $p \in \{p, q\}$, whereas writers with Chinese L1 prefer the variant q in connection with a particular set of verbs A. That information is captured by the difference in relative frequencies for the variants p and q in connection with A in the training data subsets for the two different L1s. Using separate vectors for the different L1s, explicitly provides that relevant information to the clustering algorithm. Clustering thus can identify the group A of verbs that is indicative for the classification purposes in terms of the choice of variants made by different L1s, and also of interest from the perspective of interpreting these effects in terms of SLA research.

We cluster the variables via *agglomerative hierarchical clustering*, employing some standard parameters, namely, Euclidean distance[1] and complete-linkage. We set the number of clusters $c = 1$, which means that after clustering we obtain a *dendrogram* corresponding to a *single-rooted binary tree*.

Now, the question is, how to decide, which grouping is the most appropriate one? I.e., where do we want to cut the dendrogram? We approach that issue experimentally, by systematically applying different branch length cut-offs with step $s = 0.1$ to the dendrogram, and then evaluating every grouping via text classification using the different groupings as features.

In connection with using subcat variants, realized by groups of verbs, as features there are two more points to clarify. First, how to merge clustered variables with different sets of variants? Here, we simply take the union of the variant sets as the resulting variant set of the variables group. Second, how to compute the feature values for the groups? For that we use the *micro average* measure adapted to the variationist perspective (Krivanek, 2012; Meurers et al., 2014). In sum, we apply the following steps:

1. For each of the n variables V_i and each of the m variants v_j occurring in the whole training data, calculate the matrices M_{ij}^k using the corresponding label-distinct data subset l_k:

$$M_{ij}^k = \frac{f(v_j, V_i, l_k)}{\sum_{q=1}^{m} f(v_q, V_i, l_k)}$$

   where $f(v, V, D)$ yields the frequency of the variant v realizing the variable V in the data D. Here $D = l_k$. If a variant v does not occur in the context of V using data D, $f(v, V, D) = 0$.

2. Perform a horizontal matrix concatenation of the k matrices M_{ij}^k resulting in a single matrix M_{ij} containing the variable based instances for clustering (the rows of M_{ij}).

3. Perform hierarchical clustering with the number of clusters $c = 1$, Euclidean distance, and complete-linkage as parameters.

4. Systematically apply different branch length cut-offs r using a suitable step s (here we employed $s = 0.1$). Evaluate the resulting clusters, i.e., groups of variables by using them as features in a classification setup.

[1] We also explored using other distance measures, such as the Manhattan or the Hellinger distance. Especially, the latter is supposed to be more suitable for probability-based features. However, the Euclidean distance performed best.

(a) Merging groups: Let a group (cluster) C contain x variables, each realized by a particular variant set X_i. Then the resulting variant set X^c for the variables group C is defined as the union of all variant sets X_i:

$$X^c = \bigcup_{i=1}^{x} X_i$$

(b) Encoding groups as features for classification: Calculate *micro average* for each variant $v \in X^c$ associated with the group $C = \{V_1, ..., V_n\}$, using data t:

$$mic(v, C) = \frac{\sum_{i=1}^{n} f(v, V_i, t)}{\sum_{i=1}^{n} \sum_{j=1}^{m} f(v_j, V_i, t)}$$

where $f(v, V, D)$ is defined as above (1), and $D = t$ is a given text.

5. Terminate evaluation (at the latest) after a particular cut-off r yielded one single group containing the whole variables set.

Note that the same technique can also be used to group *variants* based on their frequency proportions for all variables using the k label-based data subsets – an option we here do not go into further to keep things clear and within the space constraints.

3 Data

For the experiments described in this paper, we use the TOEFL11 corpus (Blanchard et al., 2013) introduced for the NLI Shared Task 2013 (Tetreault et al., 2013), which has become a common frame of reference for NLI research. It consists of essays written by English learners with 11 L1-backgrounds (Arabic, Chinese, French, German, Hindi, Italian, Japanese, Korean, Spanish, Telugu and Turkish) at three proficiency levels (low, medium, high). Each of the 11 L1s is represented by 1,100 essays (900 training, 100 development, 100 test). We use the union of the training and development sets for training and the standard test set for testing. In total for all L1s, we thus train on 11,000 and test on 1,100 essays.

4 Tools

We utilized the MATE tools[2] (Björkelund et al., 2010) for data preprocessing (tokenization, lemmatizing, POS-tagging) and the MATE dependency parser (Bohnet, 2010) to identify the arguments of a verb realized in a sentence, i.e., the subcat frame that was realized. For hierarchical clustering we employed WEKA (Hall et al., 2009). To process the resulting dendrograms we used the Libnewicktree[3] tree parser. Finally, classification was carried out using L2-regularized Logistic Regression from the LIBLINEAR package (Fan et al., 2008) accessed through WEKA.

5 Features

The hypothesis we are testing is whether writers with different L1s prefer different subcat variants. To systematically explore the potential benefits of feature grouping, we start with *simple features*, where every variable, i.e., verb lemma, is considered separately. We then infer sets of *complex features*, i.e., sets of various groups of variables using the proposed technique, abstracting from individual verb lemmas to classes of verbs. All feature values are calculated using *micro average* as introduced in section 2 (4b), with simple features being a special case of the complex ones, where the group C consists of a single variable ($C = \{V_1\}$).

[2] https://code.google.com/p/mate-tools

[3] https://github.com/cjb/libnewicktree

5.1 Simple features

We dependency parsed the data and extracted the corresponding argument realization patterns, i.e., the realized subcat variants, for all verbs occurring in the data. We consider the following labels as arguments:

- *sbj*: subject
- *lgs*: logical subject
- *obj*: (in)direct object or clause complement
- *bnf*: benefactor in dative shift
- *dtv*: dative in dative shift
- *prd*: predicative complement
- *oprd*: object complement
- *put*: locative complements of the verb put
- *vc*: verb chain

Utilizing the verb lemmas with their extracted subcat variants, we generated features, such as:

- *believe*_sbj
- *believe*_sbj_obj
- *may*_sbj_vc
- *put*_sbj_put
- *help*_sbj_obj
- *make*_sbj_obj_oprd

Feature reduction We performed the following three feature reduction steps due to some theoretical and practical considerations:

1. Verbs as features are rather rare. In order to reduce data sparsity issues, at this point we opted for ignoring the *different permutations* of arguments within a subcat variant. This step reduced the number of distinct variants from 355 to 218.

2. Some of the subcat variants are still rather specific and unlikely to occur frequently enough in the data. Some of them also suffer from tagging or parsing errors. So, in a second reduction step we grouped all *argument labels into three coarse-grained classes* in order to get more general patterns and to cope with data sparsity:

   - $\{sbj, lgs\} \rightarrow s$ (subject)
   - $obj \rightarrow o$ (object)
   - $\{bnf, dtv, prd, oprd, put, vc\} \rightarrow x$ (rest group)

   The number of distinct subcat variants reduced from 218 to 48. Applied to the examples listed above, we obtain features of the following form:

   - *believe*_sbj $\rightarrow$ *believe*_s
   - *believe*_sbj_obj $\rightarrow$ *believe*_s_o
   - *may*_sbj_vc $\rightarrow$ *may*_s_x
   - *put*_sbj_put $\rightarrow$ *put*_s_x
   - *help*_sbj_obj $\rightarrow$ *help*_s_o
   - *make*_sbj_obj_oprd $\rightarrow$ *make*_s_o_x, etc.

3. The last reduction step is conceptually different from the first two. It is based on theoretical considerations in connection with the variationist perspective: We are interested in the linguistic choices made by a speaker. If there is only a single variant for using a verb, we cannot observe a *choice* being made. We therefore dropped all features for verb lemmas that only occur with a single subcat variant in the training data. That reduced the number of distinct verb lemmas from originally 11,401 to 3,785.

Feature reduction clearly also means a loss of potentially indicative subcat information. A more fine-grained reduction therefore constitutes an important topic for future work.

As a result of feature reduction, we obtain 3,785 distinct verb lemmas (variables) with 14,389 subcat variants as features. That means that on average there are roughly four subcat variants per variable.

5.2 Complex features

We refer to the features defined by grouping the verb lemmas as proposed in section 2 as *complex features*. The purpose of these complex features is to abstract away from individual verb lemmas to more general classes. In contrast to Meurers et al. (2014), where verb lemmas realizing *exactly the same* subcat variants were grouped together, the technique proposed here makes it possible to systematically explore a range of different groupings of lemmas based on the *similarity* of the realized subcat variants, and to take into account the classification label. Each group of verb lemmas C and the corresponding set of subcat variants X^c constitutes a complex feature in the sense of (4a) of section 2.

6 Results

An overview of the results is presented in Figure 1. On the x-axis, the leftmost point (marked "s") corresponds to using only simple features (every verb lemma with the corresponding subcat variants is considered separately), i.e., no clustering. With increasing x-values, we go up in the dendrogram to obtain groups of verbs. The x-values are the branch length cut-offs applied to the dendrogram using a step of 0.1. The y-axis represents the accuracy of the classification on the test set, using the training-test split described in section 3 and the classifier spelled out in 4. The random baseline is 9.1%.

Model with simple and complex features ([s/c]): This is the basic setting using simple and complex features. The figure shows that 44.5% at point "s" is the highest accuracy, so feature grouping does not provide a quantitative edge. For settings incorporating complex features, the best result is 44.2%, obtained for the cut-off 0.3. The clustering technique groups verb lemmas in terms of the proportion of

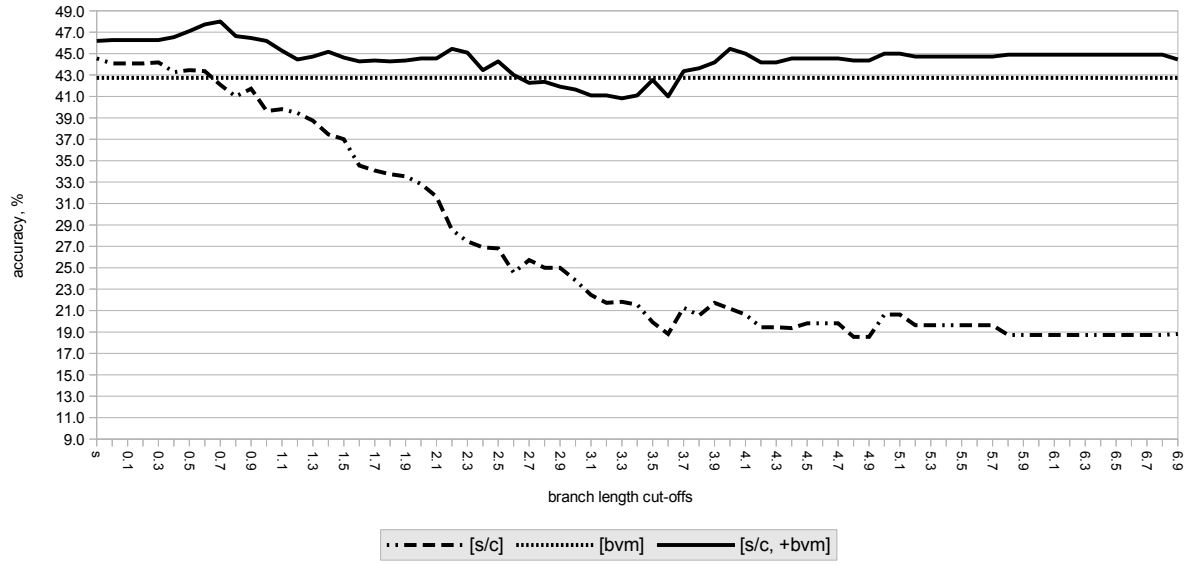


Figure 1: Accuracy of classification for different feature sets

their subcat variants. The particular verb lemmas, i.e., surface forms, which are part of a group, are not by themselves encoded in the feature space any more. For complex features the classifier therefore does not have an access to the potentially highly indicative surface based properties. Even different misspellings of the verbs can be indicative of the native language – distinctions that the basic method counting variants glosses over.[4] The disadvantage of loosing indicative surface information seems to outweigh the potential advantage of the generalization in terms of reducing data sparsity. We can validate that assumption by creating a binary verb lemma model and then combining it in a model with the simple and complex variant features.

Binary verb lemma model ([bvm]): To be able to identify the contribution of the subcat variants of individual verbs and groups of verbs, we need a way to separately quantify the information provided by the verb lemma itself (i.e., the presence of the variable, as separate from the choice of variants). In the bvm model, we thus only encode the presence/absence of the 3,785 verb lemmas for each text. The accuracy for that model is 42.7%. We included that result as a line in Figure 1 to visualize the performance relative to the other two models, which make use of subcat pattern information. Comparing the other models to that one shows the benefits of incorporating the subcat variants as features. Indeed, the other curves discussed below show better results, confirming that such features are useful.

Binary verb lemma combined with simple and complex features ([s/c, +bvm]): To validate our assumption regarding the role of surface properties, we tried a combined setup, where the [*s/c*] setting was used in combination with the binary verb lemma model [*bvm*] described above. We assume the [*bvm*] model to restore the surface information lost due to the generalization, which should improve the classification performance. Indeed, the classification performance increased compared to the basic [*s/c*] setting. For simple features, the accuracy is 46.2%, and thus 1.7% higher. Including [*bvm*] therefore is beneficial even for settings not involving clustering. For settings including complex features, the difference depends on the actual cut-off. The best performance is 48.0% obtained using the cut-off 0.7 (273 complex and 3016 simple features). The difference between the best complex feature results is 3.8%. In most of the cases, the difference is even much higher, as seen when comparing the [*s/c*] and the [*s/c, +bvm*] curves at corresponding cut-offs in Figure 1. That result supports our assumption regarding the role of the surface properties. It is also supported by the shape of the [*s/c, +bvm*] curve only. The best model using complex features (cut-off 0.7) outperforms the model solely based on simple features ("s") by 1.8%[5]. Thus, when built on top of a surface-based model such as [*bvm*], the proposed grouping and generalization technique shows practical advantages in terms of accuracy. The findings confirm the hypothesis that in general learners with different L1s seem to prefer different subcat patterns. Finally, using more data or some more frequent variables resulting in more reliable frequency distributions of the variants, is expected to increase the quantitative gains. We plan to explore this issue in our future work.

Relative performance In comparison with previous research, the proposed automatic feature grouping method outperforms the approach presented in Meurers et al. (2014), where only verb lemmas with equal subcat variant sets constituted a group. A replication of that approach employing the verb subcat features and the data setup used in this paper, showed an accuracy of 38.7%, and after adding the *[bvm]* model, we obtained 44.4%. This result is 3.6%[6] lower than our best accuracy obtained in [*s/c, +bvm*].

In order to further investigate the potential quantitative advantages of the proposed features and the clustering method, we combined a range of features using the meta-classifier approach described in Bykh and Meurers (2014). The results are summarized in Table 1. First, we combined the core 16 feature types employed in Bykh et al. (2013), namely features based on n-grams, dependencies, local trees, suffix information, linguistic complexity and lemma realization, with the best performing model in Bykh and Meurers (2014), which is based on constituency variation features plus 40 different types of n-grams, i.e., word- and lemma-based n-grams as well as two types of n-grams incorporating POS,

[4]For example, we discovered that some of the clusters contained misspelled versions of the same verbs, such as *commmunicate/communicat*, *tounderstand/ubderstand*, *exaggrate/exxagarate*, etc.

[5]$p < 0.05$ using McNemar's test.

[6]$p < 0.001$ using McNemar's test.

with $1 \leq n \leq 10$ each (see also Bykh and Meurers, 2012). This model yielded an accuracy of 85.2%.[7] Second, we added the best performing system in this paper, namely [*s/c, +bvm*] with cut-off 0.7[8] to that ensemble, and alternatively the basic setting of [*s/c, +bvm*] without clustering, i.e., using simple features only ("s"). Both options showed an increase in accuracy by the same value of 0.2%, resulting in 85.4%. To the best of our knowledge, the highest outcome obtained so far on the same data was 85.3%, reported by Ionescu et al. (2014). Thus, by using the new features explored in this paper, it was possible to slightly outperform the already very high best previous accuracy on the standard TOEFL11 data setup. However, in the comprehensive ensemble model used here, there was no quantitative difference between adding the best performing [*s/c, +bvm*] setting (cut-off 0.7), which incorporates complex features, and the lower performing version containing simple features only ("s"). Yet, there is a difference in terms of the feature counts for the two models, namely, 16,841 vs. 18,174 respectively. Thus, the version incorporating complex features is more efficient, providing the same quantitative advantage with a more compact model. This supports the assumption that the generalizations made by the technique are reasonable. The findings suggest that the approach can further advance the already high performance of the state-of-the-art NLI systems.

Rank	Id	System	Accuracy
1	A	D + E + F/G	85.4%
2	B	Ionescu et al. (2014)	85.3%
3	C	D + E	85.2%
4	D	Bykh and Meurers (2014)	84.8%
5	E	Bykh et al. (2013)	82.5%
6	F	[*s/c, +bvm*], cut-off 0.7	48.0%
7	G	[*s/c, +bvm*], s	46.2%

Table 1: Relative performance. System B is the best previously reported system based on the same data.

7 Qualitative explorations

In the previous sections we explored in detail the quantitative gains of the proposed technique using verb subcat features. In this section we sketch, how the method can be used to advance the qualitative analysis in the context of the SLA research. In particular, we investigate the hypothesis by Wang (2009), suggesting that learners with L1 Chinese overuse *pronoun-subjects* over *noun-subjects* in Chinese-English translations.[9] The findings of Wang (2009) based on translations by 81 students support the hypothesis.

In order to investigate this hypothesis, we slightly modify our features, i.e., we use only the subject part of the verb subcat, and in addition we consider only those subjects, which are tagged as personal pronouns (*prp*) or nouns (*nn*). So, based on our training data, we extract features such as *believe*_sbj+prp and *believe*_sbj+nn, or *study*_sbj+prp and *study*_sbj+nn, etc. Then we run *one vs. rest* classifiers with L1 Chinese vs. the western L1s in our set, namely French, German, Italian and Spanish. The classifiers follow the logic of the [*s/c*] and [*s/c, +bvm*] settings discussed in section 6. To determine distinctive patterns, we used the weights assigned to the features by the classifier (Malmasi and Dras, 2014).

First, we explored the general usage pattern for *sbj+prp* and *sbj+nn* variants, detached from particular verb lemmas. That was done by cutting off the dendrograms for the [*s/c*] settings at the root (cut-off 3.0), which results in having all of the considered verbs in a single cluster and hence, just the two variants, i.e., *sbj+prp* and *sbj+nn*, encoded by the relative frequency for each text. It turned out that both weights are negative, showing that there do not seem to be any pattern indicative for L1 Chinese compared to the

[7] Best ensemble optimization parameters for all ensembles in this paper: *+all, -opt* (Bykh and Meurers, 2014).

[8] This cut-off turned out to yield best results in our evaluations on both, the TOEFL11 *test* and *development* sets, thus it seems to be relatively reliable for TOEFL11 data.

[9] "Chinese people always hold the idea that human being and nature are mingled together, so Chinese people intend to make themselves as the start to narrate object things and are used to taking the pronoun as the subject. However, in the western philosophy, object is emphasized and it is believed that human being and nature are separated. So western people intend to express things from an object view and are used to taking non-pronoun such as things or abstract concept as the subject. The choice of pronoun-subject or non-pronoun-subject between Chinese and English will lead to negative transfer of mother tongue, which will make the translation of subject an improper one." (Wang 2009, p. 139)

western L1s. Second, we bring the actual verbs back into the equation, and explore the best performing [*s/c, +bvm*] setting (cut-off 2.6). Here, all verb lemmas are grouped into five clusters. The findings are summarized in Table 2.

Cluster id	# Verb lemmas	Pattern indicative for L1 Chinese	Weight
1	166	sbj+nn	< 0.01
2	100	-	-
3	312	-	-
4	65	sbj+prp	0.73
5	68	sbj+prp	0.19

Table 2: Usage pattern for L1 Chinese at the best performing dendrogram cut-off yielding five clusters.

For the cluster 1 there is some very weak (a positive weight ≈ 0) indication for the variant *sbj+nn*, which essentially can be ignored. For the two clusters 2 und 3 there is no indicative pattern for L1 Chinese, whereas for the two clusters 4 and 5, there is a clear indicative preference for the variant *sbj+prp*. In sum, for most of the verbs in our data set, there is no indicative usage pattern for L1 Chinese compared to the western L1s. However, in connection with some particular verb groups, there is an indicative preference indeed, namely, for the variant *sbj+prp*, supporting the given hypothesis. Interestingly, the method does not simply support a known hypothesis, but it makes it possible to observe subsets of verbs for which the characteristics emerge. Studying what the 65 verbs grouped in the most indicative cluster have in common thus provides the opportunity for a more fine-grained qualitative analysis in SLA research.

8 Conclusions

In this paper, we proposed and explored a grouping technique for linguistic features. The method is inspired by a variationist linguistic perspective and uses hierarchical clustering on the basis of label-informed feature representations. The approach emphasizes how the underlying linguistic structure informs the classification label, reducing potential problems arising from idiosyncrasies and sparsity of individual features. We evaluated the approach in the context of NLI using a linguistic feature type well-suited to a variationist perspective, verb subcategorization patterns, treating the verb lemmas as variables and the different patterns as variants.

We motivated why we consider the technique to be of interest from a theoretical and a practical perspective. Grouping verb lemmas based on the subcategorization information, and thus abstracting from individual occurrences to the underlying linguistic structure, resulted in a significant improvement in terms of accuracy, confirming the hypothesis that in general learners with different L1s seem to prefer different subcategorization patterns. We then turned to investigating a particular hypothesis from SLA regarding the usage of the subject as part of the verb subcategorization information. We showed that the method can discover differences in the variant realization patterns in connection with different automatically induced classes of verbs, supporting a fine-grained qualitative analysis. Linking the analysis to a theoretical perspective informed by traditional SLA research, the method seems well capable of advancing the qualitative insights in NLI – a primary concern in that field of research today. Combining features obtained by the approach proposed in this paper with a set of previously used features resulted in an accuracy of 85.4%, which is the best result reported so far on the standard TOEFL11 data setup.

In terms of future work, we plan to explore different syntactic and morphological features under a variationist perspective, to extend the qualitative analysis, and to establish a firm enough link between the data-induced patterns and the traditional insights into L1-transfer to be able to test specific SLA hypotheses. Regarding the label-informed feature grouping technique, we are also considering applying it to NLP tasks other than NLI in order to obtain a more comprehensive assessment of the method.

References

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Demonstration Volume of the 23rd International Conference on Computational Linguistics (COLING)*, pages 23–27, Beijing, China. `https://code.google.com/p/mate-tools/`.

Daniel Blanchard, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. TOEFL11: A corpus of non-native English. Technical report, Educational Testing Service.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23th International Conference on Computational Linguistics (COLING)*, pages 89–97, Beijing, China.

Richard Butterworth, Gregory Piatetsky-Shapiro, and Dan A. Simovici. 2005. On feature selection through clustering. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'2005)*.

Serhiy Bykh and Detmar Meurers. 2012. Native language identification using recurring n-grams – investigating abstraction and domain dependence. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 425–440, Mumbay, India.

Serhiy Bykh and Detmar Meurers. 2014. Exploring syntactic features for native language identification: A variationist perspective on feature encoding and ensemble optimization. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, pages 1962–1973, Dublin, Ireland.

Serhiy Bykh, Sowmya Vajjala, Julia Krivanek, and Detmar Meurers. 2013. Combining shallow and linguistically motivated features in native language identification. In *Proceedings of the 8th Workshop on Innovative Use of NLP for Building Educational Applications (BEA-8) at NAACL-HLT 2013*, Atlanta, GA.

Marcus Callies and Konrad Szczesniak. 2008. Argument realization, information status and syntactic weight – a learner-corpus study of the dative alternation. In Maik Walter and Patrick Grommes, editors, *Fortgeschrittene Lernervarietäten*, pages 165–187. Niemeyer.

Marcus Callies and Ekaterina Zaytseva. 2011. The corpus of academic learner English (CALE): A new resource for the study of lexico-grammatical variation in advanced learner varieties. In *Hedeland, Hanna and Thomas Schmidt and Kai Wörner*, pages 51–56. Multilingual Resources and Multilingual Applications (Hamburg Working Papers in Multilingualism B 96).

R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874. Software available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear`.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. In *The SIGKDD Explorations*, volume 11, pages 10–18.

Radu Tudor Ionescu, Marius Popescu, and Aoife Cahill. 2014. Can characters reveal your native language? A language-independent approach to native language identification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1363–1373. ACL.

Scott Jarvis, Yves Bestgen, and Steve Pepper. 2013. Maximizing classification accuracy in native language identification. In *Proceedings of the 8th Workshop on Innovative Use of NLP for Building Educational Applications (BEA-8) at NAACL-HLT 2013*, Atlanta, GA.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Determining an author's native language by mining a text for errors. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (KDD '05)*, pages 624–628, New York.

C. Krier, D. François, F. Rossi, and M. Verleysen. 2007. Feature clustering and mutual information for the selection of variables in spectral data. In *Proceedings of the ESANN'2007*, pages 157–162, Bruges, Belgium.

Julia Krivanek. 2012. Investigating syntactic alternations as characteristic features of learner language. Master's thesis, University of Tübingen, April.

William Labov. 1972. *Sociolinguistic Patterns*. University of Pennsylvania Press.

Anke Lüdeling. 2011. Corpora in linguistics: Sampling and annotation. In Karl Grandin, editor, *[Nobel Symposium 147] Going Digital: Evolutionary and Revolutionary Aspects of Digitization*, pages 220–243, New York. Science History Publications.

Shervin Malmasi and Aoife Cahill. 2015. Measuring feature diversity in native language identification. In *Proceedings of the 10th Workshop on Innovative Use of NLP for Building Educational Applications (BEA-10) at NAACL-HLT 2015*, pages 49–55, Denver, Colorado.

Shervin Malmasi and Mark Dras. 2014. Language transfer hypotheses with linear SVM weights. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1385–1390. ACL.

Detmar Meurers, Julia Krivanek, and Serhiy Bykh. 2014. On the automatic analysis of learner corpora: Native language identification as experimental testbed of language modeling between surface features and linguistic abstraction. In *Diachrony and Synchrony in English Corpus Studies*, pages 285–314, Frankfurt am Main. Peter Lang.

Cheong Hee Park. 2013. A feature selection method using hierarchical clustering. In R. Prasath and T. Kathirvalavakumar, editors, *Proceedings of the International Conference on Mining Intelligence and Knowledge Exploration (MIKE 2013), LNAI 8284*, pages 1–6, Virudhunagar, Madurai, India. Springer International Publishing Switzerland.

John Pate and Detmar Meurers. 2007. Refining syntactic categories using local contexts – experiments in unlexicalized pcfg parsing. In *Proceedings of the Sixth Workshop on Treebanks and Linguistic Theories (TLT 2007)*, Bergen, Norway.

David Stringer. 2008. What else transfers? In Roumyana Slabakova et al., editor, *Proceedings of the 9th Generative Approaches to Second Language Acquisition Conference (GASLA 2007)*, pages 233–241, Somerville, MA. Cascadilla Proceedings Project.

Ben Swanson and Eugene Charniak. 2013. Extracting the native language signal for second language acquisition. In *Proceedings of NAACL-HLT*, pages 85–94. ACL.

Ben Swanson and Eugene Charniak. 2014. Data driven language transfer hypotheses. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 169–173. ACL.

Sali A. Tagliamonte. 2011. *Variationist Sociolinguistics: Change, Observation, Interpretation*. John Wiley & Sons.

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *Proceedings of the 8th Workshop on Innovative Use of NLP for Building Educational Applications (BEA-8) at NAACL-HLT 2013*, Atlanta, GA, USA, June. ACL.

Yukio Tono. 2004. Multiple comparisons of IL, L1 and TL corpora: the case of L2 acquisition of verb subcategorization patterns by Japanese learners of English. In Guy Aston, Silvia Bernardini, and Dominic Stewart, editors, *Corpora and Language Learners*, pages 45–66. John Benjamins.

Xiaoru Wang. 2009. Exploring the negative transfer on english learning. *Asian Social Science*, 5(7):138–143.

Modeling Diachronic Change in Scientific Writing with Information Density

Raphael Rubino[†‡] **Stefania Degaetano-Ortlieb**[†] **Elke Teich**[†] **Josef van Genabith**[†‡]
[†]Universität des Saarlandes, [‡]DFKI
Saarbrücken, Germany
{s.degaetano,e.teich}@mx.uni-saarland.de
{raphael.rubino,josef.vangenabith}@uni-saarland.de

Abstract

Previous linguistic research on scientific writing has shown that language use in the scientific domain varies considerably in register and style over time. In this paper we investigate the introduction of information theory inspired features to study long term diachronic change on three levels: lexis, part-of-speech and syntax. Our approach is based on distinguishing between sentences from 19th and 20th century scientific abstracts using supervised classification models. To the best of our knowledge, the introduction of information theoretic features to this task is novel. We show that these features outperform more traditional features, such as token or character n-grams, while leading to more compact models. We present a detailed analysis of feature informativeness in order to gain a better understanding of diachronic change on different linguistic levels.

1 Introduction

Supervised classification has been applied to various natural language processing tasks over the past decades. To date, however, distinguishing between time periods has not received extensive attention. Early research on classifying time periods is presented in de Jong et al. (2005) for Dutch. Dalli and Wilks (2006) and Kumar et al. (2011) use word frequencies for temporal classification of documents, while Sagi et al. (2009) and Kim et al. (2014) predict semantic changes over time. While lexical features are commonly used for classification approaches of time periods, features based on more abstract linguistic levels have not yet been widely investigated.

In our study, we use supervised classification to distinguish scientific abstracts written in the 19th and 20th century at the sentence-level. From previous work, we know that in the scientific domain, shared expertise among authors and audience affects their language use. Over a longer time period, it drives the evolution of domain-specific language with respect to lexis (Halliday, 1988; Teich et al., 2016) and a more standardized and convention-driven style with respect to grammar (Biber and Gray, 2011; Biber and Gray, 2016; Banks, 2005).

Considering that language variation affects *all* linguistic levels — from sounds and words to syntactic structure — we investigate a set of features extracted at the lexis, part-of-speech and syntactic levels to test how well they act as predictors of time period-specific language use. Moreover, based on psycholinguistic evidence it has been shown that language users choose those linguistic options that they know to be relatively predictable in a specific context to optimize communication (Hale, 2001; Levy, 2008; Demberg and Keller, 2008). To model communication in this sense, in our research we employ features based on the information-theoretic notion of *surprisal* or *information density*.

Specifically, we make use of information theory inspired features on the linguistic levels of lexis, part-of-speech and syntax. In addition, these features allow an unlexicalized dense-vector representation, which enormously reduces the amount of features used for classification. Besides achieving high performance in classification, we are particularly interested in insights on long-term diachronic linguistic change, which are important to historical linguistics, sociolinguistics and the like. We do this by inspecting classification results and discriminative features more closely.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 750–761, Osaka, Japan, December 11-17 2016.

Our analyses are driven by the following assumptions:

1. *Lexical diversification*: On the lexical level, scientific abstracts from the 19th and 20th century will be well distinguished from one another, due to topical changes in the scientific domain.
2. *Grammatical consolidation*: On more abstract linguistic levels, scientific abstracts from the 19th and 20th centuries will be less well distinguished from one another, as grammatical changes develop rather slowly over time, but we expect a tendency towards denser grammatical encodings in the 20th century texts.

The remainder of the paper is structured as follows. In Section 2, we present previous work on classification of time periods, diachronic change and information density. Section 3 describes the experimental setup up followed by the results and detailed analysis in Section 4. Finally, Section 5 provides a short summary and conclusions.

2 Related work

2.1 Classification of time periods

Classification of time periods has been less investigated so far in comparison to other classification tasks. Most of the existing work is based on lexical features and the classification of documents rather than individual sentences. In the study conducted by de Jong et al. (2005), the authors classify Dutch texts according to time (considering the time span 1999 to 2005) using uni-gram language models achieving around 65% accuracy. Dalli and Wilks (2006) (considering weekly to yearly levels, with an accuracy of the yearly classifier of $\sim$88%) and Kumar et al. (2011) (yearly classification) use classification methods based on word frequencies to determine the time period a text was written.

Other approaches – also based on lexical features – investigate semantic change over time. For instance, Sagi et al. (2009) focus on specific words to identify their semantic change from Early to Modern English. Mihalcea and Nastase (2012) use supervised learning to predict a word's time period given the context it occurs in. More recently, Kim et al. (2014) use neural language models to identify words that have changed semantically from 1900 to 2009. So far, only few studies have used features other than lexical ones for time period classification. Štajner and Zampieri (2013) have used stylistic features (such as average word and sentence length, pos tag n-grams, etc.) for classification of Portuguese texts into centuries achieving an F-measure of 0.92.

Besides the fact that most approaches use lexical features to predict time periods, the common experimental setup involves document-level classification of texts. To the best of our knowledge, there has been no work on sentence-based classification of time periods. Classifying sentences rather than texts allows us to build finer-grained classification models. In our approach, we classify sentences according to time periods going beyond lexis-based representations by using information theory inspired features, which inherently account for the context of use.

2.2 Information Density (ID)

Assuming that language users strive for efficient communication, they will tend to encode their message using an approximately uniform information density that exploits channel capacity while avoiding to overload the recipient or being uninformative. Information theory (Shannon, 1949) measures the amount of information conveyed by a unit in a given context in *bits* (Shannon, 1949). This notion is also known as *surprisal* (Levy, 2008) and is formulated as the negative log probability of a unit (e.g. a word) in context (e.g. its preceding words): $S(unit_i) = -\log p(unit_i|Context)$. Based on a limited context of size n words, the surprisal value of the following word w_{n+1} corresponds to the negative log-probability: $S(w_{n+1}) = -\log P(w_{n+1}|w_1 \dots w_n)$.

There are two properties inherent to surprisal: (1) units with low probability convey more information than those with high probability, and (2) information conveyed by a unit is crucially dependent on its context. Thus, linguistic units that are highly predictable in a given context convey less information with troughs in surprisal, while less predictable units convey more information with peaks in surprisal.

Over a longer period of time, the predictability of a word will change according to its use in specific contexts. In the scientific domain, shared expertise among researchers, for example, will affect language

use and give rise to domain-specific language. Particular words (e.g. terminology) will become more predictable over time (showing lower surprisal values) and may result in shorter encodings (consider e.g. acronym use in scientific fields such as genetics). Among researchers this will optimize communication. A more conventionalized use of scientific language will result in changes of surprisal values over time with conventionalized expressions (e.g. formulaic expressions) showing lower surprisal.

However, not only changes in lexis will be reflected in changes of surprisal values. From studies on language change, we know that diachronically there has been, for example, a shift from a more verbal towards a more nominal style (cf. notably Biber and Gray (2011)). This will have an impact on surprisal values with respect to grammatical units (such as parts of speech or syntactic units), motivating the use of information theory inspired features to classify between time periods.

So far, these kinds of features have been successfully used in classification of Gospels (see Islam and Dundia (2015) being able to identify the Greek Gospel as the original text and the American and Georgian ones as translations) and classification of human translated texts (see Rubino et al. (2016) distinguishing original from manually translated texts of different levels of expertise).

2.3 Language Change

Previous computational work on diachronic change in scientific language mostly discusses short-term change (see e.g. Blei and Lafferty (2006; 2007) on changes in scientific topics and Hall et al. (2008) on the ACL anthology corpus, both using topic models) rather than long-term change and is mostly concerned with change related to lexis (such as topical shifts) rather than change on more abstract linguistic levels.

In corpus-linguistic work on language change, approaches are typically frequency-based (e.g. Biber and Gray (2011; Biber and Gray (2013; Biber and Gray (2016), Taavitsainen and Pahta (2012), Moskowich and Crespo (2012)) and do not inherently account for context – diachronic change being observed through the lens of unconditioned probabilities. In contrast, information density measures as we apply them here, are based on conditional probabilities and thus inherently take context into account. Based on our previous work on long-term change using information-theoretic features (Degaetano-Ortlieb and Teich, 2016), we have shown how these features help model diachronic change, further motivating their use to classify different time periods.

3 Experimental Setup

The experiments presented in this paper focus on the use of sentence-level information density measures — in particular n-gram log-probabilities according to a language model and n-gram distribution according to frequency quartiles — to classify texts from different time periods. In this section, we present the supervised classification setup and the set of features as well as the data used.

3.1 Supervised Classification

A linear Support Vector Machine (SVM) (Cortes and Vapnik, 1995) is used to train our time-period classification model based on a feature representation of sentences which aims at capturing the density of information. All training, development and test sentences are represented as feature vectors $\mathbf{x}_i$, and the two corpora (19th century and 20th century) are associated with a class y_i, resulting in instance-label pairs $(\mathbf{x}_i, y_i)$ with $\mathbf{x}_i \in R^n$, and $y \in \{0,1\}^l$ as a binary classification task. We use the $L2$-regularized $L2$-loss SVC implementation of LIBLINEAR (Fan et al., 2008) to solve the following optimization problem:

$$min_w w^T \frac{w}{2} + C \sum_{i=1}^{l} max(0, 1 - y_i w^T x_i)^2 \quad (1)$$

The cost parameter C is selected with grid-search using the accuracy obtained on the held-out development set. Finally, the model is evaluated using the precision, recall, f-measure per class and general accuracy obtained on the test set.

Corpus	Sentences		Tokens		Types	
	19cA	20cA	19cA	20cA	19cA	20cA
Train	20.0	20.0	890.5	495.2	28.8	31.5
Development	1.5	1.5	66.3	36.7	7.7	7.2
Test	1.5	1.5	64.2	37.0	7.7	7.3

(a) Corpora used as training, development and test sets.

Corpus	Sentences	Tokens	Types
19cLM	623.2	16,541.7	320.0
20cLM	423.4	11,137.8	261.9

(b) Corpora used as resources to train the language models and to extract n-gram frequencies.

Table 1: Statistics (in thousands) of the corpora used in our experiments.

3.2 Datasets

Four corpora are used in our experiments, two for each time period (early 19th century: 1800-1850; late 20th century: 1970-2007). Two corpora compose our training, development and test sets (henceforth: 19cA and 20cA) while two others allow us to train language models and extract n-gram frequencies (henceforth 19cLM and 20cLM). Statistics about these corpora are presented in Table 1a and Table 1b.

For the 19th century time period, we use a corpus of research articles from the Royal Society of London (Kermes et al., 2016). Abstracts are taken from this corpus to form the 19cA classification subset. For feature extraction full research articles (19cLM) are taken from the same corpus, filtering out articles with abstracts included in 19cA. For the 20th century time period, abstracts are taken from a corpus of research articles (Degaetano-Ortlieb et al., 2013) covering several disciplines[1] as our 20cA classification subset. For feature extraction, we collected abstracts from several fields (20cLM) matching those of 20cA. The main difference between 19cLM and 20cLM is the type of document used to extract them, the former being composed of full articles due to research abstract scarcity for this time period, while the latter is composed of abstracts. The classification subsets (19cA and 20cA) are pre-processed by means of regular expressions and manually verified in order to remove headlines preceding abstracts, dates, formulas and mathematical expressions, etc.

3.3 Feature Sets

We consider three sets of features: shallow base-line features, n-gram frequency features, and information density features. Both n-gram and features specifically referred to as information density features capture aspects of information density and rely on the external resources presented in Table 1b.

Shallow features Here we consider popular lexical features such as bags of character and token n-grams as a baseline, as well as bags of part-of-speech (POS) n-grams ($n \in [1; 3]$). For POS tagging and syntactic parsing, we use the Stanford NLP toolkit (Manning et al., 2014).[2] For bags of token n-grams, three feature sets are built: one taking into account all n-grams, one considering n-grams appearing at least 200 times in the training corpus and one keeping only n-grams appearing at least 500 times, noted *Tokens All*, *Tokens 200* and *Tokens 500* respectively. The two latter sets allow for more compact models and less sparsity in the feature vectors. Additionally, 13 surface features are used, extracted from the surface-level of each sentence, which aim to capture meta representations of sentences' lexical form including sentence and average word lengths, the number of punctuation marks, letter and word casing, binary values encoding whether the sentence ends with a period and starts with an uppercase letter, etc.

N-gram Frequency Features To capture the rarity of n-grams used in the sentences to classify, the percentage of n-grams in frequency quartiles are extracted ($n \in [1; 5]$). The corpora used to model the frequency quartiles are the same resources as the ones used for the language models (19cLM and

[1] computer science, computational linguistics, bioinformatics, computer-aided design, microelectronics, mechanical engineering, electrical engineering, biology, linguistics

[2] We use the bidirectional maximum entropy POS tagger with a pre-trained English model based on the WSJ sections 0-18, including word shape and distributional similarity features. The probabilistic context free grammar lexicalized parser is used to obtain syntactic information from text (Manning et al., 2014).

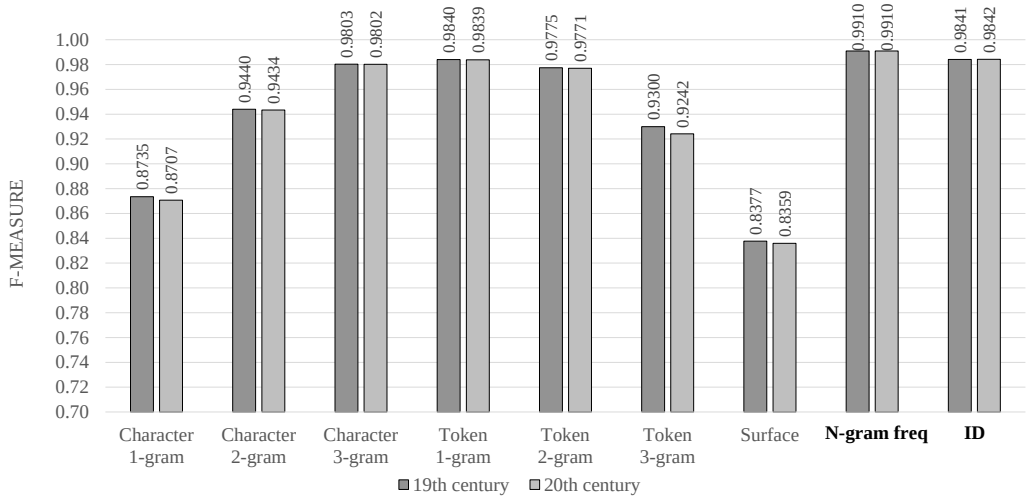


Figure 1: Classification results of 19th century and 20th century abstracts for lexis (LEX).

20cLM). Frequencies of word, part-of-speech and delexicalized flattened syntactic sequences are averaged at the sentence level, leading to 4 features per sentence (one per quartile) given one value of n, for each of the external resource used to model the quartiles (the corpora 19cLM and 20cLM). This approach leads to a dense representation of the information encoded in a sentence based on lexical, POS and syntactic information, without encoding raw word sequence n-gram features.

Information Density Features Using language models trained on sentences, delexicalized part-of-speech sequences and delexicalized flattened syntactic trees, a set of 120 sentence-level features are extracted: 15 features per individual LM resource (presented in Table 1b) and type of language model (lexical, POS and syntactic). We extract n-gram ($n \in [1; 5]$) log-probabilities (surprisal) as well as perplexities, with and without the tags indicating the beginning and ending of sentences, using the SRILM toolkit (Stolcke et al., 2011).

4 Results and Analysis

In the following, we present classification results of 19th vs. 20th century abstracts based on shallow as well as n-gram and information density features on three linguistic levels: lexis (LEX), part-of-speech (POS), and syntax (SYN). Moreover, by considering feature rankings obtained by the classification results, we analyze diachronic changes on these three linguistic levels.

4.1 Classification Results

Classification results on the lexical level (LEX) are shown in Figure 1. The bags of token n-grams features are unpruned (*Tokens All*). The best performing features are n-gram frequency (F-measure of 0.991) and ID features ranking second (0.984), both outperforming shallow features (bags of character and token n-grams, and surface features). Considering classification results on the part-of-speech level (POS), Figure 2 shows that POS 3-grams work best (0.9224) in classifying 19th c. and 20th c. abstracts, followed by POS 2-grams (0.9222) and ID features (0.9112).

Regarding classification at the syntactic level (SYN), Figure 3 shows that 19th c. and 20th c. abstracts are less well distinguished from one another in comparison to the lexical and part-of-speech level. Nevertheless, ID features work best on this task, achieving an F-measure of 0.88. Overall, ID features work relatively well on all three linguistic levels targeted in this study in comparison to other features, which work well on some levels (e.g. n-gram frequencies for lexis or POS 3-grams on the part-of-speech level), but less well on the other linguistic levels.

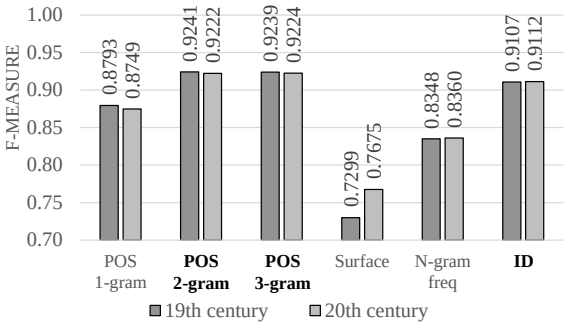


Figure 2: Classification results of 19th century and 20th century abstracts for part-of-speech (POS).

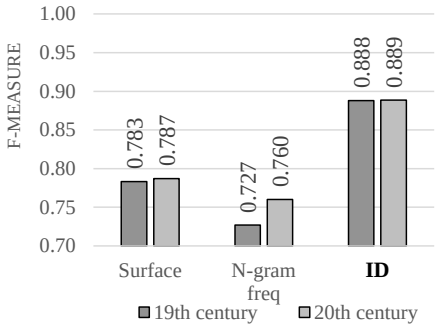


Figure 3: Classification results of 19th century and 20th century abstracts for syntax (SYN).

By considering combinations of feature sets, the best classification performance of 99.18 accuracy is obtained by a combination of n-gram frequency and ID features at all three linguistic levels (LEX, POS and SYN) (see Table 2). Moreover, looking at the number of features used for classification, better results can be obtained with a very small number of features (216) using n-gram frequency and ID features in comparison to the high number of features when using shallow features (e.g. more than 1 million features for surface tokens; see again Table 2). Pruning the low frequency n-grams considered in the bags of tokens features does not lead to accuracy improvement, but the resulting models are more compact with $3,081$ and 996 features for the *Tokens 200* and *Tokens 500* sets respectively.

			19cA			20cA		
Feature set	**Number of features**	**Accuracy**	**P**	**R**	**F**	**P**	**R**	**F**
Characters (LEX)	37,367	98.05	98.14	98.86	98.50	98.85	98.13	98.49
Tokens All (LEX)	1,896,263	98.12	97.43	98.84	98.13	98.82	97.39	98.10
Tokens 200 (LEX)	3,081	94.93	93.94	96.05	94.98	95.96	93.80	94.87
Tokens 500 (LEX)	996	90.38	89.68	91.26	90.46	91.10	89.50	90.29
POS-Tags (POS)	14,227	93.18	93.20	91.73	92.46	91.86	93.31	92.58
n-gram freq. + ID (LEX)	72	99.10	98.98	99.24	99.11	99.24	98.97	99.11
n-gram freq. + ID (POS)	72	88.85	93.20	91.73	92.46	91.86	93.31	92.58
n-gram freq. + ID (SYN)	72	88.67	91.38	89.10	90.22	89.37	91.59	90.47
n-gram freq. + ID (LEX, POS, SYN)	216	**99.18**	99.04	99.31	99.18	99.31	99.04	99.17

Table 2: Feature sets used in our experiments, number of features per set, accuracy obtained on the test set, as well as per class Precision (P), Recall (R) and F-measure (F).

4.2 Diachronic Changes at the Lexical Level

To investigate changes between 19th and 20th c. abstracts and evaluate the performance of the different feature types, we conduct a non-linear feature selection using the forest of randomized trees approach (Geurts et al., 2006) and describe the results for the top n-gram frequency and ID features in the paragraphs below. These two types of features are the focus of our study and lead to the best classification results as shown in Figure 1.

Lexis and N-gram frequency Inspecting the n-gram frequencies in detail based on feature ranking, we can observe general tendencies in lexis related change across the 19th and the 20th century. The highest ranking n-gram frequency features are based on 20cLM, comprising among the top 10 features 1- to 4-grams of very high (quartile 4) and low (quartile 1) frequency. This might be an indicator of conventionalized/formulaic language use with respect to high frequency high-order n-grams (4-grams quartile 4), on the one hand, and diversified language use with respect to low frequency low-order n-grams (1-grams quartile 1), on the other. Figure 4 shows the n-gram frequency distribution for 1- to

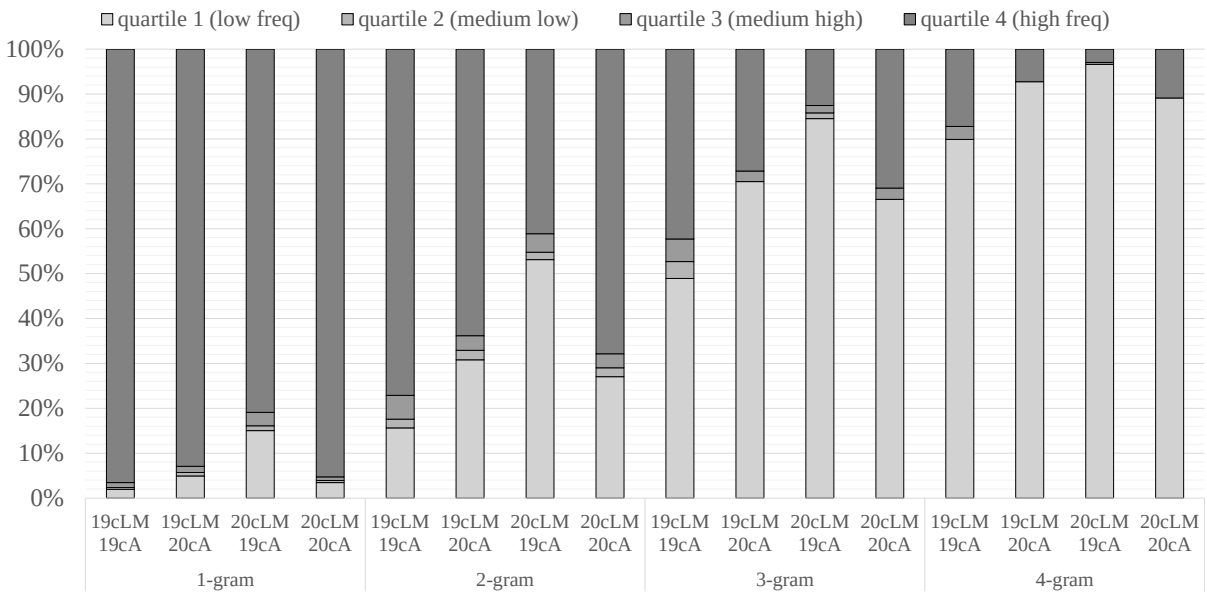


Figure 4: N-gram frequency distribution for 1- to 4-grams based on four frequency quartiles. 19cLM: 19th c. LM resource, 19cA: 19th c. abstracts, 20cLM: 20th c. LM resource, 20cA: 20th c. abstracts.

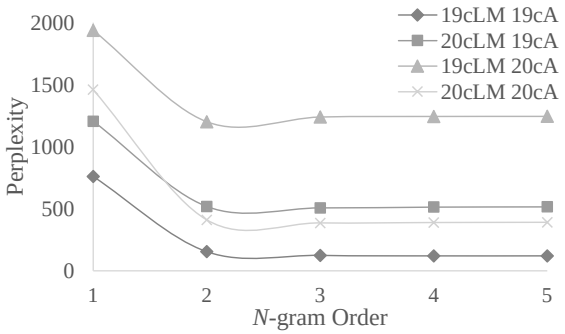


Figure 5: Averaged perplexity values obtained for 1- to 5-grams (LEX).

4-grams based on four quartiles (from high to low frequency) and on the resources used for language modeling (19cLM and 20cLM). Obviously, the higher the n-gram order, the higher the percentage of low frequency n-grams, i.e. rare n-grams. More specifically, Figure 4 shows that the 19cLM covers the 19th c. abstracts (19cA) quite well in terms of lexis, as the percentage of high frequency 1-grams (quartile 4) is high ($\sim$97%). The same can be observed for the 20cLM and the 20th c. abstracts (20cA) ($\sim$95%). However, while the 20cLM also covers relatively well the 19cA ($\sim$93% of high frequency 1-grams, quartile 4), the 19cLM on the 20cA shows a higher amount of low frequency 1-grams (quartile 1) covering high frequency 1-grams only by $\sim$80%. This indicates that while the vocabulary of 19th c. abstracts is relatively well covered by the 20th c. LM resource, the 20th c. abstracts make use of new words not covered by the 19th c. LM resource.

Considering 2-grams, which rank highest in classification, a similar but even more pronounced tendency can be observed, i.e. while the percentage of high frequency 2-grams is still relatively high for 20cLM and 19cA ($\sim$64%), 19cLM and 20cA show a relatively high amount of low frequency 2-grams ($\sim$53%). Nevertheless, the percentage of 20cLM and 19cA high frequency (quartile 4) n-grams remains higher than for 19cLM and 20cA. Thus, 20th c. abstracts have more diverse lexical n-grams than those written in the 19th c.

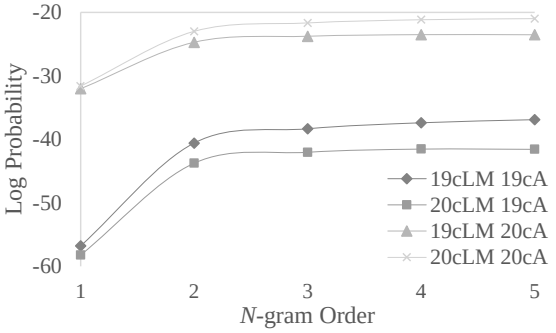


Figure 6: Averaged log probability values obtained for 1- to 5-grams (POS).

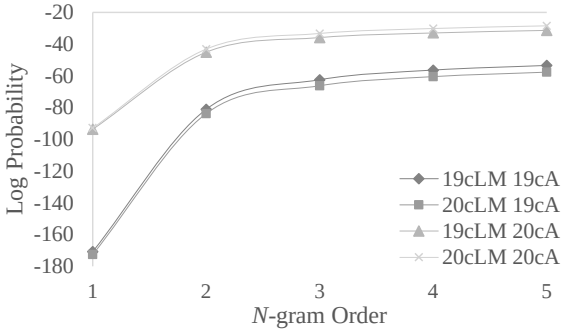


Figure 7: Averaged log probability values obtained for 1- to 5-grams (SYN).

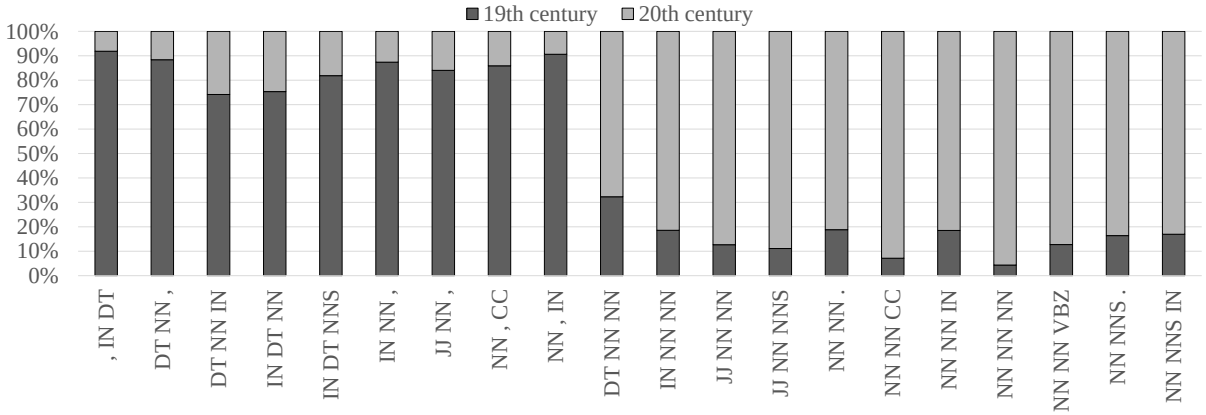


Figure 8: Distribution of top 20 POS 3-grams for the 19th c. and 20th c. abstracts. DT: determiner, CC: conjunction, JJ: adjective, IN: preposition, NN: singular common noun, NNS: plural common noun, VBZ: verb *be* present

In addition, the amount of high frequency (quartile 4) n-grams of the 19cLM and 19cA is higher than the ones for the 20cLM and 20cA. This might indicate a process of diversification in scientific writing, i.e. 19th c. texts in science are lexically closer than texts written in the 20th c.

Lexis and ID Feature ranking shows that perplexity values from 2- to 5-grams are the most discriminative ID features. Inspecting the perplexity values more closely (see Figure 5), we observe that from 2- to 5-grams 19cLM has relatively low average perplexity values for 19cA compared to the ones obtained for 20cA. While 19cLM is relatively close to 19cA, i.e. the abstracts' lexis is relatively predictable and obtains low perplexities according to the language model trained on 19cLM, lexis of 20th c. abstracts is less well predictable. This observation matches the results obtained with n-gram frequencies presented in Figure 4.

Considering 20cLM (see again Figure 5), it shows lower perplexity values for 20cA than for 19cA. However, the difference is relatively small in comparison to the difference observed for 19cLM on 19th and 20th c. abstracts. Thus, 20cLM is better in predicting lexical choices in both 19th c. and 20th c. abstracts compared to 19cLM. In terms of diachronic changes, this reflects how new lexical choices have entered scientific language, which were not present in the 19th century, while 19th c. language can still be understood by a contemporary language model. These results support the assumption of lexical diversification over time.

4.3 Diachronic Changes at More Abstract Linguistic Levels

	POS 3-gram	examples
19th century	, IN DT	*, that the*, *, in the*, *, on the*
	IN DT NN	*by the author*, *in this paper*, *of the earth*
	NN , CC	*water , and*, *acid , and*, *light , and*
	DT NN ,	*the author, , this paper, , the earth ,*
	DT NN IN	*the action of*, *the quantity of*, *the surface of*
20th century	JJ NN NN	*superficial gas velocity*, *natural language processing*, *optimal control problem*
	DT NN NN	*a computer program*, *the heat transfer*, *the nucleotide sequence*
	NN NN IN	*nucleotide sequence of*, *sequence analysis of*, *gene expression in*
	JJ NN NNS	*partial differential equations*, *open reading frames*, *linear matrix inequalities*
	NN NN .	*gene expression .*, *power consumption .*, *control system .*

Table 3: Most frequent lexical realizations of top 5 POS 3-grams for 20th c. and 19th c. abstracts

POS Sequences To investigate diachronic changes at the POS level, we consider POS 3-gram sequences, which perform best in POS-based classification (see again Figure 2). We inspect the top 20 features of the POS 3-gram sequences obtained by feature ranking and look at their frequency distribution in the training data of the 19th and 20th c. abstracts. Figure 8 shows how complex nominal structures (consisting of compounds with a at least two nouns, e.g. DT NN NN such as *the heat transfer*) are discriminative for the 20th c. abstracts, while shorter nominal structures (consisting of POS sequences with one noun, e.g. followed by a comma (DT NN , such as *the heat,*)) and prepositional phrases (e.g. IN DT NN such as *on the eye*) are discriminative for the 19th c. abstracts. This clearly reflects a shift towards a denser linguistic encoding in 20th c. abstracts, where information is more densely packed into longer nominal structures. Table 3 shows the top 5 POS 3-gram sequences of both periods with examples. We can see from the examples that while in the 19th c. there are relatively general and short nouns (such as *author*, *water*, *action*), in the 20th century more specific compound nouns are used. Inspecting these examples in their sentential context confirms the use of quite complex nominal phrases in abstracts of the 20th century (see examples (1) and (2)) vs. shorter, less complex ones in the 19th century (see examples (3) and (4)).

(1) *We have determined* ***the complete DNA nucleotide sequence of the carp Cyprinus carpio fast skeletal myosin heavy chain (MYH) gene***. (20th century)

(2) ***Nitric oxide generation rate and concentration distribution in combustors containing regions of recirculating flow*** *are calculated using* ***a computer program developed for two-dimensional elliptic compressible flows***. (20th century)

(3) *Those who cultivate* ***chemistry*** *with* ***any degree of ardour****, will be gratified to see in* ***this paper the pains*** *taken by* ***the author****, and* ***the various modes*** *he has devised, to produce* ***this compound metal*** *in its most perfect state of combination.* (19th century)

(4) ***The substance*** *here examined by* ***the author****, we are told, was first made known by* ***the celebrated Klaproth***. (19th century)

POS and ID We also inspect ID features as they achieve an F-measure above 0.90 (see Figure 2), indicating that 19th c. and 20th c. abstracts differ with respect to ID. The top three features refer to the log probabilities of 3- to 5-grams. In Figure 6, for both time periods the log probabilities increase with higher n-gram order, indicating lower surprisal values for POS n-grams of higher order. However, 19th c. abstracts differ from 20th c. abstracts as the log probabilities for the earlier period are lower (wrt both LM resources) than the ones for the 20th c. This indicates that POS sequences of 20th c. abstracts are

more predictable than those of the 19th c. abstracts, thus pointing to a more conventionalized use of POS sequences in 20th c. abstracts. These findings support our hypothesis of grammatical consolidation over time.

Syntax and ID The top three features on the syntactic level are again log probabilities of 3- to 5-grams. By inspecting the log probability distribution (see Figure 7), we see a very similar tendency to the results on POS. Thus, for both time periods the log probabilities increase with higher n-gram order, i.e. syntactic n-grams of higher order can be better predicted, indicating also on the syntactic level a more conventionalized use in 20th c. abstracts, which supports again our hypothesis of grammatical consolidation.

5 Conclusion

We have presented a sentence-based classification approach of time periods based on information theory inspired features. Our classification task focused on distinguishing 19th century and 20th century research abstracts. For this, we used features at three linguistic levels: lexis, part of speech, and flattened syntactic structure. This allows us to model not only lexical but also grammatical/stylistic long-term change in scientific writing.

Regarding classification, we show that while shallow features such as character and token n-grams achieve good results at the lexical level, applying features based on information density measures (log probability, perplexity) – achieves similar results and even outperforms shallow features at different linguistic levels. Furthermore, the best classification results were obtained by a combination of information density features considering all three linguistic levels with only a minimum number of features as we use unlexicalised dense feature-vector representations.

By a deeper analysis of the classification results, we obtained insights on long-term diachronic change with respect to our assumptions of *lexical diversification* and *grammatical consolidation*. Considering lexical diversification, new lexical choices have entered scientific writing from 19th to 20th century. Considering grammatical consolidation, a trend towards a denser linguistic encoding in terms of compact nominal structures was observed. Beyond lexical variation, we assume the methodology to be domain- and language independent. This will be pursued in future work with application on other genres/registers and languages.

Acknowledgments

This research is funded by the German Research Foundation (Deutsche Forschungsgemeinschaft) under grant SFB 1102: Information Density and Linguistic Encoding[3] and EXC-MMCI: Multimodal Computing and Interaction[4]. We would like to thank the anonymous reviewers for their insightful comments.

References

David Banks. 2005. On the Historical Origins of Nominalized Process in Scientific Text. In *English for Specific Purposes 24 (3), 347-357*.

Douglas Biber and Bethany Gray. 2011. The Historical Shift of Scientific Academic Prose in English towards Less Explicit Styles of Expression: Writing without Verbs. In Vijay Bathia, Purificación Sánchez, and Pascual Pérez-Paredes, editors, *Researching Specialized Languages*, pages 11–24. John Benjamins.

Douglas Biber and Bethany Gray. 2013. Being Specific about Historical Change: The Influence of Sub-register. *Journal of English Linguistics*, 41:104–134.

Douglas Biber and Bethany Gray, editors. 2016. *Grammatical Complexity in Academic English: Linguistic Change in Writing*. Cambridge University Press.

David M. Blei and John D. Lafferty. 2006. Dynamic Topic Models. In *Proceedings of ICML*, pages 113–120.

[3] IDEAL – http://www.sfb1102.uni-saarland.de/

[4] http://www.mmci.uni-saarland.de

David M. Blei and John D. Lafferty. 2007. A Correlated Topic Model of Science. *The Annals of Applied Statistics*, 1(1):17–35.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector Networks. *Machine learning*, 20(3):273–297.

Angelo Dalli and Yorick Wilks. 2006. Automatic Dating of Documents and Temporal Text Classification. In *Proceedings of the ACL Workshop on Annotating and Reasoning about Time and Events*, pages 17–22.

Franciska de Jong, Henning Rode, and Djoerd Hiemstra. 2005. Temporal Language Models for the Disclosure of Historical Text. In *Humanities, Computers and Cultural Heritage: Proceedings of the XVIth International Conference of the Association for History and Computing (AHC 2005)*, pages 161–168.

Stefania Degaetano-Ortlieb and Elke Teich. 2016. Information-based Modeling of Diachronic Linguistic Change: From Typicality to Productivity. In *Proceedings of the 10th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 165–173.

Stefania Degaetano-Ortlieb, Hannah Kermes, Ekaterina Lapshinova-Koltunski, and Elke Teich. 2013. SciTex - A Diachronic Corpus for Analyzing the Development of Scientific Registers. In Paul Bennett, Martin Durrell, Silke Scheible, and Richard J. Whitt, editors, *New Methods in Historical Corpus Linguistics*, volume 3 of *Corpus Linguistics and Interdisciplinary Perspectives on Language - CLIP*, pages 93–104. Narr.

Vera Demberg and Frank Keller. 2008. Data from Eye-tracking Corpora as Evidence for Theories of Syntactic Processing Complexity. *Cognition*, 109(2):193–210.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely Randomized Trees. *Machine learning*, 63(1):3–42.

John Hale. 2001. A Probabilistic Earley Parser as a Psycholinguistic Model. In *Proceedings of NAACL*, volume 2, pages 159–166.

David Hall, Daniel Jurafsky, and Christopher D. Manning. 2008. Studying the History of Ideas Using Topic Models. In *Proceedings of EMLNP*, pages 363–371.

M.A.K. Halliday. 1988. On the Language of Physical Science. In Mohsen Ghadessy, editor, *Registers of Written English: Situational Factors and Linguistic Features*, pages 162–177. Pinter.

Zahurul Islam and Natia Dundia. 2015. Finding the Origin of a Translated Historical Document. In *Proceedings of PACLIC*, pages 96–105.

Hannah Kermes, Stefania Degaetano-Ortlieb, Ashraf Khamis, Jörg Knappen, and Elke Teich. 2016. The Royal Society Corpus: From Uncharted Data to Corpus. In *Proceedings of LREC*, pages 1928–1931.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal Analysis of Language through Neural Language Models. In *Proceedings of the ACL Workshop on Language Technologies and Computational Social Science*, pages 61–65.

Abhimanu Kumar, Matthew Lease, and Jason Baldridge. 2011. Supervised Language Modeling for Temporal Resolution of Texts. In *Proceedings of CIKM*, pages 2069–2072.

Roger Levy. 2008. A Noisy-channel Model of Rational Human Sentence Comprehension under Uncertain Input. In *Proceedings of EMNLP*, pages 234–243.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of ACL: System Demonstrations*, pages 55–60.

Rada Mihalcea and Vivi Nastase. 2012. Word Epoch Disambiguation: Finding How Words Change Over Time. In *Proceedings of ACL*, pages 259–263.

Isabel Moskowich and Begoña Crespo, editors. 2012. *Astronomy 'playne and simple'. The Writing of Science between 1700 and 1900*. John Benjamins.

Raphael Rubino, Ekaterina Lapshinova-Koltunski, and Josef van Genabith. 2016. Information Density and Quality Estimation Features as Translationese Indicators for Human Translation Classification. In *Proceedings of NAACL*, pages 960–970.

Eyal Sagi, Stefan Kaufmann, and Brady Clark. 2009. Semantic Density Analysis: Comparing Word Meaning across Time and Phonetic Space. In *Proceedings of the EACL Workshop on GEMS: GEometical Models of Natural Language Semantics*, pages 104–111.

Claude E. Shannon. 1949. *The Mathematical Theory of Communication*. University of Illinois Press, 1983 edition.

Sanja Štajner and Marcos Zampieri. 2013. Stylistic Changes for Temporal Text Classification. In *Proceedings of the International Conference on Text, Speech and Dialogue, Lecture Notes in Artificial Intelligence (8082)*, pages 519–526. Springer.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. SRILM at Sixteen: Update and Outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, volume 5.

Irma Taavitsainen and Päivi Pahta, editors. 2012. *Early Modern English Medical Texts. Corpus Description and Studies*. John Benjamins.

Elke Teich, Stefania Degaetano-Ortlieb, Peter Fankhauser, Hannah Kermes, and Ekaterina Lapshinova-Koltunski. 2016. The Linguistic Construal of Disciplinarity: A Data Mining Approach Using Register Features. *Journal of the Association for Information Science and Technology (JASIST)*, 67(7):1668–1678.

Different Contexts Lead to Different Word Embeddings

Wenpeng Hu, Jiajun Zhang*, Nan Zheng*
Institute of Automation, Chinese Academy of Sciences, China
{wenpeng.hu,jjzhang}@nlpr.ia.ac.cn
nan.zheng@ia.ac.cn

Abstract

Recent work for learning word representations has applied successfully to many NLP applications, such as sentiment analysis and question answering. However, most of these models assume a single vector per word type without considering polysemy and homonymy. In this paper, we present an extension to the CBOW model which not only improves the quality of embeddings but also makes embeddings suitable for polysemy. It differs from most of the related work in that it learns one semantic center embedding and one context bias instead of training multiple embeddings per word type. Different context leads to different bias which is defined as the weighted average embeddings of local context. Experimental results on similarity task and analogy task show that the word representations learned by the proposed method outperform the competitive baselines.

1 Introduction

Different from traditional one-hot sparse vector representation, word embeddings are dense and low-dimensional. Due to natural advantage in word similarity computation, word embeddings are useful in a variety of applications, such as information retrieval (Uddin et al., 2013; Ganguly et al., 2015), sentiment analysis (Santos et al., 2014; Nguyen et al., 2015), question answering (Tellex et al., 2003) and parsing (Socher et al., 2013). Researchers learn word embeddings in various ways. Matrix Factorization methods for generating dense word embeddings have been used for years (Lund et al., 1996). While in recent years, neural network and deep learning have become popular approaches for learning word embeddings since Bengio et al. (2003) introduced feed forward neural network into traditional n-gram language models. For example, Collobert and Weston proposed a new objective function to learn word embeddings and improved word embeddings' quality (Collobert et al., 2008). Huang et al. (2012) presented a new neural network architecture which incorporated both local and global document context, and offered an impressive result. The word2vec toolkit developed by (Mikolov et al., 2013a; Mikolov et al., 2013b) implemented both Skip-gram and CBOW models which could provide high-quality word embeddings.

In spite of many successful applications, most word embeddings have a common problem that each word is represented by a single vector, subsequently ignoring polysemy. For example, the word *bank* cannot have high cosine similarity with the word *river* and *money* at the same time since these two words are so dissimilar and the single vector representation of word *bank* cannot express two different meanings. Thus, several multi-prototype models have been proposed to alleviate the problem caused by the polysemy and homonymy. Guo et al. (2014) took advantages of bilingual resources and affinity propagation clustering algorithm to learn multiple embeddings corresponding with multiple word senses. Due to the limitation of bilingual resources, they couldn't train their model on large scale corpus. Huang et al. (2012) pre-clustered the contexts of a word into K classes, and then learned K embeddings per word. K was a predefined value that would make matters confusion because different words usually had different number of senses. Neelakantan et al. (2014) shifted clusters into the training progress and

* The authors contributed equally to this paper

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 762–771, Osaka, Japan, December 11-17 2016.

proposed a non-parametric clustering model which could dynamically generate new cluster center by similarity threshold λ and thus could learn different number of embeddings per word type. Although they solved the problem caused by predefined K cluster centers, hyper-parameter λ was still hard to be defined, because the similarity thresholds for different words might be different. Moreover, they were sensitive to noise. Once a noise point happened, a new cluster center would be created. Therefore it would lead to mistakes due to no actions were taken to remove these incorrect clusters. Yang et al. (2016) proposed a supervised fine tuning framework to transform the existing single-prototype word embeddings into multi-prototype word embeddings based on lexical semantic resources. It was a good idea to obtain multi-prototype word embeddings, but it depended on the existing embeddings which had lost some information during the training process.

To the best of our knowledge, most of the prior work deal with the polysemy problem by learning multiple embeddings per word type. In this paper, we present a novel approach to take the polysemy phenomena into consideration. It is based on the assumption that the same word in different contexts has different meanings and so the polysemy problem appears. Traditional single word embedding is approximately the weighted mean of its different contextual semantics. Polysemic words owning the same appearance may mean that they share a common intrinsic quality, and the differences among them are reflected by different contexts. In other words, the different contexts of a word exert different influences, and the given context is a good indication of the direction of the word meaning. Therefore, we propose a method to model the intrinsic quality of words and obtain the influences affected by contexts.

In this paper, we make the following contributions:

- The composition and structure of our word embeddings are the word center adding the offset induced by the word contexts, which make it flexible to deal with the polysemy problem.
- There is no need to cluster before or during training process, which removes the inaccuracy caused by clustering. Each component of our word embeddings can be updated or trained thousands of times during the training process which makes it robust and less susceptible to outliers.
- We quantify the influencing degree of the local contexts by training weighted parameters. In this way, the first part in our word embeddings named word center can be trained more accuracy and less ambiguous.

2 Background: CBOW (Continuous Bag-of-Words Model)

As a popular toolkit provided by Mikolov et al., word2vec[1] has gained lots of attractions in recent years. Two models CBOW and Skip-gram (Mikolov et al., 2013a; Mikolov et al., 2013b) are used, which make it possible to train on more than 100 billion words in one day and generate high-quality word embeddings. Word2vec provides two frameworks for these two models. One is based on Hierarchical Softmax and the other is based on Negative Sampling. According to (Mikolov et al., 2013b), Negative sampling (NEG) is a simplification of Noise Contrastive Estimation (NCE) (Gutmann et al., 2012), which can improve both of the training speed and the quality of word embeddings. So we choose Negative sampling to optimize our model.

The CBOW model uses the contexts of word w to predict w. Thus, for the given contexts, word w is a positive sample and other words are negative samples. In the CBOW model, $v(w) \in R^d$ is the word embedding of word $w \in W$, where W is the word vocabulary and d is the dimension of the embedding. $C(w) \in R^d$ is the average embedding of the context for word w.

Given the context of word w, the probability that a word u can be observed is given by,

$$P(u|Context(w)) = \sigma(C(w)^T\theta^u) = \frac{1}{1+e^{-C(w)^T\theta^u}} \tag{1}$$

where θ^u are training parameters of word u according to the source code of Word2vec. For the positive samples where $u = w$, the CBOW model is maximizing formula (1), while for the negative samples

[1] https://code.google.com/p/word2vec/.

$u \in NEG(w)$ the model will minimize it which can be done as maximize $1 - \sigma(C(w)^T \theta^u)$. $NEG(w)$ is the sets of negative samples for word w. So rewrite formula (1), given a samples u, CBOW model will maximize:

$$g(w) = \prod_{u \in w \cup NEG(w)} p(u|Context(w)) \tag{2}$$

where

$$p(u|Context(w)) = \begin{cases} \sigma(C(w)^T \theta^u), & L^w(u) = 1; \\ 1 - \sigma(C(w)^T \theta^u), & L^w(u) = 0, \end{cases}$$

$$L^w(u) = \begin{cases} 1, & u = w; \\ 0, & u \neq w, \end{cases}$$

Then the whole expression of $g(u)$ can be written as:

$$g(w) = \prod_{u \in w \cup NEG(w)} [\sigma(C(w)^T \theta^u)]^{L^w(u)} \cdot [1 - \sigma(C(w)^T \theta^u)]^{1 - L^w(u)}$$

Given a training corpus $\mathcal{C}$, the word embeddings are learned by maximizing the following objective function:

$$G = \prod_{w \in \mathcal{C}} g(w) \tag{3}$$

For compute easily, the objective function take the $\log$ and switchs from product to sum:

$$\begin{aligned} \mathcal{L} = \log G &= \log \prod_{w \in \mathcal{C}} g(w) = \sum_{w \in \mathcal{C}} \log\ g(w) \\ &= \sum_{w \in \mathcal{C}} \sum_{u \in \{w\} \cup NEG(w)} \mathcal{L}(w, u) \end{aligned}$$

where

$$\mathcal{L}(w, u) = L_w(u) \cdot \log[\ \sigma(\ C_w^T\ \theta^u)] + [1 - L^w(u)] \cdot \log[1 - \sigma(C_w^T\ \theta^u)] \tag{4}$$

For the negative sampling method, Mikolov et al. found that the powered unigram distribution $U(m)^{3/4}$ outperformed significantly the unigram and the uniform distributions. The distribution that the noisy contextual words are randomly sampled by is as following:

$$P(w) = \frac{p_{unigram}(w)^{3/4}}{Z} \tag{5}$$

where $p_{unigram}(w)^{3/4}$ is the number of occurrences of the words and Z is the normalization constant.

3 The RLC (Reuse Local Context) Model

3.1 The Embedding Structure of RLC

Different from traditional one-hot vector representation approach, each dimension of word embedding represents a latent feature of the word, mentioned by (Guo et al., 2014). And the word analogical task introduced by (Mikolov et al., 2013a) means that the influence between words can be simply computed by addition and subtraction. Because of these, we can assume that the dimensions between a word embedding are relatively independent which means there are no cross influence between them. Then we introduce a weighted parameter for each word as a filter to control the influence degree given by the context of the word. And the center of the word embedding will be shifted after adding the contextual influences to word embeddings.

In our approach, the word embedding structure of word w can be written as following:

$$[Xe_w; Xm_w] \tag{6}$$

where $Xe_w \in R^d$ is the word embedding center point of word w, $Xm_w \in R^d$ is the weighted parameters.

Using this structure can obtain different word embeddings according to different contexts. If there are no contextual words, the center point of word embedding Xe_w can be used instead. Considering a context and a word sampled from it, it is impossible for the context to have negative influence to the word. So we use the sigmoid function to limit Xm_w within the range of 0 to 1. And the combination formula of word embedding for w is as follows:

$$X_w = Xe_w + \sigma(Xm_w) \cdot \frac{1}{m} \sum_{i=1}^{m} Xe^w{}_i \tag{7}$$

where m represents the number of words in the context of w; $Xe^w{}_i$ is the *i*th word in the context of w.

3.2 The Algorithm of RLC

We introduce our algorithms into CBOW model, named RLC on CBOW (RLCC). Then, we build our objective function. Unfortunately, the probability computing method that used by CBOW model is not suitable for our approach according to formula (1-3). Because they calculate the probability of word u given context $Context(w)$ using inner product of $C(w)$ and θ^u. While, θ^u are training parameters corresponding to the sample word u, which are of no use after training. So we cannot combine the influences of the given context with parameters u. If we combine every word in $Context(w)$ by formula (7), then the computational complexity of our model will be heavily increased. But we find that the function of θ^u is semantic representation for word u and it is also a kind of word embedding for word u which can be replaced by the true word embedding. So we update the probability $p(u|Context(w))$ by:

$$p(u|Context(w)) = \begin{cases} \sigma(C(w)^T X_u), & L^w(u) = 1; \\ 1 - \sigma(C(w)^T X_u), & L^w(u) = 0, \end{cases} \tag{8}$$

where

$$X_u = Xe_u + \sigma(Xm_u) * Xc$$

$$Xc = \frac{1}{m} \sum_{i=1}^{m} Xe^w{}_i = C(w) \tag{9}$$

where "$*$" denotes the product of corresponding vector components, Xc denotes the average embeddings for the current context of word w.

Negative sampling maximizes the probability of positive samples given the context and at the same time minimizes the negative ones. But there are some problems for our approach to simply use Negative sampling. In the training process, negative samples are treated completely irrelevant with the given context, and the probability that they are observed would be minimized. This is beneficial for the model to differentiate data from noise. While from the right part in formula (7) recorded as $X_{influence} = \sigma(Xm_w) \cdot \frac{1}{m} \sum_{i=1}^{m} Xe^w{}_i$ we can see that minimizing observed probability will be achieved by decreasing Xm_w, and this satisfies our goal. Unfortunately, the decreased value of $\sigma(Xm_w)$ for every iteration is $||C(w)||_2^2$ which has a mean greater than zero and should not be considered as probability. Meanwhile, according to the paper of (Mikolov et al., 2013b), we should ensure the effect that the number of negative samples should be in the range of 5-20 for small training datasets and in the range of 2-5 for large datasets. That means the number of negative samples is at least two times bigger than positive samples which makes matter worse. So we introduce extra learning rates to solve the problem:

$$\eta = L^w(u) * \alpha + \beta \tag{10}$$

where α and β are hyperparameters.

The influences of context to positive samples must be positive, so we introduce the sigmoid function to limit Xm_u within the range of 0-1. But it is not reasonable for negative samples, because the influences of context to negative samples could be negative. Meanwhile, Xc given as in formula (9) still has strong

correlation with the context embedding $C(w)$ after multiply by the weight $\sigma(Xm_u)$ which is harmful to the update of Xe_u in positive samples during the training process. We introduce new extra weighted parameters Tm during training process to solve the problem which can be seen as the weight for negative samples:

$$Xm_{u\_new} = (1-\gamma) \cdot Tm_u + \gamma \cdot \sigma(Xm_u) \tag{11}$$

where $\gamma = L^w(u) * (1-\lambda) + (1-L^w(u)) * \lambda$,$\lambda \in [0,1]$. For example, if we set $\lambda = 0$, then $\sigma(Xm_u)$ will be used as the weighted parameter when the sample is positive one.

In this case, formula (7) should be rewritten according to formula (11):

$$X_w = Xe_w + Xm_{w\_new} \cdot \frac{1}{m} \sum_{i=1}^{m} Xe^w{}_i \tag{12}$$

Combining the given objective function (3-4) and the above analysis, we can infer the update gradient in each iteration as follows and the pseudo-code is showing in the Algorithm 1.

$$\begin{aligned}
\frac{\partial \mathcal{L}(w,u)}{\partial Xe^w} &= [L^w(u) - \sigma(C(w)^T X_u)] * \frac{1}{m} * (\, X_u + C(w) * Xm_{u_new}) \\
\frac{\partial \mathcal{L}(w,u)}{\partial Xe_u} &= [L^w(u) - \sigma(C(w)^T X_u)] * C(w) \\
\frac{\partial \mathcal{L}(w,u)}{\partial Xm_u} &= [L^w(u) - \sigma(C(w)^T X_u)] * C(w) * Xc * \sigma(Xm_u) * (1-\sigma(Xm_u)) * \eta * \gamma \\
\frac{\partial \mathcal{L}(w,u)}{\partial Tm_u} &= [L^w(u) - \sigma(C(w)^T X_u)] * C(w) * Xc * (1-\gamma)
\end{aligned} \tag{13}$$

Algorithm 1 : Training Algorithm of RLC model

1. Input : Corpus $\mathcal{C}$.
2. Initialize : $Xw \; and \; Tw, \forall \; w \in Vocab, Xw \in R^{2d}, Tw \in R^d$, randomly.
3. **while** next sentence in $(\mathcal{C})$ not null **do**
4. $\quad e = 0.$
5. $\quad C_w = \frac{1}{m} \sum_{u \in Context(w)} Xe_u$
6. $\quad$ **for** $u \; = \; \{w\} \cup NEG(w)$ **do**
7. $\quad\quad Xm_{u\_new} = (1-\gamma) \cdot Tm_u + \gamma \cdot \sigma(Xm_u)$
8. $\quad\quad X_u = Xe_u + Xm_{u\_new} C_w$
9. $\quad\quad q = \sigma(C_w^T X_u)$
10. $\quad\quad g = \eta_1 (L^w(u) - q)$
11. $\quad\quad e := e + g(\, X_u + C_w Xm_{u\_new})/m$
12. $\quad\quad Xe_u := Xe_u + gC_w$
13. $\quad\quad Xm_u := Xm_u + g{C_w}^2 \sigma(Xm_u)(1-\sigma(Xm_u))\eta\gamma$
14. $\quad\quad Tm_u := Tm_u + g{C_w}^2 (1-\gamma)$
15. $\quad$ **end for**
16. $\quad$ **for** $u \in Context(w)$ **do**
17. $\quad\quad Xe_u := Te_u + e$
18. $\quad$ **end for**
19. **end while**

4 Experiments

Following previous work (Huang et al., 2012; Neelakantan et al., 2014), we choose Wikipedia corpus[2] (Shaoul et al., 2010) snapshotted at April 2010 to train embeddings. The Wikipedia corpus contains a total of about 2 million articles and 990 million tokens. In all of our experiments we remain 220682 words as the vocabulary by removing all the words with less than 55 occurrences and set the maximum context window ($m/2$) to be 8 which means 8 words before and after the word occurrence. Our hyper-parameter values are selected by manual exploration of results when using a small corpus attached by Word2vec toolkit, named text8 which has 31893 words in the vocabulary and 16 million words in the corpus after removing the words with the occurrences less than 20. In RLCC we set $\lambda = 0.5,\ \alpha = 0.4,\ \beta = 0.6$. Also we train embeddings in different dimensions and compare our methods with many state-of-the-art models.

Table 1 shows the training time of our models on the small data set text8, compared with other models from previous work. All the methods are evaluated using a single-machine with 16 threads. We see that the training speed of our model is faster than Skip-gram model but is slower than CBOW model. This indicates that our model can be acceptable with respect to efficiency.

Model	**Time (minute second)**		
	dim-50	dim-100	dim-300
CBOW	55s	1m10s	2m33s
Skip-gram	5m35s	7m32s	18m57s
RLCC	1m52s	3m1s	7m14s

Table 1: Training Time Comparison

4.1 Word Similarity

The two datasets that we used to evaluate our word embeddings are as follows: the WordSim-353 (Finkelstein et al., 2001) dataset and the Contextual Word Similarities (SCWS) dataset (Huang et al., 2012). WordSim-353 is a standard dataset for evaluating word embeddings which consists of 353 pairs of word types. The similarity of each word pair is manually rated in a scale from 0 to 10 by 13 to 16 human judgements, and each pair receives an average score. But these scores are given without any information of the context which makes our embeddings hard to use. Fortunately, we are surprised to see that our embeddings outperform competitive baselines and even the state-of-the-art embeddings by only using the word center embeddings Xe.

Because of the absence of contextual information in the WordSim-353 dataset, Huang et al. (2012) developed Stanford's Contextual Word Similarities (SCWS) dataset which consists of 2003 word pairs and associated sentential contexts. So the SCWS dataset overcomes the issue caused by no contextual information in WordSim-353 and the models designed to deal with polysemous problems could have a good testbed. Most of those models train multiple embeddings per word type to tackle the polysemous problems and there are many approaches to measure the similarities of multi-prototype embeddings according to (Reisinger et al., 2010). Then we select the best results for each model to compare with our model.

Table 2 shows our results compared to previous methods on WordSim-353 dateset and Table 3 gives the results on SCWS dataset. All the models mentioned are trained in the same Wikipedia corpus snapshotted at April 2010. The scores Huang et al. we used in the table is using the word embeddings trained and provided by (Huang et al., 2012). MSSG and NP-MSSG are the best scores provided by (Neelakantan et al., 2014). We train Skip-gram and CBOW models using 10 negative samples and a context window size of 8 on the same corpus. RLCC* is our model, and the results shown in the table are gained by only using Xe in our models. Some results are not provided in these tables because we could not find the embeddings or results provided by the authors. From Table 2 we can see that our model is able

[2]http://nlp.stanford.edu/data/WestburyLab.wikicorp.201004.txt.bz2

Model	Different dimensions		$\rho \times 100$
	dim-50	dim-100	dim-300
Huang et al.	64.2	-	-
C&W	55.3	-	-
MSSG	63.2	-	70.9
NP-MSSG	62.4	-	68.6
CBOW	65.0	66.5	65.8
Skip-gram	68.6	71.7	72.7
RLCC*	**70.8**	**73.0**	**75.0**

Table 2: Experimental Results on the WordSim-353 dataset. The numbers in the table are Spearmans correlation recorded as $\rho \times 100$ between the model's similarities and human judgments. The best results according to each dimension are in bold face.

to learn more semantic word embeddings and noticeably improves upon previous models. We believe that the improvement is mainly attributed to the fact that the learned center word embedding Xe is less ambiguous.

Table 3 shows that our model gets the best result on both the 50-dimension embeddings and the 300-dimension embeddings. Obviously, our approach can better deal with polysemous problems than the traditional multi-prototype word embeddings.

Model	Dim	NUContext	UContext
C&W	50	57.0	-
CBOW	50	64.7	-
Skip-gram	50	63.4	-
Huang et al.	50	58.6	65.7
MSSG	50	62.1	66.9
NP-MSSG	50	62.3	66.1
RLCC	50	65.3	67.3
CBOW	100	65.8	-
Skip-gram	100	64.9	-
RLCC	100	66.0	68.4
CBOW	300	66.6	-
Skip-gram	300	66.7	-
MSSG	300	65.3	69.3
NP-MSSG	300	65.5	69.1
RLCC	300	**67.6**	**69.7**

Table 3: Experimental results on the SCWS dataset. "NUContext" is the abbreviation for "Not use the contextual information". For those multi-prototype embeddings, the $globalSim$ metric which is each word's global context vector ignoring the many senses is used. While in our model, we still use Xe for each word, ignoring the weighted parameters. "UContext" is the abbreviation for "Use the contextual information". In this time, for multi-prototype embeddings models, $avgSimC$ method is applied which weights the similarity by how well each sense fits the context at hand (Neelakantan et al., 2014). All of the best results for each dimension are marked by underline, while in the two situations of using context or not, they are marked by bold.

4.2 Word Analogies

The word analogy task is introduced by (Mikolov et al., 2013a). It is a comprehension test task which is designed to measure the quality of word embeddings. This task consists of questions like, "a is to b

as c is to ___ ?". The dataset we applied in this task is built by Mikolov et al., which contains five types of semantic questions and nine types of syntactic questions with 8869 and 10675 questions respectively. The questions, for example, "Tokyo is to Japan as Beijing is to ___ ?", are answered by finding the word whose embeddings are the most similar one with $W_{Japan} - W_{Tokyo} + W_{Beijing}$ under cosine similarity.

According to (Neelakantan et al., 2014), both MSSG and NP-MSSG models achieve 64% accuracy which exceed (Huang et al., 2012) but are worse than the Skip-gram model. So we compare our model with CBOW and Skip-gram models only in this task. The results shown on Table 4 demonstrate that our model outperforms both the CBOW and Skip-gram models in most words' analogy task especially in answering semantic questions. It means that our embeddings are less constrained by syntax and are more semantic.

Model	Dim	Sem.	Syn.	Tot.
CBOW	50	59.6	59.4	59.5
Skip-gram	50	52.5	55.9	54.6
RLCC*	50	66.3	60.1	62.5
CBOW	100	72.8	70.2	71.2
Skip-gram	100	67.8	69.2	68.6
RLCC*	100	80.0	70.8	74.3
CBOW	300	77.9	**75.4**	76.4
Skip-gram	300	83.1	73.4	77.2
RLCC*	300	**90.7**	71.8	**79.0**

Table 4: Experimental results of the word analogy task show as percent accuracy. We trained CBOW and Skip-gram using the same corpus as our model used, and the training parameters were also the same as we described before.

5 Related Work

In recent years, neural network and deep learning have become popular approaches for learning word embeddings, which make it possible to study dense and high quality word embeddings. Bengio et al. (2003) introduced feed forward neural network into traditional n-gram language models, which might be the foundation work for neural network language models(NNLM). In NNLM, words were represented by a low-dimensional vector and the parameters could be learned in unsupervised methods. Collobert et al. (2008) proposed a new objective function to learn word embeddings instead of the time consuming softmax layer presented in (Bengio et al., 2003) and much improved training speed. Mnih et al. (2007) reduced the computational complexity of the Bengio's model by replacing the softmax layer with a tree-structured probability distribution.

Mikolov et al. (2013a) and Mikolov et al. (2013b) removed the hidden layers of neural network and proposed log-liner neural language models named Skip-gram and CBOW. These two models extremely reduced the computational complexity and could train word embeddings on more than 100 billion words in one day with a single machine. With the help of negative sampling method, both of the two models could obtain state-of-the-art word embeddings.

In the previous work, several multi-prototype models have been proposed to alleviate the problem caused by the polysemy and homonym. Guo et al. (2014) took advantages of bilingual resources and affinity propagation clustering algorithm to learn multiple embeddings corresponding with multiple word senses, because a polysemous word in one language could not be exactly a polysemous word in another language. Huang et al. (2012) pre-clustered the corpus into specified classes, and relabeled the tokens into different classes, then learned specified numbers of embeddings per word type. The number of each word senses was predefined as a fixed value that would make matters confusion because the different words might have different number of senses. Neelakantan et al. (2014) shifted clusters into the training progress and proposed a non-parametric clustering model which could dynamically generate new clusters based on word meaning. Fine tuning was also a good idea to generate muti-prototype word embeddings.

Yang et al. (2016) proposed a supervised fine tuning framework to transform the existing single-prototype word embeddings into multi-prototype word embeddings based on lexical semantic resources.

6 Conclusion

In this paper, we present a novel model to reuse the local context and enhance word embeddings for both monosemous and polysemous words. The proposed model is an extension to CBOW and it is designed to embed a word as a word semantic center and a weighted parameter. Weighted parameter denotes the influences given by the contexts. Experimental results show that embeddings trained by our model outperform competitive baselines and even state-of-the-art embeddings. When we focus on polysemy, our approach could shift the embedding of polysemous word into the corresponding semantic space according to the given context. In the future, we plan to make our model more explainable.

Acknowledgements

This research work has been partially funded by the Natural Science Foundation of China under Grant No. 91520204 and No. 61501463.

References

Mohammed Nazim Uddin, Trong Hai Duong, Ngoc Thanh Nguyen, Xin-Min Qi and GeunSik Jo. 2013. *Semantic similarity measures for enhancing information retrieval in folksonomies*, volume 40:1645–1653. Expert Syst. Appl.

Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes and Gregory Marton. 2003. *Quantitative evaluation of passage retrieval algorithms for question answering*. SIGIR.

Debasis Ganguly, Dwaipayan Roy and Mandar Mitra and Gareth J. F. Jones. 2015. *Word Embedding based Generalized Language Model for Information Retrieval*. SIGIR.

Cícero Nogueira dos Santos and Maíra A. de C. Gatti. 2014. *Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts*. COLING.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. *PhraseRNN: Phrase Recursive Neural Network for Aspect-based Sentiment Analysis*. EMNLP.

Richard Socher, John Bauer, Christopher D. Manning, Andrew Y. Ng. 2013. *Parsing with Compositional Vector Grammars*. ACL.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent and Christian Janvin. 2003. *A Neural Probabilistic Language Model*, volume 3:1137–1155. Journal of Machine Learning Research.

Ronan Collobert and Jason Weston. 2008. *A unified architecture for natural language processing: deep neural networks with multitask learning*. ICML.

Eric H. Huang, Richard Socher, Christopher D. Manning and Andrew Y. Ng. 2012. *Improving Word Representations via Global Context and Multiple Word Prototypes*. ACL.

Tomas Mikolov, Kai Chen, Gregory S. Corrado and Jeffrey Dean. 2013a. *Efficient Estimation of Word Representations in Vector Space*. Workshop at International Conference on Learning Representations(ICLR).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado and Jeffrey Dean. 2013b. *Distributed Representations of Words and Phrases and their Compositionality*. Advances in Neural Information Processing Systems(NIPS).

Jiang Guo, Wanxiang Che, Haifeng Wang and Ting Liu. 2014. *Learning Sense-specific Word Embeddings By Exploiting Bilingual Resources*. COLING.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos and Andrew McCallum. 2014. *Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space*. EMNLP.

Xuefeng Yang and Kezhi Mao. 2016. *Learning multi-prototype word embedding from single-prototype word embedding with integrated knowledge*, volume 56:291–299. Expert Syst. Appl.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman and Eytan Ruppin. 2001. *Placing search in context: the concept revisited*, volume 12:116–131. ACM Trans. Inf. Syst.

Michael Gutmann and Aapo Hyvärinen. 2012. *Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics*, volume 13:307–361. Journal of Machine Learning Research.

Cyrus Shaoul and Chris Westury. 2010. *The Westbury lab wikipedia corpus*.

Joseph Reisinger and Raymond J. Mooney. 2010. *Multi-Prototype Vector-Space Models of Word Meaning*, . The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics.

Andriy Mnih and Geoffrey E. Hinton. 2007. *Three new graphical models for statistical language modelling*. International Conference on Machine learning(ICML).

Kevin Lund and Curt Burgess. 1996. *Producing high-dimensional semantic spaces from lexical co-occurrence*. volume 28:203–208. Behavior Research Methods, Instrumentation, and Computers.

Machine Learning for Metrical Analysis of English Poetry

Manex Agirrezabal[1] and **Iñaki Alegria**[1] and **Mans Hulden**[2]
IXA NLP Group[1]
Department of Computer Science
Univ. of the Basque Country (UPV/EHU)
manex.aguirrezabal@ehu.eus
i.alegria@ehu.eus
Department of Linguistics[2]
University of Colorado
mans.hulden@colorado.edu

Abstract

In this work we tackle the challenge of identifying rhythmic patterns in poetry written in English. Although poetry is a literary form that makes use standard meters usually repeated among different authors, we will see in this paper how performing such analyses is a difficult task in machine learning due to the unexpected deviations from such standard patterns. After breaking down some examples of classical poetry, we apply a number of NLP techniques for the scansion of poetry, training and testing our systems against a human-annotated corpus. With these experiments, our purpose is establish a baseline of automatic scansion of poetry using NLP tools in a straightforward manner and to raise awareness of the difficulties of this task.

1 Introduction

Automatic analysis of the rhythmic patterns in poetry may appear deceptively simple. In fact, however, it presents a challenge for structured prediction methods in NLP on par with the most difficult language analysis tasks tackled today. What makes assigning rhythm to written poetry—i.e. "scansion"—a particularly knotty puzzle as a sequence labeling task in NLP is that, while the rhythm in most lines encountered in a work of poetry appears mundanely repetitive on the surface, poetry, while mostly a constrained literary form, is prone to unexpected deviations of such standard patterns. These departures of form, effortlessly understood and analyzed by competent speakers of the language, are tied to multiple levels of language processing. Sometimes, a simple lengthening of a line, the removal of a syllable, an onomatopoetic element, or even a semantic twist to the plot-line in a stanza of poetry can cue a sensitive human reader to assign an apparently deviant rhythmic pattern onto a line of verse.

As an example of the simple and straightforward, consider a line from the ninth book of *Paradise Lost*, by John Milton (Pickering, 1832, p. 128):

No more of talk where God or Angel guest

The even syllables of this line —*more*, *talk*, *God*, *An-* and *guest*—for most readers tend to appear naturally more prominent. At first glance, we might be tempted to assume that this repeats itself through the poem, which indeed is the case.

No more of talk where God or Angel guest
With Man, as with his friend, familiar us'd,
To sit indulgent, and with him partake

However, even here complications arise: in the second line above, we see that the word **with** appears as both unstressed and stressed, showing that the process of assigning prominence to certain syllables cannot depend purely on the lexical items themselves. Still, a naive sequence modeler that simply assumed that the poem follows an unstressed-stressed alternation would fare reasonably well here.

In contrast, consider the first and the fourth quatrains from a well-known poem by Theodore Roethke (1908–1963), *My Papa's Waltz* (1942), which tells the awkward story of a young boy in first-person whose father foists a late-night drunken waltz upon him in the kitchen.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 772–781, Osaka, Japan, December 11-17 2016.

The whiskey on your breath *You beat time on my head*
Could make a small boy dizzy; *With a palm caked hard by dirt,*
But I hung on like death: *Then waltzed me off to bed*
Such waltzing was not easy. *Still clinging to your shirt.*
from Roethke (2011)

The poem, which starts off in the first line with a seeming monotonous regularity of *iambic trimeter*—three syllable-pairs of DE-DUM per line—quickly departs from the form, as if mimicking the angular and erratic 3/4 waltz beats of the drunken father's performance. By the time we reach the word **dizzy** in the second line we find a peculiar extra syllable that needs to be accommodated. But this breaks the pattern, and we now need to decide whether to end the line **small boy di***zzy*, or perhaps **small** *boy* **di***zzy*? The rhymes also become slanted (as in **dizzy/easy**) perhaps invoking images of slurring, and the natural departure of the DE-DUM rhythmic patterns in what, for most readers, becomes a sequence of two stressed syllables, **beat time** *on my* **head**, conjures up images of the father's whacking the boy in an off-beat fashion.[1]

Most "standard" structured prediction methods effortlessly produce 80%-90% accuracy when assigning levels of stress to syllables, and do so by simply marking the most prominent rhythmic pattern mechanically. One cannot, however, conclude from this that automatic scansion of poetry is a simple task. It merely reflects the pattern of alternation between a large number of regular lines and the unexpected irregular interlude. Moving significantly beyond the accuracies that can be achieved with straightforward machine learning methods remains a challenge for NLP.

In this paper we explore a number of machine learning techniques to automatically assign stress to written poetry against human-annotated gold standards. While we do not expect to be able to tackle highly problematic cases whose solutions require meta-readings, such as understanding the effects of whiskey on the human sense of rhythm, our purpose is to set up a strong baseline and to explore the low-hanging fruits available to us, and to establish the inherent difficulty of the task.

2 Scansion

Conventionally, the metrical scansion of a line of poetry should yield a representation which marks every syllable with its level of stress and divides groups of syllables into units of feet. Typically two or more levels of stress are used. Consider, again, the example line from *Paradise Lost*, whose natural analysis is

```
 x  /    x   /      x    /   x  /   x    /
No more |of talk |where god |or An|gel guest
```

where we use the symbol `/` to denote stressed (ictic) syllables,[2] and `x` to denote unstressed (non-ictic) ones, as is done in Steele (1999) and the *Princeton Encyclopedia of Poetry and Poetics* (Preminger et al., 2015). The line in question follows the stress pattern

DE-DUM DE-DUM DE-DUM DE-DUM DE-DUM

and consists of five feet of two syllables each with an unstressed-stressed pattern. Indeed, this is the most common meter in English poetry, *iambic pentameter*.

The above example is rather clear-cut. How a particular line of verse *should* be scanned, however, is often a matter of contention. Consider, for example, the first three lines from the poem *Sudden Light* by the English poet Dante Gabriel Rossetti (Rossetti, 1881):[3]

Then, now,–perchance again!
O round mine eyes your tresses shake!
Shall we not lie as we have lain

[1] We are lucky to have a recording of Roethke's own rendering of the poem and know that the intended readings of the passages mentioned are **small boy di***zzy* and **beat time** *on my* **head**; see `https://www.poets.org/poetsorg/poem/my-papas-waltz-audio-only`.

[2] In the case of inline examples, we will make use of **bold** letters to denote stress.

[3] This stanza did not appear in the first edition of Rossetti's poem book, but did in the second.

The third line is the one that is somewhat ambiguous. The line in question can be read as a sequence of iambs, because the entire poem follows an iambic pattern (*Shall **we** not **lie** as **we** have **lain***). But, in a similar manner as is done in the first line from Shakespeare's 18th sonnet (Shakespeare, 1609) (***Shall** I com**pare** thee to a **sum**mers **day***), a commonplace substitution can be made, changing the first iambic foot to a trochee[4] (***Shall** we*). This leads to a so-called *trochaic substitution*. Apart from these two different options, the line could also be analyzed as consisting of two double iambs[5] (*Shall we **not lie** as we **have lain***). Finally, a last possible scansion would be to have assume a trochaic and a pyrrhic foot followed by a double iamb (***Shall** we **not lie** as we **have lain***). In other words, there exists a set of possible analyses for this line which may all be accepted as correct, or at least reasonable.

2.1 State of the art

Automatic scansion of poetry has attracted attention from numerous scholars in recent years and in the following section we discuss some of them. Some works rely on statistical analyses, like Hayward (1996) and Hayes et al. (2012). Others make use of linguistic knowledge obtained by generalizing observations found in different kinds of poetry and propose hand-written rules for the assignment of stress. Recently, as in other NLP tasks, data-driven approaches have emerged in automatic poetry analysis (Estes and Hench, 2016).

Statistics about scansion

Hayward (1996) has as its goal to investigate whether it would be possible to differentiate among the metrical patterns developed by individual writers and also the stylistic differences among periods. To this end, the authors collected a corpus of work by several poets from different time periods and built a neural network model (Rumelhart et al., 1988) to scan poems. Using this technique, Hayward analyzes the work of ten different poets and reports that the neural model of poetic meter was successful in determining "significant differences" among the analyzed poets.

Hayes et al. (2012) propose in their article a new approach to analysis in metrics. Their research is built upon two main works: generative metrics (Halle and Keyser, 1971) and their own earlier work (Hayes and Wilson, 2008) in the use of MaxEnt Grammars for the analysis of phonotactics. They propose a set of constraints that can be assumed to be active when scanning a verse line, and according to the number of times each of these constraints is not fulfilled and according to weights that each constraint have, they determine if a line is metrical or not.

Rule-based scansion

Logan (1988) documents a set of programs to analyze sound and meter in poetry. This work falls in a general genre of techniques that attempt to analyze the phonological structure of poems following the generative phonological theory outlined by Chomsky and Halle (1968) and described by Brogan (1981).

Scandroid is a program that scans English verse written in either iambic or anapestic meter, designed by Charles O. Hartman (Hartman, 1996; Hartman, 2005). The source code is publicly available.[6] The program can analyze poems and check if the predominant stress pattern is iambic or anapestic. However, if the input poem's meter is not one of those two, the system forces each line into one of them. This system represents the current state of the art in the rhythmic analysis of poetry.

AnalysePoems is another tool for identification of metrical patterns written by Plamondon (2006). In contrast with other programs, its main goal is not to perform a perfect scansion, but to only identify the predominant meter in a poem. The program also returns the rhyme scheme that the line follows, such as, **ABCB** for poems whose even lines rhyme.

Calliope is a similar tool, built on top of Scandroid (McAleese, 2007). It is an attempt to leverage syntactic information in order to improve scansion. The program does not appear to be freely available.

[4] A sequence of syllables where the first one is stressed and the second one unstressed.
[5] Double iamb: two unstressed syllables and two stressed syllables [xx//].
[6] `http://oak.conncoll.edu/cohar/Programs.htm`

One of the recent scansion implementations is *ZeuScansion* (Agirrezabal et al., 2016), a tool for scansion of English poetry, which performs poetry scansion using a simplified version of various stress assignment 'rules-of-thumb' developed by (Groves, 1998). We use this work as a baseline for our experiments.

The rule-based systems mentioned above were designed to work only with poetry in English. There exist, however, several rule-based implementations for other languages, such as Spanish (Gervás, 2000; Navarro-Colorado, 2015).

Supervised Learning & sequential modeling

Estes and Hench (2016) is a current work that makes use of supervised learning tools in order to metrically analyze poems written in Middle High German. Middle High German poetry is a hybrid between qualitative and quantitative verse, which means that both the length and the stress of syllables are taken into account for patterning in the lines. In order to perform supervised learning, they use a corpus of 825 manually annotated lines, which are annotated by the authors. They report an F-score of 0.894 on 10-fold cross-validated development data and 0.904 on held-out testing data.

Unsupervised scansion

Greene et al. (2010) uses statistical methods in the analysis of poetry. For the learning process, *The Sonnets* by Shakespeare was used, as well as a number of other works freely available online.[7] They learn word-stress patterns from the corpus using unsupervised learning and with the incorporation of rhyme and discourse models, they use this system to generate English love poetry. In addition, they also apply their models for the automatic translation of poetry, testing them with Italian three-line stanzas as a source language and English iambic pentameter verse as the target language. We have not obtained an implementation to review.

3 Corpora

As the gold standard material for training our scansion systems, we use a corpus of syllabified and scanned poetry, For Better For Verse (4B4V), from the University of Virginia (Tucker, 2011).[8] This website was originally built as part of an interactive on-line tutorial to train people in the scansion of English poetry in traditional meter. These manually annotated poems can be downloaded from a public repository on GitHub.[9]

The entire collection comprises 78 poems containing approximately 1,100 lines in total. It includes poetry covering a time-span from the 16th century until the 20th and for each century there are at least 6 poems and a maximum of 32 works. Sometimes, several analyses are given as correct in the gold standard to accommodate a natural ambiguity when performing scansion. When two or more analyses are available, we set the error-rate to be the minimum Levenshtein distance to each of the possible analyses, in the same way as in ZeuScansion (Agirrezabal et al., 2016, p. 22) was evaluated.

4 Techniques / Features

We used several Machine Learning algorithms to test our feature configurations, some of them yielding independent outputs, used as a greedy labeler, and some others resulting in structured output. As independent predictors we used an implementation of Naive Bayes (Garner, 1995), Support Vector Machines (Cortes and Vapnik, 1995; Fan et al., 2008) and the averaged Perceptron (Rosenblatt, 1958; Freund and Schapire, 1999).[10] As sequence-based predictors we used the widely employed Hidden Markov Models (Rabiner, 1989; Halácsy et al., 2007) (HMMs) and Conditional Random Fields (Lafferty et al., 2001; Okazaki, 2007) (CRFs).

[7] http://www.sonnets.org

[8] http://prosody.lib.virginia.edu/

[9] https://github.com/waynegraham/for_better_for_verse/tree/master/poems

[10] We used an Averaged Perceptron implementation publicly available at https://bitbucket.org/mhulden/pyperceptron

Feature template set

Below we show the set of feature templates that our supervised learning scansion systems use, which include:

- Basic features that are (almost) language agnostic
  - Syllable number within the word (SNOW): This specifies the syllable within the word in which we are working currently. E.g., for the word *ha-zel*, whose lexical stress is /x, the specification of the current syllable gives information about the lexical stress of the current syllable.
  - Syllable number within the line (SNL): This feature helps to model the sequence in many types of specially metered lines, e.g., iambic lines. It can resolve possibly ambiguous cases such as the verb re-cord, whose lexical stress could be said to be x/. If we know that this word appears in the last two positions of a trochaic poem, we can ensure that it will have the /x pattern.
  - Number of syllables in the line (NSL): The combination of this feature and the previous one helps in identifying the syllables at the end of a line which are usually more regular because of rhyme patterns.
  - Syllable phonological weight (SWEIGHT): this relies on a generalization that states that heavy syllables—ones which end with a coda consonant or have diphthong nucleus—attract stress. Our hypothesis is that this would be useful in the scansion system, as reflected in John Keats' poem, "*to **swell** the **gourd** and **plump** the **hazel** **shells***".[11]
  - The last 5 characters of the word (last character, last two characters, last three characters, last four characters, and last five characters) (LC1...LC5): As primary stress of the words in English is usually concentrated in the last syllables of the word (roughly the last three syllables), we expected the last characters to be informative (Hayes, 1995, p. 50). Although it could be better to use the last characters of the syllable, as it was done in Estes and Hench (2016), we tried to be more agnostic about the language in question when developing these basic features, and chose the last characters of the word instead.
  - Word length (WLEN): We expected this to be an informative feature.
- Other features
  - Word: As the main basic units of the text, we used words as features.
  - Syllable: Some syllables are almost always stressed, which could help in the inference of stress patterns. For example, in Shakespeare's Sonnets, the syllable "sire" is used 10 times and in all of them it appears as stressed.
  - POS-tag: The part of speech is a key element to decide whether a word is a content word or function word, which affects the stress in many syllables, as in the following excerpt from *The voice* by Thomas Hardy: "*call to me call to me*", both the verb *call* and the pronoun *me* have lexical stress, but the pronoun loses the prominence when read aloud because it is not a content word. Previous works on poetry analysis, such as Groves (1998), rely on this information.
  - Lexical stress (LS): Knowing the lexical stress sequence in a phrase is an important hint for deducing the rhythmic pattern of a line of poetry. We include the lexical stress of the word that we are analyzing at the moment. This lexical stress is calculated by using the NETTalk dictionary (Sejnowski and Rosenberg, 1987) and when treating out-of-vocabulary words, we calculate their stress using an SVM implementation given in Agirrezabal et al. (2014).

These last four features are extended to include their context as well. For example, we take the current syllable ($syllable[t]$) into account but also additionally its previous and next 10 syllables (syllable[t$\pm$10]). In the case of words, part of speech tags, and lexical stresses we decided to include the $\pm$ 5 surrounding elements.

[11] In this example we use an underline to mark if a syllable is heavy or not.

5 Experiments

We first performed one experiment that only included basic features that could be inferred from each word language-agnostically, and another experiment which included all the features presented above. The simple feature configuration was composed of the first ten features above (SNOW, SNL, NSL, WLEN, SWEIGHT and LC1...LC5). Training greedy sequence predictors with these attributes shows us the basic capability of our predictors using little or (almost) no linguistic information. All these results are compared with the rule-based system ZeuScansion (Agirrezabal et al., 2016), our previous work which we use as our baseline. Results with this feature configuration can be seen in Table 1 and it seems that we could reach quite acceptable (although lower than our baseline) accuracies by simply extracting basic attributes from words. The results of the classifiers using all the features are reported in Table 2. Here, both the SVM and the Perceptron see their scores improve significantly. In the case of the Naive Bayes classifier results do not improve as much as in the other cases, probably because of the sensitivity to overlapping features in Naive Bayes. The difference between the linear SVM and the Perceptron, especially in per-line accuracy, is somewhat noteworthy. Normally, we would expect the SVM, which finds a maximum-margin classification boundary, to outperform the averaged Perceptron, but that is not the case here in both the basic feature set experiment and the full feature set one.

	Per syllable (%)	Per line (%)
Baseline	86.78	26.21
Naive Bayes	78.08	10.64
Linear SVM	83.12	23.40
Perceptron	**84.86**	**29.32**

Table 1: Accuracies of different classifiers using just the basic features (10 features) presented in section 4 using 10-fold Cross-Validation.

	Per syllable (%)	Per line (%)
Baseline	86.78	26.21
Naive Bayes	80.44	13.88
Linear SVM	87.47	35.69
Perceptron	**89.34**	**43.36**

Table 2: Accuracies of different classifiers using all the features (64 features) presented in section 4 using 10-fold Cross-Validation.

From single prediction to structured prediction

As single predictors do not optimize the resulting sequence labeling, they can make simple errors that propagate throughout the line—something that could be avoided by looking at the surrounding outputs. This is the main weakness of not using structured prediction systems.

Hidden Markov Models are simple models that have been successfully used in tasks like POS-tagging, reaching reasonably good results. Conditional Random Fields are often used as an alternative model for POS-tagging and also for Named Entity Recognition and other NLP tasks (McCallum and Li, 2003).

In our experiments, although the per-syllable accuracies do not vary too much, the per-line scores improve substantially by the use of structured predictors. In table 3 the per line and per syllable accuracy of structured prediction systems can be seen (HMM and linear-chain CRF). The HMM has been trained in the standard way, that is, using single syllables (emissions) and their corresponding classes (states). The CRF model is trained analogously, i.e. using only syllables as the features, and the previous label. As expected, training the CRFs using the richer feature configurations employed in the greedy sequence predictors above yields a much higher accuracy, especially in the per line measure. These results are shown in table 4.

	Per syllable (%)	Per line (%)
Baseline	86.78	26.21
Scandroid	89.78	42.95
HMM	**90.43**	**49.88**
CRF	88.13	43.93

Table 3: Accuracy of sequential systems using just syllables.

	CRFs		
	#Features	Per syllable (%)	Per line (%)
Basic features	10	89.66	50.16
All features	64	**91.41**	**55.30**

Table 4: Accuracies of CRFs using different (best) sets of features on 10-Fold Cross-Validation.

6 Discussion & Future work

After checking all the results of the systems, we extracted all the rhythmic pattern predictions of the systems, sorted and grouped them. We can observe that the sorted results of the greedy predictors are more scattered. This is obvious since the greedy predictors do not explicitly promote any holistic coherence at the line level. In the following we show the most common patterns and their frequencies in the same dataset (where roughly 900 lines were used for training and 100 lines were used for testing).

	CRF
26	x / x / x / x / x /
13	/ x x / x / x / x /
5	x / x / x / x / x / x
5	x / x / x / x /
5	x / x /

	Linear SVM
12	x / x / x / x / x /
7	/ x x / x / x / x /
6	x / x / x / x x x /
5	/ / x / x / x / x /
4	x / x /

In this example it can be seen that given the same dataset, the variance in outputs can be quite different. Both classifiers predict an iambic pentameter most frequently, but, in the case of the Linear Support Vector Machine there are approximately 50 analyses that only appear once (with slight differences between them). On the contrary, the number of unique analyses in the CRF results is around 30. This implicit bias toward regularity is probably helpful for highly regular poetry, but also detrimental for scansion of poetry with often recurring outlier lines. This raises an interesting question that could be tackled in the future: can we include the amount the variation of a single poet as a parameter in the model? Another related question is the amount domain adaptation that can be captured—in this case scanning poetry in a meter that has not been encountered previously.

In this work we have established a baseline for the analysis of rhythmic patterns in poetry using various supervised learning methods. As seen above, there are many cases in which the assignment of stresses is not straightforward which we plan to focus on in future work. We applied typical NLP tools directly and future efforts should focus on the improvement of these techniques, especially in the analysis of lines with syllable additions, removals, ambiguous stress assignments, etc. Additionally, we performed feature selection so as to improve accuracy—this, however, yielded a minimal improvement, but one which was not statistically significant.

The strongest results that we achieved were at around 91.4% per syllable accuracy, averaging 55.3% correctly scanned poetry lines on 10-fold cross-validation (CRF). Comparing with previous approaches to poetry scansion, we outperform rule-based systems such as *Scandroid*—which achieved a 89.78% per syllable and 42.95% per line accuracy—and *ZeuScansion*—which reaches 86.78% per syllable and a 26.21% per line accuracy (Agirrezabal et al., 2016).

Comparing this work with recent results presented in Estes and Hench (2016) on Middle High German poetry, the results that we get in 10-fold cross-validation are quite similar, as they achieve a F-score of .894 on 10-fold cross-validated data and .904 on held-out testing data.

In this article and mainly during our research, we treated the problem as a kind of binary classification task, marking the syllable either as stressed or unstressed. This binarization creates conflicts in some verses in which there is a slow increase of the stress level between syllables. In some works, e.g the above-mentioned Hayes et al. (2012), a four-level stress marking system is used. We believe that doing so can avoid some of the ambiguity problems in scansion. In this sense, the problem could be recast as either a multi-class problem, or a regression problem, calculating a non-binary level of stress for each syllable.

We also expect to improve these results using more advanced techniques. Our first intention is to use current advances in Deep Learning for the analysis of poetry, mostly sequence-based learning paradigms, such as the widely used Recurrent Neural Networks with Long Short Term Memory (LSTM). Our preliminary results with such models show that these frameworks can reach comparable (and sometimes even better) accuracies without extensive work on manual extraction of features.

We have mainly built analyzers for English poetry in this work, but our intention is to investigate if the basic features presented in this work are applicable to poetry written in other languages, and perhaps locate possible typological generalizations among different languages and poetic traditions.

As the corpus of annotated poetry we have used is not very large, we also want to explore the possibility of unsupervised learning of rhythmic patterns in poetry (in a manner similar to Greene et al. (2010)). In this context, the language-agnostic features we have developed should be especially useful. To this end, we plan to learn rhythmic patterns by extracting first the basic, and (nearly) language-universal features by performing basic syllabification according to general principles (Hayes, 2011) such as onset maximization and sonority sequencing.

Acknowledgments

The first author's work has been partially funded by the University of the Basque Country (UPV/EHU) in collaboration with the Association of the Friends of Bertsolaritza under the Zabalduz program. The work of the second author was carried out as part of the TADEEP project (Spanish Ministry of Economy and Competitiveness, TIN2015-70214-P, with FEDER funding) and the ELKAROLA project (Basque Government funding).

References

Manex Agirrezabal, Jeffrey Heinz, Mans Hulden, and Bertol Arrieta. 2014. Assigning stress to out-of-vocabulary words: three approaches. *International Conference on Artificial Intelligence, Las Vegas, NV*, 27:105–110.

Manex Agirrezabal, Aitzol Astigarraga, Bertol Arrieta, and Mans Hulden. 2016. ZeuScansion: a tool for scansion of English poetry. *Journal of Language Modelling*, 4(1):3–28.

Terry VF Brogan. 1981. *English versification, 1570-1980: a reference guide with a global appendix*. Johns Hopkins University Press.

Noam Chomsky and Morris Halle. 1968. *The sound pattern of English.* Harper & Row, New York.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Alex Estes and Christopher Hench. 2016. Supervised machine learning for hybrid meter. In *Proceedings of the Fifth Workshop on Computational Linguistics for Literature*, pages 1–8. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874. Software available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear`.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296.

Stephen R. Garner. 1995. Weka: The Waikato environment for knowledge analysis. In *Proceedings of the New Zealand computer science research students conference*, pages 57–64. Citeseer.

Pablo Gervás. 2000. A logic programming application for the analysis of Spanish verse. In *Computational Logic—CL 2000*, pages 1330–1344. Springer.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 524–533. Association for Computational Linguistics.

Peter L Groves. 1998. *Strange music: the metre of the English heroic line*, volume 74. English Literary Studies.

Péter Halácsy, András Kornai, and Csaba Oravecz. 2007. HunPos: an open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 209–212. Association for Computational Linguistics.

Morris Halle and Samuel Jay Keyser. 1971. *English stress: Its form, its growth, and its role in verse*. Harper and Row.

Charles O Hartman. 1996. *Virtual muse: experiments in computer poetry*. Wesleyan University Press.

Charles O. Hartman. 2005. The Scandroid 1.1. Software available at `http://oak.conncoll.edu/cohar/Programs.htm`.

Bruce Hayes and Colin Wilson. 2008. A maximum entropy model of phonotactics and phonotactic learning. *Linguistic inquiry*, 39(3):379–440.

Bruce Hayes, Colin Wilson, and Anne Shisko. 2012. Maxent grammars for the metrics of Shakespeare and Milton. *Language*, 88(4):691–731.

Bruce Hayes. 1995. *Metrical stress theory: Principles and case studies*. University of Chicago Press.

Bruce Hayes. 2011. *Introductory Phonology*. John Wiley & Sons.

Malcolm Hayward. 1996. Analysis of a corpus of poetry by a connectionist model of poetic meter. *Poetics*, 24(1):1–11.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.

Harry M Logan. 1988. Computer Analysis of Sound and Meter in Poetry. *College Literature*, pages 19–24.

Gareth McAleese. 2007. *Improving Scansion with Syntax: an Investigation into the Effectiveness of a Syntactic Analysis of Poetry by Computer using Phonological Scansion Theory*. Ph.D. thesis, Open University.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 188–191. Association for Computational Linguistics.

Borja Navarro-Colorado. 2015. A computational linguistic approach to Spanish Golden Age Sonnets: metrical and semantic aspects. *Computational Linguistics for Literature*, page 105.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of Conditional Random Fields (CRFs). Software available at `http://www.chokkan.org/software/crfsuite/`.

William Pickering. 1832. *The Poetical Works of John Milton*, volume 2. William Pickering.

Marc R Plamondon. 2006. Virtual verse analysis: Analysing patterns in poetry. *Literary and Linguistic Computing*, 21(suppl 1):127–141.

Alex Preminger, Frank J Warnke, and Osborne Bennett Hardison Jr. 2015. *Princeton encyclopedia of poetry and poetics*. Princeton University Press.

Lawrence R Rabiner. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Theodore Roethke. 2011. *The collected poems of Theodore Roethke*. Anchor.

Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Dante Gabriel Rossetti. 1881. Poems: A New Edition. *London: Ellis & White*.

David E Rumelhart, James L. McClelland, PDP Research Group, et al. 1988. *Parallel distributed processing*, volume 1. IEEE.

Terrence J. Sejnowski and Charles R. Rosenberg. 1987. Parallel networks that learn to pronounce English text. *Complex systems*, 1(1):145–168.

William Shakespeare. 1609. *Shakespeare's sonnets*. Thomas Thorpe.

Timothy Steele. 1999. *All the fun's in how you say a thing: an explanation of meter and versification*. Ohio University Press Athens.

Herbert F Tucker. 2011. Poetic data and the news from poems: A for better for verse memoir. *Victorian Poetry*, 49(2):267–281.

Automated speech-unit delimitation in spoken learner English

Russell Moore[1] **Andrew Caines**[1] **Calbert Graham**[1] **Paula Buttery**[2]
Automated Language Teaching & Assessment Institute
[1]Department of Theoretical & Applied Linguistics
[2]Computer Laboratory
University of Cambridge, Cambridge, U.K.
rjm49,apc38,crg29,pjb48@cam.ac.uk

Abstract

In order to apply computational linguistic analyses and pass information to downstream applications, transcriptions of speech obtained via automatic speech recognition (ASR) need to be divided into smaller meaningful units, in a task we refer to as 'speech-unit (SU) delimitation'. We closely recreate the automatic delimitation system described by Lee and Glass (2012), 'Sentence detection using multiple annotations', *Proceedings of INTERSPEECH*, which combines a prosodic model, language model and speech-unit length model in log-linear fashion. Since state-of-the-art natural language processing (NLP) tools have been developed to deal with written text and its characteristic sentence-like units, SU delimitation helps bridge the gap between ASR and NLP, by normalising spoken data into a more canonical format. Previous work has focused on native speaker recordings; we test the system of Lee and Glass (2012) on non-native speaker (or 'learner') data, achieving performance above the state-of-the-art. We also consider alternative evaluation metrics which move away from the idea of a single 'truth' in SU delimitation, and frame this work in the context of downstream NLP applications.

1 Introduction

By convention, texts written using the Latin alphabet are normally subdivided into smaller units – *sentences* – by capitalised initial characters and closing full-stops (periods), question marks and exclamation marks. The sentence has in turn become an orthodox unit of analysis for much linguistic research, from natural language processing to syntactic theory. Speech, meanwhile, is hardly ever so neatly portioned up. Pauses and turn-taking (in conversation) may at first appear to correspond to sentence-marking orthographic devices, and often they do delimit sensible language chunks, but not always. Speakers pause in strange places, make false starts, leave thoughts unfinished, and interrupt or overlap each other (Sacks et al., 1974; Dingemanse and Floyd, 2014; Carter and McCarthy, in press). These characteristic features of spontaneous speech pose a problem for researchers investigating monologues or dialogues in naturalistic scenarios: what is a sentence-like unit of spoken language?

This question has been addressed by conversation analysts, acquisition researchers assessing performance accuracy, and compilers of large corpora, among others. Common practice is to transform speech recordings into written transcripts in order to work further with the data, whether with manual or automated means. Now comes the dilemma of how to subdivide those transcripts, where appropriate, into chunks one can work with. For conversational data the first division is made by speakers' turns into what are known as 'utterances'. But then how should those utterances be further subdivided (if at all) into something akin to sentences?

The consensus, theoretically-speaking, has been to identify smaller, sentence-like units of speech on syntactic and/or semantic grounds (with more emphasis on the former), often with reference to prosody – *e.g.* intonation contour and pause duration. In terms of hand-annotated corpora, annotators are expected to have a 'feel' of where unit boundaries should go. For automated processing of large speech corpora,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 782–793, Osaka, Japan, December 11-17 2016.

researchers have attempted to model this human intuition using supervised machine learning algorithms (Shriberg et al., 2000; Roark et al., 2006; Favre et al., 2008).

In engineering research, this is one task in 'metadata extraction' (MDE), along with the detection of disfluencies and orthographic conventions related to the readibility of transcripts (Walker et al., 2005). However, to the best of our knowledge, no open source software was released with these studies. Moreover, these systems have been trained and tested on corpora of telephone conversations and news broadcasts produced by native speakers of English. Our data are monologues and so we seek to replicate the set-up reported by Lee and Glass (2012), who combined multiple information sources to delimit boundaries in monologue restaurant reviews.

Our contribution is firstly one of terminology, asserting that we are searching for SPEECH-UNITS – with 'speech-unit delimitation' (SU delimitation) our preferred name for this task, thereby avoiding reference to the 'sentence'. Note that this is not a new term, but it is not always made clear what the unit of speech analysis is, or what is meant by 'speech-unit' where it is used. Secondly, we release an SU delimitation toolkit in a public repository[1]. Thirdly, we consider alternative evaluation metrics for SU delimitation, to move away slightly from the idea of a single 'ground-truth'. Finally, we report how well our system performs on monologues produced by native speakers and learners of English, achieving an F-measure of 0.674 with our best performing set-up.

2 The speech-unit

The sentence-like unit of speech has a varied history in the applied linguistic field. Foster et al. (2000) offer a thorough review of past proposals, identifying three types: 'mainly semantic', 'mainly intonational', and 'mainly syntactic' units. They settle on the last type in their 'analysis of speech unit' (AS-unit), allowing multiple clauses in one unit, based on evidence from pause studies that syntactically-coherent units play a central role in speech planning (Deschamps, 1980; Raupach, 1980). We do not dispute these findings, but note that this unit is again heavily reliant on syntax for its definition, though it is held to reflect a psychological reality.

Meanwhile, the *de facto* standard analysis unit in conversation analysis (CA) is the 'turn construction unit', types of which are identified as "sentential, clausal, phrasal, and lexical – i.e. syntactically" (Sacks et al., 1974). Reference is made to 'sound production' and the ways it can disambiguate, for example, statements and questions. But otherwise, for CA the object of analysis is the transcript, and as such, with speech in written form, syntax is king.

Researchers at the LDC adopted a deliberately less precisely specified approach in adding punctuation to transcripts of speech: their chosen unit, the 'SU', "functions to express a complete thought or idea on the part of a speaker" (Strassel, 2003). What 'SU' actually stands for is left open: "some possibilities include: Sentential Units, Syntactic Units, Semantic Units and Slash Units" (Strassel, 2003). The SU is defined on the basis of both syntax *and* semantics so we will not prioritise either *a priori*. Finally, the 'slash unit' refers to a transcription convention that may be obscure to some, and we avoid such overt esotericism. Instead we think of an SU as a 'speech unit': a generic and sufficiently transparent concept. We also note that it has been used before by Roberts and Kirsner (2000) in their study of 'temporal cycles' in speech, defining it as, "a segmented part of speech and the hesitation pause that preceded it".

Again, we see allusion to the planning process here, except in this case semantics are brought to the foreground. LDC annotators were instructed to "rely primarily on semantic and syntactic information, and secondarily on prosodic information" when listening to the speech recordings and deciding where to delimit SUs. We see the benefit of this flexible approach, drawing on multiple information sources rather than mainly syntax, and we adopt the SU as our sub-unit of choice for speech. Once we have a reliable system to automatically identify SU boundaries it is of benefit to downstream tasks of two broad types: natural language processing (NLP) and computer-assisted language learning (CALL). For NLP tasks we want to know whether the SUs are sensible and expected in some way, and for CALL we require chunks of language which are useful for automated learner assessment and feedback.

[1] `http://github.com/rjm49/multistage_segmenter`

3 Speech-unit delimitation

Experiments in SU delimitation date back to work by Shriberg, Stolcke and colleagues (Stolcke and Shriberg, 1996; Stolcke et al., 1998; Shriberg et al., 2000). They initially developed a framework to bring together lexical, discourse and prosodic cues to tag transcripts with various kinds of hidden structural information, with some of these cues at first being hand-coded. They demonstrated that a combination of prosodic and language model features produced better tagging outputs than either feature type in isolation. Later, Shriberg et al. (2000) introduced fully automated extraction of the necessary cues. Since then, the systems have been extended with syntactic features to supplement *n*-gram models (Roark et al., 2006), alternative classifiers to the early decision tree (DT) models, such as the conditional random field (CRF) (Favre et al., 2008) and deep neural network (DNN) (Xu et al., 2014), ensemble models with multiple voting (Liu et al., 2005), and to languages other than English including Czech (Kolář et al., 2006) and Chinese (Tomalin et al., 2007). Xu et al. (2014) report leading results of 0.81 F-measure (harmonic mean of precision and recall) on SU boundaries with their DNN-CRF model, outperforming a DT-CRF baseline with 0.774 F-measure.

However, most of the above-mentioned systems have been trained and evaluated on native speaker telephone conversation and broadcast news dialogues: the Switchboard and Broadcast News datasets prepared for the RT-03/04 shared tasks in MDE by the National Institute of Standards and Technology, U.S. Department of Commerce. We work with monologues recorded in language tests, and so we more closely follow the work of Lee & Glass (2012; L&G) because they trained and tested a system on monologue restaurant reviews. We choose to mimic L&G's work because firstly the results are interpretable (as opposed to machine learning with neural networks, e.g. Xu et al. (2014)). Moreover we can work with the relatively small learner datasets we have, and make use of them in an optimal fashion – *i.e.* using different corpora to train the separate components of the system where this brings performance improvement (sections 3.1 & 4.1). The modular architecture is appealing as it allows different model types to be combined – that is, models other than the finite state transducers introduced below, even though we do not do so in this work.

3.1 System architecture

The design of L&G's system involves a combination of local constraints containing prosodic and language model information, and global constraints of SU-length. One insight from related work is that a tagging approach to the problem only considers local information: if the search space is constrained between a minimum and maximum SU-length we can instead compute the likelihood of an SU boundary, denoted <$break$>, at each inter-token interval (Matusov et al., 2006). The models are implemented with finite-state transducers and combined in a log-linear fashion, such that the problem becomes not one of tagging but instead of finding the best path through the internal SU structures of our transcripts. Figure 1 gives an overview of our system architecture which is in spirit inspired by L&G though it differs in the detail, as discussed below. It is a three-part system using several probability sources (prosodic, lexical and SU-length) modelled as finite state transducers.

3.1.1 Prosodic model

As is the norm in the SU delimitation task, we build a prosodic model (PM) to predict SU boundaries. This move, and the feature types collected, reflect the assumption that speakers tend to indicate SU boundaries in consistent ways – by pausing before starting a new SU, by producing lengthened sounds in advance of an SU end, or by tell-tale discontinuities in pitch and volume levels either side of a <$break$>. Thus our feature choices are largely conventional, following the lead of Huang et al. (2006) as well as L&G (Table 1). This gives us a feature-set whose evolution can be traced back to the work of Shriberg and colleagues, with the obvious exception of turn-taking which is not available to us in monologues, though turns have been found to be a highly informative feature in dialogues (Shriberg et al., 2000).

For each token w_i we have two measures of pause duration (before and after), three phone duration features (the final phone, the sum of its vowels, and the longest phone), nine features for fundamental frequency (f0), and nine for energy – a total of 23 features for each token in the prosodic model. In the

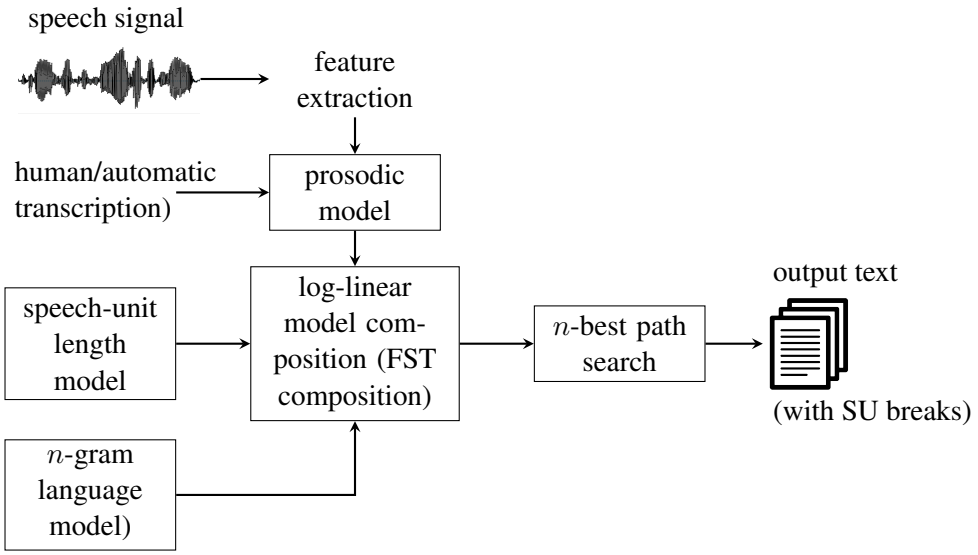


Figure 1: System diagram.

case of f0 and energy, the general task was to find maxima and minima in w_i and the following token w_{i+1}, or to calculate differences between the last frame in w_i and the first frame of w_{i+1}, or the minimum in the given recording, whilst also subtracting the speech-unit minimum from the mean across all frames in w_i, the start of w_{i+1}, and the mean of w_{i+1}.

All prosodic features were gathered automatically: transcriptions and sound files were force aligned using SPPAS (Bigi, 2012) before pause and phone durations were obtained with an R script (R Core Team, 2016), whilst fundamental frequency ('f0', measured in Hertz) and energy values (measured in decibels) were extracted in 10 millisecond frames using Praat's auto-correlated pitch and intensity tracking algorithms (Boersma and Weenink, 2016)[2]. As is conventional practice in signal processing – to normalise rapid, random changes in the signal – both f0 and energy values were smoothed using a five-point median filter and the 'robfilter' R package (Fried et al., 2014). Outlying phone durations were removed by filtering any tokens with a single phone posited to have been longer than two seconds, thereby excluding what were presumed to be gross alignment errors.

category	**features**	
pause duration	· pause before w_i · pause after w_i	
phone duration	· final phone in w_i · sum of vowel phones in w_i · longest phone in w_i	
f0	· max.f0 in w_i · min.f0 in w_i · max.f0 in w_{i+1} · min.f0 in w_{i+1}	· end of w_i − start of w_{i+1} · end of w_i − recording min.f0 · mean of w_i − recording min.f0 · start of w_{i+1} − recording min.f0 · mean of w_{i+1} − recording min.f0
energy	· per f0	

Table 1: List of prosodic features for the current word token (w_i).

This feature-set serves to capture the observed association between prosodic discontinuity and SU boundaries. That is, speakers tend to pause and lengthen SU-final tokens – hence the pause and phone duration features – and the pitch 'reset' between tokens either side of a boundary is likely to be more pronounced

[2]The Praat script was adapted from one written by Peggy Renwick, University of Georgia, for the BAAP workshop on 'methods for large-scale phonetic data analysis' held on 7 April 2014 in Oxford, U.K. We thank John Coleman, University of Oxford, for sharing it with us; `http://www.phon.ox.ac.uk/jcoleman/BAAP_workshop_info.html`.

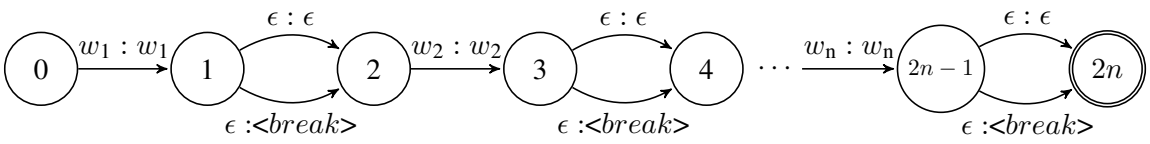


Figure 2: Prosodic model of an input string of length n.

than elsewhere: hence the focus on differences with the following token and the SU minimum (Shriberg et al., 2000). Energy features were not used by Shriberg et al. (2000) for reasons of data quality, but they were introduced by L&G on the basis of work by Huang and Zweig (2002) and so we use them here.

Per L&G, we trained a support vector machine (SVM) classifier with an RBF kernel trained on our twenty-three features, and used this to predict the probabilities of SU <$break$> tokens between word tokens. As a novel development we trained a logistic regression (LR) classifier using the six most significant features. Prosodic models can be modelled using an FST with a chain-like structure (Figure 2). Each odd-numbered node has a single arc to represent a word token, and each even-numbered node has a pair of arcs to represent what happens between those word tokens: one arc emits an empty string (modelled as an ϵ token), the other a <$break$> token, with the probabilities of these arcs being taken from the SVM or LR classifier based on the prosodic features of the given word token w_i.

3.1.2 Language model

To model the probability of local word ordering, we constructed an n-gram language model (LM). We used the OpenGRM library (Roark et al., 2012) to build models from native speaker and learner corpora. OpenGRM n-gram models are cyclic weighted FSTs, with a unigram state representing the empty string, and proper n-gram prefixes represented as their own states, so that an n-gram ($w_1..w_n$) is represented as a transition from its prefix state ($w_1..w_{n-1}$) via a word arc (w_n). Language models of this type take the same word for both input and output on each transition.

OpenGRM n-gram models use Witten-Bell smoothing by default (Witten and Bell, 1991) and back-offs are modelled using ϵ (empty string) arcs which allow the model to transition to a lower-order n-gram ($w_2..w_n$) should no state for ($w_1..w_n$) exist. To model SU delimiting <$break$> tokens we include them explicitly in the positions they occur in the training corpora. We also include an <unk> token to reserve some probability mass for out-of-vocabulary words. In all experimental settings we build 4-gram LMs.

3.1.3 Speech-unit length model

The third and final source of probabilities comes from the gold-standard length of speech-units gathered from our training corpora. Again following L&G we fit a gamma distribution to a histogram of gold-standard SU lengths. We then use this distribution to obtain the probability of a <$break$> token occurring at any given length of speech-unit, in an SU length model (SLM). These probabilities are used in the construction of a cyclic FST, where each node represents an SU of a given length (Figure 3). The transitions can accept any non-<$break$> symbol (represented here as <w>) and at each length a <$break$> is modelled by a backwards arc that restarts the 'counter' for the next SU.

Our model differs from that of L&G as we use a simple <$break$> or 'no break' probability at each length, whereas their model uses $P(length = n)$ or $P(length > n)$. As per their model we set a hard upper length L, which is corpus-dependent. We insert an SU delimiting <$break$> token when this length is reached.

3.1.4 Model composition

Across all models, probabilities are encoded as negative logs. When the models are composed, these weights, or penalties, are summed together. To find the most probable route through the combined FST we search for the combined path with the smallest weight. Since the non-<$break$> characters are accepted and emitted unchanged at each stage, it is the <$break$> tokens, or lack of them modelled as ϵ, that separate the various paths through the FST.

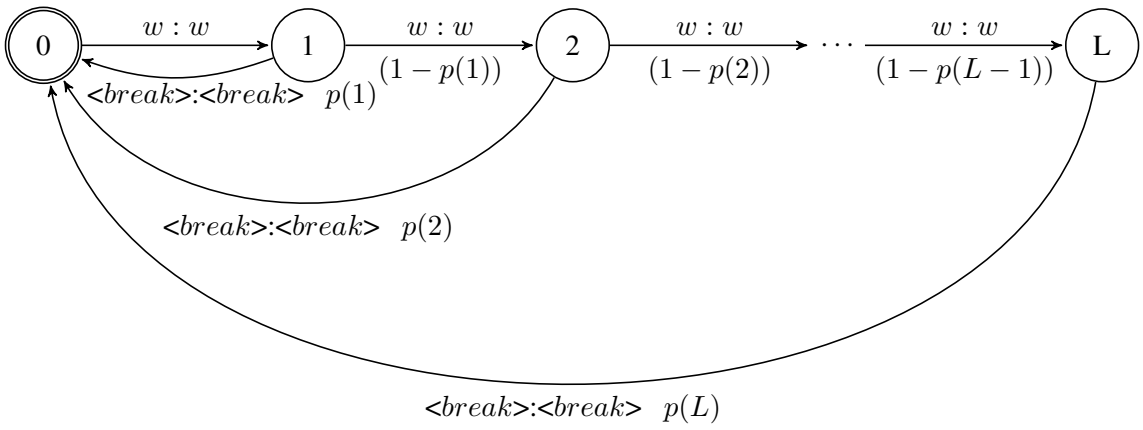


Figure 3: Speech-unit length model.

4 Experiments

The purpose of the system described above is to automatically assign SU-delimiting <$break$> tokens to transcripts of speech, based on a combination of prosodic features from the associated sound-file and probabilities from n-gram language and SU-length models.

4.1 Data

We work with a corpus of spoken learner English containing recordings of spontaneous speech from Business Language Testing Service (BULATS; `http://www.bulats.org`) oral exams. These feature 223 speakers whose first language (L1) is Gujarati – 96 male, 127 female, aged 14-50 (mean 25, median 24). Every recording has been graded by at least two expert assessors contracted by Cambridge English Language Assessment. This provides an average score for each learner which places them on the CEFR scale[3] from A1 ('beginner') to C2 ('mastery') via A2, B1, B2, C1 in increasing order of proficiency. Table 2 shows the distribution of learner CEFR levels in our BULATS corpus. It is apparent that the distribution of candidates and recordings is not equal, but there is no need for equivalence in this regard, as we seek only a representative sample of learners taking English exams. Note that there is only one candidate at the very highest C2 level, and therefore the corpus is very much a *learner* corpus mainly up to the C1 'advanced' level, rather than a corpus of Indian English learned as an L1.

CEFR level	Candidates	Recordings	Tokens
A1	33	209	6475
A2	44	288	10,597
B1	45	300	15,177
B2	44	304	17,921
C1	44	305	19,512
C2	1	6	383
Total	211	1412	70,065

Table 2: BULATS Spoken Learner Corpus.

Candidates were required to produce a monologue of twenty to sixty seconds on prompted business-related topics. Each recording was transcribed by two different crowdworkers via Amazon Mechanical Turk. Crowdworkers segmented the transcripts using punctuation, with full-stops (periods) indicating SU breaks. The two transcripts were then combined into a single version using the method described by van Dalen et al. (2015), which builds a network out of the two transcripts and uses an automatic speech

[3] 'Common European Framework of Reference for Languages' `http://www.cambridgeenglish.org/exams/cefr`

recogniser to identify an optimal path through it. Evaluating the quality of combined crowdsourced transcriptions, van Dalen et al. (2015) report a word error rate of 28.1% on another set of BULATS recordings. Inevitably, we pass word errors on through the pipeline described below, but as a method for the transcription of speech it at least gives us immediate access to large amounts of data. Phonetic transcriptions were then force-aligned with the recordings using the Hidden Markov Model Toolkit[4]. We found an error rate of 30% on phonetic alignments on a sample of the BULATS corpus, although such evaluation is not straightforward and a larger-scale exercise is needed in future work.

Transcribers indicated SU boundaries with full-stops and were asked to include all non-English words, partial words, filled pauses and repetition. The dataset features more than two hundred speakers, fourteen hundred recordings, and seventy thousand word tokens (Table 2). This is a small corpus by modern standards, and yet it is much larger than L&G's, which was 13.2k tokens. We trained a prosodic model as described in section 3.1.1 on a 90% set of 63k tokens and 2139 SUs identified by crowdworkers; the test set contains 7k tokens and 261 SUs.

We trained several 4-gram LMs (§3.1.2): firstly the learner English BULATS training set. This is only a small set of 63k tokens, and so we built a model of learner English based on a larger set of written exams from the Cambridge Learner Corpus (Nicholls, 2003), containing 766k tokens. Finally, we also trained a language model of native speaker transcripts, taken from the Switchboard Corpus of unscripted telephone conversations (Godfrey and Holliman, 1993), containing 217k tokens. Neither of the larger LMs are exactly apt for our learner monologues, in the sense that one is written language and the other is native speaker dialogue, but their greater size mitigates somewhat for the mismatch. We report in section 4.3 how all three LMs perform.

We experimented with different SLM models to little effect, and so for all experiments reported below we used an SLM model trained on our BULATS training set (§3.1.3). Maximum SU length was set to the longest SU found in the training corpus, which in this case was 185 tokens (*i.e.* if an SU reaches 185 tokens, the 186th token is a forced <$break$>). Figure 4 illustrates the distribution of SU lengths in our BULATS corpus.

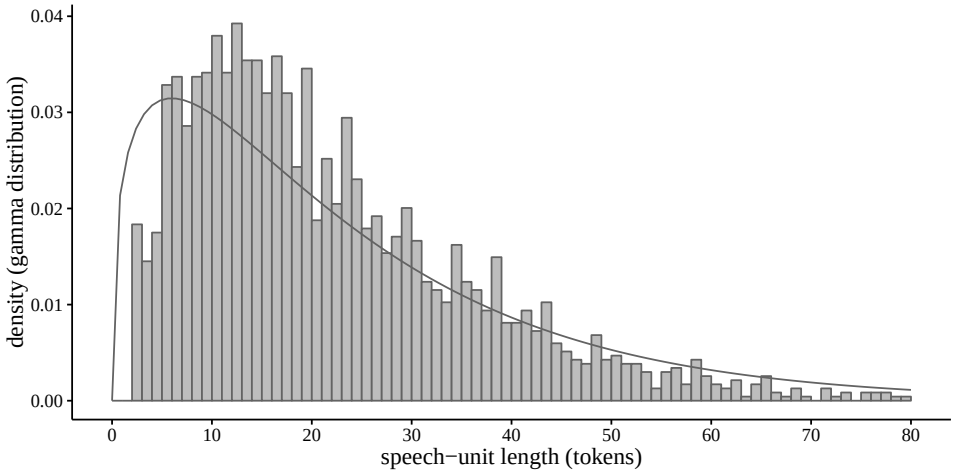


Figure 4: Density plot of speech-unit lengths in the BULATS learner corpus.

4.2 Evaluation

Lee and Glass (2012) report a BLEU-like score inspired by machine translation evaluation metrics (Papineni et al., 2002). L&G's modification is to only include n-grams consisting of hypothesised <$break$> tokens, which we have interpreted to mean any n-gram of size 1 to n with at least one <$break$> token within it. Their best performing system, with the PM weighted by 2.5, achieved a BLEU-like score of 0.56 on crowdsourced transcriptions. Unlike L&G we also report precision, recall and F-measure with

[4] `http://htk.eng.cam.ac.uk`

respect to the position of <$break$> tokens in the crowdsourced transcripts, because the BLEU-like score is a measure of *approximation* to the target inferred from the local context where <$break$> tokens occur, whereas F-measure with a single reference text is an exact score (although subject to the annotators' preferences).

These metrics very much rely on the idea of a 'gold-standard'. But as with so many language annotation tasks, the position of <$break$> tokens is highly subjective. Any downstream task requires that automatically identified SUs are useful in the sense that they can be passed to NLP tools for further inferential and processing tasks. Therefore we also report measures of SU quality, firstly with parse likelihoods from the RASP system (Briscoe et al., 2006), normalised by the number of nodes in the parse tree and averaged over each transcript. This gives us an idea of how useful the SUs are to downstream NLP tasks. Secondly we report perplexity scores for each transcript (with <$break$> tokens) obtained using the CMU-CAM toolkit (Clarkson and Rosenfeld, 1997) against a model of spoken learner English. This gives us an idea of how useful the SUs are for learner feedback in CALL systems.

Both of our new measures rely on language models and the idea of linguistic truth to some extent, but they are probabilistic, generalised over many SUs rather than the one-to-one comparison used in BLEU and precision/recall metrics. In both cases we can compare the scores for our hypothesised <$break$> tokens against the reference transcriptions and assess whether our SU delimitation system produces outputs we can work with in downstream CALL and NLP tasks. It is a matter for future work to investigate segmentation similarity metrics such as 'boundary edit distance' proposed by Fournier (2013)[5].

4.3 Results

We report modified-BLEU, precision and recall, parse likelihoods and perplexity scores (means and standard deviations) for our SU outputs in the BULATS test-set in various experimental configurations (Table 3). We tested several configurations of model type, weighting and combination, and evaluate our outputs from a held-out set of the BULATS corpus using a modified BLEU-score à la Lee and Glass (2012), information retrieval evaluation methods (IR, *i.e.* p, r, F), parse likelihoods, and perplexities.

The best performing set-up (f) features the BULATS-trained PM weighted by a factor of 5 and by recall, with a logistic regression classifier trained on the top six features. This is combined with the 4-gram Switchboard LM and BULATS SLM. Other configurations are shown for comparison. These include a baseline configuration with an unweighted SVM-based PM and learner LM/SLM (a), the same system with an LR-based PM trained on the top six features (b), and the uppermost weighting L&G apply to the PM of 2.5 (c). Both set-ups offer marked improvements on the baseline. Furthermore, a weighting of 5 on the PM offers an improved F-measure, though a reduced BLEU-like score (d). Finally, we show the performance of alternative LMs, constructed from the written learner Cambridge Learner Corpus (e) and native speaker Switchboard Corpus (f). The Switchboard LM offers a performance gain compared to the CLC, indicating that native speaker 4-grams model our test data more closely than learner writing, despite its smaller size.

We see in Table 3 that there are only small differences in BLEU-like scores, above the L&G-like baseline, in (b) to (f). This indicates that the systems are inserting <$break$> tokens in appropriate positions, given the training data, even if they are not precisely correct compared to the gold standard. Precision (p) and recall (r) confirm the differences in this regard, and show that adding weight to the PM greatly improves recall (*cf.* (b) and (c)..(f)). Increasing this weight from the 2.5 used by L&G to 5 again improves p and r (*cf.* (c) and (d)..(f)). Finally, using the larger LMs from other domains (CLC – the Cambridge Learner Corpus of written exams, and SWB – the Switchboard Corpus of telephone dialogues) leads to the most accurate SU delimitation compared to our gold standard annotations.

Our alternative measures – parse likelihood and perplexity – indicate that the hypothesised SUs score similarly across the configurations. That is, parse scores are slightly down on the gold-standard (these are negative logs, so closer to zero is more probable), whilst perplexity scores are more noticeably down

[5]We thank reviewer 1 for pointing us to this line of work. Having approached the topic from a speech engineering perspective, we were only aware of information retrieval metrics (precision, recall, *etc*) being used for evaluation, but there turns out to be a research literature in computational linguistics looking at more subtle evaluations to give credit for 'near misses' in light of the fact that annotators "frequently disagree upon the exact position of boundaries" (Artstein and Poesio, 2008).

BULATS PM	**LM**	**SLM**	**BLEU-like**	***p***	***r***	***F***	**parse likelihood mean (st.dev.)**	**perplexity mean (st.dev.)**
gold	.	.	.	.	.	.	-1.56 (0.52)	23.6 (39.9)
(a) PM_{SVM}	BULATS	BULATS	0.51	0.409	0.582	0.48	-1.58 (0.52)	35.9 (49.6)
(b) $PM_{r.6LR}$	BULATS	BULATS	0.75	0.64	0.5	0.56	-1.6 (0.53)	40.2 (60.7)
(c) 2.5*$PM_{r.6LR}$	BULATS	BULATS	0.75	0.639	0.617	0.628	-1.59 (0.54)	39.7 (60.8)
(d) 5*$PM_{r.6LR}$	BULATS	BULATS	0.74	0.653	0.686	0.669	-1.57 (0.54)	37.9 (58.7)
(e) 5*$PM_{r.6LR}$	CLC	BULATS	0.74	0.653	0.693	0.673	-1.59 (0.53)	39.2 (61.1)
(f) 5*$PM_{r.6LR}$	SWB	BULATS	0.74	0.656	0.693	0.674	-1.59 (0.54)	38.5 (60.4)

Table 3: Speech-unit delimiter output evaluation (PM: prosodic model; LM: language model; SLM: speech-unit model).

(the lower the score, the better the LM models the input) compared to gold, and the standard deviations indicate more variance within the output SU perplexities. This outcome reflects the error rate indicated by our F-measures – even though we outperform the state-of-the-art, we still have room for improvement before we can be sure that the SUs are entirely useful for downstream CALL tasks. However, the perplexity scores are low across the board and it does not seem therefore that the outputs are unfeasible. Since the parse likelihoods are similar to the gold standard, it seems that the outputs are syntactically feasible, which is promising for downstream NLP tasks – a matter we will fully evaluate in future work.

5 Discussion

Our best-performing configuration – 5*PM weighted by recall, with a logistic regression classifier trained on the top six features, and 4-gram Switchboard LM and SLM – compares favourably with the BLEU-like score of 0.56 reported by Lee and Glass (2012), and the state-of-the-art F-measure of 0.81 for SU delimitation in dialogues (Xu et al., 2014). L&G's optimal set up involves 2.5*PM based on an SVM classifier trained on their full set of twenty-three features (Table 1) and trigram LM/SLM.

It quickly became apparent to us that our prosodic model is strong (BLEU-like 0.728 alone), hence its greater weighting in our system. This was especially the case once we introduced a logistic regression (LR) classifier using the six most significant features to the PM, a refinement of L&G's method in this regard. For L&G the LM makes a huge improvement to the performance of the PM alone (from 0.13 to 0.53), while the SLM brings a more modest BLEU-like increase (to 0.56). In our case, the addition of the LM makes only a little difference (BLEU-like 0.735) with the SLM contributing another small increase (0.743). This outcome, so markedly unlike L&G's, requires several caveats. First, our LM is constructed from a relatively small corpus, with only 217k tokens in the Switchboard Corpus we use, compared to the 12m token web reviews corpus used by L&G. Secondly, it could be that native speakers of Gujarati transfer certain prosodic features that map well to speech-units when speaking English. To reinforce our results, we need to extend the system to learners with other L1s, a matter for future investigation. On the other hand, if the results stand up to further scrutiny, it suggests that the PM alone offers a good level of performance for SU delimitation, a possibility that would be beneficial resource-wise, as automatic extraction of prosodic features from the speech signal is a more straightforward exercise than subsequent combination with LM and SLM probabilities.

Moreover, we emphasise that the status of the 'gold-standard' in speech-unit annotation is uncertain. Both precision/recall and BLEU-like metrics rely on this idea, and hence we introduce other measures which relate to the 'likeliness' of the proposed SUs for downstream NLP and CALL tasks. Parse likelihoods and perplexity scores do not directly compare the hypothesised transcript to a gold-standard, but rather indicate the probability of such sequences, with <$break$> tokens positioned as they are. On these measures our outputs are syntactically feasible though more perplexing to a language model, reflecting the error rate implied by our F-measures in the range 0.4-0.7. Furthermore, as shown by the differences in BLEU-like and IR-type comparisons (Table 3) across SU delimiting systems, it is apparent that across the weighted-PM configurations, a similar performance is reported in terms of BLEU-like score, even where IR scoring varies. This suggests – because BLEU-like scores reward the prediction of n-grams commensurate with those found in the training data – that the proposed SUs in these cases may be 'good enough' even if they do not exactly correlate to the gold standard. Future work using segmentation similarity measures which explicitly reward near-misses will allow us to further investigate this matter (Fournier, 2013).

6 Conclusion

We have presented our work on speech-unit delimitation, firstly affirming that 'speech-unit' is the appropriate term for language chunks in spoken language, secondly releasing open-source software[6] to train and run an automatic SU delimiter constructed in a modular fashion per the work of Lee and Glass (2012). Thirdly, we considered alternative evaluation metrics for SU delimitation, ones which make use of the probabilities emitted by parsers and perplexity scores from statistical language models. We report the performance of our SU delimiter in various configurations on a spoken learner corpus, both with our new metrics and established BLEU-like and IR-type scores. Our best performing configuration makes use of a highly weighted LR-based PM and native speaker LM, demonstrating what we find most advantageous about this architecture: that its modular nature allows training on various sources, which is advantageous as the learner corpora we are interested in tend to be small resources.

ASR output transcripts are unpunctuated, and therefore an automated SU delimiter allows those transcripts to be subdivided and passed on to downstream applications in usable ways. In our case, we require SUs which are useful for automated learner assessment and feedback in CALL systems. In future we will continue to experiment with system configurations, data sources, and feature-sets for our SU delimiter. In addition, a further method to investigate how useful the outputs are would be extrinsic evaluation with users of a CALL system, to further consider what makes a meaningful speech-unit.

Acknowledgements

This paper reports on research supported by Cambridge English, University of Cambridge. We thank Dr Nick Saville and members of Cambridge English, Prof Ted Briscoe and colleagues in the ALTA Institute for their advice and support, and Gladys Tyen and Dimitrios Alikaniotis at DTAL. We are grateful to the three anonymous reviewers for their constructive criticisms which prompted us to make several improvements to the paper. We also thank Ann Lee and James Glass for their helpful insights into the system behind their 2012 INTERSPEECH paper, as well as Brian Roark for help with OpenGRM.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for Computational Linguistics. *Computational Linguistics*, 34.

Brigitte Bigi. 2012. SPPAS: a tool for the phonetic segmentation of speech. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC)*. European Language Resources Association.

Paul Boersma and David Weenink. 2016. Praat: doing phonetics by computer [Computer program]. Version 6.0.14.

[6] `http://github.com/rjm49/multistage_segmenter`

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP System. In *Proceedings of the COLING/ACL 2006 Interactive Presentations Session*. Association for Computational Linguistics.

Ronald Carter and Michael McCarthy. in press. Spoken Grammar: where are we and where are we going? *Applied Linguistics*. doi: 10.1093/applin/amu080 (requires access to the journal *Applied Linguistics*).

Philip Clarkson and Ronald Rosenfeld. 1997. Statistical language modeling using the CMU-Cambridge toolkit. In *Proceedings ESCA Eurospeech*.

Alain Deschamps. 1980. The syntactical distribution of pauses in English spoken as a second language by French students. In Hans Dechert and Manfred Raupach, editors, *Temporal Variables in Speech: studies in honor of Freida Goldman-Eissler*. Mouton, The Hague.

Mark Dingemanse and Simeon Floyd. 2014. Conversation across cultures. In N. J. Enfield, Paul Kockelman, and Jack Sidnell, editors, *The Cambridge Handbook of Linguistic Anthropology*. Cambridge University Press, Cambridge.

Benoit Favre, Dilek Hakkani-Tür, Slav Petrov, and Dan Klein. 2008. Efficient sentence segmentation using syntactic features. In *Proceedings of the IEEE Spoken Language Technology Workshop (SLT)*. Institute of Electrical and Electronics Engineers.

Pauline Foster, Alan Tonkyn, and Gillian Wigglesworth. 2000. Measuring spoken language: a unit for all reasons. *Applied Linguistics*, 21:354–375.

Chris Fournier. 2013. Evaluating text segmentation using boundary edit distance. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Roland Fried, Karen Schettlinger, and Matthias Borowski, 2014. *robfilter: Robust Time Series Filters*. R package version 4.1.

John Godfrey and Edward Holliman. 1993. Switchboard-1 Release 2 LDC97S62. Web Download.

Jing Huang and Geoffrey Zweig. 2002. Maximum entropy model for punctuation annotation from speech. In *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP)*.

Zhongqiang Huang, Lei Chen, and Mary Harper. 2006. An open source prosodic feature extraction tool. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*. European Language Resources Association.

Jáchym Kolář, Elizabeth Shriberg, and Yang Liu. 2006. Using prosody for automatic sentence segmentation of multi-party meetings. In *Proceedings of the 9th International Conference on Text, Speech and Dialogue (TSD'06)*, Berlin, Heidelberg. Springer-Verlag.

Ann Lee and James Glass. 2012. Sentence detection using multiple annotations. In *Proceedings of INTERSPEECH 2012*. International Speech Communication Association.

Yang Liu, Andreas Stolcke, Elizabeth Shriberg, and Mary Harper. 2005. Using conditional random fields for sentence boundary detection in speech. In *Proceedings of the 43rd Annual Meeting of the ACL*. Association for Computational Linguistics.

Evgeny Matusov, Arne Mauser, and Hermann Ney. 2006. Automatic sentence segmentation and punctuation prediction for spoken language translation. In *Proceedings of the 3rd International Workshop on Spoken Language Translation (IWSLT)*. International Speech Communication Association.

Diane Nicholls. 2003. The Cambridge Learner Corpus: error coding and analysis for lexicography and ELT. In Dawn Archer, Paul Rayson, Andrew Wilson, and Tony McEnery, editors, *Proceedings of the Corpus Linguistics 2003 conference; UCREL technical paper number 16*. Lancaster University.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

R Core Team. 2016. R: a language and environment for statistical computing.

Manfred Raupach. 1980. Temporal variables in first and second language production. In Hans Dechert and Manfred Raupach, editors, *Temporal Variables in Speech: studies in honor of Freida Goldman-Eissler*. Mouton, The Hague.

Brian Roark, Yang Liu, Mary Harper, Robin Stewart, Matthew Lease, Matthew Snover, Izhak Shafran, Bonnie Dorr, John Hale, Anna Krasnyanskaya, and Lisa Yung. 2006. Reranking for sentence boundary detection in conversational speech. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Institute of Electrical and Electronics Engineers.

Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai. 2012. The OpenGrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*. Association for Computational Linguistics.

Benjamin Roberts and Kim Kirsner. 2000. Temporal cycles in speech production. *Language and Cognitive Processes*, 15:129–157.

Harvey Sacks, Emanuel Schegloff, and Gail Jefferson. 1974. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50:696–735.

Elizabeth Shriberg, Andreas Stolcke, Dilek Hakkani-Tür, and Gökhan Tür. 2000. Prosody-based automatic segmentation of speech into sentences and topics. *Speech Communication*, 32:127–154.

Andreas Stolcke and Elizabeth Shriberg. 1996. Automatic linguistic segmentation of conversational speech. In *Proceedings of the 4th International Conference on Spoken Language Processing (ICSLP)*. International Speech Communication Association.

Andreas Stolcke, Elizabeth Shriberg, Rebecca Bates, Mari Ostendorf, Dilek Hakkani, Madelaine Plauche, Gokhan Tür, and Yu Lu. 1998. Automatic Detection of Sentence Boundaries and Disfluencies Based on Recognized Words. In *Proceedings of the 4th International Conference on Spoken Language Processing (ICSLP)*. International Speech Communication Association.

Stephanie Strassel, 2003. *Simple metadata annotation specification, Version 5.0, Linguistic Data Consortium,* `https://catalog.ldc.upenn.edu/docs/LDC2004T12/SimpleMDE_V5.0.pdf`.

Marcus Tomalin, Mark J. F. Gales, X. Andrew Liu, Khe Chai Sim, Rohit Sinha, Lan Wang, Philip C. Woodland, and Kai Yu. 2007. Improving Speech Transcription for Mandarin-English Translation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP)*.

Rogier van Dalen, Kate Knill, Pirros Tsiakoulis, and Mark Gales. 2015. Improving multiple-crowd-sourced transcriptions using a speech recogniser. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Institute of Electrical and Electronics Engineers.

Christopher Walker, Stephanie Strassel, Elizabeth Shriberg, Yang Liu, Jeremy Ang, and Haejoong Lee. 2005. RT-04 MDE Training Data Text/Annotations LDC2005T24, Linguistic Data Consortium, `http://catalog.ldc.upenn.edu/LDC2005T24`.

Ian Witten and Timothy Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37:1085–1094.

Chenglin Xu, Lei Xie, Guangpu Huang, Xiong Xiao, Eng Siong Chng, and Haizhou Li. 2014. A deep neural network approach for sentence boundary detection in broadcast news. In *Proceedings of INTERSPEECH 2014*. International Speech Communication Association.

Learning to Identify Sentence Parallelism in Student Essays

Wei Song[†], Tong Liu[†], Ruiji Fu[‡], Lizhen Liu[†], Hanshi Wang[†], Ting Liu[§]
[†]Information Engineering, Capital Normal University, Beijing
[‡]Iflytek Research Beijing, Beijing
[§]Harbin Institute of Technology, Harbin
{wsong, tongliu, lzliu, hswang}@cnu.edu.cn, rjfu@iflytek.com, tliu@ir.hit.edu.cn

Abstract

Parallelism is an important rhetorical device. We propose a machine learning approach for automated sentence parallelism identification in student essays. We build an essay dataset with sentence level parallelism annotated. We derive features by combining generalized word alignment strategies and the alignment measures between word sequences. The experimental results show that sentence parallelism can be effectively identified with a F_1 score of 82% at pair-wise level and 72% at parallelism chunk level. Based on this approach, we automatically identify sentence parallelism in more than 2000 student essays and study the correlation between the use of sentence parallelism and the types and quality of essays.

1 Introduction

Parallelism is an important rhetorical device. It can be defined as *two or more coherent text spans (phrases or sentences), which have similar syntactic structures and related semantics, and express relevant content or emotion together.* Each text span is a parallelism unit and the parallel units form a parallelism chunk. The following two sentences segmented by the semicolon form an example of sentence parallelism.

The inherent vice of capitalism is the unequal sharing of blessing;

the inherent virtue of socialism is the equal sharing of miseries. by Churchill.

Parallelism adds balance and rhythm to make speeches and writings more vivid and powerful. Moreover, parallelism also adds clarity to the sentence or even the discourse. Several sentences are expressed similarly to show that the content in the sentences are equal in importance. Therefore, properly using parallelism may improve the quality of texts. On the other hand, identifying parallelism in essays would potentially help to evaluate the quality of writings and benefit applications like essay scoring and organization evaluation.

In this paper, we study the problem of identifying parallelism in student essays. We focus on identifying sentence parallelism. A parallelism unit is a sentence, and several *parallel sentences* form a *parallelism chunk*. Parallelism identification is a task to find the parallelism chunks within essays.

This task is nontrivial. There are several factors to be considered. Since the parallel sentences should have similar structures and related semantics, they can be seen as forming a certain kind of alignment between each other. However, such alignment can exist in various levels, from surface lexical patterns to syntactic structures, semantics and even emotions. Moreover, the alignment can occur at various granularity (words, phrases, clauses or sentences). Therefore, it is difficult to design manual rules to identify sentence parallelism.

We propose a learning based framework for sentence parallelism identification. We annotate a sentence parallelism dataset consisting of about 500 student essays. This dataset allows us to derive features to model sentence parallelism and utilize machine learning to learn a prediction model. Since parallelism can be seen as a kind of alignment, we study various alignment measures to quantify the alignment between sentences. Sentence alignment depends on word alignment so that we exploit several strategies to generalize word alignment based on semantic and syntactic properties. The interactions among alignment measures and word alignment strategies generate features to represent the alignment between sentences. The experimental results show that sentence parallelism can be effectively identified. The F_1 score can reach 82% at pair-wise level and 72% at parallelism chunk level. The features based on different alignment measures and different word alignment strategies complement each other. We further study the use of sentence parallelism in more than 2000 student essays based on automated sentence parallelism identification. We observe that the use of sentence parallelism varies in narrative and argumentative essays and has a positive correlation to the quality of writings especially in argumentative essays.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 794–803, Osaka, Japan, December 11-17 2016.

2 Data

We collected student essays written by Chinese students from a senior high school during mock examinations. The essay types include narrative and argumentative essays, covering multiple topics. Two labelers were asked to label parallelism in randomly sampled essays at sentence level. They were guided by the definition of parallelism. Sentences are obtained by the sentence splitter provided by the Chinese language processing toolkit — HIT-LTP (Che et al., 2010). If a sentence contains less than four words, it is not allowed to be labeled. A sentence parallelism chunk consists of multiple sentences. The labelers recognized the sentence parallelism chunks in essays and assigned a distinct number to all parallel sentences from the same chunk in order to distinguish different chunks.

Item	Number
#Essay	544
avg. #sentence per essay	28.47
avg.#parallelism chunk per essay	2.03
avg. #sentence per chunk	2.68

Table 1: Statistics of the annotated sentence parallelism dataset.

After annotation, we collected 544 student essays, each of which has at least one sentence parallelism chunk. 30 essays were annotated by both labelers, and the Kappa value between them is 0.71 (Carletta, 1996), which indicates a moderate consistence. The mainly disagreement lies in their different judgement standards in terms of the quality of parallelism between sentences. After discussion and reaching a consensus, they reviewed all the annotations. Table 1 shows the basic statistics of the dataset.

3 Sentence Parallelism Identification

We cast sentence parallelism identification as a classification problem. Given an essay, we conduct a binary classification for every pair of sentences to determine whether they are *parallel* or *non-parallel*. Further, we get parallelism chunks according to the results of pair-wise classification.

According to the definition of parallelism, parallel sentences are expected to have sorts of alignment. The alignment can be about words, syntactic structures and semantics. In this section, we would exploit a set of alignment measures to quantity sentence alignment for a pair of sentences. For all alignment measures, the basis is the alignment between single words. Therefore, we start by studying word alignment strategies and look forward to incorporate information from multiple aspects. Then we introduce the alignment measures that are adapted for this task. Combining alignment measures and word alignment strategies, we can derive rich features to represent sentence alignment.

We use the HIT-LTP toolkit (Che et al., 2010) to conduct word segmentation, part-of-speech (POS) tagging and dependency parsing. All alignment measures would be computed on word sequences.

3.1 Word Alignment Strategies

Without loss of generality, we define a matrix R, which measures the alignment between every pair of tokens in vocabulary $\mathcal{V}$. $R(w, v)$ represents the alignment score between a pair of tokens (w, v), $w, v \in \mathcal{V}$. According to different assumption, $R(w, v)$ may have different values. We consider the following word alignment strategies:

- **Exact Match**: $R(w, v) = 1$, if $str(w) == str(v)$, otherwise $R(w, v) = 0$. $str(w)$ is the surface string of w.
- **POS Match**: $R(w, v) = 1$, if $pos(w) == pos(v)$, otherwise $R(w, v) = 0$. $pos(w)$ is the POS tag of word w.
- **Syntactic Role Match**: $R(w, v) = 1$, if $syntacticrole(w) == syntacticrole(v)$, otherwise $R(w, v) = 0$. $syntacticrole(w)$ is the syntactic role of word w. Here, we use dependency parsing to get the dependency labels as the syntactic roles. Considering the example shown in Figure 1, $syntacticrole$(春风)=SBV, $syntacticrole$(摇曳)=HED and $syntacticrole$(大地)=VOB.
- **Semantic Match**. $R(w, v) = 1$, if $similarity(w, v) > threshold$, otherwise $R(w, v) = 0$. $similarity(w, v)$ is a semantic similarity measure for w and v. Here, we compute the similarity based on word embeddings. Word embeddings are distributed representations of words learned on large scale corpus using neural networks (Mikolov et al., 2013). Each word is represented by a dense real value vector. $similarity(w, v)$ computes the cosine similarity between the vectors of w and v. The threshold is empirically set to 0.75.

The above strategies generalize words to various levels. We expect such generalization could help find alignment between sentences. For example, consider the following parallel sentences:

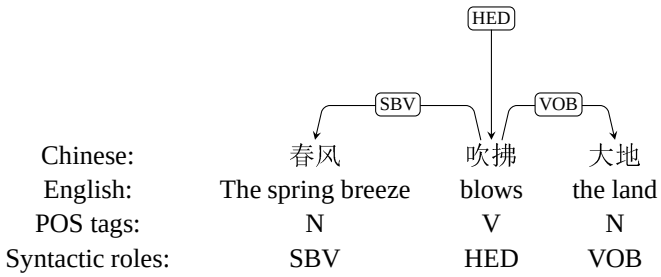


Figure 1: Dependency parsing tree for the sentence *春风吹拂大地(The spring breeze blows the land).*

春风吹拂大地, 万物复苏 (The spring breeze blows the land, reviving everything).

树根支撑枝叶,花繁叶茂 (The root supports the crown, growing the forrest).

The two parallel sentences don't share any word so that the exact match strategy fails to identify the alignment between them. However, the alignment can be captured based on POS, syntactic role and semantic matches.

3.2 Sequence Alignment Measures

We use the following algorithms to measure the alignment between sentences.

Longest Common Subsequence(LCSeq) Longest Common Subsequence algorithm is a commonly used approach to compare multiple sequences (Hirschberg, 1977). The subsequences are not required to occupy consecutive positions within the original sequences. Parallel sentences often contain longer common subsequences. The LCSeq algorithm can be effectively solved by using dynamic programming. Given two sequence $X = (x_1, x_2, ..., x_m)$ and $Y = (y_1, y_2, ..., y_n)$, the prefixes of X are X_i, i from 1 to m; the prefixes of Y are Y_j, j from 1 to n. Let $LCSeq(X_i, Y_j)$ represent the set of longest common subsequence of prefixes X_i and Y_j. This set of sequences can be got in the way below.

$$LCSeq(X_i, Y_j) = \begin{cases} \oslash & \text{if } i = 0 \text{ or } j = 0 \\ LCSeq(X_{i-1}, Y_{j-1}) \cup x_i & \text{if } R(x_i, y_j) = 1 \\ longest(LCSeq(X_i, Y_{j-1}), LCSeq(X_{i-1}, Y_j)) & \text{if } R(x_i, y_j) = 0 \end{cases}$$

$R(x_i, y_j)$ represents the condition of word alignment and can be realized using strategies in §3.1. We compute the LCSeq and the normalized length of LCSeq (NormLCSeq) for sentences s_i and s_j as features. NormLCSeq is computed as Equation 1.

$$NormLCSeq(s_i, s_j) = \frac{|LCSeq(s_i, s_j)|}{max(|s_i|, |s_j|)} \tag{1}$$

Longest Common Substrings (LCStr) Parallel sentences have a high chance to have common substrings. Therefore, we compute the longest common substrings of two sentences. Different from LCSeq, the common substrings are required to occupy consecutive positions. Therefore, high LCStr indicates a better local alignment. Different word alignment strategies can be applied. We use the length of the longest common substring as a feature.

Needleman-Wunsch Algorithm(NW) We adapt the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970) for our task. This algorithm is widely used in computational biology for finding sequence alignment among genes. Compared with LCS, it looks for an alignment between whole sequences, which maximizes an overall score function as the sum of the scores over all aligned element pairs in two sequences.

Given two sequences $X \in \mathcal{V}^*$ and $Y \in \mathcal{V}^*$, an alignment can be represented as a two-dimensional array $Align_{2\times I}^{X,Y}$ that every word in one sequence is aligned to one word in the other sequence or to an indel which is caused by inserting a word into one sequence or deleting a word from the other sequence, where I is the number of aligned element pairs. The alignment score is computed as Equation 2.

$$AlignScore(X, Y) = \sum_{i=1}^{I} S(Align_{0,i}^{X,Y}, Align_{1,i}^{X,Y}) \tag{2}$$

where $S(x, y)$ assigns a score between a pair of aligned elements.

To compute $S(x, y)$, there are two types of parameters: a *gap penalty* and a *substitution matrix*. Gap penalty values are used to penalize the score when a word in one sequence is aligned to an indel in the other. Therefore,

NW algorithm would penalize the long distance matches. The substitution matrix is used to assign alignment scores between every pairs of words. A good pair alignment will be rewarded with a higher score. Obviously, the word alignment strategies we discuss in §3.1 can be used here to construct the substitution matrix.

The alignment algorithm runs based on dynamic programming. Please refer to the details in (Needleman and Wunsch, 1970). Once we get the best alignment, we can get a best $AlignScore$. This score is correlated to the length of sequences. To reduce this effect, we use the normalized score, as shown in Equation 3, to build features .

$$NormAlignScore(X, Y) = \frac{AlignScore(X, Y)}{I} \tag{3}$$

3.3 Tree Alignment Measures

We also exploit syntactic structures. Tree kernels are the natural way to exploit syntactic structural properties, which compute the similarities between parsed trees without enumerating the whole fragment space. In this work, we parse sentences with a dependency parser. We use the Partial Tree (PT) kernel (Moschitti, 2006) to measure the similarity between two trees, since it is suitable for dependency parsing. In addition, partial tree kernel considers the ordered child sequence, which makes it suitable for our task as well.

The PT kernel is defined as:

$$K(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \triangle(n_1, n_2) \tag{4}$$

where N_{T_1} and N_{T_2} are the sets of nodes in T_1 and T_2, respectively and $\triangle(n_1, n_2)$ indicates the number of common fragments rooted at the n_1 and n_2 nodes. The kernels can be effectively computed based on dynamic programming (Moschitti, 2006):

- if $R(n_1, n_2) = 0$, then $\triangle(n_1, n_2) = 0$
- else $\triangle(n_1, n_2)$ would be computed recursively on the sets of ordered child sequences of n_1 and n_2.

Again, we can utilize strategies introduced in §3.1 to compute $R(n_1, n_2)$. Figure 1 shows a dependency parsing tree of an example sentence. We use the normalized kernel values as features, $K^{norm}(T_1, T_2) = \frac{K(T_1,T_2)}{\sqrt{K(T_1,T_1)K(T_2,T_2)}}$.

3.4 Location and Length Features

We observe that parallel sentences also locate regularly in discourse. For example, they usually occupy consecutively within the same paragraph, or locate symmetrically in multiple paragraphs. In addition, they often have close length and close number of clauses. We use the following features to describe these observations.

- **Adjacency**: if two sentences in the same paragraph, and the absolute difference of sentence indexes is smaller than 3, the feature value is set to 1, otherwise it is set to 0.
- **Location Alignment**: This feature is based on the sentence positions. If two sentences are in different paragraphs and they are both the first sentence in the paragraphs, or both the last sentence in the paragraphs, the feature value is set to 1, otherwise it is set to 0.
- **Length difference**: The absolute length difference of two sentences.
- **Clause difference**: If the number of clauses is the same, the feature is set to 1, otherwise it is set to 0. The clauses are segmented by commas.

3.5 Summarization of Features

We summarize the used features in Table 2. Except for location and length features, we use various alignment measures together with different word alignment strategies to generate features. We use both the absolute and normalized length of LCSeq scores as features. LCStr focuses on local consecutive matches, therefore we only use the length of longest common substrings as features. The tree kernel method deals with syntactic structural properties, therefore we construct two trees for each sentence based on dependency parsing. We use POS tags and dependency tags respectively as the values of the tree nodes.

Feature Set	Feature	Word Alignment Strategy
LCSeq	$\|LCSeq_{exact}(s_i, s_j)\|$, $NormLCSeq_{exact}(s_i, s_j)$	Exact match
	$\|LCSeq_{pos}(s_i, s_j)\|$, $NormLCSeq_{pos}(s_i, s_j)$	POS match
	$\|LCSeq_{semantic}(s_i, s_j)\|$, $NormLCSeq_{semantic}(s_i, s_j)$	Semantic match
	$\|LCSeq_{syntacticrole}(s_i, s_j)\|$, $NormLCSeq_{syntacticrole}(s_i, s_j)$	Syntactic role match
LCStr	$\|LCStr_{exact}(s_i, s_j)\|$	Exact match
	$\|LCStr_{pos}(s_i, s_j)\|$	POS match
	$\|LCStr_{semantic}(s_i, s_j)\|$	Semantic match
	$\|LCStr_{syntacticrole}(s_i, s_j)\|$	Syntactic role match
NW	$NormAlignScore_{exact}(s_i, s_j)$	Exact match
	$NormAlignScore_{pos}(s_i, s_j)$	POS match
	$NormAlignScore_{semantic}(s_i, s_j)$	Semantic match
	$NormAlignScore_{syntacticrole}(s_i, s_j)$	Syntactic role match
Tree Alignment	$K^{norm}_{pos}(s_i, s_j)$	POS match
	$K^{norm}_{syntacticrole}(s_i, s_j)$	Syntactic role match
LocLen	Adjacency	—
	Location Alignment	—
	Length Difference	—
	Clause Difference	—

Table 2: Summarization of the features for a sentence pair $< s_i, s_j >$.

3.6 Parallelism Chunk Identification

Given an essay, once every pair of sentences is classified as *parallel* or *non-parallel*, we construct parallelism chunks based on the classification results. We use an aggressive strategy based on transitivity: if two sentence pairs $< x, y >$ and $< x, z >$ are parallel, then $< x, y, z >$ forms a parallelism chunk, no matter whether pair $< y, z >$ is classified as parallel.

4 Evaluation

4.1 Experimental Settings

Data and Classifiers We split the essays in our dataset into 5 parts and run cross-validation. Each time, 4 parts are used for training and the remaining part is used for test. Sentences from the same parallelism chunks form a set of positive pairs, while sentences that are in the same essay but not parallel form negative pairs.

The word embeddings for semantic similarity computation are learned using the Word2Vec tool (Mikolov et al., 2013) on a dataset consisting of 85,000 student essays collected from the web. The dimension of word embeddings is 50. The size of vocabulary is about 490,000.

We adopt the Random Forests (Breiman, 2001) as the classifier and use the implementation in Scikit-learn toolkit (Pedregosa et al., 2011) with default parameters.

Evaluation Metrics We adopt precision, recall and F_1 score as evaluation metrics. The metrics can be computed at pair-wise level and parallelism chunk level respectively.

At pair-wise level, the precision and recall are computed as:

$$\textit{pair-precision} = \frac{\#\text{correctly identified parallelism sentence pairs}}{\#\text{identified parallelism sentence pairs}} \tag{5}$$

$$\textit{pair-recall} = \frac{\#\text{correctly identified parallelism sentence pairs}}{\#\text{true parallelism sentence pairs}} \tag{6}$$

At chunk level, the precision and recall are computed as:

$$\textit{chunk-precision} = \frac{\#\text{correctly identified parallelism chunks}}{\#\text{identified parallelism chunks}} \tag{7}$$

$$\textit{chunk-recall} = \frac{\#\text{correctly identified parallelism chunks}}{\#\text{true parallelism chunks}} \tag{8}$$

A correctly identified parallelism chunk means the identified chunk has the same sentences with a labeled chunk. In both cases, $F_1 = \frac{2 \times precision \times recall}{precision + recall}$.

Feature Set	pair-P	pair-R	pair-F_1	chunk-P	chunk-R	chunk-F_1
LocLen	0.71	0.41	0.52	0.38	0.32	0.35
LCSeq	0.72	0.64	0.68	0.45	0.49	0.47
LCStr	0.71	0.63	0.67	0.46	0.48	0.47
NW	0.75	0.68	0.72	0.49	0.55	0.52
Tree alignment	0.67	0.44	0.53	0.33	0.31	0.32
NW + LocLen	**0.85**	0.78	0.81	0.68	0.70	0.69
NW + Tree alignment	0.81	0.70	0.75	0.58	0.59	0.59
ALL	**0.85**	**0.79**	**0.82**	**0.73**	**0.70**	**0.72**

Table 3: Evaluation results of using different alignment measures.

Word Alignment Strategy	pair-P	pair-R	pair-F_1	chunk-P	chunk-R	chunk-F_1
Exact	0.72	0.69	0.71	0.47	0.54	0.50
POS	0.49	0.51	0.50	0.23	0.31	0.26
Syntactic role	0.77	0.59	0.67	0.52	0.52	0.52
Semantic	0.79	0.66	0.72	0.56	0.57	0.56
Exact + Syntactic role	0.81	0.71	0.76	0.62	0.65	0.64
Exact + Semantic + Syntactic role	0.82	0.72	0.77	0.64	0.65	0.64
Exact + Semantic + Syntactic role + POS	0.81	0.72	0.76	0.62	0.65	0.63

Table 4: Evaluation results of using different word alignment strategies.

4.2 Results

Table 3 shows the experimental results using different sequence alignment measures. All word alignment strategies are used in this experiment. The best alignment measure is the score computed using the Needleman-Wunsch algorithm. This is reasonable, since it captures the alignment on the whole sequence, considering the local alignments like LCStr and penalizing long distance matches, which LCSeq ignores.

Different from LCStr, LCSeq and NW, tree alignment exploits more complex structural information. So tree alignment based measures should complement sequence based measures. The best combination of tree based and sequence based measures are NW plus tree alignment.

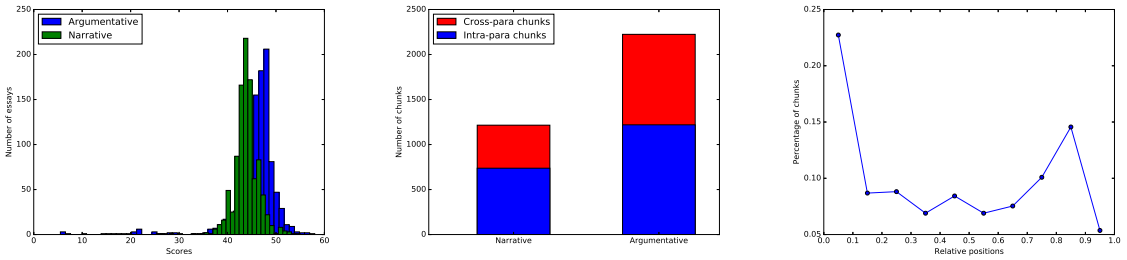
Using the location and length features (LocLen) alone leads to a low recall, but they can improve the performance when combining with other features. We can see that combining LocLen and NW achives good performance.

Combining all alignment measures achieves the best performance. The results demonstrate that pair-wise sentence parallelism can be effectively identified. The chunk-wise performance is moderate. The precision of pair-wise classification is shown to be more crucial to the chunk-wise performance.

Feature	Weight
$NormAlignScore_{semantic}$	0.161
$NormAlignScore_{exact}$	0.148
$NormAlignScore_{syntacticrole}$	0.117
$NormAlignScore_{pos}$	0.105
Location Alignment	0.078
$K^{norm}_{syntacticrole}$	0.062
$NormLCSeq_{exact}$	0.056
Length Difference	0.047
K^{norm}_{pos}	0.040
$LCStr_{exact}$	0.036

Table 5: Top ranked feature weights.

We are also interested in the contributions of various word alignment strategies. Table 4 shows the performance of using different word alignment strategies and their combinations. All alignment measures are used in this experiment. We can see that the semantic match strategy performs best. This indicates that semantic level information is important. We observe that the best combination of two strategies is the combination of exact match and syntactic role match, while the best combination of three strategies is the combination of exact match,



(a) Distribution of essays scores.

(b) Number of chunks in narrative and argumentative essays.

(c) Distribution of chunks' relative positions.

Figure 2: Basic statistics of the student essay dataset and the use of sentence parallelism in the dataset.

semantic match and syntactic role match. Different strategies complement each other except that the POS match doesn't provide extra gain in our experiments. Table 5 shows the feature weights learned by the Random Forests model. The trend is similar to the previous observations.

5 Sentence Parallelism and Essay Writing

The automated sentence parallelism identification makes it possible to study the use of sentence parallelism in student essays and its relation to essay quality on large datasets. We collected another dataset containing essays written by senior high school students in mock examinations. This dataset has 1036 narrative essays and 1064 argumentative essays, and it doesn't overlap with the dataset introduced in §2. All these essays had been scored by professional high school teachers. The scores ranges from 0 to 60. The distribution of essay scores is shown in Figure 2(a). We can see the distribution of either narrative or argumentative essays meets the normal distribution. The dataset should be representative to reflect the real situation.

5.1 How Students Use Sentence Parallelism

We use our system to process these essays and extract parallelism chunks. We extract 2224 and 1219 parallelism chunks in argumentative essays and narrative essays respectively. These parallelism chunks can be categorized into two types: *intra-para chunks* and *cross-para chunks*. Intra-para chunks contain parallel sentences within the same paragraph, while cross-para chunks have parallel sentences across multiple paragraphs.

The ratio of each type of chunks are shown in Figure 2(b). Parallelism chunks, especially the ones that cross paragraphs, are used more often in argumentative essays than in narrative essays. We examine some essays and find that in argumentative essays, students would use parallel sentences to express their main ideas that are used to support the thesis from different aspects. The parallelism can add the clarity in organization. Therefore, there are more cross-para chunks in argumentative essays.

We also examine the relative positions of parallelism chunks. We use the average sentence number of sentences in a chunk divided by the number of sentences in the essay as the relative position of the chunk. Figure 2(c) shows the distribution of relative positions, which are grouped into 10 zones. We can see the distribution is interesting that sentence parallelism is used much more often in the beginning or the ending of essays, while relatively less parallel sentences are used in the body part. We guess the reason is that since parallel sentences are used to impress the readers, the beginning and the ending parts are easier to draw readers' attentions. As a result, students tend to put impressive sentences at these important positions.

Essay type	#intra-para chunks	#cross-para chunks	#all chunks	Presence
Narrative	0.146	0.082	0.161	0.146
Argumentative	0.20	0.233	0.290	0.299

Table 6: Pearson coefficients between scores and the number of different types of parallelism chunks.

5.2 Sentence Parallelism and Essay Scores

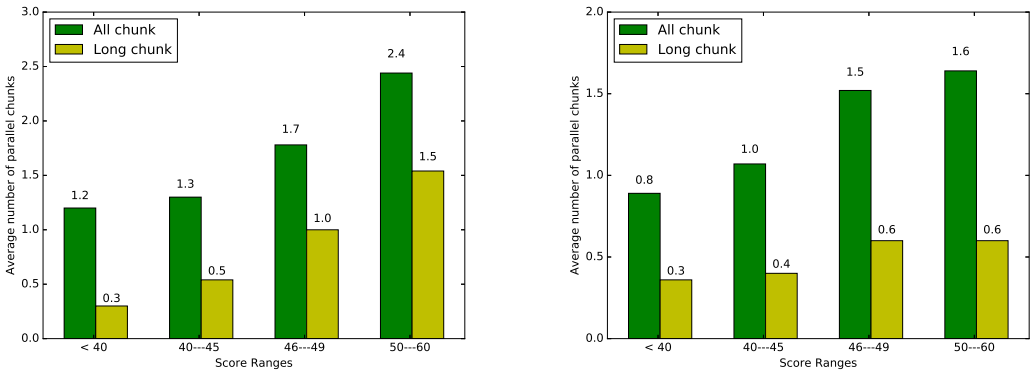
Does the use of sentence parallelism relate to the quality of essays? To answer this question, we analyze the relationship between sentence parallelism and essay scores. We first compute the pearson correlation coefficients between the number of different types of parallelism chunks and the scores of essays. Table 6 shows the results. The correlation coefficient with the number of chunks can reach 0.29 in argumentative essays. In contrast, the

Essay type	All essays	Presence	Absence
Narrative	43.74	44.09	43.13
Argumentative	45.65	46.35	42.13

Table 7: Average scores of essays in terms of the presence or absence of sentence parallelism.

correlation in narrative essays is much lower. The number of cross-para chunks has a higher correlation to essay scores in argumentative essays, but a lower correlation in narrative essays, compared with the number of intra-para chunks.

If we consider the presence or absence of parallelism, the correlations have the same trends as shown in the last column in Table 6. Table 7 shows the average scores of essays in terms of the presence or absence of sentence parallelism. In both narrative and argumentative essays, essays with sentence parallelism tend to have higher scores in average compared with the ones without sentence parallelism. Both differences are significant at 95% level (t-test, $p < 0.05$). The score difference is more obvious in argumentative essays.



(a) Argumentative essays.

(b) Narrative essays.

Figure 3: Average number of parallelism chunks in essays coming from different score ranges.

We further examine sentence parallelism in essays within 4 different score ranges, including 40 points below, 40 to 45, 46-49 and 50-60. We compute the average number of parallel sentence chunks in each score range. The result is shown in Figure 3 for argumentative and narrative essays respectively. We can see that the essays from higher score ranges have a larger average number of parallelism chunks. The trend holds for both argumentative and narrative essays.

We also want to consider the effect of the quality of sentence parallelism. Instead of dealing with the content, we consider *long chunks* whose chunk sizes are equal to or greater than 3. We simply view long chunks as high quality parallelism. The results are also shown in Figure 3. The differences among score ranges in argumentative essays are more obvious so that high quality sentence parallelism might be a useful indicator of well written argumentative essays. In contrast, long chunks appear much less in narrative essays across all score ranges.

5.3 Discussion

This section have studied the relationship between the use of sentence parallelism and the types and quality of student essays. The biggest observation is that the essay type—argumentative or narrative essay—is a key factor when we study sentence parallelism. First, the frequency and styles of using sentence parallelism are different. Parallelism is much more often used in argumentative essays. The ratio of cross-para chunks is also higher in argumentative essays. Second, the frequency or presence of using sentence parallelism has a positive correlation to the quality of essays. The correlation exists but is weak in narrative essays, while it is stronger in argumentative essays. According to the score distributions, recognizing high quality and low quality essays is crucial and challenging. High quality sentence parallelism may be useful to distinguish good and poor argumentative essays.

Notice that the ways of using parallelism may relate to how students are taught on writing, which might be different across countries and cultures. The observed statistics may also be affected by the topics of essays. Nonetheless, the observations should potentially help to design features for automated writing evaluation.

6 Related Work

6.1 Sequence Alignment

Finding the common parts among sequences have been a set of classic computer science problems. The typical problems include finding the longest common subsequence (Hirschberg, 1977), longest common substring and multiple sequence alignment (Carrillo and Lipman, 1988; Needleman and Wunsch, 1970). These techniques are commonly used in computational biology and also applied to natural language processing for constructing concept mapping dictionary (Barzilay and Lee, 2002), identifying sentence level paraphrases (Barzilay and Lee, 2003) and modeling the organization of student essays (Persing et al., 2010). In this work, we exploit these alignment measures for deriving features, since parallel sentences should have a kind of alignment.

6.2 Semantic Similarity of Texts

A large of body of previous work focuses on measuring the semantic similarity of texts. Semantic similarity of text usually depends on exploiting the semantic similarity of words and concepts (Corley and Mihalcea, 2005; Mitchell and Lapata, 2008). While the semantic similarity of words and concepts are learned based on distributional statistics (Lin, 1998; Padó and Lapata, 2007). Recently, neural networks based methods are proposed to learn the distributed representation of words on large scale of corpus (Mikolov et al., 2013). The learned word embeddings enable similar words to have a close distance in the vector space. There is also work on sentential paraphrase identification (Madnani and Dorr, 2010). Paraphrases are different expressions that convey the same meaning. Although it is similar to our task, the goals are different, since parallel sentences are not expected to have the same meaning and paraphrases are not required to have similar structures. Many researchers also exploit the structural properties of sentences to measure semantic similarity of texts, such as the tree kernel emthods (Moschitti, 2006; Mooney and Bunescu, 2005; Culotta and Sorensen, 2004).

6.3 Text Quality Analysis

Some work focuses on dealing with rhetorical device such as recognizing metaphor in texts (Shutova, 2010). Parallelism is also an important rhetorical device. Hobbs and Kehler (1997) study the clause level parallelism. However, little work has been done on sentence-level parallelism identification. These is work on predicting the quality of articles (Louis and Nenkova, 2013; Pitler and Nenkova, 2008), writing styles (Ashok et al., 2013) and student essays (Attali and Burstein, 2006). They mainly use simple shallow features, but seldomly use rhetorical device related features. Automated rhetorical device analysis should help to improve the above tasks.

7 Conclusion

We have investigated identifying sentence parallelism in student essays. We adopt machine learning to learn a prediction model based on an annotated dataset. We study various alignment measures and different word alignment strategies for deriving features. The evaluation demonstrates that our proposed method can effectively identify sentence parallelism, achieving a F_1 score of 82% at pair-wise level and 72% at parallelism chunk level. We also study the use of sentence parallelism in more than 2000 student essays based on automated parallelism identification. We find that students tend to use more sentence parallelism in argumentative essays compared with narrative essays. The essays with sentence parallelism have higher scores in average. The presence of high quality sentence parallelism shows to be an indicator of high quality argumentative essays.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (No.61402304,No.61303105), the Beijing Municipal Natural Science Foundation (No.4154065) and the Humanity & Social Science General Project of Ministry of Education (No.14YJAZH046).

References

Vikas Ganjigunte Ashok, Song Feng, and Yejin Choi. 2013. Success with style: Using writing style to predict the success of novels. *Poetry*, 580(9):70.

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater® v. 2. *The Journal of Technology, Learning and Assessment*, 4(3).

Regina Barzilay and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *EMNLP 2002*, pages 164–171. Association for Computational Linguistics.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *NAACL 2003 - Volume 1*, pages 16–23. Association for Computational Linguistics.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Jean Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. *Computational linguistics*, 22(2):249–254.

Humberto Carrillo and David Lipman. 1988. The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48(5):1073–1082.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 13–16. Association for Computational Linguistics.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment*, pages 13–18. Association for Computational Linguistics.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 423. Association for Computational Linguistics.

Daniel S Hirschberg. 1977. Algorithms for the longest common subsequence problem. *Journal of the ACM (JACM)*, 24(4):664–675.

Jerry R Hobbs and Andrew Kehler. 1997. A theory of parallelism and the case of vp ellipsis. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 394–401. Association for Computational Linguistics.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *ICML*, volume 98, pages 296–304. Citeseer.

Annie Louis and Ani Nenkova. 2013. What makes writing great? first experiments on article quality prediction in the science journalism domain. *Transactions of the Association for Computational Linguistics*, 1:341–352.

Nitin Madnani and Bonnie J Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *ACL*, pages 236–244.

Raymond J Mooney and Razvan C Bunescu. 2005. Subsequence kernels for relation extraction. In *Advances in neural information processing systems*, pages 171–178.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *European Conference on Machine Learning*, pages 318–329. Springer.

Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

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

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Isaac Persing, Alan Davis, and Vincent Ng. 2010. Modeling organization in student essays. In *EMNLP 2010*, pages 229–239. Association for Computational Linguistics.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 186–195. Association for Computational Linguistics.

Ekaterina Shutova. 2010. Models of metaphor in nlp. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 688–697. Association for Computational Linguistics.

Evaluating anaphora and coreference resolution to improve automatic keyphrase extraction

Marco Basaldella, Giorgia Chiaradia and **Carlo Tasso**
{basaldella.marco.1, chiaradia.giorgia}@spes.uniud.it
carlo.tasso@uniud.it
Artificial Intelligence Laboratory
Università degli Studi di Udine
Via delle Scienze 208, Udine, Italy

Abstract

In this paper we analyze the effectiveness of using linguistic knowledge from coreference and anaphora resolution for improving the performance for supervised keyphrase extraction. In order to verify the impact of these features, we define a baseline keyphrase extraction system and evaluate its performance on a standard dataset using different machine learning algorithms. Then, we consider new sets of features by adding combinations of the linguistic features we propose and we evaluate the new performance of the system. We also use anaphora and coreference resolution to transform the documents, trying to simulate the cohesion process performed by the human mind. We found that our approach has a slightly positive impact on the performance of automatic keyphrase extraction, in particular when considering the ranking of the results.

1 Introduction

Automatic Keyphrase Extraction (henceforth AKE), i.e. the task of extracting a list of phrases of one or more words "that capture the main topics discussed in a given document" (Turney, 2000) is a natural language processing (herein NLP) task which received widespread attention in the last years, with applications, e.g., in the fields of digital libraries (Gutwin et al., 1999) or community modelling (De Nart et al., 2015).

Many AKE algorithms have been developed, which can be roughly divided into two categories (Hasan and Ng, 2014):

- *Supervised algorithms*: after the generation of candidate keyphrases (henceforth KPs) by means of linguistic knowledge, these candidates are associated to a set of *features* such as TF-IDF, position in the text, and so on; then, a supervised machine learning (herein ML) algorithm learns over a training set how to decide if a candidate is a suitable KP or not.

- *Unsupervised algorithms*: for example, the document is represented using a graph structure, whose nodes are candidate KPs. Then, the *popularity* of each candidate is evaluated using graph algorithms usually derived from the PageRank algorithm (Mihalcea and Tarau, 2004; Wan and Xiao, 2008). Other approaches include for example clustering-based algorithms, such as the one presented in (Liu et al., 2009), or techniques which rely on building a statistical language model to rank KPs, like the one presented in (Tomokiyo and Hurst, 2003).

However, the performance of the state of the art systems is still much lower than many other NLP tasks. An idea of the current top performing systems can be obtained by looking at the results of "SEMEVAL 2010 Task 5: Automatic Keyphrase Extraction from Scientific Articles" (Kim et al., 2010), where systems were ranked by F-score on the top 15 extracted keyphrases. The best system, presented by (Lopez and Romary, 2010), achieved a score of 27.5%.

After SEMEVAL 2010, many systems tried to improve this level of performance, with an increasing focus over supervised systems. A common strategy is to look for new features to be used by ML algorithms. As an example, in (Haddoud et al., 2015) the authors were able to overcome the best SEMEVAL

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 804–814, Osaka, Japan, December 11-17 2016.

performance achieving an F-Score of 28.6% on the top 15 KPs, by introducing a feature called Document Phrase Maximality, which they claim is able to better identify overlapping KPs, i.e. keyphrases that have a part in common, like for example "engineering" and "software engineering".

In this paper we follow the path of exploring new features and new ways of using linguistic knowledge from anaphora resolution to improve AKE. We started from the following hypotheses:

- If an n-gram is referenced many times inside a document, e.g. has many *anaphors*, its level of relevance as a KP may increase;
- If a pronoun can be replaced with the noun (or noun phrase) that it substitutes, we may detect information about said noun that otherwise would be lost; this information could be used to detect better KPs.

To check if these hypotheses hold we used the following approach. First, we set a baseline to compare our hypotheses against by choosing a minimal set of features that defined a system behaving like an average SEMEVAL 2010 contestant. Then, we designed two approaches, one based on the new linguistic features and the other based on a text preprocessing stage which applies anaphora-antecedent substitutions. Finally, we evaluated the performance of several ML algorithms using the SEMEVAL 2010 dataset and different feature sets combination which include the first hypothesis, the second one, or both.

2 Related work

The use of linguistic knowledge in AKE is not new. An interesting approach is the one presented in (Hulth, 2003), where the author wanted to demonstrate that the use of linguistic knowledge can lead to more compelling results than the ones obtained with the application of statistical features only. The AKE system proposed by Hulth exploits 4 features: three introduced in (Frank et al., 1999), which are within-document-frequency, collection-frequency and position of the first occurrence of the token, plus a new one that evaluates the POS-tag assigned to a term. The KPs extracted with the linguistic approach turned out to have lower recall but greater precision than the ones computed with the statistical one. In (Nguyen and Kan, 2007) the authors used a similar approach by taking into account also suffix sequences, acronym status and POS tag sequence as an hint for terminological status of the candidates. In (Pudota et al., 2010) instead the authors designed a system which weighted KPs based on the number of nouns contained in them. This system is the ancestor of the Distiller framework (Basaldella et al., 2015), which is the software we used in this work to perform our experiments.

Another way to improve AKE could be taking advantage of linguistic knowledge from anaphora and coreference resolution, which are the fields we are going to explore in this paper. Anaphora resolution is the problem of resolving what a pronoun or a noun phrase refers to. Lappin and Leass (1994) proposed "an algorithm for identifying both intrasentential and intersentential antecedents of pronouns in text": they use syntactical and morphological filters to find out dependencies between pronouns and possible related noun phrases (herein NPs), scoring the candidates by considering salience to select an adequate antecedent for each pronoun not flagged as pleonastic.[1]

A close field to anaphora resolution is *coreference resolution*. These two fields share similar information, so they overlap in a certain way: resolving anaphora is about finding the "cohesion"[2] which points back to some previous item" as stated in (Halliday and Hasan, 2014). So, the process of binding an antecedent to an anaphora is anaphora resolution; coreference resolution instead is the process of detecting when anaphors and their antecedent(s) have in common the same referent in the real world. Consider this the example from (Mitkov, 2014):

> This book is about anaphora resolution. The book is designed to help beginners in the field and its author hopes that it will be useful.

[1] A pleonastic pronoun is tipically used in phrasal verbs as subject or object but without an effective meaning (i.e. it seems, it is known, etc.)

[2] see section 3.1

Then the NP "the book" and both the pronouns "its" and "it" are anaphors referring to the antecedent "This book", and all three anaphors have the same real-word referent, which is "this book". So anaphors and their antecedent(s) are coreferential and form a chain, called *coreference chain*, in which all the anaphors are linked to their antecedent.

In our experiment we use the Stanford Coreference Resolution System `dcoref` (Manning et al., 2014) to retrieve anaphors and referents to implement our linguistics based features. We choose this software because of its good performance, since it is the direct descendant of the top ranked system at the CoNLL-2011 shared task. Moreover, both the Distiller and Stanford's system are Java-based, thus the integration of the two systems is easier. To resolve both anaphora and coreference, `dcoref` extracts the couples of anaphors and their relative referents, according to the matching of phrases' attributes, such as gender and number.

Other strategies to anaphora resolution, as the one introduced by Ge et al. (1998), use statistical information to resolve pronouns They use the distance between pronoun and the proposed antecedent to check the probability of the connection between them, information about gender, number, and animacity of the proposed antecedent as hints for the proposed referent, and head information to make selectional restrictions and mention count.

3 Anaphora in Keyphrase Extraction

3.1 Motivation

When we (humans) communicate, whether in a spoken or written form, generally we express a $coherent$ whole, i.e., a consistent and logical collection of words, phrases, and sentences. People often use abbreviated or alternative linguistic forms, such as pronouns, referring to or replacing some items that were previously mentioned in the discourse. Thus, to fully understand the discourse we need to interpret these elements, which *depend* on the elements they refer to. In linguistics, this process of interpretation is called *cohesion* (Mitkov, 2014).

On the other hand, when a document is processed for AKE, non influential words are usually removed. These words, commonly called *stop words*, are excluded from the text because they appear to be not significant, even if they are extremely frequent. Among them there are also pronouns such as *he*, *she*, *it*, *that*, *who*, and so on. The removal of such elements causes a loss of cohesion, both syntactically and semantically.

Moreover, pronouns have a relevant role in the sentences since they allow the author to enrich his writing using a richer vocabulary, composing more complex sentences, and so on. Pronouns are parts of the text which typically have the function of a substitute: depending on the case, they can replace a subject or an object, they can indicate possession, places, or refer back to people or things previously mentioned. Given these premises, disregarding all pronouns without replacing them with a valuable substitute could lead to a loss of a syntactical and/or semantical information. In fact, during the reading process we are able to decode the information conveyed by pronouns because we automatically replace them with the entity they refer to. In NLP a similar process is performed by anaphora resolution, thus our idea is to use this information, which would be otherwise lost, for AKE.

3.2 Definitions

We use the following definitions. Words are identified with the letter w and keyphrases with kp. Given a document d, $S(kp)$ is the set of sentences $s \in d$ in which kp appears. Given a sentence $s \in S(kp)$, we denote with $|s|$ the number of words in s, with $|kp|$ the number of words in kp, with $|S(kp)|$ the number of sentences in the set, and so on. Finally, $|C(d)|$ is the number of clauses in the document d, which are defined as a "simple sentences" or, more precisely, the smallest grammatical units which can express a complete proposition[3].

[3]Here we use *clause* and *proposition* as defined in (Kroeger, 2005).

3.3 First approach: Use of anaphora in Machine Learning features

As our first approach we decided to use linguistic knowledge to produce some new features. In detail, we designed a statistical feature that counts all the pronouns/pronominal anaphors which point to an entity (the *antecedent*), and a feature based on on lexical noun phrase anaphors, which are realized as definite noun phrases and proper names (Mitkov, 2014). We will call them *nominal anaphors* and *proper name anaphors* respectively.

For the first feature we follow this process: first, we use the Stanford CoreNLP Coreference Resolution System to find all the anaphors contained in a text and link them to their antecedent. Then, we select the pronominal anaphors, which are anaphors identified by personal pronouns *(he, she, ...)*, reflexive pronouns *(him, her, ...)*, possessive pronouns *(himself, itself, ...)*, demonstrative pronouns *(that, those, ...)*, and relative pronouns *(which, who, ...)*. Finally, we normalize the counted references for each antecedent dividing them by the number of clauses in the document.

Formally, we call $PA(kp)$ the set of pronominal anaphors for which kp is the antecedent. We define:

$$numOfReference(kp) = \frac{|PA(kp)|}{|C(d)|}$$

In our opinion, the use of sentences for normalization is not correct because within a sentence we could find more than one pronoun, skewing the normalization. If we choose the number of clauses to normalize the feature we are instead sure that $0 \leq numOfReference \leq 1$. For clarity, consider the "this book" example from (Mitkov, 2014) from in Section 2: by normalizing over sentences, the value of the feature would be $\frac{2}{1} = 2$, while by normalizing over clauses the value of the feature is $\frac{2}{5} = 0.4$.

The other linguistic feature we implemented is based on *nominal* and *proper name* anaphors. A nominal anaphora instead arises when the referring expression has a non-pronominal noun phrase as its antecedent: it is the case of clauses in which anaphora and antecedent are implicitly related, i.e., they do not stand in a structural or grammatical relationship, but they are linked by a strong semantic one. Consider this example from Wikipedia[4]:

> Margaret Heafield Hamilton (born August 17, 1936) is a computer scientist, systems engineer and business owner. She was Director of the Software Engineering Division of the MIT Instrumentation Laboratory, which developed on-board flight software for the Apollo space program. In 1986, she became the founder and CEO of Hamilton Technologies, Inc. in Cambridge, Massachusetts. The company was developed around the Universal Systems Language based on her paradigm of Development Before the Fact (DBTF) for systems and software design.

Here "Margaret H. Hamilton" is the $antecedent$ and the corresponding anaphors are the underlined words in the quote. "Computer scientist", "Director of the Software Engineering Division" are all examples of *nominal anaphors*.

The Wikipedia excerpt continues with this sentence:

> Hamilton has published over 130 papers, proceedings, and reports about the 60 projects and six major programs in which she has been involved.

Here, "Hamilton" is a proper name referring to "Margaret Heafiled Hamilton", and realizes a *proper name anaphora*.

When we read these sentences we automatically link for example the concept of being a "computer scientist" to a property of the subject of this sentence, while in AKE this information is lost. Hence, the basic idea behind this feature is to reward all the candidate KPs which appear in a nominal or proper name anaphora because they implicitly refer to the mentioned subject, highlighting important aspects of it.

[4] https://en.wikipedia.org/wiki/Margaret_Hamilton_(scientist)

In details, we process the document as previously defined. Then, for each candidate KP, we count all the times it appears in the set of the lexical noun phrases, i.e., the set of nominal and proper name anaphors. Finally we normalize the obtained score by the total number of appearances of the candidate in the document.

Formally, given a keyphrase $kp \in K$ and the set of the lexical noun phrase anaphors in the document NPA, the *inAnaphora* feature can be computed as follows:

$$inAnaphora(kp) = \frac{|\{a \in NPA | kp = a\}|}{|S(kp)|}$$

3.4 Second approach: Use of anaphora for preprocessing

While the previous approaches are able to capture some information about anaphora, they are not powerful enough to catch other knowledge that anaphora convey. For example, frequency-based features such as TF-IDF, which play an important role in several AKE algorithms since its first introduction in (Frank et al., 1999), may be recalculated using the anaphora in the frequency count as well.

This leads us to a different strategy: transforming the text into something that resembles the original human reading process as described in Section 3.1. To achieve this goal, we add to the our system a preprocessing stage that receives the original text from the dataset and substitutes in it all the non pleonastic pronouns with their antecedent. After this preprocessing phase, we perform AKE as usual and then we evaluate the results.

Consider the example we introduced in Section 3.3. If we apply the pre-processing, the sentence becomes:

> Margaret Heafield Hamilton (born August 17, 1936) is a computer scientist, systems engineer and business owner. Margaret Heafield Hamilton was Director of the Software Engineering Division of the MIT Instrumentation Laboratory, MIT Instrumentation Laboratory developed on-board flight software for the Apollo space program. In 1986, Margaret Heafield Hamilton became the founder and CEO of Hamilton Technologies, Inc. in Cambridge, Massachusetts. The company was developed around the Universal Systems Language based on Margaret Heafield Hamilton paradigm of Development Before the Fact (DBTF) for systems and software design.

Unfortunately, the original articles from the SEMEVAL 2010 Task 5 are transformed into plain text using the UNIX tool `pdftotext`. The output of this tool is a very unstructured text, where not only information about title, sections, etc., is lost, but also figures and tables may be placed inside content paragraphs, sentences may be badly split, and so on. This caused problems with the anaphora resolution and substitution algorithm, whose precision was undermined by these conversion errors. Moreover, these formatting problems may cause an erroneous coreference chain where the effects of an early bad resolution are amplified while going further in the text.

Thus, to improve anaphora resolution (and then substitution) in the original text, we segment the original text into sections. In this way, we improve the reliability of the parsing tree for the sentences and so we obtain more a correct performance in searching the antecedent. To perform this segmentation we use some heuristics to distinguish the title, the authors, and the email addresses, and to detect the boundaries of sections, paragraphs, figures, and so on. As a result, the text to process becomes more similar to the original graphical appearance in the PDF format, it is more structured, and it contains also less errors.

Then, by using this structure, we work on single sections, generating and using the coreference chain with the Stanford CoreNLP Coreference Resolution System to collect all the pronominal anaphors. Finally, we go back to the antecendent of each pronoun detected in the chain and we replace the former with the latter. The text turns out to be simpler but more informative for our AKE algorithm: by replacing the pronouns in the document we have no loss of information, while we are able to recover statistical and semantical information about the antecedents that would be otherwise lost.

The choice of substituting only pronominal anaphors is justified by the fact that nominal anaphors may not be just synonyms but also very different words, possibly with a different meaning. This happens because nominal anaphors have only the property of referring to the same entity in common thus substituting a nominal anaphora with its antecedent could change the meaning of the sentence. For example, from the text above, "Computer scientist", "system engineer", "Director of the Software Engineering Division", are all references to "Margaret Heafield Hamilton", but if we substitute them with the head of chain (i.e. "Margaret Heafield Hamilton"), the meaning of the sentence is completely different. Considering proper name anaphors is worthless as well: replacing a proper name anaphora with its antecedent could lead to a substitution that in our opinion could be useless or wrong. For example, in a biography there could be more people indicated by a common surname. Thus, an arbitrary substitution of all proper name anaphors could be wrong, because the anaphora resolution software may fail to identify the correct subject: in our example, if the head of the coreference chain is "Hamilton", we risk to replace "Margaret Heafield Hamilton" with her husband, whose surname is Hamilton too.

To summarize this approach, we follow this process: first, we parse the article from the dataset and use some heuristics to divide it into correctly formatted sections. Then, we process each section with the Stanford CoreNLP Coreference Resolution System, we collect all the pronominal anaphors, and we replace each anaphora with the correct antecendent. Finally, we submit the preprocessed text to the AKE process along with the value of the *InAnaphora* feature. Regarding to the *numOfReference* feature, which concerns only pronominal anaphors, its use has to be taken into consideration as well because when documents are preprocessed with substitution, different coreference chains could be discovered.

4 Methodology

4.1 Baseline algorithm

In order to evaluate the impact of the proposed features, we used the Distiller framework to implement a baseline keyphrase extraction algorithm with few basic features. In our baseline algorithm candidate KPs are n-grams selected from the text if they match a given set part-of-speech patterns, which is one of the most common way of generating candidates in literature (Hasan and Ng, 2014).

The baseline feature set for our experiment is a set of well-accepted features for AKE, i.e., given a candidate KP, we consider:

- TF-IDF;
- relative position of the first appearance of the candidate (*height*);
- difference between the position of the last and the first appearance (*lifespan*);
- number of appearances of the candidate in the text, normalized by number of sentences.

Then, we consider a new feature set, in which we add to the baseline a fifth feature called Document Phrase Maximality (DPM), introduced by Haddoud et al. (2015). We use this feature because it supposedly should help to discriminate between candidate keyphrases which often appear as substring of another candidate. We deem this feature as necessary because, by using our substitution algorithm, we usually substitute an anaphora with a longer antecedent, thus leading to an increase of frequency of all the words contained by the antecedent. In our example, we will substitute the anaphors with "Margaret Heafield Hamilton", thus increasing the frequency, e.g., of the word "Hamilton", but DPM allows us to assign a *low* score to the single words while assigning an *high* score to the full name of the scientist.

The Machine Learning algorithms we choose are logistic regression, neural networks, and boosted decision trees, since these algorithms have a reputation of being good algorithms in the AKE community (Hasan and Ng, 2014). We used their implementation with the R software, using the `glm`, `nnet` and `C5.0` libraries to train the respective models. We used no particular tweaking on the algorithms; the neural network used was a simple Multi Layer Perceptron (MLP) with one hidden layer. Then, we ranked KPs using the raw probability output by the algorithms.

																P	R	F1	MAP
A	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	0.33	0.33	0.33	0.33
B	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	0.33	0.33	0.33	0.02
C	✓	✓	✓	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓	0.40	0.40	0.40	0.31

Table 1: Hypothetical Precision, Recall, F1-Score and MAP of three systems A, B and C over a document with 15 correct keyphrases. A tick mark (✓) indicates a system assigned correct keyphrase, while an x mark (✗) a wrong one.

We didn't choose a bigger feature set because there is no agreement on which are the "state of the art" features for AKE. The top performing systems use many different strategies: in (Lopez and Romary, 2010), the authors use few features, but a custom "post-machine learning" ranking algorithm; in (You et al., 2013) few features are used, but a different candidate generation algorithm; in (Haddoud et al., 2015), we can find more than 20 features.

Moreover, this simple feature set is enough to get an average performance on the SEMEVAL task. With the MLP, our baseline system showed an F-score of 19.69% on the best 15 keyphrases, which is good enough to be ranked 11^{th} out of 20 contestant in the SEMEVAL 2010 challenge. The same position would be achieved using logistic regression, with a score of 19.22%, while the use of decision trees causes a slip of one position down, with a score of 18.95%.

4.2 Metrics

The usual metrics used in AKE are Precision (P), Recall (R), and F1-Score, but these metrics are not the only ones that we will consider in evaluating our system. In fact, we believe that our proposed system may offer an interesting contribute even if we are just able to provide a *better ranking* of the keyphrases.

As pointed out in (Schluter, 2015), this better ranking could not be caught by the aforementioned metrics. For example, suppose we have two systems participating in SEMEVAL 2010, where algorithms are ranked by the F1-Score of the top 15 keyphrases returned by the algorithms. We call this systems A and B. Suppose that, looking at A's output, only the first 5 keyphrases are correct, while the other 10 are wrong. Then, suppose that B's output is the opposite: the first 10 keyphrases are wrong, then the next 5 are good. So, this system will have the same precision (33%), the same recall and the same F1-Score, but the ranking of the system A is arguably better than the ranking of the system B.

Therefore, to evaluate our system, we propose the use of the Mean Average Precision (MAP) metric, which is more suitable to evaluate a ranking than simple precision and recall. We borrow our definition of MAP for keyphrase extraction from (Manning et al., 2008), that is, if the set of correct keyphrases for a document D is $\{kp_{(D,1)}, \ldots, kp_{(D,n)}\}$ and $R_{D,k}$ is the set of retrieved keyphrases for the document D until you get to the k-th keyphrase,

$$MAP(D,R) = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{j}\sum_{k=1}^{j} Precision(R_{D,k})$$

Where $Precision(R_{D,k})$ is the Precision@k score of the system over document D for the first k retrieved documents.

As an example, we see the systems A and B compared in Table 1, on a document with 15 gold keyphrases. The two systems share the same P, R and F1 scores, while MAP is radically different, being much higher for system A than for system B. If we suppose to have another system C, which gets the 1^{st},2^{nd},3^{rd},5^{th},10^{th} and 15^{th} keyphrases correct, we have that this system shows higher P, R and F1 scores than systems A and B but lower MAP than A. This happens because of A's better quality of the first results or, in other words, because of the higher *"weight"* of A's correct keyphrase in the fourth position than C's correct keyphrases in tenth and fifteenth position.

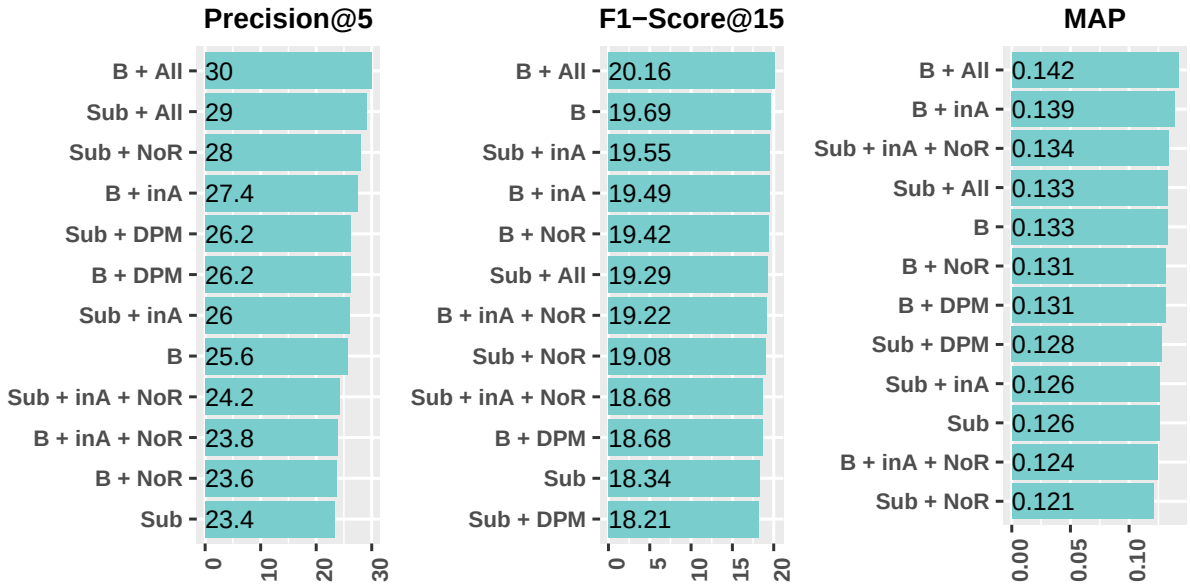


Figure 1: Scores obtained by running our keyphrase extraction algorithm over different feature sets. Note that "B" stands for "Baseline", "Sub" indicates the baseline features ran on the preprocessed documents, "inA" marks the $inAnaphora$ feature, "NoR" marks the $numberOfReference$ feature, "DPM" stands for the Document Phrase Maximality feature, and "All" indicates that the feature set contains all the aforementioned features.

5 Results

Combining the baseline features defined in Section 4.1 with our new features defined in Section 3.3 and the pre-processing technique we described in Section 3.4, we defined a total of 36 different AKE pipelines. These pipelines were built by running the three machine learning algorithms we choose on baseline feature set first, then adding DPM and our anaphora-based features, then combining the features together, both on the original SEMEVAL 2010 dataset and the text preprocessed with our technique.

The neural network was the best performing algorithm overall, and we can see its results in Figure 1. The first impression is that there is generally a little improvement in F1-Score, with the baseline algorithm on the original documents still being the second best feature set with this metric.

Nevertheless, looking at the Precision@5 score, we see that our approach has a significant impact: as shown in Figure 1 (left column), the combination of the linguistic features with the statistical ones, both in the original documents and in the ones with preprocessing, the precision score is greater than the one obtained for the baseline set. In particular, starting from the result of 25.60% for the baseline, we reach a score of 27.60% in precision just by using *inAnaphora* feature and 30.00% adding also *numOfReference* and *DPM*. A similar behavior can be observed in the results when considering that the substitution technique is performed in the preprocessing. In fact, while on the preprocessed text the baseline features show no improvement, a more interesting result can be seen using the linguistic features, for which precision score raises from the baseline's 23.40%, to 26.00% adding *inAnaphora*, and to 29.00% combining all the features together.

Looking at the scores of MAP, we can see that using all features on the original dataset offers the best ranking, as it would be expected by the high Precision@5 score; this confirms our idea of combining the anaphora-based features with DPM. Interestingly enough, most of the other feature sets show a slight decrease in the quality of the ranking, probably because the gain in precision is not high enough to balance the decrease in recall.

Taking into account just the second approach, the results provide evidence of our initial assumptions on the importance of using *inAnaphora* feature over preprocessed text. Precision and F1-Score show a more significant increment when using preprocessed documents, and the reason can be found in a second

parsing with more *coherent* coreference chains. In details, coreference resolution and so our features that depend from it improve because the substitution of pronouns with the common antecedent in the first chain produces a text with more noun phrases and less pronouns. This way, the parse tree of the preprocessed text is simpler, so the relationships between the "new" noun phrases are more clear. This allows the anaphora resolution library to find more anaphors and to better detect pleonastic pronouns, thus obtaining a more precise score for our feature.

The other ML algorithms (not shown in the figures) seem to prove the conclusions we obtain from the neural network: using either decision trees or logistic regression the behavior is similar to the one described for the neural network, with a relatively stable F1-Score on the top 15 extracted keyphrases, but a significant increase in Precision@5 and MAP score when adding linguistic features. It is interesting that for both algorithms, using all features on the original documents offers highest Precision@5 and MAP scores, confirming the results shown in Figure 1. In particular, with this features/dataset combination, with the `glm` library we see slight rises in Precision@5, F1-Score@15 and MAP from 24% to 24.4%, from 19.22% to 20.30% and from 0.127 o 0.136 respectively; for `C5.0`, while F1-Score rises from 18.95% to just 19.29%, Precision@5 and MAP shows a more significant improvement from 22.2% to 26.8% and from 0.123 to 0.137 respectively, thus supposedly showing a better ranking of the keyphrases found. On the other hand, using the same feature set over the preprocessed documents still shows an improvement from the baseline, but with slightly lower scores.

6 Conclusions

Our analysis shows that anaphora and coreference resolution can be used for AKE with significant results. Like in (Hulth, 2003), we see that by exploiting linguistic knowledge in a keyphrase extraction algorithm it is possible to increase the precision of the results. We think that it is important to analyze the relationships which could arise when linguistic features are combined together with statistical features. For example, it is clear that preprocessing the input text by substituting the pronouns with the entity they refer to could increase the frequency of certain terms, thus statistical features like DPM can be useful to gain a performance boost.

A better result could be obtained by improving anaphora resolution performance, since the software we used was not always able to find all the correct anaphors, even if it is (or it is close to) the state of the art system for anaphora and coreference resolution at the time of writing. For example, looking at the example we introduced in Section 3.3, the algorithm was not able to detect the anaphora *director* from the sentence "*She was Director of the Software Engineering Division*", which means that we would not able to detect and replace correctly all the pronouns in the coreference chains or compute the value of our features correctly. This is confirmed by the fact that the *numOfReference* feature, which is based on the count of pronominal anaphors, had a positive impact on performance even after the text preprocessing, which *should* have had replaced all the pronouns with their antecedents.

As a future work, we consider the idea of using the outcomes of our preprocessing stage for improving anaphora resolution specifically for the task of keyphrase extraction, developing an ad-hoc mining algorithm for the parsing trees, with the goal of producing a better pre-processing algorithm and finding for each valuable pronoun a good candidate antecedent. Moreover, another interesting approach would be looking for statistical features other than DPM which are able to better interact with the anaphora-related ones.

References

Marco Basaldella, Dario De Nart, and Carlo Tasso. 2015. Introducing Distiller: a unifying framework for knowledge extraction. In *Proceedings of 1st AI*IA Workshop on Intelligent Techniques At Libraries and Archives co-located with the XIV Conference of the Italian Association for Artificial Intelligence (AI*IA 2015)*. Associazione Italiana per l'Intelligenza Artificiale.

Dario De Nart, Dante Degl'Innocenti, Andrea Pavan, Marco Basaldella, and Carlo Tasso. 2015. Modelling the user modelling community (and other communities as well). In *User Modeling, Adaptation and Personalization:*

23rd International Conference, UMAP 2015, Dublin, Ireland, June 29 – July 3, 2015. Proceedings, pages 357–363. Springer International Publishing.

Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, IJCAI '99, pages 668–673. Morgan Kaufmann Publishers Inc.

Niyu Ge, John Hale, and Eugene Charniak. 1998. A statistical approach to anaphora resolution. In *In Proceedings of the Sixth Workshop on Very Large Corpora*, pages 161–170.

Carl Gutwin, Gordon Paynter, Ian Witten, Craig Nevill-Manning, and Eibe Frank. 1999. Improving browsing in digital libraries with keyphrase indexes. *Decision Support Systems*, 27(1):81–104.

Mounia Haddoud, Aïcha Mokhtari, Thierry Lecroq, and Saïd Abdeddaïm. 2015. Accurate keyphrase extraction from scientific papers by mining linguistic information. In *Proceedings of the Workshop Mining Scientific Papers: Computational Linguistics and Bibliometrics, 15th International Society of Scientometrics and Informetrics Conference (ISSI), Istanbul, Turkey*.

Michael Alexander Kirkwood Halliday and Ruqaiya Hasan. 2014. *Cohesion in english*. Routledge.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1262–1273. Association for Computational Linguistics.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, pages 216–223. Association for Computational Linguistics.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26. Association for Computational Linguistics.

Paul R Kroeger. 2005. *Analyzing grammar: An introduction*. Cambridge University Press.

Shalom Lappin and Herbert J Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational linguistics*, 20(4):535–561.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 257–266. Association for Computational Linguistics.

Patrice Lopez and Laurent Romary. 2010. HUMB: Automatic key term extraction from scientific articles in GROBID. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 248–251. Association for Computational Linguistics.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *An Introduction to Information Retrieval*. Cambridge University Press.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 404–411, Barcelona, Spain, July. Association for Computational Linguistics.

Ruslan Mitkov. 2014. *Anaphora resolution*. Routledge.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers*, pages 317–326. Springer.

Nirmala Pudota, Antonina Dattolo, Andrea Baruzzo, and Carlo Tasso. 2010. A new domain independent keyphrase extraction system. In *Digital Libraries: 6th Italian Research Conference, IRCDL 2010, Padua, Italy, January 28-29, 2010. Revised Selected Papers*, pages 67–78. Springer Berlin Heidelberg.

Natalie Schluter. 2015. A critical survey on measuring success in rank-based keyword assignment to documents. *22eme Traitement Automatique des Langues Naturelles, Caen*.

Takashi Tomokiyo and Matthew Hurst. 2003. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment - Volume 18*, MWE '03, pages 33–40. Association for Computational Linguistics.

Peter D. Turney. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, pages 855–860.

Wei You, Dominique Fontaine, and Jean-Paul Barthès. 2013. An automatic keyphrase extraction system for scientific documents. *Knowledge and Information Systems*, 34(3):691–724.

Retrieving Occurrences of Grammatical Constructions

Anna Ehrlemark and **Richard Johansson** and **Benjamin Lyngfelt**
University of Gothenburg
ehrlemark@gmail.com, richard.johansson@gu.se
benjamin.lyngfelt@svenska.gu.se

Abstract

Finding authentic examples of *grammatical constructions* is central in constructionist approaches to linguistics, language processing, and second language learning. In this paper, we address this problem as an information retrieval (IR) task. To facilitate research in this area, we built a benchmark collection by annotating the occurrences of six constructions in a Swedish corpus. Furthermore, we implemented a simple and flexible retrieval system for finding construction occurrences, in which the user specifies a ranking function using lexical-semantic similarities (lexicon-based or distributional). The system was evaluated using standard IR metrics on the new benchmark, and we saw that lexical-semantical rerankers improve significantly over a purely surface-oriented system, but must be carefully tailored for each individual construction.

1 Introduction

Linguistic theories based on the notion of *constructions* are a recent development in linguistics, which has revitalized the discussion in fields such as syntax, lexicography, and argument structure theory. In particular, it is useful for describing *partially schematic* constructions: templatic patterns that exhibit lexical as well as syntactic properties, which are too specific to be referred to general grammar rules, but too general to be attributed to specific lexical units, which is a reason why they historically have been overlooked (Fillmore et al., 1988). These constructions are very frequent, and they are not only theoretically interesting but also important in language teaching (De Knoop and Gilquin, 2016), since they are challenging for second language learners (Prentice and Sköldberg, 2011).

So far, despite the importance of construction-based theory in linguistics and language pedagogy, it has seen limited adoption in natural language processing. A major impediment to their acceptance is probably the lack of resources of a nontrivial size. Efforts to build inventories of this kind – *constructicons* – are now underway for a number of languages. The emerging practice of constructicon development, or *constructicography*, can be seen as a combination of construction grammar and lexicography (Fillmore et al., 2012). A crucial requirement for constructicon building is access to corpus search tools allowing constructicographers to *search* for occurrences of constructions: either for finding prototypical examples (Gries, 2003), or for computing statistics such as word–construction cooccurrence (Stefanowitsch and Gries, 2003). Also, for the reasons mentioned above, finding authentic examples of uses of a particular construction is also useful in language teaching situations.

In this paper, we cast the task of searching for construction occurrences as an information retrieval (IR) problem. This is fruitful in construction-based research and constructicography for a number of reasons. First, while a partially schematic construction can have a surface form that is easy to describe as a structural pattern, its exact semantic properties and restrictions can be hard to capture as a clear-cut binary decision, or may not even be known *a priori*. This makes it natural to re-rank corpus hits according to a semantics-based scoring function. Such an approach allows a researcher to explore the spectrum of search hits, from the most prototypical – which a good retrieval system should place near the top – to the more unusual cases. Also, the IR perspective is natural in terms of interaction: a user

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 815–824, Osaka, Japan, December 11-17 2016.

can pose queries, rank and re-rank repeatedly according to different functions defined on the fly, to get a comprehensive overview of the various uses of a construction in a corpus.

To open up new opportunities for research in construction retrieval, we built a benchmark collection by annotating the occurrences of six different partially schematic constructions defined in the Swedish Constructicon (Sköldberg et al., 2013). This allows us to explore different retrieval systems and evaluate them according to standard IR evaluation protocols, and we developed a construction retrieval system that finds occurrences of partially schematic constructions in Swedish corpora, based on flexible user-defined ranking functions using lexicon-based and distributional similarities to describe the construction's slot fillers. These ranking functions are learned from a small number of seed examples. Our results show that ranking functions based on lexical-semantic properties are effective, outperforming a purely surface-based baseline for all six constructions. However, because the constructions are so structurally and semantically diverse, we cannot expect a single ranking function to be the best in all cases, and in practice we observed considerable difference among constructions as to which similarities are most useful. This shows that it is crucial for the ranking functions to be flexible and user-defined, which also makes most sense from a usability perspective.

2 Construction Grammar and the Swedish Constructicon

In terms of linguistic theory, the approach is couched in Construction Grammar, for which the aforementioned abundance of semi-general and partially schematic patterns is a key motivation, and where intermingling of linguistic levels is seen as the norm, not the exception. Rejecting the sharp distinction between lexicon and grammar, constructionists regard grammatical rules, lexical idiosyncrasies and "mixed" patterns alike as *constructions*: "conventional, learned form–function pairings at varying levels of complexity and abstraction" (Goldberg 2013, p17). A methodological benefit of assuming the same kind of unit across the board is that the machinery required to account for patterns with both grammatical and lexical properties can also handle those that are purely grammatical or purely lexical, another that the absence of distinct levels eliminates the problem of borderline cases (Fillmore et al., 1988).

Applying this view to descriptive practice, a constructicon is a collection of construction descriptions, a "dictionary of constructions." First introduced for English (Fillmore et al., 2012), there are now constructicon resources under development for a number of languages, including Swedish (Sköldberg et al., 2013; Lyngfelt et al., forthcoming), Brazilian Portuguese (Torrent et al., 2014), and Japanese (Ohara, 2013). The Swedish Constructicon currently covers around 400 constructions, many of which are partially schematic patterns of the kind that is hard to account for from a grammatical or lexical perspective alone. Hence, offering a wide and relevant selection of constructions, it provides a suitable testing ground for the study at hand.

To exemplify the organization of the Swedish Constructicon, Figure 1 shows the most important parts of its entry for V_REFL.RÖRELSE (REFLEXIVE_MOTION), a frequent construction in which motion is expressed using a verb with a reflexive pronoun. The entry contains a *definition* of the construction, that presents its structure and semantics textually, a *structure sketch* describing its surface structure in a semi-formal way, and a small number of annotated *examples* from corpora that show typical uses of the construction.

Name	V_REFL.RÖRELSE
Definition	[An actor]$_{\text{ACTOR}}$, expressed with a [reflexive]$_{\text{REFL}}$, [moves]$_{\text{V}}$ [in a direction, traverses a path, from a place or to a place]$_{\text{LOCATIVE}}$. The verb usually describes the manner of the motion. The construction also encompasses actions that indicate intended motion and non-motion.
Structure sketch	V PN$_{\text{REFL}}$ [PP | ADVP]
Example	[*Vi*]$_{\text{ACTOR}}$ *fick* [*armbåga*]$_{\text{V}}$ [*oss*]$_{\text{REFL}}$ [*fram*]$_{\text{LOCATIVE}}$ *i restaurangen*. [We]$_{\text{ACTOR}}$ had to [elbow]$_{\text{V}}$ [ourselves]$_{\text{REFL}}$ [forward]$_{\text{LOCATIVE}}$ in the restaurant.

Figure 1: Swedish Constructicon entry for V_REFL.RÖRELSE (REFLEXIVE_MOTION).

3 Related work

While we are aware of no previous work approaching the general problem of searching for construction occurrences from a retrieval perspective, quantitative and corpus-based methods have been a crucial part of the construction-linguistic toolbox from its early days (Gries, 2003; Stefanowitsch and Gries, 2003; Hilpert, 2013). In particular, a number of automatic methods have been presented that mine corpora for frequent patterns. For instance, Wible and Tsao (2010) collected generalized n-grams (that is, combinations of words, PoS tags and phrase labels) and applied standard measures of collocational strength to select n-grams that seem to be recurrent patterns. The Swedish constructicon project used similar methods (Forsberg et al., 2014), but also extended the approach by Wible and Tsao (2010) by considering patterns containing phrase labels (e.g. NP-*and*-NP, *as*-Adj-*as*-NP).

The only related work we found that treats the construction detection as a ranking task is by Dubremetz and Nivre (2015), who used a ranking approach to retrieve occurrences of the rare rhetorical *chiasmus* construction. They preferred ranking over classification since the complexity of this construction and the number of borderline cases made a hard classification infeasible, and similarly to our position, they argued that the existence of borderline cases should be embraced instead of ignored. However, since their work is limited to searching for just *chiasmi*, their system was completely tailored for this case.

4 A benchmark collection for evaluating construction retrieval systems

We built a new benchmark for evaluating construction retrieval systems based on six different constructions defined in the Swedish Constructicon. The selected constructions are nontrivial in the sense that their formal description cannot be translated into a standard corpus search query (based on surface features such as words or part-of-speech tags) that captures exactly their true instances – for instance, occurrences of the well-studied *let alone* construction (Fillmore et al., 1988) are easy to spot in corpora.

The initial pass of creating the benchmark was to extract a collection of potential instances for each of the six constructions. We extracted these instances by querying the *Korp* corpus search engine, hosted by the Swedish Language Bank (Borin et al., 2012). Korp stores a large collection of corpora, currently around 10 billion tokens, and allows structural queries based on surface strings but also several types of linguistic annotation; for this experiment, we used a corpus of contemporary fiction. The web service of Korp allows users to pose queries using the CQP language (Christ, 1994), and we selected the instances by translating the structure sketch (as in the example in Figure 1) of each construction into a corresponding CQP query. The corpus search engine then returns a (possibly very large) number of hits, and depending on the formal properties of the construction, fixed lexical content, variable tokens, or syntactic restrictions of particular slots, this list is 'contaminated' to some extent by unrelated hits.

Each collection of hits was then annotated manually as true or false instances of the construction in question. Table 1 shows the number of hits for each query, and the number of true hits among them.

Construction	Total hits	True hits
V_*av*_NP	501	169
PROPORTION_*i/om*	1703	205
REFLEXIVE_MOTION	1300	338
QUANTIFYING_GENITIVE:TIME	945	97
QUANTIFYING_GENITIVE:SCALE	945	166
BOUNDED_EVENT_*på*	2000	43

Table 1: Statistics for the benchmark.

In the following, we detail the six constructions used to create the benchmark.[1]

4.1 V *av* NP

V_*av*_NP is a causal VP construction where the event or state expressed by the verb is caused by the bare noun following the preposition *av* 'of'. It includes both literal and metaphorical causative relations such as *stinka av mögel* 'stink of mold' and *dö av skam* 'die of shame' but the formal surface description

[1]For clarity, we refer to the constructions by translating their Swedish names into English in the rest of this paper, except for construction-evoking keywords included in the names.

V *av* NP_{BARE} also catches phrasal verbs such as *dra av moms* 'subtract sales tax' and passive voice constructions like *läses av flickor* 'is read by girls'. The variable construction slots are not restricted to any obvious semantic classes, although emotions and physical states are clearly salient at first glance.

4.2 PROPORTION_*i/om*

PROPORTION_*i/om* is a rate construction that combines two entities, a numerator and a denominator, joined by the preposition *i* 'in' or *om* 'about', and is restricted to temporal relations such as frequency and speed, and salary rates also fit into this scheme. Its structure sketch NP_{INDEF} [*i* | *om*] N_{DEF} catches true instances such as *två gånger om dagen* 'two times a day' and *åttio pesos i månaden* 'eighty pesos a month', as well as false positives such as *tio dagar i fängelset* 'ten days in jail'.

4.3 REFLEXIVE_MOTION

As mentioned in Section 2, REFLEXIVE_MOTION is a self-motion construction where an actor expressed with a reflexive traverses a path in a direction from a place or towards a goal. The verb typically describes the means or manner of the motion, while the prepositional/adverbial phrase contributes the direction. The formal description V PN_{REFL} [PP | ADVP] captures prototypical occurrences like *sätta sig ned* 'sit down' and *ta sig fram* 'make one's way' as well as instances with verbs that do not usually indicate motion, like *svetta sig igenom* 'sweat one's way through' and *läsa sig bakåt* 'read one's way backwards'. False hits in the search result include common reflexive verbs that do not express motion, like *känna sig glad* 'feel happy' and *anförtro sig åt* 'confide in' as well as certain lexicalized multiword expressions like *tränga sig på* 'intrude' and *sätta sig på tvären* 'be obstinate'.

4.4 QUANTIFYING_GENITIVE:TIME

The QUANTIFYING_GENITIVE:TIME construction is defined as a genitive modifier that specifies the duration of an activity. The formal description DET N_{GEN} N_{INDEF} captures true instances of the construction such as *en stunds tystnad* 'a moment's silence', but also the related scale construction *fem meters djup* 'five meter's depth' (described below), as well as other genitives like *en människas skugga* 'the shadow of a human'.

4.5 QUANTIFYING_GENITIVE:SCALE

The scale construction QUANTIFYING_GENITIVE:SCALE is the sibling of the time construction described above. Here, the genitive modifier specifies the value on a scale expressed by the noun phrase. It shares the formal description DET N_{GEN} N_{INDEF} with QUANTIFYING_GENITIVE:TIME but is semantically restricted to scalable measures like height, depth, age, size and other things that can be quantified. Consequently, the query captures true instances of the scale construction like *tusen meters höjd* 'a thousand meter's height' and *elva månaders hyra* 'eleven month's rent', but also instances of the aforementioned time construction *tre timmars seglats* 'three hour's sailing trip'. Again, other genitive phrases like *en kvinnas fot* 'the foot of a woman' also end up in the search batch.

4.6 BOUNDED_EVENT_*på*

The BOUNDED_EVENT_*på* construction is a time expression that modifies the duration in time of a completed action, expressed using the preposition *på*, corresponding to the similar English construction using *in*. It is a specific and rather restricted instance of a more general pattern for prepositional time adverbials. The construction can only be used with events of bounded aspect, and thereby specifies the time required to complete the event. A related but separate construction is used for negated events, where time adverbials with *på* can be used to describe the duration that has passed since an event took place; this construction is called SPECIFIED_TIME:POLARITY. The structure sketch *på* NP naturally translates to a search query that captures all prepositional phrases with the preposition *på*, and even though the noun slot is strictly restricted to time expressions it is impossible to delimit it from related time constructions without taking the wider context into account. The search hits include true instances of the construction such as *rummet tömdes på några sekunder* 'the room was emptied in a few seconds' as well as false

instances like the negative construction *ingen hade samlat ved på ett år* 'nobody had been collecting firewood for a year'. However, most of the answer set consists of typical PPs like *på en stol* 'on a chair'.

5 Construction retrieval with lexical-semantic reranking

The new benchmark allows us to investigate and compare different systems for retrieving occurrences of constructions in corpora. We now describe the implementation of a retrieval system for finding occurrences of constructions: in particular, we discuss how the ranking function is defined, including the various lexical-semantic similarities the user can choose from, and how the ranking function is trained from a few user-selected seed examples.

The system is built on top of the corpus search service *Korp*, described in Section 4. This is the most comprehensive service of this kind for Swedish, but any similar search tool could be used if building a similar retrieval system for other languages. As when creating the benchmark, the first step of searching for occurrences is to call the underlying corpus search system with a query that describes the surface form of the construction (corresponding to the structure sketch in Figure 1). For instance, if we are looking for occurrences of the QUANTIFYING_GENITIVE:TIME construction (e.g. *an hour's rest*), we use a query corresponding to its structure sketch DET N_{GEN} N_{INDEF}. A number of hits are then returned, out of which some are instances of the construction we are looking for, while others are unrelated. For instance, the structural pattern mentioned above will match any genitive, such as *a dog's life*.

To address this problem, we apply a *reranking* function. The user is asked to provide the system with two additional types of information: (1) a number of positive seed examples of sentences containing true instances of the construction she is looking for and (2) what linguistic properties to consider for particular slots in the search string. For instance, a user could say that the ranking function should consider the distributional similarity function based on the second word in the hit, e.g. the time word in the example above, and then select a number of occurrences such as *an hour's rest, three years' study*. With a carefully designed ranking function and representative seed examples, the system can rank time/activity expressions above other expressions matching that surface pattern.

5.1 Training the reranking function

The reranker is a scoring function $R(x)$ applied to each hit x returned by the corpus search. Learning rankers is widely studied in IR (Liu, 2009); in this work, we assume that each hit x can be analyzed using m different similarity functions σ_j, selected by the user on the fly. Assuming there are labeled examples $x_1, \ldots, x_n$, we write the ranker as

$$R(x) = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_{ij} \sigma_j(x, x_i)$$

where α_{ij} is a weight representing the contribution of similarity σ_j applied to the labeled example x_i. If the σ_j are valid kernels, this is the dual form of a linear scoring function, and the α_{ij} can be determined by training a kernelized learner such as the ranking SVM (Joachims, 2002). We prefer to use a simpler learning method that is efficient and that works with just a few examples. For these reasons, we set the weights by computing the *centroid* of the positively labeled instances: that is, by setting all α_{ij} to $1/n$. This method has trivial computation time and allows the use of any similarity function, not just kernels. More complex learners for this scenario, e.g. the one-class SVM (Manevitz and Yousef, 2001), may be considered in future work, and we could also imagine the weights being set manually by the users.

5.2 Similarity functions used in the reranker

The corpus search system takes care of basic surface-oriented features (word forms, morphology, grammatical functions and categories), so the central task of the reranker is to use representations of word meaning to go beyond the simple structural information. We investigate different measures: similarities based on hand-crafted lexicons, and distributional similarity computed from corpora.

5.2.1 Network-based similarity

SALDO (Borin et al., 2013) is a large lexical resource that connects senses of Swedish words into a hierarchical semantic network. To measure similarity between two SALDO entries, we use the measure

by Wu and Palmer (1994), based on proximity in the tree and the depth of the lowest common ancestor. This measure is a number between 1 and 0, where 1 is the score for two identical entries. A complication is that our corpora lacks sense annotation; however, since the first sense dominates overwhelmingly in corpora for most lemmas (Johansson et al., 2016), we use the first sense to compute the similarities.

5.2.2 Frame-based similarity

An alternative lexicon-based similarity function is based on the Swedish FrameNet (Friberg Heppin and Toporowska Gronostaj, 2012). This resource, similar to its English counterpart (Fillmore and Baker, 2009), maps lemmas to one or more *frames*, which for the current purposes can be seen as semantic classes. Intuitively, two words have a similar meaning if they belong to the same frame; for instance, *timme* 'hour' and *minut* 'minute' are related because they both belong to the frame CALENDRIC_UNIT. Again, we have to deal with the lack of word sense annotation in the corpora, so we define the similarity to be 1 if the two words share at least one frame, and 0 otherwise.

5.2.3 Distributional similarity

In a distributional model (Turney and Pantel, 2010), the meaning representation of a word is computed by observing the contexts in which it appears in a corpus. This is represented as a vector, which makes it possible to apply geometric operations – most importantly, to compute a word similarity by using a function such as the cosine. We trained `word2vec` (Mikolov et al., 2013) on a 1-billion mixed corpus, preprocessed by lemmatization and compound splitting. We used the default settings, except the dimensionality which was set to 512. To compute the similarity between two words, we applied the cosine to their lemma vectors.[2] Again, this similarity is at most 1, which happens if the vectors are identical.

6 Experiments

We first investigated the effect of the choice of lexical similarity. Table 2 shows the average precision scores for three different similarities: Wu–Palmer in SALDO, frame-based, and distributional. As a baseline we include a lemma-based similarity corresponding to a simple search with a number of specified lemmas in the variable construction slots (that is, we get exactly what we asked for and nothing else). As seed examples, the rerankers were trained on the first 15 positively labeled instances in the collection.

Construction	lemma	SALDO	frame	distributional
V_*av*_NP	0.69	0.73	0.63	**0.86**
PROPORTION_*i/om*	0.64	0.68	**0.95**	0.74
REFLEXIVE_MOTION	0.59	0.53	**0.61**	0.56
QUANTIFYING_GENITIVE:TIME	0.40	0.48	**0.60**	0.49
QUANTIFYING_GENITIVE:SCALE	0.64	0.63	0.52	**0.68**
BOUNDED_EVENT_*på*	0.43	0.51	0.36	**0.60**

Table 2: Effect of the choice of lexical similarity function.

The result clearly shows that reranking based on a lexical-semantic model can give very strong improvement over the lemma-based baseline. However, it should be noted that there is considerable variation in the result. For instance, for PROPORTION_*i/om* and QUANTIFYING_GENITIVE:TIME, the frame-based reranker outperforms the others significantly. As we will see in the detailed analysis, a likely explanation is that the slot fillers in these constructions have well-defined semantic restrictions that correspond cleanly to FrameNet frames, in particular time-related words (frames such as Calendric_unit and Measure_duration). In the case of V_*av*_NP, QUANTIFYING_GENITIVE:SCALE and BOUNDED_EVENT_på it is instead the distributional model that works best. The distributional model seems to work best when the slot fillers are not restricted to a narrowly defined semantic class (corresponding to a FrameNet frame), but instead belong to a broader semantic domain, like emotional states for V_*av*_NP.

The network-based lexicon similarity does better than the baseline in most cases, but never outperforms the other similarities. It is difficult to speculate about why, but we can at least conclude that FrameNet frames are better at capturing narrowly defined semantic classes and distributional models do better at generalizing beyond taxonomic similarity scores.

[2]We pick the first lemma if lemmatization is ambiguous.

6.1 Qualitative evaluation

We conducted a qualitative evaluation of the system by analyzing each construction search separately. The reranked instances were inspected with a particular focus on false positives, which are telling indicators of the shortcomings of our approach. We will also inspect the lower regions of the reranked list and say something about false negatives – true occurrences that end up near the bottom of the reranked search list. This qualitative evaluation is telling, since it may also exemplify how a constructicographer would describe delimiting characteristics of specific constructions.

6.1.1 V_*av*_NP

We expected the V_*av*_NP construction (e.g. *rodna av ilska* 'blush with anger') to be a hard nut to crack because of its great productivity and high lexical variation, but the distributional similarity performs beyond expectations. In particular, this model succeeds in generalizing beyond seen instances and gives high ranking scores to a wide array of new and creative occurrences of the V_*av*_NP construction.

Although there are no distinct semantic restrictions on the variable slots of the V_*av*_NP constructions, the state or event caused by the noun is typically physical or emotional. Representative verbs include *darra* 'shiver', *lida* 'suffer', *dö* 'die', *gråta* 'cry', *rodna* 'blush', *skälva* 'quiver', *kvida* 'whimper', *flåsa* 'pant' and *skaka* 'shake'. The noun slot is typically occupied by event nouns such as *raseri* 'rage', *smärta* 'pain', *ansträngning* 'exertion', *ängslan* 'anxiety', *migrän* 'migraine', *förälskelse* 'infatuation' and *upphetsning* 'excitement'. The distributional model excels in finding commonalities between these words, while the frame-based similarity is too restricted. The slot fillers display so much lexical variation that the frame-semantic lexicon manages to capture just a fraction of them.

6.1.2 PROPORTION_*i/om*

As mentioned above, it is hardly surprising that the search system is good at detecting the PROPORTION_*i/om* construction or that the best performing similarity feature in this case is frame-based. The rate construction (e.g. *tre gånger om dagen* 'three times a day') is strictly restricted to temporal relations, so the denominator will always be a time-related word belonging to a few well-defined frames.

6.1.3 REFLEXIVE_MOTION

The evaluation results for REFLEXIVE_MOTION (e.g. *pressa sig ut* 'press oneself out') are unimpressive. None of the rankers are particularly good at detecting this construction: the frame-based reranker just barely beats the baseline. When inspecting the results, it seems that there is too much diversity in this set, with creative metaphors frequently used, for the lexical models to be very effective.

On a lighter note, while this construction is diverse, we observed that some subsets of the occurrences can be handled nicely: if we are particularly interested in occurrences of the REFLEXIVE_MOTION construction where the verb has a more specific manner meaning, we can handpick seed examples of that kind. By doing so, sentences like (1) rise to the top of the answer set. This means that even if the overall score is low, the user can still use the search system productively to find a certain important subclass.

(1) Jag tror att han skulle ha [sovit sig igenom] hela eländet.
I think.PRS that he would.AUX have.AUX sleep.PRF REFL through hole misery.DEF
'I think that he would have slept through the whole ordeal.'

6.1.4 QUANTIFYING_GENITIVE:TIME

For the time construction QUANTIFYING_GENITIVE:TIME (e.g. *två timmars vila* 'two hours' rest'), the frame-based reranker clearly outperforms the other similarity functions in this case since the construction is restricted to time expressions, which we have already seen fit neatly into a few particular frames. Since time expressions are such a strong feature for detecting this construction, it comes as no surprise that false positives near the top of the ranked list contain time words as well. Example (2) is a false hit that is in fact an instance of the superficially similar QUANTIFYING_GENITIVE:SCALE construction.

(2) Det var [femton år-s åldersskillnad] mellan oss.
It is.PST fifteen year.PL-GEN age difference.INDEF between us.
'It was fifteen years of age difference between us.'

6.1.5 QUANTIFYING_GENITIVE:SCALE

The lexical similarities give a smaller improvement for the QUANTIFYING_GENITIVE:SCALE than for the related time construction. In particular, while the frame-based feature worked best for the time construction, it receives the lowest average precision in this case. The scalable measures that turn up here are diverse and do not seem to correspond neatly with FrameNet frames.

The distributional similarity does best at generalizing beyond seen examples, and quite impressively hands out high ranking scores to instances as diverse as *50 procents chans* '50 percent chance', *två veckors skäggstubb* 'two weeks' stubble' and *tre spalters bredd* 'three columns' width'. The precision starts to drop when false hits from the QUANTIFYING_GENITIVE:TIME construction and other constructions start to appear.

6.1.6 BOUNDED_EVENT_*på*

Even though this construction (e.g. *på två timmar* 'for two hours') is strictly restricted to time related words, the frame-based reranker performs worst in the evaluation. The explanation is quite straightforward; just spotting the time word is not enough to disambiguate BOUNDED_EVENT_*på* from related time constructions: most of the information that could be used to discriminate can be found outside of the hit. Recall that BOUNDED_EVENT_*på* only occurs with events of bounded lexical aspect.

Since we had little hope that the reranker would be effective for such a contextually dependent construction as BOUNDED_EVENT_*på*, it is a pleasant surprise to see that the distributional model is doing significantly better than the baseline. A possible explanation is that there are lexical preferences at play, beyond the more general time restriction. Short time spans like *på ett ögonblick* 'in an instant' seem more likely to be instances of this construction than cases such as *på en söndag* 'on a Sunday'. Among the top-ranked false positives, we find quite a few hits with the lexicalized phrase *på en gång* 'at once'.

Introducing more contextual information seems necessary for dealing successfully with a construction like this one. However, in this particular case it is not entirely clear how to determine the lexical aspect of the event in an automated fashion. A more straightforward feature to introduce would be negations that can be relatively easily spotted by using a list of negative polarity items.

7 Discussion

We considered the problem of searching for occurrences of a grammatical construction as a retrieval problem, and we created a new benchmark collection with annotated examples for six different constructions defined by the Swedish constructicon. This new resource allows us to investigate the effectiveness of different retrieval models. As a proof of concept, we presented a simple interactive architecture for searching for constructions, where a user provides a number of positive examples (occurrences of the construction) and tailors a ranking function based on a user-defined combination of features, and our benchmark enabled us to carry out detailed quantitative and qualitative investigations of the effect of different models of lexical representation on the retrieval performance of this system. All our experiments were carried out using Swedish, because of the availability of the Swedish Constructicon used to select the constructions in the benchmark, but our approach is general and could be ported to other languages, including English: similar corpus search tools and lexical resources are readily available.

As expected, grammatical constructions are diverse and the ranking function must be tailored for each construction. The most consistent result is that lexical-semantic models based on a constructions' slot fillers improve the reranker, but exactly which of them – lexicon-based or distributional – is most effective depends on the construction. It is important to note that the accuracy of the reranker depends on the construction definition and to which extent such semantic restrictions are in fact at play. The system should therefore be useful in the work of characterizing and defining constructions.

As we have already pointed out in the text, there are several ways to extend this work: more complex learning algorithms for the rankers could be considered, or we could make use of information beyond the slot fillers of the construction. Also, we have now studied the retrieval problem in isolation, but since our main motivation is that the system should be used in the practical work of linguists working with construction grammar, it would be interesting to investigate the usability and interaction aspect as well.

Acknowledgements

We are grateful for the availability of open lexicons, corpora, and technical infrastructures provided by *Språkbanken*, the Swedish Language Bank. The constructicon resource used in this research was developed during the project *A Swedish Constructicon*, which was funded by the Bank of Sweden Tercentenary Foundation (grant P12–0076:1). RJ was supported by the Swedish Research Council under grant 2013–4944, *Distributional methods to represent the meaning of frames and constructions*.

References

Lars Borin, Markus Forsberg, and Johan Roxendal. 2012. Korp – the corpus infrastructure of Språkbanken. In *Proceedings of the Eighth conference on International Language Resources and Evaluation (LREC-2012)*, pages 474–478, Istanbul, Turkey.

Lars Borin, Markus Forsberg, and Lennart Lönngren. 2013. SALDO: a touch of yin to WordNet's yang. *Language Resources and Evaluation*, 47(4):1191–1211.

Oliver Christ. 1994. A modular and flexible architecture for an integrated corpus query system. In *Proceedings of the 3rd Conference on Computational Lexicography and Text Research*, pages 23–32, Budapest, Hungary.

Sabine De Knoop and Gaëtanelle Gilquin, editors. 2016. *Applied Construction Grammar*. Walter de Gruyter.

Marie Dubremetz and Joakim Nivre. 2015. Rhetorical figure detection: the case of chiasmus. In *Proceedings of the Fourth Workshop on Computational Linguistics for Literature*, pages 23–31, Denver, Colorado, USA, June. Association for Computational Linguistics.

Charles J. Fillmore and Collin Baker. 2009. A frames approach to semantic analysis. In B. Heine and H. Narrog, editors, *The Oxford Handbook of Linguistic Analysis*, pages 313–340. OUP, Oxford.

Charles J. Fillmore, Paul Kay, and Mary C. O'Connor. 1988. Regularity and idiomaticity in grammatical constructions. the case of *let alone*. *Language*, 64:501–538.

Charles J. Fillmore, Russell Lee-Goldman, and Russell Rhomieux. 2012. The FrameNet Constructicon. In H. C. Boas and I. A. Sag, editors, *Sign-Based Construction Grammar*, pages 309–372. CSLI Publications, Stanford.

Markus Forsberg, Richard Johansson, Linnéa Bäckström, Lars Borin, Benjamin Lyngfelt, Joel Olofsson, and Julia Prentice. 2014. From construction candidates to constructicon entries. an experiment using semi-automatic methods for identifying constructions in corpora. *Constructions and Frames*, 6(1):114–135.

Karin Friberg Heppin and Maria Toporowska Gronostaj. 2012. The rocky road towards a Swedish FrameNet – creating SweFN. In *Proceedings of the Eighth conference on International Language Resources and Evaluation (LREC-2012)*, pages 256–261, Istanbul, Turkey.

Adele Goldberg. 2013. Constructionist approaches. In Th. Hoffmann and G. Trousdale, editors, *The Oxford Handbook of Construction Grammar*, pages 15–31. OUP, Oxford.

Stefan Th. Gries. 2003. Towards a corpus-based identification of prototypical instances of constructions. *Annual Review of Cognitive Linguistics*, 1:1–27.

Martin Hilpert. 2013. Corpus-based approaches to constructional change. In Th. Hoffmann and G. Trousdale, editors, *The Oxford Handbook of Construction Grammar*, pages 458–477. OUP, Oxford.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142, Edmonton, Canada.

Richard Johansson, Yvonne Adesam, Gerlof Bouma, and Karin Hedberg. 2016. A multi-domain corpus of Swedish word sense annotation. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Portorož, Slovenia.

Tie-Yan Liu. 2009. *Learning to Rank for Information Retrieval*. Springer Berlin Heidelberg.

Benjamin Lyngfelt, Linnéa Bäckström, Lars Borin, Anna Ehrlemark, and Rudolf Rydstedt. forthcoming. Constructicography at work: Theory meets practice in the Swedish Constructicon. In B. Lyngfelt, T. T. Torrent, L. Borin, and K. H. Ohara, editors, *Constructicography: constructicon development across languages*. John Benjamins, Amsterdam.

Larry M. Manevitz and Malik Yousef. 2001. One-class SVMs for document classification. *Journal of Machine Learning Research*, 2:139–154.

Tomáš Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositiona lity. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Kyoko Hirose Ohara. 2013. Toward constructicon building for Japanese in Japanese FrameNet. *Veredas*, 17(1):11–27.

Julia Prentice and Emma Sköldberg. 2011. Figurative word combinations in texts written by adolescents in multilingual school environments. In *Young urban Swedish. Variation and change in multilingual settings*.

Emma Sköldberg, Linnéa Bäckström, Lars Borin, Markus Forsberg, Benjamin Lyngfelt, Leif-Jöran Olsson, Julia Prentice, Rudolf Rydstedt, Sofia Tingsell, and Jonatan Uppström. 2013. Between grammars and dictionaries: a Swedish Constructicon. In *Proceedings of eLex*, pages 310–327, Tallinn, Estonia.

Anatol Stefanowitsch and Stefan Th. Gries. 2003. Collostructions: Investigating the interaction of words and constructions. *International Journal of Corpus Linguistics*, 8(2):209–243.

Tiago Timponi Torrent, Ludmila Meireles Lage, Thais Fernandes Sampaio, Tatiane da Silva Tavares, and Ely Edison da Silva Matos. 2014. Revisiting border conflicts between FrameNet and construction grammar: Annotation policies for the Brazilian Portuguese constructicon. *Constructions and Frames*, 6(1):33–50.

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

David Wible and Nai-Lung Tsao. 2010. StringNet as a computational resource for discovering and investigating linguistic constructions. In *Proceedings of the NAACL HLT Workshop on Extracting and Using Constructions in Computational Linguistics*, pages 25–31, Los Angeles, United States.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, pages 133–138, Las Cruces, United States.

Automatic Extraction of Learner Errors in ESL Sentences Using Linguistically Enhanced Alignments

Mariano Felice **Christopher Bryant** **Ted Briscoe**
ALTA Institute
Computer Laboratory
University of Cambridge Cambridge, UK
{mf501, cjb255, ejb1}@cl.cam.ac.uk

Abstract

We propose a new method of automatically extracting learner errors from parallel English as a Second Language (ESL) sentences in an effort to regularise annotation formats and reduce inconsistencies. Specifically, given an original and corrected sentence, our method first uses a linguistically enhanced alignment algorithm to determine the most likely mappings between tokens, and secondly employs a rule-based function to decide which alignments should be merged. Our method beats all previous approaches on the tested datasets, achieving state-of-the-art results for automatic error extraction.

1 Introduction

Within the field of Machine Translation (MT), one of the first steps of data processing is to align a source sentence with a target sentence. This is necessary because we want to determine which tokens and phrases in the source language map to which equivalent tokens or phrases in the target language. As this would be extremely time consuming to do manually, several tools, such as GIZA++ (Och and Ney, 2003), have been made available to do this automatically.

Within the related field of Grammatical Error Correction (GEC), we similarly want to align a source sentence with a target sentence to map errors to corrections (sometimes referred to as 'correction detection'; see example in Table 1). However, unlike in MT, the source and target sentences in GEC are in the *same* language and so a majority of tokens match. This means alignment is comparatively more straightforward and so it is more feasible to annotate texts manually, rather than automatically. In fact two of the largest publicly available GEC datasets, the First Certificate in English (FCE) corpus (Yannakoudakis et al., 2011) and the National University of Singapore Corpus of Learner English (NUCLE) (Dahlmeier and Ng, 2012), were aligned and annotated manually.

We	took	a	**guide**	tour	**on**		**center**	**city**	.
We	took	a	**guided**	tour	**of**	**the**	**city**	**center**	.

Table 1: A sample alignment between an original uncorrected sentence and its corrected version.

Nevertheless, automatic alignment of GEC data still has several advantages over manual alignment, not least because the latter is slow, laborious work. This is especially important for datasets that do not always contain explicit alignments, such as Lang-8 (Mizumoto et al., 2011), or GEC system output that needs to be aligned to the original uncorrected sentence.

Another important benefit of an automatic alignment is that it tends to be more consistent than a human alignment. For example, within both the FCE and NUCLE, strings such as *has eating* are inconsistently corrected as [*has* → *was*] or [*has eating* → *was eating*] even though they fundamentally equate to the same thing. In fact, the latter seems less desirable given the token *eating* does not actually change. A similar case is [*has eating* → *was eaten*], which is inconsistently realised either as one edit, as above,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 825–835, Osaka, Japan, December 11-17 2016.

or two edits: [*has* → *was*] and [*eating* → *eaten*]. Ultimately, it seems desirable to regularise such edits and hence reduce ambiguity in the data. If all datasets are treated in the same way, this would also make them fully compatible with each other.

Finally, automatic alignment can also simplify the annotation of new data. For instance, Sakaguchi et al. (2016) recently claimed that forcing annotators to annotate grammatical errors within the confines of an error scheme often led to unnatural sounding sentences and that unconstrained editing correlated more with human judgements. As such, if we no longer ask humans to explicitly mark edit boundaries in new data, we would need to extract this information automatically. This is particularly useful for English as a Second Language (ESL) teaching, where teachers could edit text freely and then let a computer delimit the edits.

2 Background

There is very little previous work on automatic alignment of sentences for GEC. The first attempt was made by Swanson and Yamangil (2012), who built a system to align sentences and then classify the non-match tokens type for the purposes of ESL feedback. In particular, they used the well-known Levenshtein distance to align the sentences and then classified any non-matches according to the FCE error scheme (Nicholls, 2003) using a maximum entropy classifier.

One complication noted by Swanson and Yamangil is that edits do not necessarily consist of just a single token. For instance, reordering errors (e.g. [*only can* → *can only*]) or errors involving phrasal verbs (e.g. [*look at* → *watch*]) necessarily consist of more than one token on at least one side of the edit. The Levenshtein distance, however, only aligns individual tokens and so some alignments must be merged in order to obtain multi-token edits. Swanson and Yamangil hence experimented with some basic merging strategies and found that simply merging all adjacent non-match alignments most closely approximated human alignments.

Building on this foundation, Xue and Hwa (2014) carried out an analysis of Swanson and Yamangil's work and found that approximately 70% of all errors in their error type classifier were the result of bad alignments (merged or otherwise). In order to improve on the simple *all-merge* alignment strategy, they hence trained a binary maximum entropy classifier to determine whether edits should be merged or not. They tested this merging classifier on several datasets, including NUCLE and the FCE, and reported improvements of between 5-10% for both alignment and classification compared to Swanson and Yamangil.

Despite these improvements, however, there is still a considerable margin between automatic and human edit annotation. In addition, both approaches require training on existing annotations, which vary across datasets and can often be inconsistent. Ultimately, training on different datasets leads to different results and so undermines any effort towards data standardisation.

3 Automatic Alignment

A high-quality alignment between an original and corrected sentence is crucial for deriving meaningful edits. Unfortunately, however, the most common method of aligning sentences is to use the Levenshtein distance, which only optimises in terms of insertions, deletions and substitutions. This means that, while optimal in terms of edit operation, the alignments do not take linguistic information into account and are hence not optimal in terms of human intuition (see Table 2 (a)). Human alignments, on the other hand, do make use of linguistic information, so we propose automatic alignments should do the same.

3.1 Damerau-Levenshtein

First, however, as noted by Xue and Hwa (2014), another limitation of Levenshtein is that it is unable to handle word order errors. For example, [*only can* → *can only*] is realised as [*only* → ∅], [*can* → *can*] and [∅ → *only*]; in other words, reorderings are treated as deletions followed by insertions of identical tokens. Since we also need to preserve word order errors in the data, we argue that the Damerau-Levenshtein distance is better suited for the task than standard Levenshtein because it allows for token transpositions.

```
function DL_distance_extended(a, b):
  declare d[0..length(a), 0..length(b)]
  for i := 0 to length(a) inclusive do
    d[i, 0] := i
  for j := 0 to length(b) inclusive do
    d[0, j] := j

  for i := 1 to length(a) inclusive do
    for j := 1 to length(b) inclusive do
      if a[i] = b[j] then
        d[i, j] := 0
      else
        d[i, j] := min(d[i-1, j  ] + del_cost(a[i]),
                       d[i  , j-1] + ins_cost(b[j]),
                       d[i-1, j-1] + sub_cost(a[i], b[j]))

      // Damerau-Levenshtein extension for multi-token transpositions
      k = 1
      while i > 1 and j > 1 and (i - k) >= 1 and (j - k) >= 1 and
            d[i-k, j-k] - d[i-k-1, j-k-1] > 0 do
        if sorted(lowercase(a[i-k:i+1])) = sorted(lowercase(b[j-k:j+1])) then
          d[i, j] := min(d[i, j], d[i-k, j-k] + trans_cost(a[i-k:i+1], b[j-k:j+1])
          break
        k += 1

  return d[length(a), length(b)]
```

Listing 1: Damerau-Levenshtein distance allowing for transpositions of arbitrary length.

As the majority of word order errors in NUCLE and FCE data tend to only involve two tokens, this implies that the standard Damerau-Levenshtein distance, which is likewise only able to handle two-token transpositions, is generally sufficient for our purposes. Nevertheless, while it might seem acceptable to ignore the longer word order errors, this ultimately means they will be broken up into smaller and less meaningful edits which will increase the overall number of false positives and false negatives in the alignment.

To overcome this problem, we extend the Damerau-Levenshtein distance to allow for transpositions of arbitrary length, as shown in Listing 1. This is achieved by traversing a diagonal back from the current cell in the cost matrix and looking for a source sequence that would match the target sequence in any order. The cost of a transposition of length n is defined as $n - 1$, which is compatible with the original definition.

3.2 Linguistically Motivated Alignment

In an effort to incorporate linguistic information into the alignment, we replaced the substitution cost in Damerau-Levenshtein with the function shown in Listing 2. In this function, we set the cost to 0 if the original and corrected tokens differ only in case (e.g. [*the* → *The*]), otherwise, the substitution cost is the sum of sub-costs for lemma, part of speech and character differences. Each of these sub-costs is defined as follows:

lemma cost: 0 if tokens share the same lemma or derivationally related form (e.g. 'met' and 'meeting'), otherwise 0.499.

part-of-speech cost: 0 if tokens share the same part of speech, otherwise 0.25 if both tokens are content words (adjectives, adverbs, nouns or verbs) and 0.5 in all other cases.

character cost: the proportion of character mismatches between 0 and 1, computed as the character-level Damerau-Levenshtein distance between the tokens divided by the length of their alignment.

To increase the likelihood of aligning derivationally related forms, we lemmatise each token as if it were an adjective, adverb, noun and verb. We do this because if we only lemmatise for a single part of speech, then we might overlook certain derivationally related words. For example, while the lemma of

```
function substitution(a, b):
  if lowercase(a) = lowercase(b) then
    return 0
  else
    return lemma_cost(a, b) + pos_cost(a, b) + char_cost(a, b)
```

Listing 2: Our linguistically motivated token substitution function.

(a)	This	wide	spread	propaganda	benefits	only	to	the	companys	.
	This	widespread	publicity	only	benefits	their	companies			.
(b)	This	wide	spread	propaganda	benefits	only	to	the	companys	.
	This	widespread		publicity	only	benefits		their	companies	.

Table 2: Differences between (a) standard Levenshtein and (b) linguistically-enriched Damerau-Levenshtein alignment.

the verb 'met' is 'meet', the lemma of the noun 'meeting' is 'meeting', which suggests these words are not related. By also lemmatising 'meeting' as a verb however, we find that the two tokens do share a common lemma, 'meet', which instead correctly suggests they are related and should align. Ultimately, we consider two tokens to be derivationally related if their respective sets of lemmas intersect.

The sub-costs are also set in such a way that the overall substitution function always yields values in the $[0, 2)$ range. Keeping the cost asymptotic to 2 is important to enforce a preference for substitutions over insertions and deletions (both set to 1); this is why we use a lemma cost of 0.499 instead of 0.5. We tried different combinations of these costs, provided they met this condition, but did not find significant differences in the results.

By incorporating all this additional linguistic information into the cost, we improve the likelihood that tokens with a similar etymology, spelling or function will align. This is better than the simple surface matching used by the standard token-level Levenshtein distance and hence, we argue, results in more natural, human-like alignments (see Table 2 (b)). The final alignment is retrieved by collecting the operations that make up the optimal path in the cost matrix. Given that the cost is now dependent upon a variable function, it is often the case that there is just a single optimal alignment.

3.3 Data

We evaluated our improved alignment algorithm using the public FCE (Yannakoudakis et al., 2011), NUCLE corpus (Dahlmeier et al., 2013), and CoNLL test sets (Ng et al., 2013; Ng et al., 2014). While the CoNLL data is available in a pretokenised format, the FCE data is not, and so to keep things comparable, we only worked with the untokenized CoNLL data.

It should be noted that processing each of these datasets in a standard way is not at all straightforward. For example, unlike the CoNLL data, the FCE contains nested edits; e.g. [*entery* → *entry* → *entrance*] indicates a spelling error followed by a replacement noun error. Similarly, the NUCLE corpus can be quite noisy and it is not uncommon for annotators to select entire paragraphs or even essays as edits with comments such as "Rewrite in the 3rd person", which, in the context of edit extraction, should definitely be ignored.

In addition to these, a more general problem concerns converting character level edit spans into token level edit spans; there is no guarantee that a human-annotated character span will map exactly to a token span, which has consequences for token-based processing and evaluation. Similarly, some edits change sentence boundaries, which subsequently makes aligning original sentences with corrected sentences a lot more complicated, especially when there are multiple annotators. Ultimately, we refer the reader to Bryant and Felice (2016) for more information about the challenges involved in processing these datasets and for details about how we overcame them in our implementation.

Having preprocessed the data, we used spaCy[1] v0.101.0 to tokenize (words and sentences), part of

[1]https://spacy.io/

Dataset	Sents	Edits
CoNLL 2013	1,375	3,415
CoNLL 2014 (0)	1,314	2,397
CoNLL 2014 (1)	1,319	3,331
NUCLE	55,963	43,832
FCE-test	2,715	4,776
FCE-train	31,022	48,026

Table 3: Basic statistics of the datasets we use. CoNLL 2014 was annotated by two annotators who changed different sentence boundaries.

Alignment	*Lev*	*Lev*	*DL*
Reference	*Gold*	*Gold-Min*	*Gold-Min*
CoNLL 2013	49.17	62.29	70.51
CoNLL 2014 (0)	51.09	60.40	66.81
CoNLL 2014 (1)	48.41	62.98	69.16
FCE-test	58.52	63.30	72.45

Table 4: Table showing how minimised edits in the reference (Gold-Min) and our linguistically enriched Damerau-Levenshtein algorithm (DL) perform against standard Levenshtein (Lev) and unmodified references (Gold). All scores are F_1.

speech (POS) tag and lemmatise each sentence. The basic statistics of each processed dataset are shown in Table 3.

3.4 Alignment Experiments

To establish a baseline, we simply compared a standard Levenshtein alignment against the human alignments of the CoNLL 2013, CoNLL 2014 (each annotator individually) and FCE test sets (Table 4). The results show that Levenshtein alone does not perform particularly well at this task and is only able to achieve F_1 scores of about 50%.

In addition to improving alignment quality, another aim of our work is to attempt to standardise edit annotation. As mentioned previously, human annotations sometimes include tokens that do not change; e.g. [*has eating* → *was eating*]. Automatic alignments will never match these human edits, however, because any token that is common to both sides will be considered a match and hence not part of an edit. This is undesirable, so we also minimised the gold human reference edits by recursively removing tokens that were common to both sides of the edit from the right and left hand sides. This also removes edits that annotators detected, but were unable to correct. Results showing the effect of this minimisation, as well as of comparing Levenshtein against our linguistically enhanced Damerau-Levenshtein approach, are also shown in Table 4.

The first thing to notice about these results is that the scores for the minimised gold reference are typically substantially higher than for the unmodified gold reference. In the case of CoNLL 2014 (1), using the minimised gold reference even shows an improvement of almost 15% F_1. While the increase in score is less pronounced for FCE-test, at just under 5% F_1, this nevertheless demonstrates the high degree of variability in the way GEC data is annotated and that it is highly desirable to standardise annotations. In light of this result, we only use minimised edit spans in all subsequent experiments.

Comparing Levenshtein against our own approach, we again see a significant improvement, with scores greater than 70% F_1 on some datasets. This is significant, because these results are in spite of the fact that we fail to match all multi-token edits given that we do not yet merge any alignments.

4 Alignment Merging

The output of the automatic alignment is a list of individual token-level operations that map the original sentence to the corrected sentence in terms of insertions, deletions, substitutions and transpositions. An example of this is shown in Table 5. Each of these operations involves one token at most in either sentence, except in the case of transpositions, which involve 2 or more tokens in both sentences.

In most cases, these individual operations already represent a complete error: [*propaganda* → *publicity*] is a word choice error, [*benefits only* → *only benefits*] is a word order error and [*companys* → *companies*] is a noun inflection error. Other errors, however, may involve more than one single operation. For example, the correction [*wide spread* → *widespread*] in Table 5 involves two individual operations: a substitution ([*wide* → *widespread*]) and a deletion ([*spread* → ∅]).

The statistics in Table 6 show that most errors in NUCLE and FCE-train involve only a single token

M	S	D	S	T		D	S	S	M
This	wide	spread	propaganda	benefits	only	to	the	companys	.
This	widespread		publicity	only	benefits		their	companies	.

Table 5: Individual operations obtained from automatic alignment: (M)atch, (I)nsertion, (D)eletion, (S)ubstitution and (T)ransposition.

Orig:Corr Token Edit Size	**NUCLE**		**FCE-train**	
	Cum. Freq.	**%**	**Cum. Freq.**	**%**
1:1	17,580	41.23%	23,833	51.51%
0:1	24,823	58.22%	32,875	71.06%
1:0	30,599	71.77%	37,743	81.58%
0–2:0–2	36,868	86.47%	43,593	94.22%
0–3+:0–3+	42,636	100.00%	46,265	100.00%

Table 6: Distribution of edits in minimised gold NUCLE and FCE-train according to the number of tokens on either side of the edit; e.g. there are 17,580 instances of 1:1 token substitutions in NUCLE. Total edits are lower than in Table 3 because edit minimisation causes some edits to disappear.

on either side of an edit (i.e. 0:1, 1:0 or 1:1), and so a simple *all-split* strategy that merges nothing is likely to cover most of these edits. In fact this explains why the results in Table 4 are so high; just using Damerau-Levenshtein is equivalent to the *all-split* setting. Nevertheless, multi-token edits still form an important class of learner errors and so we should attempt to handle them.

4.1 Merging Rules

In order to improve performance and capture multi-token edits, we hence implemented a recursive rule-based merging function. First, we analysed the relationship between human annotations and how they mapped to alignment operations in NUCLE and FCE-train. For example, we found that the most common multi-token errors involved phrasal verbs, such as [*look at* → *watch*]; possessive nouns, such as [*friends* → *friend 's*]; or orthographic changes, such as [*wide spread* → *widespread*]. Second, we wrote rules to merge or separate alignments based on these observed patterns.

The complete list of rules and their priority is as follows:

1. Any match operation (M) breaks a sequence into sub-sequences that are processed individually, e.g. MDDSMMTMSI is split into DDS, T and SI.
2. Any operation that involves punctuation and is followed by a token that changes case is merged, e.g. [*,* → *.*] + [*we* → *We*] becomes [*, we* → *. We*].
3. Transpositions are returned as individual edits, e.g. [*only can* → *can only*].
4. Any operation that involves a possessive suffix is merged with any previous operations, e.g. [*freinds* → *friend*] + [∅ → *'s*] becomes [*freinds* → *friend 's*].
5. Operations that add or remove whitespace between tokens are merged, even if they have unmatched apostrophes, e.g. [*sub* → *subway*] + [*way* → ∅] = [*sub way* → *subway*].
6. Substitutions between very similar tokens ($> 70\%$ character matches) are returned as individual edits, e.g. [*writting* → *writing*], unless they have the same POS as the previous token, e.g. [*eated* → *have eaten*]; the verb phrase would be split without this exception.
7. Substitutions preceded by another substitution are returned as individual edits.
8. Any combination of operations that involves at least one content word is merged, e.g. [*On* → *In*] + [*the* → ∅] + [*other* → ∅] + [*hand* → *addition*] = [*On the other hand* → *In addition*].
9. Consecutive operations that involve tokens with the same part of speech are merged, e.g. [*because* → ∅] + [*of* → *for*] = [*because of* → *for*].
10. Any determiner at the end of a sequence is returned as an individual edit.

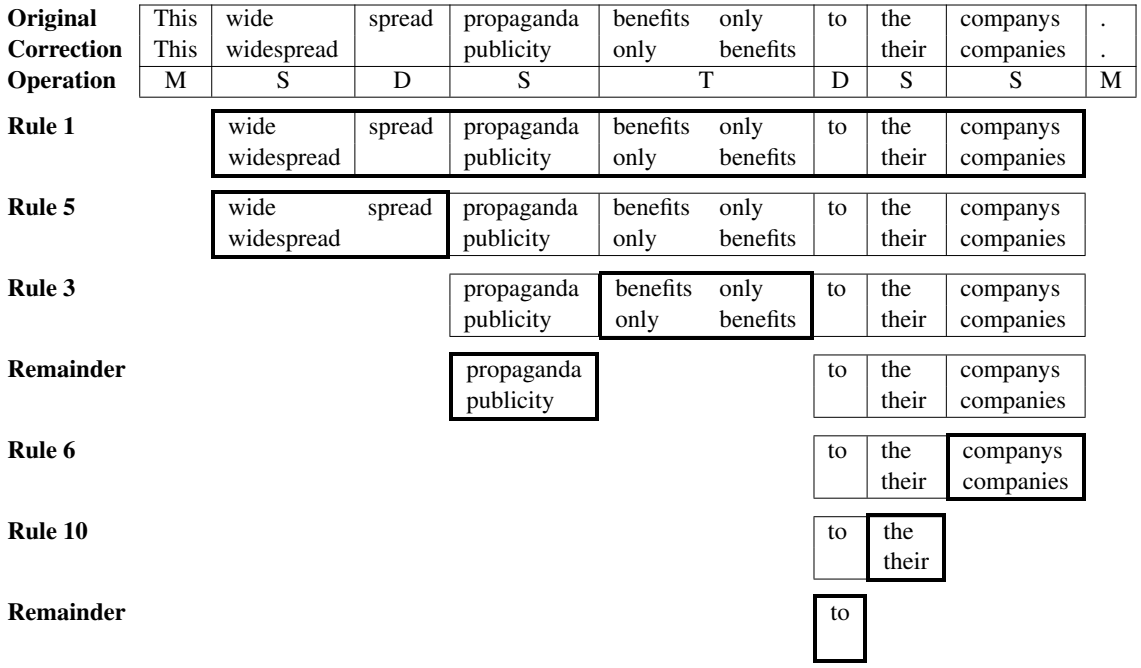


Figure 1: A step-by-step edit extraction example.

Each sequence of alignment operations between two sentences (e.g. MSDSTDSSM) is processed recursively using the above rules in a top-down fashion. Rules are applied in order, with priority relative to their position in the list. Every time an edit is returned by one of the rules, we process the remaining sub-sequences individually until they are exhausted or no more rules can be applied (see Figure 1). It should be noted that rules that iteratively grow the merge range of the alignment (e.g. #8) can be overridden by others with higher priority (e.g. #4), causing the remaining operations in the truncated subsequence to be reprocessed from scratch.

4.2 Merging Experiments

We evaluated our rule-based merging method on the CoNLL 2013, CoNLL 2014 (each annotator individually) and FCE test sets, and contrasted it against the following merging strategies:

all-split: All consecutive non-matches are split, e.g. DDSI → D, D, S, I.

all-merge: All consecutive non-matches are merged, e.g. DDSI → DDSI.

all-equal: All consecutive non-matches of the same operation are merged, e.g. DDSI → DD, S, I.

All of these methods were applied to the output of our enhanced Damerau-Levenshtein alignment described in the previous section. In addition to evaluating edit extraction against a minimised reference, we also evaluate error type classification on the merged output to replicate results for an end-to-end classification system. For this purpose, we retrained Xue and Hwa's publicly available implementation of their MaxEnt error type classifier[2] separately on NUCLE and FCE-train. This classifier was based on work by Swanson and Yamangil and is used in all classification experiments.

Results for both tasks are reported in Table 7 and reveal that our method achieves the best performance on all tasks and datasets. For edit extraction (i.e. merging), improvements in F_1 range between 4% and 12% over the second-best method (*all-merge*). We also observe that while *all-split* tends to have the highest number of TPs and lowest number of FNs, it also has the highest number of FPs, which shows how ignoring multi-token edits affects performance. In contrast, the *all-merge* strategy has the lowest number of FPs, but at the cost of also having the lowest number of TPs and highest number of FNs. This shows that each strategy has different strengths which our rule-based approach attempts to exploit.

[2] https://github.com/xuehuichao/correction_detector

Dataset	Method	Edit Extraction						Edit Extraction + Classification					
		TP	FP	FN	P	R	F_1	TP	FP	FN	P	R	F_1
CoNLL 2013	All-split	2715	1612	659	62.75	80.47	70.51	2088	2239	1286	48.26	61.89	54.23
	All-merge	2194	653	1180	77.06	65.03	70.54	1634	1213	1740	57.39	48.43	52.53
	All-equal	2417	1160	957	67.57	71.64	69.54	1856	1721	1518	51.89	55.01	53.40
	This work	2784	591	590	82.49	82.51	**82.50**	2072	1303	1302	61.39	61.41	**61.40**
CoNLL 2014 (0)	All-split	1858	1320	526	58.46	77.94	66.81	1267	1911	1117	39.87	53.15	45.56
	All-merge	1662	415	722	80.02	69.71	74.51	1062	1015	1322	51.13	44.55	47.61
	All-equal	1705	884	679	65.86	71.52	68.57	1130	1459	1254	43.65	47.40	45.45
	This work	1893	550	491	77.49	79.40	**78.43**	1242	1201	1142	50.84	52.10	**51.46**
CoNLL 2014 (1)	All-split	2635	1699	651	60.80	80.19	69.16	2052	2282	1234	47.35	62.45	53.86
	All-merge	2435	554	851	81.47	74.10	77.61	1791	1198	1495	59.92	54.50	57.08
	All-equal	2453	1174	833	67.63	74.65	70.97	1904	1723	1382	52.50	57.94	55.08
	This work	2866	598	420	82.74	87.22	**84.92**	2139	1325	1147	61.75	65.09	**63.38**
FCE-test	All-split	3660	1936	847	65.40	81.21	72.45	3073	2523	1434	54.91	68.18	60.83
	All-merge	3144	778	1363	80.16	69.76	74.60	2564	1358	1943	65.37	56.89	60.84
	All-equal	3373	1447	1134	69.98	74.84	72.33	2845	1975	1662	59.02	63.12	61.01
	This work	3861	739	646	83.93	85.67	**84.79**	3182	1418	1325	69.17	70.60	**69.88**

Table 7: Performance of different merging methods on the edit extraction and full error classification task. TP: true positives, FP: false positives, FN: false negatives, P: precision, R: recall.

Table 7 also reports performance on error classification, given edit extraction, revealing how each merging strategy affects automatic error type prediction. Results are consistent with edit extraction, although they are (expectedly) lower by an average 21.1% F_1. Improvements in F_1 between our method and the second-best range between 3.9% and 9.0%. While CoNLL 2014 (1) achieved the best result for edit extraction, FCE-test achieved the best result for error classification. This might be because the two datasets are annotated according to different error classification frameworks and the FCE annotation is more consistent than NUCLE annotations.

5 Discussion

It is worth stating that many of our reported results are actually an underestimate of true performance. This is because, despite gold reference minimisation, there is a high degree of variability in the way humans annotate this sort of data. For instance, as shown by Bryant and Ng (2015), human annotators often have very different perceptions of grammaticality and it is linguistically plausible that, for example, [*has eaten* → *was eating*] is annotated either as one edit (as above) or two edits ([*has* → *was*] + [*eaten* → *eating*]) by different, or even the same, annotators. This means the *all-split* merging strategy will never match the former while the *all-merge* merging strategy will never match the latter, even though the annotations fundamentally equate to the same thing. Due to this inconsistency, system performance will be underestimated regardless of which merge strategy you choose.

In contrast, we consider merge consistency a strength of our rule-based approach. Even if our alignment does not agree with the gold standard in some cases, at least the decision to merge or split is consistent across all similar cases. Our approach could hence be used to standardise ambiguous annotations where splits or merges are equally plausible.

Another strength of a rule-based approach is that it is easier to diagnose which rules are responsible for producing a given output sequence. This is in contrast with machine learning techniques, which additionally require feature engineering and retraining, where it is much more difficult to determine why certain edits were merged in a certain way. Nevertheless, considering only about 30% of all edits (at most) in any dataset require merging anyway, a rule-based approach seemed a more cost-effective solution.

We also investigated how our approach compared against previous approaches in terms of single-token edits and multi-token edits (Table 8). To produce comparable results for Xue and Hwa (X&H), we retrained their publicly available implementation of their MaxEnt merging classifier separately on NUCLE and FCE-train. In general, while our method tends to score slightly lower precision than the

Dataset	Method	Single-token edits			Multi-token edits		
		P	**R**	**F_1**	**P**	**R**	**F_1**
CoNLL 2013	S&Y	95.67	65.73	77.92	16.62	40.62	23.59
	X&H	90.34	73.13	80.82	24.53	48.75	32.64
	This work	88.76	87.80	**88.28**	51.00	31.87	**39.23**
CoNLL 2014 (0)	S&Y	95.79	68.59	79.94	24.11	51.52	**32.85**
	X&H	89.44	74.88	81.52	25.70	41.67	31.79
	This work	83.27	85.96	**84.60**	34.57	21.21	26.29
CoNLL 2014 (1)	S&Y	94.11	73.95	82.82	31.56	53.81	39.79
	X&H	90.71	80.87	85.51	38.06	52.38	44.09
	This work	86.32	93.64	**89.83**	71.43	40.48	**51.67**
FCE-test	S&Y	95.19	69.76	80.51	30.19	52.08	38.22
	X&H	82.98	83.90	83.44	66.27	52.72	58.72
	This work	88.35	91.00	**89.65**	74.06	50.16	**59.81**

Table 8: Performance of our proposal vs. previous methods in terms of single and multi-token edits. S&Y used Levenshtein with an *all-merge* strategy while X&H used Levenshtein with a MaxEnt merging classifier.

others in the single-token setting, it makes up for this with a much higher recall. In contrast, our method achieves a higher precision in the multi-token setting, but at the cost of a lower recall. Ultimately, however, our method increases overall performance in almost all cases, the exception being multi-token edits in CoNLL 2014 (0), which is known to be an inconsistent dataset. These results hence confirm that our rule-based merging strategy is superior to previous approaches.

In addition to a quantitative analysis, we also carried out an informal qualitative analysis of the errors made by our system. One source of errors involves tokens that are affected by more than one mistake; e.g. [*wide spraed* → *widespread*]. While our system includes a rule to merge adjacent alignments where the only difference is white space, this rule does not activate in the above case since one of the tokens also contains a misspelling. This consequently means the alignments are not merged and do not match the gold standard; such cases are difficult to handle.

Another issue is that reference minimisation is unable to deal with edits where identical tokens occur in the middle of an edit; e.g. [*can easily been* → *could easily be*]. As an automatic alignment will always consider *easily* a matched token, the remaining non-matches will become isolated and hence never merged. In this case, however, we would argue that our automatic alignment is more informative than the human alignment which needlessly includes a redundant token in the edit.

Finally, we provide a comparison between our method and the previous approaches by Swanson and Yamangil (S&Y) and Xue and Hwa (X&H) (Table 9). Our method achieves state-of-the-art performance on all tasks and datasets, with an average improvement over X&H of 6.0% F_1 for edit extraction and 4.1% F_1 when adding error classification.

6 Conclusion

We have presented a new method for extracting edits from a parallel original and corrected sentence pair based on a linguistically enhanced token alignment and rule-based merging component. Results on a number of GEC test sets show that our method outperforms all previous work on edit extraction and can also boost error classification performance.

Since we only use a few hand-coded rules, we do away with the complexity of machine learning solutions and are hence also able to annotate data much more consistently. This is particularly useful for standardising GEC datasets, which are often annotated using different guidelines.

Dataset	Method	Edit Extraction F_1	Edit Extraction + Classification F_1
CoNLL 2013	S&Y	70.42	52.85
	X&H	74.07	55.89
	This work	**82.50**	**61.40**
CoNLL 2014 (0)	S&Y	72.92	46.95
	X&H	74.25	49.15
	This work	**78.43**	**51.46**
CoNLL 2014 (1)	S&Y	76.39	56.18
	X&H	79.21	59.24
	This work	**84.92**	**63.38**
FCE-test	S&Y	73.59	59.80
	X&H	79.18	65.33
	This work	**84.79**	**69.88**

Table 9: Performance of our proposal vs. previous methods in an end-to-end edit extraction and classification task.

References

Christopher Bryant and Mariano Felice. 2016. Issues in preprocessing current datasets for grammatical error correction. Technical Report UCAM-CL-TR-894, University of Cambridge, Computer Laboratory, September.

Christopher Bryant and Hwee Tou Ng. 2015. How far are we from fully automatic high quality grammatical error correction? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 697–707, Beijing, China, July. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada, June. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The NUS corpus of learner english. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, Atlanta, Georgia, June. Association for Computational Linguistics.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning sns for automated japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155. Asian Federation of Natural Language Processing.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel R. Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. ACL.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland, USA. ACL.

Diane Nicholls. 2003. The cambridge learner corpus: Error coding and analysis for lexicography and ELT. In *Proceedings of the Corpus Linguistics 2003 conference*, pages 572–581.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel Tetreault. 2016. Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association for Computational Linguistics*, 4:169–182.

Ben Swanson and Elif Yamangil. 2012. Correction detection and error type selection as an ESL educational aid. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 357–361, Montréal, Canada, June. Association for Computational Linguistics.

Huichao Xue and Rebecca Hwa. 2014. Improved correction detection in revised esl sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 599–604, Baltimore, Maryland, June. Association for Computational Linguistics.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA, June. Association for Computational Linguistics.

Contrasting Vertical and Horizontal Transmission of Typological Features

Kenji Yamauchi **Yugo Murawaki**
Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
yamauchi@nlp.ist.i.kyoto-u.ac.jp murawaki@i.kyoto-u.ac.jp

Abstract

Linguistic typology provides features that have a potential of uncovering deep phylogenetic relations among the world's languages. One of the key challenges in using typological features for phylogenetic inference is that horizontal (spatial) transmission obscures vertical (phylogenetic) signals. In this paper, we characterize typological features with respect to the relative strength of vertical and horizontal transmission. To do this, we first construct (1) a spatial neighbor graph of languages and (2) a phylogenetic neighbor graph by collapsing known language families. We then develop an autologistic model that predicts a feature's distribution from these two graphs. In the experiments, we managed to separate vertically and/or horizontally stable features from unstable ones, and the results are largely consistent with previous findings.

1 Introduction

Centuries of research in historical linguistics have identified groups of languages deriving from single common ancestors. Each of these groups, called a language family, is organized as a tree that reflects its evolutionary history. Such language families include Indo-European, Austronesian and Bantu languages.

Despite the huge success, historical linguistics fails to link some languages with others, and hence they are called language isolates. For example, Basque, Burushaski and Japanese have no established relatives. Although there are several attempts to uncover deep phylogenetic relations, in which the results are often represented as *macrofamilies*, they remain controversial (Dolgopolsky, 1998; Greenberg, 2000).

We argue that the limitation of the mainstream approach is that it relies on lexical traits. Language families are established by demonstrating that its member languages share *cognates*, or words that have a common etymological origin.[1] Lexical data are also the main target of modern statistical methods that are typically used to date the common ancestor (Swadesh, 1971; Gray and Atkinson, 2003). Language isolates are called so exactly because they lack reliable cognates.

For this reason, we follow a different line to research that makes use of typological data for phylogenetic inference (Tsunoda et al., 1995; Dunn et al., 2005; Teh et al., 2008; Longobardi and Guardiano, 2009; Murawaki, 2015). Linguistic typology is a cross-linguistic study that classifies the world's languages according to structural properties such as basic word order (SVO, SOV, etc.) and the presence or absence of tone. On the one hand, typological features, by definition, allow us to compare an arbitrary pair of languages including language isolates. On the other hand, they pose several new challenges to us. While the sharing of cognates is a direct indicator of shared ancestry, the sharing of the same basic word order SOV, for example, is only a weak signal because it occurred multiple times in multiple places. We believe that computer-intensive statistical methods can help taming the inherent uncertainty.

In this paper, we give a step forward toward typology-based phylogenetic inference. We specifically tackle the problem that horizontal (spatial) transmission obscures vertical (phylogenetic) signals. Like

[1] In historical linguistics, cognates are distinguished from loanwords. We broadly refer to both the traditional *comparative method* and modern statistical methods as lexicon-based approaches because they require cognate identification. Note that historical linguists often use regular sound changes, in addition to cognates themselves, as features to determine the relative order of multiple branching events (Pellard, 2009).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 836–846, Osaka, Japan, December 11-17 2016.

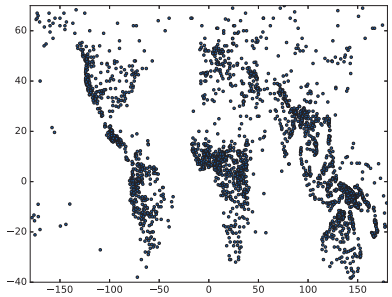


Figure 1: Map of languages in WALS.

loanwords, typological features can be borrowed from one language to another. In fact, non-tree-like evolution has been one of the central topics in linguistic typology (Trubetzkoy, 1928). Lexicon-based approaches, especially statistical ones, often address this problem by narrowing the scope to *basic vocabulary,* or a list of basic concepts. Words on the list are assumed to be resistant to borrowing (Swadesh, 1971), and some are even claimed to be extremely stable (Pagel et al., 2013). By contrast, typological features, in their original forms, are a mosaic with varying degrees of resistibility. We need to start with quantitatively characterizing each typological feature in this respect.

We present a probabilistic model that directly contrasts vertical and horizontal transmission of typological features. We begin by noting that anthropologists have worked on similar problems. From this field, we borrow a graph-based model (Towner et al., 2012) and extend it to model typological features. In this model, languages of the world are mapped to two neighbor graphs. One encodes vertical transmission and the other represents horizontal transmission. These two graphs are used to predict the distribution of a given feature, and the relative strength of the two modes of transmission is inferred. As a practical application, we use this model to impute missing values that are ubiquitous in the typological database.

In the experiments, we first evaluated the proposed model with missing value imputation and confirmed that the proposed model outperformed simple baselines for this task. We then used the model to estimate the relative strength of the two modes of transmission. Our model managed to separate vertically and/or horizontally stable features from unstable ones, and the results are largely consistent with previous findings. Our code is available online at `https://github.com/yustoris/autologistic-coling-2016`.

2 Data and Preprocessing

As the database of typological features, we used the online edition[2] of the World Atlas of Language Structures (WALS) (Haspelmath et al., 2005). As of 2016, it contained 2,679 languages and 192 features. The language–feature matrix was very sparse, however. Less than 15% of elements were present. We focused on 48 features, which covered at least 20% of languages. Some previous studies discarded languages with few observed features (Murawaki, 2015; Takamura et al., 2016). To avoid an unexpected bias, we used all but 72 languages in the database. The languages we excluded were sign languages, pidgins and creoles, which were all classified as *other* in WALS.

An item of the language–feature matrix was a categorical value. For example, Feature 81A "Order of Subject, Object and Verb" has 7 possible values, `SOV`, `SVO`, `VSO`, `VOS`, `OVS`, `OSV` and `No dominant order`, and every language took one of the 7 values. We used these categorical feature values as they were. Although the mergers of some fine-grained feature values seem desirable (Daumé III and Campbell, 2007; Greenhill et al., 2010; Dediu, 2010; Dunn et al., 2011), we leave them for future work.

Languages were associated with single-point geographical coordinates (longitude and latitude), as shown in Figure 1. We constructed a spatial neighbor graph by linking any pair of languages that were within the distance of R km.[3] We set R = 1,000 following da Silva and Tehrani (2016). The average

[2] `http://wals.info/`

[3] Grouping languages by distance is a very rough approximation. Ideally, the spatial neighbor graph should reflect geographic features such as mountains, rivers and oceans (Bouckaert et al., 2012).

number of spatial neighbors was 89.1.

We also constructed a weighted variant of the spatial neighbor graph such that spatially distant pairs of languages carried smaller weight. We set $1 - r/R$ to each edge where r was the distance between the neighboring languages.

Detailed phylogenetic trees were not provided by WALS, but each language is given two levels of groupings: family and genus.[4] We used genera to construct a phylogenetic neighbor graph in which every pair of languages within a genus was linked. The average number of linguistic neighbors was 30.8.

3 Background

3.1 Horizontal Transmission in Phylogenetic Inference

Although our ultimate goal is to infer past states of languages of the world, incorporating both vertical and horizontal transmission into a model is a notoriously difficult task due to excessive flexibility. Although the conventional tree model also requires a huge search space, it can be handled by computer-intensive statistical models (Felsenstein, 1981; Gray and Atkinson, 2003). Intuitively, the degree of uncertainty increases as we trace the states of languages back to the past, but at the same time, the tree model reduces the degree of freedom by repeatedly merging nodes into a parent. Horizontal transmission brings extra freedom that currently cannot be modeled without imposing some strong assumptions on it (Nelson-Sathi et al., 2010).

Most previous studies on phylogenetic inference employ the tree model even if they take horizontal transmission into account (Greenhill et al., 2010; Dediu, 2010; Dunn et al., 2011). Given typological features and a tree that is constructed by human experts or a lexicon-based phylogenetic model, they estimate the rate of change of each typological feature over time. We speculate that if a typological feature is prone to horizontal transmission, it is likely judged to be unstable by the exclusively vertical model. However, this cannot be confirmed in a straightforward manner.

Typologists have identified several geographical areas where the distribution of typological features suggests extensive horizontal transmission (Campbell, 2006). These areas are called linguistic areas, and features shared there are referred to as areal features. Daumé III (2009) incorporated linguistic areas into a phylogenetic tree and demonstrated that the use of linguistic areas improved phylogenetic reconstruction. In his Bayesian generative model, each feature of a language has a latent variable which determines whether it is derived from an areal cluster or the tree. As shown in Table 2 of his paper, summary statistics of this variable indicate how likely a feature is transmitted vertically or horizontally. Although linguistic areas are discussed in depth in linguistic typology, horizontal transmission does not necessarily result in areal clusters. For this reason, we make a weaker assumption as to the form of horizontal transmission. We use a single spatial neighbor graph of the world although not all pairs of languages are reachable in this graph.

3.2 Feature Stability Indices

Instead of reconstructing past states of languages, linguists often draw information from the current distribution of typological features. They claim that some features are more stable than others. In an extreme case, it is argued that some features reflect time depth of 10,000 years or more (Nichols, 1994).

With the notion of stability, some typologists try to quantitatively characterize typological features. They typically devise stability indices through series of deterministic arithmetic operations (Nichols, 1992; Nichols, 1995; Parkvall, 2008; Wichmann and Holman, 2009). This line of research was reviewed by Wichmann (2015) while Dediu and Cysouw (2013) conducted empirical comparison of several methods.

The intuition behind these methods is that if the same feature value is shared by a group of languages, defined vertically or horizontally, the feature in question must be stable. For a given feature, Nichols (1992) calculated the ratio of languages not taking the modal value in a given group and then computed

[4] Other resources such as Glottolog (Hammarström et al., 2016) and Ethnologue (Lewis et al., 2014) provide more detailed hierarchical classifications. However, there seems to be no way of selecting groups of languages at some specific level(s) of phylogenetic granularity.

the inter-group average. If groups are defined phylogenetically, the result implies vertical (in)stability and the same is true of areally defined groups. Parkvall (2008) also considered how often feature values were shared among a given group but designed a more complex formula to calculate *genealogical cohesiveness* C_{FAM} and *areal cohesiveness* C_{ARE}. Here per-group indices are simply averaged. The final stability index was defined as $S = C_{\mathrm{FAM}}/C_{\mathrm{ARE}}$. Wichmann and Holman (2009) presented a similar method but focused on vertical stability. Their stability index was adjusted for unrelated languages.

While we share the basic idea with these previous studies, our model is different from theirs. Instead of treating each group separately, we directly model the global distribution of a given feature with the hope that our model captures some universal tendencies. It is built upon a theoretical foundation of probability theory. Although computer intensive, the model itself is much simpler than series of arithmetic operations. Finally, we incorporate vertical and horizontal transmission into a single model. Since the two modes of transmission are directly contrasted, the outcome is more easily interpretable.

3.3 Autologistic Model for Cultural Traits

We find that a parallel can be drawn between anthropology and linguistics with respect to vertical and horizontal transmission. In anthropology, the two modes of transmission are known as phylogenesis and ethnogenesis, respectively (Collard and Shennan, 2000). Phylogenesis assumes the transmission of cultural traits primarily from ancestral to descendant populations while ethnogenesis assumes heavy influence from transmission between populations. Thus some models developed in the field of anthropology are applicable to linguistic data.

Towner et al. (2012) propose a variant of the autologistic model to contrast vertical and horizontal transmission. The autologistic model (Besag, 1974) is widely used to model the spatial distribution of a feature. It assumes that the value of a random variable depends probabilistically on the values of its neighbors. Towner et al. (2012)'s extension incorporates two neighbor graphs with associated weight parameters. Once these parameters are inferred from data, the relative strength of the dependency according to the graphs can be measured.

Towner et al. (2012) applied the autologistic model to cultural traits (features) of Western North American Indian societies, such as the presence or absence of agriculture, the tendency toward exogamy, and types of social structure. They constructed a spatial neighbor graph using longitude and latitude data of the societies while a phylogenetic neighbor graph was created by collapsing known language families. They empirically demonstrated that both transmission modes were non-negligible for the majority of traits. The same model was applied to folktales of Indo-European societies by da Silva and Tehrani (2016). At an intermediary step toward discovering folktales with deep historical roots, the model was used to filter out those with strong horizontal signals.

Our model is based on Towner et al. (2012)'s but differs mainly in three points. First, the latter only deals with binary features while we extend the model to handle categorical features. Although the original database of cultural traits was coded categorically, Towner et al. (2012) chose its small subset by dropping unsuitable features and by merging feature values. Second, the latter has four model variants that are compared using model selection criteria. Since the model comparison is performed for a fixed set of parameters, parameters are chosen by grid search. We omit model comparison for simplicity and directly optimize parameters with a gradient-based method. Third, since Towner et al. (2012) only use high-coverage features, they simply drop languages with missing values from the neighbor graphs. By contrast, the typological database is too sparse to ignore missing values. To cope with the inherent uncertainty, we resort to a sampling-based method.

3.4 Missing Value Imputation

The idea behind missing value imputation (MVI) of typological features in previous studies is that some features depends on others. Greenberg (1963) presents several rules that hold true for the world's languages, one of which is as follows: In languages with prepositions, the genitive almost always follows the governing noun, while in languages with postpositions it almost always precedes. In practical applications, we do not limit our scope to pairs of features but use a set of features to predict missing features.

At a preprocessing step, Murawaki (2015) used a variant of multiple correspondence analysis (Josse et al., 2012) for MVI. Takamura et al. (2016) chose a logistic model to investigate the predictive power of features. In their experiments, the discriminative classifier was given all but one feature of a given language and predicted the value of the remaining feature. They repeatedly selected one language for evaluation and trained the classifier using all other languages in the typological database.

Another idea, which we explore in this paper, is that phylogenetically or spatially close languages tend to share the same feature value. Note that proximity is implicitly utilized by the dependency-based approach because languages with similar feature combinations happen to be phylogenetically or spatially close ones. In fact, Takamura et al. (2016) conducted a type of ablation experiments in which they modified the training data (1) by removing languages sharing the same ancestor with the target language or (2) by excluding languages spatially close to the target. They demonstrated that accuracy dropped in either setting. This implies that vertical and horizontal clues are useful for MVI although they did not control for the tendency of smaller training data to decrease test accuracy.

In this paper, we exploit phylogenetic and areal proximity more directly for MVI. Note that the proximity-based approach is complementary to the dependency-based one. A combined model is expected to improve accuracy, but we leave it for future work.

4 Autologistic Model

4.1 Model

Like feature stability indices explained in Section 3.2, our model assumes that if the same feature value is shared by a group, the feature in question is stable. However, the collection of such groups is implicitly represented as a single neighbor graph. For each language, the model counts how many of its neighbors share the same feature value.

Our model incorporates both phylogenetic and spatial neighbor graphs. Given these graphs, our model predicts the distribution of each feature. If the same feature values are shared by many pairs of languages in the phylogenetic graph, it implies a strong vertical association, and the same holds for the spatial association.

Formally, let $\mathrm{x} = (x_1, x_2, \cdots, x_L)$ be the sequence of feature values, where x_i is the value of the i-th language and takes one of K categorical values (K differs according to feature types). Given a neighbor graph, the model checks if a language shares the feature value with its neighbors. For the unweighted spatial neighbor graph, let $S(\mathrm{x})$ be the number of pairs sharing the same value. For the weighted variant, the number of pairs is replaced with the sum of the edge weights. Analogously, let $T(\mathrm{x})$ be the number of pairs in the phylogenetic graph sharing the same value.[5] For each feature value k, $U_k(\mathrm{x})$ is the number of languages taking the value. Then the probability of x is given by

$$P(\mathrm{x}|\theta, \lambda, \beta) = \frac{\exp\left(\theta S(\mathrm{x}) + \lambda T(\mathrm{x}) + \sum_k \beta_k U_k(\mathrm{x})\right)}{\sum_{\mathrm{x}'} \exp\left(\theta S(\mathrm{x}') + \lambda T(\mathrm{x}') + \sum_k \beta_k U_k(\mathrm{x}')\right)}. \quad (1)$$

The denominator is the normalization term. β_k corresponds to the probability of taking the value k if the two neighbor graphs are not counted. θ and λ are parameters for spatial and phylogenetic associations, respectively. If θ (λ) is positive, spatial (phylogenetic) neighbors help predicting the value of the language in question. We estimate θ, λ, and β for each feature type while keeping the same spatial and phylogenetic neighbor graphs.

4.2 Inference

Given a feature distribution x, we want to infer θ, λ and β. If all languages are present, the objective function to maximize is $\log P(\mathrm{x}|\theta, \lambda, \beta)$. Unfortunately, the typological database is very sparse, and we have to deal with the inherent uncertainty. Suppose that x is decomposed into the observed portion $\mathrm{x}_{\mathrm{obs}}$ and the remaining missing portion $\mathrm{x}_{\mathrm{lat}}$. We use the notation $\mathrm{x}_{\mathrm{obs}} \oplus \mathrm{x}_{\mathrm{lat}}$ to recover x. Marginalizing $\mathrm{x}_{\mathrm{lat}}$,

[5] Up to this point, we placed vertical elements before horizontal ones. Hereafter we follow Towner et al. (2012) by reversing the order in the model description.

Model	Accuracy (%)	
	Macro	Micro
Global majority	55.30	54.22
Neighborhood	59.20	59.18
Proposed (Spatially unweighted)	61.98	61.97
Proposed (Spatially weighted)	61.84	61.82

Table 1: Results of missing value imputation.

we derive the modified log-likelihood function

$$L(\theta, \lambda, \beta; \mathrm{x}_{\mathrm{obs}}) = \log \sum_{\mathrm{x}'_{\mathrm{lat}}} P(\mathrm{x}_{\mathrm{obs}} \oplus \mathrm{x}'_{\mathrm{lat}} | \theta, \lambda, \beta). \tag{2}$$

We perform gradient-based training to maximize the objective. A simple gradient ascent algorithm would update λ as follows:

$$\lambda \leftarrow \lambda + \eta_t \frac{\partial L(\theta, \lambda, \beta; \mathrm{x}_{\mathrm{obs}})}{\partial \lambda}, \tag{3}$$

where η_t is a learning rate that decays according to time t. Instead of the simple gradient ascent algorithm, an adaptive extension of the optimization algorithm called Adam (Kingma and Ba, 2015) is used. θ and β are updated similarly.

The derivative of the log-likelihood function with respect to λ is

$$\frac{\partial L(\theta, \lambda, \beta; \mathrm{x}_{\mathrm{obs}})}{\partial \lambda} = \mathrm{E}_{\mathrm{x}'_{\mathrm{lat}} \sim P(\mathrm{x}'_{\mathrm{lat}} | \mathrm{x}_{\mathrm{obs}}, \theta, \lambda, \beta)} \left[S(\mathrm{x}_{\mathrm{obs}} \oplus \mathrm{x}'_{\mathrm{lat}}) \right] - \mathrm{E}_{\mathrm{x}' \sim P(\mathrm{x}' | \theta, \lambda, \beta)} \left[S(\mathrm{x}') \right]. \tag{4}$$

Both terms are computationally intractable due to the combinatorial explosion in the number of possible state x'. We approximate the expectation with samples. We collect samples of x' from the probability distribution and take the average of $S(\mathrm{x}')$ to estimate the expectation. We use Gibbs sampling to generate samples of x'. They are obtained by iteratively updating x'_i, one element of x' while keeping the remaining portion x'_{-i} fixed. The next value of x'_i is stochastically selected according to

$$P(x'_i = k | \mathrm{x}'_{-i}, \theta, \lambda, \beta) \propto \exp\left(\theta g_{i,k} + \lambda h_{i,k} + \beta_k\right), \tag{5}$$

where $g_{i,k}$ is the number of i's neighbors in the spatial neighbor graph taking the value k (or the sum of the edge weights for the weighted spatial neighbor graph). $h_{i,k}$ is defined analogously for the phylogenetic neighbor graph. For the first term of Eq. 4, we only update $\mathrm{x}'_{\mathrm{lat}}$ while keeping $\mathrm{x}_{\mathrm{obs}}$ unchanged. All elements are updated for the second term.

λ and θ are initialized with 0. β_k is set to the log-probability of taking the value k in $\mathrm{x}_{\mathrm{obs}}$. In this initial setting, Eq. 5 reduces to the empirical probability according to $\mathrm{x}_{\mathrm{obs}}$ and forces $\mathrm{x}_{\mathrm{lat}}$ to largely imitate $\mathrm{x}_{\mathrm{obs}}$.

5 Experimental Results

5.1 Missing Value Imputation

We indirectly evaluated the model performance with missing value imputation. If neighboring languages have some predictive power, our model must predict missing values better than chance. Although we have no ground truth for the missing portion of the original database, we can evaluate MVI by hiding some observed features and checking how well they were recovered. For each feature type, we conducted 10-fold cross validation.

We used the default settings for the hyperparameters of Adam (Kingma and Ba, 2015) (not to be confused with the parameters of the autologistic model): $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. We ran 100 iterations for parameter estimation. After that, we sampled $\mathrm{x}_{\mathrm{lat}}$ as follows. After 50 burn-in iterations, we collected 150 consecutive samples, one per iteration. For each language, we chose the most frequent feature value among the collected samples as the final output.

We compared the proposed model with two simple baselines.

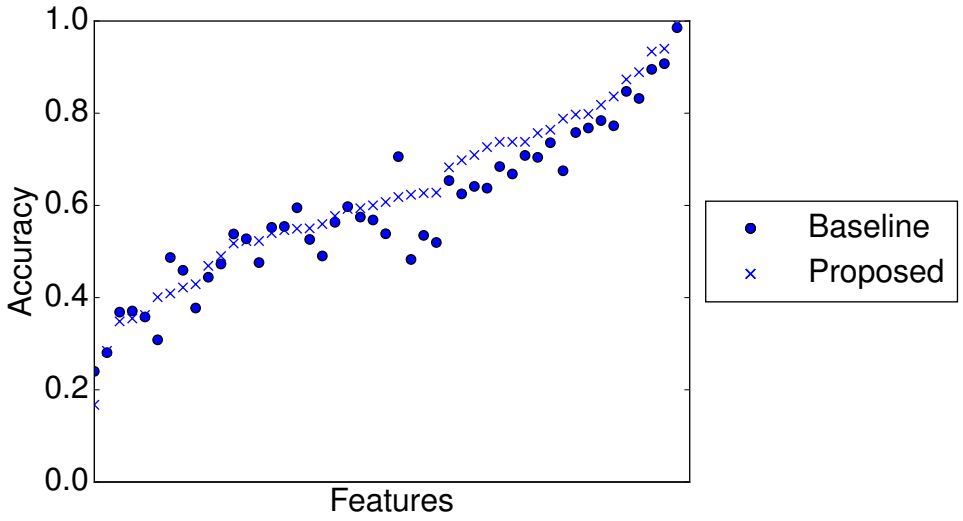


Figure 2: The MVI accuracy of the proposed model compared with that of the neighborhood baseline on a per-feature basis. Each cross denotes the accuracy of the proposed model with respect to a feature while the corresponding baseline accuracy is marked by a circle. Features are ordered in ascending order of the accuracy of the proposed model.

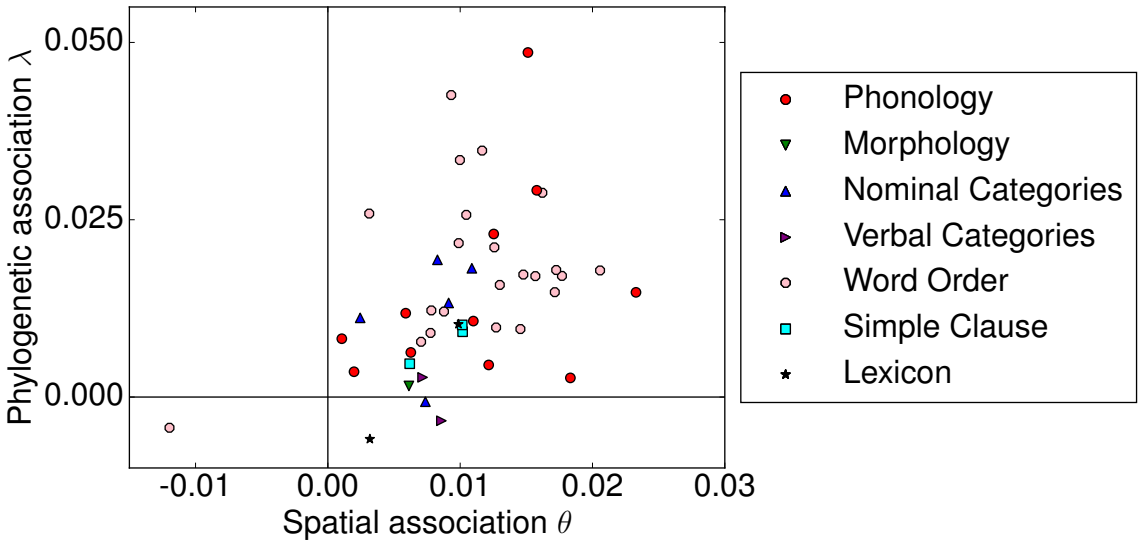


Figure 3: Estimates for phylogenetic (λ) and spatial (θ) associations. Each point denotes a feature. The shapes of the points represent broad categories of features (called *Area* in WALS).

	Feature type	Accuracy (%)
143G	Minor morphological means of signaling negation	99.31
11A	Front Rounded Vowels	93.99
90C	Postnominal relative clauses	93.39
130A	Finger and Hand	88.91
18A	Absence of Common Consonants	87.34
38A	Indefinite Articles	36.23
37A	Definite Articles	35.48
1A	Consonant Inventories	34.85
144A	Position of Negative Word With Respect to Subject, Object, and Verb	28.45
144L	The Position of Negative Morphemes in SOV Languages	16.75

(a) Features ranked by MVI accuracy of the proposed model.

	Feature type	Difference (%)
51A	Position of Case Affixes	+14.03
90A	Order of Relative Clause and Noun	+11.31
116A	Polar Questions	+10.81
3A	Consonant-Vowel Ratio	+9.25
69A	Position of Tense-Aspect Affixes	+9.17
26A	Prefixing vs. Suffixing in Inflectional Morphology	-3.72
144B	Position of negative words relative to beginning and end of clause and with respect to adjacency to verb	-4.55
144L	The Position of Negative Morphemes in SOV Languages	-7.26
4A	Voicing in Plosives and Fricatives	-7.79
129A	Hand and Arm	-8.75

(b) Features ordered by gain or loss. The proposed model is compared with the neighborhood baseline.

Table 2: Top 5 and bottom 5 features selected by two criteria.

Global majority Choose the most frequent value among $\mathrm{x}_{\mathrm{obs}}$ and always output the value.

Neighborhood For each language, collect neighbors in $\mathrm{x}_{\mathrm{obs}}$ and draw a value from the empirical distribution. If one of the two graphs has no observed neighbor, choose the other. If both are available, randomly choose one. If neither is available, draw from the empirical distribution according to the whole $\mathrm{x}_{\mathrm{obs}}$.

The proposed model itself had two variants: one with the unweighted spatial neighbor graph and the other with the weighted graph.

The results are shown in Tables 1 and 2(a). Our model outperformed the two baselines although the improvement was not so impressive as that of the dependency-based methods (Murawaki, 2015; Takamura et al., 2016). The edge weighting for the spatial neighbor graph had little effect on imputation performance.

Table 2(b) and Figure 2 compare the proposed model (spatially unweighted) with the neighborhood baseline in detail. As observed by Takamura et al. (2016), huge improvements were observed for Feature 81A "Order of Subject, Object and Verb" and some other word order related features. By contrast, the accuracy dropped sharply for Feature 144L "The Position of Negative Morphemes in SOV Languages," which might be explained by its mosaic-like distribution.

5.2 Parameter Estimation

In the next experiment, we used all observed features and estimated parameters θ, λ and β for each feature type. We chose the unweighted spatial neighbor graph for this experiment. We used the same hyperparameter settings for Adam and ran 100 iterations for parameter estimation.

	Feature type	θ
143G	Minor morphological means of signaling negation	0.0330
11A	Front Rounded Vowels	0.0233
144A	Position of Negative Word With Respect to Subject, Object, and Verb	0.0206
19A	Presence of Uncommon Consonants	0.0183
81A	Order of Subject, Object and Verb	0.0177
94A	Order of Adverbial Subordinator and Clause	0.0031
37A	Definite Articles	0.0024
8A	Lateral Consonants	0.0020
3A	Consonant-Vowel Ratio	0.0011
144L	The Position of Negative Morphemes in SOV Languages	-0.0120

(a) Features ranked by the spatial association θ.

	Feature type	λ
143G	Minor morphological means of signaling negation	0.0833
7A	Glottalized Consonants	0.0486
90A	Order of Relative Clause and Noun	0.0426
90C	Postnominal relative clauses	0.0347
87A	Order of Adjective and Noun	0.0334
26A	Prefixing vs. Suffixing in Inflectional Morphology	0.0016
38A	Indefinite Articles	-0.0007
69A	Position of Tense-Aspect Affixes	-0.0033
144L	The Position of Negative Morphemes in SOV Languages	-0.0043
129A	Hand and Arm	-0.0059

(b) Features ranked by the phylogenetic association λ.

Table 3: Features ranked by θ and λ. Top 5 and bottom 5 features are shown for each parameter.

Table 3 shows top- and bottom-ranked features for each parameter, and Figure 3 depicts the relations between the two parameters. 43 out of 47 features were in the top-right quadrant, meaning that they were both spatially and phylogenetically predictive. Nearly all word order-related features were given positive λ, which was consistent with previous findings (Daumé III, 2009). Wichmann and Holman (2009) also judged Feature 38A "Indefinite Articles" as "very unstable" although their result disagree with ours for Feature 69A "Position of Tense-Aspect Affixes" (stable) and Feature 129A "Hand and Arm" (stable).

In this study, we used genera to construct the phylogenetic neighbor graph, and thus λ can be seen as a genus-level global stability index. Although Feature 83A "Order of Object and Verb" is considered phylogenetically stable by the model, it is well known that the OV languages of India and predominantly VO languages of Europe constitute the Indo-European language family. Much work needs to be done to uncover deep phylogenetic relations. Despite the limitation, we argue that our present study presents a first necessary step toward typology-based phylogenetic inference.

6 Conclusion

In this paper, we quantitatively characterized features of linguistic typology with respect to vertical and horizontal transmission. We presented an autologistic model with which we can estimate the relative strength of transmission. Our model is simple and extensible, and the result are easy to interpret because the two modes of transmission are directly contrasted.

The high sparsity of the typological database remains a huge problem. Although our model recovers missing values better than simple baselines, the gain is not large. In the future, we would like to incorporate into the autologistic model a more effective imputation method that takes advantage of inter-feature dependencies.

Acknowledgments

We thank Mark N. Grote and Mary C. Towner for providing their code. Needless to say, any oversights and mistakes are ours. This work was partly supported by JSPS KAKENHI Grant Number 26730122.

References

Julian Besag. 1974. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 192–236.

Remco Bouckaert, Philippe Lemey, Michael Dunn, Simon J. Greenhill, Alexander V. Alekseyenko, Alexei J. Drummond, Russell D. Gray, Marc A. Suchard, and Quentin D. Atkinson. 2012. Mapping the origins and expansion of the Indo-European language family. *Science*, 337(6097):957–960.

Lyle Campbell. 2006. Areal linguistics. In *Encyclopedia of Language and Linguistics, Second Edition*, pages 454–460. Elsevier.

Mark Collard and Stephen J. Shennan. 2000. Ethnogenesis versus phylogenesis in prehistoric culture change: a case-study using European Neolithic pottery and biological phylogenetic techniques. In Colin Renfrew and Katherine V. Boyle, editors, *Archaeogenetics: DNA and the Population Prehistory of Europe*. McDonald Institute for Archaeological Research.

Sara Graça da Silva and Jamshid J Tehrani. 2016. Comparative phylogenetic analyses uncover the ancient roots of Indo-European folktales. *Royal Society Open Science*, 3(1).

Hal Daumé III and Lyle Campbell. 2007. A Bayesian model for discovering typological implications. In *ACL*, pages 65–72.

Hal Daumé III. 2009. Non-parametric Bayesian areal linguistics. In *HLT-NAACL*, pages 593–601.

Dan Dediu and Michael Cysouw. 2013. Some structural aspects of language are more stable than others: A comparison of seven methods. *PLoS ONE*, 8(1):1–20.

Dan Dediu. 2010. A Bayesian phylogenetic approach to estimating the stability of linguistic features and the genetic biasing of tone. *Proc. of the Royal Society of London B: Biological Sciences*, 278(1704):474–479.

Aharon Dolgopolsky. 1998. *The Nostratic Macrofamily and Linguistic Palaeontology*. The McDonald Institute for Archaeological Research.

Michael Dunn, Angela Terrill, Ger Reesink, Robert A. Foley, and Stephen C. Levinson. 2005. Structural phylogenetics and the reconstruction of ancient language history. *Science*, 309(5743):2072–2075.

Michael Dunn, Simon J. Greenhill, Stephen C. Levinson, and Russell D. Gray. 2011. Evolved structure of language shows lineage-specific trends in word-order universals. *Nature*, 473(7345):79–82.

Joseph Felsenstein. 1981. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17(6):368–376.

Russell D. Gray and Quentin D. Atkinson. 2003. Language-tree divergence times support the Anatolian theory of Indo-European origin. *Nature*, 426(6965):435–439.

Joseph H. Greenberg, editor. 1963. *Universals of language*. MIT Press.

Joseph H. Greenberg. 2000. *Indo-European and Its Closest Relatives: The Eurasiatic Language Family, Volume 1: Grammar*. Stanford: Stanford University Press.

Simon J. Greenhill, Quentin D. Atkinson, Andrew Meade, and Russel D. Gray. 2010. The shape and tempo of language evolution. *Proc. of the Royal Society B*, 277(1693):2443–2450.

Harald Hammarström, Robert Forkel, Martin Haspelmath, and Sebastian Bank, editors. 2016. *Glottolog 2.7*. Max Planck Institute for the Science of Human History.

Martin Haspelmath, Matthew Dryer, David Gil, and Bernard Comrie, editors. 2005. *The World Atlas of Language Structures*. Oxford University Press.

Julie Josse, Marie Chavent, Benot Liquet, and François Husson. 2012. Handling missing values with regularized iterative multiple correspondence analysis. *Journal of Classification*, 29(1):91–116.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations*.

M. Paul Lewis, Gary F. Simons, and Charles D. Fennig, editors. 2014. *Ethnologue: Languages of the World, 17th Edition*. SIL International. Online version: http://www.ethnologue.com.

Giuseppe Longobardi and Cristina Guardiano. 2009. Evidence for syntax as a signal of historical relatedness. *Lingua*, 119(11):1679–1706.

Yugo Murawaki. 2015. Continuous space representations of linguistic typology and their application to phylogenetic inference. In *Proc. of NAACL-HLT*, pages 324–334.

Shijulal Nelson-Sathi, Johann-Mattis List, Hans Geisler, Heiner Fangerau, Russell D. Gray, William Martin, and Tal Dagan. 2010. Networks uncover hidden lexical borrowing in Indo-European language evolution. *Proceedings of the Royal Society B*.

Johanna Nichols. 1992. *Linguistic Diversity in Space and Time*. University of Chicago Press.

Johanna Nichols. 1994. The apread of language around the Pacific rim. *Evolutionary Anthropology: Issues, News, and Reviews*, 3(6):206–215.

Johanna Nichols. 1995. Diachronically stable structural features. In Henning Andersen, editor, *Historical Linguistics 1993. Selected Papers from the 11th International Conference on Historical Linguistics, Los Angeles 16–20 August 1993*. John Benjamins Publishing Company.

Mark Pagel, Quentin D. Atkinson, Andreea S. Calude, and Andrew Meade. 2013. Ultraconserved words point to deep language ancestry across Eurasia. *Proc. of the National Academy of Sciences*, 110(21):8471–8476.

Mikael Parkvall. 2008. Which parts of language are the most stable? *STUF-Language Typology and Universals Sprachtypologie und Universalienforschung*, 61(3):234–250.

Thomas Pellard. 2009. *Ōgami—Éléments de description d'un parler du Sud des Ryūkyū*. Ph.D. thesis, Ecole des Hautes Etudes en Sciences Sociales (EHESS). (in French).

Morris Swadesh. 1971. *The Origin and Diversification of Language*. Aldine Atherton.

Hiroya Takamura, Ryo Nagata, and Yoshifumi Kawasaki. 2016. Discriminative analysis of linguistic features for typological study. In *Proc. of LREC*, pages 69–76.

Yee Whye Teh, Hal Daumé III, and Daniel Roy. 2008. Bayesian agglomerative clustering with coalescents. In *NIPS*, pages 1473–1480.

Mary C. Towner, Mark N. Grote, Jay Venti, and Monique Borgerhoff Mulder. 2012. Cultural macroevolution on neighbor graphs: Vertical and horizontal transmission among western north American Indian societies. *Human Nature*, 23(3):283–305.

Nikolai Sergeevich Trubetzkoy. 1928. Proposition 16. In *Acts of the First International Congress of Linguists*, pages 17–18.

Tasaku Tsunoda, Sumie Ueda, and Yoshiaki Itoh. 1995. Adpositions in word-order typology. *Linguistics*, 33(4):741–762.

Søren Wichmann and Eric W. Holman. 2009. *Temporal Stability of Linguistic Typological Features*. Lincom Europa.

Søren Wichmann. 2015. Diachronic stability and typology. In Claire Bowern and Bethwyn Evans, editors, *The Routledge Handbook of Historical Linguistics*, pages 212–224. Routledge.

How Regular is Japanese Loanword Adaptation? A Computational Study

Lingshuang Jack Mao and **Mans Hulden**
Department of Linguistics
University of Colorado
{lima4664,mans.hulden}@colorado.edu

Abstract

The modifications that foreign loanwords undergo when adapted into Japanese have been the subject of much study in linguistics. The scholarly interest of the topic can be attributed to the fact that Japanese loanwords undergo a complex series of phonological adaptations, something which has been puzzling scholars for decades. While previous studies of Japanese loanword accommodation have focused on specific phonological phenomena of limited scope, the current study leverages computational methods to provide a more complete description of all the sound changes that occur when adopting English words into Japanese. To investigate this, we have developed a parallel corpus of 250 English transcriptions and their respective Japanese equivalents. These words were then used to develop a wide-coverage finite state transducer based phonological grammar that mimics the behavior of the Japanese adaptation process, mapping e.g **cream** [kɹiːm] to [kɯ.ɾiː.mɯ]. By developing rules with the goal of accounting completely for a large number of borrowings, and analyzing forms mistakenly generated by the system, we discover an internal inconsistency within the loanword phonology of the Japanese language, something arguably underestimated by previous studies. The result of the investigation suggests that there are multiple **dimensions** that shape the output form of the current Japanese loanwords. These dimensions include orthography, phonetics, and historical changes.

1 Introduction

Borrowing lexical items from one language to another is a common linguistic phenomenon. For example, someone can wear a **beret** /bəˈreɪ/ while enjoying **sushi** /ˈsuʃi/ next to a pet **alpaca** /ælˈpækə/. Loanword adaptation refers to the process of how foreign sounds or phonological structures are made to conform to the sounds and structures of the recipient language (Goldsmith et al., 2011). Loanword phonology, as the study of the systematicity of such adaptation, attracts scholarly attention because it arguably 'pushes the limit' of the phonological system of the borrowing language in order to preserve the phonetic quality and phonological structure of the source language. This study aims to investigate how 'systematic' or 'unsystematic' a loanword phonology can be. We choose Japanese as our specific research context.

Loanwords in Japanese are ubiquitous. According to Tsunoda (1988), loanwords account for 10 to 25 percent of the lexicon in nationally circulated, news-oriented weeklies, and up to 70 percent in professional journals, medicine and science in particular. However, any attempt to provide a general account for the Japanese loanword adaptation mechanism is an ambitious enterprise because the loanword lexicon contains many phonological exceptions whose formation processes have become opaque. Furthermore, most earlier works in Japanese loanword phonology focus only on particulars or a handful of specific arguments revolving around a limited number of constraints, or assume that a set of rules or constraints discussed have covered all the phenomena; few works have actually given a complete overview of the process. Therefore, a need for a fuller description of the loanword system has motivated this study.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 847–856, Osaka, Japan, December 11-17 2016.

2 Related work

2.1 Lexical stratification of the Japanese language

It is well-known that the Japanese lexicon is stratified along the lines of etymology, and Itō and Mester (1993) formalizes such a stratification on a phonological level. That work claims that the Japanese lexicon can be categorized into the strata *Yamato* (native), *Sino-Japanese* (lexicon historically imported from China), *Mimetic* (various onomatopoeic expressions), and *Foreign* (modern loanwords, usually from western languages), based on how prototypical or peripheral the words are with respect to the phonological constraints of the language. The main argument is that each stratum has a different degree of strictness in applying certain phonological constraints. The following example illustrates this claim further with three well-known constraints on Japanese words:

1. ***P**: /p/ is only tolerated in geminates or partial geminates
2. ***NT**: post-nasal obstruents must be voiced
3. **CODACOND**: coda consonants may not have a place feature

The following table adapted from Itō and Mester (1993) shows how each constraint applies to each lexical stratum:

Lexical stratum	***P**	***NT**	**CODACOND**
Yamato	Yes	Yes	Yes
Sino-Japanese	Yes	No	Yes
Mimetic	No	Yes	Yes
Foreign	No	No	No

Table 1: Coverage of three constraints for each lexical stratum in stratified models of the Japanese lexicon.

In Table 1 we can see that in the *Yamato* stratum, all three constraints are all expected to be satisfied; in *Sino-Japanese* and *Mimetic*, only two are mandatory; while in *Foreign*, none are mandatory. Thus, Itō and Mester (1993) hypothesize that the *Yamato* stratum forms the core of the Japanese lexicon, because it violates the fewest constraints, while the *Foreign* stratum forms the outermost periphery of the lexicon because it violates the largest number of these constraints. Figure 1 shows a visual representation of the phonological stratification of Japanese lexicon.

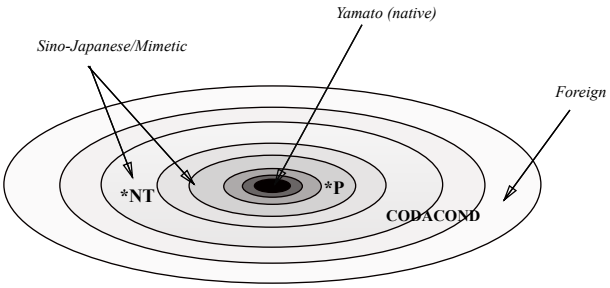


Figure 1: The core-periphery structure, adapted from Itō and Mester (1993)

This model of stratification of the Japanese lexicon has informed us that when approaching the periphery from the core, more and more forms are considered 'grammatical' because constraints becoming weaker and weaker until abolished altogether. Since the *Foreign* stratum sits on the periphery, we can assume that the loanword phonology actually **contains** the core phonology. Therefore, studying the loanword phonology matters to linguists who wish to study essential characteristics of the entire phonological system of a language and the tension between the native and foreign lexical strata.

2.2 Generative Phonology vs. Optimality Theory

As in most areas of phonology, the majority of previous work on Japanese loanword phonology has focused on analyzing its various aspects in the framework of rule-based Generative Phonology (Chomsky and Halle, 1968) and/or the newer and more fashionable constraint-based framework of Optimality Theory (OT) (Prince and Smolensky, 1993). Generative phonology regards a grammar as a set of phonological calculations (i.e. rules) that map an *underlying representation* (UR) into a *surface representation* (SR). OT on the other hand does not model the grammar of a language as a sequential algorithm; rather, a grammar is a set of defined structural limitations to be obeyed, each with a certain ranking of importance, known as *constraints*, and these constrains also interact with each other through their ranking. From a set of possible candidates of a certain form, the grammar chooses the candidate with the fewest violations of the constraints (i.e the 'optimal' one). Earlier works on Japanese loanwords are mainly based on generative phonology where the source language is the UR and the adapted Japanese loanword is the SR; with OT, a similar setup is assumed, but the SR is dictated by interaction of constraints.

2.3 Previous work and current study

Works on Japanese loanwords are substantial (e.g. Ichikawa (1929), Lovins (1975), Shirai (1999), Kubozono et al. (2013)) As a representative of earlier work in this field, Lovins (1975), in her dissertation *Loanwords and the phonological structure of Japanese*, takes a rule-based approach to investigate the loanword adaptation mechanisms from Western languages to Japanese. Her analysis is very detailed and the coverage of phenomena is also quite extensive, but, because the analysis and statistics are done manually, the whole endeavor is inevitably cumbersome. Some exceptions and corner cases are not clearly stated and explained, and the relations between loanwords and the rest of the Japanese language remains obscure. Such an effort of attempting to account for various linguistic forms has extended into recent work. For instance, Shoji and Shoji (2014) have proposed an OT account for vowel epenthesis and consonant deletion in Japanese loanwords adapted from English. However, such a framework predicts numerous Japanese outputs for an English word.

From those studies, not only did we see a limited scope of the investigation, but also a general underlying assumption that there exists a phonological system such that it can accurately transform the source language input into the borrowing language outputs, and that such a phonological system can potentially account for all the exceptions encountered in previous studies. In order to test the validity of this assumption, we decided to conduct a corpus-based computational study on Japanese loanword phonology. We decided to adopt a rewrite rule approach instead of OT because, as Karttunen (2005) points out, OT grammars are very difficult and tedious to debug; on the other hand, grammars based on rewrite rules built using finite-state transducer (FST) techniques are straightforward to construct and easily debugged.

What is different between our approach and previous ones is that we do not test or intuit the result of the rule compilation manually; rather, we utilize a more rigorous method: composing finite-state transducers to model phonological rules for mimicking the loanword adaptation processes, and automatically calculating coverage and adjusting the rules to maximize coverage. The benefit of the computational approach is obvious: we can immediately obtain the accuracy of the system output during each round of the rule development and modification process, which is nearly impossible when resorting to paper-and-pencil approaches.

3 Japanese loanword corpus compilation

We manually compiled a word list of 250 tokens of Japanese loanwords from a Japanese-Multilingual Dictionary—essentially a multilingual lexical database with Japanese as the pivot language (Breen, 2004). The word list contains parallel phonetic transcriptions of both English (source) and Japanese (borrowing) entries, and we designed it carefully in the following ways:

1. The word list includes all the phonemes in English and Japanese.
2. The word list reflects as many different phonological interactions as possible collected from previous studies.

3. The word list intentionally includes 'peculiar instances' discussed in previous studies (e.g. /pɪk.nɪk/ ↦ [pi.kɯ.nik.kɯ] in which the word-final /k/ is geminated while the word-medial /k/ is not.) (Kubozono et al., 2008).
4. We only included words with an English origin although some of them were often mistakenly attributed to other languages by some scholars. For example, Energie /ɛnɛʁgi/ ↦ [e.ne.rɯ.giː] (German) vs. energy /ɛnəʤi/ ↦ [e.na.jiː] (British English).

For majority of the words, we used standard British English transcription as the input instead of American English. The differences between the two varieties are largely irrelevant for the purpose of this study. But assuming British English to be the source is slightly preferred because it lacks the postvocalic /ɹ/ which is easier to adapt to Japanese as the language does not permit consonant codas except for the moraic nasal /N/ (e.g. /gɪtɑ(ɹ)/ ↦ [gitaː], and /fɔ(ɹ)k/ ↦ [foːkɯ]). However, some words were obviously borrowed from American English, and in such cases, American English transcription was used. For example, the Japanese SR for 'schedule' is [sɯ.ke.jɯɯ.rɯ] which is obviously derived from the American variant /skɛʤʊl/ as opposed to the British /ʃɛdjuːl/. In addition, the phonetic transcription is broad enough to reflect phonological processes rather than phonetic details. For instance, the Japanese adaptation of egg /ɛg/ was transcribed as [eg.gɯ] although voiced geminates are often devoiced in actual speech.

Some Japanese loanwords have two forms, as shown in the study of truncation/epenthesis loan doublets by Arakawa (1977):

	Truncation form	**Epenthesis form**	**English**
(1)	[pok.ke]	[po.ket.to]	pocket /pɒkɪt/
	[pɯ.ɾiN]	[pɯ.diŋ.gɯ]	pudding /pʊdɪŋ/
	[ɾa.mɯ.ne]	[ɾe.mo.neː.do]	lemonade /lɛməneɪd/

In these cases, only the epenthesis form was selected to be included in the corpus for the following reasons. First, the truncation form largely depends on the phonetics and phonology of the source language instead of the borrowing language. For example, because English phonology permits unreleased coda stops, such as /t/ in the source word pocket /pɒkɪt/, it provides a base for the Japanese loanword system to adapt the input also without the final consonant [t], namely [pok.ke]. Second, the epenthesis form usually keeps all the original consonants (or their adapted version) in the output, which contains more complete linguistic information. Third, the truncation form is not productive in the Japanese loanword system, and many loanwords do not have such forms. For example, gadget /gæʤɪt/ can only be adapted as [ga.jet.to] even though the English input ends with a /t/ that need not to be released.

We primarily relied on previous literature to decide which words to include in the corpus and how many instances to include for each process. Because we focused on compiling a computational grammar by composition of finite-state transducers, we believe that our corpus comprehensively reflects the Japanese loanword adaptation patterns gathered from previous literature. For a single segment, either a vowel or a consonant, previous literature and observations suggest that the adaptation is mostly predictable, so a few examples of each segment was considered sufficient. For instance, /θ/ ↦ [s] and /ð/ ↦ [z]. When dealing with unpredictable cases where a single input can be adapted into multiple output forms, more examples were included to ensure the generality of the rules. For example, /i/ is adapted as [iː] in comedy /kɒmɪdi/ ↦ [ko.me.d**iː**]; however, /i/ is adapted as [i] in celery /sɛləri/ ↦ [se.ro.r**i**]. In such cases, we included numerous examples of both adaptations and chose the rule that gave the better overall coverage. For more complex phonological phenomena such as gemination and vowel epenthesis, generally a large sample was included to ensure the coverage of all the phenomena described by previous literature.

In total, we discovered that the majority of the adaptation phenomena could be accounted for by a relatively small number of rules, the number of tokens included in the corpus was deemed sufficient for the purpose of this study.

4 Implementation

4.1 Finite-state phonology

The loanword adaptation grammar is implemented as a sequence of phonological rewrite rules in the Xerox finite-state calculus (Beesley and Karttunen, 2003). This is the standard mode of implementing complex hand-written phonological grammars computationally, capturing phenomena such as phonological alternations (Karttunen and Beesley, 2005), syllabification processes (Hulden, 2006), optimality-theoretic constraints (Karttunen, 1998), and nonconcatenative morphophonological processes (Beesley, 1998). The system relies on the ability to convert phonological replacement (or 'rewrite') rules into individual finite state transducers. A collection of such transducers can then be composed in a certain order, yielding one monolithic transducer, the end result essentially replicating the effect of applying multiple phonological rules in a sequence. This grammar transducer can also be applied in the inverse direction, mapping from a loanword phonological form to its possible sources. Although we don't explore this possibility here, the invertibility of a transducer is useful for debugging our rules. While finite-state tools have also been used to model other prominent approaches to describing loanword adaptation such as OT grammars (Karttunen, 1998; Gerdemann and van Noord, 2000; Gerdemann and Hulden, 2012), we restrict ourselves to the rewrite-rule model in this study.

The grammar itself is developed using the *foma*-toolkit (Hulden, 2009). A summary of the core formalism is given in Table 2. Although the formalism offers a vast number of operations, we essentially only make use of a standard context-conditioned rewrite rule, which has the following appearance

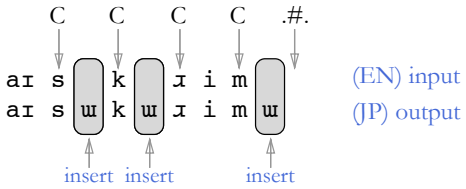
(2) `LHS -> RHS || LC _ RC`

which essentially reads as: replace all occurrences of the pattern `LHS` with the pattern `RHS`, whenever it occurs between the patterns `LC` and `RC`. The *patterns* in question can be expressed as *regular expressions*. Unlike the rewrite rules found in the phonological literature (Hayes, 2011), such a replacement rule is taken to operate simultaneously (and not iteratively in any directional manner) replacing all occurrences of `LHS` whenever warranted by the rule. For example, a rough approximation of accommodating the Japanese prohibition against complex consonant clusters, solved by epenthesis of [ɯ] between two adjacent consonants, or between a consonant and end-of-word, could be expressed as:

(3) `[..] -> ɯ || C _ [C|.#.]`

In effect, this says we insert a high unrounded back vowel [ɯ], whenever we have a consonant on the left-hand side and a consonant or end-of-word on the right. The rule itself compiles into an FST that gives rise to mappings such as the one below, translating the English phonetic representation of **ice cream** into a Japanese one (disregarding other relevant changes that also occur).

(4)



Note that this rule is simplified for the sake of exposition, and we will later provide a more accurate rule for the same phenomenon.

4.2 Developing and debugging rewrite rules

During development, we automatically checked the accuracy of our rules against the corpus, which contained not only the source forms (English) but also the attested output forms, i.e. the Japanese equivalents. This permitted us to gauge the coverage of each rule separately and iteratively address shortcomings of the adaptation grammar. Below is an example of three of our rules (here called R1, R2, and R3) that are active in transforming the English input **Christmas** /krɪsməs/ into the corresponding Japanese SR [kɯ.ri.sɯ.ma.sɯ].

`AB`	Concatenation
`A|B`	Union
`A & B`	Intersection
`A*`	Kleene Star
`~A`	Complement
`?`	Any symbol in alphabet
`0`	The empty string (epsilon)
`[` and `]`	Grouping brackets
`A -> B`	Change A to B
`[..] -> B`	Epenthesize B
`|| C _ D`	Context specifier
`.#.`	End or beginning of string
`def X ...`	Label a rewrite rule/regular expression

Table 2: An overview of the relevant regular expression and rewrite rule notation in *foma*.

R1: Epenthesize the vowel [ɯ][1] between two consonants or at the end of a word. The first consonant cannot be /n/, /t/, /d/, /tt/, or /dd/.

- `def insertu [..] -> u || C & ~[n|t|d|tt|dd] _ [C|.#.];`

R2: English input /ɪ/ becomes Japanese output /i/ in any phonological context.

- `def ichange ɪ -> i;`

R3: English input /ə/ becomes Japanese output /a/ in any phonological context.

- `def schwa ə -> a;`

Although they work, the context-free rule (R3) is too general. For example, for the input word, **violin** /vaɪəlɪn/ ↦ [bai.o.rin], it is necessary to apply another rule:

R4: English input /ə/ becomes Japanese output /o/ in any phonological context.

- `def schwa ə -> o;`

In order to reconcile the conflict between R3 and R4, new information has to be coded to differentiate versions of the input /ə/. We observed that the adaptation of /ə/ heavily depends on the original *orthographic* representation. Therefore, we coded each input /ə/ with the written orthographic symbol it originated from. For instance, 'əE' means that the input segment is /ə/ and it is represented as the letter **e** orthographically. After incorporating orthographic shape into the English transcription, we collapsed R3 and R4 into a single rule:

R5: The rule now can adapt /ə/ into different Japanese output forms.

- `def schwa əE -> e, əA -> a, əO -> o;`

Since the finite-state calculus allows us to view the input and output (domain and range) of a transducer as an automaton, we can compose our grammar with identity transducers that represent the source and target forms, respectively. This allows for quick identification of which rules fail and which overgenerate, which is helpful for rapid debugging and deployment—a standard technique in finite-state development (Beesley and Karttunen, 2003).

(5) `Sourcewords .o. Grammar .o. Targetwords`

After repeatedly refining our rules we arrive at the final Japanese loanword adaptation grammar.

5 Results

5.1 Phonological rules for adaptations from English to Japanese

By manual development of 34 phonological rules (3 insertion + 1 deletion + 30 alternation rules), the entire grammar correctly maps 215 out of 250 (86.0%) input forms to their Japanese counterparts.

[1]For convenience, we use [u] to represent [ɯ]—as there is no high back round vowel in Japanese, this cannot cause a conflict.

5.2 Error types

Before any rule is formally introduced into the system, we always test if the rule in question increases the overall coverage of the grammar. Although the final phonological system achieves high coverage for modeling loanword adaptation from English to Japanese, it is linguistically informative to analyze what kinds of mistakes the system makes. The mistakes arguably represent not only the shortcomings of this specific system, but also the characteristics of the Japanese loanword phonology as a whole.

Specifically, the Japanese SR forms which are mistakenly generated by the loanword adaptation grammar can be categorized into 3 types. Each type is presented in Table 3 with examples taken from the corpus for illustration.

English Input		System Output	Correct SR	Error Type	Percentage among mistaken outputs
diamond	/daɪəmənd/	dai.a.mon.do	dai.**y**a.mon.do		
steak	/steɪk/	sɯ.teː.kɯ	sɯ.teː.k**i**	Remainder of historical forms	28.6%
sport	/spɔːt/	sɯ.poː.to	sɯ.poː.**ts**ɯ		
gasoline	/gæsəlin/	gya.so.rin	**ga**.so.rin		
volunteer	/vɒlənˈtɪə/	bo.ran.chi.a	bo.ran.**t**i.a	Phonotactics relaxed	22.9%
teenager	/tineɪʤɜː/	chi.neː.jaː	**t**iː.neː.jaː		
bolt	/bəʊlt/	boː.rɯ.to	**bo**.rɯ.to		
tough	/tʌf/	taf.fɯ	ta.**f**ɯ	Unexpected segmental lengths	48.5%
sausage	/sɒsəʤ/	so.sej.ji	**soː.seː.**ji		

Table 3: Error types of the system output

From these mistaken forms, various types of linguistically useful information can be gleaned. First, historical forms, which comply more strictly with the native phonology of Japanese, and new adapted forms, which are more likely to be phonologically relaxed, exist simultaneously in the Japanese loanword lexicon. For example, the newer loanword **bike** /baɪk/ is adapted as [bai.k**ɯ**], while the older loanword **strike** /straɪk/ becomes [sɯ.to.rai.k**i]** due to a vowel harmony system in archaic Japanese (Shoji and Shoji, 2014). Secondly, some phonological processes, such as gemination, are unpredictable. It is difficult to explain why **bag** /bæg/ is adapted as /ba**g.g**ɯ/ but **bug** /bʌg/ becomes /ba.**gɯ**/ which forbids the gemination of the word-final voiced stop [g]. Therefore, the only way to model this type of ambiguous adaptation is to use segment-level statistics to help determine which adaptation has the highest probability to be the correct output form. This type of error also suggests that there might not be a single consistent way to generalize the loanword system of the Japanese language because the explanation for many corner cases and exceptions has been lost in history.

5.3 Rule modifications

According to these error types, we considered whether it was possible to modify the rules to eliminate these errors. Such a question can help us get closer to the ultimate goal: to construct a definitive phonological system which accurately produces adapted output forms. However, we noticed that when one rule is modified, it usually negatively affects the overall accuracy of the system output, because, although it 'fixes' a few current erroneous outputs, it also ruins previously correct forms. Table 4 shows how the number of mistaken outputs change when the current rules are modified according to various parameters.

Original number of erroneous forms	Parameter of modification	Current number of erroneous forms
35	Always inserting [y] between [i] and any vowel	35
35	No voiced geminates allowed	38
35	No velar stops allowed to be palatalized	38
35	No [t] allowed to be palatalized	38
35	First vowel of a trisyllabic word not allowed to be long	40
35	First vowel of any word not allowed to be long	71
35	No orthographic information coded into the lexicon	72-76 (depends one which uniform output is chosen, e.g. /ə/ ↦ [a], [o], or [e])
35	Only epenthesizing the default vowel [ɯ]	80

Table 4: Changes before and after rule modifications

As we see, none of the modifications given in Table 4 is really satisfactory since each of them increases the number of overall erroneous output forms of the system. This indicates that the essential 34 rules discussed in §5.1 have covered most of core phenomena in Japanese loanword phonology, and the remaining erroneous output forms are indeed exceptions outside of the set the core rules. This also means that there is clear limit to the predictability of the phonological system that drives the accommodation of loanwords in Japanese. In other words, after a certain point—corresponding to roughly 90% per type—there is a residue of internal inconsistency which is not capturable by any satisfactory phonological generalization.

6 Discussion

Because of such inconsistency within the system, if any analysis attempts to achieve a comprehensive account for Japanese loanword phonology, the model must involve external **dimensions** other than the one-to-one correspondence between the source tokens and borrowing language output. **Dimension** here refers to any domain which most significantly contributes to the nativization process of a certain group of loanwords. Based on the analysis, we incorporate four dimensions into our phonological system:

1. **Phonetic dimension**: phonetic characteristics of the source language heavily influences the form of output
   - Example: *pocket* /pɒkɪt/ ↦ [pok.ke] because English permits an unreleased /t/ at the end of the word, which is difficult for non-native speakers to perceive. As the result, Japanese adaptation also overlooks the final /t/ phoneme.
2. **Phonemic dimension**: The adapted form corresponds to all or almost all segments in the source language.
   - Examples: *pocket* /pɒkɪt/ ↦ [pok.ket.to] because all the segments in the source form are transformed into corresponding segments in the adapted form. In other words, the number of segments in the adapted form is no smaller than that in the source form.
3. **Orthographic dimension**: the orthography of the segment in the source language clearly determines or heavily influences the adaptation choice.
   - Example: *waffle* /wɒfəl/ ↦ [wa**f.f**ɯ.rɯ], where the gemination of [f] is clearly motivated by the orthography because the phonological environment where it occurs does not usually motivate gemination in Japanese. However, not every geminate reflects its original orthography. Compare *bottle* /bɒtəl/ ↦ [bo.**t**o.rɯ].
4. **Historical dimension**: For historical reasons, older forms survived diachronic changes and have become the only accepted adaptation in the current loanword system.
   - Example: The accepted Japanese loanword for the input *sport* /spɔːt/ is [sɯ.poː.**tsɯ**] with the affricate [ts] as the output instead of the newer adaptation mapping /t/ ↦ [t].

Notice that the orthographic and historical dimensions are not on the same hierarchical level as the phonetic and phonemic dimensions; rather, they exist as submembers of the phonemic dimension (see Figure 2 (a) and (b) for two examples). This hierarchical model successfully captures the fact that orthography and historical factors are only relevant when the phonemic dimension is chosen first.

7 Conclusion

Most previous studies on Japanese loanword phonology or on any phonology in general tend to take a 'paper-and-pencil' approach. Although doing phonology manually in fieldwork is considered traditional and long-standing, computational tools can arguably assist scholars to understand a new phonological system much more efficiently. In this study, we have investigated the phonological system of the Japanese loanwords using the finite-state tool, *foma*. By observing a carefully-compiled corpus and manually developing rewrite rules in *foma*, we obtained a holistic 'phonological grammar' of the adaptation process of Japanese. By analyzing the types of residual errors to which no good solution exists, modifying rules according to different linguistic parameters, we have discovered an internal inconsistency of the phonological system itself. This reveals linguistically important information about the phonological system:

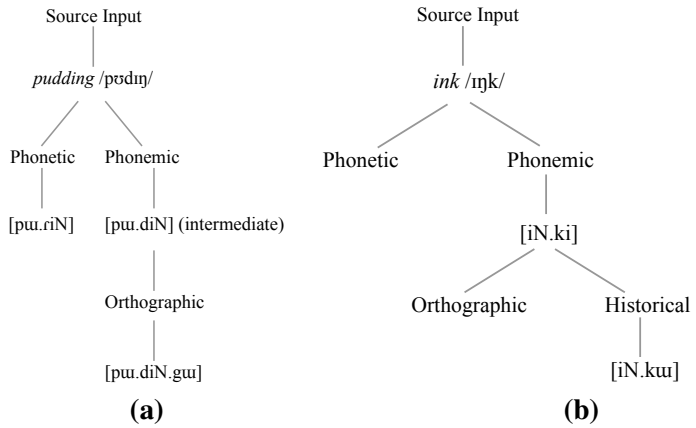


Figure 2: The hierarchical structure of phonetic, phonemic, and orthographic dimensions (a), and (b), the incorporation of a historical dimension into the hierarchical structure of dimensions.

multiple dimensions exist simultaneous within the Japanese loanword phonology that can represented in a tree-like hierarchical structure, and each foreign word has been adapted into Japanese through one specific dimension. All these discoveries would have been extremely difficult, if not impossible, to make without the computational tools available.

References

Sōbē Arakawa. 1977. *Gairaigo ziten*. Kadokawa, Tokyo, 2nd edition.

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI Publications, Stanford, CA.

Kenneth R. Beesley. 1998. Arabic morphology using only finite-state operations. In *Proceedings of the Workshop on Computational Approaches to Semitic languages*, pages 50–57. Association for Computational Linguistics.

James Breen. 2004. JMDict: a Japanese-multilingual dictionary. In *Proceedings of the Workshop on Multilingual Linguistic Ressources*, pages 71–79. Association for Computational Linguistics.

Noam Chomsky and Morris Halle. 1968. *The sound pattern of English.* Harper & Row.

Dale Gerdemann and Mans Hulden. 2012. Practical finite state optimality theory. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, pages 10–19, Donostia–San Sebastián, July. Association for Computational Linguistics.

Dale Gerdemann and Gertjan van Noord. 2000. Approximation and exactness in finite state optimality theory. In *Proceedings of the Fifth Workshop of the ACL Special Interest Group in Computational Phonology*.

John A. Goldsmith, Jason Riggle, and Alan C. L. Yu, editors. 2011. *The handbook of phonological theory*, volume 75. John Wiley and Sons, New York.

Bruce Hayes. 2011. *Introductory Phonology*. John Wiley & Sons.

Mans Hulden. 2006. Finite-state syllabification. *Lecture Notes in Artificial Intelligence*, 4002:86–96.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In *Proceedings of the 12th conference of the European Chapter of the Association for Computational Linguistics,*, pages 29–32.

Sanki Ichikawa. 1929. *Foreign Influences on the Japanese Language*. Japanese Council Institute of Pacific Relations, Tokyo.

Junko Itō and Armin Mester. 1993. Japanese phonology: constraint domains and structure preservation. *The handbook of phonological theory*, pages 871–838.

Lauri Karttunen and Kenneth R. Beesley. 2005. Twenty-five years of finite-state morphology. *Inquiries Into Words, a Festschrift for Kimmo Koskenniemi on his 60th Birthday*, pages 71–83.

Lauri Karttunen. 1998. The proper treatment of optimality theory in computational phonology. In *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing (FSMNLP)*.

Lauri Karttunen. 2005. Finnish optimality-theoretic prosody. In *International Workshop on Finite-State Methods and Natural Language Processing*, pages 9–10. Springer.

Haruo Kubozono, Junko Ito, and Armin Mester. 2008. Consonant gemination in Japanese loanword phonology. In *18th International Congress of Linguistics, Seoul*.

Haruo Kubozono, Hajime Takeyasu, and Mikio Giriko. 2013. On the positional asymmetry of consonant gemination in Japanese loanwords. *Journal of East Asian Linguistics*, 22(4):339–371.

Julie B. Lovins. 1975. Loanwords and the phonological structure of Japanese. *Indiana University Linguistics Club*.

Alan Prince and Paul Smolensky. 1993. Optimality theory: Constraint interaction in generative grammar. *Ms. Rutgers University Cognitive Science Center*.

Setsuko Shirai. 1999. Gemination in loans from English to Japanese. *Master's thesis*.

Shinichi Shoji and Kazuko Shoji. 2014. Vowel epenthesis and consonant deletion in Japanese loanwords from English. *Proceedings of the Annual Meetings on Phonology*, 1(1).

Waka Tsunoda. 1988. The influx of English in Japanese language and literature. *World Literature Today*, 62(3):425–430.

Using Linguistic Data for English and Spanish Verb-Noun Combination Identification

Uxoa Iñurrieta, Arantza Díaz de Ilarraza, Gorka Labaka, Kepa Sarasola
IXA NLP group, University of the Basque Country
usoa.inurrieta|a.diazdeilarraza|gorka.labaka|kepa.sarasola@ehu.eus

Itziar Aduriz
Department of Linguistics, University of Barcelona
itziar.aduriz@ub.edu

John Carroll
Department of Informatics, University of Sussex
j.a.carroll@sussex.ac.uk

Abstract

We present a linguistic analysis of a set of English and Spanish verb+noun combinations (VNCs), and a method to use this information to improve VNC identification. Firstly, a sample of frequent VNCs are analysed in-depth and tagged along lexico-semantic and morphosyntactic dimensions, obtaining satisfactory inter-annotator agreement scores. Then, a VNC identification experiment is undertaken, where the analysed linguistic data is combined with chunking information and syntactic dependencies. A comparison between the results of the experiment and the results obtained by a basic detection method shows that VNC identification can be greatly improved by using linguistic information, as a large number of additional occurrences are detected with high precision.

1 Introduction

Multiword Expressions (MWEs) are recurrent combinations of two or more words expressing a single unit of meaning, this meaning not always derivable directly from the meanings of the component words (Sag et al., 2002). Therefore, Natural Language Processing (NLP) tasks that need to be sensitive to lexical meaning should treat MWEs as single units. However, this is a challenging problem since many MWEs can have multimple morphosyntactic variants, which makes them difficult to recognise or generate. Examples (1)-(3) below contain *take steps*; correct translation of this MWE into another language, for instance, requires it to be recognised as a single unit[1].

(1) The Government will *take* all the necessary *steps* to prepare.

(2) They set out five important *steps* the Minister needs to *take*.

(3) What were the *steps* that should have been *taken*?

Although the most straightforward method for recognising MWEs is to attempt to match word sequences against entries in a lexicon, this method does not work for combinations that can have multiple variants. This is often the situation for verb+noun combinations (VNCs), since this kind of MWE is usually morphosyntactically flexible.

In the case of Machine Translation (MT), there are two challenges that need to be addressed concerning VNCs: (1) the detection of a given combination in the source language, and (2) its translation into the target language. If the first part fails, the words that constitute the MWE will be translated separately,

[1]Google Translate English-French and English-Spanish https://translate.google.co.uk apparently detects *take steps* as an MWE in (1) but not in (2) or (3).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 857–867, Osaka, Japan, December 11-17 2016.

which will usually result in an incorrect translation. Then, for the second part, it is vital to have the necessary information to know what translation should be given to each VNC. A further problem arises here, since the morphosyntax of this kind of MWE varies a great deal from one language to another, meaning that it is not necessarily translated by another VNC into the target language. This problem is especially acute when the source and target languages are typologically different, as with English, Spanish and Basque[2]. This is what happens in example (4).

(4) English (EN): *get married* (V+V)
Spanish (ES): *contraer matrimonio* (V+N)
'contract marriage'
Basque (EU): *ezkondu* (V)
'(to) marry'

In this paper, we present a linguistic analysis undertaken with the aim of improving the detection of VNCs in *Matxin* (Mayor et al., 2011), a rule-based MT system which translates English and Spanish into Basque. Although we ground our study in this particular MT system, our methodology, analysis and conclusions are relevant to any kind of NLP task that needs to be sensitive to lexical meaning.

The paper is structured as follows. After discussing related work (Section 2), we present our linguistic analysis (Section 3) including: our procedure for VNC tagging, how we classify the combinations, and levels of inter-annotator agreement. In Section 4 we present a VNC detection experiment, and give the results obtained by combining linguistic information with chunking and dependency parsing. Finally, in Section 5, we draw conclusions and propose directions for future work.

2 Related Work

It is widely acknowledged that good MWE processing strategies are necessary for NLP systems to work effectively (Sag et al., 2002), since these kinds of word combinations are very frequent in both text and speech. It is estimated that the number of MWEs in an English speaker's vocabulary is of the same order of magnitude as that of single words (Jackendoff, 1997), and that at least one MWE is used per sentence on average (Sinclair, 1991).

Various classifications of MWEs have been proposed, employing different criteria to match the requirements of a particular kind of target application. Some researchers propose a binary categorisation of literal and non-literal word combinations (Birke and Sarkar, 2006; Cook et al., 2008), whereas others propose a grading containing several MWE types based on semantic idiomaticity, considered as a continuum (Wulff, 2008). Within the Meaning-Text Theory, collocations are sorted according to the notion of lexical functions (Mel'ćuk, 1998), that is, taking into account how the component words are semantically related. Furthermore, some experiments have investigated automatic methods—such as distributional similarity or word embeddings—for the task of classification, leading to fairly good results (Baldwin et al., 2003; McCarthy et al., 2003; Fazly et al., 2007; Rodríguez-Fernández et al., 2016).

In addition to MWE classification, a great deal of work has been undertaken over the last two decades on MWE acquisition (Ramisch, 2015) and identification (Li et al., 2003; Seretan and Wehrli, 2009; Sporleder and Li, 2009). Precise and detailed syntactic information is crucial for both tasks, and, at the same time, MWE identification can also help parsers obtain better results (Seretan, 2013). Moreover, accurate MWE detection is crucial for MT, since MWEs vary greatly from one language to another, and are not usually translated word for word. In the context of MT systems, Wehrli (2014) states "the non-identification of collocations dramatically affects the quality of the output".

3 Linguistic Analysis

The linguistic analysis we present here aims at improving MWE processing in MT. More specifically, we base our study on Matxin (Mayor et al., 2011), a rule-based MT system for English-Basque and

[2]Whereas English (Germanic) and Spanish (Romance) are Indo-European languages, Basque is a non-Indo-European language which moreover belongs to no known language family.

Spanish-Basque translation. One of the problems Matxin has concerning MWEs is that it currently fails to detect many instances of morphologically flexible word combinations, since it only searches for word sequences against entries in a lexicon.

As mentioned in Section 1, our study foccusses on one particular kind of MWE: verb+noun combinations (VNCs). As well as the principal constituents of a verb and a noun, we also allow for combinations containing a preposition and/or a determiner in between. Candidate combinations were first gathered from machine-readable dictionaries and were then searched for in corpora, the most frequent combinations being selected for detailed analysis.

More details about the procedure for selecting the combinations are given in the following subsections, as well as explanations of a manual tagging process, the criteria used to classify the combinations, and the overall results and conclusions drawn from this analysis. How this information is used for VNC identification is explained in Section 4.

3.1 Selection of Verb+Noun Combinations

The Spanish combinations for this study were extracted from the Elhuyar Spanish-Basque dictionary[3], and the corpus used to obtain frequency information was made up of 491,853 sentences taken from a Spanish-Basque parallel corpus containing a range of text genres. A total of 150 distinct VNCs were selected, each of which occurred more than five times as a word sequence in the corpus.

For English, our original intention was to extract combinations from the Elhuyar *English*-Basque dictionary, in part because the Basque translations would be useful for the translation process in the MT system. However, the dictionary contained too few combinations for this study, so instead we decided to use the Oxford Collocations Dictionary (Deuter, 2008). After extracting the combinations matching our grammatical pattern, we searched for them in the British National Corpus (Burnard, 2007). If the verb and the noun (and the preposition, when necessary) were found as main elements in adjacent chunks more than 500 times, the combination was selected. The final set consisted of 173 combinations in all.

3.2 Tagging Process

The combinations were tagged manually and classified along lexico-semantic and morphosyntactic dimensions, as discussed in the next sections. Although annotators looked at corpora to take decisions, the tagging was not done on instances in a corpus but on combinations out of sentential context. Therefore, each annotator gave each combination a single tag per task.

The lexico-semantic classification was done for two reasons: to determine which combinations were worth detecting and which ones should not be treated as MWEs, and because making groups depending on the combinations' idiomaticity was considered relevant for the later translation process. The morphosyntactic data, on the other hand, was analysed to be used for VNC detection (Section 4).

The tagging was performed by five linguists, all of whom are Spanish native speakers and fluent in English. Firstly, a 'super-annotator' tagged all the data, comprising a total of 323 distinct combinations in Spanish and English. Then, the data were split in four parts, and a further four annotators each tagged one of these parts, following the guidelines created for this purpose.

3.3 Lexico-Semantic Classification

The tags assigned by the annotators separated the combinations into four lexico-semantic groups, from less to more idiomatic: (1) free expressions, (2) collocations and light verb constructions, (3) metaphoric expressions, and (4) idioms. This was not an easy task, as the boundaries between one group and another are not always clearly defined. Idiomaticity is rather understood as a continuum (Wulff, 2008), and some combinations are very difficult to classify (we return to this point in Section 3.5).

Idioms (also called **opaque expressions**) are combinations in which the whole meaning cannot be understood by looking at the meanings of the words separately. Two clear examples of these would be the sentences in examples (5) and (6), which are impossible to interpret correctly without knowledge of

[3] http://hiztegiak.elhuyar.eus/

the figurative meaning of the expressions in italics.

(5) Do not believe her, she is just *pulling your leg*.
= Do not believe her, she is just *joking*.

(6) Ese chico *no se corta un pelo*, es un descarado.
'That boy *does not cut a hair*, he is shameless.'
= That boy *is never intimidated*, he is shameless.

Metaphoric expressions are not used in their literal sense either, but it is possible to understand their meaning in terms of a metaphor, as in examples (7) and (8).

(7) He did not come to the meeting and the boss *had a word* with him.
= He did not come to the meeting and the boss *spoke* with him.

(8) Las experiencias de ese tipo *dejan huella*.
'These kinds of experiences *leave (a) mark*.'
= These kinds of experiences have a very significant effect (on people's life).

Unlike the combinations in examples (5)–(8), those in examples (9) and (10) are easily understandable on the basis of their component words; they belong to the group of **collocations and light verb constructions**. Collocations are defined as lexically constrained and recurrent combinations of words which are in a given syntactic relation (Evert, 2008; Bartsch, 2004). When they are VNCs, the verb is often a very common word which is semantically bleached—meaning that it loses its usual sense to a certain extent (Butt, 2010). These kinds of combinations are called light verb constructions (LVCs). Examples (9) and (10) would be classified in this group.

(9) Volunteers *gave support* to disadvantaged children.

(10) La educación *tiene* vital *importancia* para los niños desaventajados.
'Education *has* vital *importance* for disadvantaged children.'

Finally, **free expressions** are groups of words that can be combined freely, that is, following the standard lexical and grammatical rules of a given language. These kinds of expressions are not idiomatic, and are thus not considered MWEs, as in examples (11) and (12). Therefore, the combinations sorted in this group by the annotators were excluded for the later detection experiment (Section 4).

(11) They *are using a* new *technique* now.

(12) Este año *iremos a un lugar* diferente.
'This year *we will go to a* different *place*.'

As mentioned in Section 3.2, we consider that classifying the VNCs is relevant for translation. Our hypothesis is that the kind of translation a VNC should be given is often dependent on its lexico-semantic class. For instance, the combinations we have analysed so far suggest that, although idioms are usually translated by other (morphosyntactically equivalent or non-equivalent) idioms into the target language, they are unlikely to receive a word-for-word translation (see example (13)). On the other hand, in collocations, the noun is very likely to receive a direct translation, whereas the verb is often given a translation other than the one expected when it is not part of the collocation (see example (14)).

(13) EN: *pull* (somebody)*'s leg*
ES: *tomar el pelo* (a alguien)
'take (somebody)'s hair'
EU: (norbaiti) *adarra jo*

'play (somebody) the horn'

(14) EN: *take steps*
ES: *dar pasos*
'give steps'
EU: *pausoak eman*
'give steps'

We will not focus on the correlation between VNC classes and their translation in this paper. However, we do consider it an interesting topic for future investigation.

3.4 Morphosyntactic Classification

As well as the lexico-semantic tagging described above, we examined morphosyntactic features of combinations to classify them into three groups: (1) fixed combinations, (2) semi-fixed combinations, and (3) morphosyntactically free combinations. The annotators had to consider five questions to determine how fixed the combinations were:

- Does the noun phrase (NP) have a determiner? (always/never/optional)
- Is the NP singular or plural? (singular/plural/optional)
- Can there be a modifier (i.e. an adjective) inside the NP? (yes/no)
- Can the verb and the NP be separated by other words? (yes/no)
- Can the order of the elements be altered? (yes/no)

A given VNC needed to be classified as completely free when: the determiner and the number of the NP were marked as optional; there could be a modifier inside the NP; the verb and the NP could be separated by other words; and the order of the elements in the expression was judged to be alterable. When some of the answers were different to these, the combination had to be marked as semi-fixed, and as completely fixed if all the answers were different (that is, when the syntactic variability of the VNC was completely restricted).

None of the combinations was tagged as **fixed** by both the super-annotator and the second annotator, but this was not surprising, as VNCs which do not accept any kind of morphosyntactic variation are extremely rare. Usually, they can undergo some alterations (**semi-fixed expressions** as in examples (15) and (16)), or they can even be completely flexible (**morphosyntactically free expressions** as in examples (17) and (18)).

(15) be in love; be always in love; *be in the love; *be in loves.

(16) dar paso (a algo); dar siempre paso (a algo); *dar pasos (a algo)
'give way (to sth); always give way (to sth); *give ways (to sth)'

(17) cause a problem; cause two important problems; the problem was caused

(18) hacer un favor; hacer un gran favor; hacer dos favores; el favor que se hizo
'do a favour; do a big favour; do two favours; the favour that was done'

As these features have a direct impact on the detection of the combinations, the answers to the above-mentioned questions were also specified by the super-annotator one by one, so that this information could later be used to improve detection (see Section 4).

	Lexico-semantics	Morphosyntax
Agreement	70.52%	84.39%
κ	0.55	0.55

Table 1: IAA for English VN combinations.

	Lexico-semantics	Morphosyntax
Agreement	76.00%	81.34%
κ	0.63	0.61

Table 2: IAA for Spanish VN combinations.

3.5 Inter-Annotator Agreement

Inter-annotator agreement (IAA) was measured in two ways: the percentage of combinations in which the annotators agreed, and Cohen's Kappa, κ (Cohen, 1960).

As shown in Tables 1 and 2, annotator agreement was 70% to 84% for all tagging tasks and for both languages. With κ scores between 0.55 and 0.63, we conclude that the task is coherent and that the tagging results are usable for further investigation. The lexico-semantic IAA for English is similar to the IAA obtained in previous related work (Fazly et al., 2007; Vincze, 2012), and for Spanish it is appreciably higher.

Consistent with previous work (Seretan, 2013), we found that in our selection of 323 of the most frequently occurring VNCs in Spanish and English, collocations and LVCs are the most common type of combination, and that opaque expressions (idioms) are very scarce.

We also found that the combinations that led to disagreements among annotators were not classified in random groups, but were almost always in classes lexico-semantically (and morphosyntactically) close to each other (see Tables 3 and 4). Indeed, only a few combinations were classified in two groups that were not directly adjacent on the idiomaticity continuum. This provides further evidence that MWEs form a continuum of idiomaticity with no clear boundaries between MWE types (McCarthy et al., 2003).

		Other annotators			
		Idiom	**Metaphoric**	**Colloc/LVC**	**Free**
Super-annotator	**Idiom**	**0**	0	0	0
	Metaphoric	1	**24**	0	1
	Colloc/LVC	0	12	**73**	22
	Free	0	2	13	**25**

Table 3: Confusion matrix for English showing lexico-semantic tag agreement between the annotators.

		Other annotators			
		Idiom	**Metaphoric**	**Colloc/LVC**	**Free**
Super-annotator	**Idiom**	**1**	0	1	0
	Metaphoric	0	**20**	2	1
	Colloc/LVC	0	8	**69**	15
	Free	0	1	8	**24**

Table 4: Confusion matrix for Spanish showing lexico-semantic tag agreement between the annotators.

4 Identification Experiment

To test whether the analysed morphosyntactic data (see Section 3.3) could improve MWE detection, we undertook an experiment where three **identification methods** were combined and compared: (A)

the old one, used by Matxin, which searched only for word sequences; (B) a second one, based on the analysed linguistic data and automatically-produced chunking information; and (C) a third one, based on the analysed linguistic data and automatically-produced syntactic dependencies. Depending on how morphosyntactically fixed a given combination was, more or less linguistic restrictions were applied to identify them.

The **experimental set** was made of the combinations presented in Section 3, excluding the ones tagged as completely free by the super-annotator (Section 3.3). The final set consisted of 117 combinations in Spanish and 133 in English.

4.1 Results of the English Experiment

The corpus used for the experiment on English VNCs was the British National Corpus (Burnard, 2007), and chunking and dependency information was computed by the Stanford parser (Manning et al., 2014). A total of 152,051 occurrences of the 133 VNCs were identified by combining all three methods, 78.92% of which were not detected by method A, currently used for English-Basque translation in Matxin. Figure 1 shows the percentages of all the instances detected by each of the methods.

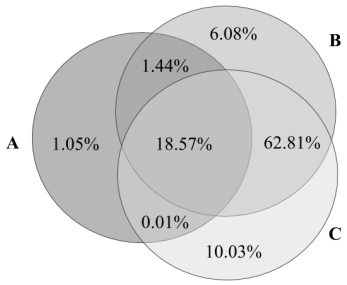


Figure 1: Percentages of English VNC occurrences identified by each method. (For clarity, areas are not drawn in scale with percentages)

We cannot calculate recall since our evaluation dataset contains only the occurrences identified collectively by the three methods, and it is almost certain that some occurrences of the VNCs under investigation were not detected. For future work, we would need to use MWE-tagged corpora to calculate recall, such as the Prague Czech-English Dependency Treebank (Uresova et al., 2013). In any case, the results obtained clearly show that the number of identified occurrences is increased considerably by using linguistic data specific to VNCs, as well as confirming that VNCs are commonly used in multiple morphosyntactic variations, as only 21.08% of the instances could be identified by searching for word sequences against entries in a lexicon.

To estimate the precision of VNC detection, we considered a representative sample of the full set, and evaluation was carried out manually by linguists. The precisions of methods B and C were not as good as that of method A. However, the evaluation on instances identified by both B and C methods reveals that detection quality is still very high when linguistic data specific to VNCs is combined with parsing (the second row of scores in Table 5).

	Additional VNCs %	**Precision**
Method A (in all)	21.08%	99%
Method B+C but not A	62.81%	96%
Method B only	6.08%	70%
Method C only	10.03%	79%

Table 5: Identification precision for the additional VNC occurrences detected in English

The least satisfactory results were those obtained by method B. When verifying the results, we noticed that the vast majority of false instances detected were light verb constructions (LVCs) containing verbs

that could also work as auxiliaries. In example (19), for instance, *have influences* is erroneously detected since *influence* is mis-analysed as the object of *have* rather than the subject of *have been likened*.

(19) These *influences have* also been likened to the forces effected by a millenarian journey to a new faith...

The overall improvement we obtained was substantial, as expected from previous work. Li et al. (2003) report an F-score improvement of 9 percentage points (86.9% to 95.6%) when using parsers and hand-crafted lexical patterns to identify phrasal verbs in English, as well as a precision improvement of 8 percentage points (90% to 98%). In our case, precision falls from 99% to 93% when combining all three methods, but the number of new instances detected suggests an appreciable increase in recall. As we already mentioned, MWE-annotated corpora would be needed to calculate recall and F-score and compare our results to those reported by other authors.

4.2 Results of the Spanish Experiment

For the experiment on Spanish, VNCs were searched in 15,182,385 sentences taken from the parallel English-Spanish corpus made public for the shared task in the ACL 2013 workshop on statistical MT[4], and the parser used was Freeling (Padró and Stanilovsky, 2012). A total of 433,092 occurrences were identified, 27.80% of which were not detected by method A (the percentages of the combinations identified by each method are shown in Figure 2). Consistent with the results obtained for English, this further reveals that the morphosyntactic data we took into account (Section 3.4) is very relevant for VNC identification.

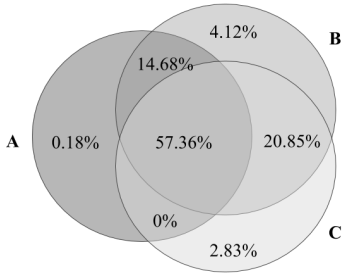


Figure 2: Percentages of Spanish VNC ocurrences identified by each method

Furthermore, as well as the quantity improving considerably, the manual evaluation reveals that the quality of our method is also very satisfactory. As is shown in Table 6, methods B and C, although not as precise as method A, got very good precision scores.

	Additional VNCs %	**Precision**
Method A (in all)	72.20%	99%
Method B+C but not A	20.85%	97%
Method B only	4.12%	93%
Method C only	2.83%	83%

Table 6: Identification precision for the additional VNCs detected in Spanish

As the corpora and parsers we used were different for English and Spanish, the experiments in both languages are not really comparable. However, it is evident that the improvement obtained for English was considerably higher than the one obtained for Spanish. Taking into account that the Freeling and Stanford parsers work in similar ways and that the manual tagging of the VNCs was done following the same criteria, this difference could suggest that syntactic variations of VNCs other than the canonical form are more common in English than in Spanish. One of the possible reasons for this could be the

[4] http://www.statmt.org/wmt13/translation-task.html

different word order inside NPs in both languages. In Spanish, adjectives can either precede or follow the head noun, whereas in English adjectives are almost never placed after the noun: *importantes pasos* or *pasos importantes* vs. *important steps* but not **steps important*. An exhaustive analysis would be needed to verify this hypothesis or identify other possible reasons.

5 Conclusions and Future Work

Morphosyntactically flexible MWEs constitute a problem for NLP systems, which often fail to process these kinds of word combinations correctly. In this paper, we presented a linguistic analysis undertaken with the aim of improving the identification of VNCs, as well as an experiment which shows how linguistic data can improve identification results greatly.

Firstly, we classified a selection of frequent VNCs in English and Spanish, following both lexico-semantic and morphosyntactic criteria. A total of 323 distinct combinations (173 in English and 150 in Spanish) were tagged by several annotators, with very reasonable inter-annotator agreement scores (κ 0.55 to 0.63). We noted moreover that the combinations that led to disagreements among annotators were always tagged in groups that were lexico-semantically and morphosyntactically close to each other, which gives further evidence that idiomaticity should be viewed as a continuum. More detailed morphosyntactic information was also specified for each combination, and this information was then used to improve VNC identification.

Our experiment confirmed that specific linguistic data about VNCs is useful for the identification of this kind of word combination, as it allows for the recognition of occurrences that do not match a combination's canonical form. Indeed, a large number of instances that were not identified by searching for fixed word sequences could be identified by combining linguistic data with chunking information and syntactic dependencies, with fairly good precision scores (79% to 97%).

Building on the satisfactory results obtained, we will test our methods in the context of MT, and we will keep analysing more VNCs. The next step will be to explore what kind of data is needed for an adequate translation of VNC combinations within MT systems. In addition, we intend to investigate how semantic information can be used within the translation process.

Acknowledgements

Uxoa Iñurrieta's doctoral research is funded by the Spanish Ministry of Economy and Competitiveness (BES-2013-066372). The work was carried out in the context of the SKATeR (TIN2012-38584-C06-02) and TADEEP (TIN2015-70214-P) projects. We thank Diana McCarthy for helpful advice, as well as Begoña Altuna, Nora Aranberri, Ainara Estarrona and Larraitz Uria for helping us with the tagging work.

References

Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows. 2003. An empirical model of multiword expression decomposability. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, 89–96.

Sabine Bartsch. 2004. *Structural and Functional Properties of Collocations in English: A Corpus Study of Lexical and Pragmatic Constraints on Lexical Co-occurrence*. Gunter Narr Verlag, Tübingen.

Julia Birke and Anoop Sarkar. 2006. A clustering approach for nearly unsupervised recognition of nonliteral language. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, 329–336.

Lou Burnard (ed.) 2007. *Reference Guide for the British National Corpus (XML Edition)*. Oxford University Computing Services.

Miriam Butt. 2010. The light verb jungle: Still hacking away. In Mengistu Amberber, Brett Baker, and Mark Harvey (eds.), *Complex Predicates: Cross-linguistic Perspectives on Event Structure*. Cambridge University Press, 48–78.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1): 37–46.

Paul Cook, Afsaneh Fazly, and Suzanne Stevenson. 2008. The VNC-Tokens dataset. In *Proceedings of the LREC Workshop Towards a Shared Task for Multiword Expressions (MWE 2008)*, 19–22.

Margaret Deuter (ed.) 2008. *Oxford Collocations Dictionary for Students of English*. Oxford University Press.

Stefan Evert. 2004. *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD dissertation, IMS, University of Stuttgart.

Stefan Evert. 2008. Corpora and collocations. In Anke Lüdeling and Merja Kytö (eds.), *Corpus Linguistics: An International Handbook*. Mouton de Gruyter, Berlin, 1212–1248.

Afsaneh Fazly and Suzanne Stevenson. 2007. Distinguishing subtypes of multiword expressions using linguistically-motivated statistical measures. In *Proceedings of the ACL-SIGLEX Workshop on a Broader Perspective on Multiword Expressions*, 9–16.

Ray Jackendoff. 1997. *The Architecture of the Language Faculty*. MIT Press, Cambridge, MA.

Wei Li, Xiuhong Zhang, Cheng Niu, Yuankai Jiang, and Rohini Srihari. 2003. An expert lexicon approach to identifying English phrasal verbs. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics: Long Papers*, 513–520.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55–60.

Aingeru Mayor, Iñaki Alegria, Arantza Díaz de Ilarraza, Gorka Labaka, Mikel Lersundi, and Kepa Sarasola. 2011. Matxin, an open-source rule-based machine translation system for Basque. *Machine Translation*, 25(1): 53–82.

Diana McCarthy, Bill Keller, and John Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, 73–80.

Igor A. Mel'ćuk. 1998. Collocations and lexical functions. In Anthony P. Cowie (ed.), *Phraseology. Theory, Analysis, and Applications*. Oxford University Press, 23–53.

Lluís Padró and Evgeny Stanilovsky. 2012. FreeLing 3.0: Towards wider multilinguality. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, 2473–2479.

Carlos Ramisch. 2015. *Multiword Expressions Acquisition: A Generic and Open Framework*. Springer International Publishing, Switzerland.

Sara Rodríguez-Fernández, Luis Espinosa-Anke, Roberto Carlini, and Leo Wanner. 2016. Semantics-driven recognition of collocations using word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL): Short Papers*, 499–505.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics (CICLING'02)*. Springer Berlin Heidelberg, 1–15.

Violeta Seretan and Eric Wehrli. 2009. Multilingual collocation extraction with a syntactic parser. *Language Resources and Evaluation*, 43(1): 71–85.

Violeta Seretan. 2013. On collocations and their interaction with parsing and translation. *Informatics*, 1(1): 11–33.

John Sinclair. 1991. *Corpus, Concordance, Collocation*. Oxford University Press.

Caroline Sporleder, and Linlin Li. 2009. Unsupervised recognition of literal and non-literal use of idiomatic expressions. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 754–762.

Zdenka Uresova, Jana Sindlerova, Eva Fucikova, and Jan Hajic. 2013. An analysis of annotation of verb-noun idiomatic combinations in a parallel dependency corpus. In *Proceedings of the Ninth Workshop on Multiword Expressions (MWE 2013)*, 58–63.

Veronika Vincze. 2012. Light verb constructions in the SzegedParalellFX English-Hungarian parallel corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, 2381–2388.

Eric Wehrli. 2014. The relevance of collocations for parsing. In *Proceedings of the 10th Workshop on Multiword Expressions (MWE 2014)*, 26–32.

Stefanie Wulff. 2008. *Rethinking Idiomaticity: A Usage-based Approach*. Continuum, London, New York.

Analyzing Gender Bias in Student Evaluations

Andamlak Terkik[1], **Emily Prud'hommeaux**[2], **Cecilia O. Alm**[2],
Christopher M. Homan[1], **Scott Franklin**[3]

[1]Golisano College of Computing & Information Sciences, Rochester Institute of Technology
[2]College of Liberal Arts, Rochester Institute of Technology
[3]College of Science, Rochester Institute of Technology
{†at3616|†emilypx|†coagla|§cmh|†svfsps}@{†rit.edu|§cs.rit.edu}

Abstract

University students in the United States are routinely asked to provide feedback on the quality of the instruction they have received. Such feedback is widely used by university administrators to evaluate teaching ability, despite growing evidence that students assign lower numerical scores to women and people of color, regardless of the actual quality of instruction. In this paper, we analyze students' written comments on faculty evaluation forms spanning eight years and five STEM disciplines in order to determine whether open-ended comments reflect these same biases. First, we apply sentiment analysis techniques to the corpus of comments to determine the overall affect of each comment. We then use this information, in combination with other features, to explore whether there is bias in how students describe their instructors. We show that while the gender of the evaluated instructor does not seem to affect students' expressed level of overall satisfaction with their instruction, it does strongly influence the language that they use to describe their instructors and their experience in class.

1 Introduction

Student evaluations of teachers (SETs), in which students are asked to provide their assessment of the quality of instruction they have received in a particular course, have been in use for over a century. At the end of a course, students are given forms, in paper or electronic format, containing a series of questions about the course and instructor, some requiring Likert-type scale responses and others seeking free text responses. SETs have increasingly become the de facto standard for evaluating university-level teaching performance (Centra and Gaubatz, 2000). The impact of these surveys on faculty is enormous, as they affect tenure, promotion, and compensation decisions.

Despite playing an outsized role in assessing teaching effectiveness, SETs have numerous shortcomings as tools for this task. Students, for example, are understandably often not well equipped to determine an instructor's knowledge of the subject matter, and they can be unreliable judges of how well they have mastered material, one important and generally accepted measure of teaching effectiveness. Perhaps more troublingly, several studies have demonstrated biases, whether conscious or unconscious, in students' evaluations of women instructors and instructors of color.

Most previous research in this area has focused on numerical (more precisely, *ordinal categorical*) ratings of various qualities related to teaching effectiveness. In this paper, we instead focus on computational analysis of the text responses to open-ended questions found in SETs. In the first part of this paper, we present a supervised instructor satisfaction classifier trained to identify the satisfaction polarity of SET comments. Next, we apply this model to a much larger dataset to examine how satisfaction varies across both genders. Finally, we analyze the patterns of word choice associated with each gender in order to explore how students' language changes according to the gender of the instructor they are evaluating.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 868–876, Osaka, Japan, December 11-17 2016.

2 Background

Several previous studies have found evidence for some sort of gender bias in SETs, although the results have been somewhat mixed and inconclusive. Female instructors reportedly receive lower overall numerical ratings than their male counterparts, particularly when male students evaluate them (Basow and Silberg, 1987; Basow, 1995; Young et al., 2009; Boring, 2015). At the same time, women whose personality and teaching style conform to expected gender stereotypes (e.g., warmth, helpfulness, accessibility) tend to receive higher marks overall, regardless of the gender of the evaluating students (Bennett, 1982; Kierstead et al., 1988). Gender bias also seems to vary according to the subject matter, level, and department of the course being taught (Basow, 1995; Centra and Gaubatz, 2000).

The rise of online instruction has provided a useful mechanism for identifying gender bias under more controlled conditions. MacNell et al. (2015) recently investigated the presence of gender bias in SETs of an online college-level social science class. In their experiment, two instructors, one male and one female, each taught two sections of the same course in an online setting: one in which the students were led to believe their instructor was a woman, the other in which they believed their instructor to be a man. Students in the sections with the *perceived* female instructors gave the instructors significantly lower scores in six areas, including the overall rating, than the students in the sections with the *perceived* male instructors. The differences in scores assigned to the instructors were not significantly different, however, across *actual* gender. Remarkably, it was noted that the instructor with the highest ratings was the female instructor who was perceived to be a man, while the instructor with the lowest ratings was the male instructor who was perceived to be a woman.

This previous work on gender bias in SETs has relied primarily on students' numerical ratings of their instructors for a variety of qualities related to their teaching effectiveness. In our research, we focus on the open-ended text comments that students are sometimes allowed to provide in order to elaborate on their numerical ratings. We rely on techniques and algorithms typically used for the NLP task of sentiment analysis, in which a variety of linguistic features are used to identify positive and negative tone expressed in natural language text (Mohammad, 2016; Liu, 2012). Our approaches to the task of identifying the degree of student satisfaction in their instructors, as expressed in their written comments, are inspired by, though distinct from, seminal work by Turney and colleagues (2002; 2003). Although our methods and features are not independently novel, the application of these methods in combination to the task of analyzing comments in SETs and the framing of the task itself constitute new and important contributions to the fields of NLP, gender studies, and education theory.

3 Data

SETs from a period of eight years were collected from a variety of undergraduate courses in math, physics, statistics, biology, and chemistry offered at a four-year, degree-granting institution in the United States. STEM courses were chosen because of national focus on gender disparities in physical sciences and the potential to eventually explore differences between scientific fields that do show such disparities (physics, mathematics) and those that do not (biology). Focusing on STEM also allows us to sample the majority of students, as the introductory courses in these fields are often service courses required for computing and engineering majors and can be used to meet distribution requirements for students in the humanities and social sciences.

We divide the SETs into three groups, which we call *small-labeled*, *medium-unlabeled*, and *large-unlabeled*. Each item in the first two sets contains one student response to the prompt: "*Comment on the instructor's strength and weaknesses*." The *large-unlabeled* set contains responses to multiple distinct prompts for comments on specific qualities associated with teaching effectiveness, including helpfulness, materials, organization, and presentation. Table 1 shows the the number of comments in the three groups.

The *small-labeled* dataset is the full set of "strengths and weaknesses" comments for two introductory statistics courses taught by multiple professors over several semesters, manually labeled by an undergraduate research assistant. We instructed this annotator to rate the level of satisfaction expressed in the comments. The options were *very satisfied*, *somewhat satisfied*, *neither satisfied nor dissatisfied*, *somewhat dissatisfied* and *very dissatisfied*. The comments in the remaining two datasets were not manually

Data set	Size
small-labeled	2,076
medium-unlabeled	15,896
large-unlabeled	107,855

Table 1: The three comment sets used in this study.

	Fleiss' Kappa	%Overlap
5 classes	0.49	0.61
3 classes	0.58	0.73

Table 2: Agreement scores for all annotators.

labeled. We trained and tested our classifier, discussed in detail below, on the *small-labeled* data, and then used it to predict the satisfaction level of the comments in the *medium-unlabeled* set. The *large-unlabeled* set was used for computing analytics and identifying terms strongly associated with either gender.

3.1 Inter-rater Agreement of Manual Annotations

Annotating the satisfaction level of a given text is an inherently subjective task. Since our classifier, described below, is trained on these annotations, it is important to estimate their reliability. Three individuals (co-authors), including the research assistant who rated the entire *small-labeled* set, annotated a subset of 100 comments using the previously described scheme. All of the annotations were performed after anonymizing the text and replacing all gendered titles, nouns, and pronouns with their equivalent gender neutral placeholders as explained below in Section 3.2. We then analyzed this newly annotated set by computing Fleiss' kappa (Randolph, 2009), using an online tool (Geertzen, 2012). Fleiss' kappa is a variant of Cohen's kappa (Byrt et al., 1993) that measures agreement among more than two raters.

The kappa scores were computed for two groupings. In the first grouping, each of the five satisfaction classes is considered independently. In the second, the two extreme classes on each side of the range were merged, yielding a 3-class rating scheme. The 3-class agreement scores exhibit less ambiguity, resulting in improved agreement.

Table 2 shows the Fleiss' kappa and overlap percentage of the results for both the 5-class and 3-class groupings. Both of these scores fall in the range of 0.4 to 0.6, which indicates moderate agreement under most interpretations of kappa in the psychology literature (Landis and Koch, 1977). It is also worth noting that inter-rater agreement scores such as the kappa score greatly depend on the level of subjectivity inherent in the task itself.

3.2 Data Preprocessing

Before extracting features for the satisfaction classifier, we anonymized the text by replacing all first and last names with a placeholder, merged all gendered pronouns (e.g., both *he* and *she* became *he/she*), replaced all words referring to a particular gender with a gender-neutral equivalent (e.g., *guy* and *lady* became *person*), downcased, and removed special characters. Another crucial step undertaken during preprocessing was the handling of negation terms. For instance, phrases such as *great at teaching* were differentiated from negative sentiments such as *not great at teaching*. This was achieved by applying the negation term *not* to each term that follows it until a special character or other negation term is encountered, using the method described by Narayanan et al. (2013).

The negation routine works by first detecting the negation markers *not* or *n't*. Whenever these markers are encountered, words that follow them are transformed into new terms prefixed with *not_*. After a negation marker is set, it negates every word that follows until a punctuation mark or another negation term is encountered. For instance, *not great at teaching* would be turned into *not not_great not_at not_teaching*. Term negation was responsible for an increase of about 4 percentage points in classification accuracy.

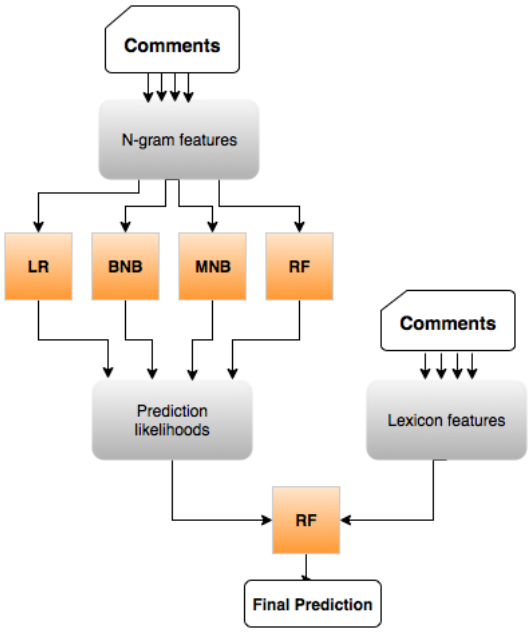


Figure 1: Architecture of the satisfaction classifier.

4 Method

4.1 Lexical Features

We extracted two types of lexical features from the data: n-gram features and sentiment term scores. Unigrams, bigrams, and trigrams served as features for the first tier of the classifier. Bigrams and trigrams can model useful local contextual features that unigrams are unable to model. For example, while unigrams features would be sufficient to capture single-word terms such as *intelligent* and *nice*, higher-order n-grams are required to capture the composite meaning found in phrases such as *extremely well* or *hardly ever available*. Over 30,000 n-grams were extracted from the dataset, resulting in a feature vector of this length for each comment.

Sentiment term scores were obtained by computing the aggregate positive and negative scores for each comment. To compute these aggregate scores, the prior polarities of the terms were determined using domain-independent lexicons. We relied on three general-purpose sentiment lexicons: the MPQA lexicon (Wilson et al., 2005), the NRC emotion lexicon (Mohammad and Turney, 2010; Mohammad and Turney, 2013), and Bing Liu's opinion lexicon (Liu, 2012). For each comment, the aggregate raw positive and negative term scores were computed from the scores from each of the three lexicons. Therefore, a 6-valued (i.e., 3 dictionaries x 2 sentiments) feature vector was computed for each comment.

4.2 Classifier Architecture

The classifier was designed as a two-tier system called *stacked generalization* (Wolpert, 1992), illustrated in Figure 1. The first tier comprises four classifiers: a random forest model, a multinomial naive Bayes model, a Bernoulli naive Bayes model, and a logistic regression model, each trained on unigram, bigram, and trigram features. The class likelihood predictions obtained from these four models, along with sentiment term scores, were then used to train a final classifier in the second tier. We used for this final classifier a random forest with parameters similar to those in the first tier.

Both the multinomial and the Bernoulli naive Bayes models have performed well in previous sentiment classification tasks (Pang et al., 2002) similar to ours. With both models we used Laplacian smoothing (i.e., $\alpha = 1.0$) with uniform priors. We trained and evaluated the random forest with 100 trees having a maximum depth of 80. The framework was implemented using the scikit-learn machine learning library (Pedregosa et al., 2011).

Total comments	2076
Total lexicon types	73802
Total tokens	80970
Type/token ratio	0.9
Avg. comment length	206 characters

Table 3: Descriptive statistics for the *small-unlabeled* data set.

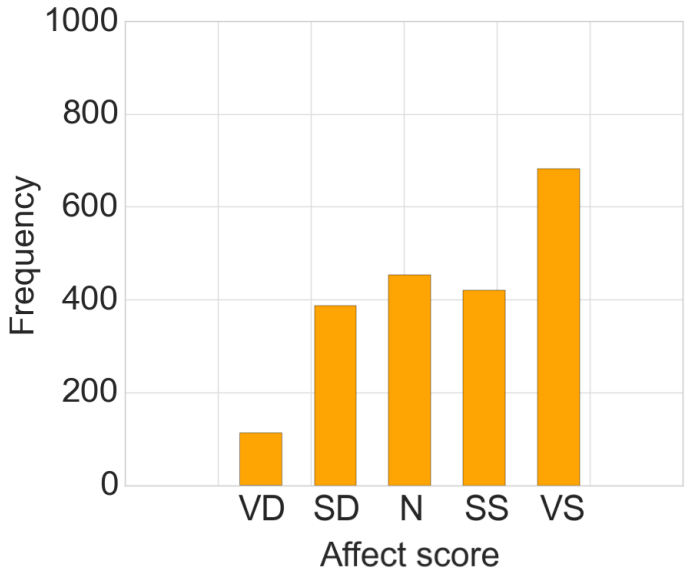


Figure 2: Distribution of 5 satisfaction levels in the *small-labeled* data set.

4.3 Results

We considered two classification tasks.

1. The *extremes* task: distinguishing *very dissatisfied* from *very satisfied*

2. The *merged* task: distinguishing (*very dissatisfied* or *somewhat dissatisfied*) from (*somewhat satisfied* or *very satisfied*)

In each case, we tested the classifier by running 10-fold cross validation on the *small-labeled* data. Table 3 presents statistics compiled from those tests. Figure 2 shows the distribution of satisfaction labels.

We evaluate a majority-class baseline and a Radial Basis Function (RBF) kernel-based SVM with penalty term $C = 2.0$, along with the main ensemble classifier. The SVM classifier utilized the same feature set as the ensemble one. However, instead of a two-tier architecture, the SVM directly combines the n-gram features with sentiment term score features. We report the classification accuracy and F1-scores in Table 4.

We see that our ensemble classifier yields large improvements over the majority-class baseline. This is especially true for the more challenging *merged* task. It also outperforms the SVM classifier by a notable margin for both tasks. Given that inter-annotator reliability on the 3-class task was just under $\kappa = .6$, achieving classification accuracy of 81% is impressive. Although there is room for improvement, these results demonstrate the efficacy of our framework.

	Baseline	SVM	RF Ensemble
Extremes task (n=795)			
Accuracy	86%	86%	**91%**
F1-score	80%	80%	**91%**
Merged task (n=1602)			
Accuracy	69%	79%	**81%**
F1-score	57%	77%	**81%**

Table 4: Classification accuracy and F1-score for both tasks. Boldfacing marks performance increases.

	Women	Men
small-labeled		
Satisfied	74%	62%
Dissatisfied	26%	38%
medium-unlabeled		
Satisfied	94%	94%
Dissatisfied	6%	6%

Table 5: Affect distribution broken down by gender. The top half shows breakdown of affect as annotated in the *small-labeled* set. The bottom half shows breakdown of affect in *medium-unlabeled* as predicted by the ensemble classifier.

5 Further Analysis

5.1 Satisfaction by Gender

For the *small-labeled* dataset, we computed the ratio of manually labeled *satisfied* to *dissatisfied* comments by gender. For the *medium-unlabeled* dataset, we computed this ratio using the satisfaction values from our classifier. Table 5 shows the results of these comparisons.

In the manually annotated dataset, male instructors receive slightly less favorable satisfaction ratings, in contrast to previous work reporting higher numerical ratings for male instructors. We note that this discrepancy is unlikely to be related to the gender of the students themselves since men were somewhat over-represented in the student body of the university from which the SETs were gathered. Rather, it seems that students were more satisfied with the instruction that they received from female instructors in the two introductory statistics courses from which these comments were drawn. We note, however, that satisfaction is a relatively subjective concept, and so may be influenced by the annotator's perception.

This difference in satisfaction disappears in the *medium-unlabeled* set, where the classifier predicts satisfaction levels to be equally distributed between genders. This could be an artifact of the larger size of the dataset, the broader range of course subject matter and level, the larger number of instructors, or simply the behavior of the classifier itself. In any case, our results do not seem to provide evidence for the presence of gender bias in students evaluations of teaching effectiveness. This does not preclude, however, the possibility of differences in students' language use according to the gender of the instructor being evaluated.

5.2 Gendered Language

In order to understand how word usage differs by the gender of the rated instructor, we first normalized and lemmatized the *large-unlabeled* set to account for morphological variation and abbreviation. We then ranked words based on their strength of co-occurrence, in terms of mutual information (MI) (Church and Hanks, 1990), with each gender. We selected the top 200 words from this ranking and sorted them into two groups based on their semantic functions. The first group contains terms used to *address* or *refer* to the instructor, and the second contains words *describing* an instructor or a student's experiences. We report the occurrence count of each term per 1000 comments after adjusting for the number of comments for each gender.

	F (per 1000)	M (per 1000)
Prof./professor	167	**209**
<Last name>	139	**151**
Dr.	75	**77**
teacher	**95**	79
instructor	172	172
<First name>	22	21

Table 6: Terms of address used to refer to faculty. Term frequency per 1000 comments adjusted by number of comments for both genders.

Word	F	M	Diff
amazing	**32**	18	128%
love(d)	**59**	32	84%
wonderful	**28**	12	57%
organized	**243**	178	37%
willing	**114**	88	30%
helpful	**454**	402	13%
tangent(s)	3	**16**	400%
funny	4	**14**	250%
knowledgeable	21	**33**	57%
interesting	68	**92**	35%
understanding	110	**126**	15%

Table 7: Gender differences in words used to describe men vs. women faculty. Values are per 1000 comments, adjusted by number of comments for each gender.

Table 6 shows that students are more likely to refer to their male instructors with the appropriate professional title (e.g., *Prof.*, *Dr.*) and by their last names. Conversely, female instructors are more likely to be referred to by their first names or descriptors that do not reflect their status as university professors (e.g., *the teacher* or *the instructor*). It is important to keep in mind that these comments were compiled from an institution having a faculty with roughly similar distributions of professional qualifications for both genders. These results therefore demonstrate an unwarranted bias toward more frequent use of prestigious titles and traditionally respectful forms of address for male instructors, regardless of their actual academic qualifications.

Table 7 shows the words with the most extreme differences in frequency according to instructor gender. Women were far more likely to be described with very positive but generic terms (*amazing, wonderful, loved*) than men. Perhaps more interestingly, students tended to describe women more often than men in terms of how the instructors impacted their learning experiences (*organized, willing, helpful*). Men, on the other hand, were recognized primarily for personal qualities (*funny, knowledgeable, interesting, understanding*) that may be independent of their ability to teach. The only negative term on this list, *tangent*, was also the term with the largest relative frequency difference between genders.

6 Conclusion

In this paper, we investigated the use of computational methods to analyze the language used in open-ended comments from student evaluations of teaching effectiveness in order to explore the possibility that gender bias exists in these evaluations. In contrast to previous research that relies on numerical ratings, our results fail to reveal differences by instructor gender in overall student satisfaction, as expressed in written comments. This results holds whether those satisfaction values are determined via direct human annotation or from machine learning models trained on the annotations. We do, however, observe real,

qualitative, gender-based differences in the language students use when providing written comments about their instructors.

Future work will follow several distinct but related paths. First, we will continue to develop our classifier using more complex features of language, such as those derived from semantic role labels or extracted from neural word embeddings or other vector space models. Secondly, we will explore the various student-assigned numerical ratings that accompany the text comments analyzed here. In particular, we hope to compare these ratings with our automatically generated satisfaction ratings in order to see the degree to which positive comments correlate strongly with high numerical ratings. As for gendered language, we plan expand our analysis by exploring whether certain syntactic structures are more strongly associated with either gender. Finally, we plan to investigate the various potential confounding factors in data, including subject matter, level, and instructor rank, as well as features of the students themselves, in order to shed light on the mixed and inconclusive evidence for gender bias described in the literature.

Acknowledgments

Many thanks to Ja'Nai Gray for providing manual annotations of the data used in this study and to the anonymous reviewers for their helpful feedback and suggestions.

References

Susan A. Basow and Nancy T. Silberg. 1987. Student evaluations of college professors: Are female and male professors rated differently? *Journal of Educational Psychology*, 79(3):308–314.

Susan A. Basow. 1995. Student evaluations of college professors: When gender matters. *Journal of Educational Psychology*, 87(4):656–665.

Sheila K. Bennett. 1982. Student perceptions of and expectations for male and female instructors: Evidence relating to the question of gender bias in teaching evaluation. *Journal of Educational Psychology*, 74(2):170–179.

Anne Boring. 2015. Gender Biases in Student Evaluations of Teachers. Technical report.

Ted Byrt, Janet Bishop, and John B. Carlin. 1993. Bias, prevalence and kappa. *Journal of Clinical Epidemiology*, 46(5):423–429.

John A. Centra and Noreen B. Gaubatz. 2000. Is there gender bias in student evaluations of teaching? *Journal of Higher Education*, pages 17–33.

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

Jeroen Geertzen. 2012. Inter-rater agreement with multiple raters and variables. https://nlp-ml.io/jg/software/ira/.

Diane Kierstead, Patti D'Agostino, and Heidi Dill. 1988. Sex role stereotyping of college professors: Bias in students' ratings of instructors. *Journal of Educational Psychology*, 80(3):342–344.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Lillian MacNell, Adam Driscoll, and Andrea N. Hunt. 2015. What's in a name: Exposing gender bias in student ratings of teaching. *Innovative Higher Education*, 40(4):291–303.

Saif M. Mohammad and Peter D. Turney. 2010. Emotions evoked by common words and phrases: Using Mechanical Turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34. Association for Computational Linguistics.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Saif M. Mohammad. 2016. Sentiment analysis: Detecting valence, emotions, and other affectual states from text. In Herb Meiselman, editor, *Emotion Measurement*. Elsevier.

Vivek Narayanan, Ishan Arora, and Arjun Bhatia. 2013. Fast and accurate sentiment classification using an enhanced naive Bayes model. In *Intelligent Data Engineering and Automated Learning–IDEAL 2013*, pages 194–201. Springer.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing-Volume 10*, pages 79–86.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830.

Justus J. Randolph. 2009. Free-marginal multirater kappa (multirater κ_{free}): An alternative to Fleiss' fixed-marginal multirater kappa. *Advances in Data Analysis and Classification*, 4.

Peter D. Turney and Michael L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, 21(4):315–346.

Peter D. Turney. 2002. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 417–424.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354. Association for Computational Linguistics.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5(2):241–259.

Suzanne Young, Leslie Rush, and Dale Shaw. 2009. Evaluating gender bias in ratings of university instructors' teaching effectiveness. *International Journal for the Scholarship of Teaching and Learning*, 3(2):19.

Adverse Drug Reaction Classification With Deep Neural Networks

Trung Huynh[1,3], **Yulan He**[2], **Alistair Willis**[1] **and Stefan Rüger**[1]
[1]Knowledge Media Institute, Open University, UK
[2]Systems Analytics Research Institute, Aston University, UK
[3]Google UK
trunghlt@gmail.com, y.he@cantab.net
{alistair.willis, stefan.rueger}@open.ac.uk

Abstract

We study the problem of detecting sentences describing adverse drug reactions (ADRs) and frame the problem as binary classification. We investigate different neural network (NN) architectures for ADR classification. In particular, we propose two new neural network models, Convolutional Recurrent Neural Network (CRNN) by concatenating convolutional neural networks with recurrent neural networks, and Convolutional Neural Network with Attention (CNNA) by adding attention weights into convolutional neural networks. We evaluate various NN architectures on a Twitter dataset containing informal language and an Adverse Drug Effects (ADE) dataset constructed by sampling from MEDLINE case reports. Experimental results show that all the NN architectures outperform the traditional maximum entropy classifiers trained from n-grams with different weighting strategies considerably on both datasets. On the Twitter dataset, all the NN architectures perform similarly. But on the ADE dataset, CNN performs better than other more complex CNN variants. Nevertheless, CNNA allows the visualisation of attention weights of words when making classification decisions and hence is more appropriate for the extraction of word subsequences describing ADRs.

1 Introduction

Adverse Drug Reactions (ADRs) are potentially very dangerous to patients and are amongst the top causes of morbidity and mortality (Pirmohamed et al., 2004). Many ADRs are hard to discover as they happen to certain groups of people in certain conditions and they may take a long time to expose. Healthcare providers conduct clinical trials to discover ADRs before selling the products but normally are limited in numbers. Thus, post-market drug safety monitoring is required to help discover ADRs after the drugs are sold on the market. In the United States, Spontaneous Reporting Systems (SRSs) is the official channel supported by the Food and Drug Administration. However these system are typically under-reported and many ADRs are not recorded in the systems. Recently unstructured data such as medical reports (Gurulingappa et al., 2012b; Gurulingappa et al., 2012a) or social network data (Ginn et al., 2014; Nikfarjam et al., 2015) have been used to detect content that contains ADRs. Case reports published in the scientific biomedical literature are abundant and generated rapidly. Social networks are another source of redundant data with unstructured format. While an individual tweet or Facebook status that contains ADRs may not be clinically useful, a large volume of these data can expose serious or unknown consequences.

Common approaches to detect content with ADRs used Support Vector Machines (SVMs), Random Forest, Maximum Entropy classifiers with heavily hand-engineered features (Rastergar-Mojarad et al., 2016; Sarker et al., 2016; Zhang et al., 2016). These features normally include n-grams with different weighting schemes. When used with unigrams, these approaches suffer from the fact that their models do not take in account the interaction between terms and their orders. This problem can partially be solved by using bi-grams or trigrams. However this leads to the number of features exploding, and the models are thus easily overfitted. Meanwhile neural networks with pre-trained word representations have

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 877–887, Osaka, Japan, December 11-17 2016.

had some successes in other text classification tasks (Kalchbrenner et al., 2014; Kim, 2014; Zhou et al., 2015; Yang et al., 2016). Word representations that are typically pre-trained with unlabelled data are matrices that can be used to project words into a dense low-dimensional space (typically from 50 to 300 dimensions). These neural networks often contain convolutional filters or recurrent connections that compute weighted sums of words and their contexts.

In this paper, we train word embeddings and use them as parameters to different neural network architectures to classify documents to whether they contain ADR content. We show that even without engineered features our neural networks with word embeddings outperform maximum-entropy classifiers with different weighting schemes for n-gram features.

The rest of the paper is organised as follows. Section 2 discusses related work on ADR detection from text and briefly describes word embeddings. Various neural network architectures including two new models, Convolutional Recurrent Neural Networks (CRNN) and Convolutional Neural Network with Attention (CNNA), are presented in Section 3. Experimental setup and results are discussed in Section 4 and 5 respectively. Finally, Section 6 concludes the paper.

2 Related Work

2.1 ADR Detection from Text

Natural Language Processing (NLP) approaches have been used to detect ADRs and their relations from Electronic Health Records (EHR) (Wang et al., 2009; Friedman, 2009) and clinical reports (Aramaki et al., 2010; Gurulingappa and Fluck, 2011). Both EHRs and clinical reports have several advantages over plain text or social network data such as they contain more complete records of patients' medical history, treatments, conditions. Leaman and Wojtulewicz (2010) are ones of the first to attempt to extract ADRs from text and social networks. They generated a golden data set for DailyStrength[1], a social network where its users share health-related struggles and successes with each other, and lexicons created from UMLS Methathesaurus[2], SIDER (Kuhn et al., 2010) and The Canada Drug Adverse Reaction Database[3]. Their data set contains a total of $6,890$ comment records. Their approach is rather straightforward, which is to use direct matches of terms in their built lexicons against terms tokenised from the comments. They reported a precision of 78.3%, a recall of 69.9% and an F-score of 73.9%. Further work that focused on exploring existing or expanded lexicons to find ADRs can be found at (Benton et al., 2011; Harpaz et al., 2012; Gurulingappa et al., 2012b; Yates and Goharian, 2013; Liu and Chen, 2013). Lexicon-based approaches are limited in the number of drugs studied or the number of target ADRs. Nikfarjam and Gonzalez (2011) introduces a rule-based approach on the same DailyStrength data set. Though it does not perform as well as the lexicon-based approach, it can detect expressions not included in the lexicons.

With the emergence of annotated data, there have been more machine-learning based approaches to ADRs detection. Gurulingappa et al. (2011) used Decision Trees, Maximum Entropy and SVMs with many engineered features. They obtained an F-score of 77% for ADR class with ADE data set. Sarker and Gonzalez (2015) used SVMs with different feature sets from combined data sets (ADE, Twitter and DailyStrength). They observed that combining Twitter with ADE data sets or DailyStrength with Twitter data sets help improving their performances. Nikfarjam et al. (2015) used Conditional Random Fields to simultaneously detect ADRs and the condition for which the patient is taking the drug. In addition to traditional features, they introduced embedding clusters features trained with word2vec and k-means clustering. Rastergar-Mojarad et al. (2016) and Zhang et al. (2016) used ensemble models that combine decision trees (Random Forest) or different classifiers with various features.

Overall, approaches to ADR detection have been limited with shallow models and heavily engineered features. There has been a lack of an end-to-end approach that relies on redundancy of unannotated and annotated data.

[1] `http://www.dailystrength.org/`

[2] National Library of Medicine. 2008. UMLS Knowledge Sources.

[3] `http://www.hc-sc.gc.ca/dhp-mps/medeff/index-eng.php`

2.2 Word Embeddings

Most of deep neural networks in NLP utilise an embedding that projects each unique word into a dense lower-dimensional space (typically from 50 to 300 dimensions) and use it as the input of the network. An *embedding* is a matrix $\mathbb{R}^{\mathrm{v}\times\mathrm{s}}$, where $\mathrm{v} \in \mathbb{R}$ is the size of vocabulary and $\mathrm{s} \in \mathbb{R}$ is the number of dimensions in the low dimensional space. These embeddings are normally trained from unlabelled text that are usually redundant in huge amounts from sources like Wikipedia or CommonCrawl[4]. The embeddings are usually trained in a fashion so that the dot product of vectors of a word and its neighbour word preserves the words' point-wise mutual information (PMI) (Mikolov et al., 2013; Pennington et al., 2014; Levy and Goldberg, 2014; Shazeer et al., 2016).

After being trained, these vectors can be used to look for word synonyms by looking for words with their vectors closest to the searched word's vector. They can also be used to answer certain types of questions like "what is to Italy like Paris to France?" by looking for words with vectors that is closest to vector $\overrightarrow{\mathrm{Paris}} - \overrightarrow{\mathrm{France}} + \overrightarrow{\mathrm{Italy}} = \overrightarrow{\mathrm{Rome}}$ (Mikolov et al., 2013). By representing words using these vectors, the model captures derived information from co-occurrences of the contained words from the unsupervised pre-training. Additionally using lower dimensional vector space also helps reduce overfitting. A tokenised sentence or document with their tokens projected by an embedding becomes a dense matrix that can then be fed as an input into a neural network.

3 Methods

In this section, we introduce a number of neural network architectures and propose two new models, Convolutional Recurrent Neural Networks (CRNN) and Convolutional Neural Network with Attention (CNNA)[5].

3.1 Convolutional Neural Network (CNN)

Deep Convolutional Neural Networks (CNN)s are recently extensively used in many computer vision (Alex Krizhevsky et al., 2012; Szegedy et al., 2014; Simonyan and Zisserman, 2014; He et al., 2015) and NLP tasks. In NLP, CNNs (Figure 1a) were previously used successfully in sentence classification and sentiment analysis (Collobert et al., 2011; Kim, 2014; Zhou et al., 2015). The network starts with a convolutional layer with Rectified Linear Units (RLUs) (Glorot et al., 2011). A RLU takes an input and returns the original input if it is larger than 0, otherwise, it returns 0. The convolutional filters normally have the same width as the word vectors, thus, produce feature maps with only 1 column. The network is then stacked by a max pooling layer that picks the maximum element from each column. The last layer is a feedforward layer to an output layer with either sigmoid (Equation 3) or softmax (Equation 4) activations depending on whether the classification is binary or multinomial. The mathematical formulations for different layers of the CNN are:

$$l^k_{1_{i1}} = \max\{(W^k_1 * \mathrm{X})_{i1}, 0\}, \tag{1}$$

$$l_{2_k} = \max_i\{l^k_{1_{i1}}\}. \tag{2}$$

If it is binary classification, we set

$$l_3 = \frac{1}{1 + \exp(-W_3^\top l_2 - b_3)}, \tag{3}$$

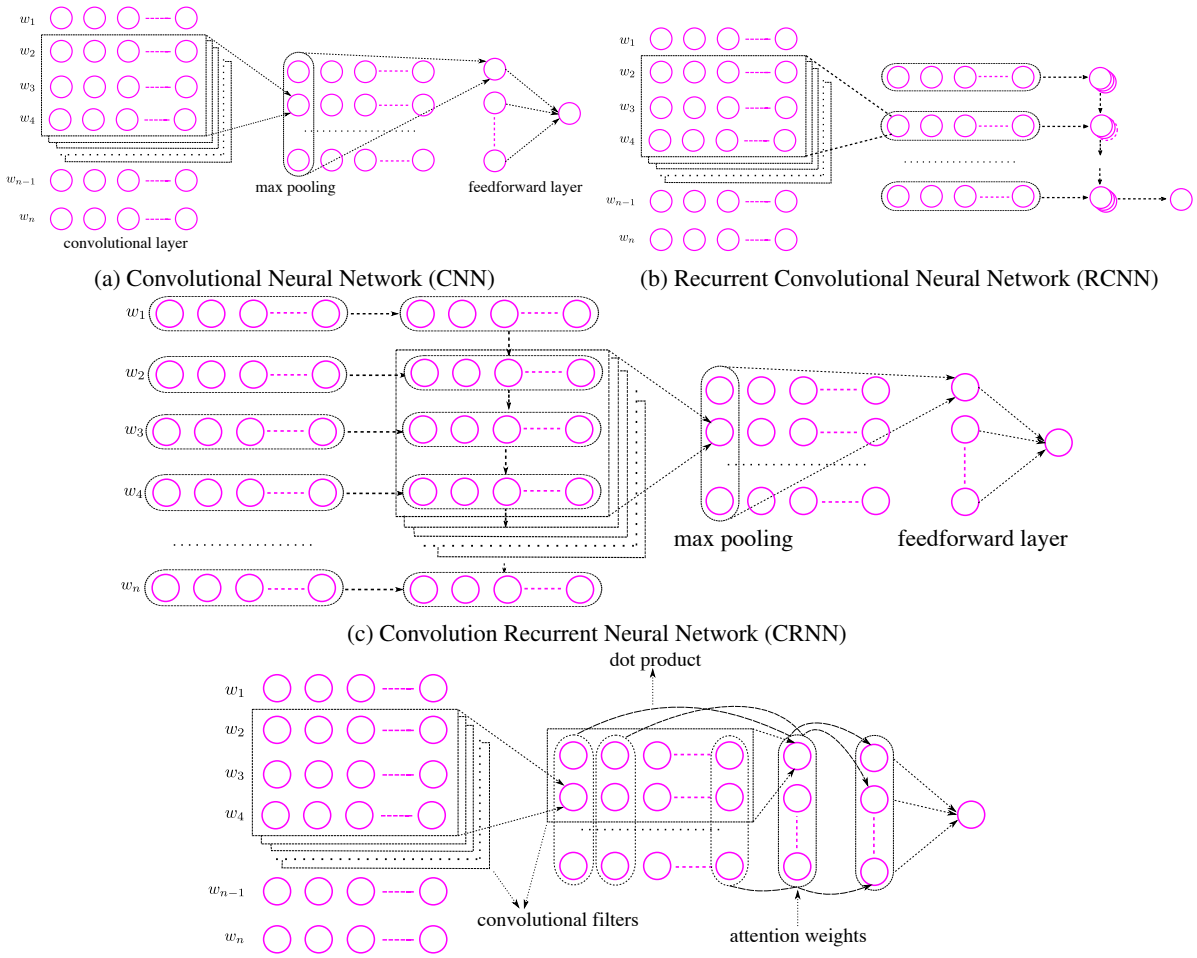
or, otherwise, if it is multinomial classification

$$l_{3_i} = \frac{\exp(W_3^\top l_2 + b_3)_i}{\sum_j \exp(W_3^\top l_2 + b_3)_j}. \tag{4}$$

Here, $\mathrm{X} \in \mathbb{R}^{\mathrm{d}\times\mathrm{s}}$ is the input matrix after the projection, $\mathrm{d} \in \mathbb{N}$ is the document length, $\mathrm{s} \in \mathbb{N}$ is the word vector length, $*$ denotes convolution, $W^i_1 \in \mathbb{R}^{\mathrm{h}\times\mathrm{e}}$, $W_3 \in \mathbb{R}^{\mathrm{k}\times 1}$ are the neural network weights, $b_3 \in \mathbb{R}$ is the bias term, $\mathrm{h} \in \mathbb{N}$ is the convolutional filter height and $\mathrm{k} \in \mathbb{M}$ is the number of convolutional filters.

[4] `http://commoncrawl.org/`

[5] Source code is available at `https://github.com/trunghlt/AdverseDrugReaction`



(a) Convolutional Neural Network (CNN)

(b) Recurrent Convolutional Neural Network (RCNN)

(c) Convolution Recurrent Neural Network (CRNN)

(d) Convolutional Neural Network with Attention (CNNA)

Figure 1: Various neural network architectures.

3.2 Recurrent Convolutional Neural Network (RCNN)

Another architecture that has achieved comparable results in sentence classification task is Recurrent Convolutional Neural Network (RCNN) (Zhou et al., 2015). The RCNN (Figure 1b) also starts with a convolutional layer like the CNN but followed by a recurrent layer rather than a max pooling layer. The convolutional filters have the same width as the embedding and are applied in the manner that the outputs have the same number of rows as the input. We also use the Rectified Linear function as the activation function for the convolutional layer. For the recurrent layer, at time step t, the recurrent node takes the input from the outputs produced by all the convolutional filters at row t and previous values at time step $t-1$. For activation, we use Gated Recurrent Units (Cho et al., 2014). Finally the nodes at the last time step are fully connected to a single node with a sigmoid activation to produce binary classification:

$$l^k_{1_{t1}} = \max\{(W^k_1 * \mathrm{X})_{t1}, 0\}, \tag{5}$$

$$l_{2_{t_j}} = \mathrm{gru}(l^*_{1_{t1}}), \tag{6}$$

$$l_3 = \frac{1}{1+\exp(-W_3^\top l_{2_\mathrm{d}} - b_3)}, \tag{7}$$

where $\mathrm{gru}(\mathrm{X}): \mathbb{R}^{d\times k} \rightarrow \mathbb{R}^{d\times r}$ denotes Gated Recurrent Unit (GRU) with input X, $r \in \mathbb{R}$ is the size of the output of the RNN, $t \in \mathbb{R}$ denotes a time step that is equivalent to the order of the window that produces the values from convolutional filters.

GRUs are recurrent units which have additional gating units. The gating units modulate the flow of information inside the unit. The activation h^i_j of a GRU at time t is a linear interpolation between

previous activations:

$$\mathrm{h}_t^j = (1 - \mathrm{z}_t^j)\mathrm{h}_{t-1}^j + \mathrm{z}_t^j \tilde{h}_t^j,$$

where z_t^j acts as a gate which decides how much the unit updates its content and it is computed by $\mathrm{z}_t^j = \sigma(W_z \mathrm{x}_t + U_z \mathrm{h}_{t-1})^j$, while $\tilde{\mathrm{h}}_t^j$ is a candidate activation, computed similarly to traditional recurrent unit, $\tilde{h}_t^j = \tanh\left(W\mathrm{x}_t + U(\mathrm{r}_t \odot \mathrm{h}_{t-1})\right)^j$, where r_t is a reset gate and $\odot$ is an element-wise multiplication. These reset gates can be computed similarly to the update gate $r_t^j = \sigma(W_r \mathrm{x}_t + U_r \mathrm{h}_{t-1})^j$.

The idea behind gated flows is to enable information further in the past to be propagated to the current unit with fewer time steps. With fewer time steps, the error gradient is passed by back-propagation more efficiently due to the propagated gradient is less prone to vanishing or exploding. (Cho et al., 2014) show that GRUs have better performance than traditional $\tanh$ and comparable performance to Long Short-Term Memory (LSTM) units.

3.3 Convolutional Recurrent Neural Network (CRNN)

Inspired by RCNN, we introduce a new architecture called Convolutional Recurrent Neural Network (Figure 1c) that stacks a convolutional layer on top of a recurrent layer, which is opposite to a RCNN. The intuition behind this is that the recurrent layer can capture the global contexts before information passed to the convolutional layer. The convolution and max-pooling layers replace the traditional average over hidden features or only hidden features at the last word in the sentence. We use GRUs for the recurrent layers and RLUs for the convolutional layer:

$$l_{1_i} = \text{gru}(\mathrm{X}_{i*}), \tag{8}$$

$$l_{2_{i1}^k} = \max\{(W_2^k * l_1)_{i1}, 0\}, \tag{9}$$

$$l_3 = \frac{1}{1 + \exp(-W_3^\top l_2 - b_3)}. \tag{10}$$

3.4 Convolutional Neural Network with Attention (CNNA)

Inspired by the works from (Bahdanau et al., 2015; Hermann et al., 2015; Rush et al., 2015; Rocktäschel et al., 2016; Yang et al., 2016) which use the attention mechanism that the generation of outputs at each consecutive time step is conditioned on different subsets of the input, we introduce a new architecture built on top of the CNN with additional attention mechanism (Figure 1d). The addition is one-filter convolutional layer on top of the direct outputs from the first convolutional layer. The outputs of this convolutional layer are normalised with *softmax* function so that they can have a sum of 1, which we call *attention weights*. These *attention weights* are then multiplied with the outputs from the first convolution (*dot product*). The outputs of this dot product are forward connected to a perceptron for binary classification.

The advantage of introducing the attention mechanism is that we can use these attention weights to extract words that the model mainly uses for the prediction. In practice, we found it very interesting and helpful to see which words are more weighted in the model's decisions (see Figure 2 in Section 5).

Even though getting more popular, attention mechanism has been mostly applied with recurrent neural networks (Bahdanau et al., 2015; Hermann et al., 2015; Rush et al., 2015; Rocktäschel et al., 2016; Yang et al., 2016). There are recently some works that incorporate attention mechanism with CNNs (Yin et al., 2016; Yin et al. , 2016). In (Yin et al., 2016), attention weights are computed differently by taking the dot product between the representation of the input query and the sentences in question-answer tasks. In (Yin et al. , 2016), even though called attention, the attention layers behave more like feature maps than traditional attention weights (multiplied with features) and are computed by matching two feature maps.

4 Experimental Setup

4.1 Datasets

We use two datasets for the evaluation of various neural network architectures. The first one is a Twitter dataset (Sarker et al., 2016) published for a shared task in Pacific Symposium on Biocomputing, Hawaii,

2016. The tweets associated with the data were collected using generic and brand names of the drugs, and also their possible phonetic misspellings. The tweets were annotated for presence of ADRs. In the shared task, 70% $(7,575)$ of the original data set is shared for training and the rest of the data is used for evaluation. Owing to Twitter's data terms and conditions, only the tweet ids are contained in the original file. At the time of this experiment, we could download only $5,108$ tweets (with 557 tweets with ADR descriptions) as many tweets are no longer accessible. Due to the difference in the size of the experimental data set, we can not compare our results directly with the previously reported baselines. Thus we reuse the codes published by (Zhang et al., 2016) that perform classification with the various algorithms (see Section 4.2 for further details).

The second dataset, the ADE (adverse drug effect) corpus, was created by (Gurulingappa et al., 2012b) by sampling from MEDLINE case reports[6]. Each case report provides important information about symptoms, signs, diagnosis, treatment and follow-up of individual patients. The ADE corpus contains $2,972$ documents with $20,967$ sentences. Out of which, $4,272$ sentences are annotated with names and relationships between drugs, adverse effects and dosages.

For both datasets, we use 10-stratified-fold cross-validation and report precision, recall and F-scores of various methods.

4.2 Baselines

For the Twitter dataset, it was reported from the shared task that both the best (Rastergar-Mojarad et al., 2016) and the second best (Zhang et al., 2016) approaches are classifiers with engineered features. In order to directly compare our results with the existing approaches, we have reimplemented these classifiers based on the published code by (Zhang et al., 2016) including term-matching classifier based on an ADR lexicon, maximum entropy with n-grams and TFIDF weightings or NB log-count ratio, and maximum entropy with word embeddings. We describe each of these methods below:

- *Term-matching based on an ADR lexicon* (**TM**). An existing ADR lexicon[7] is directly used for ADR detection. The lexicon contains $13,699$ terms describing side effects from COSTART, SIDER, CHV and DIEGO_Lab. A document is classified as positive if it contains a term from the lexicon.

- *Maximum-Entropy classifier with n-grams and TFIDF weightings* (**ME-TFIDF**). For a document $d \in \mathcal{D}$, an n-gram i has a weight of

$$F_i(d) = \begin{cases} \big(1 + \log(n_i(d))\big) \times \log\big(1 + \frac{|\mathcal{D}|+1}{|\{d' \in \mathcal{D} | n_i(d') > 0\}|+1}\big) & \text{if } n_i(d) > 0 \\ 0 & \text{otherwise,} \end{cases}$$

  where $n_i(d)$ is the number of times a term i appears in document d.

- *Maximum-Entropy classifier with n-grams and NB log-count ratio* (**ME-NBLCR**). Each n-gram i has a weight of

$$f_i = \begin{cases} \log\big(\frac{1+\sum_{d:y(d)=1} n_i(d)}{\sum_{i' \in \mathcal{V}}(1+\sum_{d:y(d)=1} n_{i' \in \mathcal{V}}(d))} \times \frac{\sum_{i' \in \mathcal{V}}(1+\sum_{d:y(d)=-1} n_{i'}(d))}{1+\sum_{d:y(d)=-1} n_i(d)}\big) & \text{if } n_i(d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

  where $\mathcal{V}$ is a set of all n-grams and $y(d) \in \{1, -1\}$ is the true label of each document.

- *Maximum-Entropy classifier with mean word embeddings* (**ME-WE**). This method simply uses the average of embeddings of words in each document as their input into a maximum-entropy classifier.

For the ADE dataset, the best performance published is 0.81 in F-score using SVMs trained from a rich set of features including n-grams, UMLS semantic types and concept IDs, synset expansions, polarity indicator features, ADR lexicon matches, and topics, etc. (Sarker and Gonzalez, 2015). However, since our ME-NBLCR outperforms SVMs on ADE, we don't report the results using SVMs here.

[6] https://www.nlm.nih.gov/bsd/indexing/training/PUB_050.htm

[7] http://diego.asu.edu/downloads/publications/ADRMine/ADR_lexicon.tsv

Method	Twitter Dataset				ADE Dataset			
	Precision	**Recall**	**F1**	**AUC**	**Precision**	**Recall**	**F1**	**AUC**
TM	0.13	0.89	0.23	0.59	0.30	0.99	0.46	0.53
ME-TFIDF	0.33	0.70	0.45	0.85	0.74	0.86	0.80	0.94
ME-NBLCR	0.79	0.14	0.23	0.83	0.91	0.79	0.84	0.95
ME-WE	0.27	0.73	0.40	0.82	0.48	0.70	0.57	0.76
CNN	0.47	0.57	**0.51**	**0.88**	0.85	0.89	**0.87**	**0.97**
CRNN	0.49	0.55	**0.51**	0.87	0.82	0.86	0.84	0.96
RCNN	0.43	0.59	0.49	0.87	0.81	0.89	0.83	0.92
CNNA	0.40	0.66	0.49	0.87	0.82	0.84	0.83	0.95

Table 1: Adverse drug reaction classification results on the Twitter and ADE datasets.

4.3 Training of Neural Networks

In all the described neural network architectures in Section 3, the training algorithm is Adadelta (Zeiler, 2012) with learning rate of 1.0, decay rate (ρ) of 0.95 using library Keras[8]. The embedding is trained together with other parameters. For each fold, we split the training dataset into training and validating sets. The training stops when there is no performance improvement on the validation set after 5 consecutive epochs. The batch size is set as 50. All convolutional window has a size of 5.

5 Results

We compare the precision, recall and F-scores of the positive class (instances labeled as containing the description of adverse drug reactions) of neural network architectures with the baselines in Table 1. Since both the Twitter and ADE datasets contain imbalanced class distribution, we also report the Area Under the ROC Curve (AUC) results. It can be observed that in general, results on the ADE dataset are better than those on the Twitter dataset. This is perhaps not surprising since tweets contain a lot of ill-grammatical sentences and short forms. Simply relying on an ADR lexicon for the detection of ADRs from text gives the worst results. Among the baselines, the best performing method is ME-TFIDF on the Twitter dataset where an F-score of 0.45 and an AUC value of 0.85 are obtained. But on the ADE dataset with more formal language, ME-NBLCR gives superior results compared to ME-TFIDF with an F-score of 0.84 and an AUC value of 0.95. Training MaxEnt from aggregated word embeddings (ME-WE) outperforms the term matching method (TM), but performs worse than both ME-TFIDF and ME-NBLCR.

All the neural network architectures perform similarly on the Twitter dataset and they improve upon the best baseline method ME-TFIDF by 4-6% in F-score and 2-3% in AUC. On the ADE dataset, CNN outperforms other neural network architectures and its performance gain over ME-NBLCR is 7% in F-score and 3% in AUC. Overall, CNN gives the best results although CRNN and CNNA are quite close to CNN in terms of AUC values. It is not very straightforward to explain why CNNs are better than the recurrent architectures in our experiments. Our hypothesis is that as ADR descriptions are composed of short fragments of texts, convolutions with small windows are enough to capture necessary information for ADR classification.

Since CNNA assigns a weight to each word when making classification decision, we show in Figure 2 a visualisation of attention weights of sampled tweets from the Twitter dataset. Words with higher attention weights are highlighted with darker blue colour. We can observe that most of the highlighted words are indeed related to descriptions of adverse drug effects. For example, "neck ache" and "lower back pain" in the fifth tweet and "dry eyed" in the seventh tweet. The above results suggest that although CNNA gives slightly worse results compared to CNN for ADR classification, it presents results in a more interpretable form and could be potentially used for the extraction of word subsequences actually

[8] http://keras.io/

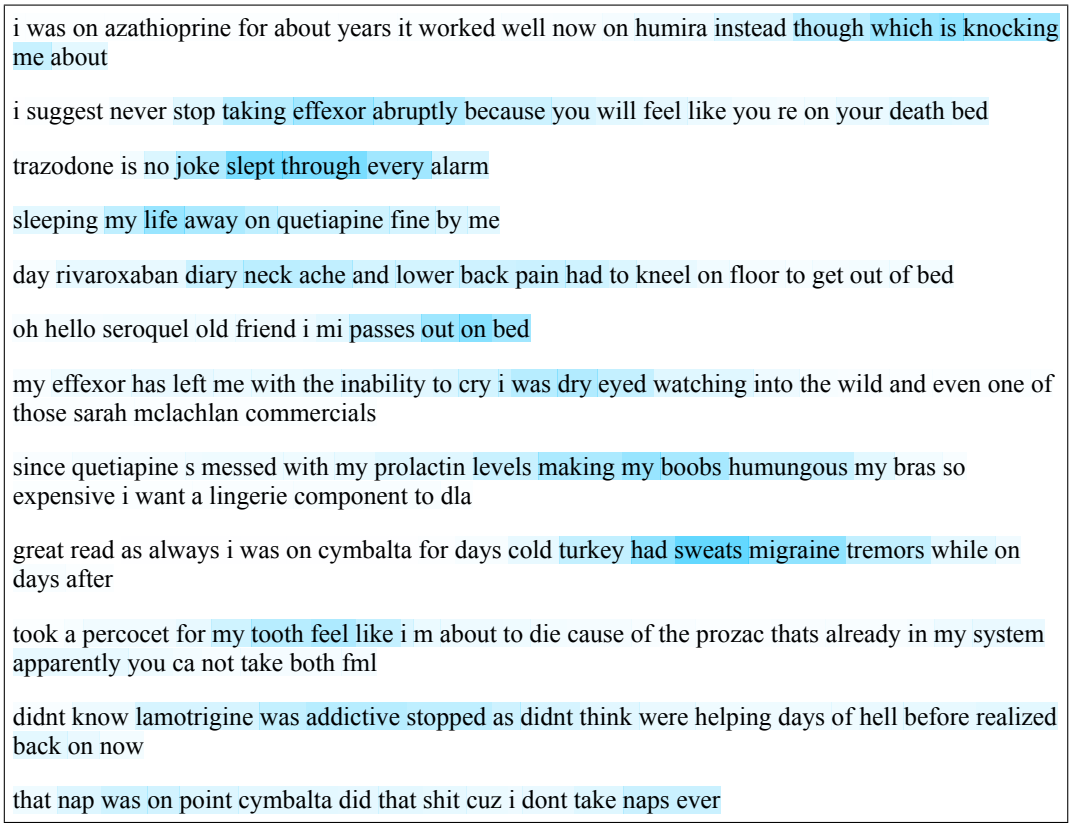


Figure 2: Sampled tweets with weighted highlights from attention weights.

describing ADRs. As such, CNNA would be a better candidate than CNN for more fine-grained ADR extraction.

6 Conclusion

This paper has explored different neural network (NN) architectures for ADR classification. In particular, it has proposed two new neural network models, Convolutional Recurrent Neural Network (CRNN) and Convolutional Neural Network with Attention (CNNA). Experimental results show that all the NN architectures outperform the traditional Maximum Entropy classifiers trained from n-grams with different weighting strategies considerably on both the Twitter and the ADE datasets. Among NN architectures, no significant differences were observed on the Twitter dataset. But CNN appears to perform better compared to other more complex CNN variants on the ADE dataset. Nevertheless, CNNA allows the visualisation of attention weights of words when making classification decisions and hence is more appropriate for the extraction of word subsequences describing ADRs.

Acknowledgements

YH is partly funded by the EPSRC AMR4AMR project (grant number EP/M02735X/1).

References

[Alex Krizhevsky et al.2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems (NIPS)*, pages 1097–1105.

[Aramaki et al.2010] Eiji Aramaki, Yasuhide Miura, Masatsugu Tonoike, Tomoko Ohkuma, Hiroshi Masuichi, Kayo Waki, and Kazuhiko Ohe. 2010. Extraction of adverse drug effects from clinical records. *Studies in Health Technology and Informatics*, 160 (PART 1):739–743.

[Bahdanau et al.2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *5th International Conference on Learning Representations (ICLR)*, pages 1–15.

[Benton et al.2011] Adrian Benton, Lyle Ungar, Shawndra Hill, Sean Hennessy, Jun Mao, Annie Chung, Charles E. Leonard, and John H. Holmes. 2011. Identifying potential adverse effects using the web: A new approach to medical hypothesis generation. *Journal of Biomedical Informatics*, 44(6):989–996.

[Cho et al.2014] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Dzmitry Bahdanau, Holger Schwenk, Yoshua Bengio, Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Empiricial Methods in Natural Language Processing (EMNLP)*.

[Collobert et al.2011] Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

[Friedman2009] Carol Friedman. 2009. Discovering Novel Adverse Drug Events Using Natural Language Processing and Mining of the Electronic Health Record. In *12th Conference on Artificial Intelligence in Medicine (AIME)*, pages 1–5.

[Ginn et al.2014] Rachel Ginn, Pranoti Pimpalkhute, Azadeh Nikfarjam, Apur Patki, Karen Oconnor, Abeed Sarker, Karen Smith, and Graciela Gonzalez. 2014. Mining Twitter for Adverse Drug Reaction Mentions: A Corpus and Classification Benchmark. In *proceedings of the 4th Workshop on Building and Evaluating Resources for Health and Biomedical Text Processing (BioTxtM)*.

[Glorot et al.2011] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. In *14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 315–323.

[Gurulingappa and Fluck2011] H Gurulingappa and J Fluck. 2011. Identification of adverse drug event assertive sentences in medical case reports. In *1st international workshop on knowledge discovery and health care management (KD-HCM) co-located at the European conference on machine learning and principles and practice of knowledge discovery in databases (ECML PKDD)*, pages 16-27.

[Gurulingappa et al.2012a] Harsha Gurulingappa, Abdul Mateen-Rajput, and Luca Toldo. 2012a. Extraction of potential adverse drug events from medical case reports. *Journal of biomedical semantics*, 3:15.

[Gurulingappa et al.2012b] Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012b. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of Biomedical Informatics*, 45(5):885–892.

[Harpaz et al.2012] R Harpaz, W DuMouchel, N H Shah, D Madigan, P Ryan, and C Friedman. 2012. Novel data-mining methodologies for adverse drug event discovery and analysis. *Clinical Pharmacology & Therapeutics*, 91(6):1010–1021.

[He et al.2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Rian Sun. 2015. Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385*.

[Hermann et al.2015] Karl Moritz Hermann, Tomáš Kočiský,, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman and Phil Blunsom. 2015. Teach machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

[Kalchbrenner et al.2014] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (AL)*.

[Kim2014] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

[Kuhn et al.2010] Michael Kuhn, Monica Campillos, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. 2010. A side effect resource to capture phenotypic effects of drugs. *Molecular systems biology*, 6(343):343.

[Leaman and Wojtulewicz2010] Robert Leaman and Laura Wojtulewicz. 2010. Towards internet-age pharmacovigilance: extracting adverse drug reactions from user posts to health-related social networks. In *Proceedings of the workshop on biomedical natural language processing (BioNLP)*, pages 117-125.

[Levy and Goldberg2014] Omer Levy and Yoav Goldberg. 2014. Neural Word Embedding as Implicit Matrix Factorization. *Neural Information Processing Systems (NIPS)*.

[Liu and Chen2013] Xiao Liu and Hsinchun Chen, 2013. AZDrugMiner: An Information Extraction System for Mining Patient-Reported Adverse Drug Events in Online Patient Forums. In *Proceedings of the International Conference on Smart Health (ICSH)*, pages 134–150.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *Neural Information Processing Systems (NIPS)*, pages 1–9.

[Nikfarjam and Gonzalez2011] Azadeh Nikfarjam and Graciela H Gonzalez. 2011. Pattern Mining for Extraction of mentions of Adverse Drug Reactions from User Comments. *AMIA Annual Symposium Proceedings*, 2011:1019–1026.

[Nikfarjam et al.2015] Azadeh Nikfarjam, Abeed Sarker, Karen O'Connor, Rachel Ginn, and Graciela Gonzalez. 2015. Pharmacovigilance from social media: Mining adverse drug reaction mentions using sequence labeling with word embedding cluster features. *Journal of the American Medical Informatics Association*, 22(3):671–681.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe : Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[Pirmohamed et al.2004] Munir Pirmohamed, Sally James, Shaun Meakin, Chris Green, Andrew K Scott, Thomas J Walley, Keith Farrar, B Kevin Park, and Alasdair M Breckenridge. 2004. Adverse drug reactions as cause of admission to hospital: prospective analysis of 18,820 patients. *BMJ*, 329(7456):15–19.

[Rastergar-Mojarad et al.2016] Majid Rastergar-Mojarad, Ravikumar Komandur Elayavilli, Yue Yu, and Hongfang Liu. 2016. Detecting signals in noisy data - can ensemble classifiers help identify adverse drug reaction in tweets? In *Proceedings of the Social Media Mining Shared Task Workshop at the Pacific Symposium on Biocomputing*.

[Rocktäschel et al.2016] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *International Conference in Learning Representation (ICLR)*.

[Rush et al.2015] Alexander M. Rush, Sumit Chopra, Jason Weston. 2015. A neural attention model for sentence summarization. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[Sarker and Gonzalez2015] Abeed Sarker and Graciela Gonzalez. 2015. Portable automatic text classification for adverse drug reaction detection via multi-corpus training. *Journal of Biomedical Informatics*, 53:196–207.

[Sarker et al.2016] Abeed Sarker, Azadeh Nikfarjam, and Graciela Gonzalez. 2016. In Proceedings of the Social Media Mining Shared Task Workshop at the Pacific Symposium on Biocomputing. pages 581–592.

[Shazeer et al.2016] Noam Shazeer, Ryan Doherty, Colin Evans, and Chris Waterson. 2016. Swivel: Improving Embeddings by Noticing What's Missing. *arXiv preprint arXiv:1602.02215*.

[Simonyan and Zisserman2014] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*.

[Szegedy et al.2014] C Szegedy, W Liu, Y Jia, and P Sermanet. 2014. Going deeper with convolutions. *arXiv preprint arXiv: 1409.4842*.

[Wang et al.2009] Xiaoyan Wang, George Hripcsak, Marianthi Markatou, and Carol Friedman. 2009. Active Computerized Pharmacovigilance Using Natural Language Processing, Statistics, and Electronic Health Records: A Feasibility Study. *Journal of the American Medical Informatics Association*, 16(3):328–337.

[Yang et al.2016] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical Attention Networks for Document Classification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

[Yates and Goharian2013] Andrew Yates and Nazli Goharian. 2013. ADRTrace: Detecting expected and unexpected adverse drug reactions from user reviews on social media sites. In *European Conference on Information Retrieval (ECIR)*, pages 816–819.

[Zeiler2012] Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*.

[Zhang et al.2016] Zhifei Zhang, Jian-yun Nie, and Xuyao Zhang. 2016. An Ensemble Method for Binary Classification of Adverse Drug Reactions From Social Media. In *Proceedings of the Social Media Mining Shared Task Workshop at the Pacific Symposium on Biocomputing*.

[Yin et al.2016] Wenpeng Yin, Sebastian Ebert, Hinrich Schütze. 2016. Attention-Based Convolutional Neural Network for Machine Comprehension. In *Proceedings of NAACL Human Computer QA Workshop*.

[Yin et al. 2016] Wenpeng Yin, Hinrich Schütze, Bing Xiang, Bowen Zhou. 2016. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of Associations for Computational Linguistics*.

[Zhou et al.2015] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. 2015. A C-LSTM Neural Network for Text Classification. *arXiv preprint arXiv:1511.08630*.

Chinese Preposition Selection for Grammatical Error Diagnosis

Hen-Hsen Huang, Yen-Chi Shao, and Hsin-Hsi Chen
Department of Computer Science and Information Engineering
National Taiwan University
No. 1, Sec. 4, Roosevelt Road, Taipei, 10617 Taiwan
{hhhuang, ycshao}@nlg.csie.ntu.edu.tw; hhchen@ntu.edu.tw

Abstract

Misuse of Chinese prepositions is one of common word usage errors in grammatical error diagnosis. In this paper, we adopt the Chinese Gigaword corpus and HSK corpus as L1 and L2 corpora, respectively. We explore gated recurrent neural network model (GRU), and an ensemble of GRU model and maximum entropy language model (GRU-ME) to select the best preposition from 43 candidates for each test sentence. The experimental results show the advantage of the GRU models over simple RNN and n-gram models. We further analyze the effectiveness of linguistic information such as word boundary and part-of-speech tag in this task.

1 Introduction

As learning Chinese has been popular world-wide, Chinese word spelling checking and grammatical error diagnosis for learners of Chinese language is recently advanced in the NLP community. In the NLP-TEA shared tasks (Yu et al., 2014; Lee et al., 2015a, Lee et al., 2015b), four types of grammatical errors including disorder, redundant, missing, and mis-selection are defined. These tasks focus on detecting and identifying grammatical errors, but not addressing error correction.

In English, selection of appropriate prepositions is a barrier to non-native language learners (Chodorow et al., 2007; Felica and Pulman, 2008). Many studies deal with the issue of preposition error detection and correction (Dale et al., 2012; Ng et al., 2013). The selection of Chinese prepositions is also challenging to non-native learners, especially for some common prepositions. (S1) and (S2) show a real case from the HSK corpus. The preposition 從 (cóng, "from") in (S2) is preferred to 在 (zài, "on/in/at") in (S1). In this paper, we investigate Chinese preposition selection.

(S1) **在** 公眾 利益 方面 來看 (**on** the view point of public interest)
(S2) **從** 公眾 利益 方面 來看 (**from** the view point of public interest)

A model trained on an error-annotated dataset benefits from learning the mapping between wrong instances and their corrected counterparts (Cahill et al., 2013). However, large error-annotated dataset for Chinese preposition error correction is still not available. In this paper, we utilize a large-scale corpus written by native Chinese speakers to deal with this problem. We adopt language models based on recurrent neural networks (RNNs) (Mikolov et al., 2011) to capture word dependencies in sentences. Cutting-edge methods like noise contrastive estimation (NCE) (Chen et al., 2015a) and gated recurrent unit (GRU) (Cho et al., 2014) are explored in RNNs. For the task of 43-way classification, i.e., to select the best preposition from 43 candidates, the gated recurrent neural network model (GRU) achieves an accuracy of 74.05% and an MRR of 83.08% on the Chinese Gigaword corpus (L1 corpus), and achieves an accuracy of 60.13% and an MRR of 72.54% on the HSK corpus (L2 corpus). An ensemble of gated

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,
pages 888–899, Osaka, Japan, December 11-17 2016.

recurrent neural network model and maximum entropy language model (GRU-ME) further improves the accuracy and the MRR of the GRU model on the Gigaword corpus to 76.71% and 84.89%.

The contribution of this paper is two-fold. From technological point of view, to the best of our knowledge, this paper is the first attempt to deal with Chinese preposition selection based on gated recurrent neural network model and an ensemble of GRU and ME. From linguistic point of view, it discusses the unique phenomena of Chinese prepositions with empirical evidence. The rest of this paper is organized as follows. Section 2 introduces the related work of grammatical error diagnosis in English and Chinese. We further introduce Chinese prepositions in Section 3. Section 4 shows L1 and L2 corpora used in this study. Section 5 presents our approach to Chinese preposition selection. Section 6 shows the experimental results. We further analyze the results in Section 7. Finally, Section 8 concludes this work.

2 Related Work

English preposition error detection has attracted much attention for years. Felice and Pulman (2007) proposed a voted perceptron classifier for disambiguating the uses of five common prepositions including "in", "of", "on", "to", and "with". In the work of Felice and Pulman (2008), error detection of nine common prepositions is tackled with the maximum entropy classifier. Chodorow et al. (2007) deal with the detection of preposition errors of non-native learners. In addition to error detection, some studies address the task of English preposition selection. Bergsma et al. (2009) propose a supervised language model for preposition correction. Tetreault et al. (2010) introduce parse features for this task. Cahill et al. (2013) propose a preposition error correction model trained on error-annotated data, and treated the revision logs of Wikipedia as a large error-annotated corpus. Xiang et al. (2013) propose a hybrid approach to deal with preposition selection. Zhang and Wang (2014) introduce a framework for English grammatical error correction using the maximum entropy language model for the replacement errors. Ramisa et al. (2015) address the task of preposition prediction for image descriptions with multimodal features. Related evaluations are covered in the shared tasks of HOO 2011 (Dale and Kilgarriff, 2011), HOO 2012 (Dale et al., 2012), CoNLL 2013 (Ng et al., 2013) and CoNLL 2014 (Ng et al., 2014).

In addition to Chinese spelling checking (Lee et al., 2015b), grammatical error detection in Chinese has been investigated recently. Wang (2011) shows common Chinese grammatical errors like missing components and error word orderings. Lin (2011), Yu and Chen (2012), and Cheng et al. (2014) focus on the detection and correction of word ordering errors in Chinese written by foreign students in the HSK corpus. Shiue and Chen (2016) determine if a Chinese sentence contains word usage errors.

In the NLP-TEA shared tasks (Yu et al., 2014; Lee et al., 2015a), detection of four grammatical errors are targeted. Lin and Chan (2014) train SVM classifiers with various bigram features. Zampiperi and Tan (2014) propose a frequency-based approach based on a large general corpus. Zhao et al. (2014; 2015) model the task of correction as machine translation in such a way that the wrong sentences are translated to correct ones. The preposition error detection is one of error cases in NLP-TEA shared tasks. However, the preposition correction is beyond the scope of NLP-TEA.

Different from previous works, our work focuses on the correction of Chinese prepositions. We aim at selecting suitable prepositions from a set of 43 common prepositions. To overcome the limitation of error-annotated dataset, we propose an unsupervised approach based on language models, which can be trained on a large scale L1 corpus without the need of annotation.

3 Chinese Prepositions

A preposition is a function word that is followed by a noun phrase to introduce a preposition phrase (PP). It indicates a relation between the noun phrase and other words within the sentence. For example, "in", "on", "to", "with", and "of" are most common prepositions in English. In the Penn Treebank 3 corpus (Marcus et al., 1999), 186 distinct English words are tagged with the part-of-speech (POS) tag "P" (i.e. preposition). In the 186 prepositions, some of them are abbreviations such as "altho" (although) and "w." (with).

In the same manner, we found a total of 288 distinct Chinese prepositions in the Chinese Treebank 8.0 corpus (Xue et al., 2013). The total occurrences of them are 54,239 in 71,369 sentences. In the 288 prepositions, 199 of them appear more than once, and 44 of them appear more than 100 times. The top three most frequent prepositions are 在 (zài, "on/in/at"), 對 (duì, "to/at"), and 從 (cóng, "from").

The CKIP group (1993) categorized Chinese prepositions into 66 types of senses[1]. For example, these words 直到 (zhídào), 迄 (qì), 等到 (děng dào), 比及 (bǐ jí), 及至 (jízhì), and 待到 (dài dào) share the same meaning of "until" when they are used as preposition. Note that some prepositions have other usages. The words 與 (yǔ) and 和 (hàn) can be used as preposition with the meaning of "with", while they are also common conjunctions with the meaning of "and". The word 對 (duì) can be used as preposition (to/at), noun (a pair of), adjective (right/correct), adverb (yes), and verb (be opposite/reply) with various senses. Another highly ambiguous word 給 (gěi) can be used as preposition (to/for/with), verb (give/let), and emphatic particle. Both 對 (duì) and 給 (gěi) are common Chinese words, they have multiple senses by their own, and they are interchangeable in some situations.

Some preposition/noun pairs usually collocate. For example, the combination of the noun 辦公室 (office) and the preposition 在 (zài, "in") forms a common preposition phrase 在辦公室 (in the office). In a complicated sentence like (S3), there are eight words in-between 在 and 辦公室. Such a long distance dependency is challenging to language models.

(S3) 我 **在** 設於 上海 浦東 機場 西面 的 口岸 服務 **辦公室** (I am **in** the port service **office** located in the west of the Pudong airport, Shanghai)

In addition, the combination of the preposition and the noun varies according to semantics. For the noun 辦公室 (office) in (S4) and (S5), different prepositions are used. The ambiguity of preposition selection not only causes confusion to non-native learners, but also makes challenges in natural language processing.

(S4) 我 **在** 辦公室 上班 (I work **in** the office)
(S5) 我 **從** 辦公室 出發 (I depart **from** the office)

4 Datasets

We adopt the Tagged Chinese Gigaword (CGW) corpus 2.0[2] (Huang, 2009; Huang et al., 2008) as the L1 corpus. It contains 2,803,632 documents and 831,748,000 words with part-of-speech (POS) tags. By removing the sentences without prepositions, a total of 23,486,882 sentences covering 155 prepositions are collected from the Gigaword corpus. We randomly select 60% of sentences for training models due to the computation limitations. Additional 5,000 and 200,000 sentences are randomly selected as development data and test data, respectively. The development data is used for validation.

HSK dynamic composition corpus is adopted as the L2 corpus[3]. It collects articles written by students from foreign countries to study Chinese in Beijing Language and Culture University. Total 46 error categories range from character level, word level, sentence level, to discourse level are annotated and corrected in the HSK corpus. The CKIP segmentation system is used to perform Chinese word segmentation and POS tagging on the sentences in HSK because the POS tagging in the CGW corpus follows the CKIP style.

From the HSK corpus, total 745 sentences consisting of preposition errors are extracted from word-level grammatical error set. Limited to the small L2 dataset, these 745 sentences are used as test data only. The statistics of L1 and L2 datasets are listed in Table 1.

Dataset	Source	Number of Sentences
Training set (L1)	CGW	14,092,128
Development set (L1)	CGW	5,000
Test set (L1)	CGW	200,000
Test set (L2)	HSK	745

Table 1: Statistics of the L1 (CGW) and L2 (HSK) datasets used in experiments.

[1] http://rocling.iis.sinica.edu.tw/CKIP/tr/9305_2013%20revision.pdf
[2] https://catalog.ldc.upenn.edu/LDC2009T14
[3] http://202.112.195.192:8060/hsk/login.asp

Among the Chinese prepositions mentioned in Section 3, 43 prepositions, most of which are common words, appear in the HSK dataset. Note that 33 of them belong to the list of the top 50 prepositions in CGW. Furthermore, these 43 prepositions cover 88.61% of preposition uses in CGW. Figure 1 lists the 43 prepositions used in this work by the order of their occurrences.

在 (on/in/at) 對 (to/at) 從 (from) 用 (with/using) 以 (with/by) 跟 (with) 由 (by/from) 給 (to/for/with) 為 (for) 和 (to) 向 (to/toward) 與 (to) 像 (like) 對於 (for/regarding) 當 (at) 把 (owing to) 到 (to) 靠 (by) 於 (in/to/on/for/at/of) 被 (passive particle) 關於 (about/on) 隨著 (along) 比 (than) 根據 (according to/based on) 如 (as) 依 (according to/by) 就 (on) 針對 (against) 離 (from) 按照 (according to/by) 替 (for) 至於 (touching) 受 (passive particle) 至 (to/until) 等到 (until) 據 (according to) 按 (according to/by) 往 (to/toward) 經由 (via) 同 (with) 憑 (by) 趁 (at) 正當 (at)

Figure 1: Prepositions in the HSK corpus.

5 Methods

This work deals with the preposition selection error. Given a set of Chinese prepositions, *PS*, and a sentence $S = (x_1, x_2, ..., x_{i-1}, x_i, x_{i+1}, ..., x_n)$, where x_i is a preposition, we try to substitute x_i for each preposition p in *PS*, and choose the most probable preposition $\hat{p}$.

$$\hat{p} = \underset{p \in PS}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_{i-1}, p, x_{i+1}, \ldots, x_n)$$

Prepositions are a closed set of function words. Thus, it is feasible to substitute each of all prepositions for the location of a preposition. As listed in Section 4, *PS* contains 43 common prepositions which appear in both the CGW corpus and the HSK corpus. The probability of a sentence is estimated by a language model. The traditional n-gram language model is considered as the baseline in this work. The SRI language modeling toolkit (SRILM)[4], an implementation of n-gram language model, is adopted. The major disadvantage of n-gram is that the number of its parameters growths exponentially as the order of n-gram increases. As a result, the order of n-gram usually ranges between bigram and 5-gram. The n-gram model with high order is not impractical.

Recently, language models based on recurrent neural networks (RNNs) such as simple RNN, long short-term memory (LSTM), and gated recurrent unit (GRU) (Mikolov et al., 2011; Hochreiter and Jürgen Schmidhuber, 1997; Cho et al., 2014) have been shown to outperform traditional approaches in speech recognition and other applications. Figure 2 illustrates a basic recurrent neural network. The hidden state s_t is passed to the next step for the following update.

$$s_t = f(Wx_t + Us_{t-1})$$

where *f* is an activation function such as a sigmoid function or a tanh function, and *W* and *U* are parameters to be learned. In other words, the history information is kept.

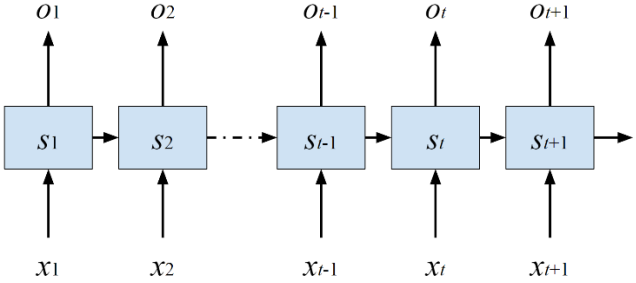


Figure 2: Recurrent neural network (RNN).

[4] http://www.speech.sri.com/projects/srilm/

Compared to n-gram, RNN based language model can capture longer distance dependencies with less parameters. As mentioned in Section 3, long distance dependency modelling is crucial to preposition selection. For this reason, we employ the simple RNN and the GRU language models to estimate the probability of a preposition in a given sentence. Instead of the single sigmoid or tanh function used by the simple RNN, GRU model, which is simplified from LSTM, uses a structure namely gated recurrent unit in the hidden layer.

$$s_t = (1 - z)h + zs_{t-1}$$
$$h = \tanh(W_h x_t + r_t(U_h s_{t-1}))$$
$$z = \sigma(W_z x_t + U_z s_{t-1})$$
$$r = \sigma(W_r x_t + U_r s_{t-1})$$

As shown in Figure 3, the update gate *z* decides how much the hidden state *s* is updated with the candidate hidden state $\tilde{s}$, while the reset gate *r* decides how much the memory to be forgotten. GRU is reported to be better for long-term dependency modeling (Chung et al., 2014; Chen et al., 2015b) than the simple RNN. Compared to LSTM, GRU requires few parameters for the same size of hidden layer.

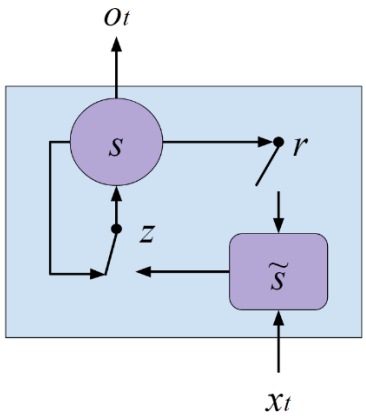


Figure 3: Gated recurrent unit (GRU).

In this work, we use the noise contrastive estimation (NCE) (Chen et al., 2015a) as the output layer for both simple RNN and GRU language models. The training performance of NCE is comparable to the class-based softmax, but NCE is faster in testing stage and can speed up together with GPU in training stage. The implementation of RNN models is based on faster-rnnlm, a toolkit for RNN language modelling[5].

6 Experiments

Three language models, n-gram, simple RNN, and GRU, are evaluated in the experiments with various configurations. The n-gram models are trained with the orders from 2 to 12. The simple RNN and the GRU models are trained with the hidden sizes of 128, 256, and 512. The number of noise samples of NCE is set to 20. Moreover, all language models are trained on three linguistic levels:

(1) Character level (**char**): each unit is a Chinese character.
(2) Word level (**word**): each unit is a Chinese word.
(3) Word level with POS tag (**w/p**): each unit is a combination of a word and its POS tag.

The performance is measured by accuracy and mean reciprocal rank (MRR). The accuracy is defined as number of correct selection versus number all test instances. The MRR is defined as $\frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{rank}_i}$ where *N* is number of instances, and rank_i is the rank of the correct preposition among *PS* according to its probability. In L1 testing, we check if the predicted preposition is the same as the original preposition in the sentence. In L2 testing, we check if the predicted preposition is the one corrected by annotators. McNemar test is used for significance testing at p=0.05.

[5] https://github.com/yandex/faster-rnnlm

Experimental results in L1 and L2 are shown in Table 2. For the n-gram models with different orders (from 2 to 12), we only show the highest performances due to the limited space. The subscripts of the simple RNN and the GRU models denote their hidden sizes. The best performing configuration for each of n-gram, simple RNN, and GRU models is highlighted in bold. In general, the larger the hidden layer, the better the performances of the simple RNN and the GRU language models. GRU model outperforms simple RNN and n-gram models in most cases. The best n-gram model is trained on word level with POS tags, the best simple RNN model is trained on word level using a hidden size of 512, and the best GRU model, which is also the best model of all, is trained on word-level with POS tag using a hidden size of 512. In both L1 and L2 testing, the best GRU model significantly outperforms the best simple RNN model.

The n-gram model and the GRU model perform better on the word level with POS tag, while the simple RNN model performs better on the word-level. On the other hand, all the n-gram, simple RNN, and GRU models trained on character level perform poorly. Grouping characters into words may help language models to capture long distance dependency. The neural networks are capable of learning feature representation from raw data. The linguistic information like POS tags still increases the performances of the GRU model.

The best performing model, GRU_{512} trained on word level with POS tag, achieves an accuracy of 74.05% and an MRR of 83.08% on L1, and an accuracy of 60.13% and an MRR of 72.54% on L2. The precision@3 is 91% in L1 testing, and 81% in L2 testing. For a task of 43-way classification, this result is promising.

Among the three best performing n-gram, simple RNN and GRU models, the smallest performance gap between L1 and L2 is found in the GRU model. Compared to the other two models, GRU model achieves better testing fitness and less overfitting. However, the performance gap between L1 and L2 is still an issue to be tackled in the future.

LM	CGW (L1)						HSK (L2)					
	Character		Word		Word/POS		Character		Word		Word/POS	
	ACC	MRR	ACC	MRR	ACC	MRR	ACC	MRR	ACC	MRR	ACC	MRR
N-gram	66.90	76.54	66.78	76.49	**69.11**	**78.68**	40.54	54.91	42.42	55.73	**43.62**	**56.65**
RNN_{128}	33.47	48.34	67.39	77.50	67.14	77.46	25.91	40.67	50.47	63.26	49.13	63.12
RNN_{256}	42.87	57.98	68.61	78.64	67.25	77.65	25.64	42.10	50.74	63.95	49.80	63.12
RNN_{512}	51.44	64.55	**71.80**	**81.03**	71.04	80.40	35.57	49.60	**55.97**	**68.56**	50.34	63.84
GRU_{128}	45.11	58.57	63.78	74.91	68.09	78.41	27.11	41.77	48.59	62.20	51.14	64.91
GRU_{256}	49.94	63.35	70.24	79.89	72.05	81.48	34.50	49.03	55.97	68.46	57.99	70.27
GRU_{512}	55.77	68.56	72.42	81.71	**74.05**	**83.08**	40.94	55.56	56.78	69.92	**60.13**	**72.54**

Table 2: Accuracies of n-gram, simple RNN, and GRU models with different configurations in L1 and L2. All the numbers are shown in percentage (%).

7 Discussion

Table 3 shows the performances of most frequent prepositions by the best performing n-gram, simple RNN and GRU models in L2 testing. The last row represents the performances of all prepositions in micro average. In each row, the best precision (P), recall (R), and F-score (F) are highlighted. The confusion matrix of the best performing GRU model in L2 testing is shown in Table 4. Most frequent prepositions are listed. Each row represents the sample in an actual preposition, while each column of the matrix represents the samples in a predicted preposition.

The preposition 在 (zài), the most frequent Chinese preposition, covers the meanings of *on*, *in*, and *at* in English. The precision of the second preposition 對 (duì, "to/at") achieved by the GRU model is only 59.76%. As shown in Table 4, 18 instances of 在 (zài, "on/in/at") and 10 instances of 給 (gěi, "for/to") are misclassified to 對. As a result, the recall of 給 (gěi, "for/to/with") and the precision of 對 are poor. In fact, they are sometimes interchangeable. For instance, using 對 in place of 給 in (S6) is also correct. However, only one correct preposition is labeled in the HSK corpus. In other words, our models are under-estimated due to the one-answer evaluation. In English, prepositions are also reportedly more than one way to correct (Bryant and Ng, 2015).

(S6) 吸菸 **給** 人 們 的 健康 帶來 不好 的 影響 (smoking causes bad effect **to** human health)

The third frequent preposition 從 (cóng, "from") tends to confuse with 在 (zài, "on/in/at") and 以 (yǐ, "by/with"). The fourth and the fifth prepositions 用 (yòng, "by/with") and 以 (yǐ, "by/with") share similar meanings and tend to be confusing.

Preposition	N-Gram Word/POS			RNN_{512} Word			GRU_{512} Word/POS		
	P	R	F	P	R	F	P	R	F
在 (on/in/at)	52.22	69.86	59.77	68.66	**84.02**	75.56	**79.70**	73.52	**76.48**
對 (to/at)	50.66	61.11	55.40	**63.70**	73.81	68.38	59.76	**80.16**	**68.47**
從 (from)	70.59	26.09	38.10	**87.18**	36.96	51.91	74.70	**67.39**	**70.86**
用 (by/with)	63.16	32.43	42.86	71.43	27.03	39.22	**77.78**	**37.84**	**50.91**
以 (by/with)	27.59	25.00	26.23	34.55	59.38	43.68	**41.51**	**68.75**	**51.76**
跟 (with)	**60.00**	19.35	29.27	50.00	**38.71**	**43.64**	**60.00**	29.03	39.13
由 (by/from)	33.33	31.82	32.56	54.55	54.55	54.55	**72.22**	**59.09**	**65.00**
給 (for/to/with)	50.00	5.88	10.53	**100.00**	5.88	11.11	**100.00**	**17.65**	**30.00**
Average of all	46.14	43.62	41.71	60.15	56.78	55.88	**63.63**	**60.13**	**59.20**

Table 3: Performances of most frequent prepositions by the best performing n-gram, simple RNN and GRU models in L2 testing. All numbers are shown in percentage (%).

Actual	Predicted Prepositions							
	在 (on/in/at)	對 (to/at)	從 (from)	用 (by/with)	以 (by/with)	跟 (with)	由 (by/from)	給 (for/to/with)
在 (on/in/at)	161	18	7	1	1	1	0	0
對 (to/at)	8	101	3	0	4	1	1	0
從 (from)	10	3	62	1	9	1	0	0
用 (by/with)	2	3	2	14	9	0	0	0
以 (by/with)	0	4	2	0	22	1	0	0
跟 (with)	2	3	0	0	0	9	0	0
由 (by/from)	1	1	0	2	1	0	13	0
給 (for/to/with)	0	10	0	0	0	0	0	3

Table 4: Confusion matrix of the GRU_{512} on the word level with POS tag in L2 testing.

Figure 4 shows the accuracies of the n-gram, RNN_{512}, and GRU_{512} models on word level with and without POS tag with respect to difference sentence lengths. We divide the sentences in L2 test set into five groups according to their length. The average sentence length is 9.13 words, and the longest sentence consists of 32 words. The information of POS tag generally improves the performance for the cases of longer sentences. In particular, the GRU_{512} model on word-level with POS tag outperforms other models for the sentences of lengths ≥ 4 words. In the test set of L2, only 10 sentences (1.3%) are shorter than 4 words.

(S7) is an instance correctly predicted by the GRU_{512} model, while n-gram and RNN_{512} output a wrong outcome as (S8). In this case, the first word 在 (zài, "in") and the last word 裡 (lǐ, "inside") form a circumposition for the noun phrase 那些保守的家庭 (those conservative family). The pair of words 在... 裡 (inside) is a common usage in Chinese. The GRU_{512} model successfully captures their dependency although there are four words in-between.

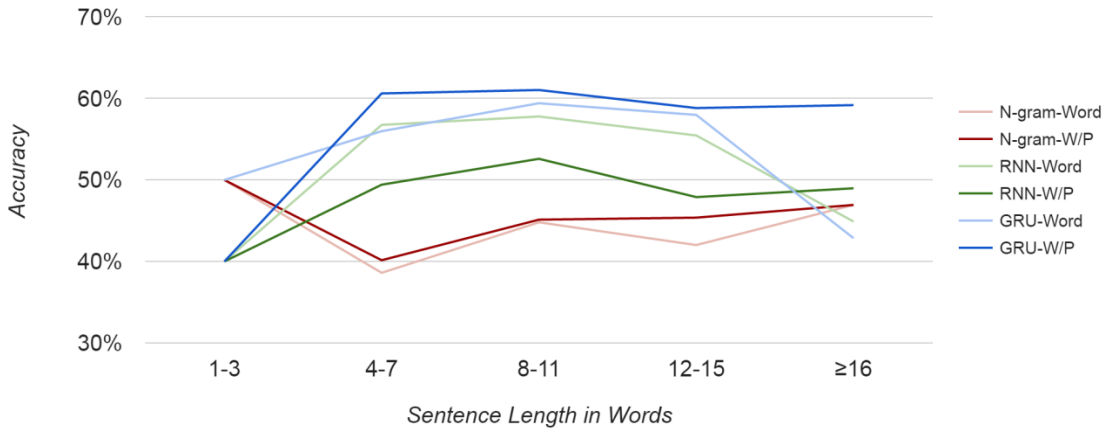


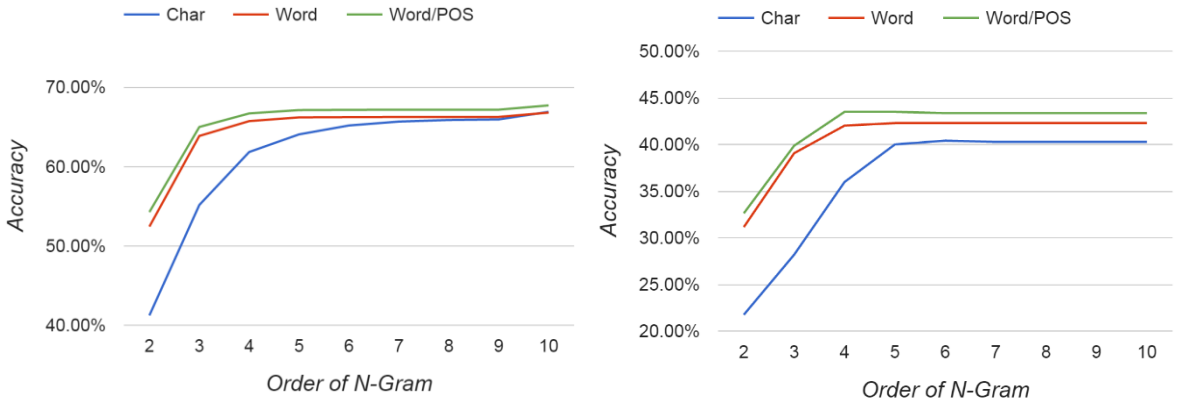
Figure 4: Accuracy versus sentence length in L2 testing.

(S7) **在** 那些 保守 的 家庭 裡 (Inside those conservative family)
(S8) **對** 那些 保守 的 家庭 裡 (To those conservative family)

(S9) shows a much longer distance dependency successfully estimated by GRU_{512}. In this case, there are eight words in-between the word pair 以...為 (aim at ... as). The RNN_{512} selects 就 (jiù, "just/on") in the sentence (S10), which is not fluent. The n-gram model, even worse, selects 在 (zài, "in") and makes an incomprehensible sentence (S11).

(S9) 一些 家長 也 **以** 把 孩子 送入 純 女 或 純 男校 為 第一 選擇 (Some parents aim at sending their children into the pure female or pure male school as the first choice)
(S10) 一些 家長 也 **就** 把 孩子 送入 純 女 或 純 男校 為 第一 選擇
(S11) 一些 家長 也 **在** 把 孩子 送入 純 女 或 純 男校 為 第一 選擇

Figure 5 illustrates accuracies of varying order of the n-gram on character level, word level, and word level with POS tag. Results of L1 and L2 testing are shown in Figure 5(a) and Figure 5(b), respectively. As the order of n-gram increases, the models on word level and on word level with POS tag faster growth. In L1 testing, the n-gram model on character level finally achieves an accuracy close those of other two models on word level. In L2 testing, the performance gaps among the three models are more apparent. The information of Chinese word segmentation and POS tags not only speeds up training, but also improves the generalization.



(a) L1 Testing (b) L2 Testing

Figure 5: Accuracy versus order of n-gram in L1 and L2 testing.

Previous work suggests that the ensemble of RNN and traditional language model may improve the performance, especially for the RNN models with small hidden layer (Mikolov, 2012; Mikolov et al., 2011). We build an ensemble model GRU-ME by joint training the GRU model with a 4-gram maximum entropy language model (Berger et al., 1996). The input unit is word with POS tag (w/p), which has best performance in the experiments. The feature size of maximum entropy model is 1 billion. Figure 6 compares the performances between GRU and GRU-ME in L1 and L2. In L1 testing, ensembling the maximum entropy model with GRU significantly increases the performances of three GRU models, especially the ones with smaller hidden layer. In L2 testing, ensembling the maximum entropy model increases the performances of the GRU model with hidden sizes of 128. The results confirm that the RNN models with smaller hidden size gain from the combination of the traditional language model. The performances of the GRU models with larger hidden layer, however, are decreased with the combination of maximum entropy model and GRU. This phenomenon suggests that GRU-ME better fits the training data, but may suffer from overfitting. In contrast, the GRU model with a large hidden size is more robust when it is applied to another corpus.

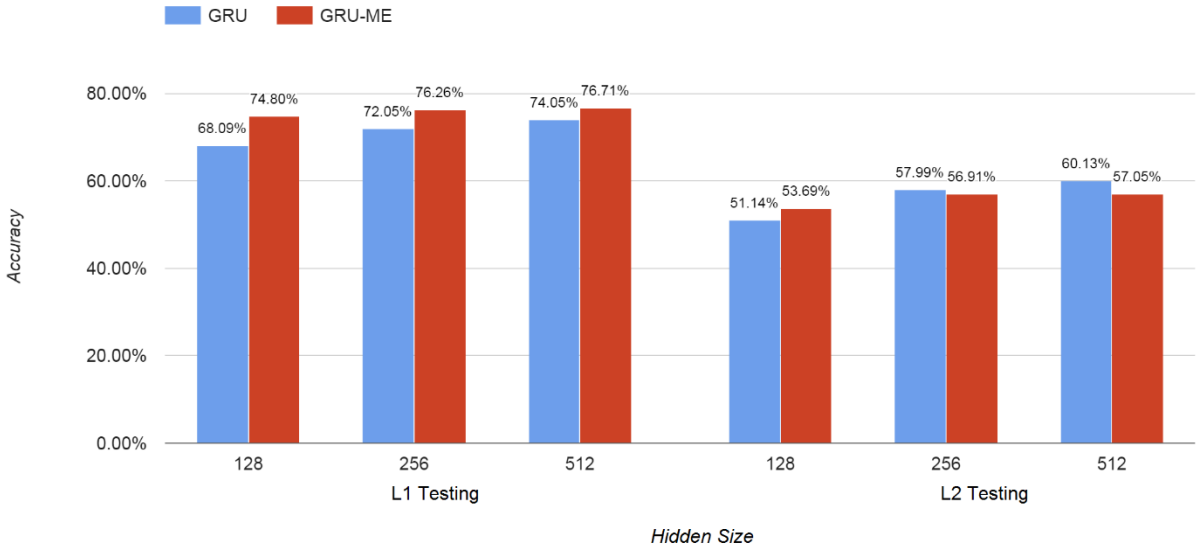


Figure 6: Performances of the GRU and the ensemble of GRU with Maxent (GRU-ME) models in L1 and L2 testing.

8 Conclusion

This work addresses the issue of Chinese preposition selection. We propose a method that uses language models to predict the most probable preposition in a given context. The classical n-gram models and the recurrent neural network models are explored. For the task of modelling Chinese prepositions, the experimental results show the advantage of the GRU models over simple RNN and n-gram models, especially for the cases involving longer distance dependency. In addition, linguistic information from Chinese word segmentation and the POS tagging improve the performances of n-gram and neural network language models. We will further adapt this approach to detection and correction for other grammatical errors in future work.

9 Acknowledgements

This research was partially supported by Ministry of Science and Technology, Taiwan, under grants MOST-104-2221-E-002-061-MY3 and MOST-105-2221-E-002-154-MY3, and National Taiwan University under grant NTU-ERP-104R890858.

References

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):1-36.

Shane Bergsma, Dekang Lin, and Randy Goebel. 2009. Web-Scale N-gram Models for Lexical Disambiguation. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence* (*IJCAI-09*), pages 1507-1512.

Christopher Bryant and Hwee Tou Ng. 2015. How Far are We from Fully Automatic High Quality Grammatical Error Correction? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (*ACL-IJCNLP*), pages 697–707, Beijing, China.

Aoife Cahill, Nitin Madnani, Joel Tetreault, and Diane Napolitano. 2013. Robust Systems for Preposition Error Correction Using Wikipedia Revisions. In *Proceedings of NAACL-HLT 2013*, pages 507–517.

Xie Chen, Xunying Liu, Mark J. F. Gales, and Philip C. Woodland. 2015a. Recurrent Neural Network Language Model Training with Noise Contrastive Estimation for Speech Recognition. In *Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5411-5415, South Brisbane, Australia.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015b. Gated Recursive Neural Network for Chinese Word Segmentation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1744–1753, Beijing, China.

Shuk-Man Cheng, Chi-Hsin Yu, and Hsin-Hsi Chen. 2014. Chinese Word Ordering Errors Detection and Correction for Non-Native Chinese Language Learners. In *Proceedings of the 25th International Conference on Computational Linguistics* (*COLING*), pages 23-29, August 2014, Dublin, Ireland.

Chinese Knowledge Information Processing Group (CKIP). 1993. Technical Report 93-05: Chinese Part-of-Speech Analysis. Academia Sinica, Taipei.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. arXiv preprint arXiv:1406.1078.

Martin Chodorow, Joel R. Tetreault, and Na-Rae Han. 2007. Detection of Grammatical Errors Involving Prepositions. In *Proceedings of the 4th ACL-SIGSEM Workshop on Prepositions*, pages 25–30, Prague, Czech Republic.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv preprint arXiv:1412.3555v1

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A Report on the Preposition and Determiner Error Correction Shared Task. In *Proceedings of the 7th Workshop on the Innovative Use of NLP for Building Educational Applications*, pages 54–62, Montreal, Canada.

Robert Dale and Adam Kilgarriff. 2011. Helping our own: The HOO 2011 Pilot Shared Task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, Nancy, France.

Rachele De Felice and Stephen G. Pulman. 2007. Automatically Acquiring models of Preposition Use. In *Proceedings of the 4th ACL-SIGSEM Workshop on Prepositions*, pages 45–50, Prague, Czech Republic.

Rachele De Felice and Stephen G. Pulman. 2008. A Classifier-based Approach to Preposition and Determiner Error Correction in L2 English. In *Proceedings of the 22nd International Conference on Computational Linguistics* (*COLING*), pages 169–176.

Chu-Ren Huang, Lung-Hao Lee, Jia-Fei Hog, Wei-guang Qu, and Shiwen Yu. 2008. Quality Assurance of Automatic Annotation of Very Large Corpora: a Study based on heterogeneous Tagging System. *In Proceedings of the Sixth International Conference on Language Resources and Evaluation* (*LREC'08*), pages 2725–2729.

Chu-Ren, Huang. 2009. *Tagged Chinese Gigaword Version 2.0 LDC2009T14*. Web Download. Philadelphia: Linguistic Data Consortium.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780.

Lung-Hao Lee, Liang-Chih Yu, and Li-Ping Chang. 2015a. Overview of the NLP-TEA 2015 Shared Task for Chinese Grammatical Error Diagnosis. In *Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications* (*NLP-TEA*), pages 1–6, Beijing, China.

Lung-Hao Lee, Gina-Anne Levow, Shih-Hung Wu, and Chao-Lin Liu. 2015b. Introduction to the Special Issue on Chinese Spell Checking. *ACM Transactions on Asian and Low-Resource Language Information Processing* (*Special Issue on Chinese Spell Checking*), 14(4):14.

Jia-Na Lin. 2011. *Analysis on the Biased Errors of Word Order in Written Expression of Foreign Students*. Master Thesis. Soochow University.

Chuan-Jie Lin and Shao-Heng Chan. 2014. Description of NTOU Chinese Grammar Checker in CFL 2014. In *Proceedings of the 22nd International Conference on Computers in Education*, pages 75–78, Nara, Japan.

Mitchell Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. *Penn Treebank 3 LDC99T42*. Web Download. Philadelphia: Linguistic Data Consortium.

Tomas Mikolov. 2012. *Statistical Language Models Based on Neural Networks.* Doctoral Dissertation. Brno University of Technology.

Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukas Burget, and Jan "Honza" Cernocky. 2011. RNNLM – Recurrent Neural Network Language Modeling Toolkit. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop.*

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 Shared Task on Grammatical Error Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria.

Arnau Ramisa, Josiah Wang, Ying Lu, Emmanuel Dellandrea, Francesc Moreno-Noguer, and Robert Gaizauskas. 2015. Combining Geometric, Textual and Visual Features for Predicting Prepositions in Image Descriptions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), pages 214–220, Lisbon, Portugal.

Yow-Ting Shiue and Hsin-Hsi Chen. 2016. Detecting Word Usage Errors in Chinese Sentences for Learning Chinese as a Foreign Language. In *Proceedings of 10th Language Resources and Evaluation Conference*, pages 220-224, Portorož, Slovenia.

Joel Tetreault, Jennifer Foster, and Martin Chodorow. 2010. Using Parse Features for Preposition Selection and Error Detection. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 353–358, Uppsala, Sweden.

Zhuo Wang. 2011. *A Study on the Teaching of Unique Syntactic Pattern in Modern Chinese for Native English Speaking Students*. Master Thesis. Northeast Normal University.

Yang Xiang, Bo Yuan, Yaoyun Zhang, Xiaolong Wang, Wen Zheng, and Chongqiang Wei. 2013. A Hybrid Model for Grammatical Error Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning* (*CONLL)*: *Shared Task*, pages 115–122, Sofia, Bulgaria.

Nianwen Xue, Xiuhong Zhang, Zixin Jiang, Martha Palmer, Fei Xia, Fu-Dong Chiou, and Meiyu Chang. 2013. *Chinese Treebank 8.0 LDC2013T21*. Web Download. Philadelphia: Linguistic Data Consortium.

Chi-Hsin Yu and Hsin-Hsi Chen. 2012. Detecting Word Ordering Errors in Chinese Sentences for Learning Chinese as a Foreign Language. In *Proceedings of COLING 2012: Technical Papers*, pages 3003–3018, COLING 2012, Mumbai, India.

Liang-Chih Yu, Lung-Hao Lee, and Li-Ping Chang. 2014. Overview of Grammatical Error Diagnosis for Learning Chinese as a Foreign Language. In *Proceedings of the 22nd International Conference on Computers in Education*, pages 42-47.

Marcos Zampieri and Liling Tan. 2014. Grammatical Error Detection with Limited Training Data: The Case of Chinese. In In *Proceedings of the 22nd International Conference on Computers in Education*, pages 69-74, Nara, Japan.

Longkai Zhang and Houfeng Wang. 2014. A Unified Framework for Grammar Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 96–102, Baltimore, Maryland.

Yinchen Zhao, Mamoru Komachi, and Hiroshi Ishikawa. 2014. Extracting a Chinese Learner Corpus from the Web: Grammatical Error Correction for Learning Chinese as a Foreign Language with Statistical Machine

Translation. In *Proceedings of the 22nd International Conference on Computers in Education*, pages 56–61, Nara, Japan.

Yinchen Zhao, Mamoru Komachi, and Hiroshi Ishikawa. 2015. Improving Chinese Grammatical Error Correction using Corpus Augmentation and Hierarchical Phrase-based Statistical Machine Translation. In *Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications*, pages 111–116, Beijing, China.

Extending the Use of Adaptor Grammars for Unsupervised Morphological Segmentation of Unseen Languages

Ramy Eskander* **Owen Rambow*** **Tianchun Yang†**
* Center for Computational Learning Systems
†Department of Computer Science
Columbia University
New York, NY, USA
{rnd2110@,rambow@ccls.,ty2313}columbia.edu

Abstract

We investigate using Adaptor Grammars for unsupervised morphological segmentation. Using six development languages, we investigate in detail different grammars, the use of morphological knowledge from outside sources, and the use of a cascaded architecture. Using cross-validation on our development languages, we propose a system which is language-independent. We show that it outperforms two state-of-the-art systems on 5 out of 6 languages.

1 Introduction

Morphological segmentation, the splitting of words into smaller units (morphs), is an important sub-task in several natural language processing (NLP) applications. With the increasing interest in NLP for low-resource languages, unsupervised morphological segmentation becomes a crucial pre-processing step to reduce data sparseness: instead of working on a large vocabulary of plausible words, a smaller set of smaller word units is processed.

A well-known toolkit used for unsupervised segmentation is Morfessor (Creutz and Lagus, 2007), which is a generative probabilistic model. In competition, Adaptor Grammars (AGs) (Johnson et al., 2007) represent a framework for specifying compositional nonparametric Bayesian models and are applied in unsupervised segmentation with notable success (Johnson, 2008).

AGs generalize probabilistic context-free grammars by allowing some nonterminals to be "adapted, which allows for dependencies between applications of these rules. Sirts and Goldwater (2013) present an in-depth investigation of the use of AGs. They make two important contributions. First, they discuss the effect of the underlying grammar on the results of unsupervised morphological segmentation. Second, they investigate two ways of using a small amount of annotated data during training. They show that while for English, Morfessor remains the top performing system, on three other languages their approach can beat the high Morfessor baseline.

A typical application for unsupervised morphological segmentation involves situations in which we are confronted with a low-resource language for which no prior NLP work exists, and for which we cannot annotate even a small corpus (either for lack of time, or because no annotators are available). Therefore, in this paper, we are interested in remaining language-independent and entirely unsupervised. We make the following contributions:

- We follow the insight of Sirts and Goldwater (2013) that the underlying grammar in an unsupervised AG approach matters. We explore a much larger set of grammars. We show that for most languages we develop on, the grammars we propose in this paper allow for the best AG results to date. The grammars are discussed in Section 4.

- We explore the use of "scholar-seeded knowledge". Here, instead of annotating even a small set with the desired result of the machine learning process (a segmentation), we search the web for easily accessible information about affixes in our language of interest, and explicitly include these in the grammar used in the AG approach (before learning happens). This can be done in a few hours by a scholar who has never studied the language. We show that generally scholar-seeded knowledge increases the performance. We discuss scholar-seeded knowledge in Section 5.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 900–910, Osaka, Japan, December 11-17 2016.

- We introduce an AG-based approach to approximate the effect of scholar-seeded knowledge: we use a high-precision AG to derive a set of affixes in a first round, and then we insert these into the AGs for the second round. We call this approach "cascaded adaptor grammars". We show that again, our approach generally improves performance. We discuss our cascaded architecture in Section 6.

- The best performing AG-based configuration differs from language to language. However, we would like to have a language-independent system which we can apply to unseen languages. Sirts and Goldwater (2013) solve this problem by using a hand-annotated tuning set to choose the best underlying grammar. We want to remain entirely unsupervised. Therefore, we perform a cross-validation on the six languages, using the five development languages of one fold to determine which grammar to apply to the held-out unseen language. We find that we always obtain the same cascade of the same two grammars for all languages if we do not consider scholar seeding. For the scholar-seeded approach we get a tie between two grammars. We use these cross-validation results to define our LIMS system, which is a language-independent morphological segmentation system, and show that it always outperforms Morfessor and on five out of six languages outperforms the best single AG of (Sirts and Goldwater, 2013). LIMS is our main contribution, and its performance on unseen languages in the cross-validation is our main result.

2 Related Work

Early research on unsupervised morphological segmentation was performed by extensive manual rule engineering, which was very expensive. With the advance of machine learning, minimum description length (MDL) based unsupervised approaches were applied for morphological segmentation in several languages (Goldsmith, 2001). However, this required extensive manual work, which was then replaced by maximum likelihood optimization (Creutz and Lagus, 2002).

Morfessor (Creutz and Lagus, 2007) is a commonly used system for unsupervised morphological segmentation. It is based on a generative probabilistic model. Because of its broad use, we use Morfessor as a reference system in this paper. Another system was developed by Poon et al. (2009), who apply classic log-linear models that use contextual and global features.

Nonparametric Bayesian methods with probabilistic grammars, such as Dirichlet process mixture models (see Antoniak (1974), Pitman (2002)) are widely used in unsupervised learning for NLP. Johnson et al. (2007) introduce nonparametric Bayesian models on whole tree structures, namely; Adaptor Grammars (AGs). AGs provide a flexible distribution over parse trees and are successfully applied in unsupervised segmentation (Johnson, 2008).

Sirts and Goldwater (2013) explore the use of AGs for minimally supervised morphological segmentation. They also compare the performance of different grammar trees. In this paper, we explore a much larger set of grammars, and simulate the performance of doing supervised morphological segmentation without the use of any annotated data or scholar-seeded knowledge.

Snyder and Barzilay (2008) propose a discriminative model for unsupervised morphological segmentation by using morphological chains to model the word formation process. A main drawback in their system is the slow learning curve, which requires a vast amount of data to learn from.

Wang et al. (2016) propose novel neural network architectures that learn the structure of input sequences directly from raw input words and are subsequently able to predict morphological boundaries. The architectures rely on Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997).

3 Problem Definition, Data, and Experimental Setup

The specific problem we are tackling is a segmentation of words in a language into a sequence of morphs. We do not rewrite or normalize morphs, we do not identify the stem, and we do not identify morphological features. For example, the English word *repayments* should be returned as *re pay ment s*.

We perform experiments on six languages, namely English, German, Finnish, Turkish, Zulu and Estonian. The data for English, German, Finnish, Turkish, and Estonian is the data from MorphoChallenge.[1]

[1] http://research.ics.aalto.fi/events/morphochallenge/

For Zulu, we used the Ukwabelana corpus (Spiegler et al., 2010). The sizes of our corpora are summarized in Table 1. Note that we reduced the German dev corpus to obtain only instances with true segmentation; therefore, our results are not comparable to other published results on this corpus. However, all comparisons we present in this paper are always based on the same train and dev corpora.

We use the Adaptor Grammar (AG) package available from Mark Johnson.[2] We run the experiments using nine grammars of different characteristics, which we present in detail in Section 4. All experiments are run using transductive learning, i.e., we include the evaluation data in the unsupervised learning along with the training data. The AG parameters are the same as the ones used by (Sirts and Goldwater, 2013) with 500 sampling iterations instead of 1,000 iterations; early experiments showed that the results of 500 iterations are nearly as good as 1,000 iterations, while fewer iterations decreased performance. We adapt all nonterminals except nontermials with recursive rules. For the AG results, we report the average of five different runs. We also run Morfessor2[3] as a baseline (Virpioja et al., 2013).

Language	Training	Dev
English	50,851	1,171
German	50,537	540
Finnish	51,305	1,031
Turkish	51,096	1,531
Zulu	50,597	1,000
Estonian	50,374	1,497

Table 1: Size of corpora (in types) of our development languages

We evaluate the segmentation against the DEV data from MorphoChallenge. Our evaluation metric is EMMA (Spiegler and Monson, 2010), which is based on morph recognition. EMMA has the advantage that it can return a meaningful result on unsegmented words (also see (Virpioja et al., 2011)).

4 Underlying Grammars

There are three fundamental dimensions in designing the grammars. The first dimension is how the grammar generates prefix, stem, and suffix. The first option is that a grammar does not explicitly model the division into prefix, stem, and suffix at all and only has morphs ("morph-only"); this is illustrated by Morph+SM in Figure 1. If we assume that we do want an explicit modeling of prefixes, stems, and suffixes, we have a "tripartite" grammar. The "tripartite" grammar is illustrated by PrStSu (tripartite, Figure 2). As an example, we give a schematic tree for the word *repayments*; we omit many details such as the handling of the beginning and end of word markers, the details of the recursion within nonterminals with plural names such as `PrefixMorphs` and `SuffixMorphs`, and the details of the generation of morphs through a recursive generation of characters. The plus signs in the trees are just for orientation, they are not actually generated.

A second dimension of modeling is the levels which are represented in the nonterminals. All grammars represent morphs. The tripartite grammars also explicitly model prefixes and suffixes. In addition, we follow Sirts and Goldwater (2013) in allowing morphs to be composed of submorphs; even when we distinguish prefix morphs from suffix morphs in a grammar, the submorphs are shared (as is the case in (Sirts and Goldwater, 2013)). A second option we take from Sirts and Goldwater (2013) is the possibility of having compounding, which is implemented as an iterated nonterminal immediately below the word level. It allows for the generation of compounds in which each compound element has its own prefix, stem and suffix (for example, German noun compounds such as *Ver+läng+er+ung+s+ge+such+e* 'requests for extension').

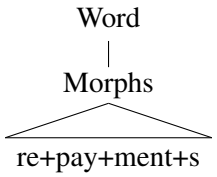


Figure 1: An analysis of *repayments* in Spine2+SM, a morph-only grammar. Submorphs are not shown.

A third dimension is the choice of the division into morphs for the output. If the grammar contains several levels of nonterminals (for example, compounds and morphs, or morphs and submorphs), then morph boundaries can be chosen at different levels (compound, morph, submorph). In the descriptions of our grammars below, we always list the level at which we define the morpheme boundary.

[2] `http://web.science.mq.edu.au/~mjohnson/Software.htm`
[3] `http://www.cis.hut.fi/projects/morpho/`

In developing our set of grammars, we started out with 44 grammars, which included the grammars used in (Sirts and Goldwater, 2013). Using EMMA F-measure as our criterion, we eliminated grammars which did not perform well across our languages, and we eliminated grammars which seemed to perform very similarly to other grammars. We ended up with nine grammars which we use for segmentation; one of these (PrStSu2b+Co+SM) was retained not because of its performance on F-measure, but on precision, which we only use for the first iteration in cascaded AG (which we will present in Section 6).[4] We now present our nine grammars.

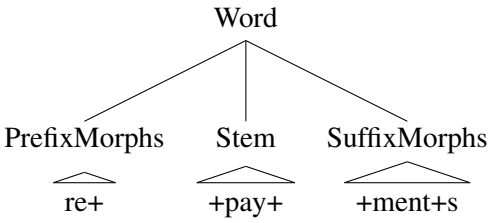


Figure 2: An analysis of *repayments* in PrStSu, a tripartite grammar. Submorphs are not shown.

We first have a morph-only grammar.

• Morph+SM: a word is recursively modeled as a sequence of morphs that consists of a lower level of submorphs, and the segmentation is based on the morph level. This grammar is grammar (2) (AG SubMorphs) from (Sirts and Goldwater, 2013), and we list it among our baselines.

We continue with tripartite grammars.

• Simple: a word is modeled as a sequence of an optional prefix, a stem, and an optional suffix, with no modeling of morphs beyond these three segments. The segmentation is based on the upper prefix, stem and suffix level.

• Simple+SM: the same as Simple with the introduction of a lower submorph level. The segmentation is still based on the prefix, stem and suffix level.

• PrStSu: a word is modeled as a prefix, stem and suffix sequence, where the prefix and suffix are sequences of zero or more morphs. The segmentation is based on the prefix, stem and suffix level. A sample analysis is shown in Figure 2. The segmentation is based on the prefix, stem and suffix level.

• PrStSu+SM: the same as PrStSu with the introduction of a lower submorph level shared by prefix and suffix morphs. The segmentation is based on the prefix morph, stem and suffix morph level.

• PrStSu+Co+SM: the same as PrStSu+SM with the introduction of an upper compound level. The segmentation is based on the prefix, stem and suffix level.

• PrStSu2a+SM: in contrast with PrStSu, where the prefix or suffix can simply be empty (but always exists in the derivation), we now allow the derivation to not have a prefix and/or a suffix. Specifically, a word is modeled as a stem-suffix sequence or a prefix and stem-suffix sequence, where the stem-suffix is a stem or a stem and a suffix. The prefix, stem and suffix are sequences of one or more morphs, with the introduction of a lower submorph level. The segmentation is based on the prefix, stem and suffix level. This grammar is an implementation of grammar (3) (AG Compounding) from (Sirts and Goldwater, 2013) without the compounding (since the compounding did not perform well for our languages), and we list it among our baselines in the result table.

• PrStSu2b+SM: this grammar is similar to PrStSu2a+SM but instead of modeling the the word as a prefix and stem-suffix sequence it is modeled reversely as a prefix-stem and suffix sequence.

• PrStSu2b+Co+SM: the same as PrStSu2b+SM but the upper compound level is added.

The results of the adaptor grammar experiments for our six development languages, English, German, Finnish, Turkish, Zulu and Estonian are in Table 2. Some observations:

- There is vast variation among languages in how grammars perform and which grammar is best.
- One of the grammars always beats the Morfessor baseline, and we always beat the AG baselines of Sirts and Goldwater (2013) except for Finnish. Some of the margins are small and probably not statistically significant.
- Grammar PrStSu2b+Co+SM, which has three levels of morphological representation (compounds, morphs, and submorphs) has very low recall across all languages, perhaps because the resulting morphs are too small (i.e., longer morphs are not predicted).

[4]The grammars are available at `http://www.cs.columbia.edu/~rambow/ag/ag.html` .

	English			German			Finnish		
Grammar	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Morfessor	79.7	81.4	80.5	79.0	69.6	74.0	74.5	61.8	67.5
Morph+SM	84.3	77.6	80.8	80.5	66.2	72.6	76.8	59.7	67.2
PrStSu2a+SM	75.9	80.3	78.0	80.1	72.6	76.2	74.2	**68.8**	**71.4**
Simple	64.5	72.1	68.1	71.7	67.5	69.6	69.6	61.6	65.3
Simple+SM	78.6	73.1	75.7	81.8	69.0	74.9	77.8	57.1	65.9
PrStSu	71.0	77.5	74.1	72.7	68.4	70.5	69.8	51.4	59.2
PrStSu+SM	81.2	**83.1**	82.1	81.3	**76.8**	**79.0**	66.6	59.8	63.0
PrStSu+Co+SM	89.7	76.5	**82.6**	**82.4**	61.6	70.5	81.5	58.3	68.0
PrStSu2b+SM	60.9	75.6	67.5	75.8	74.5	75.2	64.0	60.6	62.2
PrStSu2b+Co+SM	**92.4**	50.2	65.0	78.9	39.4	52.5	**93.0**	42.5	58.3
	Turkish			Zulu			Estonian		
Grammar	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Morfessor	67.4	46.6	55.1	53.4	33.8	41.4	75.0	81.0	77.9
Morph+SM	69.6	43.0	53.2	59.3	36.2	45.0	81.5	83.8	82.6
PrStSu2a+SM	75.8	55.1	63.8	52.2	42.1	46.6	66.8	82.7	73.9
Simple	64.8	45.7	53.6	59.4	49.5	54.0	65.5	78.9	71.6
Simple+SM	69.4	41.4	51.9	58.6	37.2	45.5	79.9	83.6	81.7
PrStSu	68.5	45.7	54.8	68.8	48.0	56.5	72.0	80.7	76.1
PrStSu+SM	77.8	**55.4**	**64.7**	57.4	43.5	49.5	65.3	81.1	72.3
PrStSu+Co+SM	71.1	40.6	51.7	59.9	34.7	43.9	84.8	**83.9**	**84.3**
PrStSu2b+SM	55.3	45.5	49.9	**72.0**	**51.5**	**60.1**	56.9	77.4	65.6
PrStSu2b+Co+SM	**82.3**	25.3	38.8	63.9	19.9	30.3	**93.9**	59.0	72.5

Table 2: Adaptor-grammar results (percent Emma) for English, German, and Finnish (above) and Turkish, Zulu, and Estonian (below). The best result per column is highlighted in boldface. The first three rows are baselines, and the next seven rows are tripartite grammars.

5 Scholar-Seeded Knowledge

The intuition behind the use of scholar-seeded knowledge is that for many languages, we have more or less extensive descriptions of their morphology. In fact, traditional descriptive grammars often concentrate on morphology. Today, a lot of information is available online. These resources provide lists of affixes, often in the form of paradigms or tables. Typically, only a very small number of lexemes are used to illustrate the morphology, or the affixes are simply listed without stems. Thus, the data is not a representative (type or token) sample of actual words in the language. We investigate the question of whether this data can be used in unsupervised segmentation. We note that this data is not "data" in the normal sense of machine learning: it is not in the same format as the desired output (i.e., segmented words). Therefore, this is not a case of semi-supervised machine learning, as Sirts and Goldwater (2013) explore.

Adaptor grammars is a framework that is particularly well suited for applying scholar-seeded knowledge as AG takes as input a hand-crafted grammar. Into this grammar, we can explicitly insert the affixes we have gleaned from the literature and from online sources. In Section 4, we investigated nine different grammars. We can insert the same affixes into all of these grammars in the position where morphs are generated. Of course, we continue to allow the grammars to generate new morphs, as we do not expect the sources to contain complete lists.

For these experiments, we consulted only online resources. We spent about two hours per language, and assembled between 30 and 120 affixes.

The results are shown in Table 3. We only show F-measure, and repeat the result for each grammar

Grammar	English			German			Finnish		
	Std.	**Sch.**	**Casc.**	**Std.**	**Sch.**	**Casc.**	**Std.**	**Sch.**	**Casc.**
Morfessor	80.5			74.0			67.5		
Morph+SM	**80.8**	75.2	75.3	72.6	**74.0**	73.1	**67.2**	63.9	63.3
PrStSu2a+SM	78.0	77.8	**79.6**	76.2	**77.3**	76.8	71.4	72.4	**73.3**
Simple	**68.1**	65.1	64.6	**69.6**	61.0	61.4	**65.3**	59.6	58.9
Simple+SM	**75.7**	72.2	72.0	**74.9**	68.8	68.9	**65.9**	60.8	60.8
PrStSu	**74.1**	64.2	64.5	**70.5**	67.3	67.6	59.2	60.3	**62.3**
PrStSu+SM	**82.1**	81.8	80.9	79.0	**79.3**	77.7	63.0	**72.9**	72.7
PrStSu+Co+SM	**82.6**	80.8	78.2	**70.5**	67.9	66.9	**68.0**	64.5	63.8
PrStSu2b+SM	67.5	69.1	**70.0**	75.2	76.6	**76.7**	62.2	64.9	**65.2**
PrStSu2b+Co+SM	65.0	**65.1**	65.0	52.5	52.6	**53.0**	58.3	58.4	**58.9**

Grammar	Turkish			Zulu			Estonian		
	Std.	**Sch.**	**Casc.**	**Std.**	**Sch.**	**Casc.**	**Std.**	**Sch.**	**Casc.**
Morfessor	55.1			41.4			77.9		
Morph+SM	53.2	54.5	**55.9**	45.0	47.6	**49.0**	**82.6**	71.2	71.1
PrStSu2a+SM	**63.8**	63.4	57.2	46.6	**56.5**	47.2	73.9	82.3	**82.8**
Simple	**53.6**	50.7	44.7	**54.0**	41.5	41.6	**71.6**	69.9	69.9
Simple+SM	**51.9**	51.0	49.3	**45.5**	44.1	44.1	**81.7**	77.3	77.3
PrStSu	**54.8**	**54.8**	51.4	**56.5**	46.4	46.1	**76.1**	70.2	69.5
PrStSu+SM	**64.7**	51.2	59.1	49.5	**65.7**	61.1	72.3	80.4	**80.5**
PrStSu+Co+SM	51.7	**52.3**	49.1	43.9	44.0	**44.1**	84.3	**84.4**	77.3
PrStSu2b+SM	49.9	**51.3**	51.0	**60.1**	58.2	47.7	65.6	**66.3**	66.2
PrStSu2b+Co+SM	38.8	39.5	**40.1**	30.3	**33.9**	30.3	72.5	**72.7**	**72.7**

Table 3: Adaptor-grammar results (percent Emma F-measure) for English, German, and Finnish (above) and Turkish, Zulu, and Estonian (below) for standard (Std; repeated from Table 2), scholar-seeded (Sch), and cascaded approaches (Casc). Boldface indicates best result by language for that grammar. The first three rows are baselines, and the next seven rows are tripartite grammars.

from Table 2 in the first column of each language. The second column shows the scholar-seeded result. (We discuss the third column in Section 6.) As we can see, the seeding of the grammars with some initial morphs does not, in general, improve our results. We provide a more detailed discussion at the end of the next section.

6 Cascaded Adaptor Grammars

In this approach, we investigate whether we can find the list of affixes which we use in scholar-seeded knowledge automatically, using AGs themselves. The basic approach is as follows:

1. We choose a grammar that has a high precision according to Emma across our development languages. The reason to choose a high precision (rather than a high F-measure) is that we want to be certain of having true affixes in the grammar, rather than having as many affixes as possible (even if some are not correct). We choose grammar PrStSu2b+Co+SM, which achieves the highest precision of all our grammars for English, Finnish, Turkish and Estonian, and close to highest for German. Only for Zulu is the precision mediocre. However, we want to choose a single grammar independently of the language, as we do not want to tune our approach to the language (we have no annotated tuning set).

2. We use this grammar to run AGs in a first iteration, and identify a list of prefixes and suffixes. We order the affixes by frequency.

English: +s, +ed, +ing, +e, +'s, re+, +es, +er, +y, a+, de+, +a, in+, co+, +ers, **ma+**, **ca+**, **se+**, **u+**, **e+**, **n+**, con+, **pa+**, +ly, **o+**, **ra+**, **+o**, **la+**, **ro+**, **ha+**, **ba+**, **mo+**, **ho+**, di+, +s', +ion, pro+, **sa+**, be+, **po+**
German: +en, +e, +er, +t, +s, ver+, be+, ge+, +ung, +es, +te, er+, +ten, +n, an+, ein+, aus+, ab+, +ungen, un+, vor+, zu+, +a, re+, ueber+, ent+, **s+**, +ischen, auf+, +et, +ern, +lich, +in, **+-**, unter+, +ische, **ma+**, +o, +ende, +enden

Figure 3: English and German affixes produced by our cascaded AG, shown in order of frequency in the corpus; incorrect affixes are in **boldface**.

3. We then include the top n affixes from our list in the grammars, in the same way we do for scholar-seeded knowledge. We run AG again, in a second iteration, using these modified grammars. We perform experiments on all our languages with $n = 10, 20, 30, 40, 50, 100$, which we refer to as Cascade$n$.

For example, grammar PrStSu2b+Co+SM finds the prefixes and suffixes for English and German shown in Figure 3; we list the 40 most common affixes (not including the empty prefix and the empty suffix, which are also generated). We show incorrect affixes in boldface. As we can see, the 15 most frequent affixes found for English are indeed correct affixes of English, but among the subsequent 25 most frequent affixes, only seven are correct.[5] In contrast, for German, a language with much richer inflectional and derivational morphology, all but three affixes are correct among the top 40 (and the top 25 are all correct).

The results for Cascade40 (i.e., using the top 40 affixes from the first iteration in the second iteration) are shown in the third column for each language in Table 3. We chose $n = 40$ since it has the best performance across all languages.

We now jointly discuss the results for the scholar-seeded approach (column 2 in Table 3, Section 5) and the cascaded approach (column 3, this section).

- If we simply look for the language-specific best performance, we see that the best score is achieved by a Standard AG for English and Turkish; by a scholar-seeded AG for German, Zulu, and Estonian; and by a cascaded AG for Finnish.

- The cascaded approach in general achieves results that are comparable to the scholar-seeded approach, i.e., we have shown that we can use AGs to provide information that is equivalent for the AG to what we can obtain from scholarly sources and the Internet.

- The scholar-seeded and cascaded approaches outperform the basic AG approach for some languages and some of our grammars. However, many grammars do not profit from scholar seeding or cascading. For example, for grammars PrStSu+SM and PrStSu+Co+SM, for all six languages the best configuration is the simple AG.

- Only for German and Zulu is the best performing configuration exactly the same; each of the other languages has a different configuration as its best performing. We thus do not see a cross-linguistic generalization emerge readily from Table 3.

If we were interested in optimizing performance for each language separately (i.e., by using the development set on which we are reporting results as a tuning set), then we would be done now. However, if we want to optimize our result across languages (as we do in this paper), we need to look more closely at the results, which we do in the next section.

[5]We consulted the Wiktionary pages for English affixes (`https://en.wiktionary.org/wiki/Category:English_prefixes` and `https://en.wiktionary.org/wiki/Category:English_suffixes`), but discarded some affixes which we did not feel were relevant (such and $n+$ as a typographical variant of μ). For German, we used our native speaker knowledge, as the corresponding Wiktionary pages do not contain all inflected affixes.

7 Finding the Optimal Language-Independent System

So far, we have discussed different approaches and presented results only on the development sets. In this paper, we are not interested in maximizing a single language-specific system; instead, we are interested in finding a single system which will perform well on an entirely unseen language (for which we have no annotation at all). To do this, we perform leave-one-out cross validation *on languages*. In each of the six folds of the cross validation, we choose one language in turn as the test language. We average the results for the other five languages (which become our development languages in this fold) for all grammars, for Standard, Cascaded, and Scholar-Seeded. We are interested in two configurations: no use of scholar-seeded knowledge (Standard or Cascaded), and inclusion of scholar-seeded knowledge (Scholar-Seeded). For each of these two configurations, we determine which grammar performs best on average across the five development languages of the fold. We then apply this grammar to the held-out test language. This means that for the held-out language, the choice of grammar was not in any way influenced by any observation from that language.[6] We first describe which grammars we choose in this manner.

- For the configuration without scholar-seeded knowledge, the cross-validation results are shown in table 4. We find that the best performing system for all six averages across five development languages (with the sixth language being held out) is a cascade, starting with PrStSu2b+Co+SM, and then inserting the obtained affixes into PrStSu+SM and running this modified PrStSu+SM. We will call this cascade of AGs LIMS, for Language-Independent Morphological Segmenter.

- For the configuration with scholar-seeded knowledge, the cross-validation results are shown in table 5. We find that the best performing system is split. For three held-out languages (Finnish, Turkish, and Estonian), the best performing grammar for the average of the other five languages is PrStSu+SM (augmented with scholar-seeded affixes), while for the other three held-out languages (English, German, and Zulu), it is PrStSu2a+SM, as shown in table 5. Since we need to choose a single configuration, we (arbitrarily) choose the grammar which we have already chosen for the configuration without scholar-seeded knowledge, namely PrStSu+SM, and we call this system LIMS-Scholar.

Held-out Language (HL)	**Best Grammar for {All - HL} (=G)**	**Ave. F-Score of G on {All - HL}**	**F-Score of G on HL**	**Oracle**
English	PrStSu+SM	70.2%	80.9%	82.6%
German	PrStSu+SM	70.8%	77.7%	79.0%
Finnish	PrStSu+SM	71.9%	72.7%	73.2%
Turkish	PrStSu+SM	74.6%	59.1%	64.7%
Zulu	PrStSu+SM	74.2%	61.1%	61.1%
Estonian	PrStSu+SM	70.3%	80.5%	84.3%

Table 4: Leave-one-out cross-validation results with no use of scholar-seeded knowledge. The Oracle result is the best performing configuration not using scholar-seeded knowledge on the held-out language, as shown in Table 3.

We now have two segmentation systems. LIMS is a black box system which can be applied to any language. LIMS-Scholar is a system which requires input in the form of a list of affixes. Clearly, its performance depends on the list of affixes provided. We present experimental results for these two systems in Table 6. The top two rows are baselines: the Morfessor system used out of the box (Morfessor2[7]), and grammar Morph+SM, which is the same as the AG SubMorphs grammar of Sirts and Goldwater (2013),

[6] We acknowledge that early elimination of additional grammars, not discussed in this paper, was in fact done by considering all languages.

[7] http://www.cis.hut.fi/projects/morpho/

Held-out Language (HL)	Best Grammar for {All - HL} (=G)	Ave. F-Score of G on {All - HL}	F-Score of G on HL	Oracle
English	PrStSu2a+SM	70.4%	77.8%	81.8%
German	PrStSu+SM	70.5%	77.3%	79.3%
Finnish	PrStSu2a+SM	71.7%	72.9%	72.9%
Turkish	PrStSu+SM	76.0%	51.2%	63.4%
Zulu	PrStSu2a+SM	74.6%	56.5%	65.7%
Estonian	PrStSu+SM	70.2%	84.4%	84.4%

Table 5: Leave-one-out cross-validation results with scholar-seeded knowledge. The Oracle result is the best performing configuration using scholar-seeded knowledge on the held-out language, as shown in Table 3.

which obtains the best average across their five development languages (our six development languages but not Zulu). We use our reimplementation of the grammar, and the results we obtain with this grammar are somewhat better than the results published in (Sirts and Goldwater, 2013), probably because of slightly different parameter settings. (Recall that our German data is different from the German data used in (Sirts and Goldwater, 2013).) The following two rows are our two systems. Since LIMS is always the same system, this row is the same as the penultimate column in Table!4. And the last two rows are the best results we obtained for that language across all of our grammars, without and with scholar seeded knowledge. We note that these last two rows are oracle experiments in the sense that we observe all of our results and choose the best configuration. However, if (for some odd reason) we wanted to perform unsupervised segmentation of words in one of our development languages, we would use these systems in the bottom two rows, though we have not demonstrated their performance on unseen test data in those languages (because that is not the goal of this paper).

System	English	German	Finnish	Turkish	Zulu	Estonian	Avg.
Morfessor	80.5%	74.0%	67.5%	55.1%	41.4%	77.9%	66.1%
Morph+SM	80.8%	72.6%	67.2%	53.2%	45.0%	**82.6%**	66.9%
LIMS	80.9%	77.7%	72.7%	**59.1%**	61.1%	80.5%	**72.0%**
LIMS-Scholar	**81.8%**	**79.3%**	**72.9%**	51.2%	**65.7%**	80.4%	71.9%
Best Standard/Cascaded	82.6%	79.0%	73.3%	64.7%	61.1%	84.7%	74.2%
Best Scholar-Seeded	82.1%	79.3%	72.9%	63.4%	65.7%	84.4%	74.6%

Table 6: A comparison between our systems and two baselines: Morfessor and Morph+SM (= AG SubMorphs of (Sirts and Goldwater, 2013)). The two best performing grammars from our experiments are also shown as an oracle result. The best result among the baselines and our systems is boldfaced.

Table 6 shows that for all of our held-out development languages except Turkish and Estonian, LIMS and LIMS-Scholar both outperforms both Morfessor and AG SubMorphs. For Turkish, LIMS outperforms both baselines, but LIMS-scholar outperforms neither. For Estonian, both LIMS and LIMS-scholar outperform Morfessor, while AG SubMorphs performs better than either of our systems. Furthermore, we see that in four of the six held-out languages, LIMS-Scholar outperforms LIMS, but on average LIMS slightly outperforms LIMS-Scholar because of the poor performance on Turkish. We note again that the performance of LIMS-Scholar depends on the quality of the scholar-seeded knowledge, so that one should be cautious with drawing conclusions. However, it appears overall that the cascaded approach of LIMS is a perfectly adequate alternative to using the scholar-seeded knowledge required for LIMS-Scholar.

8 Conclusion and Future Work

We have investigated the issue of unsupervised segmentation using Adaptor Grammars. Unlike recent work, we remain entirely unsupervised, i.e., we do not assume that we have some data which has been annotated by hand. We have experimented with different forms of grammars, the notion of seeding the grammar with knowledge obtained from scholarly publications and the web, and an architecture in which we obtain an equivalent amount of information using an AG, resulting in a cascaded architecture.

In this paper, we are not interested in maximizing performance on our development languages individually, and therefore we have not presented results on held-out test sets for the development languages. Instead, we have performed a cross-validation experiment on languages. We determine the best configuration to apply to the unseen language using only the other development languages, not the unseen language itself. We obtain two systems, which we call LIMS and LIMS-Scholar, with the latter using scholar-seeded knowledge. We show that LIMS outperforms Morfessor on all languages (LIMS-scholar on all but one language), and LIMS outperforms the previous best Adaptor Grammar results on all but one language (LIMS-scholar on all but two languages).

In future work, we intend to perform an extrinsic evaluation, in which an outside task, which requires morphological segmentation in its input, will be used to compare different settings. We will also investigate whether we can estimate the number of affixes to use in the cascaded approach; Figure 3 shows that choosing the top 40 affixes for all languages is not a good choice. Furthermore, we would like to determine in an entirely unsupervised manner the best underlying grammar for a language. This is an appealing goal as the best performance for a language is often superior to that obtained by LIMS or LIMS-Scholar.

Acknowledgments

We thank Kairit Sirts for her insights and for sharing her grammar trees, which we used as a baseline, and some of the data. We also thank three anonymous reviewers for their helpful comments. This work is supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL) contract number W911NF-12-C-0012. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

References

Charles E. Antoniak. 1974. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The Annals of Statistics*, 2(6):152–1174.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*, pages 21–30. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.*, 4(1):3:1–3:34, February.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational linguistics*, 27(2):153–198.

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

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Adaptor grammars: a framework for specifying compositional nonparametric bayesian models. In B. Sch"olkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 641–648, Cambridge, MA. MIT Press.

Mark Johnson. 2008. Unsupervised word segmentation for Sesotho using adaptor grammars. In *Proceedings of the Tenth Meeting of ACL Special Interest Group on Computational Morphology and Phonology*, pages 20–27, Columbus, Ohio, June. Association for Computational Linguistics.

Jim Pitman. 2002. Combinatorial stochastic processes. *Lecture Notes for St. Flour Summer School*.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 209–217, Boulder, Colorado, June. Association for Computational Linguistics.

Kairit Sirts and Sharon Goldwater. 2013. Minimally-supervised morphological segmentation using adaptor grammars. *Transactions of the Association for Computational Linguistics*, 1(May):231–242.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of ACL-08: HLT*, pages 737–745, Columbus, Ohio, June. Association for Computational Linguistics.

Sebastian Spiegler and Christian Monson. 2010. Emma: A novel evaluation metric for morphological analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1029–1037, Beijing, China, August. Coling 2010 Organizing Committee.

Sebastian Spiegler, Andrew van der Spuy, and Peter A. Flach. 2010. Ukwabelana - an open-source morphological zulu corpus. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1020–1028, Beijing, China, August. Coling 2010 Organizing Committee.

Sami Virpioja, Ville T. Turunen, Sebastian Spiegler, Oskar Kohonen, and Mikko Kurimo. 2011. Empirical comparison of evaluation methods for unsupervised learning of morphology. *Traitement Automatique des Langues*, 52(2):45–90.

Sami Virpioja, Peter Smit, Stig-Arne Grnroos, and Mikko Kurimo. 2013. Morfessor 2.0: Python implementation and extensions for morfessor baseline. Technical report, Aalto University.

Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. 2016. Morphological segmentation with window LSTM neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*.

CharNER: Character-Level Named Entity Recognition

Onur Kuru
Koç University, Turkey
okuru13@ku.edu.tr

Ozan Arkan Can
Koç University, Turkey
ocan13@ku.edu.tr

Deniz Yuret
Koç University, Turkey
dyuret@ku.edu.tr

Abstract

We describe and evaluate a character-level tagger for language-independent Named Entity Recognition (NER). Instead of words, a sentence is represented as a sequence of characters. The model consists of stacked bidirectional LSTMs which inputs characters and outputs tag probabilities for each character. These probabilities are then converted to consistent word level named entity tags using a Viterbi decoder. We are able to achieve close to state-of-the-art NER performance in seven languages with the same basic model using only labeled NER data and no hand-engineered features or other external resources like syntactic taggers or Gazetteers.

1 Introduction

Named Entity Recognition is commonly formulated as a word-level tagging problem where each word in the sentence is mapped to a named entity tag. A typical approach is to slide a window over each word position to extract features for a classifier that produces tags. Moreover, one can allow the classifications at adjacent positions to interact by chaining local classifiers together and perform joint inference. To achieve good performance, one has to overcome the data sparsity problem in the labeled training data. This is achieved by handcrafting good word-level features by exploiting affix, capitalization, or punctuation (Zhang and Johnson, 2003), using the output of syntactic analyzers (part-of-speech taggers, chunkers) and external resources such as gazetteers, word embeddings, word cluster ids to improve performance further (Turian et al., 2010; Ratinov and Roth, 2009). These solutions require significant engineering effort or language specific resources that are not readily available in all languages of interest.

It is even harder to design a multilingual model with handcrafted features considering each language offers different challenges. A distinguishing feature for one language may not be informative for another. For example, capitalization (a typical feature for English NER) is not a distinguishing orthographic feature for the Arabic script. Models for languages with agglutinative or inflectional morphologies may need to utilize the output of a language specific morphological analyzer. In some morphologically rich languages, production of hundreds of words from a given root is common, which makes models even more susceptible to the data sparsity problem.

This work is motivated by the desire to eliminate the tedious work of feature engineering, language specific syntactic and morphological analyzers, and language specific lexical or named-entity resources which are considered necessary to accomplish good performance on Named Entity Recognition. We accomplish this by combining the following ideas: First, instead of considering entire words as the basic input features, we take the characters as the primary representation as in (Klein et al., 2003; Gillick et al., 2016). Second, we use a stacked bidirectional Long Short Term Memory (LSTM) model (Hochreiter and Schmidhuber, 1997) which is able to operate on sequential data of arbitrary length and encode observed patterns in its memory at different scales. Finally, we use a Viterbi decoder to convert the character level tag probabilities produced by the LSTM into consistent word level tags.

Considering characters as the primary representation proves fruitful in several ways. Characters provide sub-word-level syntactic, morphological, and orthographic information that can be directly exploited by our model, whereas word-based models have to incorporate this information using feature

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 911–921, Osaka, Japan, December 11-17 2016.

engineering. Furthermore, useful sub-word features may vary from language to language, which our model can automatically learn but word-based models would have to incorporate using language-specific resources and features. Finally, character-based models reduce the size of the input vocabulary compared to word-level models, using fewer parameters, increasing computational and statistical efficiency.

We report results similar to or better than the state-of-the-art in resource-free Named Entity Recognition in seven languages. Moreover, our proposed model is not limited to Named Entity Recognition in particular and can be applied to other tagging tasks such as part-of-speech tagging.

In the rest of the paper we discuss related work in Section 2, detail our model and describe the input/output representation, stacked bidirectional LSTMs and inference details in Section 3. In section 4, we describe evaluation datasets in detail and present our experiments and results. Section 5 summarizes our contributions.

2 Related Work

In this section, we first outline the previous work on NER with word-level inputs then move onto character-based NER models. Next, we summarize the applications of character-based models in NLP in general.

2.1 Word based NER

Early successful studies on NER use hand-crafted features and language specific name lists with word-level classifiers. Both of the first place submissions in CoNLL-2002 (Spanish & Dutch), CoNLL-2003 (English & German) NER shared tasks (Carreras et al., 2002; Florian et al., 2003) use a rich set of handcrafted features along with gazetteers to achieve top performance. Subsequently, semi-supervised approaches (Ando and Zhang, 2005; Suzuki and Isozaki, 2008; Turian et al., 2010) have reported better results by utilizing large unlabeled corpora. Demir and Ozgur (2014) employs a semi-supervised learning approach to achieve best result for Czech. Darwish (2013) exploits cross-lingual features and knowledge bases from English data sources to achieve the top performance on Arabic. The current state-of-the-art system for Turkish (Seker and Eryigit, 2012) is based on Conditional Random Field (CRF) and utilizes language dependent features along with gazetteers.

2.2 Character based NER

Klein et al. (2003) introduce a Hidden Markov Model (HMM) with character-level inputs to alleviate the data sparsity problem inherent in word-level inputs. Their character-level HMM achieves a 30% error reduction over an HMM with word-level inputs. However, the character-level HMM suffers from unrealistic independence assumptions and it is not able to compete with their best system where they utilize a maximum-entropy conditional Markov model using richer set of features (word, all substrings of the word, part-of-speech and chunk tags). Ma and Hovy (2016) utilize pretrained word embeddings and character-level representation of a word by using a combination of Convolutional Neural Network (CNN), bidirectional LSTM and CRF. They report the best published result for English. Lample et al. (2016) also utilize characters to construct word representations with LSTM-CRF model and relies on unsupervised word representations extracted from unannotated corpora. They announce the best results for German and Spanish. Gillick et al. (2016) adapt the sequence-to-sequence model used for machine translation (Sutskever et al., 2014) to part-of-speech tagging and NER. The model inputs the text as sequence of bytes and outputs span annotations of the form (phrase start byte, length of the phrase in bytes, type of the phrase). Our model CharNER has a similar input/output representation with the character-based HMM of (Klein et al., 2003) and employs a much more compact network than (Gillick et al., 2016).

2.3 Other character-based models

Character-based models have been used successfully for NLP tasks other than NER as well. Depending on the nature of the task, characters are utilized in two different ways. One line of work uses characters to form a word representation for each token in a sentence (Kim et al., 2015; Ling et al., 2015a; Ballesteros

et al., 2015). Alternatively character representations are used without mapping to words first (Klein et al., 2003; Zhang and LeCun, 2015; Ling et al., 2015b; Dhingra et al., 2016).

Ling et al. (2015a) construct vector representations of words by composing characters using bidirectional LSTMs and Kim et al. (2015) employs a convolutional neural network (CNN) over characters to form word representations. Ling et al. (2015a) achieve state-of-the-art results in language modeling and part-of-speech tagging by utilizing these word representations. Kim et al. (2015) use word representations constructed by CNN with recurrent neural network for language modeling. They show that taking characters as the primary representation is sufficient to encode both semantic and orthographic information and their model is on par with the existing state-of-the-art despite having significantly fewer parameters. Ballesteros et al. (2015) employs the same strategy with (Ling et al., 2015a) to represent each token for a continuous-state dependency parsing model. They show that the parsing model benefits from incorporating the character-based encodings of words for morphologically rich languages.

Zhang and LeCun (2015) demonstrate that convolutional neural networks are successful at mapping characters directly to ontology/sentiment classes and text categories. Ling et al. (2015b) introduce a neural machine translation model that views the input and output sentences as sequences of characters rather than words. They show that the model is capable of interpreting and generating unseen word forms. They achieve translation results that are on par with conventional word-based models. Dhingra et al. (2016) proposes a model which finds vector space representations of whole tweets by utilizing character sequences rather than words. Their model performs significantly better compared to word-based counterpart when the input contains many out-of-vocabulary words or unusual character sequences.

3 Model

In this section we outline the architecture of our model, CharNER. We first define the input/output representation. Next we provide preliminary background on LSTMs and bidirectional LSTMs. Then we explain the deep stacked bidirectional LSTM network used in CharNER. Finally, we detail the decoding process.

3.1 Input/Output Representation

Since Named Entity Recognition is proposed as a word-level tagging problem, all of the proposed data sets use word-level tags to denote named entity phrases. A named entity (NE) phrase may span multiple words, hence a NE tag is composed of concatenation of a position indicator (B- Beginning, I- Inside) and a NE type (PER, ORG, LOC, ...). In addition, an O tag indicates that a token is not inside a NE phrase. A named entity phrase starts with a B- tag and if it consists of multiple words, the following word tags are prefixed with I-.

Our model, however, examines a sentence as a sequence of characters and outputs a tag distribution for each character. Therefore, we convert word-level tags to character-level tags. We abandon the position indicator prefixes (B-, I-) and use phrase types (PER, ORG, ...) directly as character tags. If a character subsequence in a sentence constitutes a NE phrase, all of the characters in that subsequence (including spaces) receive the same NE phrase tag. Otherwise, characters get the outside tag (O). Figure 1 shows word-level and character-level tags for an example sentence.

John	works	for	Globex	Corp.	.
B-PER	O	O	B-ORG	I-ORG	O

J	o	h	n		w	o	r	k	s		f	o	r		G	l	o	b	e	x		C	o	r	p	.		.
P	P	P	P	O	O	O	O	O	O	O	O	O	O	O	G	G	G	G	G	G	G	G	G	G	G	G	O	O

Figure 1: An example sentence with word level and character level NER tags.

3.2 Long Short-Term Memory

Recurrent Neural Networks (RNN) have recently achieved state of the art results in natural language processing tasks such as language modeling (Mikolov et al., 2010), parsing (Dyer et al., 2015), and machine translation (Sutskever et al., 2014). One major problem with simple RNNs is that they are difficult to train for long term dependencies due to the vanishing and the exploding gradient problems (Bengio et al., 1994). Hochreiter and Schmidhuber (1997) developed Long Short-Term Memory (LSTM) to overcome the long term dependency problem. They introduced a special memory cell which is controlled by input, output and forget gates. The input gate controls how much new information should be added to current cell, the forget gate controls what old information should be deleted. The output gate controls the information flow from the cell to the output. Many variants of the LSTM have been developed, in this study we use the LSTM architecture with peephole connections which was proposed by Gers et al. (2000). The LSTM memory cell is defined by the following equations[1]:

$$
\begin{aligned}
f_t &= \sigma(\boldsymbol{W_{fx}} x_t + \boldsymbol{W_{fh}} h_{t-1} + w_{fc} * c_{t-1} + b_f) \\
i_t &= \sigma(\boldsymbol{W_{ix}} x_t + \boldsymbol{W_{ih}} h_{t-1} + w_{ic} * c_{t-1} + b_i) \\
c_t &= f_t * c_{t-1} + i_t * \tanh(\boldsymbol{W_{cx}} x_t + \boldsymbol{W_{ch}} h_{t-1} + b_c) \\
o_t &= \sigma(\boldsymbol{W_{ox}} x_t + \boldsymbol{W_{oh}} h_{t-1} + w_{oc} * c_t + b_o) \\
h_t &= o_t * \tanh(c_t)
\end{aligned}
$$

where σ is the logistic sigmoid function, $\tanh$ is the hyperbolic tangent function and $*$ is element wise multiplication. f, i and o are the forget, input and output gates respectively, c denotes the cell vector, and h is the hidden state vector. All gate vectors and the cell vector have the same dimensionality as the hidden state vector. Bold upper case letters stand for matrices, lowercase variables are vectors, and subscripts indicate the connection (e.g. $\boldsymbol{W_{fx}}$: input to forget gate weight matrix).

3.3 Bidirectional LSTMs

It is a common approach to use both preceding and following tokens to derive features for the current token in natural language processing tasks. If we look at the LSTM equations, the current output depends only on previous inputs, the initial cell value and hidden state. Graves and Schmidhuber (2005) proposed bidirectional LSTM (BLSTM) to gain information from future inputs. In a BLSTM, two LSTM components are present, namely the forward LSTM and the backward LSTM. The forward LSTM traverses the sequence in the forward direction and the backward LSTM traverses same sequence in the reverse order using h_{t+1} and c_{t+1} are used instead of h_{t-1} and c_{t-1} for the gate calculations. For example, input gate at time t is calculated using the following:

$$
i_t = \sigma(\boldsymbol{W_{ix}} x_t + \boldsymbol{W_{ih}} h_{t+1} + w_{ic} * c_{t+1} + b_i)
$$

In a bidirectional model the output at time t depends on both the forward hidden state $\overrightarrow{h_t}$ and the backward hidden state $\overleftarrow{h_t}$.

3.4 Deep BLSTMs

The CharNER model uses bidirectional stacked LSTMs (Graves et al., 2013) to map character sequences to tag sequences. Figure 2 demonstrates the model overview. The network takes characters as the input sequence and each character is fed into the first forward and backward LSTM layers as one-hot vectors. The output of the first forward and backward layers are concatenated and fed into the next layer. The same process is carried on for the additional BLSTM layers. To obtain the distribution over the tag at position t, an affine transformation followed by a softmax is applied to the hidden representation of the final BLSTM.

[1] We denote matrices with bold upper case letters and vectors with lower case letters.

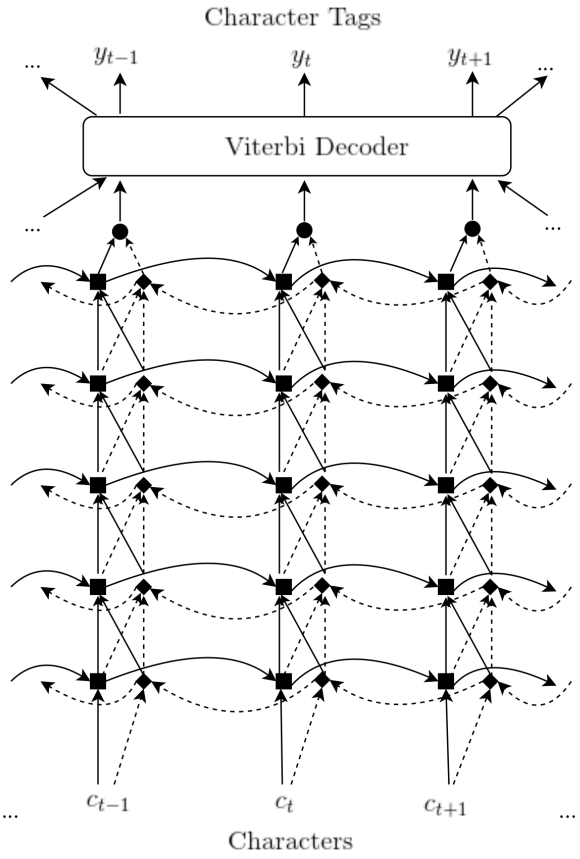


Figure 2: CharNER model: 5-layer Bidirectional LSTM Network with a Viterbi decoder. Each square and diamond node represents a forward and backward hidden LSTM layer, respectively. Circles denote the output layer (i.e. softmax layer). Solid lines show forward connections and dashed lines show backward connections. The model takes characters as an input sequence and each character is represented with a one-hot vector (c_t). A Viterbi decoder takes the sequence of character tag probabilities that is produced by the softmax layer and produces most likely character tag sequence (y) that is consistent at the word level.

3.5 Decoder

The deep BLSTM gives us a tag distribution for each character position. In this section we discuss the final step of turning these character level tag distributions into word level tags.

In early experiments, we observed that the most probable character tags within a word were not always consistent. For example, the model may assign higher probability to person (P) tags in the beginning of a word and organization (G) tags at the end of the same word. Even though the deep BLSTM has access to both left and right input contexts, it is unable to learn word level consistency for output tags. To remedy this, we use a decoder similar to (Wang et al., 2015).

Given a character sequence $c_1, c_2, ..., c_n$ (henceforth denoted with $[c]_1^n$), and a tag set $y \in \mathcal{Y}$, the decoder takes output tag probabilities from the LSTM, $o(c_i)_{y_i} = p(y_i \mid [c]_1^i, [c]_i^n)$, as emission probabilities and exploits transition matrices, A_{ij}, that only allow tags consistent within a word. Three types of transitions can occur between consecutive character tags (y_{i-1}, y_i) according to the position of the character at hand. A character is either followed by a fellow character in the same word ($c \rightarrow c$) or it can be the last character of a word followed by space ($c \rightarrow s$) where c denotes a character inside word boundaries and s denotes the delimiter space. Finally, it can be a space character followed by the first character of the next word ($s \rightarrow c$). The entries in Table 1 show transition matrices A_{ij} for these three states.

	PER	ORG	O	PER	ORG	O	PER	ORG	O
PER	1	0	0	1	0	1	1	0	0
ORG	0	1	0	0	1	1	0	1	0
O	0	0	1	0	0	1	1	1	1
	$c \rightarrow c$			$c \rightarrow s$			$s \rightarrow c$		

Table 1: Example transition matrices for two phrase types (PER, ORG) and Outside (O). c denotes a character inside word boundaries and s denotes the delimiter space.

The score of a sentence $[c]_1^n$ along a path of tags $[y]_1^n$ is computed by the product of transition scores

and model output probabilities:

$$s([c]_1^n, [y]_1^n) = \prod_{i=1}^{n} A_{y_{i-1}y_i} o(c_i)_{y_i} \quad (1)$$

The path of tags with the highest sentence score is computed by:

$$[y^*]_1^n = \arg\max_{[y]_1^n} s([c]_1^n, [y]_1^n) \quad (2)$$

To find the path of tags with the highest sentence score we use the Viterbi algorithm (Viterbi, 1967). Once a character tag sequence is decoded without violating word boundaries, it is trivial to convert these tags to word-level tags.

4 Experiments

This section presents our experiments and results. We start by describing the evaluation datasets with their properties. We detail our network training and demonstrate how changing the shape of the network effects performance. We present the results of our model on a variety of languages and compare with related work.

4.1 Datasets

In order to evaluate our system, we applied our model to NER datasets in various languages. The shared tasks of CoNLL-2002 and CoNLL-2003 provide NER data for four languages, Spanish, Dutch, English and German. In addition to these datasets, we evaluated our model on languages with rich morphologies: Arabic, Czech and Turkish. The Turkish NER dataset was prepared by (Tür et al., 2003). For Czech, Konkol and Konopík (2013) make the dataset prepared by (Ševčíková et al., 2007) CoNLL compatible. Benajiba et al. (2007) prepared ANERCorp for Arabic Named Entitiy Recognition. All of the datasets except Turkish and Arabic have conventional training, development and test splits. The Turkish dataset is provided with training and test sets. We separated a small fraction of the training set for development. Since the Arabic dataset (ANERCorp) do not have training and test splits, we applied Monte Carlo cross validation (Arlot et al., 2010) with 3 random runs. All of the datasets are in CoNLL format and we used the CoNLL evaluation script to report phrase-level F1 scores. Table 2 gives the number of sentences.

	Arabic	Czech	Dutch	English	German	Spanish	Turkish
Train	3988	4644	15806	14041	12152	8323	30000
Dev.	-	572	2895	3250	2867	1915	2237
Test	797[2]	577	5195	3453	3005	1517	3336

Table 2: Number of Sentences for Training, Development and Test sets.

In some morphologically rich languages production of hundreds of words from a given root is common, which increases data sparsity. We present unique, phrase-wide and corpus-wide unknown word percentages in Table 3 where a word is considered unknown if test set contains it but the word is absent in training. The Unique row counts an unknown word only once while the Phrase and Corpus rows consider all the occurrences of an unknown word in NE phrases and the whole test set, respectively.

	Arabic	Czech	Dutch	English	German	Spanish	Turkish
Unique	34.88	43.18	43.42	38.92	47.46	26.85	30.08
Phrase	18.82	35.54	39.28	31.27	42.20	18.39	13.82
Corpus	15.52	21.33	10.08	12.18	14.28	6.25	11.22

Table 3: Unknown Word Percentages.

[2]Randomly sampled with replacement for each run.

We give a brief comparison word-level and character-level views in Table 4. The number of unique tokens and average sequence length are indicated for each language. Switching from word-level to character-level view reduces the number of unique tokens drastically while making the input sequences longer. For direct comparison we trained and evaluated the same CharNER architecture (5 layer BLSTM with hidden size of 128) on both word-level and character-level data. The character-level model yields improved results for all datasets.

		Arabic	Czech	Dutch	English	German	Spanish	Turkish
Char	X	136	136	101	85	96	92	100
	μ_l	149	142	70	76	105	173	90
	F1	75.12	76.87	79.03	90.75	71.64	79.02	93.58
Word	X	27050	34869	27803	23623	32932	26099	63703
	μ_l	27	25	12	14	17	31	13
	F1	74.50	56.77	61.72	84.12	50.68	68.09	91.82

Table 4: Comparison of word-level and character-level views. X denotes the number of unique input tokens and μ_l denotes the average sequence length. F1 gives the best scores achieved on development sets when the same architecture is used on both word-level and character-level data.

4.2 Network Training

We use Adam (Kingma and Ba, 2014) for the gradient based training of the network[3] and we find that the default parameters given in the original study work well for this task. We update our network parameters after each mini-batch of 32 sentences. Before mini-batching, we sorted sentences according to character length to group sentences with similar lengths to speed up training. Also, we shuffle the order of mini-batches prior to each epoch. We use 5 BLSTM layers of size 128 (both forward and backward) stacked on top of each other and apply dropout (Srivastava et al., 2014) to outputs of each layer including the input layer. When no dropout is used, the network overfits the training data quickly and loses the generalization power. Notice that dropout at the input layer turns off bits of characters at random positions in the sequence. Applying dropout to character sequences yields +2 F1 score (absolute). To stabilize the network training, we use gradient norm clipping to prevent gradients from diverging (Pascanu et al., 2012). We set the maximum total norm of the gradients as 1. We tested several configurations of hyperparameters of the network and picked parameters that work well across the all seven languages. Although one may obtain further improvements by tuning hyperparameters as well as depth and width of the network for each dataset individually, we kept the hyperparameters and model configuration same for each language to demonstrate that one can achieve adequate results with the same model across many languages.

	Hidden Size		
Depth	64	128	256
1	52.06	56.27	57.98
2	60.31	68.72	70.77
3	67.22	71.09	70.84
4	70.16	71.70	71.60
5	70.79	72.19	70.53

Table 5: F1 scores on Czech test set for networks with different depths and hidden sizes. Depth denotes the number of layers the network has and Hidden Size denotes the number of hidden units for each layer.

4.3 Network Shape

We experimented with different configurations by varying the number of layers the network has (depth) and number of hidden units in each layer (width). We evaluated different volumed networks on Czech dataset which is modest in size and have conventional training, development and test splits. We summarized the results in Table 5.

Table 5 shows that increasing the depth of the model is more beneficial than increasing the width of the model. Although the wider model (width=256) yields better results when the network is shallow, its advantage disappears as the network gets deeper. Nevertheless, performance of narrow model (width=64) is limited compared to the model with medium width (width=128).

4.4 Need for a Decoder

As discussed in Section 3.5, the deep BLSTM does not output probabilities that always favor a single tag within a word. Character level NER is a structured learning problem, i.e. there are dependencies between the outputs that are difficult to capture by the deep BLSTM which only has access to the input sequence. To quantify the effect of capturing these dependencies, we ran a simpler baseline model for comparison. We did not apply Viterbi decoding to the output probability distributions of the network and picked the most probable tag for each character. Then, we used majority voting between characters of the same word to assign single tag to a word. With this scheme, we observed 2 F1 (absolute) performance drop on Czech dataset.

4.5 Results and Discussion

Here we present the performance of our model on languages with diverse characteristics. Since our model only uses the labeled training data, and no external resources such as gazetteers, we selected previous works which report the top scores without use of any additional data to make a fair comparison. We also included the best results which do make use of arbitrary external resources for each language. We summarized all the comparisons in Table 6.

The results highlight that our model attains good performance and it is robust across variety of languages. We achieve similar to or better than the state-of-the-art in Named Entity Recognition that use no external resources. Abdul-Hamid and Darwish (2010) employ a CRF sequence labeling model which is trained on features that primarily use character n-gram of leading and trailing letters in words and word n-grams. They assert that the proposed features help overcome some of the morphological and orthographic complexities of Arabic. Although they do not utilize any external resource, they apply Arabic specific input preprocessing before training which may be the reason for better performance. Demir and Ozgur (2014) employs a window-based classifier approach with language independent features for

	Arabic	Czech	Dutch	English	German	Spanish	Turkish
Best	84.30 [1]	75.61 [2]	82.84 [3]	91.21 [4]	78.76 [5]	85.75 [5]	91.94 [6]
	79.90	68.38	78.08	80.79	-	-	82.28
Best w/o External	81.00 [7]	68.38 [2]	78.08 [3]	84.57 [3]	72.08 [3]	81.83 [3]	89.73 [2]
CharNER	78.72	72.19	79.36	84.52	70.12	82.18	91.30

Table 6: Phrase-level F1 scores. The bottom row presents our model's results across seven languages. The middle row consists of models that report the top scores while not using any external resources, comparable to our model. The top row presents state-of-the-art models that achieve the best results in each language utilizing external resources like word embeddings or Gazetteers. The top row also reports the scores of best models when they only use NER training data, if available. Works are numbered in the order of appearance: [1] (Darwish, 2013), [2] (Demir and Ozgur, 2014), [3] (Gillick et al., 2016), [4] (Ma and Hovy, 2016), [5] (Lample et al., 2016), [6] (Seker and Eryigit, 2012), [7] (Abdul-Hamid and Darwish, 2010)

[3]Code repository: https://github.com/ozanarkancan/char-ner

Czech and Turkish. We outperform their results by a considerable margin for both languages. Gillick et al. (2016) reports the top scores with no external resources for Dutch, English, German and Spanish. They achieve higher F1 score for German dataset, however, our model performs better for Dutch and Spanish despite having 50% less parameters. Our result is on par with (Gillick et al., 2016) for English dataset.

Most state-of-the-art NER models are obtained by handcrafting good word-level features and generally utilizing external information sources. Models usually resort to additional training resources: (1) when training and test set are not from the same data generating distribution (English) or (2) training set is small (Arabic and Czech). Ma and Hovy (2016) report the best published F1 score (91.21%) for CoNLL-2003 English dataset. Without pretrained word embeddings however, their model loses approximately 10 F1 score (absolute). Lample et al. (2016) also relies on unsupervised word representations extracted from unannotated corpora. They announce the best results for German and Spanish datasets. On the other hand, Demir and Ozgur (2014) utilize pretrained word embeddings along with their corresponding word cluster ids to achieve top performance on Czech dataset. Darwish (2013) outperforms (Abdul-Hamid and Darwish, 2010) by 3.4 F1 score by exploiting cross-lingual features and knowledge bases from English data sources. Moreover, language specific expertise can be incorporated to improve the performance. The current state-of-the-art system for Turkish (Seker and Eryigit, 2012) achieves 91.94% F1 score by using language dependent features along with gazetteers. Finally, Gillick et al. (2016) achieves the best result for Dutch by using concatenated Dutch, English, German, Spanish training sets as one. Even though we do not use any additional training source, the performance of our model is competitive for Czech, Dutch, Spanish and Turkish.

5 Contributions

We describe a character-level tagger employing a deep bidirectional LSTM architecture and evaluate it on the Named Entity Recognition task. We showed that taking characters as the primary representation is superior to considering words as the basic input unit. Our main contribution is to show that the same deep character level model is able to achieve good performance on multiple languages without hand engineered features or language specific external resources.

In our current research, we are exploring ways to boost the performance of our model using semi-supervised and transfer learning. Moreover, our method may be promising for languages written without space characters such as Chinese and Japanese since word segmentation error affects the score of NER. Also, there is nothing specific to NER in our model, we are planning to evaluate it on other tasks such as part-of-speech tagging and shallow parsing.

Acknowledgements

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) grants 114E628 and 215E201.

References

Ahmed Abdul-Hamid and Kareem Darwish. 2010. Simplified feature set for arabic named entity recognition. In *Proceedings of the 2010 Named Entities Workshop*, pages 110–115. Association for Computational Linguistics.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853.

Sylvain Arlot, Alain Celisse, et al. 2010. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79.

Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. *arXiv preprint arXiv:1508.00657*.

Yassine Benajiba, Paolo Rosso, and José Miguel Benedíruiz. 2007. Anersys: An arabic named entity recognition system based on maximum entropy. In *Computational Linguistics and Intelligent Text Processing*, pages 143–153. Springer.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Xavier Carreras, Lluis Marquez, and Lluís Padró. 2002. Named entity extraction using adaboost. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–4. Association for Computational Linguistics.

Kareem Darwish. 2013. Named entity recognition using cross-lingual resources: Arabic as an example. In *ACL (1)*, pages 1558–1567.

Hakan Demir and Arzucan Ozgur. 2014. Improving named entity recognition for morphologically rich languages using word embeddings. In *Machine Learning and Applications (ICMLA), 2014 13th International Conference on*, pages 117–122. IEEE.

Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W Cohen. 2016. Tweet2vec: Character-based distributed representations for social media. *arXiv preprint arXiv:1605.03481*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. Association for Computational Linguistics.

Felix Gers, Jürgen Schmidhuber, et al. 2000. Recurrent nets that time and count. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 3, pages 189–194. IEEE.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1296–1306, San Diego, California, June. Association for Computational Linguistics.

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

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE.

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

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D Manning. 2003. Named entity recognition with character-level models. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 180–183. Association for Computational Linguistics.

Michal Konkol and Miloslav Konopík. 2013. Crf-based czech named entity recognizer and consolidation of czech ner research. In *Text, Speech, and Dialogue*, pages 153–160. Springer.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015a. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015b. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Gökhan Akin Seker and Gülsen Eryigit. 2012. Initial explorations on using crfs for turkish named entity recognition. In *COLING*, pages 2459–2474.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *ACL*, pages 665–673.

Gökhan Tür, Dilek Hakkani-Tür, and Kemal Oflazer. 2003. A statistical information extraction system for turkish. *Natural Language Engineering*, 9(02):181–210.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Andrew J Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. A unified tagging solution: Bidirectional lstm recurrent neural network with word embedding. *arXiv preprint arXiv:1511.00215*.

Tong Zhang and David Johnson. 2003. A robust risk minimization based named entity recognition system. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 204–207. Association for Computational Linguistics.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Magda Ševčíková, Zdeněk Žabokrtský, and Oldřich Krůza. 2007. Named entities in czech: annotating data and developing ne tagger. In *Text, Speech and Dialogue*, pages 188–195. Springer.

A Neural Model for Part-of-Speech Tagging in Historical Texts

Christian Hardmeier
Uppsala University
Dept. of Linguistics and Philology
751 26 Uppsala, Sweden
first.last@lingfil.uu.se

Abstract

Historical texts are challenging for natural language processing because they differ linguistically from modern texts and because of their lack of orthographical and grammatical standardisation. We use a character-level neural network to build a part-of-speech (POS) tagger that can process historical data directly without requiring a separate spelling normalisation stage. Its performance in a Swedish verb identification and a German POS tagging task is similar to that of a two-stage model. We analyse the performance of this tagger and a more traditional baseline system, discuss some of the remaining problems for tagging historical data and suggest how the flexibility of our neural tagger could be exploited to address diachronic divergences in morphology and syntax in early modern Swedish with the help of data from closely related languages.

1 Introduction

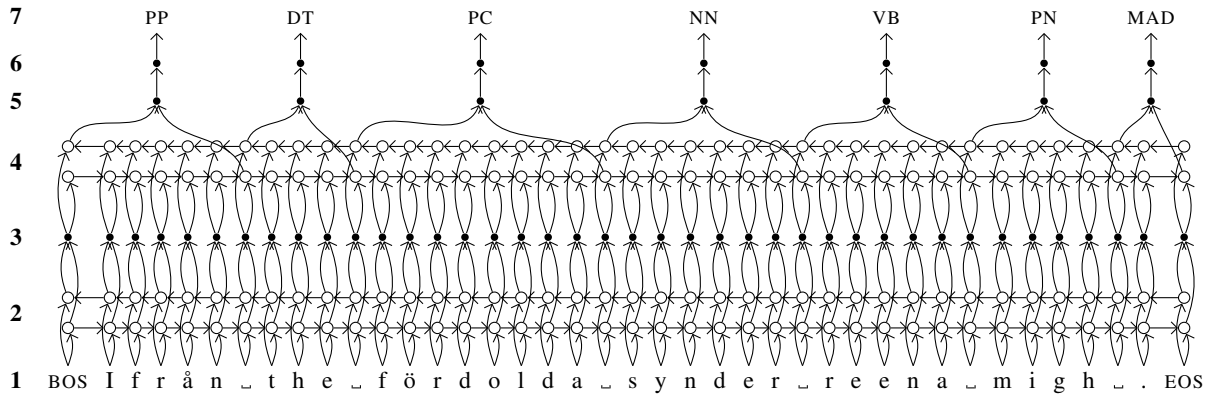
Most tools for automatic linguistic text annotation are based on supervised learning and trained on manually annotated text samples such as treebanks. This approach works best when the texts to be annotated are very similar to the language in the training corpora. The greater the differences, the more difficult it becomes to do automatic annotation with high accuracy. One application that poses particular challenges is automatic processing of historical texts. Language records from a few centuries ago are often still intelligible to modern readers, but they can nonetheless exhibit substantial divergence from later language use in terms of orthography, morphology, syntax, etc. Moreover, the languages we speak and write have undergone relatively recent processes of standardisation. Historically, there was much more variety in spelling and grammar both across and within texts, making the data sparseness problems we know from modern language processing even more acute. Standard approaches to deal with this challenge include manual or semi-automatic annotation of historical data sets to train language processing tools or automatic spelling normalisation to convert historical into modern spellings for the purpose of applying standard tools for modern language.[1] In this work, we present a neural network model to do part-of-speech (POS) tagging in historical texts. Our model uses a modern POS-tagged data set and a historical corpus with original and normalised spellings for training, but reads historical data without specific preprocessing at test time. We test the model on a Swedish verb identification and a German POS tagging task and analyse the output of the model to identify some remaining challenges to be addressed in future work.

2 Model Architecture

The core of our neural network is a POS tagger. The network takes as input a sentence in the form of a sequence of characters. For each character, it computes a representation in the form of a dense, approximately 50-dimensional vector that captures information about the character and its preceding and following context. The vector representations occurring at word boundaries are then used to predict a POS tag for each of the words in the sentence. At training time only, the model contains additional components

[1] For an overview of the relevant literature, we refer the reader to the recent PhD thesis by Pettersson (2016).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 922–931, Osaka, Japan, December 11-17 2016.



1 Character embeddings **2** Bidirectional GRU layer **3** Linear combination layer **4** Bidirectional GRU layer
5 Linear combination **6** Hidden layer with tanh activation **7** Softmax over POS tag vocabulary

Figure 1: Neural network architecture

	Swedish	German
Alphabet size	97	105
Character embeddings (**1**)	50	
First bidirectional GRU layer (**2**)	100	
Second bidirectional GRU layer (**4**)	51	
Final hidden layer (**6**)	300	100 *or* 300
POS tagset size (**7**)	29	58

Table 1: Neural network layer sizes

to ensure that the context-dependent vector-space representations created by the model are similar for historical data in original and normalised form.

Our character-level POS tagging model is shown in Figure 1. It is inspired by the work of Ling et al. (2015). The input of the model is a sentence split into characters. No normalisation or preprocessing is done at this point, and the input vocabulary consists of all Unicode code points encountered in the historical training set or its normalised form. In addition to the uppercase and lowercase letters of the modern alphabet, this also includes various forms of punctuation and letters with different diacritics, some of which are specific to the transcriptions of historical texts. The input characters are first transformed into dense character embeddings using a lookup table with trainable weights (**1**). Then, the entire sequence is scanned with a bidirectional recurrent neural network (Schuster and Paliwal, 1997) composed of gated recurrent units (GRUs; **2**) (Cho et al., 2014). The output states of the GRUs are passed through a linear layer and fed as inputs into another GRU layer (**4**). Up to this point, we are still processing the data at the character level and taking into consideration the context of the entire sentence. Unlike the model by Ling et al. (2015), our tagger never creates cacheable word embeddings that are independent of the surrounding words. We expect that this optimisation, which is used to speed up tagging in the Ling et al. model, would be less effective for historical than for modern text because of the greater spelling variability.

The transition to the word level, a prerequisite for predicting word-level POS tags, is done in the next step by combining, for each word, the final state reached by the forward and backward part of the bidirectional layer **4** after processing the word in question. These states are combined linearly (**5**), fed into a hidden layer using the hyperbolic tangent activation function (**6**) and passed on to a final softmax layer that outputs a probability distribution over the POS tagset (**7**).

The layer sizes of our network are shown in Table 1. Owing to memory limitations of the hardware we trained our systems on (Nvidia K20 GPUs with 5 GB of RAM), we could not test larger layer sizes systematically. Increasing the size of the hidden layer **6** from 100 to 300 brought a consistent improvement of 1–2 percentage points in F-score or accuracy for all experiments on Swedish. For German, the results were less conclusive, and the overall best model has a hidden layer of size 100.

Swedish				German			
Modern POS-tagged corpus		*Sent.*	*Tokens*	*Modern POS-tagged corpus*		*Sent.*	*Tokens*
Stockholm-Umeå corpus	Training	73,243	1,153,545	NEGRA corpus	Training	19,602	337,702
	Validation	500	7,287		Validation	500	8,415
	Test	500	5,924		Test	500	8,979
Historical corpus				*Historical corpus*			
Gender and Work corpus	Training	540	28,237	GerManC	Training	2,048	43,298
	Validation	60	2,590		Validation	186	4,216
	Development	600	33,544				
	Test	300	14,672		Test	216	4,845

Table 2: Corpus data overview

3 Model Training

The situation we consider in our experiments is one in which we have access to a POS-tagged training corpus of modern language as well as an unrelated corpus of historical texts in original and modern spelling, but not a POS-tagged training corpus of historical text. This corresponds to the actual situation for Swedish. Our historical training corpus for German does in fact contain a small amount of gold-standard POS annotations. In this paper, these are not used other than for comparison and evaluation.

The training objective we optimise our models for is to *maximise POS tagging performance on the tagged corpus* whilst *ensuring that the RNN states generated from historical texts in original spelling are similar to those arising from the corresponding normalised forms.* To achieve this, we compute two types of training error at every training step. The first is obtained by feeding a training example from the modern POS-tagged training set into the neural network shown in Figure 1. The *POS training error* E_{POS} of the training example is defined as the cross-entropy of the predicted tag distribution with respect to the gold-standard solution. For the second, we take a training example from the historical corpus and separately calculate the context-dependent word representations of the original historical text and its normalised form using layers **1** to **5** of the neural network, but omitting the hidden layer **6** and the final softmax layer. The *normalisation training error* E_{norm} of the training example is the squared error between the representation generated from the historical spellings and the representation of the normalised forms. The training examples used for the calculation of the two error types are independent from each other and paired randomly. The overall training objective is a weighted combination of the two error types:

$$E_{\text{total}} = \lambda E_{\text{POS}} + (1-\lambda) E_{\text{norm}} \tag{1}$$

To train our model, we apply minibatch stochastic gradient descent with a learning rate of 0.01, together with gradient clipping (Pascanu et al., 2013) to a maximum ℓ_2 norm of 10. The minibatch size was set to 30. We found that this batch size tended to give better results than smaller batches. Larger values could not be tested because of memory restrictions of our computer systems. The input sentences from both the POS-tagged corpus and the historical training corpus are cut at word boundaries into segments of approximately 25 words. To improve training efficiency, minibatches are formed from segments of similar length. The systems are trained on 100,000 to 200,000 minibatches, which corresponds to a wall-time limit of approximately 48 hours per training run. The error on the validation set is checked after every 3,000 batches. The set of parameters selected for evaluation purposes is the one that achieved the lowest validation error during training.

4 Tasks, Data Sets and Baseline System

We apply our model to two different tasks known from the literature. For Swedish, we address the problem of verb identification in historical texts. This task was introduced by Pettersson and Nivre (2011). It is motivated by its use in a historical research project named *Gender and Work* (Fiebranz et al., 2011). The goal of the *Gender and Work* project is to study the activities that men and women, respectively, carried out for a living in early modern Sweden (1550–1800). One of the core methods used in this project

was the systematic identification of verb phrases describing such activities in historical documents. In the course of the project, Pettersson and her colleagues developed data sets and methods to support the automatic annotation of such verb phrases (Pettersson, 2016). For German, the availability of a corpus of historical texts with gold-standard POS annotations allows us to tackle the more general task of POS tagging for historical texts.

For each language, we need a modern corpus annotated with POS tags and a corpus of historical texts in original and normalised spelling. Additionally, we need historical data with verb annotations or POS tags to evaluate our systems. Table 2 shows an overview of the corpora used in our experiments. For Swedish, we closely follow the setup of the experiments of Pettersson (2016). As a modern resource annotated with POS tags, we use version 2.0 of the Stockholm-Umeå corpus (SUC), a fairly large balanced collection of Swedish texts from the 1990's (Gustafson-Capková and Hartmann, 2006). We removed the last 1,000 sentences of the corpus to be used, in equal parts, as validation and test sets. As a historical training corpus, we have the *Gender and Work* corpus (Fiebranz et al., 2011; Pettersson, 2016). The split of this corpus into different data sets corresponds to the experiments of Pettersson (2016). The *training* and *validation* sets (corresponding to the training and tuning sets of Pettersson's spelling normalisation experiments) are used for neural network training and validation. The *development* set (corresponding to Pettersson's spelling normalisation evaluation set, which she subsequently used as a development set for verb phrase identification) was used as a test set during development. Finally, the *test* set (corresponding to Pettersson's verb phrase evaluation set) was used as a held-out set for the final evaluation of our model.

For German, our modern POS-tagged resource is the NEGRA corpus (Skut et al., 1997). As for SUC, we removed the last 1,000 sentences for validation and testing. Our historical data for German comes from the gold-standard portion of the GerManC corpus (Scheible et al., 2011), a corpus of early modern German (1650–1800) annotated with normalised spelling, lemmas and POS tags. The manually annotated gold standard part of this corpus consists of 24 documents. We set aside two of the more recent documents each for validation ("Ursprung", 1772; "Wolfenbüttel 1", 1786) and testing ("Gottesdienst", 1770; "Anton Reiser", 1790) and use the rest as training data.

Our baseline systems are modelled on the best-performing approach of Pettersson (2016) and consist of a pipeline that first normalises the spelling of the historical texts to be as similar as possible to modern orthography and then applies standard natural language processing tools trained on modern resources. The spelling normalisation component is a character-based statistical machine translation (SMT) system (Pettersson et al., 2013) implemented with the Moses toolkit (Koehn et al., 2007). It is a phrase-based SMT model with phrase length 10, disabled reordering and a 10-gram language model with modified Kneser-Ney smoothing (Chen and Goodman, 1998). The feature weights are tuned with minimum error-rate training (Och, 2003) to optimise the character error rate of the output. The default values of the Moses training pipeline and decoder are used for all other settings. After spelling normalisation, we run the HunPos tagger (Halácsy et al., 2007) for verb identification and POS tagging. Our HunPos models for Swedish and German are trained on exactly the same modern data sets as our own neural network tagger. For German, we also have the possibility to train HunPos on historical text with gold-standard POS tags from the GerManC corpus as another point of comparison.

5 Results

Table 3 shows the results of our two best-performing neural network taggers together with some comparative figures. The POS weight λ refers to the parameter in the error function in Equation 1. With equal weights for the POS tagging error and the normalisation error, our system reaches an F-score of 0.8668 on the development set and 0.8427 on the test set. Precision is higher than recall on the development set, but on the test set they are fairly balanced. Increasing the POS weight to 0.8 leads to an improvement to 0.8695 on the development set, which also carries over to the test set and gives us an F-score of 0.8529, about one percentage point over the result with equal weights. Decreasing the POS weight to 0.2 gives lower scores (not reported here).

The most interesting point of comparison is, of course, the HunPos system with SMT normalisation that emerged as the best model from the study of Pettersson (2016). Our own implementation of this

	SUC tagging	Historical verb identification					
		Development set			Test set		
POS weight	Accuracy	Precision	Recall	F-score	Precision	Recall	F-score
$\lambda = 0.5$	0.9625	0.8927	0.8424	0.8668	0.8454	0.8400	0.8427
$\lambda = 0.8$	0.9637	0.8909	0.8490	0.8695	0.8612	0.8448	0.8529
$\lambda = 1.0$	0.9534	0.8013	0.6517	0.7188	0.7623	0.6566	0.7055
HunPos with SMT normalisation	–	0.8773	0.8776	0.8775	0.8477	0.8729	0.8601
HunPos without normalisation	0.9772	0.7683	0.6173	0.6846	0.7202	0.6130	0.6623

Table 3: Results for the Swedish verb identification task

		POS tagging accuracy			Historical verb identification					
		NEGRA	GerManC		Development set			Test set		
POS weight	Layer **6** size		*dev*	*test*	P	R	F	P	R	F
$\lambda = 0.8$	100	0.9695	0.8382	0.8615	0.8780	0.9000	0.8889	0.9022	0.9008	0.9015
$\lambda = 0.8$	300	0.9692	0.8036	0.8520	0.8386	0.9091	0.8724	0.8796	0.8768	0.8782
$\lambda = 0.5$	300	0.9667	0.8157	0.8594	0.8612	0.9023	0.8812	0.8994	0.8864	0.8928
$\lambda = 1.0$	300	0.9661	0.8183	0.8444	0.8281	0.8318	0.8299	0.8325	0.8192	0.8258
HunPos trained on NEGRA										
with SMT normalisation		–	0.8577	0.8625	0.9116	0.8909	0.9011	0.8981	0.8880	0.8930
without normalisation		0.9952	0.8107	0.8353	0.8338	0.7295	0.7782	0.8643	0.7744	0.8169
HunPos trained on GerManC		0.7737	0.9082	0.9154	0.9044	0.8818	0.8930	0.8820	0.8608	0.8713

Table 4: Results for German POS tagging and verb identification

system achieves an F-score of 0.8601 on the test set, about half a percentage point better than the result of 0.855 reported by Pettersson for her corresponding system. The difference could be due to the feature weight settings of the SMT normalisation model or to some other minor difference in training parameters. Compared with those results, the scores achieved by our neural network tagger are very close, but still slightly lower. This corroborates Pettersson's finding that SMT normalisation is a very strong method for processing Swedish historical texts.

The other two contrastive systems reported in Table 3 are trained towards tagging modern Swedish without specific accommodations for historical text. Our character-based neural network tagger trained with a λ weight of 1.0 (at an F-score of 0.7055) seems to be slightly more robust to the unexpected historical spellings than HunPos (at 0.6632), but as expected, both models perform substantially worse in this setting.

The results of our experiments with German are in Table 4. The table includes POS tagging accuracies for the modern (NEGRA) and historical (GerManC) corpora. For better comparison with Swedish, it also includes precision, recall and F-score values for a historical verb identification task. These results were derived from the POS tagging results by considering only those word classes that would be tagged as verbs in the Swedish verb identification setup (i. e., finite, infinite and imperative forms of main, auxiliary and modal verbs, but not participles since they have a separate tag in the SUC tag set).

Unlike for Swedish, we do not see a consistent advantage from enlarging the final hidden layer **6** in the German experiments, and indeed the best overall score on the GerManC corpus is achieved with a layer **6** size of 100 in combination with a λ weight of 0.8. The development score of our best system is 2.8 percentage points above the HunPos baseline without normalisation, which already performs quite well on this task, but still 1.9 points below the baseline with SMT normalisation. On the test set, the neural system performs almost on a par with the SMT normalisation baseline; the small remaining difference corresponds to only 5 additional mistagged tokens out of 4,845. Increasing the size of layer **6** to 300 without additional regularisation results in development accuracy scores on the order of the HunPos baseline without normalisation. On the test set, these systems still outperform the unnormalised system and achieve scores that are only 0.4–1 percentage points lower than those of the comparison systems.

When we evaluate the experiments as a verb identification task, we see mixed results. Spelling normalisation, either in the form of a normalisation error term during neural network training or as a separate preprocessing step, has a clear advantage, but all normalisation-aware systems achieve fairly similar F-scores around or just under 90 %. Pitting our best-performing system against the baseline with SMT normalisation, we find that the latter has an advantage on the development set, but the former wins on the test set. Compared to the results on Swedish, the performance on German is even more similar. On the whole, we can conclude that both system types are clearly viable approaches to this task.

It is interesting to observe that the additional constraint we impose on the neural network tagger by requiring that its internal representation of historical spellings should be similar to that of modern text does not have a negative effect on its tagging performance for modern text. Indeed, both of the adapted Swedish systems in Table 3, while still about 1.5–2.5 percentage points below the performance of HunPos, achieve higher scores on the SUC test set than the tagger trained without the additional constraint. For German, tagging performance on modern text lags more behind the HunPos benchmark and the effect of adding the normalisation error is smaller, but still slightly positive. We have not studied this result in detail, but one could speculate that the normalisation error term adds a form of regularisation to the model that improves its performance on the original domain.

6 Qualitative Observations and Discussion

To gain a clearer picture of the strengths and weaknesses of our tagging models, we subjected the output of the models on the final test sets to a manual qualitative study. The study was done informally by looking through the original text, the spelling produced by the SMT normalisation step, the gold-standard annotations and the annotations generated by our own best system and by the HunPos baseline with SMT normalisation in parallel. For each language, we checked approximately 20 % of the test set data. For German, we additionally consulted the confusion matrices for the test set annotations generated by the baseline with SMT normalisation and by our best neural network tagger.

6.1 Swedish

For Swedish, we find very few qualitative differences between the baseline tagger and our neural system. By and large, both systems seem to struggle with the same difficulties, and they often make the same errors in parallel. In the Swedish verb identification task, by far the most common source of errors was a confusion between the tags for common nouns, NN, and verbs, VB. We encountered both nouns marked as verbs and vice versa, and both errors were frequent in both systems, making it difficult to draw conclusions about the properties of a specific system from these observations. Another type of error that occurred frequently in both systems was a confusion between verbs (VB) and participles (PC), which is understandable since participles are inflected verb forms and the tested systems are explicitly designed to be tolerant towards orthographical details. Other frequent sources of errors included confusions of verbs with adverbs (AB), adjectives (JJ) and proper nouns (PM). These occurred a bit more frequently in the output of the neural tagger, but they were well attested in the HunPos output as well.

One peculiarity that is specific to our neural network tagger is that it is much more likely to output the tag UO (foreign word) than HunPos (262 instances versus 11 in the test set). In general, it does this in quite reasonable ways, for instance for tagging the Latin words *pastor in* in a list of parish priests. However, since the foreign words in the SUC training corpus are mostly in English, a language scarcely attested in our early modern corpus, it sometimes overgenerates the UO tag in incorrect contexts, for instance for the word *tree* ‘three’ (modern spelling *tre*). More seriously, it occasionally seems to interpret the presence of the letter *w*, which is frequent in early modern Swedish, but missing in the modern Swedish alphabet except for its occurrence in foreign-language words, as a cue for generating the tag UO. To address these problems, we might consider augmenting the training data with sentences from relevant foreign languages, to familiarise the model with foreign words it might encounter, or with artifically generated historical spelling, to make it more robust to the expected spelling variance.

Our clear impression from the inspection of the tagging output for the historical Swedish found in the *Gender and Work* corpus is that the most important potential gains for this type of text and task are

unlikely to be realised by tweaking the implementation of the tagger, but require a more targeted approach to handle the specific differences between historical and modern language. The work in this paper, as well as the baselines we compare with, primarily addresses orthographical differences between historical and modern text. It is well known and acknowledged in the literature, however, that the diachronic differences in language development affect all parts of language, including not only orthography but also morphology, syntax, the lexicon and so forth. Pettersson (2016) provides a good overview of different linguistic properties that are relevant for historical language processing. In the case of verb identification in historical Swedish, the errors made by our systems suggest that least morphology and syntax should be considered for better results.

In terms of Swedish morphology, a very significant development that has taken place between the early modern period and now is the decline of verb inflection. In contemporary Swedish, verbs are not inflected for person. However, the complete disappearance of person inflection is relatively recent; until the first half of the 20th century, Swedish verbs had different forms for singular and plural, and in the early modern texts in our test set, we find a separate form for first person plural (as opposed to third person, second person not being attested in the sample we inspected). The tense forms of the first person plural have an ending in *-om*, which is easily confused with similar endings of other word classes by a purely orthographical approach:

> . . . och *worom* [VB, JJ] begiärandes / at klara Gudz ord måtte blifwa predikat kring om alt Riket.
> . . . and *were* desirous / that the clear word of God should be preached in all the country.

> . . . och *hördom* [NN, NN] thesligest theras predikan och disputatien som samma nya tro sagdes predika / och *funnom* [PP, VB] doch i sanningen thet rykte oredeliga fört wara. . .
> . . . and *heard* also the sermons and teachings of those who were said to preach that new faith / and *found* in reality that this rumour was spread dishonestly. . .

The three words in italics are all first person plural verbs; the POS tags in brackets were assigned to them by the neural tagger and the SMT-normalised baseline, respectively. In the first case, the HunPos tagger assigns the tag JJ because the word *worom* is incorrectly normalised to the adjective *varm* 'warm' by the spelling normaliser. In the second example, both taggers select an incorrect noun tag, presumably because they recognise *-dom* as a derivational suffix for abstract nouns (as in *visdom* 'wisdom'). The third example is incorrectly tagged as a preposition by the neural tagger, possibly because of the similarity of its ending with Swedish prepositions like *inom* 'within' or *förutom* 'except'. HunPos tagged it correctly even though it did not get transformed into a correct modern word form by the spelling normaliser and must therefore have been treated as an unknown word by the tagger.

Another morphological phenomenon that occurs very frequently in our texts is the derivational suffix *-liga* that is used to form adverbs, as in the word *oredeliga* 'dishonestly' in the previous example. In modern Swedish, the corresponding suffix is *-ligen*. In principle, this transformation is accessible to spelling normalisation, but the problem is that *-liga* could also plausibly be a plural ending of an adjective or a common noun derived from an adjective. Moreover, *-a* is the ending of the infinitive or third person plural of a verb. Accordingly, both taggers frequently assign adjective, noun or verb tags to these adverbs.

The syntax of early modern Swedish is strongly influenced by German. In particular, it is very common for subordinate clauses to have verb-final word order, as in both clauses of the second example above. In this particular example, the verbs were tagged correctly by HunPos, but the neural tagger failed to parse the clause *som samma nya tro sagdes predika* 'who were said to preach that new faith' correctly. In this indirect speech construction, the noun phrase *samma nya tro* 'the same new faith' is the object of the verb *predika* 'to preach', which in turn is governed by the passive verb *sagdes* 'were said to'. The clause could be rendered in modern Swedish as *som sades predika denna nya tro*. The neural tagger chooses to interpret the verb *predika* as the homonymous noun *predika* 'sermon', an interpretation that makes perfect sense in the light of contemporary Swedish grammar, which does not allow a direct object to precede the governing verb as early modern Swedish did.

While the two tagging approaches sometimes make different choices for individual examples, both of them are clearly affected by the problems outlined above. An interesting fact about early modern Swedish is that many of its historical features, despite having disappeared completely from the modern form of the language, are still attested in other, closely related contemporary languages. In particular, morphological features similar to those of early forms of Swedish can be found in present-day Icelandic, and modern German still exhibits some of the syntactic patterns that were common in early modern Swedish. We believe that the versatility of vector-space embeddings will make it possible to exploit resources from those languages to train models for historical forms of Swedish by integrating them in a similar way as we integrated the historical and modern data resources in this work. In this sense, the neural method has a clear advantage of flexibility over a pipeline approach with an explicit spelling normalisation stage.

6.2 German

The German test data is rather different from the Swedish test set, mostly because it is from a later period. The two texts we selected for testing in German are from the late 18th century. This is the age of authors like Goethe and Schiller, whose works had a lasting influence on the German language. While the writing style of that epoch may seem a bit archaic to a speaker of modern German, it is still perfectly readable and much closer to present-day German in terms of syntax and morphology than the texts of the *Gender and Work* corpus are to present-day Swedish.

In the German data, we can find some distinctive tagger-specific patterns. A recurring problem in the HunPos output is the incorrect assignment of an adjective tag ADJA to an attributive possessive pronoun that should be tagged PPOSAT. This invariably concerns the pronoun *unser* 'our', which historically and dialectally can have oblique forms with elided *e* such as *unsrem* (dative). These forms do not get translated into their modern standard spellings like *unserem* and are therefore not recognised by the modern tagger. The neural tagger handles these forms without any problems. HunPos also has a tendency to mix up common nouns (NN) with adjectives (ADJA or ADJD), whereas the neural tagger is more prone to confuse common nouns with proper nouns (NE).

A large class of errors that we find in the output of both taggers is the confusion of finite verbs (VVFIN, VAFIN and VMFIN) with infinite verbs (VVINF and corresponding tags for auxiliaries and modals) and, to a lesser extent, participles (VVPP etc.). The underlying problem for most of these examples is the homonymy of first and third person plural forms and infinitives. One of the texts in the test set is a homily that extensively mixes general exhortations in the form of infinitives with first person plural verbs, as in the following example:

> Es heißt nicht Werke der Barmherzigkeit deswegen *thun*, weil wir begangne Bosheiten, die wir nicht aufrichtig *bereuen*, dadurch auszulöschen … *glauben*.
>
> It does not mean to *do* acts of charity because we *believe* we can thus eliminate sins that we have committed … and do not sincerely *regret*.

Here, *thun* is an infinitive, while the following verbs are first person plural forms, but both taggers frequently confuse the two. Unfortunately, it is difficult to evaluate these examples correctly because the gold standard itself is inconsistent. The GerManC gold standard was produced semi-automatically with automatic annotation followed by manual error correction (Scheible et al., 2011). Since the infinitives and first person plural forms are homonymous and freely mixed in the text, disambiguating them is difficult for a tagger and not entirely trivial even for a human. Looking through the homily mentioned above, we quickly found more than 25 instances of incorrectly tagged verb forms that had probably escaped the manual correction pass. It is therefore unclear to what extent the gold standard can be trusted for this specific distinction in this specific text type.

7 Conclusion

In this paper, we have presented a new method for POS tagging historical texts with a character-based recurrent neural network. Our neural tagger can be trained on a combination of a modern tagged corpus and a historical corpus in original and normalised spelling. At training time, we use a two-part error

function that combines optimisation for POS tagging performance with a criterion to ensure that historical and modern spellings are represented similarly by the neural network. The trained model can then be used to process historical data directly, without explicit spelling normalisation and achieves a level of performance that is very close to that of a state-of-the-art solution with explicit SMT-based normalisation. In a manual study of the output of our own tagger and that of a baseline with explicit spelling normalisation, we have identified the most important remaining problems for the tasks under consideration. While the 18th century German texts exhibited general tagging problems that were more reminiscent of domain adaptation than peculiar to the historical nature of the texts, the older Swedish texts clearly suffer from specific problems due to language development. We suggest that our neural tagging approach opens up new ways for tackling these problems with the help of data from other, closely related languages. This is an approach that we plan to explore further in future work.

Acknowledgements

The author would like to thank Eva Pettersson for providing the data sets used in the experiments. This work was supported by the Swedish Research Council under grant 2012-916. The computations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at the Lunarc centre under project SNIC 2016/1-238.

References

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report, Computer Science Group, Harvard University, Cambridge (Mass.).

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha (Qatar), October. Association for Computational Linguistics.

Rosemarie Fiebranz, Erik Lindberg, Jonas Lindström, and Maria Ågren. 2011. Making verbs count: The research project 'Gender and Work' and its methodology. *Scandinavian Economic History Review*, 59(3):273–293.

Sofia Gustafson-Capková and Britt Hartmann. 2006. Manual of the Stockholm Umeå Corpus version 2.0. https://spraakbanken.gu.se/parole/Docs/SUC2.0-manual.pdf.

Péter Halácsy, András Kornai, and Csaba Oravecz. 2007. HunPos – an open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics. Companion Volume: Proceedings of the Demo and Poster Sessions*, pages 209–212, Prague (Czech Republic), June. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics: Demonstration session*, pages 177–180, Prague (Czech Republic).

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon (Portugal), September. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo (Japan).

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1310–1318, Atlanta (Georgia, USA).

Eva Pettersson and Joakim Nivre. 2011. Automatic verb extraction from historical Swedish texts. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 87–95, Portland (Oregon, USA), June. Association for Computational Linguistics.

Eva Pettersson, Beáta Megyesi, and Jörg Tiedemann. 2013. An SMT approach to automatic annotation of historical text. In *Proceedings of the Workshop on Computational Historical Linguistics at NODALIDA*, pages 54–69, Oslo (Norway).

Eva Pettersson. 2016. *Spelling Normalisation and Linguistic Analysis of Historical Text for Information Extraction*, volume 17 of *Studia Linguistica Upsaliensia*. Acta Universitatis Upsaliensis, Uppsala.

Silke Scheible, Richard J. Whitt, Martin Durrell, and Paul Bennett. 2011. A gold standard corpus of early modern German. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 124–128, Portland (Oregon, USA), June. Association for Computational Linguistics.

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

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 88–95, Washington (District of Columbia, USA), March. Association for Computational Linguistics.

Extracting Discriminative Keyphrases with Learned Semantic Hierarchies

Yunli Wang	Yong Jin	Xiaodan Zhu	Cyril Goutte
NRC Canada	U. New Brunswick	NRC Canada	NRC Canada
Scientific Data Mining	CS Faculty	Text Analytics	Multilingual Text Processing
Ottawa ON, Canada	Fredericton NB, Canada	Ottawa ON, Canada	Ottawa ON, Canada
Yunli.Wang@nrc.ca	Yong.Jin@unb.ca	Xiaodan.Zhu@nrc.ca	Cyril.Goutte@nrc.ca

Abstract

The goal of keyphrase extraction is to automatically identify the most salient phrases from documents. The technique has a wide range of applications such as rendering a quick glimpse of a document, or extracting key content for further use. While previous work often assumes keyphrases are a *static* property of a given documents, in many applications, the appropriate set of *keyphrases* that should be extracted depends on the set of *documents* that are being considered together. In particular, good keyphrases should not only accurately describe the content of a document, but also reveal what discriminates it from the other documents. In this paper, we study this problem of extracting *discriminative* keyphrases. In particularly, we propose to use the hierarchical semantic structure between candidate keyphrases to promote keyphrases that have the right level of specificity to clearly distinguish the target document from others. We show that such knowledge can be used to construct better discriminative keyphrase extraction systems that do not assume a static, fixed set of keyphrases for a document. We show how this helps identify key expertise of authors from their papers, as well as competencies covered by online courses within different domains.

1 Introduction

The purpose of keyphrase extraction is to automatically identify the most salient phrases from documents. Keyphrases (of which keywords are a special case) are widely used for providing a quick glimpse of various types of documents, such as news, technical documents, etc. Automatically extracting the relevant keyphrases therefore has a wide range of applications and accordingly has attracted much attention from the scientific community.

Previous work, however, often assumes that keyphrases are a *static* property of documents, that is, a given document would always produce a fixed set of keyphrases. Many approaches were developped for that purpose. For example, the Keyphrase Extraction Algorithm, or KEA (Witten et al., 1998), uses a supervised learning method (Naïve Bayes) to predict keyphrases based on their lexical features. Turney (2000) developed a genetic algorithm (GenEx) to extract keyphrases, and showed that this outperformed the well-known C4.5 algorithm. More recent work on supervised keyphrase extraction used, e.g., a combination of lexical and syntactic features (Hulth, 2003) or other statistical classifiers such as support vector machine (SVM) (Zhang et al., 2006) or conditional random fields (CRF) (Zhang et al., 2008). Unsupervised methods were also proposed, based on a graph-based ranking model (Mihalcea and Tarau, 2004), or using co-occurrences (Matsuo and Ishizuka, 2004), enriched with WordNet (Martinez-Romo et al., 2016). Unsupervised keyphrase extraction was also applied to shorter texts from twitter, using multiple random walks to topic context (Zhao et al., 2011) or unsupervised feature extraction (Marujo et al., 2015). The use of hierarchical information to extract keyphrases was explored in (Smatana and Butka, 2016; Berend, 2016). Although these supervised and unsupervised methods achieve improved performance, little work has been done to generate discriminative keyphrases based on other documents in the group.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 932–942, Osaka, Japan, December 11-17 2016.

In many applications, the appropriate set of keyphrases that should be generated depends on the context in which the document is considered, and particularly the *group* of documents being considered. For example, the keyphrases that represent the competencies of a researcher or a job hunter could depend on the groups of researchers or resumes that are being considered. Similarly, the keyphrases describing an online course depend on the set of courses under consideration: *Machine Learning* may be the appropriate descriptive keyphrase to describe a course within a set of Computer Science courses, but it is not very useful to a student considering a set of Machine Learning courses. Note that we differentiate the *collection* of documents, e.g. the online course descriptions, and the *group* of documents considered within the collection. For example, the subset of Machine Learning courses we use later is a *group* of 25 documents within the 1132 courses in the entire Coursera collection. Although previous work takes into account the specificity of terms within the *collection* (for example using inverse document frequency), they do not target discriminative keyphrases within a *group*. Restricting the collection and the extraction to the subset of documents in the group in order to use existing approaches has the important downside that it degrades the estimates of term/phrase frequency the extraction relies on. This is especially problematic for supervised approaches that require annotated documents.

In this paper, we study the problem of extracting *discriminative* keyphrases, that depend on the group of documents under consideration within a larger collection. We embed keyphrases in a semantic hierarchical structure using a *Deep Belief Network* (DBN) to characterize the relationship between pairs of phrases. We show that such knowledge can be used to build a discriminative keyphrase extraction system that adapts to the set of documents considered instead of returning a fixed set of keyphrases for a document. We test our approach on two tasks. First, using scientific articles, we extract keyphrases that identify authors expertise from the articles they published. Using a hierarchy of concepts learned from a scientific book, we show that this allows us to contrast researchers within different but related domains. The set of expertise keyphrases differs, for the same researcher, depending on the domain and the set of peers. In our second collection, we explore the problem of extracting keyphrases describing competencies taught by online courses. A semantic hierarchical structure of course phrases guides the extraction towards keyphrases that distinguish one course from the set of courses it is compared to. This is illustrated on two overlapping domains, showing that descriptive keyphrases for the same course may differ depending on the other courses within the domain.

2 Method

The discriminative keyphrase extraction relies on a keyphrase similarity decribed in Section 2.1, used to compute a similarity-based score (Section 2.2). We then extend that score with a semantic hierarchy learned using a Deep Belief Network, as described in Section 2.3.

2.1 Embedding-based Keyphrase Similarity

In order to measure the semantic similarity between two keyphrases p and q, we employ the widely used cosine similarity. This requires some kind of vector representation for both phrases. Learning representations for words, phrases or documents is central to natural language understanding. Vector representations learned using neural networks, a.k.a. embeddings, have recently shown to be effective in a wide range of tasks (Collobert et al., 2011; Mikolov et al., 2013). In our work, we use these low-dimensional vector representations to encode the meaning of each keyphrase.

Starting from word representations obtained from `word2vec`[1], we follow a standard approach to obtain a phrase representation, by averaging the vectors of each component word:

$$\mathbf{p} = \frac{1}{|p|} \sum_{w \in p} \mathbf{w}, \tag{1}$$

where p, w are the phrase and words respectively, $\mathbf{p}$ and $\mathbf{w}$ are their vector representations and $|p|$ is the number of words in p. For example, the embedding for *Machine Learning* is the average of the vector

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

representations for *Machine* and for *Learning*. The similarity between two phrases p and q is:

$$\text{cosine}(\mathbf{p}, \mathbf{q}) = \frac{\langle \mathbf{p}, \mathbf{q} \rangle}{\sqrt{\langle \mathbf{p}, \mathbf{p} \rangle \langle \mathbf{q}, \mathbf{q} \rangle}} = \frac{\sum_i p_i q_i}{\sqrt{(\sum_i p_i^2)(\sum_i q_i^2)}} \tag{2}$$

where i runs over the dimensions of the chosen embedding space, and $\langle \cdot, \cdot \rangle$ is the scalar product notation. Note that despite its simplicity, arithmetic average has been found to be very effective among many alternatives when combining word vectors to represent phrases (Mitchell and Lapata, 2008).

2.2 Similarity-based Discriminative Keyphrase Extraction

Now equipped with a similarity between keyphrases, we turn to extracting *discriminative* keyphrases for a document. In the similarity-based approach, we consider every candidate keyphrase p, and compare it to all other keyphrases from the group of document by computing the average similarity score between p and all other keyphrases q from all documents in the group. In our example, this would be all expertise keyphrases extracted for all researchers in the group considered:

$$sScore(p) = \frac{1}{C} \sum_{q \in \mathcal{K}} \text{cosine}(\mathbf{p}, \mathbf{q}), \tag{3}$$

where $\mathcal{K}$ is the set of keyphrases extracted from all documents, and $C = |\mathcal{K}|$ is the total number of keyphrases extracted in the group. We use $sScore(p)$ to rank all candidate keyphrases for the document (or researcher) under concern. We consider various ways to select the best discriminative keyphrases below and compare these different strategies in the experimental section.

Top: Pick the top N keyphrases, i.e. most similar with other candidates on average. These should be "safe bets" but not too specific;

Bottom: Pick the bottom N keyphrases, i.e. most dissimilar with other candidates on average. These should be very specific but also noisy;

Middle: Pick the middle N keyphrases, These may strike the right balance: some similarity with the rest, i.e. not noisy, but not too similar, i.e. specific to a document.

This separation is somewhat crude, but further investigation in Section 4.2 show that further refinements do not yield better performance.

2.3 Hierarchy-based Discrimintive Keyphrase Extraction

In order to use hierarchical semantic information to extract discriminative keyphrases, we first need to model the hierarchical information between keyphrases. We generalize the linear projection for hierarchical relations proposed by Fu et al. (2014), by using a Deep Belief Network (DBN) to model this relationship on keyphrase embeddings. A d-dimensional vector for each keyphrase is obtained using again `word2vec`, as in Section 2.1. The hierarchical information between two keyphrases p and q is then modeled as a binary classification problem: From a $2 \times d$ dimensional input containing the embeddings $\mathbf{p}$ and $\mathbf{q}$, the model predicts whether q is a child of p (positive class) or not (negative class).

DBNs are deep learning models consisting of multiple layers of hidden variables, often used to obtain abstract representations (e.g., features) for raw inputs. They were shown to be effective in many problems (Bengio, 2009). We use a typical DBN architecture composed of two hidden layers between one input and one output layer. Pairs of adjacent layers in the DBN are trained in a greedy layer-wise fashion as described in (Hinton et al., 2006). The principle of greedy layer-wise unsupervised training is widely applied to train DBNs with Restricted Boltzmann Machines (RBMs) as the building blocks for each layer. It mainly consists of two steps: (1) train each RBM in an unsupervised way to obtain the initial weights; and (2) starting from these initial weights, train the network in a supervised way using backpropagation. A RBM is a type of undirected graphical model (Hinton, 2010). Given a vector of visible (input) binary units $\mathbf{v} \in \{0, 1\}^{|\mathbf{v}|}$ and a vector of binary hidden units $\mathbf{h} \in \{0, 1\}^{|\mathbf{h}|}$, connections

NIPS	**DL**			**ML**		
	Researchers	keyphrases	gold.std	Researchers	keyphrases	gold.std
	10	206	77	10	173	70
Coursera	**MLC**			**CSDS**		
	Courses	keyphrases	gold.std	Courses	keyphrases	gold.std
	25	436	101	31	1190	292

Table 1: The number of total keyphrases and gold standard keyphrases in NIPS and Coursera datasets

between the visible and hidden units are weighted by the $|\mathbf{h}| \times |\mathbf{v}|$ matrix $\mathbf{W}$. Given bias terms $\mathbf{a}$ and $\mathbf{b}$ for the visible and hidden units, respectively, the energy function is given by:

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{a}^T\mathbf{v} - \mathbf{b}^T\mathbf{h} - \mathbf{h}^T\mathbf{W}\mathbf{v} \tag{4}$$

The joint probability distribution over visible and hidden units $P(\mathbf{v}, \mathbf{h})$, and the marginal distribution $P(\mathbf{v})$ over the visible units are defined as:

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z}e^{-E(\mathbf{v},\mathbf{h})} \quad \text{and} \quad P(\mathbf{v}) = \frac{1}{Z}\sum\nolimits_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{5}$$

with $Z = \sum_{\mathbf{v}}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$ the partition function. The RBM is trained using Gibbs sampling, alternatively sampling $\mathbf{h}$ given $\mathbf{v}$, and $\mathbf{v}$ given $\mathbf{h}$ from the conditional probabilities:

$$P(h_j = 1|\mathbf{v}) = \sigma(b_j + \mathbf{W}_{j\cdot}\mathbf{v}), \quad \text{and} \quad P(v_i = 1|\mathbf{h}) = \sigma(a_i + \mathbf{W}_{\cdot i}^T\mathbf{h}) \tag{6}$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function, $\mathbf{W}_{j\cdot}$ the j-th row and $\mathbf{W}_{\cdot i}$ the i-th column of weight matrix $\mathbf{W}$. Gibbs sampling allows us to get unbiased samples of the expectation of $v_i h_j$ under the distribution specified by the model, from which the RBM can be learned using contrastive divergence (Hinton et al., 2006).

Once the DBM model has been trained, we use it to predict the hierarchical relationship between pairs of candidate keyphrases. Specifically, given a candidate keyphrase, we form pairs with all the other keyphrases from the considered group of documents. For each pair, a prediction is made using the trained DBN model, indicating the hierarchical relationship between the two keyphrases in the pair. From these predictions, we estimate the number of children M of the candidate keyphrase in the group. This hierarchical information allows us to estimate the position of the candidate keyphrase in the semantic hierarchy. A large M indicates that the keyphrase is relatively high in the tree. Otherwise, the keyphrase is likely located at a lower level in the hierarchy. We incorporate this information in a hierarchy-based score by combining it with the discriminative score from Eq. 3 and modulating the trade-off through an exponent α:

$$hScore^{\alpha}(p) = sScore \times \left(\frac{1}{M}\right)^{\alpha} \tag{7}$$

3 Experiments

Two collections were used in this study: NIPS (Neural Information Processing Systems) conference papers and Coursera courses. The collections are described below and summarized in Table 1.

3.1 The NIPS Data

The NIPS data was obtained from `http://www.cs.nyu.edu/~roweis/data.html`. The dataset contains papers published at the NIPS conference from 1987 to 1999. All texts from volumes 0 to 12 were combined as one corpus to train the word embeddings using the `word2vec` tool. We set the window size to 8 and vector dimension to 200 so that each word is represented by a 200-dimensional numerical vector.

The DBN classifier modelling the hierarchical semantic information was trained from the table of content (TOC) of a machine learning book (Bishop, 2006). All words were converted to lowercase and unrelated keyphrases (e.g., from introduction and exercises) were removed. A phrase pair (p, q) was marked as positive if p was the ancestor of q in the TOC tree. Negative pairs were sampled from keyphrases without hierarchical relationship (e.g., from different chapter titles). The labeled training data includes 424 positive and 576 negative examples, used to train the DBN.

We selected two groups of researchers from the NIPS authors: one for deep learning (DL) and one for machine learning (ML). We selected ten researchers in each group, and collected reference expertise keyphrases from the following resources: homepages, resumes, Google Scholar webpages, LinkedIn profiles, as well as other related webpages. From the extracted list of keyphrases, four human annotators manually selected the gold standard expertise keyphrases from candidate keyphrases for each researcher and each group. Note that researchers who appear in both groups can have two distinct sets of reference discriminitive keyphrases, depending on which researcher group is considered.

3.2 The Coursera Data

Coursera is one of largest online education platforms, providing thousands of massive open online courses. One purpose of extracting discriminative keyphrases from course descriptions is to help students choose courses according to their interests. Discriminitive keyphrases among similar courses are more useful and meaningful than general keyphrases. We collected the course information of 1132 courses using the Coursera API[2] (Coursera, 2016).

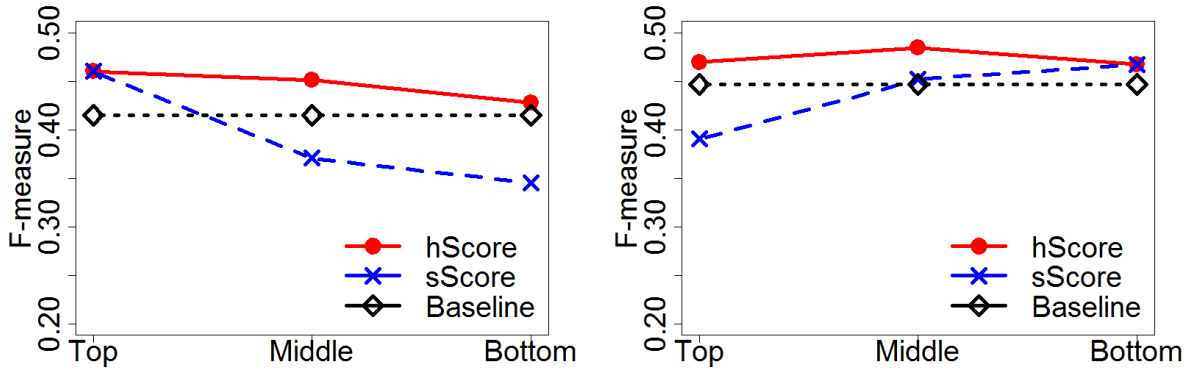
To obtain the semantic keyphrase hierarchy from the Coursera data, word vectors were trained using `word2vec` based on the whole Coursera corpus, and the DBN model was built based on the hierarchical pairs of phrases from courses. Phrases extracted from the course titles and course descriptions formed positive example pairs, while pairs of keyphrases occurring in the course description of the same course were negative examples. In total, 1945 positive and 455 negative keyphrase pairs were used to train a DBN model on the Coursera data.

Groups of similar courses were identified by clustering courses based on their textual descriptions. We removed punctuation, stop words and numbers and used tf-idf to generate document profiles. Thirty clusters were generated using k-means. The second largest cluster was identified as grouping a computer science and data science (CSDS) courses. We selected 31 courses with more than 30 candidate keyphrases from that CSDS cluster. We also selected 25 courses in the Machine Learning (MLC) subdomain under Data Science. Courses in MLC are more homogeneous than in CSDS, but may have fewer keyphrases (Table 1). In both groups (CSDS and MLC), candidate keyphrases were extracted from the course names and course descriptions. Part-of-speech patterns based on Brill's part-of-speech tagger (Turney, 1997) were used to extract candidate keyphrases. Two researchers picked the reference discriminative keyphrases from the set of candidates, for each course in CSDS and MLC (Summary in Table 1). Note again that the same course can have different reference discriminative keyphrases, depending on whether it is considered in the MLC or CSDS groups.

3.3 NIPS Data Results

Similarity- and hierarchy-based scores were used to extract the discriminative keyphrases from all articles of each researcher. Eight keyphrases in the DL group and seven keyphrases in the ML group were extracted for each researcher. The experimental results for both groups are illustrated in Figure 1. We measure performance using the F_1 score (Van Rijsbergen, 1979). The baseline method corresponds to randomly selecting keyphrases within the set of candidates. Its performance is the expected F_1 score under a uniform probability of extraction. For similarity- and hierarchy-based methods, the $sScore$ and $hScore$ were computed for each candidate expertise keyphrase for each researcher, and ranked in descending order of score. We measure the performance when selecting the top, middle or bottom keyphrases from the ranked list. Results from Figure 1 show that the hierarchy-based method ($hScore$) always outperforms both the baseline and the similarity-based approach ($sScore$). The similarity-based

[2] `https://building.coursera.org/app-platform/catalog/`



(a) DL

(b) ML

Figure 1: Performance of discriminative keyphrase extraction using similarity-based ($sScore$) and hierarchy-based ($hScore$) methods from the top ($\alpha = 0$), middle ($\alpha = 0.25$), and bottom ($\alpha = -2$) keyphrases in the NIPS DL (a) and top ($\alpha = -1$), middle ($\alpha = 3$), and bottom ($\alpha = 0$) in ML (b) groups of researchers. Baseline is a random choice among candidate keyphrases.

method sometimes performs worse than the baseline, which performs quite well as the candidate lists are small. These results show that the hierarchical information is clearly beneficial for discriminative keyphrase extraction. The best performance is achieved by mid-level keyphrases in the ML dataset and top level keyphrases in the DL dataset. This suggests that the hierarchy-based method does a good job pushing the relevant discriminative keyphrases towards the top in the narrower DL domain.

3.4 Coursera Data Results

The same setup was used to extract keyphrases that represent the concepts covered in Coursera courses. Ten keyphrases were extracted from the CSDS group, since each course has at least 30 keyphrases. For the MLC group, we extracted only 5 keyphrases as there are fewer candidates. Examples are given below for 2 course:

Course #1—Machine Learning: Clustering & Retrieval

1. MLC: mixed membership, expectation maximization, dirichlet allocation, other documents, latent dirichlet

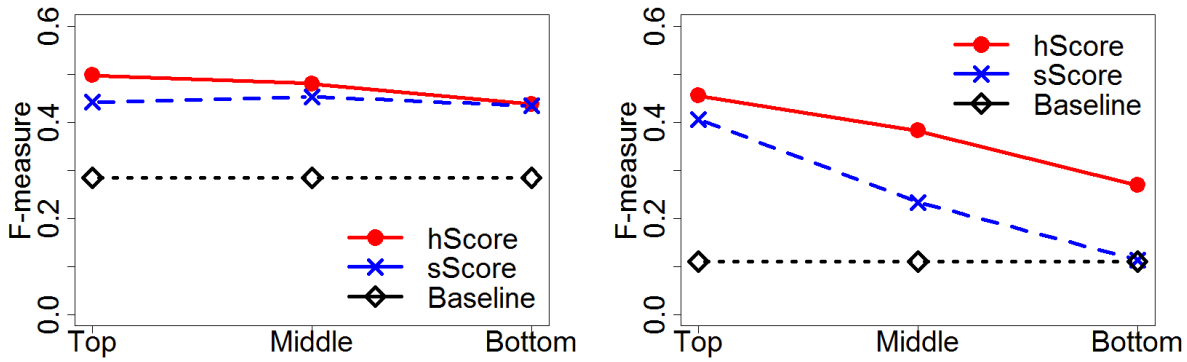
2. CSDS: document retrieval, similar documents, membership modeling, mapreduce learning, case study

Course #2—Machine Learning Capstone : An Intelligent Application with Deep Learning

1. MLC: product recommender, deep features, deep learning, intelligent application, pretrained models

2. CSDS: product recommender, learning classifiers, neural network, activation functions, pre-trained models

Results are presented in Figure 2, using the F_1 score for the top, middle, and bottom keyphrases. They show that both the similarity- and hierarchy-based methods outperform the baseline by a large margin in most situations. This is in part due to the fact that there are many more candidate keyphrases than in the NIPS data, so that the baseline's expected performance is much lower. The middle keyphrases yield the best performance for $sScore$ on the MLC group, otherwise the top keywords reach the best performance. This suggests again that both scores generally do a good job pushing the most discriminative



(a) MLC

(b) CSDS

Figure 2: Performance of discriminative keyphrase extraction using similarity-based ($sScore$) and hierarchy-based ($hScore$) methods from the top ($\alpha = -7$), middle ($\alpha = 0.5$), and bottom ($\alpha = 5$) keyphrases in the Coursera MLC (a) and the top ($\alpha = -7$), middle ($\alpha = 0.25$), and bottom ($\alpha = 10$) CSDS (b) groups of courses. Baseline is a random choice among candidate keyphrases.

competency keyphrases to the top of the list. The hierarchy-based method again consistently outperforms the similarity-based method. This suggests that the semantic hierarchy brings an important information that is useful for extracting discriminative keyphrases and provides a clear boost in performance.

4 Discussion

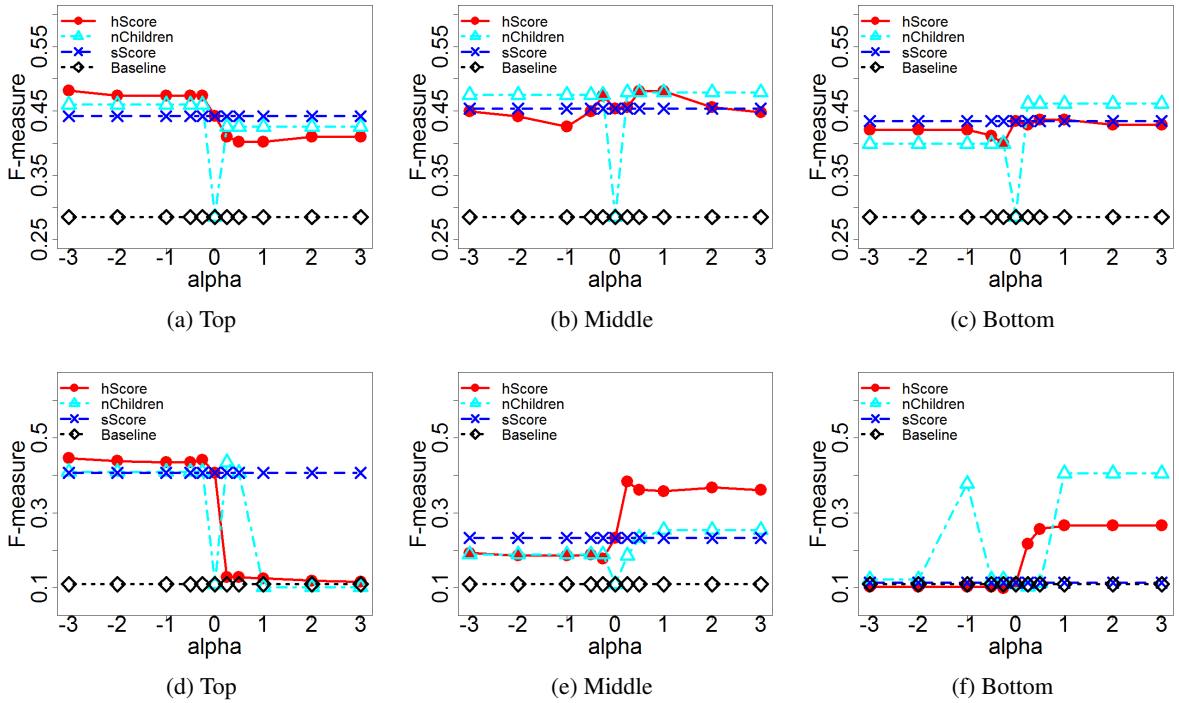
We analyze and discuss below two additional issues on the Coursera datasets: the effect of the hierarchical information on the hierarchy-based score, and the optimal selection of the discriminative keyphrases.

Note also that in this study, we used DBN to learn the hierarchical relationship between keyphrase pairs. Other classifiers could be used to model this relationship. However, DBN are expected to perform well on high dimensional word-embedding and are able to model non linear relationships between word pairs.

4.1 Effect of Hierarchical Information on the Hierarchy-based Scores

We saw that the hierarchy-based method outperforms the similarity-based method in all experiments. The $hScore$ is a trade-off between the number of children M and the $sScore$, to which it reduces when $\alpha = 0$ (Eq. 7). To further investigate the role of the hierarchical information in the $hScore$, we vary the value of α and compare it to the $sScore$, the baseline, and a new score using only the number of children, $nChildren = M^{-\alpha}$. We show their performance on the top, middle, and bottom level keyphrases in Figure 3. Note that negative α promote keyphrases with more children (more general keyphrases), while positive α push these keyphrases down the ranked list and favours more specific (less children) keyphrases. When $\alpha = 0$, $hScore = sScore$ and $nChildren$ is constant and performs the same as the random selection baseline.

On both course groups (MLC and CSDS), the best performance is achieved by the $hScore$ using negative α, on the top level keyphrases. Although the number of children behaves similarly in that regime, its performance is slightly lower, indicating that the keyphrase similarity still plays a key role. The difference in behaviour between $hScore$ and $nChildren$ is more pronounced on the bottom and middle level keyphrases. In particular, the $nChildren$ clearly outperforms the $hScore$ for positive α, but the resulting performance is still lower than what $hScore$ achieves on the top level keyphrases. Note also that the flexibility provided by the α parameter allows $hScore$ and $nChildren$ to always outperform the other two scores ($sScore$ and random baseline) for at least some value of α, again confirming the positive role of the hierarchical semantic information. In conclusion, this confirms that the best performance is usually obtained using the combination of similarity and hierarchy information implemented in the



(a) Top (b) Middle (c) Bottom

(d) Top (e) Middle (f) Bottom

Figure 3: Performance of several discriminative scores with varying trade-off parameter α, on the top (a), middle (b), and bottom (c) keyphrases for the Coursera MLC (top) and CSDS (bottom) course groups.

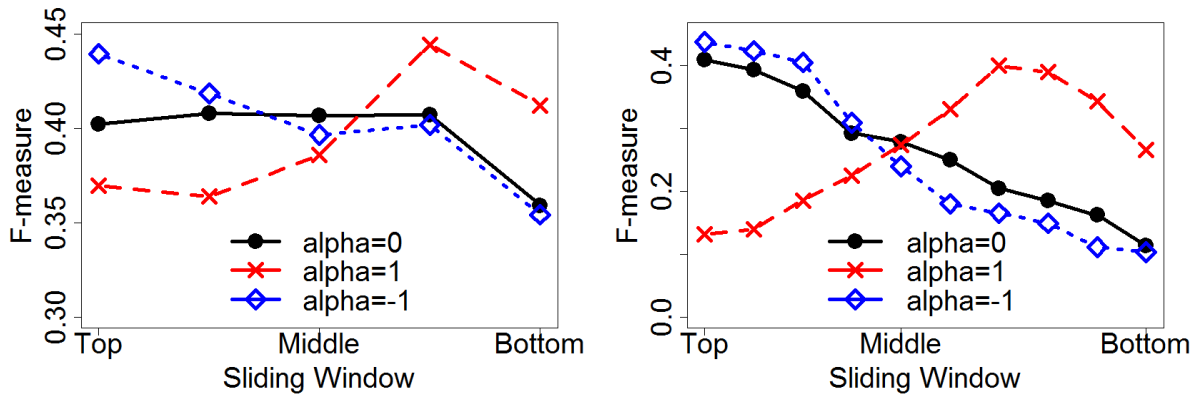
$hScore$, and that picking the keyphrases with the top scores works best.

4.2 Selection of Discriminative Keyphrases

We previously selected the discriminative keyphrases from the top, middle and bottom of the ranked candidate keyphrases. Although picking the top keyphrases usually works best, in the NIPS ML group, the middle level performed better. We investigate the influence of the location of the keyphrases in the list by computing the F_1 score of keyphrases in a sliding window on the MLC and CSDS course groups. Results are shown in Figure 4. Discriminative keyphrases were selected using five windows of five keyphrases (from top to bottom)[3] for the MLC group, and ten windows of ten keyphrases (from top to bottom) for the CSDS group. Results show that candidate keyphrases with high similarities and more children (top on the list when $\alpha = 0$ and $\alpha = -1$) perform very well for both groups. When $\alpha = 1$, keyphrases with more children were pushed down to the end of the ranked list, and they were selected as "lower intermediate" level keyphrases and performed better in both groups. There is a balance between the similarity with other keyphrases and number of children in the hierarchy in these two datasets, and $hScore$ is able to find the optimal parameter for extracting discriminative keyphrases.

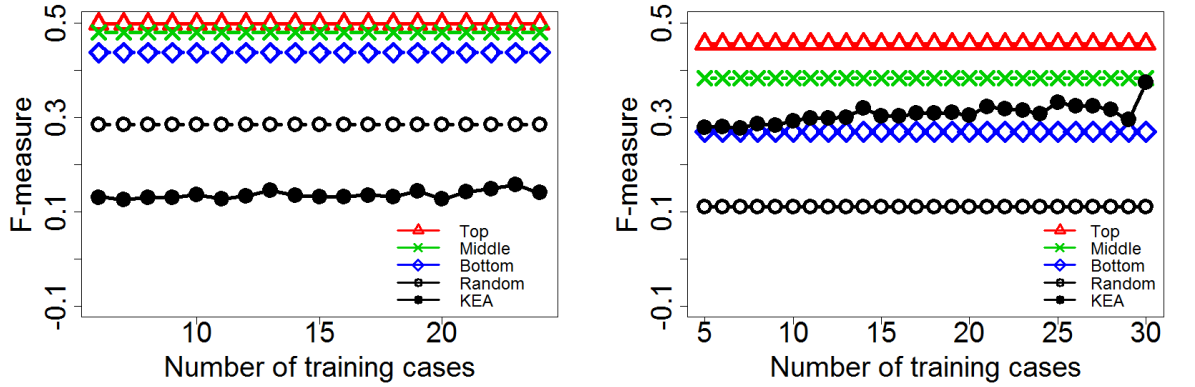
We also compared the performance of our method to KEA using the R package RKEA, in Figure 5. Whereas our method extracts discriminative keyphrases from the candidate keyphrases in a totally unsupervised manner (once the hierarchy is estimated), KEA uses a supervised learning methods to directly extract keyphrases from the text. In order to estimate the performance, we therefore average F_1 over 50 random choices of (labelled) training examples. As KEA does not use the same candidate keyphrases as our method, any partial match between keyphrases extracted by KEA and the gold standard is counted as a positive. We see from Figure 5 that the performance of KEA improves as the number of training cases increases, for both MLC and CSDS. However, on MLC is does worse than the random baseline, likely because it picks keyphrases that are not even among the candidate keyphrases. This is due to the fact that MLC contains courses with very short course description (sometimes as short as a

[3]For example, for a ranked list of 25 candidates, Figure 4(a) would show the performance of picking keyphrases in ranks 1–5 (top), ranks 6–10, ranks 11–15 (middle), ranks 16–20 and ranks 21–25 (bottom).



(a) MLC (b) CSDS

Figure 4: Performance of selecting discriminative keyphrases from a sliding window in Coursera data. Th x-axis indicates the position of the selected keyphrases in the ranked list of candidates (top to bottom).



(a) MLC (b) CSDS

Figure 5: Comparison with KEA in Coursera data.

couple sentences). KEA performed relatively better on CSDS, as that group contains courses with longe descriptions. Overall, our methods performed better than KEA on both datasets.

5 Conclusions

We propose a novel approach to keyphrase extraction, with a goal of finding phrases that both describe a document and differentiate it from a set of texts it is compared with. Previous work often assumes keyphrases are a *static* property of a document, while this work allows us to go beyond most state-of-the-art algorithms and generate keyphrases that depend on the set of documents under consideration, to generate discriminative descriptions of documents. This is done by learning the hierarchical semantic relation between concepts, and using this hierarchy to inform the keyphrase extraction process. We illustrate this on two datasets: a collection of scientific articles from which we extract keyphrases describing the expertise of authors in two related fields, and a collection of on-line courses from which we extract keyphrases describing the competencies covered by the courses, within two domains. Our experiments show that our method can extract domain-specific keyphrases, and that the hierarchical semantic information is useful for extracting the discriminative keyphrases from a group of similar articles or courses.

Acknowledgements

This work was funded by the NRC program Learning and Performance Support Systems (LPSS).

References

Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and Trends in Machine Learning archive*, 2:1–127.

Gábor Berend. 2016. Exploiting extra-textual and linguistic information in keyphrase extraction. *Natural Language Engineering*, 22:73–95, 1.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

Ronan Collobert, Jason Weston, Leon Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537.

Coursera. 2016. Catalog APIs. Accessed: 2016-03-18.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics*.

Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18.

Geoffrey Hinton. 2010. A practical guide to training Restricted Boltzman Machines. UTML TR 2010-03, Dept. Computer Science, University of Toronto, August.

A. Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 216–223.

Juan Martinez-Romo, Lourdes Araujo, and Andres Duque Fernandez. 2016. Semgraph: Extracting keyphrases following a novel semantic graph-based approach. *Journal of the Association for Information Science & Technology*, 67(1):71–82.

Luis Marujo, Wang Ling, Isabel Trancoso, Chris Dyer, Alan W Black, Anatole Gershman, David Martins de Matos, João Neto, and Jaime Carbonell. 2015. Automatic keyword extraction on twitter. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 637–643.

Y. Matsuo and M. Ishizuka. 2004. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13:2004.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Texts. In *Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, June.

M. Smatana and P. Butka. 2016. Extraction of keyphrases from single document based on hierarchical concepts. In *2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 93–98, Jan.

Peter D. Turney. 1997. Extraction of keyphrases from text: Evaluation of four algorithms. *National Research Council Technical Report*, pages 1–31.

Peter D. Turney. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336.

C. J. Van Rijsbergen. 1979. *Information Retrieval*. Butterworth, 2nd edition.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1998. KEA: Practical automatic keyphrase extraction. In *IN PROCEEDINGS OF THE 4TH ACM CONFERENCE ON DIGITAL LIBRARIES*, pages 254–255.

Kuo Zhang, Hui Xu, Jie Tang, and Juanzi Li. 2006. Keyword extraction using support vector machine. In *Proceedings of the 7th International Conference on Advances in Web-Age Information Management*, WAIM '06, pages 85–96, Berlin, Heidelberg. Springer-Verlag.

Chengzhi Zhang, Huilin Wang, Yao Liu, Dan Wu, Yi Liao, and Bo Wang. 2008. Automatic keyword extraction from documents using conditional random fields. *Journal of Computational Information Systems*, 4:1169–1180.

Wayne X. Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee P. Lim, and Xiaoming Li. 2011. Topical keyphrase extraction from Twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 379–388, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hashtag Recommendation Using End-To-End Memory Networks with Hierarchical Attention

Haoran Huang, Qi Zhang, Yeyun Gong, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing,
School of Computer Science, Fudan University
{huanghr15, qz, yygong12, xjhuang}@fudan.edu.cn

Abstract

On microblogging services, people usually use hashtags to mark microblogs, which have a specific theme or content, making them easier for users to find. Hence, how to automatically recommend hashtags for microblogs has received much attention in recent years. Previous deep neural network-based hashtag recommendation approaches converted the task into a multi-class classification problem. However, most of these methods only took the microblog itself into consideration. Motivated by the intuition that the history of users should impact the recommendation procedure, in this work, we extend end-to-end memory networks to perform this task. We incorporate the histories of users into the external memory and introduce a hierarchical attention mechanism to select more appropriate histories. To train and evaluate the proposed method, we also construct a dataset based on microblogs collected from Twitter. Experimental results demonstrate that the proposed methods can significantly outperform state-of-the-art methods. By incorporating the hierarchical attention mechanism, the relative improvement in the proposed method over the state-of-the-art method is around 67.9% in the F1-score.

1 Introduction

Along with the rapid development of social media, many people write brief text updates about their life on the go. Among these thousands of millions of microblogs posted every day, some contain # in front of words or unspaced phrases. The # symbol, called a *hashtag*, is usually used to mark keywords or topics in a microblog. Social media users originally created it to categorize messages. Now, hashtags have been widely used in a variety of circumstances. Hashtagged words that become very popular are often trending topics. Various works have also shown that hashtags can provide valuable information about different problems such as twitter spammer detection (Benevenuto et al., 2010), popularity prediction (Tsur and Rappoport, 2012), and sentiment analysis (Wang et al., 2011).

With the increasing requirements, the hashtag recommendation task has received considerable attention in recent years. Discriminative models have been proposed from different aspects using various kinds of features and models (Heymann et al., 2008; Liu et al., 2011), collaborative filtering (Kywe et al., 2012), generative models (Ding et al., 2013; Godin et al., 2013; She and Chen, 2014), and convolutional neural networks (CNN) (Gong and Zhang, 2016). Some of the previous works treated this task as a multi-class classification problem and used word-level features and exquisitely designed patterns to perform the task. Numerous existing studies utilized the word trigger assumption (Liu et al., 2011; Ding et al., 2013) and introduced topical machine translation models to achieve the task.

Due to the advantages of deep neural networks and the effectiveness of these methods in various NLP tasks, convolutional neural networks have also been applied to the hashtag recommendation task (Gong and Zhang, 2016). Some have also treated the hashtag recommendation task as a multi-class classification problem and incorporated an attention mechanism to handle the trigger words. This method only used a microblog as input. It did not take the history information of the user into account. However,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 943–952, Osaka, Japan, December 11-17 2016.

the microblogs a user posted in recent history can represent their interests to some degree. Previous works (Zhang et al., 2014) also studied this issue using generative models, and the experimental results demonstrated the usefulness of the histories of users.

In this work, to incorporate the histories of users, we propose a novel end-to-end memory network architecture to combine the microblog textual information and corresponding user history to perform the task. We regard the user history as an external memory for the microblog. The memory networks can help to extract the useful features from the external memory, which are relevant to the microblog and hashtag, to construct user interest representations. With the underlying intuition that not all microblogs in the user history are equally relevant for recommending hashtags, and not all of the words in a microblog are equally important, we introduce a novel hierarchical attention mechanism and integrate it with a memory network to capture two insights about the posting histories of users. Experimental results demonstrate that the performance of the method incorporating the hierarchical attention mechanism is also better than the method without it.

The main contributions of this work can be summarized as follows:

- To incorporate the user history information, we propose to extend the end-to-end memory networks to perform the hashtag recommendation task.
- Since not all of the microblogs and words in a microblog are equivalent in importance, we introduce a novel hierarchical attention mechanism and integrate it with the end-to-end memory networks.
- Experimental results using a dataset collected from a real microblogging service demonstrated that the proposed method can achieve significantly better performance than the state-of-the-art methods.

2 Related Work

2.1 Hashtag Recommendation

In recent years, various studies have been conducted on this task(Kywe et al., 2012; Ding et al., 2013; Godin et al., 2013; Sedhai and Sun, 2014; Wang et al., 2014; Gong and Zhang, 2016; Shi et al., 2016).

Kywe et al.(2012) used a similarity based method to solve this problem. They recommends hashtag by combining hashtags from the similar tweets as well as hashtags from the similar users. Many other approaches focus on the topic modelling(Ding et al., 2013; Godin et al., 2013; She and Chen, 2014). Based on the assumption that the hashtag and the trigger words of the tweets are two different language and have the same meaning, Ding et al.(2013) proposed to use translation process to model this task. In contrast to the methods that focuses on topic modelling, Shi et al.(2016) proposed a learning-to-rank approach for modelling hashtag relevance. Due to advantages of deep neural netowrks, CNN has been applied on this task(Gong and Zhang, 2016). Most of the works are based on textual information of tweets. However, some other works found there were different types of information which are helpful. Zhang et al.(2014) proposed a topical model based method to incorporate the temporal and personal information. Sedhai and Sun(2014) combined the textual information and hyperlinked information to recommend hashtags.

In this work, we propose a novel networks architecture to combine the textual information and the corresponding user history to perform the task.

2.2 Attention and Memory

The second relevant line of work is the research on attention mechanisms and memory networks. Attention mechanisms have been widely used in many studies and have shown to achieve promising result on several tasks, such as generating handwriting (Graves, 2013), machine translation (Bahdanau et al., 2014), speech recognition (Chorowski et al., 2014), action recognition (Sharma et al., 2015), caption generation (Xu et al., 2015) and so on. In recent months, memory networks(Weston et al., 2014) have been proposed and applied on natural language question answering, which have four component: input (I), generalization (G), output (O) and response (R) component. After then, Sukhbaatar et al.(2015) proposed a end-to-end memory networks and applied it on question answering and language modeling.

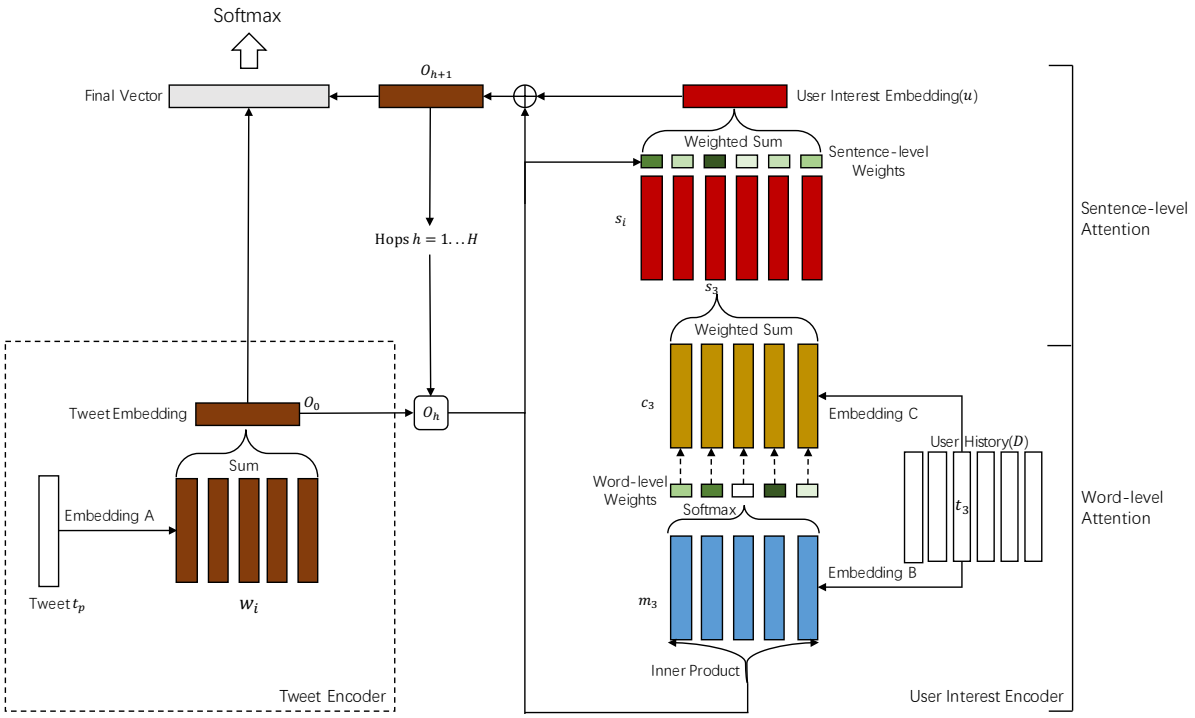


Figure 1: End-To-End Memory Networks with Hierarchical Attention

The memory networks architecture has been adopted in dialog systems(Dodge et al., 2015; Bordes and Weston, 2016; Weston, 2016) and query answering(Kumar et al., 2015; Weston et al., 2015).

In this work, we treat user history as the external memory and use a memory networks architecture with hierarchical attention to encode the user interest.

3 Approach

3.1 Preliminary

Given a tweet t_p, our task is to recommend a hashtag that is most relevant to the tweet. Based on this definition, we formulate the hashtag recommendation task as a multi-class classification problem. From the above description, we can see that the user information is also very important because each user always focuses on several aspects that may relate to the hashtag. Meanwhile, the interests of users can in most cases be represented by the tweets that they post. Hence, we use the tweet set D to represent the interests of users. Each tweet is a word sequence denoted by $t = \{w_1, w_2, ..., w_N\}$, where N is the length of the tweet. The user history contains many tweets denoted by $D = \{t_1, t_2, ..., t_M\}$, where M is the size of the history document. Let $H_p = \{H_{p1}, H_{p2}, ..., H_{p|H_p|}\}$ be the set of candidate hashtags.

3.2 The Proposed Methods

To solve this classification problem, we propose a novel end-to-end memory network architecture (HMemN2N) to combine the tweet textual information and corresponding user history, which is shown in Figure 1.

In our proposed models, we first use a tweet encoder to embed the tweet t_p. Then, the user interest encoder can extract the interest information of the user from the user history D with the help of t_p. Finally, we combine the information from two encoders and use a softmax layer to score and recommend a hashtag list. In this work, we regard the user history as an external memory for the tweet t_p. The introduction of the memory networks can help to extract the useful features from the external memory to build the user interest representations. With a underlying intuition that not all tweets in the user history are equally relevant for recommending the hashtag, not all words in each tweet are equally important,

and the tweets in the history document are relatively independent and focus on different aspects, we introduce a hierarchical attention mechanism into memory networks to capture two insights about the user history document and obtain a high-quality user interest representation.

We describe the details of different parts of Figure 1 in the following sections.

3.2.1 Tweet Encoder

As shown, the first part of the original input is the tweet t_p, and it is treated as a bag-of-words (BoW) representation. Then, we embed each word w in t_p in a continuous space and sum the embedding vectors to obtain an embedding representation. Specifically, the embedding matrix A(of size $dim \times |V|$, where V is the vocabulary and dim is the embedding dimension) is used to look up the vectors for words w, and the representation can be calculated as follows: $o_0 = \sum_i^N Aw_i$. The embedding o_0 is also treated as an initial input of the user interest encoder.

3.2.2 User Interest Encoder

The second part of the input is the user history, which is stored in the memory. We use a memory with a two-tier architecture: sentence and word. Based on the previously provided notations, the document is a tweet set: $D = \{t_1, t_2, ..., t_M\}$, which is a sentence-level structure. Then, each tweet $t_i \in D$ is divided into a bag-of-words representation: $t_i = \{w_{i1}, w_{i2}, ..., w_{iN}\}$.

To obtain the user interest representation, we propose a two-level encoder architecture, which can then be stacked in what is called multiple hops, denoted as $h = 0, 1, 2, ..., H$.

Now, we first introduce a word-level attention mechanism for embedding the tweets t_i. There are two components: input memory and output memory. In the input memory component, given an input set $\{t_1, t_2, ..., t_i\}$, each word $w_{ij} \in t_i$ is embedded using a matrix B of size $dim \times |V|$ into memory vectors $\{m_{ij}\}$ of dimension d, giving $m_{ij} = Bw_{ij}$. The input memory representation m_i of tweet t_i is a matrix of size $N \times dim$, where N is the length of the tweet. However, because not all words contribute equally to the tweets meaning, and the importance degree of the word w_{ij} should be considered in our models, we propose a probability layer to achieve the goal. The match between o_h and memory vectors m_{ij} is then computed by taking the inner product followed by a softmax:

$$p_{ij} = \frac{exp(o_h^{\mathrm{T}} m_{ij})}{\sum_n^N exp(o_h^{\mathrm{T}} m_{in})}, \tag{1}$$

where o_h is the internal input state in hop h, and p_{ij} is a probability vector over the input memory.

In the output memory component, each word $w_{ij} \in t_i$ has a corresponding output vector c_{ij}, which is obtained using an embedding matrix C with the same size. Then, the output representation s_i of tweet t_i is constructed by summing the output vector c_{ij}, weighted by the probability p_{ij}, and the equation of this operation is as follows:

$$s_i = \sum_j^N p_{ij} c_{ij}, \tag{2}$$

From the above procedure, the memory is converted into a matrix s that contains M tweet embedding vectors of size dim. Next, we propose a sentence-level method to extract sentences that are important to the user and aggregate the representation of this information to form the user interest representation. The weight p_{s_i} of sentence s_i is calculated, and the user interest vector u is formed by the weighted sum of the tweet embeddings:

$$m_{s_i} = tanh(W_o o_h + W_s s_i), \tag{3}$$

$$p_{s_i} = \frac{exp(W_{ms}^{\mathrm{T}} m_{s_i})}{\sum_j^{|M|} exp(W_{ms}^{\mathrm{T}} m_{s_j})}, \tag{4}$$

$$u = \sum_i^{|M|} p_{s_i} s_i, \tag{5}$$

where $|M|$ is the number of tweets in the users history document, and the parameters in these equations are W_o, W_s, and W_{ms}.

By implementing the hops operator, the memory can be read and write iteratively using the state o_h. At the last step of each hops h, a new output state o_{h+1} is updated with $o_{h+1} = o_h + u$, where u is the user interest embedding obtained in this hop. The last output state o_H is regarded as a high-level representation of user insterest.

3.2.3 Final Prediction

Finally, the tweet embedding o_0 and the user interest output o_H can also be concatenated into the final vector f, giving $f = o_0||o_H$. Then, we can use the final vector to predict the recommended hashtag through a softmax layer:

$$p(y = h_{pi}|f; \theta_s) = \frac{exp({\theta_s^{h_{pi}}}^{\mathrm{T}}(W_f f + b_f))}{\sum_j^{|H_p|} exp({\theta_s^{h_{pj}}}^{\mathrm{T}}(W_f f + b_f))}, \tag{6}$$

where W_f, b_f and θ_s are parameters, H_p is the candidate hashtag set and h_{pi} is the i-th hashtag in H_p.

According to the scores from the last softmax layer, we can list a top-ranked recommended hashtags for each tweet.

3.3 Training

The training objective function in this work is:

$$J = \sum_{(t_p, D, h_p) \in S} -logp(h_p|t_p; D), \tag{7}$$

where h_p is the hashtag for tweet t_p and S is the training set.

The parameter list of our model is:

$$\theta = \{A, B, C, W_o, W_s, W_{ms}, W_f, b_f, \theta_s\}, \tag{8}$$

where A, B and C are three embedding matrix. W_o, W_s and W_{ms} are the parameters of attention layer in user interest encoder. W_f, b_f and θ_s are the parameters of the final predict layer.

In this study, we use stochastic gradient descent (SGD) with the adagrad update rule to optimise our model. Dropout regularization has proved to be an effective method for reducing the overfitting in deep neural networks with millions of parameters. In this work, we use it and add $l2$-norm regularization terms for the parameters of the network to augment the cost function.

Table 1: Statistics of the evalution dataset

#Tweets	#Users	#Hashtags	#Avg.Hashtag/Tweet
288,545	36,003	3,883	1.28

4 Experiment

In this section, we first introduce the data collection. Then, we describe the experiment configurations and baseline methods. Finally, the evaluation results and analyses are given.

4.1 Dataset and Setup

We started by using Twitter's API to collect public tweets from randomly selected users. In a first step, we randomly selected 40,000 users and crawled their tweets. In this step, we obtained 77,995,265 tweets. In the second step, we selected users with more than 50 tweets and filtered out the tweets whose language was not English. Then, we filtered out the hashtags whose frequencies were very low. Third, we extracted the tweets in the original corpus that contained hashtags.

Based on the statistics, there were 281,345 tweets in our evaluation collection, which belonged to 36,003 different users. The unique number of hashtags in the corpus was 3,883, and the average number of hashtags in each tweet was 1.28. The list of hashtags annotated by their users were treated as the ground truth. The detailed statistics are listed in Table 1. All of the tweets were processed by removing stopwords and special characters. In our experiment, we split the dataset into a training set, validation set, and test set. There were 232,378 tweets in the training set and 28,092 in the validation set. The remaining 20,875 tweets were in the test set.

In this work, the poster of each tweet had a history. For the user history, we assumed that the latest tweets posted could represent the user's current interests and could be stored in the memory. In this work, the memory capacity was restricted to the most recent 5 tweets posted by the users, the maximum length of the tweets was 52, any tokens out of this range were discarded and any hashtags occuring in the history had been removed. The embedding dimension of our model was set to 300, and the number of hops was set to 2 unless noted otherwise. This configuration was also used in the other methods described in the following paragraphs. The network was used for training for 60 epochs with early stopping. The learning rate was set to l = 0.01, and the dropout rate was 0.2.

The three metrics used in this experiment to evaluate the quality of our model were the precision, recall, and F1-score (denoted as P, R, and $F1$, respectively). The number of recommended hashtags for each tweet are denoted as k, where $k = \{1, 2, 3, 4, 5\}$ and the precision, Recall, and F1-score at the k result are denoted as $P@k$, $R@k$, and $F1@k$, respectively.

4.2 Baseline

In this section, to compare with our model, we select some effective methods as a baseline and introduce a degeneration model, described as follows:

- **NB**: To achieve the task, we convert the hashtag recommendation problem into a classification problem. We apply Naive Bayes to model the posterior probability of each hashtag using only the textual information of the tweets.

- **NB+H**: To assess the usefulness of the user interest information, the textual information and user history are given to Naive Bayes to recommend hashtags.

- **SVM**: We use the pre-trained word vector [1], which was trained using a portion of a Google News dataset containing 300-dimensional vectors for 3 million words using the continuous bag-of-words model, and sum them as the feature vector of the tweet as features, which are used to implement the support vector machine for the recommendation.

- **SVM+H**: The information of the user interest history is added to the features and the support vector machine is used to achieve the task.

- **TTM**: TTM was proposed by (Ding et al., 2013) for hashtag recommendation. The authors proposed a topical translation model to recommend hashtags, which only used the tweet content.

- **CNN-Attention**: CNN-Attention was proposed by (Gong and Zhang, 2016). It was a convolutional neural network architecture with an attention mechanism, and it was the state-of- the-art method for this task. In this paper, we compare our method with it.

- **MemN2N**: The user interest encoder in our model is a memory network architecture with a hierarchical attention mechanism. Now, we replace it with the end-to-end memory network method proposed by (Sukhbaatar et al., 2015) and compare it with our proposed model.

[1] https://code.google.com/archive/p/word2vec/

Table 2: Result of different methods on the evaluation collection

Method	Precision	Recall	F1
NB	0.123	0.098	0.109
NB+H	0.198	0.156	0.175
SVM	0.232	0.181	0.203
SVM+H	0.312	0.241	0.272
TTM	0.234	0.190	0.210
CNN-Attention	0.328	0.255	0.287
MemN2N	0.501	0.403	0.446
HMemN2N	0.538	0.436	0.482

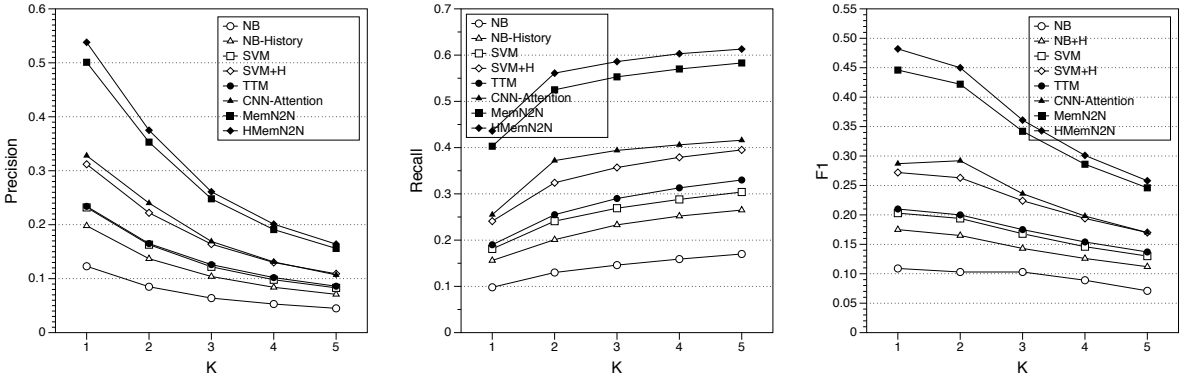


Figure 2: Precision, Recall, and F1-Score with different number of recommendation hashtags

4.3 Result and Discussion

In Table 2, we list the trend recommendation performances on our dataset using the different methods. The three metric results listed in Table 2 were obtained when we recommended the top hashtag for each tweet, i.e., $k = 1$.

In Table 2, we can see that the proposed method HMemN2N is better than all the other methods because it obtains the best result on all these metrics. Our approach provides improvements of 0.21 in precision, 0.181 in recall, and 0.195 in the F1-score over CNN-Attention, which is by far the state-of-the-art method for this task. Compared with the degeneration model MemN2N, our approach also shows a significant improvement. HMemN2N achieves a relative improvement of 7.4% in precision, 8.2% in recall, and 8.1% in the F1-score over MemN2N. The results show the practical applicability of our model, which can be used to provide users with good recommendations.

Observing the comparisons of the "NB" and the "NB+H", and the "SVM" and the "SVM+H", it is clear that the user interest history is a key ingredient in the recommendation, which is the strongest confirmation that much important information in the user history can be used to recommend hashtags in social media. This strongly suggests that we should find an effective method to extract useful information from the user history. In this paper, we provide a memory network architecture to solve this problem. The history is treated as an external memory, and a hierarchical attention mechanism is provided to help the model to extract the information, where the attention operation can be performed repeatedly and iteratively. The properties of this aspect of our proposed model have been proven to be effective by observing the results shown in Table 2.

CNN-Attention is the latest method used for this task, and it showed a good performance in this work. However, CNN-Attention only considers the content of the tweets and uses an attention mechanism to find important words in tweets. With a intuition that users will tend to repeatedly use the same hashtag, the performances of CNN-Attention and HMemN2N have a huge gap, which is mainly caused by not considering the user information. The results of the topical translation model (TTM) were obviously worse than those of our method, because it also only uses the textual information of the tweets.

Table 3: Parameter Influence on the evaluation collection

Model	embedding dim	# of hops	Precision	Recall	F1
CNN-Attention	300	-	0.328	0.255	0.287
MemN2N	300	2	0.501	0.403	0.446
	300	3	0.481	0.385	0.428
HMemN2N	300	1	0.521	0.421	0.466
	50	2	0.462	0.370	0.411
	100	2	0.508	0.410	0.454
	200	2	0.530	0.430	0.475
	300	2	0.538	0.436	0.482
	400	2	0.536	0.437	0.481
	500	2	0.538	0.437	0.482
	300	3	0.534	0.433	0.478
	300	4	0.528	0.428	0.473
	300	5	0.521	0.421	0.466

Considering the comparison between MemN2N and HMemN2N, the results show that it is necessary to introduce a hierarchical structure to store and select information. Each tweet has its own degree of importance in different recommendations, and each word in each tweet should also be given individual attention. Our model further utilizes an attention mechanism with a hierarchical structure to improve the information extraction. Compared to MemN2N, HMemN2N had superior performances across the board, which clearly demonstrated the effectiveness of the hierarchical mechanism.

In Figure 2, we list the result of models with different numbers of recommendation hashtags. The number of hashtags recommended k ranges from 1 to 5. From Figure 2, we can see that HMemN2N outperforms all of the baseline methods in all three metric curves with varying k. Clearly, the precision result decreases as k increases, and the recall result increases as k increases. The highest F1-score is obtained when we recommend the top 1 hashtag for each tweet. By analyzing the result, we can see that our model achieves the best performance in this task because all of the curves for HMemN2N are the highest in the graphs.

4.4 Parameter Influence

To evaluate the influence of the parameters used in our model, we changed the critical parameters to those in our dataset and list the results in Table 3. The effects of different hop numbers are shown, and the performances with different embedding dimensions are investigated.

From Table 3, we observe that changing the number of hops has some impact on the overall performance. However, the results disprove that more hops are better, because when the number of hops is larger than 2, the performances of HMemN2N and MemN2N are both decreasing.

The results listed in Table 3 also show the contribution of the embedding dimension to the performance. We fix the number of hops to 2 and vary the embedding dimension. A higher embedding dimension results in a better performance. When the dimension is low such as $dim = 50$ or $dim = 100$, the result is very poor. Our proposed model performs very well with a high embedding dimension. When the dimension is equal to 300, 400 or 500, we all can obtain the good performance. The size of the embedding dimension represents the expression ability of each word, and a higher dimension can enhance the text feature expression ability. To recommend a more appropriate hashtag, it is suggested to choose a high embedding dimension.

5 Conclusion

In this paper, we proposed a novel end-to-end memory network architecture that combines a tweets textual information and the corresponding user interest information for the hashtag recommendation task. The user interest history was a key ingredient of the recommendation and was adopted in this work. We

treated the user history as an external memory and proposed a novel memory network with a hierarchical attention mechanism to encode the user interest. To evaluate the proposed method, we collected data from real word twitter services. The experimental results on the evaluation dataset demonstrated that the proposed method could achieve better results than the current state-of-the-art methods for this task because they do not consider the user information.

Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011, 61473092, and 61472088), the National High Technology Research and Development Program of China (No. 2015AA015408), and IBM Faculty Award 2016.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Fabricio Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgilio Almeida. 2010. Detecting spammers on twitter. In *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, volume 6, page 12.

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *CoRR*, abs/1605.07683.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. End-to-end continuous speech recognition using attention-based recurrent NN: first results. *CoRR*, abs/1412.1602.

Zhuoye Ding, Xipeng Qiu, Qi Zhang, and Xuanjing Huang. 2013. Learning topical translation model for microblog hashtag suggestion. In *Proceedings of IJCAI 2013*.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. 2015. Evaluating prerequisite qualities for learning end-to-end dialog systems. *CoRR*, abs/1511.06931.

Fréderic Godin, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. 2013. Using topic models for twitter hashtag recommendation. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 593–596. ACM.

Yeyun Gong and Qi Zhang. 2016. Hashtag recommendation using attention-based convolutional neural network. In *IJCAI 2016, Proceedings of the 26rd International Joint Conference on Artificial Intelligence, New York City, USA., July 9-15, 2016*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Paul Heymann, Daniel Ramage, and Hector Garcia-Molina. 2008. Social tag prediction. In *SIGIR*.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285.

Su Mon Kywe, Tuan-Anh Hoang, Ee-Peng Lim, and Feida Zhu. 2012. On recommending hashtags in twitter networks. In *Social Informatics*, pages 337–350. Springer.

Zhiyuan Liu, Xinxiong Chen, and Maosong Sun. 2011. A simple word trigger method for social tag suggestion. In *Proceedings of EMNLP*, pages 1577–1588. Association for Computational Linguistics.

Surendra Sedhai and Aixin Sun. 2014. Hashtag recommendation for hyperlinked tweets. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, pages 831–834.

Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. 2015. Action recognition using visual attention. *CoRR*, abs/1511.04119.

Jieying She and Lei Chen. 2014. Tomoha: Topic model-based hashtag recommendation on twitter. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 371–372. ACM.

Bichen Shi, Georgiana Ifrim, and Neil J. Hurley. 2016. Learning-to-rank for real-time high-precision hashtag recommendation for streaming news. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 1191–1202.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2440–2448.

Oren Tsur and Ari Rappoport. 2012. What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 643–652. ACM.

Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. 2011. Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1031–1040. ACM.

Yuan Wang, Jishi Qu, Jie Liu, Jimeng Chen, and Yalou Huang. 2014. What to tag your microblog: Hashtag recommendation based on topic analysis and collaborative filtering. In *Web Technologies and Applications - 16th Asia-Pacific Web Conference, APWeb 2014, Changsha, China, September 5-7, 2014. Proceedings*, pages 610–618.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR*, abs/1410.3916.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698.

Jason Weston. 2016. Dialog-based language learning. *CoRR*, abs/1604.06045.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2048–2057.

Qi Zhang, Yeyun Gong, Xuyang Sun, and Xuanjing Huang. 2014. Time-aware personalized hashtag recommendation on social media. In *COLING*, pages 203–212.

Automatic Labelling of Topics with Neural Embeddings

Shraey Bhatia[1,2], **Jey Han Lau**[1,2] and **Timothy Baldwin**[2]
[1] IBM Research
[2] Dept of Computing and Information Systems,
The University of Melbourne
shraeybhatia@gmail.com, jeyhan.lau@gmail.com, tb@ldwin.net

Abstract

Topics generated by topic models are typically represented as list of terms. To reduce the cognitive overhead of interpreting these topics for end-users, we propose labelling a topic with a succinct phrase that summarises its theme or idea. Using Wikipedia document titles as label candidates, we compute neural embeddings for documents and words to select the most relevant labels for topics. Compared to a state-of-the-art topic labelling system, our methodology is simpler, more efficient, and finds better topic labels.

1 Introduction

Topic models are a popular approach to detecting trends and traits in document collections, e.g. in tracing the evolution of a scientific field through its publications (Hall et al., 2008), enabling visual navigation over search results (Newman et al., 2010a), interactively labelling document collections (Poursabzi-Sangdeh et al., 2016), or detecting trends in text streams (Wang and McCallum, 2006; AlSumait et al., 2008). They are typically unsupervised, and generate "topics" t_i in the form of multinominal distributions over the terms w_j of the document collection ($\Pr(w_j|t_i)$), and topic distributions for each document d_k in the collection, in the form of a multinomial distribution over topics ($\Pr(t_i|d_k)$). Traditionally, this has been carried out based on latent Dirichlet allocation (LDA: Blei et al. (2003)) or extensions thereof, but more recently there has been interest in deep learning approaches to topic modelling (Cao et al., 2015; Larochelle and Lauly, 2012).

In contexts where the output of the topic model is presented to a human user, a fundamental concern is the best way of presenting the rich information generated by the topic model, in particular, the topics themselves, which provide the primary insights into the document collection. The de facto topic representation has been a simple term list, in the form of the top-10 terms in a given topic, ranked in descending order of $\Pr(w_j|t_i)$. The cognitive overhead in interpreting the topic presented as a list of terms can be high, and has led to interest in the task of generating labels for topics, e.g. in the form of textual descriptions (Mei et al., 2007; Lau et al., 2011; Kou et al., 2015), visual representations of the topic words (Smith et al., to appear), or images (Aletras and Stevenson, 2013). In the former case, for example, rather than the top-10 terms of ⟨*school, student, university, college, teacher, class, education, learn, high, program* ⟩, a possible textual label could be simply EDUCATION. Recent work has shown that, in the context of a timed information retrieval (IR) task, automatically-generated textual labels are easier for humans to interpret than the top-10 terms, and lead to equivalent-quality relevance judgements (Aletras et al., 2014). Despite this, the accuracy of state-of-the-art topic generation methods is far from perfect, providing the motivation for this work.

In this paper, we propose an approach to topic labelling based on word and document embeddings, which both automatically generates label candidates given a topic input, and ranks the candidates in either an unsupervised or supervised manner, to produce the final topic label. Our contributions in this work are: (1) a label generation approach based on combined word and document embeddings, which is both considerably simpler than and empirically superior to the state-of-the-art generation method; (2) a

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 953–963, Osaka, Japan, December 11-17 2016.

simple label ranking approach that exploits character and lexical information, which is also superior to the state-of-the-art ranking approach; and (3) release of an open source implementation of our method, including a new dataset for topic label ranking evaluation.[1]

2 Related Work

Mei et al. (2007) introduced the task of generating labels for LDA topics, based on first extracting bigram collocations from the topic-modelled document collection using a lexical association measure, and then ranking them based on KL divergence with each topic. The approach is completely unsupervised.

Lau et al. (2011) proposed using English Wikipedia to automatically label topics. First, they map the topic to a set of concepts by querying Wikipedia using the top-10 topic terms based on: (a) Wikipedia's native search API; and (b) Google's search API, with site restriction. The top-8 article titles from each of these two sources are pooled to generate the primary candidate topic labels. Secondary labels are generated from component n-grams contained within the primary candidates, and filtering out incoherent and unrelated titles using the `RACO` measure (Grieser et al., 2011) to measure similarity with the primary labels, based on Wikipedia document categories. The combined set of primary and secondary label candidates is then ranked using a number of lexical association features, either directly in an unsupervised manner, or indirectly based on training a support vector regression model. The authors provide an extensive analysis of their method with that of Mei et al. (2007), and find their label generation and ranking methodology to be empirically superior (in both an unsupervised and supervised setting). In this paper, we seek to improve upon the topic labelling benchmark set by Lau et al. (2011).

Hulpus et al. (2013) developed a graph-based method for topic labelling, leveraging the structure of DBpedia concepts. Their approach is styled around graph-based word sense disambiguation, and extracts a set of DBpedia concepts corresponding to the top-N terms of a topic. They then construct a graph centred around DBpedia concepts and filter noise based on graph connectivity (the hypothesis being that sense graphs of words from a topic should be connected). To find the best label for a topic, they experiment with a variety of graph centrality measures.

In work slightly further afield, Zhao et al. (2011) proposed topical keyphrase extraction for Twitter. Although the work focuses mainly on Twitter, the methodology can be applied to other domains and to label topics. Zhao et al. (2011) follow a three-step process for keyphrase extraction: (1) keyword ranking; (2) candidate keyphrase generation (based on the individual keywords); and (3) keyphrase ranking. They use a novel topic context-sensitive *PageRank* method to regularise topic scores for keyword ranking, and a probabilistic scoring method that takes into account relevance, interestingness and keyphrase length for keyphrase ranking.

Not directly for the purposes of topic labelling but as a relevant pretraining method, Mikolov et al. (2013) proposed `word2vec` to learn word embeddings, which they found to perform strongly over a range of word similarity tasks, and also to be useful for initialising deep learning models. Two approaches are proposed in the paper: `cbow` and `skip-gram`. `cbow` combines neighbouring words to predict a target word, while `skip-gram` uses the target word to predict neighbouring words. Both approaches use a feedforward neural network with a non-linear hidden layer to maximize the objective function; to improve computational efficiency, the authors propose using negative sampling. In this paper, we will use `word2vec` as a means of generating topic term and label representations.

As an extension of `word2vec`, Le and Mikolov (2014) introduced `doc2vec` to learn embeddings for word sequences (e.g. paragraphs or documents). By treating each document as a word token, the same `word2vec` methodology can be used to learn document embeddings. The authors propose two implementations: `dbow`, which uses the document vector to predict its document words, and is the `doc2vec` equivalent of `skip-gram`; and `dmpv`, which uses a small window of words and concatenates them with the document vector to predict a document word, and is the `doc2vec` equivalent of `cbow`.[2] Compared to `dbow`, `dmpv` takes into account the local word ordering, and has a higher number of parameters, since the input is a

[1] `https://github.com/sb1992/NETL-Automatic-Topic-Labelling-`

[2] Strictly speaking, `cbow` combines word vectors by summing them, while `dmpv` combine word vectors and document vector by concatenating them.

concatenation of vectors. As with word2vec, we will use doc2vec as an alternative means of generating topic term and label representations.

Building on word2vec, Kou et al. (2015) experimented with neural embeddings in the context of topic labelling. In addition to skip-gram and cbow word vectors, the authors also included letter trigram vectors of a word, with the rationale that it generalises over morphologically-related forms of the same word. Their methodology consists of first generating candidate labels for topics from topic-related documents using a chunk parser. By representing both topic words and topic labels using word embeddings and letter trigrams, they rank the labels using cosine similarity to obtain the best label for a topic. In their evaluation, they find that simple letter trigrams are ultimately the most reliable means of label ranking.

3 Methodology

Following Lau et al. (2011), our method is made up of two steps: (1) topic label generation based on English Wikipedia; and (2) topic label ranking, based on a supervised learn-to-rank model. We detail each of these steps below.

3.1 Candidate Generation

To match topics to Wikipedia articles,[3] Lau et al. (2011) used an IR approach, by querying English Wikipedia with the top-N topic terms. However, in order to do this, they required external resources (two search APIs, one of which is no longer publicly available), limiting the general-purpose utility of the method. We propose an alternative approach: precomputing distributed representations of the topic terms and article titles using word2vec and doc2vec.

To this end, we train a doc2vec model on the English Wikipedia articles, and represent the embedding of a Wikipedia title by its document embedding. As doc2vec runs word2vec internally, word embeddings are also learnt during the training. Given the top-N topic terms, the topic embedding is represented by these terms' word embeddings. Based on the findings of Lau and Baldwin (2016) that the simpler dbow has less parameters, trains faster, and performs better than dmpv in several extrinsic tasks, we experiment only with dbow.[4] In terms of hyper-parameter settings, we follow the recommendations of Lau and Baldwin (2016).[5]

In addition to doc2vec, we also experiment with word2vec to generate embeddings for Wikipedia titles. By treating titles as a single token (e.g. concatenating *financial crisis* into *financial_crisis*) and greedily tokenising the text of all of the Wikipedia articles, we can then generate word embeddings for the titles. For word2vec, we use the skip-gram implementation exclusively.[6]

For both doc2vec and word2vec, we first pre-process English Wikipedia,[7] using Wiki Extractor to clean and extract Wikipedia articles from the original dump.[8] We then tokenise words with the Stanford CoreNLP Parser (Klein and Manning, 2003), and lowercase all words. We additionally filter out articles where the article body is made up of less than 40 words, and also disambiguation pages. We also remove titles whose length is longer than 4 words, as they are often too specific or inappropriate as topic labels (e.g. *List of Presidents of the United States*). For word2vec, we remove any parenthesised sub-component of an article title — e.g. in the case of *Democratic Party (United States)*, we remove *(United States)* to generate *Democratic Party* — as we would not expect to find verbatim usages of the full title. This has the potential side-effect of mapping multiple articles onto a single ambiguous title, resulting in multiple representations for *Democratic Party*. While acknowledging that there are instances where the more specific title may be appropriate as a label, the generalised version is always going to be a hypernym of the original, and thus appropriate as a label candidate.

[3] As of 2016 there are over 5 million documents in English Wikipedia.

[4] We use Gensim's implementation of both doc2vec and word2vec for all experiments: https://radimrehurek.com/gensim/.

[5] doc2vec hyper-parameters: sub-sampling threshold $= 10^{-5}$, vector size $= 300$, window size $= 15$, negative sample size $= 5$, and training epochs $= 20$.

[6] word2vec hyper-parameters: sub-sampling threshold $= 10^{-5}$, vector size $= 300$, window size $= 5$, negative sample size $= 5$, and training epochs $= 100$.

[7] The English Wikipedia dump used in all experiments is dated 2015-12-01.

[8] https://github.com/attardi/wikiextractor/

Top-10 Topic Terms	word2vec Labels	doc2vec Labels
	software	microsoft visual studio
blogs, vmware, server, virtual, oracle,	desktop	desktop virtualization
update, virtualization, application,	operating system	microsoft exchange server
infrastructure, management	virtualization	cloud computing
	middleware	windows server 2008

Table 1: The top-5 labels generated using doc2vec and word2vec title embeddings for the topic provided

Given a topic, we measure the relevance of each title embedding (generated by either doc2vec or word2vec) based on the pairwise cosine similarity with each of the word embeddings for the top-10 topic terms, and aggregate by taking the arithmetic mean. Formally, the doc2vec relevance (rel_{d2v}) and word2vec relevance (rel_{w2v}) of a title a and a topic T is given as follows:

$$rel_{d2v}(a,T) = \frac{1}{|T|}\sum_{v\in T}\cos\left(E^{d}_{d2v}(a), E^{w}_{d2v}(v)\right) \quad (1)$$

$$rel_{w2v}(a,T) = \frac{1}{|T|}\sum_{v\in T}\cos\left(E^{w}_{w2v}(a), E^{w}_{w2v}(v)\right) \quad (2)$$

where $E^{d}_{d2v}(x)$ is the document embedding of title x generated by doc2vec; $E^{w}_{d2v}(y)$ is the word embedding of word y generated by doc2vec; $E^{w}_{w2v}(z)$ is the word embedding of word z generated by word2vec; $v \in T$ is a topic term; $|T|$ is the number of topic terms (10 in our experiments); and $\cos(\vec{x},\vec{y})$ is the cosine similarity function.

The idea behind using both doc2vec and word2vec to generate title embeddings is that we observe that the two models favour different types of labels: doc2vec tends to favour fine-grained concepts, while word2vec favours more generic or abstract labels. As an illustration of this, in Table 1 we present one of the actual topics used later in our evaluation, and the top-5 article titles based on doc2vec and word2vec. This dichotomy is rooted in the differences in the modelling of context in the two models. In doc2vec, the title embedding is determined by the words that belong to the title, each of which is in turn determined by its context of use; it thus directly captures the compositional semantics of the title. With our word2vec method, on the other hand, the title embedding is determined directly by the neighbouring words of the title token in text, oblivious to the composition of words in the title.

To combine the strengths of doc2vec and word2vec, for each topic we generate a combined candidate ranking by summing the relevance scores using top-100 candidates from doc2vec and word2vec:[9]

$$rel_{d2v+w2v}(a,T) = rel_{d2v}(a,T) + rel_{w2v}(a,T) \quad (3)$$

3.2 Candidate Ranking

The next step after candidate generation is to re-rank them based on a supervised learn-to-rank model, in an attempt to improve the quality of the top-ranking candidates.

The first feature used in the supervised reranker is *LetterTrigram*, and based on the finding of Kou et al. (2015) that letter trigram vectors are an effective means of ranking topic labels. Our implementation of their method is based on measuring the overlap of letter trigrams between a given topic label and the topic words. For each topic, we first convert each topic label and topic words into multinomial distributions over letter trigrams, based on simple maximum likelihood estimation.[10] We then rank the

[9] From preliminary experiments we found that summing only the top-100 candidates from doc2vec and word2vec is better than summing all candidates. As we remove the parenthesised sub-component of an article title for word2vec (*Democratic Party (United States)* $\longrightarrow$ *Democratic Party*) , we observe that these titles tend to be very general and can ocassionally produce very high cosine similarity and skew the combined score for a number of similar labels (e.g. causing *Democratic Party* from a host of countries (*Democratic Party (United States)*, *Democratic Party (Australia)*, etc) to appear in the top ranking).

[10] For topic words, the letter trigrams are generated by parsing each of the topic words as separate strings rather than one concatenated string.

labels based on their cosine similarity with the topic words. The rank value constitutes the first feature of the supervised learn-to-rank model. Additionally, this ranking by letter trigram method also forms our unsupervised baseline, as we found that it to have the best unsupervised ranking performance of all our features, consistent with the findings of Kou et al. (2015).[11]

The second feature is *PageRank* (Page et al., 1998), in an attempt to prefer labels which represent more "core" concepts in Wikipedia. *PageRank* uses directed links to estimate the significance of a document, based on the probability of a random web surfer visiting a web page by either following hyperlinks or randomly transporting to a new page. We construct a directed graph from Wikipedia based on hyperlinks within the article text, and from this, compute a *PageRank* value for each Wikipedia article (and hence, title).[12]

Our last two features are lexical features proposed by Lau et al. (2011): (1) *NumWords*, which is simply the number of words in the candidate label (e.g. *operating system* has 2 words); and (2) *TopicOverlap*, which is the lexical overlap between the candidate label and the top-10 topic terms (e.g. *desktop virtualization* has a *TopicOverlap* score of 1 in our example from Table 1).

Given these features and a gold standard order of candidates (detailed in Section 4.1), we train a support vector regression model (SVR: Joachims (2006)) over these four features.

4 Datasets

For direct comparison with Lau et al. (2011), we use the same set of topics they used in their experiments. These were generated from 4 different domains: BLOGS, BOOKS, NEWS and PUBMED. In general, BLOGS, BOOKS and NEWS cover wide-ranging topics from product reviews to religion to finance and entertainment, whereas PubMed is medical-domain specific.

BLOGS is made up of 120k blog articles from the Spinn3r blog dataset; BOOKS is made up of 1k English language books from the Internet Archive American Libraries collection; NEWS is made up of 29k New York Times articles from English Gigaword; and PUBMED is made up of 77k PubMed biomedical abstracts. Lau et al. (2011) ran LDA on these documents and generated 100 topics for each domain. They filtered incoherent topics using an automated approach (Newman et al., 2010b), resulting in 45, 38, 60, 85 topics for BLOGS, BOOKS, NEWS and PUBMED, respectively.

4.1 Gold Standard Judgements

To evaluate our method and train the supervised model, gold-standard ratings of the candidates are required. To this end, we used CrowdFlower to collect human judgements.[13] We follow the approach of Lau et al. (2011), presenting 10 pairings of topic and candidate label, and asking human judges to rate the label on an ordinal scale of 0–3 where 0 indicates a completely inappropriate label, and 3 indicates a very good label for the given topic.

To control for annotation quality, we make use of the original annotations released by Lau et al. (2011). We select labels with a mean rating ≥ 2.5 (good labels) and ≤ 0.5 (bad labels) to serve as controls in our tasks. We include an additional topic–label control pair in addition to the 10 topic–label pairs in a HIT. Control pairs are selected randomly without replacement, and randomly injected into the HIT. To pass quality control, a worker is required to rate bad labels ≤ 1.0 and good labels ≥ 2.0. A worker is filtered out if his/her overall pass rate over all control pairs is < 0.75.

Each candidate label was rated by 10 annotators. Post-filtered, we have an average of 6.4 annotations for each candidate label.[14] To aggregate the ratings for a candidate label, we compute its mean rating, and rank the candidate labels based on the mean ratings to produce the gold standard ranking for each topic.

[11] Note that we do not make use of the noun chunk-based label generation methodology of Kou et al. (2015), in line with the findings of Lau et al. (2011) that Wikipedia titles give rise to better label candidates than n-grams extracted from the topic-modelled documents.

[12] We use the following implementation for *PageRank*: `https://www.nayuki.io/page/computing-wikipedias-internal-pageranks/`

[13] `https://www.crowdflower.com/`

[14] Post-filering, some candidates ended up with less than 3 annotations; these candidates were posted for another annotation round to gather more annotations.

Test Domain	Training	Top-1 Avg.		nDCG-1		nDCG-3		nDCG-5	
		LGNB	NETL	LGNB	NETL	LGNB	NETL	LGNB	NETL
BLOGS	Baseline	1.84	**1.91**	0.75	**0.77**	0.77	**0.82**	0.79	**0.83**
	In-Domain	1.98	**2.00**	0.81	0.81	0.82	**0.85**	0.83	0.84
	Cross-domain: BOOKS	1.88	**1.91**	0.77	0.78	0.81	**0.83**	0.83	0.83
	Cross-domain: NEWS	**1.97**	1.92	**0.80**	0.78	0.83	0.84	0.83	0.84
	Cross-domain: PUBMED	**1.95**	1.90	**0.80**	0.77	0.82	0.83	0.83	0.83
	Cross-domain: All 3	—	1.92	—	0.78	—	0.84	—	0.84
	Upper Bound	2.45	**2.48**	1.00	1.00	1.00	1.00	1.00	1.00
BOOKS	Baseline	1.75	**1.97**	0.77	0.78	0.77	**0.82**	0.79	**0.83**
	In-Domain	1.91	**1.99**	**0.84**	0.82	0.81	0.82	0.83	0.84
	Cross-domain: BLOGS	1.82	**2.02**	0.79	**0.83**	0.81	0.82	0.82	**0.84**
	Cross-domain: NEWS	1.82	**1.99**	0.79	**0.81**	0.81	0.82	0.83	0.84
	Cross-domain: PUBMED	1.87	**1.97**	0.81	0.80	0.82	0.82	0.83	0.84
	Cross-domain: All 3	—	2.03	—	0.83	—	0.83	—	0.84
	Upper Bound	2.29	**2.49**	1.00	1.00	1.00	1.00	1.00	1.00
NEWS	Baseline	1.96	**2.04**	0.80	**0.82**	0.79	**0.84**	0.78	**0.85**
	In-Domain	2.02	2.02	**0.82**	0.80	0.82	**0.84**	0.84	0.85
	Cross-domain: BLOGS	2.03	2.03	**0.83**	0.81	0.82	**0.84**	0.84	0.85
	Cross-domain: BOOKS	**2.01**	1.98	**0.82**	0.79	0.82	0.83	0.83	0.84
	Cross-domain: PUBMED	2.01	2.00	**0.82**	0.79	0.82	0.83	0.83	0.84
	Cross-domain: All 3	—	1.99	—	0.79	—	0.84	—	0.84
	Upper Bound	2.45	**2.56**	1.00	1.00	1.00	1.00	1.00	1.00
PUBMED	Baseline	1.73	**1.94**	0.75	**0.79**	0.77	**0.80**	0.79	**0.82**
	In-Domain	1.79	**1.99**	0.77	**0.81**	0.82	0.81	**0.84**	0.82
	Cross-domain: BLOGS	1.80	**1.98**	0.78	**0.80**	0.82	0.81	**0.84**	0.82
	Cross-domain: BOOKS	1.77	**1.98**	0.77	**0.80**	0.82	0.81	0.83	0.82
	Cross-domain: NEWS	1.79	**1.98**	0.77	**0.80**	0.82	0.81	**0.84**	0.82
	Cross-domain: All 3	—	2.01	—	0.81	—	0.81	—	0.82
	Upper Bound	2.31	**2.51**	1.00	1.00	1.00	1.00	1.00	1.00

Table 2: Results across the four domains. Boldface indicates the better system between NETL and LGNB (with an absolute difference > 0.01).

We collect judgements for the top-19 candidates from the unsupervised ranking.[15] For candidate ranking (Section 3.2), we are therefore re-ranking the top-19 candidates.

5 Experiments

In this section we present the results of our topic labelling experiments, and compare our method with that of Lau et al. (2011). Henceforth we refer to our method as "NETL" (neural embedding topic labelling), and Lau et al. (2011) as "LGNB".

Following LGNB, we use **top-1 average rating** and **normalized discounted cumulative gain (nDCG)** (Järvelin and Kekäläinen, 2002; Croft et al., 2009) as our evaluation metrics. Top-1 average computes the mean rating of the top-ranked labels, and provides an evaluation of the absolute utility of the preferred labels. nDCG, on the other hand, measures the relative quality of the ranking, calibrated relative to the ratings of the gold-standard ranking. Similarly to LGNB, we compute nDCG for the top-1 (nDCG-1), top-3 (nDCG-3), and top-5 (nDCG-5) ranked labels.

5.1 Results

Following LGNB, we present results for: (a) the unsupervised ranker (based on letter trigrams); (b) the supervised re-ranker in-domain, based on 10-fold cross validation, averaged over 10 runs with different

[15] Ideally we would have liked to have collected judgements for as many candidates as possible, but due to budget constraints we were only able to have the top-19 annotated.

Domain	Topic Terms	Label Candidate
BLOGS	vmware server virtual oracle update virtualisation application infrastructure management microsoft	virtualisation
BOOKS	church archway building window gothic nave side value tower	church architecture
NEWS	investigation fbi official department federal agent investigator charge attorney evidence	criminal investigation
PUBMED	rate population prevalence study incidence datum increase mortality age death	mortality rate

Table 3: A sample of topics and their topic labels generated by NETL.

partitionings; (c) the supervised re-ranker cross-domain; and (d) the upper bound, based on a perfect ranking of the candidates. For cross-domain learning, we train our model using one domain and test it on a different domain, or alternatively combine data from three domains and test on the remaining fourth domain, e.g. training on BOOKS +NEWS +PUBMED and testing on BLOGS. Cross-domain results give us a more accurate picture of the performance of our methodology in real-world applications (where it would be unrealistic to expect that there would be manual annotations of label candidates for that domain). We primarily use the in-domain results to gauge the relative quality of the cross-domain results.

We present the results in Table 2, displaying the performance of NETL and LGNB side by side for ease of comparison.[16] For each domain, the unsupervised baseline of NETL is based on the overlap of letter trigrams of the generated candidates with topic words (see Section 3.2). The unsupervised baseline of LGNB ranks the labels using a lexical association measure (Pearson's χ^2).

Looking at the performance of our method, we can see that the supervised system improves over the unsupervised baseline across all domains with the exception of a small drop observed in NEWS. Surprisingly, there is relatively little difference between the in-domain and cross-domain results for our method (but greater disparity for LGNB, especially over BOOKS; for NEWS, our cross-domain models actually outperform the in-domain model). The most consistent cross-domain results are generated when we combine all 3 domains, an unsurprising result given that it has access to the most training data, but encouraging in terms of having a single model which performs consistently across a range of domains.

We next compare NETL to LGNB, first focusing on the top-1 average rating metric. The most striking difference is the large improvement over PUBMED. LGNB attributed the poor performance over PUBMED to it being more domain-specific (and a poorer fit to Wikipedia) than the other domains, and suggested the need of biomedical experts for annotation. Our experiments found, however, that the performance of PUBMED is comparable to other domains. Additionally, the improvement in BOOKS is also quite substantial. Overall, NETL is more consistent across different domains and outperforms LGNB over 2 domains (BOOKS and PUBMED), and the difference between NETL and LGNB is small for NEWS and BLOGS. The other observation is the upper bound performance of NETL is uniformly better than that of LGNB, implying we are also generating better label candidates (we revisit this in detail in Section 5.2.1).

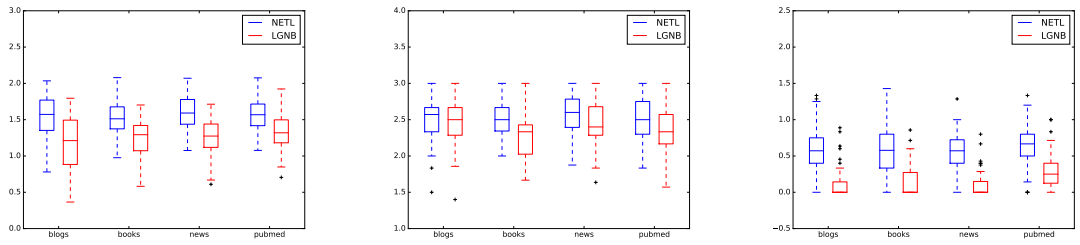
Moving to nDCG, the performance difference for nDCG-3/5 is largely indistinguishable for the two systems. LGNB, however, outperforms NETL in NEWS for nDCG-1 whereas NETL does better inPUBMED for nDCG-1 .

To give a sense of the sort of labels generated by NETL, we present a few topics and their top-ranked labels in Table 3.

5.2 Breaking Down NETL vs. LGNB

The results for NETL and LGNB in Table 2 conflate the candidate label selection and ranking steps, making it hard to get a sense of the relative impact of the different design choices implicit in the two sub-tasks. To provide a better comparison between the two methodologies, we present experiments

[16]LGNB results are taken directly from the original paper.



(a) Mean Ratings (b) Maximum Ratings (c) Minimum Ratings

Figure 1: Boxplots of ratings of candidates generated by NETL and LGNB.

evaluating the candidate generation and ranking method of the two systems separately.

5.2.1 Candidate Generation

In Table 2 we saw that NETL has a higher upper bound than that of LGNB, suggesting that the generated candidate labels were on average better. This is despite the average number of topic label candidates actually being higher for LGNB (25 vs. 19). Here, we present a more rigorous evaluation of the candidate generation method of both systems.

For each topic, we determine the mean, maximum and minimum label ratings for a given topic, and plot them in boxplots in Figure 1, aggregated per domain. The mean rating boxplot shows the average quality of candidates, while the maximum (minimum) rating boxplot reveals the average best (worst) quality of candidates that are generated by the two systems.

Looking at the boxplots, we see very clearly that NETL generates on average higher-quality candidates. Across all domains for mean, maximum and minimum ratings, the difference is substantial.

To provide a quantitative evaluation, we conduct one-sided paired t-tests to test the difference for all pairs in the boxplots. Except for the maximum ratings on BLOGS, all tests are significant ($p < 0.05$). These results demonstrate that NETL generates better candidates than LGNB (in all of the best-case, average-case and worst-case scenarios).

5.2.2 Candidate Ranking

Next, we directly compare the ranking method of NETL and LGNB. Using candidates generated by NETL, we re-rank the candidates using the ranking method of each of LGNB and NETL, and compare the results.

Both LGNB and NETL train an SVR re-ranker, using a partially-overlapping set of features. For LGNB, the ranking methodology uses 7 lexical association measures (PMI, Student's t-test, Dice's coefficient, Pearson's χ^2 test, log likelihood ratio, conditional and reverse conditional probability), 2 lexical features (the same 2 features that NETL uses: *NumWords* and *TopicOverlap*), and a search engine score based on Zettair. NETL, on the other hand, uses only 4 features: *LetterTrigram*, *PageRank*, *TopicOverlap* and *NumWords*.

For LGNB, we exclude the Zettair search engine score feature (as it was found to be an unimportant feature), and generate the lexical association features by parsing English Wikipedia. We train 2 SVR models using LGNB and NETL features. Results are presented in Table 4.

Using the same candidates, we see that NETL's features produce better rankings, outperforming LGNB's features across all domains. This shows that not only does NETL generate better candidates, but also ranks them better than LGNB.

6 Discussion

To better understand the contribution of each feature in NETL, we perform feature ablation tests (Table 5). An interesting observation is that different features appear to have different impact depending on the

Test Domain	Features	Top-1 Avg.	nDCG-1	nDCG-3	nDCG-5
BLOGS	LGNB	1.92	0.79	0.81	0.82
	NETL	**2.00**	**0.81**	**0.85**	**0.84**
BOOKS	LGNB	1.86	0.77	0.79	0.80
	NETL	**1.99**	**0.82**	**0.82**	**0.84**
NEWS	LGNB	1.87	0.75	0.79	0.81
	NETL	**2.02**	**0.80**	**0.84**	**0.85**
PUBMED	LGNB	1.89	0.77	0.79	0.81
	NETL	**1.99**	**0.81**	**0.81**	0.82

Table 4: Comparison of ranking performance with NETL features and LGNB features. Boldface indicates the better system between NETL and LGNB (with an absolute difference > 0.01).

Test Domain	BLOGS	BOOKS	NEWS	PUBMED
All Features	2.00	1.99	2.02	1.99
−*LetterTrigram*	1.99 (−.01)	1.96 (−.03)	2.02 (±.00)	1.93 (−.06)
−*PageRank*	1.93 (−.07)	1.980 (−.01)	2.00 (−.02)	2.00 (+.01)
−*TopicOverlap*	2.00 (±.00)	2.03 (+.04)	2.04 (+.02)	1.95 (−.04)
−*NumWords*	1.97 (−.03)	2.02 (+.03)	2.02 (±.00)	2.00 (+.01)

Table 5: Feature ablation results based on in-domain top-1 average ratings.

domain. Looking at BLOGS, we find that there is a considerable drop in top-1 average rating when we remove the *PageRank* feature. Similarly, ablating *LetterTrigram* appears to have a significant impact on PUBMED as well as some influence on BOOKS. As far as NEWS is concerned, we observe feature ablation does not play a big role. These observations indicate there is some degree of complementarity between these features, and that combining them produces robust and consistent performance across different domains.

Additionally, we explored using different numbers of topic terms when computing topic and title relevance for candidate ranking (we tested using top-5/10/15/20 topic terms). In general, we find that performance drops with the increase in topic terms. We also experiment with weighting each topic term with its word probability. We observed an improvement, although the difference is so marginal that we omit the results from the paper. Lastly, we tried computing relevance by first computing the centroid of topic terms before computing the cosine similarity with a candidate title. Again, we found little gain with this approach.

One feature type that we expect would have high utility is graph connectivity over the graphical structure of the Wikipedia categories or similar, along the lines of Hulpus et al. (2013). We leave this to future work. Methods based on keyphrase extraction such as Zhao et al. (2011) are also potentially worth exploring, although it remains to be seen whether notions such as "interestingness" benefit topic label selection.

7 Conclusion

We propose a neural embedding approach to automatically label topics using Wikipedia titles. Our methodology combines document and word embeddings to select the most relevant labels for topics. Compare to a state-of-the-art competitor system, our model is simpler, more efficient, and achieves better results across a range of domains.

References

Nikolaos Aletras and Mark Stevenson. 2013. Representing topics using images. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2013)*, pages 158–167, Atlanta, USA.

Nikolaos Aletras, Timothy Baldwin, Jey Han Lau, and Mark Stevenson. 2014. Representing topics labels for exploring digital libraries. In *Proceedings of Digital Libraries 2014*, London, UK.

Loulwah AlSumait, Daniel Barbará, and Carlotta Domeniconi. 2008. On-line LDA: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM-08)*, pages 3–12, Washington, DC, USA.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A novel neural topic model and its supervised extension. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2210–2216, Austin, USA.

W. Bruce Croft, Donald Metzler, and Trevor Strohman. 2009. *Search Engines: Information Retrieval in Practice*. Addison Wesley.

Karl Grieser, Timothy Baldwin, Fabian Bohnert, and Liz Sonenberg. 2011. Using ontological and document similarity to estimate museum exhibit relatedness. *ACM Journal of Computing and Cultural Heritage*, 3:10:1–10:20.

David Hall, Daniel Jurafsky, and Christopher D. Manning. 2008. Studying the history of ideas using topic models. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 363–371, Honolulu, USA.

Ioana Hulpus, Conor Hayes, Marcel Karnstedt, and Derek Greene. 2013. Unsupervised graph-based topic labelling using DBpedia. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, pages 465–474, Rome, Italy.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4).

Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226, Philadelphia, USA.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 423–430, Sapporo, Japan.

Wanqiu Kou, Li Fang, and Timothy Baldwin. 2015. Automatic labelling of topic models using word vectors and letter trigram vectors. In *Proceedings of the 11th Asian Information Retrieval Societies Conference (AIRS 2015)*, pages 229–240, Brisbane, Australia.

Hugo Larochelle and Stanislas Lauly. 2012. A neural autoregressive topic model. In *Advances in Neural Information Processing Systems 25*, pages 2708–2716.

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 78–86, Berlin, Germany.

Jey Han Lau, Karl Grieser, David Newman, and Timothy Baldwin. 2011. Automatic labelling of topic models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL HLT 2011)*, pages 1536–1545, Portland, USA.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, volume 14, pages 1188–1196, Beijing, China.

Qiaozhu Mei, Xuehua Shen, and Chengxiang Zhai. 2007. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2007)*, pages 490–499, San Jose, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

David Newman, Timothy Baldwin, Lawrence Cavedon, Sarvnaz Karimi, David Martinez, and Justin Zobel. 2010a. Visualizing document collections and search results using topic mapping. *Journal of Web Semantics*, 8(2–3):169–175.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010b. Automatic evaluation of topic coherence. In *Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2010)*, pages 100–108, Los Angeles, USA.

Larry Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The PageRank citation ranking: Bringing order to the web. Stanford Digital Libraries SIDL-WP-1999-0120.

Forough Poursabzi-Sangdeh, Jordan Boyd-Graber, Leah Findlater, and Kevin Seppi. 2016. ALTO: Active learning with topic overviews for speeding label induction and document labeling. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1158–1169, Berlin, Germany.

Alison Smith, Tak Yeon Lee, Forough Poursabzi-Sangdeh, Leah Findlater, Jordan Boyd-Graber, and Niklas Elmqvist. to appear. Evaluating visual representations for topic understanding and their effects on manually generated labels. *Transactions of the Association for Computational Linguistics*.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-Markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 424–433, Philadelphia, USA.

Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim, and Xiaoming Li. 2011. Topical keyphrase extraction from Twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies – Volume 1*, pages 379–388, Portland, USA.

Memory-Bounded Left-Corner Unsupervised Grammar Induction on Child-Directed Input

Cory Shain
The Ohio State University
shain.3@osu.edu

William Bryce
University of Illinois
at Urbana-Champaign
bryce2@illinois.edu

Lifeng Jin
The Ohio State University
jin.544@osu.edu

Victoria Krakovna
Harvard University
vkrakovna@fas.harvard.edu

Finale Doshi-Velez
Harvard University
finale@saes.harvard.edu

Timothy Miller
Boston Children's Hospital &
Harvard Medical School
timothy.miller@childrens.harvard.edu

William Schuler
The Ohio State University
schuler@ling.osu.edu

Lane Schwartz
University of Illinois
at Urbana-Champaign
lanes@illinois.edu

Abstract

This paper presents a new memory-bounded left-corner parsing model for unsupervised raw-text syntax induction, using unsupervised hierarchical hidden Markov models (UHHMM). We deploy this algorithm to shed light on the extent to which human language learners can discover hierarchical syntax through distributional statistics alone, by modeling two widely-accepted features of human language acquisition and sentence processing that have not been simultaneously modeled by any existing grammar induction algorithm: (1) a left-corner parsing strategy and (2) limited working memory capacity. To model realistic input to human language learners, we evaluate our system on a corpus of child-directed speech rather than typical newswire corpora. Results beat or closely match those of three competing systems.

1 Introduction

The success of statistical grammar induction systems (Klein and Manning, 2002; Seginer, 2007; Ponvert et al., 2011; Christodoulopoulos et al., 2012) seems to suggest that sufficient statistical information is available in language to allow grammar acquisition on this basis alone, as has been argued for word segmentation (Saffran et al., 1999). But existing grammar induction systems make unrealistic assumptions about human learners, such as the availability of part-of-speech information and access to an index-addressable parser chart, which are not independently cognitively motivated. This paper explores the possibility that a memory-limited incremental left-corner parser, of the sort independently motivated in sentence processing theories (Gibson, 1991; Lewis and Vasishth, 2005), can still acquire grammar by exploiting statistical information in child-directed speech.

2 Related Work

This paper bridges work on human sentence processing and syntax acquisition on the one hand and unsupervised grammar induction (raw-text parsing) on the other. We discuss relevant literature from each of these areas in the remainder of this section.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 964–975, Osaka, Japan, December 11-17 2016.

2.1 Human sentence processing and syntax acquisition

Related work in psycholinguistics and cognitive psychology has provided evidence that humans have a limited ability to store and retrieve structures from working memory (Miller, 1956; Cowan, 2001; McElree, 2001), and may therefore employ a left-corner-like strategy during incremental sentence processing (Johnson-Laird, 1983; Abney and Johnson, 1991; Gibson, 1991; Resnik, 1992; Stabler, 1994; Lewis and Vasishth, 2005). Schuler et al. (2010) show that nearly all naturally-occurring sentences can be parsed using no more than four disjoint derivation fragments in a left-corner parser, suggesting that general-purpose working memory resources are all that is needed to account for information storage and retrieval during online sentence processing. These findings motivate our left-corner parsing strategy and depth-bounded memory store.

An extensive literature indicates that memory abilities develop with age (see e.g. Gathercole, 1998 for a review). Newport (1990) proposed that limited processing abilities actually facilitate language acquisition by constraining the hypothesis space (the 'less-is-more' hypothesis). This theory has been supported by a number of subsequent computational and laboratory studies (e.g, Elman, 1993; Goldowski & Newport, 1993; Kareev et al., 1997) and parallels similar developments in the 'curriculum learning' training regimen for machine learning (Bengio et al., 2009).[1] Research on the acquisition of syntax has shown that infants are sensitive to syntactic structure (Newport et al., 1977; Seidl et al., 2003) and that memory limitations constrain the learning of syntactic dependencies (Santelman and Jusczyk, 1998). Together, these results suggest both (1) that the memory constraints in infants and young children are even more extreme than those attested for adults and (2) that these constraints impact – and may even facilitate – learning. By implementing these constraints in a domain-general computational model, we can explore the extent to which human learners might exploit distributional statistics during syntax acquisition (Lappin and Shieber, 2007).

2.2 Unsupervised grammar induction

The process of grammar induction learns the syntactic structure of a language from a sample of unlabeled text, rather than a gold-standard treebank. The constituent context model (CCM) (Klein and Manning, 2002) uses expectation-maximization (EM) to learn differences between observed and unobserved bracketings, and the dependency model with valence (DMV) (Klein and Manning, 2004) uses EM to learn distributions that generate child dependencies, conditioned on valence (left or right direction) in addition to the lexical head. Both of these algorithms induce on gold part-of-speech tag sequences.

A number of successful unsupervised raw-text syntax induction systems also exist. Seginer (2007) (CCL) uses a non-probabilistic scoring system and a dependency-like syntactic representation to bracket raw-text input. Ponvert et al. (2011) (UPPARSE) use a cascade of hidden Markov model (HMM) chunkers for unsupervised raw-text parsing. Christodoulopoulos et al. (2012) (BMMM+DMV) induce part-of-speech (PoS) tags from raw text using the Bayesian multinomial mixture model (BMMM) of Christodoulopoulos et al. (2011), induce dependencies from those tags using DMV, and iteratively re-tag and reparse using the induced dependencies as features in the tagging process. In contrast to ours, none of these systems employ a left-corner parsing strategy or model working memory limitations.

3 Methods

Experiments described in this paper use a memory-bounded probabilistic sequence model implementation of a left-corner parser (Aho and Ullman, 1972; van Schijndel et al., 2013) to determine whether natural language grammar can be acquired on the basis of statistics in transcribed speech within human-like memory constraints. The model assumes access to episodic memories of training sentences, but imposes constraints on working memory usage during sentence processing. The core innovation of this paper is the adaptation of this processing model to Bayesian unsupervised induction using constrained priors.

[1] The 'less-is-more' hypothesis has been a subject of controversy, however. See e.g. Rohde and Plaut (2003) for a critical review.

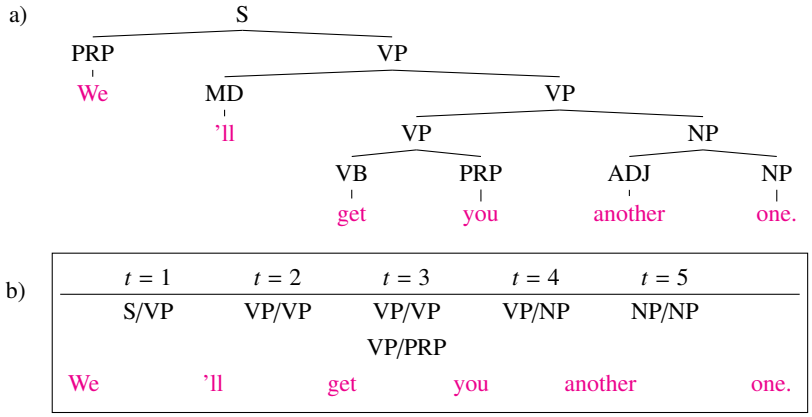


Figure 1: Trees and partial analyses for the sentence '*We'll get you another one*', taken from the training corpus. Derivation fragments are shown vertically stacked between words, using '/' to delimit top and bottom signs.

3.1 Left-corner parsing

Left-corner parsing is attractive as a sentence processing model because it maintains a very small number of disjoint derivation fragments during processing (Schuler et al., 2010), in keeping with human working memory limitations (Miller, 1956; Cowan, 2001; McElree, 2001), and correctly predicts difficulty in recognizing center-embedded, but not left- or right-embedded structures (Chomsky and Miller, 1963; Miller and Isard, 1964; Karlsson, 2007). A left-corner parser maintains a sequence of derivation fragments $a/b, a'/b', \ldots$, each consisting of an active category a lacking an awaited category b yet to come. It incrementally assembles trees by forking off and joining up these derivation fragments, using a pair of binary decisions about whether to use a word w to start a new derivation fragment (initially a complete category c):[2]

$$\frac{a/b \quad w}{a/b \quad c} \; b \xrightarrow{+} c \; \ldots \; ; \quad c \rightarrow w \tag{F=1}$$

$$\frac{a/b \quad w}{c} \; a = c; \; b \rightarrow w \tag{F=0}$$

and whether to use a grammatical inference rule to connect a complete category c to a previously disjoint derivation fragment a/b:

$$\frac{a/b \quad c}{a/b'} \; b \rightarrow c \; b' \tag{J=1}$$

$$\frac{a/b \quad c}{a/b \quad a'/b'} \; b \xrightarrow{+} a' \; \ldots \; ; \quad a' \rightarrow c \; b' \tag{J=0}$$

These two binary decisions have four possible outcomes in total: the parser can fork only (which increases the number of derivation fragments by one), join only (which decreases the number of derivation fragments by one), both fork and join (which keeps the number of derivation fragments the same), or neither fork nor join (which also preserves the number of derivation fragments).

An example derivation of the sentence '*We'll get you another one,*' in shown in Figure 1.

3.2 Probabilistic sequence model

A left-corner parser can be modeled as a probabilistic sequence model using hidden random variables at every time step for *Active* categories A, *Awaited* categories B, *Preterminal* or part-of-speech (POS) tags P, and an observed random variable W over *Words*. The model also makes use of two binary switching

[2]Here, $b \xrightarrow{+} c \; \ldots$ constrains c to be a leftmost descendant of b at some depth.

variables at each time step, F (for *Fork*) and J (for *Join*) that guide the transitions of the other states. These two binary switching variables yield four cases: 1/1, 1/0, 0/1 and 0/0 at each time step.

Let D be the depth of the memory store at position t in the sequence, and let the state $q_t^{1..D}$ be the stack of derivation fragments at t, consisting of one active category a_t^d and one awaited category b_t^d at each depth d. The joint probability of the hidden state $q_t^{1..D}$ and observed word w_t, given their previous context, are defined using Markov independence assumptions and the fork-join variable decomposition of van Schijndel et al. (2013), which preserves PCFG probabilities in incremental sentence processing:

$$\begin{aligned}
\mathsf{P}(q_t^{1..D}\ w_t \mid q_{1..t-1}^{1..D}\ w_{1..t-1}) &= \mathsf{P}(q_t^{1..D}\ w_t \mid q_{t-1}^{1..D}) && (1)\\
&\stackrel{\text{def}}{=} \mathsf{P}(p_t\ w_t\ f_t\ j_t\ a_t^{1..D}\ b_t^{1..D} \mid q_{t-1}^{1..D}) && (2)\\
&= \mathsf{P}_{\theta_P}(p_t \mid q_{t-1}^{1..D}) \cdot \\
&\quad \mathsf{P}_{\theta_W}(w_t \mid q_{t-1}^{1..D}\ p_t) \cdot \\
&\quad \mathsf{P}_{\theta_F}(f_t \mid q_{t-1}^{1..D}\ p_t\ w_t) \cdot \\
&\quad \mathsf{P}_{\theta_J}(j_t \mid q_{t-1}^{1..D}\ p_t\ w_t\ f_t) \cdot \\
&\quad \mathsf{P}_{\theta_A}(a_t^{1..D} \mid q_{t-1}^{1..D}\ p_t\ w_t\ f_t\ j_t) \cdot \\
&\quad \mathsf{P}_{\theta_B}(b_t^{1..D} \mid q_{t-1}^{1..D}\ p_t\ w_t\ f_t\ j_t\ a_t^{1..D}) && (3)
\end{aligned}$$

The part-of-speech p_t only depends on the lowest awaited (b_{t-1}^d) category at the previous time step, where d is the depth of the stack at the previous time step and $q_\perp$ is an empty derivation fragment:

$$\mathsf{P}_{\theta_P}(p_t \mid q_{t-1}^{1..D}) \stackrel{\text{def}}{=} \mathsf{P}_{\theta_P}(p_t \mid d\ b_{t-1}^d);\ \ d = \max_{d'}\{q_{t-1}^{d'} \neq q_\perp\} \tag{4}$$

The lexical item (w_t) only depends on the part of speech tag (p_t) at the same time step:

$$\mathsf{P}_{\theta_W}(w_t \mid q_{t-1}^{1..D}\ p_t) \stackrel{\text{def}}{=} \mathsf{P}_{\theta_W}(w_t \mid p_t) \tag{5}$$

The fork decision f_t is assumed to be independent of previous state $q_{t-1}^{1..D}$ variables except for the previous lowest awaited category b_{t-1}^d and part of speech tag p_t:

$$\mathsf{P}_{\theta_F}(f_t \mid q_{t-1}^{1..D}\ p_t\ w_t) \stackrel{\text{def}}{=} \mathsf{P}_{\theta_F}(f_t \mid d\ b_{t-1}^d\ p_t);\ \ d = \max_{d'}\{q_{t-1}^{d'} \neq q_\perp\} \tag{6}$$

The join decision j_t is decomposed into fork and no-fork cases depending on the outcomes of the fork decision:

$$\mathsf{P}_{\theta_J}(j_t \mid q_{t-1}^{1..D}\ f_t\ p_t\ w_t) \stackrel{\text{def}}{=} \begin{cases} \mathsf{P}_{\theta_J}(j_t \mid d\ a_{t-1}^d\ b_{t-1}^{d-1});\ \ d = \max_{d'}\{q_{t-1}^{d'} \neq q_\perp\} & \text{if } f_t = 0 \\ \mathsf{P}_{\theta_J}(j_t \mid d\ p_t\ b_{t-1}^d);\ \ d = \max_{d'}\{q_{t-1}^{d'} \neq q_\perp\} & \text{if } f_t = 1 \end{cases} \tag{7}$$

When f_t=1, that is, a fork has been created, the decision of j_t is whether to immediately integrate the newly forked derivation fragment and transition the awaited category above it (j_t=1) or keep the newly forked derivation fragment (j_t=0). When f_t=0, that is, no fork has been created, the decision of j_t is whether to reduce a stack level (j_t=1) or to transition both the active and awaited categories at the current level (j_t=0).

Decisions about the active categories $a_t^{1..D}$ are decomposed into fork- and join-specific cases depending on the previous state $q_{t-1}^{1..D}$ and the current preterminal p_t. Since the fork and join outcomes only allow a single derivation fragment to be initiated or integrated, each case of the active category model only nondeterministically modifies at most one a_t^d variable from the previous time step:[3]

$$\mathsf{P}_{\theta_A}(a_t^{1..D} \mid q_{t-1}^{1..D}\ f_t\ p_t\ w_t\ j_t) \stackrel{\text{def}}{=}$$

$$\begin{cases} [\![a_t^{1..d-2} = a_{t-1}^{1..d-2}]\!] \cdot [\![a_t^{d-1} = a_{t-1}^{d-1}]\!] \cdot [\![a_t^{d+0..D} = a_\perp]\!];\ \ d = \max_{d'}\{q_{t-1}^{d'} \neq q_\perp\} & \text{if } f_t = 0,\ j_t = 1 \\ [\![a_t^{1..d-1} = a_{t-1}^{1..d-1}]\!] \cdot \mathsf{P}_{\theta_A}(a_t^d \mid d\ b_{t-1}^{d-1}\ a_{t-1}^d) \cdot [\![a_t^{d+1..D} = a_\perp]\!];\ \ d = \max_{d'}\{q_{t-1}^{d'} \neq q_\perp\} & \text{if } f_t = 0,\ j_t = 0 \\ [\![a_t^{1..d-1} = a_{t-1}^{1..d-1}]\!] \cdot [\![a_t^{d} = a_{t-1}^{d}]\!] \cdot [\![a_t^{d+1..D} = a_\perp]\!];\ \ d = \max_{d'}\{q_{t-1}^{d'} \neq q_\perp\} & \text{if } f_t = 1,\ j_t = 1 \\ [\![a_t^{1..d-0} = a_{t-1}^{1..d-0}]\!] \cdot \mathsf{P}_{\theta_A}(a_t^{d+1} \mid d\ b_{t-1}^d\ p_t) \cdot [\![a_t^{d+2..D} = a_\perp]\!];\ \ d = \max_{d'}\{q_{t-1}^{d'} \neq q_\perp\} & \text{if } f_t = 1,\ j_t = 0 \end{cases} \tag{8}$$

[3]Here $[\![\phi]\!]$ is a (deterministic) indicator function, equal to one when ϕ is true and zero otherwise.

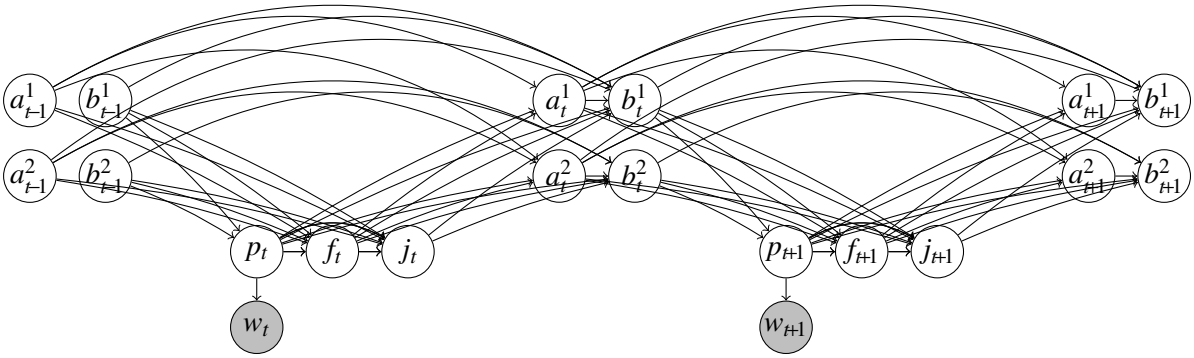


Figure 2: Graphical representation of probabilistic left-corner parsing model expressed in Equations 6–9 across two time steps, with $D = 2$.

Decisions about the awaited categories $b_t^{1..D}$ also depend on the outcome of the fork and join variables. Again, since the fork and join outcomes only allow a single derivation fragment to be initiated or integrated, each case of the awaited category model only nondeterministically modifies at most one b_t^d variable from the previous time step:

$$\mathsf{P}_{\theta_B}(b_t^{1..D} \mid q_{t-1}^{1..D}\ f_t\ p_t\ w_t\ j_t\ a_t^{1..D}) \stackrel{\text{def}}{=}$$

$$\begin{cases} [\![b_t^{1..d-2} = b_{t-1}^{1..d-2}]\!] \cdot \mathsf{P}_{\theta_B}(b_t^{d-1} \mid d\ b_{t-1}^{d-1}\ a_{t-1}^{d}) \cdot [\![b_t^{d+0..D} = b_\perp]\!]; \ d = \max_{d'}\{q_{t-1}^{d'} \neq q_\perp\} & \text{if } f_t = 0,\ j_t = 1 \\ [\![b_t^{1..d-1} = b_{t-1}^{1..d-1}]\!] \cdot \mathsf{P}_{\theta_B}(b_t^{d} \mid d\ a_t^{d}\ a_{t-1}^{d}) \cdot [\![b_t^{d+1..D} = b_\perp]\!]; \ d = \max_{d'}\{q_{t-1}^{d'} \neq q_\perp\} & \text{if } f_t = 0,\ j_t = 0 \\ [\![b_t^{1..d-1} = b_{t-1}^{1..d-1}]\!] \cdot \mathsf{P}_{\theta_B}(b_t^{d} \mid d\ b_{t-1}^{d}\ p_t) \cdot [\![b_t^{d+1..D} = b_\perp]\!]; \ d = \max_{d'}\{q_{t-1}^{d'} \neq q_\perp\} & \text{if } f_t = 1,\ j_t = 1 \\ [\![b_t^{1..d-0} = b_{t-1}^{1..d-0}]\!] \cdot \mathsf{P}_{\theta_B}(b_t^{d+1} \mid d\ a_t^{d+1}\ p_t) \cdot [\![b_t^{d+2..D} = b_\perp]\!]; \ d = \max_{d'}\{q_{t-1}^{d'} \neq q_\perp\} & \text{if } f_t = 1,\ j_t = 0 \end{cases} \tag{9}$$

Thus, the parser has a fixed number of probabilistic decisions to make as it encounters each word, regardless of the depth of the stack. A graphical representation of this model is shown in Figure 2.

3.3 Model priors

Induction in this model follows the approach of Van Gael et al. (2008) by applying nonparametric priors over the active, awaited, and part-of-speech variables. This approach allows the model to learn not only the parameters of the model—such as what parts of speech are likely to be created from what awaited categories—but also the cardinality of how many active, awaited, and part of speech categories are present, in a fully unsupervised fashion. No labels are needed for inference, which alternates between inferring these unseen categories and the associated model parameters.

The probabilistic sequence model defined above, augmented with priors, can be repeatedly sampled to obtain an estimate of the posterior distribution of its hidden variables given a set of observed word sequences. Priors over the syntactic models are based on the infinite hidden Markov model (iHMM) used for part-of-speech tagging (van Gael et al., 2009). In that model, a hierarchical Dirichlet process HMM (Teh et al., 2006) is used to allow the observed number of states—corresponding to parts of speech—in the HMM to grow as the data requires. The hierarchical structure of the iHMM ensures that transition distributions share the same set of states, which would not be possible if we used a flat infinite mixture model.

A fully infinite version of this model uses nonparametric priors on each of the active, awaited, and part-of-speech variables, allowing the cardinality of each of these variables to grow as the data requires. Each model draws a base distribution from a root Dirichlet process, which is then used as a parameter to an infinite set of Dirichlet processes, one each for each applicable combination of the conditioning

variables a_{t-1}, b_{t-1}, p_{t-1}, j_t, f_t, a_t, and b_t:

$$\beta_A \sim GEM(\gamma_A) \tag{10}$$

$$\mathsf{P}_{\theta_A}(a_t^d \mid d\ b_{t-1}^{d-1}\ a_{t-1}^d) \sim DP(\alpha_A, \beta_A) \tag{11}$$

$$\mathsf{P}_{\theta_A}(a_t^{d+1} \mid d\ b_{t-1}^d\ p_t) \sim DP(\alpha_A, \beta_A) \tag{12}$$

$$\beta_B \sim GEM(\gamma_B) \tag{13}$$

$$\mathsf{P}_{\theta_B}(b_t^{d-1} \mid d\ b_{t-1}^{d-1}\ a_{t-1}^{d-1}) \sim DP(\alpha_B, \beta_B) \tag{14}$$

$$\mathsf{P}_{\theta_B}(b_t^d \mid d\ a_t^d\ a_{t-1}^d) \sim DP(\alpha_B, \beta_B) \tag{15}$$

$$\mathsf{P}_{\theta_B}(b_t^d \mid d\ b_{t-1}^d\ p_t) \sim DP(\alpha_B, \beta_B) \tag{16}$$

$$\mathsf{P}_{\theta_B}(b_t^{d+1} \mid d\ a_t^{d+1}\ p_t) \sim DP(\alpha_B, \beta_B) \tag{17}$$

$$\beta_P \sim GEM(\gamma_P) \tag{18}$$

$$\mathsf{P}_{\theta_P}(p_t \mid d\ b_{t-1}^d) \sim DP(\alpha_P, \beta_P) \tag{19}$$

where DP is Dirichlet process and GEM is the stick-breaking construction for DPs (Sethuraman, 1994). Models at depth greater than one use the corresponding model at the previous depth as a prior.

3.4 Inference

Inference is based on the beam sampling approach employed in van Gael et al. (2009) for part-of-speech induction. This inference approach alternates between two phases in each iteration. First, given the distributions θ_F, θ_J, θ_A, θ_B, θ_P, and θ_W, the model resamples values for all the hidden states $\{q_t^d, p_t\}$. Next, given the state values $\{q_t^d, p_t\}$, it resamples each set of multinomial distributions θ_F, θ_J, θ_A, θ_B, θ_P, and θ_W. The sampler is initialized by conservatively setting the cardinalities of the number of active, awaited, and part-of-speech states we expect to see in the data set, randomly initializing the state space, and then sampling the parameters for each distribution θ_F, θ_J, θ_A, θ_B, θ_P, and θ_W given the randomly initialized states and fixed hyperparameters.

As noted by Van Gael et al. (2008), token-level Gibbs sampling in a sequence model can be slow to mix. Preliminary work found that mixing with token-level Gibbs sampling is even slower in this model due to the tight constraints imposed by the switching variables—it is technically ergodic but exploring the state space requires many low probability moves. Therefore, the experiments described in this paper use sentence-level sampling instead of token-level sampling, first computing forward probabilities for the sequence and then doing sampling in a backwards pass; resampling the parameters for the probability distributions only requires computing the counts from the sampled sequence and combining with the hyperparameters. To account for the infinite size of the state spaces, these experiments employ the beam sampler (Van Gael et al., 2008), with some modifications for computational speed.

The standard beam sampler introduces an auxiliary variable u at each time step, which acts as a threshold below which transition probabilities are ignored. This auxiliary variable u is drawn from $Uniform(0, p(q_t^{1..D}|q_{t-1}^{1..D}))$, so it will be between 0 and the probability of the previously sampled transition. The joint distribution over transitions, emissions, and auxiliary variables can be reduced so that the transition matrix is transformed into a boolean matrix with a 1 indicating an allowed transition. Depending on the cut-off value u, the size of the instantiated transition matrix will be different for every time-step.

Values of u can be sampled for active, awaited, and POS variables at every time step, rather than a single u for the transition matrix. It is possible to compile all the operations at each time step into a single large transition matrix, but computing this matrix is prohibitively slow for an operation that must be done at each time step in the data.

To address this issue, the learner may interleave several iterations holding the cardinality of the instantiated space fixed with full beam-sampling steps in which the cardinality of the state space can change.

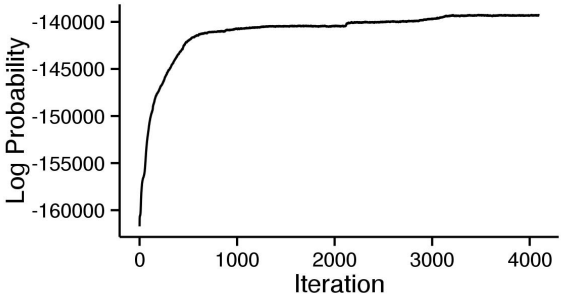


Figure 3: Log Probability (with punc)

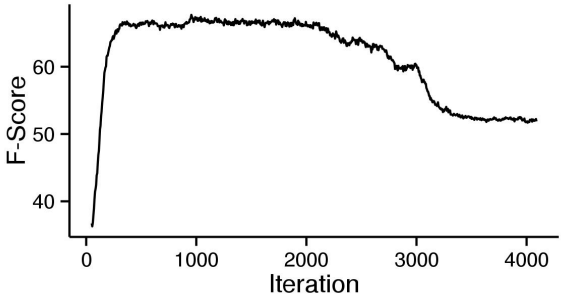


Figure 4: F-Score (with punc)

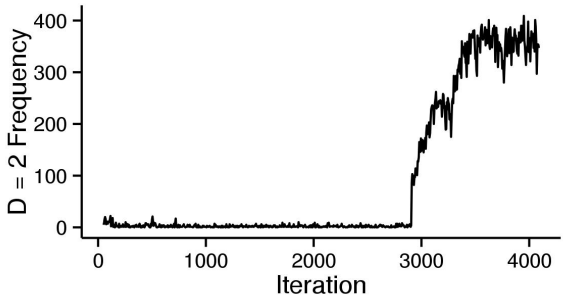


Figure 5: Depth=2 Frequency (with punc)

When the cardinality of the state space is fixed, the learner can multiply out the states into one large, structured transition matrix that is valid for all time steps. The forward pass is thus reduced to an HMM forward pass (albeit one over a much larger set of states), vastly improving the speed of inference. Alternating between sampling the parameters of this matrix and the state values themselves corresponds to updating a finite portion of the infinite possible state space; by interleaving these finite steps with occasional full beam-sampling iterations, the learner is still properly exploring the posterior over models.

3.5 Parsing

There are multiple ways to extract parses from an unsupervised grammar induction system such as this. The optimal Bayesian approach would involve averaging over the values sampled for each model across many iterations, and then use those models in a Viterbi decoding parser to find the best parse for each sentence. Alternatively, if the model parameters have ceased to change much between iterations, the learner can be assumed to have found a local optimum. It can then use a single sample from the end of the run as its model and the analyses of each sentence in that run as the parses to be evaluated. This latter method is used in the experiments described below.

4 Experimental Setup

We ran the UHHMM learner for 4,000 iterations on the approximately 14,500 child-directed utterances of the Eve section of the Brown corpus from the CHILDES database (MacWhinney, 2000).[4] To model the limited memory capacity of young language learners, we restricted the depth of the store of derivation fragments to two.[5] The input sentences were tokenized following the Penn Treebank convention and converted to lower case. Punctuation was initially left in the input as a proxy for intonational phrasal cues (Seginer, 2007; Ponvert et al., 2011), then removed in a follow-up experiment.

[4]We used 4 active states; 4 awaited states; 8 parts of speech; and parameter values 0.5 for α_a, α_b, and α_c, and 1.0 for α_f, α_j, and γ. The burnin period was 50 iterations.

[5]This limited stack depth permits discovery of interesting syntactic features – like subject-aux inversion – while modeling the severe memory limitations of infants (see §2.1). Greater depths are likely unnecessary to parse child-directed input (e.g., Newport et al., 1977).

	With punc			No punc		
	P	R	F1	P	R	F1
UPPARSE	60.50	51.96	55.90	38.17	48.38	42.67
CCL	64.70	53.47	58.55	56.87	47.69	51.88
BMMM+DMV (directed)	62.08	62.51	62.30	61.01	59.24	60.14
BMMM+DMV (undirected)	63.63	**64.02**	**63.82**	61.34	**59.33**	**60.32**
UHHMM-4000, binary	46.68	58.28	51.84	37.62	46.97	41.78
UHHMM-4000, flattened	**68.83**	57.18	62.47	**61.78**	45.52	52.42
Right-branching	68.73	**85.81**	**76.33**	**68.73**	**85.81**	**76.33**

Table 1: Parsing accuracy on Eve with and without punctuation (phrasal cues) in the input. The UHHMM systems were given 8 PoS categories while the BMMM+DMV systems were given 45. UPPARSE and CCL do not learn PoS tags. Only the UHHMM systems model limited working memory capacity or incremental left-corner parsing.

To generate accuracy benchmarks, we parsed the same data set using the three competing raw-text induction systems discussed in §2: CCL (Seginer, 2007), UPPARSE (Ponvert et al., 2011),[6] and both directed and undirected variants of BMMM+DMV (Christodoulopoulos et al., 2012).[7] The BMMM+DMV system generates dependency graphs which are not directly comparable to our phrase-structure output, so we used the algorithm of Collins et al. (1999) to convert the BMMM+DMV output to the flattest phrase structure trees permitted by the dependency graphs.

We evaluated accuracy against hand-corrected gold-standard Penn Treebank-style annotations for Eve (Pearl and Sprouse, 2013). All evaluations were of unlabeled bracketings with punctuation removed.[8] Accuracy results reported for our system are extracted from arbitrary samples taken after convergence had been reached: iteration 4000 for the with-punc model, and iteration 1500 for the no-punc model (see Figures 3 and 6, respectively).

5 Results

Figures 3, 4, and 5 show (respectively) log probability, f-score, and depth=2 frequency by iteration for the UHHMM trained on data containing punctuation. As the figures show, the model remains effectively depth 1 until around iteration 3000, at which point it discovers depth 2, rapidly overgeneralizes it, then scales back to around 350 uses over the entire corpus. Around this time, parsing accuracy drops considerably. This result is consistent with the 'less-is-more' hypothesis (Newport, 1990), since accuracy decreases near the point when the number of plausible hypotheses suddenly grows. In our system, we believe this is because the model reallocates probability mass to deeper parses. Nonetheless, as we show below, final results are state of the art.

We sampled parses from iteration 4000 of our learner for evaluation. As shown in Table 1, initial accuracy measures are worse than all four competitors. However, our system generates exclusively binary-branching output, while all competitors can produce the higher arity trees attested in the PTB-like evaluation standard (notice that our recall measure for the binary branching output beats both CCL and UPPARSE). To correct this disadvantage, we flattened the UHHMM output by first converting binary trees to dependencies using a heuristic that selects for each parent the most frequently co-occurring child category as the head, then converting these dependencies back into phrase structures using the Collins et al. (1999) algorithm. As shown in Table 1, recall remains approximately the same while precision predictably improves, resulting in higher overall F-measures that beat or closely match those of all competing systems.[9]

[6]Using the best cascaded parser settings from that work: probabilistic right-linear grammar with uniform initialization.

[7]We ran both variants of the BMMM+DMV system for 10 generations, with 500 iterations of BMMM and 20 EM iterations of DMV per generation, as was done by Christodoulopoulos et al. (2012).

[8]Note that while punctuation was removed for all evaluations, inclusion/removal of punctuation in the training data was an independent variable in our experiment.

[9]It happens to be the case that these child-directed sentences are heavily right-branching, likely due to the simplicity and

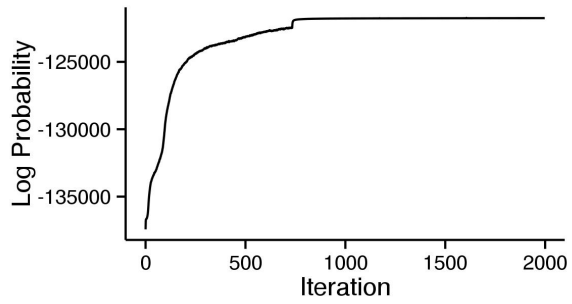


Figure 6: Log Probability (no punc)

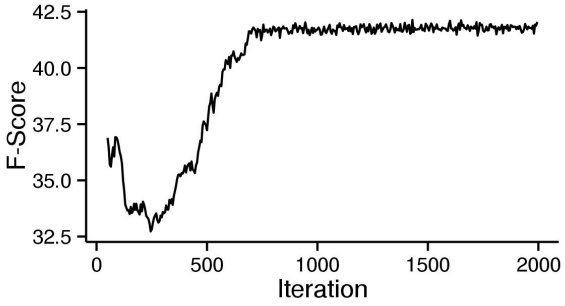


Figure 7: F-Score (no punc)

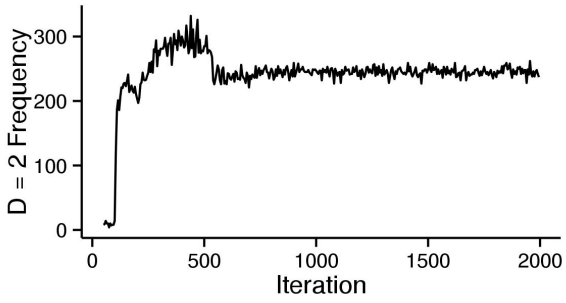


Figure 8: Depth=2 Frequency (no punc)

Figures 6, 7, and 8 show (respectively) log probability, f-score, and depth=2 frequency by iteration for the UHHMM trained on data containing no punctuation. Somewhat surprisingly, the model discovers depth 2 and converges much more quickly than it did for the with-punc corpus, requiring fewer than 1000 iterations to converge. This is possibly due to the slight reduction in corpus size. As in the case of the with-punc trained learner, once depth 2 is discovered, the system quickly overgeneralizes, then converges in a consistent range (in this case around 250 uses of depth 2).

To evaluate accuracy on the punctuation-free data, we sampled parses from iteration 1500 of our learner. Results are given in Table 1. Binary UHHMM results are on par with UPPARSE, worse than CCL, and considerably worse than BMMM+DMV, while flattened UHHMM results show higher overall F-measures than both CCL and UPPARSE. BMMM+DMV suffers less in the absence of punctuation than the other systems (and therefore generally provides the best induction results on no-punc). The large drop in UHHMM accuracy with the removal of punctuation provides weak evidence for the use of intonational phrasal cues in human syntax acquisition.

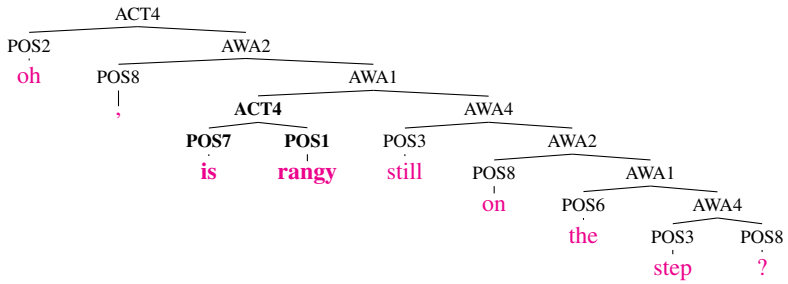
While the BMMM+DMV results are on par with ours, it is important to note that we used a severely restricted number of categories in order to improve computational efficiency. For example, our system was given 8 PoS tags to work with, while BMMM+DMV was given 45. Finer grained state spaces in a more efficient implementation of our learner will hopefully improve upon the results presented here.

Finally, it is interesting to observe that the uses of depth 2 shown in Figures 5 and 8 are in general linguistically well-motivated. They tend to occur in subject-auxiliary inversion, ditransitive, and contraction constructions, in which depth 2 is often necessary in order to bracket auxiliary+subject, verb+object, and verb+contraction together, as illustrated in Figure 9. Unfortunately, due to the flat representation of these constructions in the gold standard trees, this insight on the part of our learner is not reflected in the accuracy measures in Table 1.

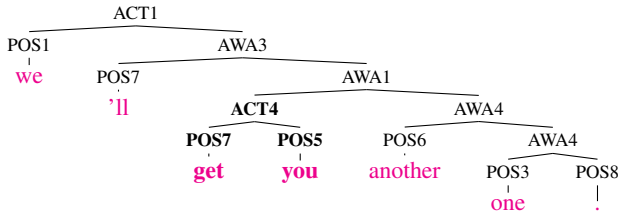
---

short length of child-directed utterances, and therefore the right-branching baseline (RB) outperforms all systems by a wide margin on this corpus. However, we argue that such utterances are a more realistic model of input to human language learners than newswire text, and therefore preferable for evaluation of systems that purport to model human language acquisition. Our system learns this directional bias from data, and does so at least as successfully as its competitors.

1. **Subject-auxiliary inversion:**



2. **Ditransitive:**



3. **Contraction:**

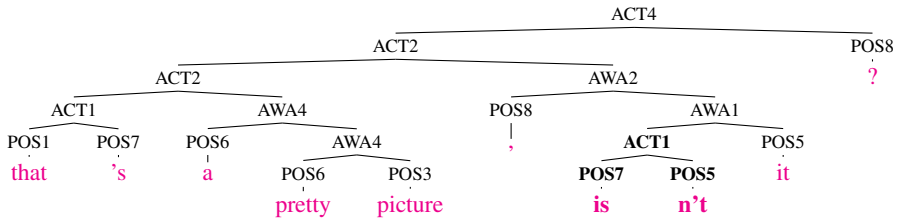


Figure 9: Actual parses from UHHMM-4000 (with punctuation), illustrating the use of depth 2 (bold) for subject-aux inversion, ditransitives, and contractions.

6 Conclusion

This paper presented a grammar induction sytem that models the working memory limitations of young language learners and employs a cognitively plausible left-corner incremental parsing strategy, in contrast to existing raw-text induction systems. The fact that our system can model these aspects of human language acquisition and sentence processing while achieving the competitive results shown here on a corpus of child-directed speech indicates that humans can in principle learn a good deal of natural language syntax from distributional statistics alone. It also shows that modeling cognition more closely can match or improve on existing approaches to the task of raw-text grammar induction.

In future research, we intend to make use of parallel processing techniques to increase the speed of inference and (1) allow the system to infer the optimal number of states in each component of the model, permitting additional granularity that might enable it to discover subtler patterns than is possible with our currently-restricted state inventories, and (2) allow the system to make use of depths 3 and 4, modeling working memory capacities of older learners.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments. This project was sponsored by the Defense Advanced Research Projects Agency award #HR0011-15-2-0022. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

Steven P. Abney and Mark Johnson. 1991. Memory requirements and local ambiguities of parsing strategies. *J. Psycholinguistic Research*, 20(3):233–250.

Alfred V. Aho and Jeffery D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling, Vol. 1: Parsing*. Prentice-Hall, Englewood Cliffs, New Jersey.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48, Montreal.

Noam Chomsky and George A. Miller. 1963. Introduction to the formal analysis of natural languages. In *Handbook of Mathematical Psychology*, pages 269–321. Wiley, New York, NY.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2011. A Bayesian mixture model for part-of-speech induction using multiple features. In *Proceedings of EMNLP*, pages 638–647, Edinburgh, Scotland, 7.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2012. Turning the pipeline into a loop: Iterated unsupervised dependency parsing and PoS induction. In *NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 96–99, Montreal, Canada, 6.

Michael Collins, Jan Hajic, Lance A. Ramshaw, and Christoph Tillman. 1999. A statistical parser for Czech. In *Proceedings of ACL*.

Nelson Cowan. 2001. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24:87–185.

Jeffrey L. Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48:71–99.

Susan E. Gathercole. 1998. The development of memory. *Journal of Child Psychology and Psychiatry*, 39:3–27.

Edward Gibson. 1991. *A computational theory of human linguistic processing: Memory limitations and processing breakdown*. Ph.D. thesis, Carnegie Mellon.

Boris Goldowsky and Elissa Newport. 1993. Modeling the effects of processing limitations on the acquisition of morphology: the less is more hypothesis. In Jonathan Mead, editor, *Proceedings of the 11th West Coast Conference on Formal Linguistics*, pages 234–247.

Philip N. Johnson-Laird. 1983. *Mental models: Towards a cognitive science of language, inference, and consciousness*. Harvard University Press, Cambridge, MA, USA.

Yakoov Kareev, Iris Lieberman, and Miri Lev. 1997. Through a narrow window: Sample size and the perception of correlation. *Journal of Experimental Psychology*, 126:278–287.

Fred Karlsson. 2007. Constraints on multiple center-embedding of clauses. *Journal of Linguistics*, 43:365–392.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*.

Shalom Lappin and Suart M. Shieber. 2007. Machine learning theory and practice as a source of insight into universal grammar. *Journal of Linguistics*, 43:1–34.

Richard L. Lewis and Shravan Vasishth. 2005. An activation-based model of sentence processing as skilled memory retrieval. *Cognitive Science*, 29(3):375–419.

Brian MacWhinney. 2000. *The CHILDES project: Tools for analyzing talk*. Lawrence Elrbaum Associates, Mahwah, NJ, third edition.

Brian McElree. 2001. Working memory and focal attention. *Journal of Experimental Psychology, Learning Memory and Cognition*, 27(3):817–835.

George A. Miller and Stephen Isard. 1964. Free recall of self-embedded english sentences. *Information and Control*, 7:292–303.

George A. Miller. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97.

Elissa Newport, Henry Gleitman, and Lila Gleitman. 1977. Mother, I'd rather do it myself: Some effects and non-effects of maternal speech style. In Catherine F. Snow, editor, *Talking to Children*, pages 109–149. Cambridge University Press, Cambridge.

Elissa Newport. 1990. Maturational constraints on language learning. *Cognitive Science*, 14:11–28.

Lisa Pearl and Jon Sprouse. 2013. Syntactic islands and learning biases: Combining experimental syntax and computational modeling to investigate the language acquisition problem. *Language Acquisition*, 20:23–68.

Elias Ponvert, Jason Baldridge, and Katrin Erik. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086, Portland, Oregon, 6.

Philip Resnik. 1992. Left-corner parsing and psychological plausibility. In *Proceedings of COLING*, pages 191–197, Nantes, France.

Douglas L.T. Rohde and David C. Plaut. 2003. Less is less in language acquisition. In Philip Quinlan, editor, *Connectionist modelling of cognitive development*. Psychology Press, Hove, UK.

Jenny R Saffran, Elizabeth K Johnson, Richard N Aslin, and Elissa L Newport. 1999. Statistical learning of tone sequences by human infants and adults. *Cognition*, 70(1):27–52.

Lynn Santelman and Peter W. Jusczyk. 1998. Sensitivity to discontinuous dependencies in language learners: Evidence for limitations in processing space. *Cognition*, 69:105–34.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2010. Broad-coverage incremental parsing using human-like memory constraints. *Computational Linguistics*, 36(1):1–30.

Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 384–391.

Amanda Seidl, George Hollich, and Peter W. Jusczyk. 2003. Early understanding of subject and object wh-questions. *Infancy*, 4(3):423–436.

Jayaram Sethuraman. 1994. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650.

Edward Stabler. 1994. The finite connectivity of linguistic structure. In *Perspectives on Sentence Processing*, pages 303–336. Lawrence Erlbaum.

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Jurgen Van Gael, Yunus Saatci, Yee Whye Teh, and Zoubin Ghahramani. 2008. Beam sampling for the infinite hidden Markov model. In *Proceedings of the 25th international conference on Machine learning*, pages 1088–1095. ACM.

Jurgen van Gael, Andreas Vlachos, and Zoubin Ghahramani. 2009. The infinite HMM for unsupervised PoS tagging. (August):678–687.

Marten van Schijndel, Andy Exley, and William Schuler. 2013. A model of language processing as hierarchic sequential prediction. *Topics in Cognitive Science*, 5(3):522–540.

‘Calling on the classical phone’: a distributional model of adjective-noun errors in learners’ English

Aurélie Herbelot
Centre for Mind/Brain Sciences
University of Trento
aurelie.herbelot@unitn.it

Ekaterina Kochmar
ALTA Institute
University of Cambridge
ek358@cl.cam.ac.uk

Abstract

In this paper we discuss three key points related to error detection (ED) in learners’ English. We focus on content word ED as one of the most challenging tasks in this area, illustrating our claims on adjective–noun (AN) combinations. In particular, we (1) investigate the role of context in accurately capturing semantic anomalies and implement a system based on distributional topic coherence, which achieves state-of-the-art accuracy on a standard test set; (2) thoroughly investigate our system’s performance across individual adjective classes, concluding that a class-dependent approach is beneficial to the task; (3) discuss the data size bottleneck in this area, and highlight the challenges of automatic error generation for content words.

1 Introduction

Error detection (ED) in the prose of ‘English as a Second Language’ (ESL) learners has recently attracted much attention (Ng et al., 2014; Ng et al., 2013; Dale et al., 2012; Dale and Kilgarriff, 2011). Earlier work on ED in ESL writing mostly focused on grammatical errors and errors in function words (Felice and Pulman, 2008; Gamon et al., 2008; Tetreault et al., 2010; Gamon, 2010; Rozovskaya and Roth, 2010a; Dahlmeier and Ng, 2011b; Ng et al., 2013). Lately, the focus has shifted to other error types, with the recent shared tasks encompassing all errors (Ng et al., 2014; Daudaravicius et al., 2016). In Ng et al. (2014), errors in content words are reported to be the second most frequent error type among 28 categories, accounting for 11.8% of all errors in the training and for about 14% in the test data, yet most teams scored poorly in this category suggesting that this is a challenging and mostly unsolved problem.

Current ED approaches can be broadly described as either *modular*, addressing one error type in particular, or as *comprehensive*, spanning all error types, as in case of the SMT-based techniques (Felice et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2014). The modular approaches rely on the systematic and recurrent nature of the error patterns, and on the availability of closed confusion sets which enable casting the task as a multi-class classification problem. Since content words do not assume a finite set of confusions, it has been shown that ED for these combinations cannot be performed in a similar way (Kochmar and Briscoe, 2014; Rozovskaya et al., 2014). State-of-the-art SMT-based approaches also struggle with content word errors. We argue that ED systems for words which carry lexical meaning should necessarily involve a semantic component, which is typically not needed for other error types.

From a pedagogical point of view, detecting content word errors is an important task. Since content words carry the semantics of a sentence as well as the communicative intent of the writer, incorrect uses may lead to misunderstandings and meaning distortions: for example, *classic* and *classical* are frequently confused by language learners, yet *classic dance* and *classical dance* clearly have different denotations. The importance of content word knowledge in language learning has been demonstrated in the previous ESL studies: James (1998) points out that language learning itself is sometimes equated with mastering vocabulary. Leacock et al. (2014) mention an experiment in which teachers of English were asked to rank errors according to their gravity, and word choice errors were ranked as one of the two most serious

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 976–986, Osaka, Japan, December 11-17 2016.

error categories. The study of Leacock and Chodorow (2003) also demonstrates that errors in content words have a direct impact on overall results in the *Test of English as a Foreign Language* (*TOEFL*).

In this paper, we focus on the underexplored task of content word error *detection*, independently of correction (see §2). We follow the semantically motivated approach outlined by Kochmar and Briscoe (2014) (henceforth K&B) for adjective–noun (AN) combinations,[1] building on their work by integrating context information in the classification. That is, we want to learn that although *classical dance* is more frequent than *classic dance*, the latter is correct in a context such as *They performed a classic Scottish dance*. In §3, we propose that features based on distributional topic coherence (Newman et al., 2010) can catch semantically anomalous ANs by modelling the effects of errors on the coherence of their context. A simple system based on this idea obtains state-of-the-art results. In §4, we show that the combination of the proposed in-context (*IC*) system with the out-of-context (*OOC*) ED system of K&B can further improve results, as long as the *OOC* system's error recall is sufficient. A thorough investigation of our system reveals that its performance is dependent on the adjective classes (see §5). This leads us to the conclusion that content word errors should be treated in a class-dependent way.

Finally, we show that availability of high-quality learner data for training the ED algorithms is of paramount importance. We note that certain error types, having recurrent error patterns, allow for straightforward artificial error data generation. However, we experimentally show that quality artificial data cannot be so easily generated for content words (see §6).

2 Related work

2.1 ED in content words

Previous approaches to error detection and correction of content words fall into two paradigms. One focuses on correction only, assuming that errors are detected by a separate ED algorithm (Liu et al., 2009; Dahlmeier and Ng, 2011a). The second performs error detection and correction through a single algorithm (Chang et al., 2008; Futagi et al., 2008; Park et al., 2008; Yi et al., 2008). The latter type relies on comparison of the learner's choice with possible alternatives: if any alternative scores higher than the original according to the chosen frequency-based metric, the original combination is flagged as an error and the alternative is suggested as a correction (Leacock et al., 2014). Approaches based on this idea have a number of weaknesses. In particular, they rely on the availability of a set of plausible alternatives and are unable to detect errors in the absence of such alternatives, even though a number of studies (Leacock et al., 2009; Chodorow et al., 2010; Andersen et al., 2013) have shown that ED alone (without correction) is useful for language learners. Crucially, K&B have also shown that some original word combinations can be felicitous even when some alternatives score higher, leading to over-corrections which can be hugely detrimental to ESL learners. Such considerations speak for considering ED as a separate task.

Prior proposals have rarely analysed content word errors from a semantic perspective. K&B have focused on ED for AN phrases and shown that approaches aimed at detecting semantic deviance can also identify errors in content words. These authors cast ED as a binary classification task and train a machine learning classifier using features derived from compositional distributional semantics. They obtain 81% accuracy, and show that their semantically motivated approach outperforms the state-of-the-art in ED so far. One drawback of their method is that they do not take context into account: features are based on the distributions of an AN's components and their composition, but not on the particular context where it is used. When evaluating their system in context by comparing the system's predictions with human, context-sensitive annotation, the authors note that accuracy drops to 65%. Our approach takes both semantic aspect *and* surrounding context into account.

2.2 Learner data

Since the standard approaches to ED rely on machine learning, availability of learner data is of paramount importance. Some error types allow for straightforward generalisation from seen examples (e.g., errors in function words or mechanical errors), but errors in content words appear to be less systematic. Therefore, it is crucial to have sufficient, thoroughly annotated learner data. To the best of our knowledge,

[1] We also believe that our approach can ultimately be applied to other types of content word combinations.

the AN dataset released by K&B[2] is the only publicly available dataset of learners' errors in content words that satisfies quality requirements. ANs are labelled with error type (semantically-related or form-related confusion, or no relation) and possible corrections are suggested. The data contains a two-level annotation, with the ANs being labelled as correct or incorrect out of their context (*OOC*), as well as in the original context of use (*IC*). For example, *classic dance* is annotated as correct *OOC*, but incorrect *IC* whenever it is used erroneously in place of *classical dance*. The dataset contains 798 ANs that are extracted from the *Cambridge Learner Corpus* (*CLC*)[3] and are unattested in the *British National Corpus* (*BNC*). This data is interesting for a linguistically-motivated investigation of learners' errors: K&B demonstrated that approaches that simply rely on frequency and collocational strength do not perform well on it.

The dataset contains 892 unique contexts, and in our experiments, we use a subset of 824 contexts (see §3). The lower bound for *IC*-annotated ANs is estimated as the majority class baseline and equals 0.55. The upper bound estimated as the average inter-annotator agreement is 0.74.

3 Topic coherence for error detection

3.1 Topic coherence

Topic coherence is a measure of the semantic relatedness of the items in a given set of words, which has mostly been studied from the perspective of 'topic modelling' techniques (Steyvers and Griffiths, 2007). Topic modelling is a text classification method which generates so-called topics from a corpus by analysing word co-occurrences, and subsequently models any new text in terms of those topics. A topic is expressed as a list of keywords supposed to be highly characteristic for a subject: for instance, {*film, actor, cinema, Hollywood*} might be the main keywords for a *film* topic. In order to obtain an intrinsic evaluation of such models, recent work has started investigating whether topics produced by standard techniques can be said to be 'coherent', i.e. whether topic keywords belong together, from a human point of view (Chang et al., 2009; AlSumait et al., 2009; Newman et al., 2010; Mimno et al., 2011).

Even though they stem from research on topic modelling, topic coherence measures can be applied to any set of words, and might for instance tell that the set {*chair, table, office, team*} is more coherent than {*chair, cold, elephant, crime*}. They are well suited to model semantic association and we hypothesise that they can tell us something about the semantic validity of a sentence.

3.2 Experimental setting

Following Newman et al. (2010), we define the coherence $\mathcal{COH}$ of a set of words $w_1...w_n$ as the mean of their pairwise similarities:

$$\mathcal{COH}(w_{1...n}) = mean\{Sim(w_i, w_j), ij \in 1...n, i < j\}$$

We estimate similarity as the cosine distance between two words in a distributional space (Turney and Pantel, 2010). In that setup, the meaning of a word is a vector that lives in a space where dimensions correspond to linguistic contexts. The vector's components reflect how characteristic a context is for the word under consideration.

Our hypothesis is that some lexical errors might result in a sharp variation of semantic coherence. Consider an example from learners' data:

(1) ... it was very difficult for my friends to call me with the *classical* *phone*...

The adjective *classical* is distributionally associated with the arts, collocating with nouns like *dance, music, style* or *literature*. Its similarity to *friend, call* or *phone* is much lower than the pairwise similarities of those words alone. We hypothesise that the inclusion of the unrelated *classical* in the sentence would thus have an adverse effect on its overall coherence.

[2] http://ilexir.co.uk/applications/adjective-noun-dataset/

[3] http://www.cup.cam.ac.uk/gb/elt/catalogue/subject/custom/item3646603/Cambridge-International-Corpus-Cambridge-Learner-Corpus/

Context size	1	2	3	4	5
SVM (low)	0.59 (± 0.02)	0.58 (± 0.02)	0.58 (± 0.01)	0.58 (± 0.02)	0.58 (± 0.02)
SVM (high)	0.59 (± 0.02)	0.59 (± 0.03)	0.58 (± 0.02)	0.59 (± 0.02)	0.59 (± 0.02)
+ adj. (low)	0.62 (± 0.04)	0.62 (± 0.04)	0.62 (± 0.03)	0.62 (± 0.02)	0.62 (± 0.04)
+ adj. (high)	0.64 (± 0.04)	**0.66** (± 0.06)	0.64 (± 0.04)	0.63 (± 0.02)	0.64 (± 0.02)

Table 1: SVM classification accuracy over different context sizes with three $\mathcal{COH}$ features, and with added *adjective* feature. The 'low/high' scores are the lowest/highest across all values of the C parameter.

Our learners' data consists of the AN dataset (see §2), spell-checked to correct orthographic errors. We build a distributional semantics space from the BNC,[4] using lemmatised word windows as context (size=10), the top 2000 most frequent content words as dimensions, and Positive Pointwise Mutual Information (PPMI) as weighting measure.[5] For each instance in the learners' data, we define a context window W as the AN under consideration and n words on each side of that AN. Allowable context words are nouns, verbs, adjectives and adverbs for which a BNC distribution is available. When the AN contains a word not found in the BNC, we discard the corresponding instances, ending up with 824 items out of the original 892 in the AN dataset. We are interested in three measures:

- the topic coherence $\mathcal{COH}$ of context W;
- the topic coherence $\mathcal{COH}_{\text{-}adj}$ of the context without the *adjective*;
- the topic coherence $\mathcal{COH}_{\text{-}noun}$ of the context without the *noun*.

Our starting hypothesis is that when the AN is erroneous, omitting either the adjective or the noun in the calculation results in a significant variation of the original coherence score.

3.3 Topic coherence results

We perform SVM classification with 5-fold cross-validation, using the coherence figures $\mathcal{COH}$, $\mathcal{COH}_{\text{-}adj}$ and $\mathcal{COH}_{\text{-}noun}$ as features. The order of the data is randomised before creating the folds, and the ratio of correct/incorrect instances kept equal between folds (55% correct to 45% incorrect). We use *SVM^{light}* (Joachims, 1999) with an RBF kernel to classify the data. We tune penalty parameter C, experimenting with the values of C in the range 10-200. Since the size of the AN dataset does not allow for the use of a development set, we report the lowest and highest system performance across all folds.

Table 1 shows that our simple 3-feature system reaches 59% accuracy. We attempt to improve on this by specifying which adjective occurs in the AN: we add 61 binary features to the SVM, corresponding to the 61 different adjectives in the data, and 'turn on' the feature matching the adjective under consideration for each data point. This step results in a further increase in accuracy, reaching 66%, which is on a par with the result reported by K&B when taking the *OOC* annotation to an in-context setting. This result is highly encouraging since our system is overall much simpler: given an available distributional semantic space, coherence values can be computed very straightforwardly and the SVM classifier relies on few features. We also note that the system is stable across various values of C: the differences between lowest and highest scores are not significant given the variability observed across all 5 folds. In the rest of this paper, we only report our highest scores under the understanding that varying C does not significantly affect results.

The best accuracy is obtained for a context size of 2, but the differences in performance between various context sizes are not statistically significant either. Most likely, the ideal context window for the task depends on the sentence. In some cases, larger context is actually harmful to ED, as in Ex. 2 below, where the context words are mostly about shopping/buying and do not have a straightforward association with either *cat* or *funny*. In contrast, Ex. 3 needs a larger window to catch that the sentence is not about different *bears* drinking but rather about a restaurant with *beers*.

[4] http://www.natcorp.ox.ac.uk/

[5] The space was built using the DISSECT toolkit (Dinu et al., 2013), available at http://clic.cimec.unitn.it/composes/toolkit/index.html.

	System description	Classifier	Accuracy
COH	§3.3	SVM	0.66
+ COMPDIST	feat. combination (§4.1)	SVM	0.68
+ COMPDIST	pipeline (§4.2)	SVM	0.67
+ *OOC* gold	§4.2	SVM	0.76
COMPDIST	K&B	DT	0.64
+ COH	feat. combination (§4.1)	DT	0.66

Table 2: Classifier combination accuracies

(2) I went shopping yesterday, and I've bought a new shirt. I had to buy it because it had a *funny cat* on it. It was quite cheap, it costs just £4.

(3) In the second one you can eat some easy food as salads, but you also can drink a great number of *different bears*.

An analysis of our results shows that the classifier is well-balanced, achieving 0.66 and 0.65 precision for correct and incorrect instances, as well as 0.65 and 0.66 recall. This is an improvement over the context-insensitive system from K&B, which scored much better recall on correct than incorrect instances (72% vs 58%).

4 Combining classifiers

K&B showed that it is possible to classify *OOC*-annotated content word errors with high accuracy: the authors reported an accuracy of 81% on this task. This means that regardless of context, we can learn that e.g. **big quantity* is incorrect. In the next set of experiments, we investigate the benefits of including context-insensitive classifier in our system: since the ANs that are annotated as errors *OOC* are also errors *IC*, we would expect the system to benefit overall from context-insensitive annotation.

We consider combinations of the following two systems and their respective semantic information:

- COMPDIST is the *context-insensitive* system of K&B, which we ran on our subset of 824 contexts. The AN vectors are built using the *multiplicative* model of semantic composition (Mitchell and Lapata, 2008). A set of measures that can distinguish between the representations of the correct and incorrect ANs is applied, including, for example, *vector length*, *cosine similarity* between the AN vector and the input noun, and so forth.[6] The values of these measures are then used by an ED algorithm running over a *Decision Tree* (DT) classifier.[7]

- COH is the *context-sensitive* system presented in this paper, with three coherence-based features and a feature representing the adjective in the AN (see §3.3).

We examine two architectures: in a first experiment, we simply concatenate the features from COMPDIST and COH and input the resulting vector into (a) an SVM classifier, as used in this paper; (b) a DT classifier, as used in K&B. In a second experiment, we design a pipeline system, where the classification of COMPDIST is fed into the topic coherence model. All results reported in this section are for a context size of 2. A summary can be found in Table 2.

4.1 Direct feature combination

We first run COMPDIST in isolation over the *IC* annotation, to get a baseline accuracy. This results in a performance of 64%, just below our COH system accuracy of 66% (see Table 2).

We then proceed with feature concatenation, starting with the full feature set and then applying ablation tests to identify the best-performing features. The features are fed into the DT classifier of K&B on one hand (line COMPDIST+COH in Table 2) and our SVM classifier on the other hand (line COH+COMPDIST).

[6] For the full feature set, please consult K&B.

[7] We use the *NLTK* implementation (Bird et al., 2009).

1 usual (4/4)	.83 strong (15/18)	.63 fast (7/11)	.41 historical (5/12)
1 rapid (1/1)	.83 clear (5/6)	.62 small (15/24)	.40 economic (2/5)
1 magic (1/1)	.80 actual (4/5)	.62 nice (39/62)	.33 deep (1/3)
1 incorrect (3/3)	.75 bad (24/32)	.62 important (18/29)	.30 whole (3/10)
1 elder (16/16)	.72 good (39/54)	.60 unique (6/10)	.25 heavy (2/8)
1 economical (33/33)	.71 hard (5/7)	.60 high (3/5)	.20 true (1/5)
1 classical (5/5)	.70 main (7/10)	.60 electric (3/5)	.18 certain (2/11)
1 classic (3/3)	.70 different (29/41)	.60 correct (3/5)	.14 precious (1/7)
.90 funny (10/11)	.69 best (36/52)	.57 near (4/7)	.14 particular (1/7)
.89 suitable (8/9)	.68 typical (11/16)	.53 wrong (7/13)	.14 ancient (1/7)
.89 soft (8/9)	.67 big (63/94)	.50 short (6/12)	0 far (0/2)
.89 full (8/9)	.66 various (6/9)	.50 present (2/4)	0 false (0/2)
.89 convenient (8/9)	.64 proper (9/14)	.47 common (8/17)	0 electrical (0/3)
.87 large (7/8)	.63 great (26/41)	.42 appropriate (3/7)	

Table 3: Per-adjective precision values for SVM classification, sorted from highest to lowest

Using the DT classifier, the best accuracy of the direct combination of features is 66% with the feature set including *cosine similarity* to the input noun, *ranked density*, *adjective* and a *coherence-based* feature based on $(\mathcal{COH} - \mathcal{COH}_{\text{-}adj})$. The improvement over baseline is however not statistically significant.

Adding the COMPDIST features to the SVM COH system results in a similar improvement, reaching 68% from a 66% baseline, using the *cosine similarity* to the noun, and semantic neighbourhood features.

4.2 Pipeline system

To test the actual effect of the topic coherence features at in-context classification stage, we first attempt to add the *OOC* gold annotation to our system, in the form of a new feature. The baseline created by the *OOC* gold annotation is very high: running the classifier over that one feature results in 73% accuracy. Nevertheless, performance increases when the gold annotation is combined with COH. The best result is 76% – a 3% improvement (statistically significant at $p = 0.03$). This is over the human upper bound of 74%, and it shows that the topic coherence features perform well in contextualising the *OOC* annotation.

However, since a 'real-world' pipeline system does not have access to the gold annotations, we replace the gold annotations with the output of COMPDIST. In this setting, the combination only produces minimal improvement, reaching 67% accuracy with the SVM and 66% with the DT classifier. The reason for this result is low error recall of the *OOC* system which is tuned towards high precision because of the strong negative impact on the learners when wrongly reporting an error. While this is sensible from an educational point of view, it means that we are only recalling 17% of erroneous ANs at the *OOC* stage. We conclude that improving *OOC* detection can hugely benefit the overall system.

5 Adjective-dependent classification analysis

5.1 COH analysis

Our experiments with the coherence-based system showed that it is particularly accurate in classifying form-related errors: the accuracy on *classic*, *classical*, *economical*, *elder*, *electric*, *electrical* and *historical* – which are responsible for 80% of the form-related errors in our data – is 77%. Otherwise, the accuracy of the system is generally dependent on the adjective being classified. Table 3 shows precision values for each adjective in our data.[8] While *economical* (33 instances) and *funny* (11 instances) achieve 100% and 90% precision respectively, *certain* (11 instances) and *ancient* (7 instances) only reach 18% and 14%. Roughly speaking, adjectives expressing a sentiment (*funny*, *suitable*, *convenient*, *bad*, *good*, *best*, *great*, *nice*) are to be found at the top of the table, while no such consistency is to be found for the quantity adjectives: *large*, *big*, *small*, *high*, *deep*, *short* and *heavy* span a whole range of precision values, from 87% down to 25%. We conclude that the adjectives in our dataset can behave very differently with respect to the types of errors they attract, and a single classifier may not be able to model all cases equally well. In the next set of experiments, we thoroughly investigate our system's performance

[8] Morphologically related forms, for example *big* and *biggest*, are collapsed together.

Adjective	Best training elements	Accuracy
appropriate	{nice, good, best, different, bad, short, fast}	71.43%
bad	{unique}	78.12%
best	{nice, good, different, fast, funny, unique}	71.70%
big	{proper}	68.09%
correct	{nice, good, best, different, bad, short, fast, unique}	80.00%
economic	{strong, typical, elder, certain}	80.00%
economical	{small, strong, typical, elder, proper, certain}	100.00%
elder	{economical, small, strong, typical, proper, certain}	100.00%
funny	{big}	90.91%
good	{nice, best, different, fast}	70.91%
great	{wrong, main}	69.05%
nice	{good, best, different, fast}	67.74%
precious	{funny}	71.43%
small	{big, proper, funny}	68.00%

Table 4: Best training elements for a subset of adjectives, together with accuracy

across individual adjective classes. Since we lack data for a separate development set, these experiments present the analysis of the data rather than actual classification results but we believe that these results can inform future studies.

5.2 Modelling the AN data

We hypothesise that some adjectives behave in a similar way with respect to their interaction with topic coherence, and may be classifiable under a joint category. Since there are no obvious confusion sets for content words to guide category formation (see §1), we attempt to model the AN set in a purely data-driven way. We first train a classifier over each adjective with frequency ≥ 10, thus obtaining 27 individual classifiers. We then apply each classifier to every single other adjective and record which one(s) perform(s) best for that item. For example, we verify how well *ancient* is classified by each of the 26 models and record the classifier trained on *unique* as the one performing best. We take this as evidence that *ancient* and *unique* share some properties with respect to the task.

Table 4 shows accuracy for some adjectives, with the best recorded training set(s). The overall accuracy, averaged over all adjectives, is 75%. This result is on a par with human performance estimated at 74% (see §2). Per-class precision is 80% on errors and 73% on correct instances, while recall is 59% for errors and 88% for correct ANs.

We note two trends: (a) adjectives of judgement (*appropriate*, *bad*, *correct*, *nice*, *precious*) tend to be best trained by other judgement adjectives (*best*, *good*, *nice*); (b) adjectives for which form-related errors are frequent (*classic*/*classical*, *economic*/*economical*, *elder*) tend to get their best accuracy when trained on the same set (*strong*, *typical*, *elder*, *certain*).

These results suggest that training adjective classes separately could have a very positive impact. For instance, let's consider the set {*nice*, *good*, *best*, *bad*, *convenient*, *suitable*, *appropriate*}. Training each adjective over the other members of the set results in an increase in performance for those adjectives: e.g., accuracy for *nice* increases by 5 points, for *appropriate* by 14 points. Similarly, training {*small*, *big*, *large*} as a set gives a 5-point improvement on our best results.

However, due to the relatively small size of the dataset, it is impossible to have a development phase to choose the best training sets, and a separate test phase to verify robustness: confirming that *bad* is indeed best trained on *unique* would overall require more data than the 32 and 10 instances currently available.

5.3 Adjective-specific system combination

We have shown that COH performs differently on adjective-specific subsets and we have assumed a complex interaction between topic coherence and error patterns specific for particular adjectives. Our tests using COMPDIST and COH with the DT classifier confirm that the performance of the two systems on the adjective-specific subsets in the data is also different: for example, COH performs well on the adjectives *large*, *bad* and *good* (see Table 3), while COMPDIST achieves better results on *short* and *heavy*. In the next experiment, we combine the two systems in an integrated adjective-specific ED algorithm.

We implement an oracle system via a voting step based on the COMPDIST and COH adjective-specific performance. That is, for each test item, we use the prediction of the system which has best accuracy for the particular adjective under consideration. This oracle system achieves an accuracy of 71% which compares favourably to the results of COMPDIST (64%), COH (66%) alone, as well as the direct combination of the features used by the two systems (68%). Although this result is an upper bound since we lack data to set up a separate development set, we can reliably make two observations. First, we confirm that using adjective-specific information in ED improves the algorithm performance. Second, we note that the voting system's upper bound is comparable to human performance, indicating that the combination of a strong *OOC* baseline and a relatively simple semantic model of context provides the necessary conditions for ideal results: the combined system covers all relevant information for the task, and training on an expanded dataset can be expected to drastically improve performance.

6 Generating errors

Earlier we noted that high-quality learner data is crucial for ED and that, due to the size of the K&B dataset, we could not verify the results of our experiments on a separate development set (§3 and §5). Since annotated content word error data is expensive and time-consuming to produce, in this final set of experiments we attempt to generate more data in an automated way. Artificial error generation has previously been demonstrated to be useful for function word ED (Foster and Andersen, 2009; Rozovskaya and Roth, 2010b; Felice and Yuan, 2014). Following this line of work, we attempt to produce data by random substitution of adjectives.

For each adjective, we extract new data from a section of *ukWaC* (Baroni et al., 2009) totalling 1M tokens, POS-tagged and parsed with the RASP parser (Briscoe et al., 2006). We collect word windows containing the adjective under consideration, using the pattern [word$_{-2}$] [word$_{-1}$] [ADJ] [noun] [word$_{+1}$] [word$_{+2}$], thus using a 2-word context around the AN. Next, we randomly shuffle the adjectives and their contexts so that for a particular context window W_k associated with an adjective a_k, we replace a_k with a_m – an adjective linked to another context window – assuming that in most cases, such substitutions will produce incorrect instances. Then, we collect all incorrect instances for a given adjective and concatenate them with an equal number of correct uses, giving us a balanced training set for that adjective. The size of each training set is dependent on the overall frequency of the adjective, ranging from 20 to 2600 instances. Around half of the adjectives have a training set with over 1000 instances, the vast majority (93%) have at least 100 training examples.

Training and testing on this data unfortunately does not produce the expected improvements, with the accuracy falling to 56%. We conclude that the nature of the training data is vital to the performance of the system: in complex tasks like content word ED, automatically generated examples are no substitute for real human errors, and the subtle semantic phenomena occurring in learners' writing cannot be easily reproduced. The absence of clear confusion sets for content word errors makes the task of error generation particularly arduous. The erroneous *calling on the classical phone* is a case in point, showing that a wrong use of *classical* does not necessarily derive from a confusion with *classic*: the speaker probably meant *landline*. Such cases show that the diversity of content words errors makes artificial error generation less viable than for function words, and illustrates the value of 'real', annotated learner data.

7 Conclusion

We have investigated the linguistic felicity of AN phrases through the lens of distributional topic coherence and conclude by showing how this work can inform future research on content word ED.

First, we showed that using topic coherence features to model context leads to accuracy figures that are competitive with previously reported results (K&B). Framing ED in terms of coherence is linguistically sensible and computationally efficient. There are benefits in using *OOC* and *IC* systems in a pipeline architecture, but this relies on the *OOC* system having good enough recall.

Second, we found that per-adjective classification could in principle approach human-like performance. However, proper training and evaluation requires a larger dataset than is currently available.

Thirdly, we investigated the automatic creation of training data. Our experiments demonstrated that real learners' data cannot be easily substituted: in contrast with function words, content words are not naturally associated with clear confusion sets which might guide data generation.

In further work, we would like to pursue our investigation of the linguistic factors that govern certain types of errors. One interesting avenue would be to research the influence of the learner's L1 language on the observed semantic mistakes: we could imagine, for instance, that some systematicity could be captured in the effects of polysemy across languages (e.g. *heavy* might be more—or at least differently—polysemous in English compared to the learner's L1).

We will also concentrate on the expansion of the available dataset in a controlled fashion, ensuring that enough data is supplied for individual adjective training and testing. Whilst our results on error generation indicate that automatic methods may not be suited to the task, more sophisticated procedures could be tried out. For instance, it is conceivable that a distributional analysis of a (non-annotated) learners' corpus would highlight certain systematic errors which would be replicable on a larger scale. With more training data available, another interesting avenue would be to further explore adjective-dependent classification approaches and adjective category formation.

Acknowledgments

We are grateful to the COLING reviewers for their helpful feedback, and for their interesting suggestions for further work. Ekaterina Kochmar's research is supported by Cambridge English Language Assessment via the ALTA Institute. Aurélie Herbelot's contribution to this paper was similarly supported by ALTA.

References

Loulwah AlSumait, Daniel Barbará, James Gentle, and Carlotta Domeniconi. 2009. Topic significance ranking of LDA generative models. In *Machine Learning and Knowledge Discovery in Databases*, pages 67–82. Springer.

Øistein E. Andersen, Helen Yannakoudakis, Fiona Barker, and Tim Parish. 2013. Developing and testing a self-assessment and tutoring system. In *Proceedings of the BEA-2013*, pages 32–41.

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

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

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 77–80.

Yu-Chia Chang, Jason S. Chang, Hao-Jan Chen, and Hsien-Chin Liou. 2008. An automatic collocation writing assistant for Taiwanese EFL learners: A case of corpus-based NLP technology. *Computer Assisted Language Learning*, 21(3):283–299.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-graber, and David M Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems*, pages 288–296.

Martin Chodorow, Michael Gamon, and Joel Tetreault. 2010. The utility of grammatical error detection systems for English language learners: Feedback and Assessment. *Language Testing*, 27(3):335–353.

Daniel Dahlmeier and Hwee Tou Ng. 2011a. Correcting Semantic Collocation Errors with L1-induced Paraphrases. In *Proceedings of the EMNLP-2011*, pages 107–117.

Daniel Dahlmeier and Hwee Tou Ng. 2011b. Grammatical error correction with alternating structure optimization. In *Proceedings of the ACL-HLT 2011*, pages 915–923.

Robert Dale and Adam Kilgarriff. 2011. Helping Our Own: The HOO 2011 Pilot Shared Task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 242–249.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A Report on the Preposition and Determiner Error Correction Shared Task. In *Proceedings of the BEA-2012*, pages 54–62.

Vidas Daudaravicius, Rafael E. Banchs, Elena Volodina, and Courtney Napoles. 2016. A Report on the Automatic Evaluation of Scientific Writing Shared Task. In *Proceedings of the BEA-2016*.

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013. General estimation and evaluation of compositional distributional semantic models. In *Workshop on Continuous Vector Space Models and their Compositionality*, pages 50–58, Sofia, Bulgaria.

Rachele De Felice and Stephen G. Pulman. 2008. A classifier-based approach to preposition and determiner error correction in L2 English. In *Proceedings of the COLING-2008*, pages 169–176.

Mariano Felice and Zheng Yuan. 2014. Generating artificial errors for grammatical error correction. In *Proceedings of the Student Research Workshop at the ACL-2014*, pages 116–126.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the CoNLL 2014: Shared Task*.

Jennifer Foster and Øistein E Andersen. 2009. Generrate: generating errors for use in grammatical error detection. In *Proceedings of the BEA-2009*, pages 82–90.

Yoko Futagi, Paul Deane, Martin Chodorow, and Joel Tetreault. 2008. A computational approach to detecting collocation errors in the writing of non-native speakers of English. *Computer Assisted Language Learning*, 21(4):353–367.

Michael Gamon, Jianfeng Gao, Chris Brockett, Alexander Klementiev, William Dolan, Dmitriy Belenko, and Lucy Vanderwende. 2008. Using contextual speller techniques and language modeling for ESL error correction. In *Proceedings of IJCNLP-2008*, pages 491–511.

Michael Gamon. 2010. Using mostly native data to correct errors in learners' writing. In *Proceedings of the NAACL-2010*, pages 163–171.

Carl James. 1998. *Errors in Language Learning and Use: Exploring Error Analysis.* London: Longman.

Thorsten Joachims, 1999. *Making Large-Scale SVM Learning Practical.* Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The AMU system in the CoNLL-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation. In *Proceedings of the CoNLL-2014: Shared Task*.

Ekaterina Kochmar and Ted Briscoe. 2014. Detecting learner errors in the choice of content words using compositional distributional semantics. In *Proceedings of the COLING-2014*, pages 1740–1751.

Claudia Leacock and Martin Chodorow, 2003. *Automated Grammatical Error Detection. In M. D. Shermis and J. C. Burstein (eds.), Automated Essay Scoring: A Cross-Disciplinary Perspective*, pages 195–207. Mahwah, NJ: Lawrence Erlbaum Associates.

Claudia Leacock, Michael Gamon, and Chris Brockett. 2009. User Input and Interactions on Microsoft Research ESL Assistant. In *Proceedings of the BEA-2009*, pages 73–81.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. 2014. *Automated Grammatical Error Detection for Language Learners, Second Edition. Synthesis Lectures on Human Language Technologies*. Morgan and Claypool Publishers.

Anne Li-E Liu, David Wible, and Nai-Lung Tsao. 2009. Automated suggestions for miscollocations. In *Proceedings of the BEA-2009*, pages 47–50.

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the EMNLP-2011*, pages 262–272.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-HLT 2008*, pages 236–244.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Proceedings of the NAACL-HLT 2010*, pages 100–108.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 Shared Task on Grammatical Error Correction. In *Proceedings of the CoNLL-2013: Shared Task*, pages 1–12.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the CoNLL-2014: Shared Task*, pages 1–14.

Taehyun Park, Edward Lank, Pascal Poupart, and Michael Terry. 2008. Is the sky pure today? AwkChecker: an assistive tool for detecting and correcting collocation errors. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 121–130.

Alla Rozovskaya and Dan Roth. 2010a. Annotating ESL errors: Challenges and rewards. In *Proceedings of the BEA-2010*, pages 28–36.

Alla Rozovskaya and Dan Roth. 2010b. Training paradigms for correcting errors in grammar and usage. In *Proceedings of the NAACL-HLT 2010*.

Alla Rozovskaya, Dan Roth, and Vivek Srikumar. 2014. Correcting Grammatical Verb Errors. In *Proceedings of EACL-2014*, pages 358–367.

Mark Steyvers and Tom Griffiths. 2007. Probabilistic topic models. In T. Landauer, D. McNamara, S. Dennis, and W. Kintsch, editors, *Latent Semantic Analysis: A Road to Meaning*.

Joel Tetreault, Jennifer Foster, and Martin Chodorow. 2010. Using parse features for preposition selection and error detection. In *Proceedings of the ACL-2010: Short Papers*, pages 353–358.

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

Xing Yi, Jianfeng Gao, and William B. Dolan. 2008. A Web-based English Proofing System for English as a Second Language Users. In *Proceedings of the IJCNLP-2008*, pages 619–624.

Are Cohesive Features Relevant for Text Readability Evaluation?

Amalia Todirascu
FDT, LiLPa
Université de Strasbourg
todiras@unistra.fr

Thomas François
Post-doc FNRS
IL&C, CENTAL
UCLouvain
thomas.francois@uclouvain.be

Delphine Bernhard
FDT, LiLPa
Université de Strasbourg
dbernhard@unistra.fr

Núria Gala
LIF-CNRS, Aix-Marseille Université
nuria.gala@univ-amu.fr

Anne-Laure Ligozat
LIMSI-CNRS, Orsay
annlor@limsi.fr

Abstract

This paper investigates the effectiveness of 65 cohesion-based variables that are commonly used in the literature as predictive features to assess text readability. We evaluate the efficiency of these variables across narrative and informative texts intended for an audience of L2 French learners. In our experiments, we use a French corpus that has been both manually and automatically annotated as regards to co-reference and anaphoric chains. The efficiency of the 65 variables for readability is analyzed through a correlational analysis and some modelling experiments.

1 Introduction

Since the 1920's, various readability formulae have been designed to match texts with the reading skills of specific readers. The most famous of these formulas, such as Flesch's (1948) or Dale and Chall's (1948) are typical of what are called "classic" formulas. They rely on a few lexico-syntactic characteristics (e.g., the average number of words per sentence or the average number of syllables per word) to estimate the reading difficulty of a text. This strategy worked to some extent, but, from the late 70's onward, classic formulae have been seriously criticised. Zakaluk and Samuels (1988, 124) thus said: "A basic limitation of readability formulas is that they ignore such critical text factors as cohesiveness and macrolevel organization".

Studies in readability from this period stressed the importance of higher textual dimensions, focusing on inference load (Kemper, 1983), conceptual density (Kintsch and Vipond, 1979), or organisational aspects (Meyer, 1982). As a result, the classic lexico-syntactic features were disregarded for years. However, Miller and Kintsch (1980) soon noticed that including lexico-syntactic features in their cognitive readability formulas improved performance. Chall and Dale (1995, 111) had a more mixed opinion, arguing that variables based on higher textual dimensions "discriminate better among materials requiring greater maturity in reading ability", while classic lexico-syntactic variables work better to discriminate at lower levels of difficulty.

Recently, taking advantage of the opportunities offered by Natural Language Processing (NLP) techniques, readability studies have tried to leverage the semantic and discursive properties of texts to better model text difficulty (Pitler and Nenkova, 2008; Feng et al., 2009). Among those high-level dimensions that have attracted substantial attention are the level of cohesion and coherence of texts. Although psycholinguistic experiments have shown that a higher level of cohesion and coherence between a pair of related sentences decreases their reading time (Kintsch et al., 1975; Mason and Just, 2004), the added value of these textual dimensions for readability models (compared to traditional features) remains unclear, as it will be covered in more details in Section 2.

This is why this paper aims at further investigating the importance of cohesion aspects for the assessment of text readability, as the cohesive dimension is the one that have been investigated the most (see Section 2.2). Based on a corpus of texts for learners of French as a foreign language (L2), which has

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 987–997, Osaka, Japan, December 11-17 2016.

been manually annotated for co-reference chains, the three following research questions will be investigated: (1) are cohesive features relevant for text readability assessment? (2) what is the impact of NLP routines, which are error-prone, on the efficiency of cohesiveness features? and (3) does the genre of the texts (here narrative and informative) influence the discriminating power of cohesiveness features? The methodology applied to investigate these three questions is described in Section 3, while the results are presented in Section 4. The paper concludes with a discussion and some perspectives in Section 5.

2 Cohesion Features to Assess Text Readability

2.1 Coherence and Cohesion

Coherence is defined as a "semantic property of discourse, based on the interpretation of each individual sentence relative to the interpretation of other sentences" (Van Dijk, 1977, 93). The order of the ideas, a logical structuring of the text and coherent relations (consequence, cause-effect) between sentences facilitate the reader's understanding of a specific topic. In addition, readers might use external knowledge as regards the specific situation described in the text.

Cohesion is a property of text represented by explicit formal grammatical ties (discourse connectives) and lexical ties that signal how utterances or larger text parts are related to each other. Halliday and Hasan (1976) identify specific cohesive devices aiming to reinforce lexical ties, such as anaphoric chains or co-reference chains (Schnedecker, 1997), as well as lexical chains (sets of expressions related by hypernymy or hyponymy relations or expressions from the same domain, e.g. *patient–disease-treatment*).

Anaphoric chains are composed of two expressions, one antecedent and one anaphora. In Figure 1, the interpretation of the definite noun phrase *the ship* (the anaphora) is dependent on its antecedent (*the RMS Titanic*). Co-reference chains are composed of at least three referring expressions corresponding to the same discourse entity (Schnedecker, 1997). In Figure 1, the expressions *Edward Smith, an English naval reserve officer, He, He* refer to the same entity, the Titanic's commander. Lexical chains are composed of associated words or expressions related by ontological relations (synonymy, hypernymy, hyponymy) or relative to the same domain (Hirst and St-Onge, 1998), such as *naval reserve officer, vessels, ship sank, voyage* (Figure 1).

Edward John Smith was *an English naval reserve officer*. *He* served as commanding officer of numerous White Star Line vessels. *He* is best known as *the captain of* ***the RMS Titanic***, perishing when **the ship** sank on the 15th April 1912. (Wikipedia)

Figure 1: Example of anaphoric and of co-reference chain.

These three devices strengthen the links between several utterances and contribute to the overall understanding of the text (Charolles, 1995). Lexical chains are effective mechanisms to find the main domain or theme of the document. Cohesive devices such as anaphora or co-reference chains correspond to one entity expressed by various linguistic expressions (so called mentions). These expressions are related by complex morpho-syntactic, syntactic or semantic constraints (Grosz et al., 1995). Mentioning the same entity several times reinforces text cohesion (Poesio et al., 2004), (Hobbs, 1979). Cohesive devices reinforce local coherence relations in some specific genres (persuasive genres) (Berzlnovich and Redeker, 2012).

An interesting characteristic of cohesive devices is that their use is dependent on the type or genre of texts (Carter-Thomas, 1994). For instance, informative texts use specific referential expressions such as definite or demonstrative noun phrases as mentions, while narrative texts contain more chains composed of proper nouns or personal pronouns (Schnedecker, 2005). The composition, the length or the choice of the first mention of the co-reference chain is also dependent on the genre. For instance, in newspapers portraits (Schnedecker, 2005), co-reference chains start with a proper noun and contain mainly definite noun phrases and personal pronouns. For example, in law and administrative texts, reference chains start with indefinite noun phrases and the mentions are mainly definite or demonstrative noun phrases (Longo and Todirascu, 2014).

In this article, we consider explicit lexical ties such as anaphoric, co-reference and lexical chains as cohesive features. We study the correlation between these cohesive features and text complexity.

2.2 Coherence and Cohesion in Readability

As both coherence and cohesion are important text properties that are known to influence the readability of texts, readability studies have attempted to exploited both dimensions. However, most studies focused on phenomena that falls inside the category of cohesion as defined in Section 2.1 which is why we decided to focus on cohesive features in this paper.

The first to investigate the issue of text cohesion in readability analysis is probably Bormuth (1969). He considered that the correct identification of anaphoric relations was a prerequisite to the correct understanding of a text and thus computed 12 variables based on various characteristics of anaphora, showing that the density of anaphora to be the best predictor with a $r = 0.532$.

More recently, text cohesion were investigated in readability with another approach that relies on latent semantic analysis (LSA) (Landauer et al., 1998). This technique projects sentences in a semantic space in which each dimension roughly corresponds to a semantic field. This makes it possible to better measure the semantic similarity between sentences, since it can capture lexical chains through lexical repetitions, even through synonyms or hyponyms. However, this method cannot detect cohesive clues such as ellipsis, pronominal anaphora, substitution, causal conjunction, etc. Folz et al. (1998) were the first to apply this technique to readability, by computing the average similarity between each pair of sentences in a text. This variable was also included in Coh-Metrix (Graesser et al., 2004), along with similar measures such as word overlap, noun overlap, stem overlap, and argument overlap. However, the efficiency of this variable for readability was not assessed before Pitler and Nenkova (2008), who measured its association with text difficult and obtained a non significant correlation ($r = -0.1$). Later, McNamara et al. (2010) reached a similar conclusion, showing that an LSA-based variable has not much of a predictive power. On the opposite, François and Fairon (2012; 2013) obtained a higher correlation ($r = 0.63$) for an L2 corpus, while Dascalu et al. (2013) got good discriminating features using both LSA and LDA (Latent Dirichlet Allocation), when classifying TASA (Touchstone Applied Science Associates) texts.

An alternative approach to LSA, Lexical Tightness (LT), was suggested by Flor et al. (2013). They define the LT of a text as the mean value of the Positive Normalized Pointwise Mutual Information for all pairs of content-word tokens in a text. It represents "the degree to which a text tends to use words that are highly inter-associated in the language". They obtained a good correlation between this new cohesive metric and the grade levels on two corpora (respectively $r = -546$ and $r = -0,441$). Interestingly, they also show that LT works better to discriminate between literary texts than informative ones.

Another approach is to detect co-reference chains and compute some of their characteristics. Barzilay and Lapata (2008) considered a text as a matrix of discourse entities present in each sentence. The cohesive level of a text is then computed based on the transitions between those entities. Pitler and Nenkova (2008) implemented this model through 17 readability variables, but none was significantly correlated with difficulty. Feng et al. (2009) also replicated this technique, without getting more efficient features. Dascalu et al. (2013) computed other characteristics of lexical chains and co-reference pairs (such as the number of chains, the distance between entities, the average word length of entities, etc.). However, with these features, they only reached a precision of respectively 0.367 and 0.384 for a six-class classification problem.

Todirascu et al. (2013) argued that these mixed results might be due to approximations of the NLP systems, since automatically annotating co-reference chains remains a challenge. They manually annotated co-reference chains in 20 texts and correlated various characteristics of lexical chains with the difficulty of these texts. They showed that considering the type of entities, and not only their syntactic transitions, could be valuable. However, only four features appeared to be significantly correlated with difficulty, possibly due to the limited size of their corpus.

3 Methodology

Faced with this mixed findings in the literature regarding the efficiency of cohesive features for the assessment of text readability, our goal is to further investigate this issue. In particular, we present experiments focusing on cohesive features : anaphora chains, reference chains and lexical chains (evaluating sentence similarity).

For this purpose, we followed three steps: (1) we manually annotated a corpus of 83 French texts with co-reference chains and anaphoric chains; (2) we applied RefGen (Longo and Todirascu, 2010; Longo, 2013), a tool that automatically identifies co-reference and anaphoric chains in French, on the same corpus ; and (3) we evaluated the discriminating power of 65 coherence and cohesion-based features to assess text readability, comparing the results obtained on the manual and automatic annotation.

3.1 Corpus Description and Annotation

The corpus used in this study is a subset of the corpus of FFL (French as a Foreign Language) texts gathered by François (2009), which includes 2,160 texts extracted from 28 FFL textbooks. All the textbooks comply with the Common European Framework of Reference for Languages (CEFR), a standard scale for foreign language education in Europe that uses 6 levels (A1 to C2). Therefore, each text was assigned the level of the textbook it came from. In this study, we use a stratified sampling to select informative texts and narrative texts from the levels A2 to C1 (about 11 texts for each combination of level and genre).

In a second step, the corpus was annotated for co-reference chains (containing at least three mentions) and anaphoric chains (two mentions) by six human annotators, following an annotation guide. The annotation process was as follows: first, all mentions were detected, then we assigned an identification number to the chain containing the mention, finally the syntactic role as well as the type of the mention were annotated (see Figure 2 for an example of the annotation format). We used 16 different mention categories (e.g. proper names, indefinite NP, definite NP, personal pronouns, etc.) and 6 syntactic functions: S-subject, OD - direct object, OI - indirect object, CN - genitive, Mod - modifier, and X - other functions. Additionally, we annotated adverbs (*ici, là-bas*), resumptive anaphora or groups (the pronoun *ils* in Fig. 2 refers to the group composed of Antoine and Catherine).

Based on these guidelines, a common batch, composed of 10 randomly selected files, was annotated by all the annotators. It was used to identify annotation divergences between annotators[1] and to correct the annotation guide. We computed the overall inter-annotator agreement on this common batch using the mean Krippendorff's alpha on each text and we obtained 0.47, which corresponds to a *moderate* agreement between annotators. Such value is however not unusual in co-reference annotation (Muzerelle et al., 2014). Then, following the annotation guide, each expert annotated a batch of 12 texts from the corpus. At the end of the process, the principal annotator checked all batches against the guidelines, thus creating the reference for our experimentation.

[Antoine]_1/S/NPr/partie(3) fait la connaissance de [Catherine]_2/CN/NPr/partie(3). [Antoine]_1/S/NPr est [un beau parleur]_1/X/GNI et [la jeune fille]_2/S/GND [s']_2/X/Pronref intéresse à [lui]_1/OI/Pron. [Ils]_3/S/Pron vont au cinéma ensemble.

Figure 2: Example of annotated data : the number of the entity, the syntactic function and the category, eventually the relation with other referents : []_nb/syn/category/relation.

3.2 Automatic Annotation

For the automatic annotation of co-reference chains, we used a rule-based tool which identifies co-reference chains for French written texts, RefGen (Longo and Todirascu, 2010), (Longo, 2013). RefGen is one of the few systems available for French[2]. This tool integrates a POS-tagger adapted for French,

[1]Common problems that arose are: incorrect delimitation of mentions, wrong labelling of mention type or of the syntactic functions, wrong chain delimitation and relation categories (anaphoric vs co-reference).

[2]Other systems for French were proposed by Lassalle and Denis (2011) - but it detects only bridging anaphora - and by Desoyer et al. (2014), whose system detects coreference in oral data.

TTL[3](Ion, 2007), which provides the lexical category, the lemmas and simple chunk annotations (noun phrases, verb phrases). RefGen applies a set of preprocessing tools to identify complex noun phrases, named entities and impersonal pronouns.

Using information from the preprocessing step, RefGen identifies candidates for low accessibility mentions (proper nouns or named entities, definite noun phrases, indefinite noun phrases) (Ariel, 2001). These candidates open new co-reference chains. Anaphoric candidates with a high accessibility (personal pronouns, reflexive pronouns, demonstrative determiners or possessive determiners, demonstrative pronouns) are compared with possible antecedents. If the pair of candidates satisfies a complex set of syntactic, morpho-syntactic and semantic constraints, then the pair is included in a co-reference chain.

RefGen identifies almost all of the manually annotated categories, with the exception of resumptive anaphora. Concerning demonstrative NPs, the tool identifies only simple cases of antecedence (those with the same lexical head *le chien - ce chien*). Another significant drawback of this tool is that it is not able to handle complex referents such as groups or collections of objects. Adverbs are not considered as potential mentions by the tool.

3.3 Features

We replicated most features introduced in the literature described in section 2 and added new variables: the proportion of deictic pronouns, of resomptive anaphora and of adverbs, as well as the probability that a specific type of mention might open a co-reference chain in a given text. We ended up with 67 variables, divided in six classes :

1. **POS tag-based variables**: Pronouns and articles are crucial elements of cohesion. We computed 10 variables based on these parts-of-speech, namely the ratio of pronouns and nouns (1); the average proportion of pronouns per sentence (2) and per word (3); the average proportion of personal pronouns per sentence (4) and per word (5); the average proportion of possessive pronouns per sentence (6) and per word (7); the average proportion of definite articles per sentence (8), per word (9), and the ratio of definite articles with respect to the total number of articles (10). We also computed the ratio of proper names per word (11).

2. **Lexical cohesion measures**: We replicated several methods aimed at measuring lexical cohesion in a text as the average cosine similarity between adjacent sentences. These sentences were projected either in a word space, transformed with tf-idf (term frequency-inverse document frequency) only, or in a concept space, which was obtained with LSA. We defined 6 features, taking into account various linguistic entities: the inflected forms in the texts (word overlap) (12); the lemmas (13); only the nouns, proper names, and pronouns, either through their lemmas (14), or their inflected forms (15); a token-based LSA (16) and a lemma-based LSA (17).

3. **Entity cohesion**: Mentions of co-reference chains are often found in adjacent sentences and they often have the same syntactic function as the antecedent found in the previous sentence. For example, a proper noun is the subject of sentence *n* and the anaphoric pronoun referring to it is often the subject of sentence *n+1* (" Subject to Subject" transition). However, the syntactic functions of mentions might change across sentences : the object of the sentence *n* becomes the subject of the next sentence. Drawing from Pitler and Nenkova (2008)'s work, we replicated several variables evaluating the relative frequency of the possible transitions between the four syntactic functions played by the entity in sentences *n* and *n+1* : subject (S), object (O), other complements (X), and (N) when the entity is absent in the next sentence (variables 18 to 29), but also the number of transitions (30).

4. **Entity density**: We computed the average proportion of referring entities included in co-reference chains (simple and complex noun phrases, pronouns, etc.) per document normalized by the number of words (31), the proportion of the number of entities per document normalized by the number of words (32), the proportion of unique entities per document normalized by the number of words (33), and the average number of words per entity normalized by the number of words (34).

[3]Tokenizing, Tagging and Lemmatizing free running texts

5. **Co-reference chain properties**. We included several properties of co-reference chains: the proportion of various types of mentions (variables 35–46): indefinite NP, definite NP, proper names, personal pronouns, possessive determiners, demonstrative determiners, reflexive pronouns, relative pronouns, NPs without a determiner, indefinite pronouns, demonstrative pronouns, the average length of reference chains. The proportion of the opening mentions of the co-reference chains are also computed (variables 47–57): indefinite NPs, definite NPs, proper names, NPs without a determiner, demonstrative NPs but also pronouns (personal, demonstrative, indefinite, relative), possessives. As we mentioned in section 2.2., the composition and the structure of the co-reference chains are influenced by the genres or the type of the texts. These variables are used to evaluate the correlation between text types and the various types of mentions. Additionally, for the manually annotated corpus, we count additional features such as the proportion of specific deictic pronouns (such as *en,y*) (58), the proportion of adverbs as mentions (59), the resumptive pronouns (60), complex mentions (including groups or collections) (61). We compute also the proportion of these categories being used to open a new chain (variables 62–65).

6. **Classic features** : Finally, we replicated two efficient features from the readability literature as a baseline: the mean number of word per sentence or NMP (66), which provides an indication of the syntactic complexity, and a unigram model (67), estimating the vocabulary difficulty.

4 Results

We assessed the efficiency of our cohesive features through three devices. First, we computed their Spearman correlation with the CEFR levels of the texts in our corpus (see Table 1) in order to evaluate their informativeness when considered in isolation. Second, we computed a semi-partial correlation ($sr_{k(66,67)}$) (Kerlinger and Pedhazur, 1973, 92) between the target variable and the text CEFR levels, while controlling for the two classic variables (NMP and unigram). The reason for this analysis had been put forward by Boyer (1992) who said "it is conceivable that there are relations between the surface features of the text measured by [classic] readability formula and text characteristics of higher level". Therefore, semi-partial correlation will help determine whether our variables contribute to text readability prediction with additional information besides sentence length and word frequency. Third, all significant variables as regards the semi-partial correlation have been combined within a readability model and compared with a classic readability formula. In this section, we will first discuss the efficiency of the variables on the manually annotated corpus, then on the one automatically annotated with RefGen, then modelling experiments are discussed.

4.1 Results on the Manually-annotated Corpus

First, simple variables measuring the use of pronouns and articles based on POS-tagged information are correlated with text readability (e.g. nb. of pronouns per sentence: $\rho = 0.24$; nb. of definite articles per sentence: $\rho = 0.22$). This effect was also found by Todirascu et al. (2013), but it is likely to be due to sentence length because the semi-partial correlations – when controlling for sentence length – are not significant neither for the number of pronouns per sentence ($sr = 0.14$) nor for the number of definite articles per sentence ($sr = -0.11$). Besides, the correlations for the number of pronouns ($\rho = 0.04$) and of definite articles ($\rho = 0.01$) are nonsignificant when normalized at the word level. The situation is the same on narrative and informative texts.

Interestingly, semi-partial correlation are significant for the number of pronouns per word ($sr = 0.25$) and for the number of personal pronouns ($sr = 0.23$), on all texts. The more difficult a text is, the more pronouns are used. Pronoun resolution requires background knowledge and high reading proficiency, which explains their higher frequency in difficult texts, even when text length is controlled. For comparison, Pitler and Nenkova (2008) found no effect for both variables.

There is a very interesting pattern of results for lexical coherence measures. As regards the correlation, there is a clear distinction between the four features based on word overlap (and their variation) – none of which are significant –, and the two LSA-based features, which are significant. The LSA-based feature using lemma is the second best feature after NMP on the whole corpus, while the token variant is the

very best feature for the informative texts. Such efficiency is in line with previous results (François and Fairon, 2012; Dascalu et al., 2013), but the semi-partial correlation offers a more nuanced figure, since the features based on LSA are not efficient when word frequency and sentence length are controlled. On the other hand, a more naive approach such as word overlap appears to provide more specific information as shown by the semi-partial correlations computed on informative texts ($sr = -0.41$ for lemma overlap and $sr = -0.4$ for NP word overlap).

Another interesting feature is the number of chains, which is negatively correlated with text complexity for all texts ($\rho = -0.22$) and narrative texts ($\rho = -0.35$): the lower the number of chains is (which means less referents but longer chains), the more difficult a text is. Besides, the ratio of unique entities is a valuable feature for all texts ($\rho = -0.26$) as well as for narrative texts ($\rho = -0.38$). More difficult narrative texts have a lower number of unique entities, probably because they include longer descriptions of the same elements, psychological introspection, or repetitions of the same mention. However, semi-partial correlations show that these variables are redundant with sentence length and word frequency, whereas the average word length of entity then becomes significant ($sr = -0.28$).

On the contrary, the proportion of the various syntactic transition types in a text hardly conveys information about text difficulty. Out of the 12 types of transitions, only "Object to None" is significant for all texts ($\rho = 0.24$) and for informative texts ($\rho = 0.42$). This feature means that the distance between two consecutive mentions of the same chain is larger than one sentence, a phenomenon that often occurs in informative texts where the same referent may be repeated across the text, even after several sentences. It should also be mentioned that the "Object to Object" transition was found significant ($\rho = 0.41$ and $sr = 0.40$) exclusively in narrative texts. On the whole, we are much in line with the negative results of Pitler and Nenkova (2008) as regards this category of variables.

Finally, Todirascu et al. (2013) suggested to consider the proportion of the different types of the entities and found both the proportion of pronouns and indefinite NP to be useful features. Globally, variables in this category show a poor correlation in our experiment. The type of entities that emerged as noticeable is the proportion of demonstrative NP ($\rho = 0.22$) in all texts, which nevertheless loses significance on the two sub-genre corpora as well as when sentence length and word frequency are controlled ($sr = -0.06$). It is also interesting to note that the proportion of the first mention of a chain being specific deictic pronouns is significant for all texts ($\rho = 0.22$), and even stronger when the two classic variables are controlled ($sr = 0.24$). A summary of the correlations for the most interesting features is available in Table 1.

4.2 Results on the Automatically-annotated Corpus

When comparing the manual and the automatic annotations, when relevant,[4] we find some features in which the two approaches converge such as the number of transitions, the proportion of mentions being a pronoun or a proper noun, etc. These are cases corresponding to easier phenomena to detect automatically. Conversely, some variables demonstrate large discrepancies in effectiveness between their manual and automatic versions, such as the average word length of entities, the proportion of "Object to Object" transitions, the proportion of definite mention, or the proportion of the first mention being a definite or a proper noun.

In such cases, especially in narrative texts, the automatic version appears to be more efficient, even when the semi-partial correlation is concerned. This is probably a side effect of annotation errors by RefGen, but as a result, more variables appear significant with the automatic annotation. Text complexity in narrative texts is thus correlated with the proportion of definite articles ($\rho = 0.37$) and proper nouns ($\rho = -0.39$), the proportion of chains starting with definite articles ($\rho = 0.52$) or proper noun ($\rho = -0.38$), the average word length of entities ($\rho = -0.48$) as well as with the proportion of syntactic transition "O to O" ($\rho = 0.41$). For informative texts, text difficulty is positively correlated with the proportion of transitions "O to N" ($\rho = 0.32$) and the proportion of first mention being a proper noun ($sr = 0.32$), but negatively correlated with average word length of entities ($\rho = -0.31$).

[4]Several features – those from the first, second, and sixth class in Section 3.3–, were only computed automatically. As a consequence, Table 1 provides only one value per subcorpus.

Variables	corpus (all)		corpus (narr.)		corpus (inf.)	
	manual	auto	manual	auto	manual	auto
Pronoun/sent.	0.24* / 0.14		0.32* / 0.24		0.38* / 0.16	
Pronoun/word	0.04 / 0.25*		0.22 / 0.26		0.03 / 0.17	
Pers. pron./word	-0.04 / 0.23*		0.07 / 0.23		-0.13 / 0.14	
LSA_Token	0.32** / 0.12		0.13 / 0.04		0,52*** / 0.23	
LSA_Lemma	0.28** / 0.14		0.20 / 0.01		0,43** / 0.23	
coRef_Lemma	-0.15 / -0.16		0.06 / 0		-0,4** / -0.41*	
coRef_NP_Lemma	0 / -0.11		0.25 / 0.07		0.31* / -0.4*	
nb. transitions	-0.15 / 0.12	0.10 / 0.18	-0.15 / 0.14	0.14 / 0.16	-0.17 / -0.10	0.07 / 0.08
X to N	0.12 / -0.07	0.20 / 0.21	0.26 / -0.03	0.27 / 0.3	-0.02 / -0.07	0.13 / 0.16
O to O	0.06 / 0.06	0.21 / 0.12	-0.09 / -0.06	0,41** / 0.40**	0.23 / 0.1	0,04 / -0.04
Nb. chains/words	-0.22 / -0.21	0.11 / 0.10	-0.35* / -0.33*	0.30 / 0.20	-0.11 / -0.1	-0.03 / -0.1
Nb. unique entity	-0.26* / -0.15	0.12 / 0.10	-0.38* / -0.24	0.32 / 0.21	-0.17 / -0.11	-0.04 / -0.12
Av. length of entity	-0.14 / -0.28*	-0.34** / -0.26*	-0.15 / -0.10	-0.48* / -0.36*	-0.20 / -0.31*	-0.31* / -0.23
Definite	0 / -0.26*	0.18 / -0.01	0.12 / -0.06	0.37* / 0.04	-0,20 / -0.3	0.04 / -0.05
Dem	0.22* / -0.06	NA / NA	0.21 / 0.07	NA / NA	0.2 / -0.07	NA / NA
Indefinite	0.04 / -0.05	-0.12 / -0.26*	-0.14 / -0.20	-0.11 / -0.14	0.21 / 0	-0.14 / -0.27
Pron	0.10 / 0.28*	0.11 / 0.21	0.19 / 0.25	0.18 / 0.21	0.18 / 0.28	0.07 / 0.15
Proper	-0.12 / -0.07	-0.05 / 0	-0.30 / -0.21	-0,39* / -0.37*	0 / 0.11	0.22 / 0.25
1st Definite	0.09 / -0.07	0.23* / 0.15	0.28 / 0.24	0.52*** / 0.37*	-0.17 / -0.20	0.03 / -0.05
1st PRONSPEC	0.22* / 0.24*	NA / NA	0.33* / 0.31	NA / NA	NA / NA	NA / NA
1st Proper Noun	0.02 / 0.07	-0.05 / 0	-0.07 / -0.09	-0.38* / -0.32*	0.07 / 0.25	0.24 / 0.32*
NMP	0,35** / NA		0,27 / NA		0,50** / NA	
unigram	-0,25* / NA		-0,32* / NA		-0,28 / NA	

Table 1: Spearman correlation / semi-partial correlation computed for each variable and difficulty. Significance of the correlation coefficient is indicated as follows: $^{*}: < 0.05$; $^{**}: < 0.01$; and $^{***}: < 0.001$.

4.3 Cohesion and Coherence Variables in Readability Models

In this section, the efficiency of our cohesive and coherence features for readability is tested in the context of actual readability models. On the corpus of 83 texts, we defined 4 sets of features to be used either for a classification task (with SVM classifier) or a regression task (with ϵ-SVR). The first set, that serves as a baseline, includes only sentence length (NMP) and a unigram model (ML1)[5] model have been trained. The second set includes NMP, ML1, and all variables that have been detected as significant by the correlation on the manual corpus (parsimonious_manu) and was trained on the manually annotated version of the data. The third set includes NMP, ML1 and all variables that have been detected significant by the correlation on the automatic corpus (parsimonious_auto) and was trained on the automatically annotated version of the data. The last set includes all variables (full model) and was trained on the manually annotated data, as we are interested to get the best performance possible. The optimal kernel and associated meta-parameters for all models (see Table 2) were selected via a grid-search conducted using a 10-fold cross-validation process. Once the best parameters were known, the performance of each model were then measured with two metrics – accuracy and mean average error (MAE).

Feature set	nb. variables	Model	kernel	C	Others param.	accuracy	MAE
baseline	2	SVM	RBF	5	$\gamma = 0.5$	43.6	0.89
baseline	2	SVR	polynomial	5	deg = 2; $\epsilon = 0.5$	/	1.03
parsimonious_manu	10	SVM	linear	0.1	/	43.4	0.81
parsimonious_manu	10	SVR	RBF	100	$\epsilon = 0.1$; $\gamma = 0.0001$		0.91
parsimonious_auto	8	SVM	RBF	500	$\gamma = 0.1$	41.5	0.85
parsimonious_auto	8	SVR	RBF	100	$\epsilon = 0.5$; $\gamma = 0.0001$	/	0.94
full model	67	SVM	polynomial	5	deg. = 2	40.6	0.89
full model	67	SVR	RBF	5	$\epsilon = 1$; $\gamma = 0.01$	/	0.93

Table 2: Accuracy and values of meta-parameters for the 4 models.

First, all classification models perform better than their regression counterparts. However, even for the former, no model using coherence or cohesive features is able to overcome a simple model based

[5] As those variables have been automatically computed, the results are the same for both versions of the corpus (manually and automatically annotated).

on sentence length and word frequency. The one that performs better is the SVM parsimonious model based on the manual annotation (MAE = 0.81), but compared to the SVM baseline (MAE = 0.89), the difference is not significant using a paired T-test ($t = -1.43; p = 0.19$). It is also interesting to note that using automatically detected features seems to slightly degrade performance compared to the manual annotation, although such difference is clearly not significant with a paired T-test ($t = -0.78; p = 0.45$). This is the case even though automatically-computed variables were characterized by slightly better correlations as found in Section 4.2.

5 Discussion and Conclusion

To conclude, we have performed a detailed analysis of 65 cohesive features commonly used in the readability literature. The parameterization of these variables requires heavy NLP processing and is prone to errors. We showed that nevertheless they do not seem to contribute much to the prediction of text readability when compared with simple predictors such as word frequency and sentence length. On the one hand, 6 features only were found to be significant by semi-partial correlation (when sentence length and word frequency were controlled for). On the other hand, integrating the best cohesive features in a readability model did not bring significant improvement over a simple baseline on our French data. The first lesson learned is that such kind of features, although quite popular in the literature, have an efficiency that is subject to caution, at least in the context of readability prediction, as it was already reported by some of the previous studies.

Another interesting insight of our analysis is the use of semi-partial correlation to analyze the efficiency of variables for readability. Previously, some authors (Pitler and Nenkova, 2008; François and Fairon, 2012; Todirascu et al., 2013) only used Pearson or Spearman correlations to identify and quantify the effect of a text characteristic on readability and we showed that, as was suggested by Boyer (1992), higher textual dimensions can be much correlated with lexical or syntactic features. A good example in this regard was the impact of LSA-based features. Similar to previous studies (François and Fairon, 2012; Dascalu et al., 2013), we found a large effect for this predictor, which completely vanished once word frequency and sentence length were controlled for. This allowed us to reconcile to some extent contradictory findings in this regard.

Our experiments also showed large differences between the manual and automatic annotation of lexical chain properties, which seems to lead to a loss of performance when such predictors are included into a full readability model. This should however be replicated using different co-reference extraction tools, as some of the errors are typical of the RefGen tool that we used.

Finally, the third question that we planned to investigate was whether the behavior of lexical and co-reference chains differs in narrative and informative texts, in relation to text difficulty. We noticed that the variables significantly correlated with difficulty vary depending on the genre of texts. On narrative texts, the number of chains, the number of unique entities or the ratio of first mention being a specific deictic pronoun were relevant, whereas the average word length of entities, the LSA-based features and the word overlap features were relevant for informative texts.

However, there are some limitations to our study and further investigation would be necessary before discarding co-reference chain-based features for readability. First, we have experimented on an L2 corpus, while the cohesive aspects might be more relevant for L1 texts. Moreover, the study was performed on French and the results might vary from one language to another (although our findings are mostly in line with results on English). Finally, it is not excluded that some properties of the lexical and co-reference chains that we did not consider (e.g. mean distance in words between the various entities of a chain) could demonstrate a stronger discriminative power.

References

M. Ariel. 2001. Accessibility theory: An overview. In T. Sanders, J. Schliperoord, and Spooren W., editors, *Text representation. Human cognitive processing series*, pages 29–87. John Benjamins.

R. Barzilay and M. Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

I. Berzlnovich and G. Redeker. 2012. Genre-dependent interaction of coherence and lexical cohesion in written discourse. *Corpus Linguistics and Linguistic Theory*, (8):183–208.

J.R. Bormuth. 1969. *Development of Readability Analysis*. Technical report, Projet n.7-0052, U.S. Office of Education, Bureau of Research, Department of Health, Education and Welfare, Washington, DC.

J.Y. Boyer. 1992. La lisibilité. *Revue française de pédagogie*, 99(1):5–14.

S. Carter-Thomas. 1994. Langue de spécialité : cohésion, culture et cohrence. *ASp*, 5-6:61–67.

J.S. Chall and E. Dale. 1995. *Readability Revisited: The New Dale-Chall Readability Formula*. Brookline Books, Cambridge.

M. Charolles. 1995. Cohesion, coherence et pertinence de discours. *Travaux de Linguistique*, 29:125–151.

E. Dale and J.S. Chall. 1948. A formula for predicting readability. *Educational research bulletin*, 27(1):11–28.

M. Dascalu, P. Dessus, Ş. Trausan-Matu, M. Bianco, and A. Nardy. 2013. Readerbench, an environment for analyzing text complexity and reading strategies. In *Artificial Intelligence in Education*, pages 379–388. Springer.

A. Desoyer, F. Landragin, I. Tellier, A. Lefeuvre, and J.-Y. Antoine. 2014. Les coréférences à l'oral : une expérience d'apprentissage automatique sur le corpus ancor. *TAL*, 55:97–121.

L. Feng, N. Elhadad, and M. Huenerfauth. 2009. Cognitively motivated features for readability assessment. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 229–237.

R. Flesch. 1948. A new readability yardstick. *Journal of Applied Psychology*, 32(3):221–233.

M. Flor, B. Beigman Klebanov, and K. M. Sheehan. 2013. Lexical tightness and text complexity. In *Proceedings of the Workshop on Natural Language Processing for Improving Textual Accessibility*, pages 29–38.

P.W. Foltz, W. Kintsch, and T.K. Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse processes*, 25(2):285–307.

T. François and C. Fairon. 2013. Les apports du TAL à la lisibilité du français langue étrangère. *Traitement Automatique des Langues (TAL),*, 54(1):171–202.

T. François. 2009. Combining a statistical language model with logistic regression to predict the lexical and syntactic difficulty of texts for FFL. In *Proceedings of the 12th Conference of the EACL : Student Research Workshop*, pages 19–27.

T. François and C. Fairon. 2012. An "AI readability" formula for French as a foreign language. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP 2012)*, pages 466–477.

A.C. Graesser, D.S. McNamara, M.M. Louwerse, and Z. Cai. 2004. Coh-Metrix: Analysis of text on cohesion and language. *Behavior Research Methods, Instruments, & Computers*, 36(2):193–202.

B. Grosz, A. Joshi, and S. Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational linguistics*, 21(2):203–225.

M.A.K. Halliday and R. Hasan. 1976. *Cohesion in English*. Longman, London.

G. Hirst and D. St-Onge. 1998. Lexical chains as representation of context for the detection and correction of malapropisms. In C. Fellbaum, editor, *WordNet:An Electronic Lexical Database and Some of its Applications*. The MIT Press, Cambridge, MA.

J. Hobbs. 1979. Coherence and coreference. *Cognitive Science*, 3(1):67–90.

R. Ion. 2007. *Word Sense Disambiguation Methods Applied to English and Romanian*. Ph.D. thesis, Romanian Academy, Bucharest.

S. Kemper. 1983. Measuring the inference load of a text. *Journal of Educational Psychology*, 75(3):391–401.

F.N. Kerlinger and E.J. Pedhazur. 1973. *Multiple regression in behavioral research*. Holt, Rinehart and Winston, New York.

W. Kintsch and D. Vipond. 1979. Reading comprehension and readability in educational practice and psychological theory. In L.G. Nilsson, editor, *Perspectives on Memory Research*, pages 329–365. Lawrence Erlbaum, Hillsdale, NJ.

W. Kintsch, E. Kozminsky, W.J. Streby, G. McKoon, and J.M. Keenan. 1975. Comprehension and recall of text as a function of content variables1. *Journal of Verbal Learning and Verbal Behavior*, 14(2):196–214.

T.K. Landauer, P.W. Foltz, and D. Laham. 1998. An introduction to latent semantic analysis. *Discourse processes*, 25(2):259–284.

E. Lassalle and P. Denis. 2011. Leveraging different meronym discovery methods for bridging resolutionin french. In *Anaphora Processing and Applications, Proceedings of the 8th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 2011)*.

L. Longo and A. Todirascu. 2010. Genre-based reference chains identification for french. *Investigationes Linguisticae*, 21:57–75.

L. Longo and A. Todirascu. 2014. Vers une typologie des chaînes de référence dans des textes administratifs et juridiques. *Langages. Les chanes de référence*, pages 79–98.

L. Longo. 2013. *Vers des moteurs de recherche intelligents : un outil de détection automatique de thèmes. Méthode basée sur l'identification automatique des chaînes de référence*. Ph.D. thesis, University of Strasbourg.

R.A. Mason and M.A. Just. 2004. How the brain processes causal inferences in text. *Psychological Science*, 15(1):1–7.

D.S. McNamara, M.M. Louwerse, P.M. McCarthy, and A.C. Graesser. 2010. Coh-Metrix: Capturing linguistic features of cohesion. *Discourse Processes*, 47(4):292–330.

B.J.F. Meyer. 1982. Reading research and the composition teacher: The importance of plans. *College composition and communication*, 33(1):37–49.

J.R. Miller and W. Kintsch. 1980. Readability and recall of short prose passages: A theoretical analysis. *Journal of Experimental Psychology: Human Learning and Memory*, 6(4):335–354.

J. Muzerelle, A. Lefeuvre, E. Schang, J.-Y. Antoine, A. Pelletier, D. Maurel, I. Eshkol, and J. Villaneau. 2014. Ancor_centre, a large free spoken french coreference corpus: Description of the resource and reliability measures.

E. Pitler and A. Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 186–195.

M. Poesio, R. Stevenson, B. DiEugenio, and J. Hitzeman. 2004. Centering : A parametric theory and its instantiations. *Computational Linguistics*, 30(3):309–363.

C. Schnedecker. 1997. Nom propre et chaînes de référence. *Recherches Linguistiques*, 21.

C. Schnedecker. 2005. Les chaînes de référence dans les portraits journalistiques : éléments de description. *Travaux de Linguistique*, 2:85–133.

A. Todirascu, T. François, N. Gala, C. Fairon, A.-L. Ligozat, and D. Bernhard. 2013. Coherence and cohesion for the assessment of text readability. *Natural Language Processing and Cognitive Science*, pages 11–19.

T. Van Dijk. 1977. *Text and Context: Exploration in the Semantics and Pragmatics of Discourse*. Longman, London.

B.L. Zakaluk and S.J. Samuels. 1988. Toward a New Approach to Predicting Text Comprehensibility. In B.L. Zakaluk and S.J. Samuels, editors, *Readability: Its Past, Present and Future*, pages 121–144. International Reading Association, Newark, Delaware.

Named Entity Recognition for Linguistic Rapid Response in Low-Resource Languages: Sorani Kurdish and Tajik

Patrick Littell, Kartik Goyal, David Mortensen, Alexa Little, Chris Dyer, Lori Levin
Carnegie Mellon University
Language Technologies Institute
5000 Forbes Ave., Pittsburgh PA 15213
plittell@cs.cmu.edu, kartikgo@cs.cmu.edu, dmortens@cs.cmu.edu
alexanicolelittle@gmail.com, cdyer@cs.cmu.edu, lsl@cs.cmu.edu

Abstract

This paper describes our construction of named-entity recognition (NER) systems in two Western Iranian languages, Sorani Kurdish and Tajik, as a part of a pilot study of *Linguistic Rapid Response* to potential emergency humanitarian relief situations. In the absence of large annotated corpora, parallel corpora, treebanks, bilingual lexica, etc., we found the following to be effective: exploiting distributional regularities in monolingual data, projecting information across closely related languages, and utilizing human linguist judgments. We show promising results on both a four-month exercise in Sorani and a two-day exercise in Tajik, achieved with minimal annotation costs.

1 Introduction

This paper describes our rapid construction of NER systems, as a part of a pilot study of *Linguistic Rapid Response* to potential emergency humanitarian relief situations. When a disaster strikes a community that speaks a low-resource language without an existing NLP infrastructure, how long would it take to put such an infrastructure, however imperfect, in place? What kinds of systems can be in place within 24 or 48 hours, and what levels of performance can we expect? What resources – besides existing gazetteers, parallel corpora, etc. – can be assembled in this timeframe and brought to bear on NER?

Contemporary techniques for creating natural language processing (NLP) tools are dominated by supervised learning approaches, in which large quantities of high quality data are annotated in a task-specific fashion and utilized along with manually-built collaborate resources like WordNet, Ontonotes, etc. These techniques have provided very good performance, but with 7,000 languages in the world these resources cannot feasibly be compiled in advance, and could not be assembled within an emergency timeframe.

Following many examples of "surprise language exercises" (Oard, 2003), the work described in this paper tackles the problem of rapid development of important NLP tools and resources, with a focus on NER, for low resource languages. We assume that for a "low resource" language, there is some monolingual corpus data and little to no annotated data for supervised training. A major theme underlying this work is a focus on building "omnivorous" models and pipelines that, instead of relying on elaborate and robust linguistic resources compiled ahead of time, try to opportunistically incorporate linguistic theory, informal and non-expert intuitions about the language and task at hand, and resources adapted from closely related languages, all while avoiding extensive manual annotation.

An important feature of our work is the use of human linguists who do not speak the language but are familiar with the structure of human languages in general, have some knowledge about the language family in question, and can absorb facts about the language quickly by reading reference grammars and looking at data. Specifically, in this project, linguists worked interactively with unsupervised morphology induction, annotated named entities, and identified thresholds for automatic tagging of multi-word named entities.

We illustrate this approach for two low resource languages, Sorani Kurdish and Tajik, provided during two surprise language evaluations. Software development and Sorani NER processing took place over

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 998–1006, Osaka, Japan, December 11-17 2016.

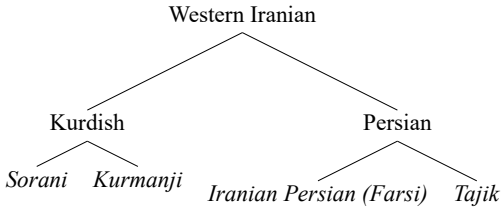


Figure 1: Sorani, Tajik, and selected relatives

a four-month period, while the Tajik surprise-language event took place in only 36 hours. Hence, all techniques described in this paper, whether performed by machines or humans or both, have runtimes in minutes or hours rather than days.

2 Data sources

2.1 Sorani Kurdish

Sorani (or "Central") Kurdish is a Western Iranian language in the Kurdish family, spoken by about 6.7 million people in Iraqi Kurdistan and the Kurdistan Province of Iran. Unlike other Kurdish languages, but like the more distantly related Persian (or "Farsi"), it is written in a Perso-Arabic script, albeit with modifications (like additional vowel glyphs) that make it more suitable for writing Sorani. We obtained Sorani monolingual data from the LCTL language pack (Simpson et al., 2008).

2.2 Kurmanji Kurdish

Some of the challenge of Sorani NER is orthographic in nature, since its Perso-Arabic script, while closer to the phonemic form of words than most other Perso-Arabic scripts, still has some significant ambiguities in vowel representation, and lacks an uppercase-lowercase distinction, which is an important feature for NER. In order to partially mitigate these ambiguities, we also made use of Kurmanji (or "Northern") Kurdish data from the Pewan news corpus (Esmaili et al., 2013). Kurmanji is spoken primarily in eastern Turkey by about 20 million speakers, and unlike Sorani is written in a Roman script. Sorani and Kurmanji are sometimes described as dialects of the same languages, but sometimes described as different languages due to their significant morphological differences.

2.3 Tajik Persian

Tajik is a variety of Persian spoken primarily in Tajikistan by about 8 million speakers, and is written primarily in Cyrillic script. We obtained Tajik monolingual data from the Leipzig corpus of news crawls (Biemann et al., 2007).

2.4 IPA representations

So that resources in different languages and scripts could be more directly compared, and so that judgments about the data could be made rapidly by linguists without native proficiency in the Perso-Arabic and Cyrillic writing systems, we produced representations of the Sorani, Kurmanji, and Tajik data in the International Phonetic Alphabet (IPA). To disambiguate ambiguous Sorani forms, we used a conditional random field (CRF) (Lafferty et al., 2001) that utilized a combination of human judgments, universal phonetic features, and language models of related languages; we describe this "IPAization" process in more detail in Mortensen et al. (2016) and Littell et al. (2016).

3 Named Entity Recognition

For the NER task, we focused on identifying mentions of persons (PER), locations (LOC), and organizations (ORG) in the textual data. Our core system is a CRF based system with L1-regularization, where x is the input sequence and output y is the appropriate tag sequence, and no features look beyond a history of length 1.

$$f(\mathbf{y} \mid \mathbf{x}) = \sum_{i=1}^{|\mathbf{x}|} f(\mathbf{x}, \mathbf{y}_i, \mathbf{y}_{i-1}). \quad (1)$$

Based on previous work in the area (Tjong Kim Sang and De Meulder, 2003; McCallum and Li, 2003; Sha and Pereira, 2003), we begin with a standard set of features commonly used for training NER systems:

- Current token and Current tag
- Previous token and Current tag
- Next token and Current tag
- Current+previous token and current tag
- Current+next token and current tag
- First five features but with previous tag
- First five features conjoined with both current and previous tags
- Contains foreign script characters
- Indicator features for tokens containing digits
- Features about capitalization information
- Prefix features

However, given the paucity of training data, these features are numerous and sparse, we do not expect the CRF model to perform well with standard features alone, even with L1 regularization.

Hence, keeping up with the general theme of this work, we aim to induce features in an unsupervised manner by exploiting the distributional regularities in the monolingual data, guiding the unsupervised sub-components to encode our intuitions and knowledge about the task and the target language, and utilize features from closely-related languages.

3.1 Gazetteers

For Sorani, the absence of a capitalization feature in its Perso-Arabic script posed a difficulty, as capitalization serves as a valuable feature for NER. However, as noted in §2, the closely related Kurmanji Kurdish is written in a Roman script that does distinguish capitalization.

Using the IPA representations of the Sorani and Kurmanji texts and a frequency-weighted edit distance algorithm, we inferred for each Sorani token a corresponding Kurmanji token (e.g., ⟨ɛvɣanistan,ɛfganistan⟩), and assigned to each Sorani token the capitalization frequency of the Kurmanji word (in this case, *Efganistan*, which had a capitalization frequency of 1.0). These features were used a "probabilistic gazetteer" in lieu of a real Sorani gazetteer, where the capitalization frequency is treated as if it represented the probability that a word occurred in a gazetteer. We describe this gazetteer inference strategy in greater detail in Littell et al. (2016).

For Tajik, we constructed a more traditional gazetteer using Tajik's relatively extensive Wikipedia. Since Wikipedia titles are linked between Wikipedias in different languages, we had parallel English and Tajik titles; we filtered the English titles by heuristics including capitalization, and used the corresponding Tajik titles as the gazetteer entries.

3.2 Unsupervised morphology induction

The Western Iranian languages are morphologically rich, with Sorani in particular having a high degree of morphological complexity (Walther, 2011; Esmaili and Salavati, 2013). In such languages, the presence of certain morphemes is strongly correlated with certain grammatical functions being present, which could be informative for the problem of identifying and discriminating named entities.[1]

While unsupervised induction of morphological grammars is a long-standing problem, the inferred morphological analyses typically diverge from conventional linguistic analyses rather substantially. We addressed this shortcoming by using feedback from human linguists using a modification of the interactive learning paradigm proposed by Hu et al. (2011). While superficially related to active learning (Settles,

[1] While Western Iranian languages also utilize prefixes and root modification, we concentrated here on suffixes alone; this simplifies the model and concentrates on those morphological alternations we believe more likely to be relevant to NER.

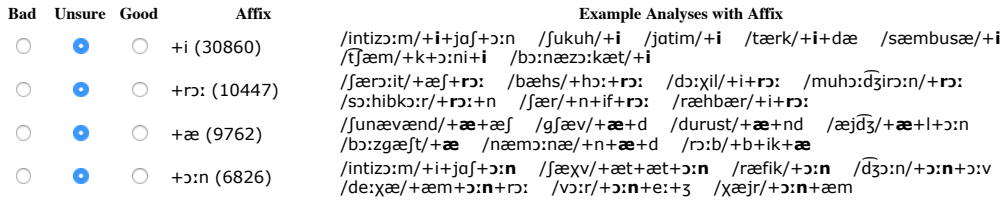


Figure 2: Screenshot of the feedback interface containing analyses for Tajik.

2012), the interactive paradigm charges the annotator with the task of identifying subjective systematic errors, instead of picking new instances to be annotated. In case of morphology learning, we let linguists give their judgment on the quality of affixes, which is used to constrain the hypothesis space and tune the parameters of the unsupervised learner. The feedback interface is shown in Fig. 2.

We used a hierarchical Bayesian model of morphological segmentation. Our prior expectations are (1) that stems should be more diverse than suffixes, but both should be reused when possible, (2) that individual suffixes are likely to be very short, and (3) that suffixes are likely to proceed in a characteristic order (e.g., *-ion* and *-al* are both English suffixes, but *-ion-al* is valid and occurs in many words, for example in *inspirational*, while *-al-ion* is not valid).

To encode the first two assumptions, we assumed that the distributions over stems and suffixes are governed by a Dirichlet processes with parameters set to encourage lower entropy samples for suffixes and higher entropy samples for stems. The base distribution $\mathsf{Word}(\lambda)$ is a process that generates a word by sampling a length from a Poisson distribution with mean λ and then choosing characters randomly for each position. To capture the fact that affixes proceed in a characteristic order, we in turn assumed these were generated by a bigram Markov process governed by a hierarchical Dirichlet process (Teh et al., 2006). The generative process is stated in Algorithm 1.

Algorithm 1 Morphological induction

$\boldsymbol{\theta} \sim \mathsf{DP}(\alpha_1, \mathsf{Word}(\lambda = 6))$
$\boldsymbol{\varphi} \sim \mathsf{DP}(\alpha_2, \mathsf{Word}(\lambda = 1))$
$\boldsymbol{\varphi}_{\cdot|x} \sim \mathsf{DP}(\alpha_3, \boldsymbol{\varphi})\ \forall x \in \Sigma^*$
for `each word` w `in surface vocabulary` V **do**
  `Draw # of suffixes` $\ell \sim \mathsf{Geom}(\rho = 0.9)$
  `Draw stem` $b \sim \mathsf{Cat}(\boldsymbol{\theta})$
  $s_{-1} = \langle \mathsf{b} \rangle$
  for `each suffix index` i `from 1 to` ℓ **do**
    `Draw affix` $s_i \sim \mathsf{Cat}(\boldsymbol{\varphi}_{\cdot|s_{-1}})$
    $w = w + s_i$
    $s_{-1} = s_i$
  end for
end for

Since our model does not depend on anything but word types, the observed data is just the vocabulary (in IPA form) of the target language, and the goal of inference is to find the distribution over segmentations given our model and the vocabulary. To do so, we use block Gibbs sampling (marginalizing the draws from the Dirichlet processes). Since we are considering all analyses of a word at once, constraints against certain morphemes are trivial to incorporate. For the experiments reported below, we ran 1000 iterations of Gibbs sampling, then obtained feedback followed by a further 1000 iterations twice.

3.3 Unsupervised class induction: Hard clustering

One way of reducing sparseness due to the lexicalization of the features is to map the types or tokens of the monolingual corpus to a smaller number of classes. We try to obtain mappings such that the tokens/types sharing similar characteristics are mapped to one class. For example, if "Lord" and "Lady" are mapped to the same class, then two different sequences "Lord Palmerston" and "Lady Grey" will share the class information and will, if "Palmerston" is annotated as a name in the training data, be more likely to predict "Grey" as a name as well. We focus on context-aware class induction, particularly modeling the classes

with a first-order Markovian assumption.

Brown et al. (1992) introduced a bottom-up agglomerative word clustering algorithm which generates a hard clustering (i.e., a word belongs to only one cluster). With this hard clustering assumption, we aim to achieve a clustering C that maximizes the log-likelihood of the data, $\log P(w_1, ...w_n, \mathbf{C}(w_1), ..\mathbf{C}(w_n))$, i.e.

$$\arg\max_{\mathbf{C}} \log \prod_{i=1}^{n} p(\mathbf{C}(w_i) \mid \mathbf{C}(w_{i-1})) \times p(w_i \mid \mathbf{C}(w_i))$$

We experimented with 500 and 1000 clusters. Manual qualitative analysis of these clusters revealed that they created several meaningful groups: foreign words, numbers, names, etc.

3.4 Unsupervised class induction: Soft clustering with the influence of collocations

In addition to the traditional "hard" Brown clusterings, we also experimented with soft clusterings where the parameters can be influenced by external knowledge, using Expectation Maximization (Dempster et al., 1977). We focused on influencing our NER system using information about collocations – bigrams that co-occur with frequency greater than chance – in the monolingual text. The intuition behind collocations is that many names of people ("barak obama"), places ("arabistani saudi"), and organizations ("bomdodi telefonhoi") are expected to be identified as collocations.

We warm-started with the distributions obtained by the Brown cluster algorithms, smoothed via additive (dirichlet) smoothing. Our pilot experiments showed that this resulted in better performance when compared to the performance with random initializations. The runtime (O($token * iterations * C^2$)) is higher than that of Brown algorithm (O($types * C^2 + tokens$)) because we estimate all the distributions of the probabilistic HMM using dynamic programming (Rabiner and Juang, 1986). Hence, subsequent models used interpolated stochastic batch updates (Liang and Klein, 2009) instead of batch updates so that the convergence is faster.

We used a likelihood ratio test (Dunning, 1993), which is a form of hypothesis testing that decides whether the second word in the bigram is unusually associated with the first word of the bigram or not, to determine an initial list of possible NE collocations. For both Sorani and Tajik, this measure results in desirable LR score graphs for bigrams with a distinct "elbow" for both the languages. However, determining the threshold, below which a collocation should not be considered genuine for the purposes of further steps, was done manually by a human linguist looking at IPA representations of the collocations. This is a judgment that need not be made by a native speaker, or even an expert in the language in question, but just someone with some familiarity with the language or language group, the ability to read IPA, and enough real-world knowledge to recognize when a list of two-word phrases in IPA switches from mostly referring to names and places, to referring to names and places only occasionally.

We use these collocations, as generated by the likelihood ratio test and thresholded by human judgment, to bias unsupervised class induction over words in the target language, so that collocations are encouraged to fall into the same clusters. To our knowledge, this work is the first to bias unsupervised class induction using collocation knowledge.[2]

EM based optimization allows us to bias the parameters of the HMM, to encourage collocations to fall into same clusters. Hence, whenever we observe collocations in the monolingual data during the E step, we use an Identity matrix as the transition matrix, i.e. $P(C(w_i) \mid C(w_{i-1})) = 1$. The M step is performed as usual, resulting in learning of parameters affected by the biased expectation counts.

As discussed above, constraining the collocation members to belong to the same clusters is attractive, but the collocations that we estimated automatically are certainly not pure. Hence, we introduce posterior regularization (PR) (Ganchev et al., 2010) into our HMM inference algorithm. We want the posterior of the HMM distribution to reflect the fact that adjacent tokens in identified collocations in the monolingual data tended to belong to same clusters.

[2]It should be noted that this is different from work in Liang (2005), which used the mutual information between adjacent types directly as features in the learning model. We avoid this method to keep our feature space small, in light of the paucity of training data.

For this technique, if we denote the original HMM distribution by $p(\mathbf{C} \mid \mathbf{W})$ with parameters θ, and a variational approximation $q(\mathbf{C})$ to the original distribution which respects our collocation based constraints i.e. $E_q(\phi(\mathbf{W}, \mathbf{C})) = -1$ where, $\phi(w_i, C(w_i), C(w_{i-1})) = -1$ if $C(w_i) = C(w_{i-1})$ and 0 otherwise, for $w_i s$ that are members of collocations; then the objective that we optimize becomes:

$$\arg\min_{\theta} \text{KL}(q \mid\mid p) \text{subject to } E_q(\phi(\mathbf{W}, \mathbf{C})) = -1 + \epsilon$$

When this objective is solved using its dual, the variational approximation q (after optimization for the dual variable λ) looks like:

$$q^*(\mathbf{C}) = \frac{p_\theta(\mathbf{C}|\mathbf{W}) exp(-\lambda^*.\phi(\mathbf{W}, \mathbf{C}))}{Z(\lambda^*)}$$

Since the constraints ϕ are local at the level of transition probabilities, the PR solution can be easily incorporated into the dynamic program of HMM.

3.5 Experiments

We present our results on the task of named entity recognition for Sorani and Tajik. For Sorani, the training (2175 instances) and test (212 instances) data was obtained from the annotated NER data in the LCTL language pack. For Tajik, we had access to a native speaker who, in about four hours, annotated 600 examples, which we split into 250 training instances, 250 test instances, and 100 development instances.

Features	**Rec.**	**Prec.**	**F1**
Std.	0.412	0.694	0.517
Std.+Br	0.476	0.702	0.567
Std.+Br+Gaz	0.490	0.750	0.593
Std.+Br+Gaz+Mph	0.509	**0.751**	**0.606**
Std.+PR+Gaz+Mph	**0.513**	0.741	**0.606**

Table 1: Results on Sorani NER

Features	**Rec.**	**Prec.**	**F1**
Std	0.302	0.709	0.423
Std+Br	0.511	0.657	0.574
Std+Br+Gaz	0.512	0.656	0.575
Std+Br+Gaz+Mph	0.517	**0.668**	**0.583**
Std+PR+Gaz+Mph	**0.537**	0.637	**0.583**

Table 2: Results on Tajik NER

In Tables 1 and 2, 'Std' refers to the standard features used in supervised NER systems (§3), 'Br' to the class features obtained from the Brown algorithm (§3.3), 'PR' to the class features from the EM and posterior regularization algorithm (§3.4), 'Gaz' to Gazetteer based features (§3.1), and 'Mph' to features from morphology induction (§3.2).

As we can observe, systems based only on standard features (§3) perform comparatively poorly, while adding Brown clusters lead to a large gain in recall especially in the Tajik condition. In both languages, the Brown and PR conditions perform similarly on F1, with the Brown condition having higher precision and the PR condition having higher recall. The reduction of precision in the PR condition is most likely a result of expanding the feature space with both Brown clusters and EM-based clusters.

The "gazetteer" for Sorani, which attempts to fabricate capitalization values for Sorani by comparison with Kurmanji words (§3.1), led to improvements in both recall and precision, while the Tajik gazetteer, collected from Tajik Wikipedia titles, did not lead to significant gains. Manual inspection of the resulting

	Sorani			Tajik		
Features	**Rec.**	**Prec.**	**F1**	**Rec.**	**Prec.**	**F1**
PER	0.406	0.709	0.516	0.446	0.458	0.452
LOC	0.680	0.801	0.735	0.589	0.737	0.655
ORG	0.343	0.604	0.438	0.136	0.375	0.200

Table 3: Error analysis on NER

Features	**Rec.**	**Prec.**	**F1**
Std+PR+Gaz+Mph	0.409	0.669	0.508

Table 4: Results on Tajik NER, using 1350 linguist annotations in lieu of native-speaker annotations

gazetteer revealed it to be fairly noisy with respect to the forms of the names; while the entries appeared to be, for the most part, genuinely named entities, many of them appear to have been converted directly from the Persian Wikipedia. Since Persian script does not completely represent vowels, the Tajik authors in many cases were likely guessing at the vowels when they were unfamiliar with the named entity.

Morphological features (§3.2) provided small improvements for both languages. Interestingly, the morphological features affected both the languages differently; while the Sorani system relied more on the induced stems, the Tajik system relied more on the suffixes. This may reflect the complexity of Sorani morphology (Walther, 2011; Esmaili and Salavati, 2013), in which many apparent affixes are actually enclitic, and therefore might not provide category information about their hosts as reliably as true suffixes do.

In Table 3, we observe that our NER systems are best at identifying LOC and are slightly worse at identifying PER. However, they perform substantially worse on identifying ORG because their proper noun parts can be confused with both locations and persons, and they often involve common words (e.g. "association", "for", etc.) that in other contexts are not part of NEs. Note that unlike in English, capitalization is not as helpful in distinguishing multiword NEs in Sorani and Tajik. Sorani "capitalization" here is only a feature inferred on a word-by-word basis from Kurmanji text, as detailed in §3.1, and Tajik generally uses Russian-style capitalization conventions in which only the first word in a multiword NE needs to be capitalized, making ORG identification much more difficult than in languages that capitalize all or most words in an ORG.

For the Tajik condition, we also had linguists – without prior experience in Tajik but generally familiar with Western Iranian languages – annotate another 1350 instances in sixteen person-hours. This was made possible by the IPA conversion step mentioned in §2.4, since the linguists did not have native proficiency in reading Cyrillic text.

Using this larger set instead of the smaller native-speaker-annotated set, we achieved similar results, with lower recall but higher precision (Table 4). This is a promising result, as it suggests that a team of linguists, even those without prior familiarity with the language, can create useful training data in a short time even when native informants are unavailable.

4 Conclusion

Our work demonstrates that, by using tools, data resources, and human resources (like linguists and language consultants) in innovative ways, it is possible to overcome some of the obstacles to developing standard NLP tools like NER systems for low-resource languages.

We built Named Entity Recognizers for Sorani Kurdish and Tajik in a manner which, while requiring minimal human annotator effort, managed to successfully incorporate informal intuitions and linguistic knowledge about the task and the languages into the system and seeks to identify and exploit latent patterns in the monolingual data.

To this end, we developed unsupervised class induction systems that were influenced by noisy collocation lists, and morphology induction systems that could be biased by subjective human feedback.

Moreover, we also showed that mapping the orthographic representation of a language to a general phonological representation not only enables efficient human analysis and annotation, but also opens avenues for transferring linguistic information from related languages.

Acknowledgments

This work was supported by the U.S. Army Research Laboratory and the U.S. Army Research Office, under the LRRT extension to contract/grant number W911NF10-1-0533.

References

Chris Biemann, Gerhard Heyer, Uwe Quasthoff, and Matthias Richter. 2007. The Leipzig corpora collection: Monolingual corpora of standard size. *Proceedings of Corpus Linguistic 2007*.

Peter F. Brown, Peter V. de Souza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, 19(1):61–74.

Kyumars Sheykh Esmaili and Shahin Salavati. 2013. Sorani Kurdish versus Kurmanji Kurdish: An empirical comparison. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 300–305.

Kyumars Sheykh Esmaili, Shahin Salavati, Somayeh Yosefi, Donya Eliassi, Purya Aliabadi, Shownem Hakimi, and Asrin Mohammadi. 2013. Building a test collection for Sorani Kurdish. In *2013 ACS International Conference on Computer Systems and Applications (AICCSA)*, pages 1–7.

Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049.

Yuening Hu, Jordan Boyd-Graber, and Brianna Satinoff. 2011. Interactive topic modeling. In *Proc. ACL*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Percy Liang and Dan Klein. 2009. Online EM for unsupervised models. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*, pages 611–619. Association for Computational Linguistics.

Percy Liang. 2005. *Semi-supervised learning for natural language*. Ph.D. thesis, Massachusetts Institute of Technology.

Patrick Littell, David Mortensen, Kartik Goyal, Chris Dyer, and Lori Levin. 2016. Bridge-language capitalization inference in Western Iranian: Sorani, Kurmanji, Zazaki, and Tajik. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of LREC 2016, Tenth International Conference on Language Resources and Evaluation*, pages 3318–3324.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics.

David Mortensen, Patrick Littell, Akash Bharadwaj, Kartik Goyal, Chris Dyer, and Lori Levin. 2016. Panphon: A resource for mapping IPA segments to articulatory feature vectors. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*.

Douglas W. Oard. 2003. The surprise language exercises. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(3):79–84, September.

Lawrence Rabiner and Biing-Hwang Juang. 1986. An introduction to hidden Markov models. *ASSP Magazine, IEEE*, 3(1):4–16.

Burr Settles. 2012. *Active Learning*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics.

Heather Simpson, Christopher Cieri, Kazuaki Maeda, Kathryn Baker, and Boyan Onyshkevych. 2008. Human language technology resources for less commonly taught languages: Lessons learned toward creation of basic language resources. *Collaboration: Interoperability between people in the creation of language resources for less-resourced languages*, page 7.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2006. Hierarchical Dirichlet processes. *JASA*, 101(476):1566–1581.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, vol. 4*, pages 142–147. Association for Computational Linguistics.

Géraldine Walther. 2011. Fitting into morphological structure: Accounting for Sorani Kurdish endoclitics. In A. Ralli, G. Booij, S. Scalise, and A. Karasimos, editors, *Proceedings of the 8th Mediterranean Morphology Meeting*, pages 299–321.

Multilingual Supervision of Semantic Annotation

Peter Exner **Marcus Klang** **Pierre Nugues**

Lund University
Department of Computer Science
Lund, Sweden
{Peter.Exner, Marcus.Klang, Pierre.Nugues}@cs.lth.se

Abstract

In this paper, we investigate the annotation projection of semantic units in a practical setting. Previous approaches have focused on using parallel corpora for semantic transfer. We evaluate an alternative approach using loosely parallel corpora that does not require the corpora to be exact translations of each other. We developed a method that transfers semantic annotations from one language to another using sentences aligned by entities, and we extended it to include alignments by entity-like linguistic units. We conducted our experiments on a large scale using the English, Swedish, and French language editions of Wikipedia. Our results show that the annotation projection using entities in combination with loosely parallel corpora provides a viable approach to extending previous attempts. In addition, it allows the generation of proposition banks upon which semantic parsers can be trained.

1 Introduction

Data-driven approaches using natural language processing tackle increasingly complex tasks with ever growing scales and in more varied domains. Semantic role labeling is a type of shallow semantic parsing that is becoming an increasingly important component in information extraction (Christensen et al., 2010), question answering (Shen and Lapata, 2007), and text summarization (Khan et al., 2015).

The development of semantic resources such as FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2005) made the training of models for semantic role labelers using supervised techniques possible. However, as a consequence of the considerable manual efforts needed to build proposition banks, they exist only for a few languages. An alternative approach to using supervision is to transfer knowledge between resources, a form of distant or related supervision. Methods for directly projecting semantic labels from a resource-rich language to a resource-scarce one were introduced in Padó (2007).

In this paper, we describe a method for aligning and projecting semantic annotation in loosely parallel corpora by using entities and entity-like linguistic units. Our goal is to generate multilingual PropBanks for resource-scarce languages. We used multiple language editions of Wikipedia: An English edition annotated up to a semantic level using the PropBank semantic roles, and syntactically annotated editions of Swedish and French Wikipedias. By aligning Wikipedias by entities, we constructed loosely parallel corpora and we used them to generate PropBanks in Swedish and French. We provide an evaluation of the quality of the generated PropBanks, together with an evaluation on two external FrameNets.

2 Previous Work

As an alternative to using supervised efforts for relation extraction, distant supervision can be employed to transfer relational knowledge representations from one resource to another. Distant supervision for relation extraction was introduced by Craven and Kumlien (1999) in the context of biomedical information extraction. Mintz et al. (2009) describe a method of using an external knowledge base as an indirect way of annotating text. Hoffmann et al. (2010) introduced the usage of Wikipedia infoboxes in distantly supervised relation extraction.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1007–1017, Osaka, Japan, December 11-17 2016.

The concept of transferring linguistic annotation, in the context of part-of-speech tags, across parallel corpora was introduced in Yarowsky et al. (2001). Cross-lingual annotation projection of FrameNet semantics has been described by Padó and Lapata (2009) and Basili et al. (2009). In Van der Plas et al. (2011), the authors describe an automatic method of direct transfer of PropBank semantics requiring no manual effort. Akbik et al. (2015) describe an approach to generate multilingual PropBanks using filtered annotation projection and bootstrap learning in order to handle errors stemming from translation shifts in corpora.

Most previous approaches have used professionally translated parallel corpora, mainly EuroParl (Koehn, 2005) and United Nations Corpora (Rafalovitch and Dale, 2009), to transfer semantic annotation. However, creating these resources requires manual efforts; they are thus limited in size and in the number of languages they cover. In contrast to parallel corpora, loosely parallel corpora describe similar concepts and events, but are not necessarily the result of a focused effort to translate a large corpus.

In Exner et al. (2015), we introduced the concept of using entities as a method for aligning sentences and transferring semantic content in loosely parallel corpora. However, the presented approach has the following limitations: (1) it was evaluated on one language only and (2) the evaluation was performed on the generated PropBank itself.

The contributions of this paper are the following: (1) We extend Exner et al. (2015) by including pronouns and other linguistic units that in a local context exhibit the characteristics of entities. (2) We present and evaluate two methods for aligning sentences by using entities. (3) We demonstrate the effectiveness and generalizability of our approach by projecting semantic annotations to two languages, Swedish and French, and we evaluate it using two external proposition databases, the Swedish SweFN++ (Borin et al., 2010) and French ASFALDA (Candito et al., 2014; Djemaa et al., 2016) that are both semantically-annotated corpora using adaptations of FrameNet frames. (4) We release the source code used in the annotation projection and we provide the generated PropBanks in Swedish and French[1].

3 Method

The aim of the method is to generate PropBank-like resources by fully annotating sentences in target languages using semantic content, in whole or partially, from a source language. We start with loosely parallel corpora in two languages: a **source language** (SL) expressing the semantic content that we want to transfer to a **target language** (TL). We then disambiguate and uniquely identify the entities in all the sentences. By using the unique identifier of each entity, we gain the ability to align sentences from two different languages forming sentence pairs (s_{SL}, s_{TL}). We annotate the (s_{SL}, s_{TL}) pairs, s_{SL} to semantic and syntactic levels and s_{TL} to a syntactic level. From each (s_{SL}, s_{TL}) pair, we learn the alignments between predicates (p_{SL}) in s_{SL} and verbs (v_{TL}) in s_{TL}. Finally, using the aligned entities and the predicate-verb alignments in each (s_{SL}, s_{TL}) pair, we transfer the semantic annotation in the form of predicate-argument structures. Figure 1 shows an overview of this approach.

3.1 Using Loosely Parallel Corpora

A prerequisite to projecting semantic annotation between two sentences is that they share the same semantic structure. To this end, we assumed that entities have a constraining property on the sets of predicate-argument structures they can instantiate. By aligning loosely parallel corpora through entities, pairs of sentences in two different languages that we will extract, although they are not translations of each other, should overall express the same semantic content. Furthermore, we believe that by applying our method on a large scale, the most frequent alignments of entities will elicit valid alignments.

In this context, even partial semantic content from a source sentence, s_{SL}, may be useful for annotating a target sentence, s_{TL}. As an example, consider the following sentence pair:

s_{SL} It$_{\text{A0}}$ features$_{01}$ Kelsey Grammer$_{\text{A1}}$ in his ninth ...
and is the first time$_{\text{AM-TMP}}$ the Simpsons$_{\text{A0}}$ visit$_{01}$ Italy$_{\text{A1}}$
s_{TL} I avsnittet besöker familjen Simpsons Italien
In the episode visit the family Simpsons Italy

[1] http://semantica.cs.lth.se

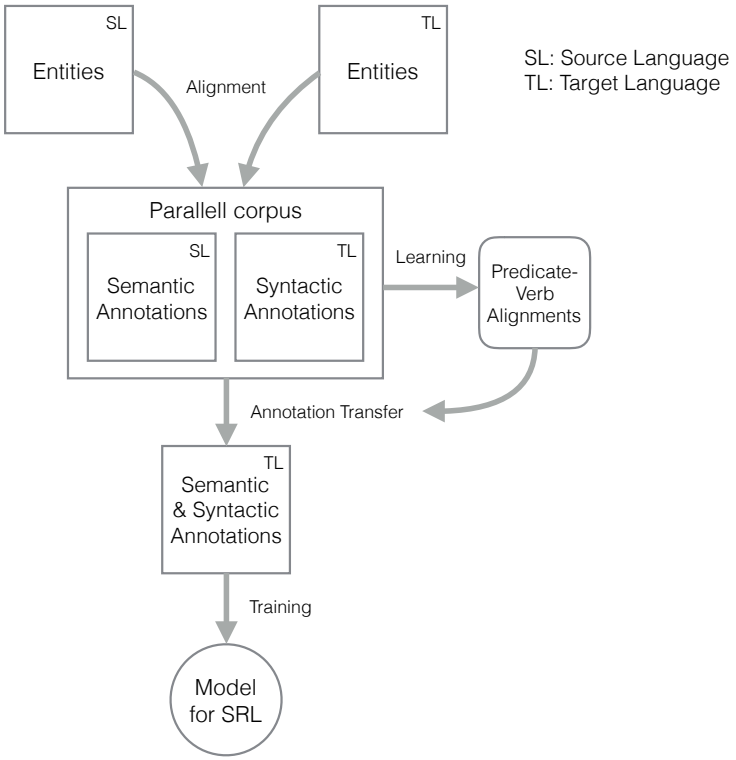


Figure 1: An overview of the approach for transferring semantic annotation from a **source language** (SL) to a **target language** (TL)

in which s_{SL} has been aligned with s_{TL} through the entities $(the\ Simpsons,\ Italy)$. s_{SL} expresses the two predicates $feature.01(it_{A0},\ Kelsey\ Grammer_{A1})$ and $visit.01(the\ first\ time_{AM-TMP},\ the\ Simpsons_{A0},\ Italy_{A1})$. Although s_{TL} is not an exact translation of s_{SL}, as it lacks the predicate $features.01$ and the temporal argument ($AM\text{-}TMP$) of $visit.01$, the partial transfer of the semantic content enables us to annotate s_{TL} with the predicate $besöka.01(familjen\ Simpsons_{A0},\ Italien_{A1})$.

3.2 Entity Disambiguation

Entity linking is the process of finding mentions, e.g. persons, cities, organizations, events, concepts, in text, and if available, assign them with a unique identifier provided by a knowledge base. We used Wikidata Q-numbers as identifiers as they provide globally unique identifiers between the different language editions of Wikipedia. As an example, consider the following entities:

Beijing, *Pékin*, and *Pequim*

as expressed in English, French, and Portuguese respectively. Although they have differing surface forms, they are all linked to the Q956 Wikidata number, as well as in 190 other languages. In total, Wikidata covers a set of more than 13 million items that defines the entity search space.

To carry out entity linking, we reimplemented a variant of TagME (Ferragina and Scaiella, 2010). The motivating factors behind our reimplementation were:

1. It enabled us to resolve mentions to identifiers in Wikidata, providing us with multilingual and coherent entity identifiers.

2. By using the same entity linker for multiple languages, we obtained a more consistent mention resolution across all the languages.

3. It eased the adaptation to new execution environments, in our case a cluster of computing nodes.

Our implementation of TagME requires minimal grammatical information as it only needs mention statistics derived from anchors and a dictionary of mention-entity pairs and of incoming links.

The entity linking algorithm consists of four steps: detection, candidate voting, selection, and resolution of overlapping mentions.

1. We find all the possible mentions consisting of tokens in sequences up to a maximum length of 6. The mentions found at this stage might be overlapping. We treat overlapped mentions independently and they contribute votes to all the other mentions. As an example, consider the following sentence:

   Prime Minister of Japan

   containing the two mentions: *Japan* and *Prime Minister of Japan*. In this case, the overlapped mention *Japan* will contribute a vote to the overlapping mention *Prime Minister of Japan*.

2. We compute the votes for each candidate belonging to a mention. To bound the computation time, we use voting groups consisting of a collection of mentions using a sliding window approach. The vote weight per candidate is the sum of all the inlink relatedness between all the candidates (Ferragina and Scaiella, 2010). In our case, we use all the candidates in a voting group.

3. We rank all the candidates per mention using the computed votes. We then prune the mention list using a coherence criterion and a threshold that we set empirically.

4. In the final step, we resolve the mention overlap using a greedy algorithm. The algorithm selects the overlapping mention, where the entity candidate has the largest global vote, removing all the locally overlapping mentions, until there is no overlap globally.

3.3 Syntactic and Semantic Annotation

In our experimental setup, we used the English edition of Wikipedia as our SL, and we annotated it with syntactic and semantic dependencies. For the syntactic-semantic parsing, we used an open-source semantic role labeler (Choi, 2012) trained on OntoNotes 5.0 (Weischedel et al., 2013).

We transferred the semantic annotation to two TLs, the Swedish and French editions of Wikipedia, both annotated with syntactic dependencies. For French syntactic parsing, we applied a transition-based dependency parser (Bohnet and Nivre, 2012; Bohnet and Kuhn, 2012) trained on a French Treebank described in Candito et al. (2010). Correspondingly, to preprocess the Swedish edition of Wikipedia, we applied a pipeline consisting of a POS tagger (Östling, 2013) and a syntactic dependency parser (Nivre et al., 2006).

3.4 Extension to Entity-like Tokens

Entities have the property of being uniquely identifiable across languages on a global scope. However, an obvious drawback to using entities as a means of aligning sentences and transferring roles, is that roles are not always instantiated by entities. To reclaim these instances, we extended the entity alignment to include entity-like **linguistic units** (LU). We focused on units that have the property of being uniquely identifiable and limited to the scope of a sentence pair. Units correspond to sequences of tokens the entity disambiguator has either failed to classify as an entity or otherwise lack the ability to be uniquely identified in a global context.

Our algorithm detects entity-like LUs as spans of tokens sharing the same surface form in both s_{SL} and s_{TL}. In addition, we set the constraint that they occur at most once in each sentence. As a consequence, this removes any misalignment issue since a LU in s_{SL} can be matched to only one LU in s_{TL}. This method enables us to include amounts, dates, and noun phrases that the entity disambiguator fails to detect.

Using similar constraints, we also include pronouns in the detection of entity-like LUs. However, rather than using the surface form of pronouns, which would unlikely match across languages, we instead categorize them by case, gender, and number. For English, Swedish, and French, third person singular pronouns have different surface forms based on gender. Therefore, in order to increase precision, we

limit the detection to only include third person pronouns. Although this constraint certainly limits the recall, this should not significantly impact the training procedure as the pronouns in the first and second persons are in very limited numbers in Wikipedia.

3.5 Aligning Sentences

The first challenge in transferring semantic annotation between loosely parallel corpora is to align sentences expressing the same semantic content. Our baseline method for aligning sentences extracts all the entities from a sentence and forms entity-sentence pairs, $(e_1...e_n, s)$. By aligning entities in different entity-sentence pairs, we form new triples containing a source sentence, a target sentence, and the subset of entities by which they are aligned $(s_{SL}, s_{TL}, e_1...e_s)$, where $k_{min} \leq s \leq k_{max}$ and k_{min}, k_{max} are prior parameters of our choice.

The baseline method is, in its simplicity, independent of any syntactic or lexical markup. It only requires the annotations from an entity disambiguator. However, one drawback lies in the inclusion of entities ungoverned by any predicate. As a consequence, the alignment of partial semantic content, as described in Sect. 3.1, becomes problematic. We therefore extended this baseline algorithm by using sets of entities projected by either arguments in s_{SL} or a verb in s_{TL}. Using this projection method, we then form entity-sentence pairs:

> $(e_1...e_p, s)$, where each entity in $(e_1...e_p)$ is governed by an argument belonging to a predicate in s_{SL}

and

> $(e_1...e_v, s)$, where each entity in $(e_1...e_v)$ is governed by a verb in s_{TL}.

The method for aligning entities in different entity-sentence pairs remains the same as for the baseline method. In Sect. 4.1, we investigate the effectiveness of the two methods under different settings.

3.6 Forming Predicate-Verb Alignments

Although we use entities as a mechanism to align sentences and transfer predicate-argument roles, predicates in s_{SL} and verbs in s_{TL} cannot be aligned by entities alone. In addition, some sentence pairs contain more than one predicate or verb, sharing the same subset of entities. This creates a combinatorial problem, where one predicate in s_{SL} could possibly be aligned to two or more verbs in s_{TL}, or vice versa. Furthermore, the application of a semantic parser to each s_{SL} annotates each predicate with a sense. This requires a method to induce new predicates and senses for the verbs in s_{TL}.

Most previous work relies on word alignments or uses bilingual dictionaries to transfer the predicate annotation between languages. However, when applied to new languages and domains, these approaches face a scaling problem requiring either training on parallel corpora or otherwise dictionaries which may not be available for every language.

Our approach builds on Exner et al. (2015) and automatically infers new predicate labels while scaling with the size of corpora and domains. A formal description of our alignment is:

1. We determine all the combinations of predicate-verb pairs, (p_i, v_k), extracted from all (s_{SL}, s_{TL}) pairs, where $p_i \in s_{SL}$ and $v_k \in s_{TL}$.

2. We assign $count(p_i, v_k)$ as the number of (p_i, v_k) in all (s_{SL}, s_{TL}), where $s_{SL} \in SL$ and $s_{TL} \in TL$.

3. For each $p_i \in SL$, we form alignments as $(p_i \rightarrow v_k) = max(count(p_i, v_1), ..., count(p_i, v_n))$.

4. For each $v_k \in (p_i \rightarrow v_k)$, we form a new TL predicate by using the lemma of v_k and an incremental counter based on the number of times v_k has appeared in an alignment.

We select the verb candidates for the alignment using lexical and syntactical rules to filter auxiliary verbs and other non-predicates.

3.7 Transferring Propositions

Given a pair of aligned sentences, (s_{SL}, s_{TL}), we transfer the semantic annotation from a predicate, $p_{SL} \in s_{SL}$, to a verb, $v_{TL} \in s_{TL}$, if $(p_{SL} \rightarrow v_{TL}) = max(count(p_i \rightarrow v_{TL})), \quad (p_i \rightarrow v_{TL}) \in (s_{SL}, s_{TL}), \quad \forall p_i \in s_{SL}$. If a s_{TL} is supervised by more than one s_{SL}, we select the s_{SL} having the larger subset of aligned entities with s_{TL}. We restrict the semantic transfer to predicate-argument structures containing at least one numbered argument and a temporal or location modifying argument, or at least two numbered arguments.

We transfer the argument roles by using the aligned entities between s_{SL} and s_{TL}. We assign the argument role to the governing token in the token span covered by each entity. However, if the argument token in s_{SL} is dominated by a preposition, we search for a preposition in s_{TL} governing the entity and assign it the argument role. We obtain the complete argument spans by taking the yield from the argument token.

4 Evaluation

In this section, we evaluate the approach described in Sect. 3 and we apply it to three language editions of Wikipedia in order to generate PropBanks for two languages: Swedish and French. The evaluation tries to answer the following questions:

1. How do different parameters and methods affect our approach?

2. What is the quality of the generated PropBanks and what level of performance can we expect in a practical setting?

3. Are there any differences between the languages, and if so what causes them?

4.1 Experimental Setup

For our experimentations, we chose the English, Swedish, and French editions of Wikipedia. These three Wikipedias are all among the top 6 in terms of article counts. As SL, we selected the English edition, and as TLs we select Swedish and French editions. We preprocessed all the articles to filter infoboxes, lists, diagrams, and to keep only text without any markup. Table 1 summarizes the statistics of the linguistic units in our chosen Wikipedias.

LANGUAGE	TOKENS	SENTENCES	ENTITIES	PREDICATES	ARGUMENTS
English	3825M	279M	439M	186M	450M
Swedish	481M	71M	58M	-	-
French	1269M	74M	181M	-	-

Table 1: Characteristics of Wikipedias used in the experimental setup

4.2 Predicate-Verb Alignment

We first evaluated how the predicate→verb alignment method described in Sect. 3.6 performs under different conditions and we examined how the number of entities, the method used, and the frequency affect the quality of the alignments. We grouped the English→Swedish alignments by their frequency into three bands: High, medium, and low. We then randomly sampled alignments from each band, in total 100 alignments and we used them to evaluate their precision. We defined precision as the number of English→Swedish alignments that we evaluate as correct divided by the total number of alignments in a sample. Figure 2 shows the precision and number of alignments using different number of entities and methods.

We observe that the precision increases with the number of entities used in the alignments. However, this increase is followed by a decrease in the number of alignments created. We also note that in all the

alignments, our projection method outperforms our baseline method for aligning sentences in terms of precision. Using three projected entities, we reach a precision of roughly 80% and 1,000 alignments.

We also investigated if the higher frequency of an alignment improved precision. Figure 3 shows the breakdown of precision curves into three frequency bands, formed using projected alignments. We observe that using three projected entities, alignments with high-medium frequencies show little to no error. This provides empirical evidence to our hypothesis in Sect. 3.1, that the most frequent alignments of entities will elicit valid alignments and that precision will scale with the amount of data used by the method.

The combination of aligning sentences with three projected entities gave us the optimal trade off between precision and number of alignments created. Therefore, in the rest of the evaluation, we use these settings.

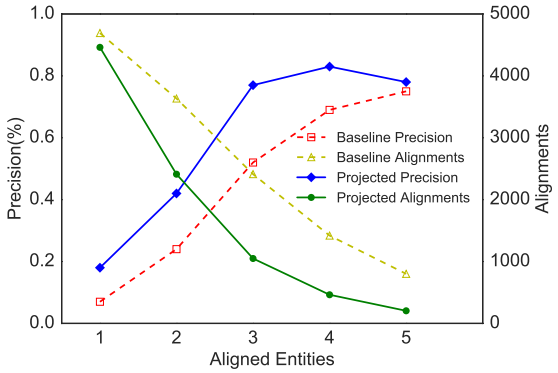


Figure 2: Graph of predicate→verb alignment precision and count under different parameter settings.

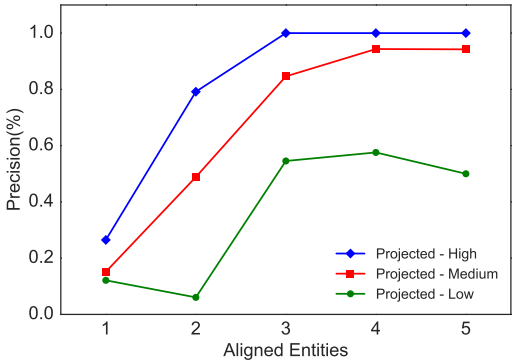


Figure 3: Graph of projected predicate→verb alignment precision, broken down by frequency band: high, medium, and low

4.3 Generated PropBanks

Using the annotation projection methods described in Sect. 3, we generated PropBanks in Swedish and French. We limited the PropBanks to only include fully annotated sentences and we removed the sentences exhibiting parsing errors, such as sentences having more than one syntactic root. We used these generated corpora to perform the error analysis in Sect. 4.5.

To evaluate our approach in a practical and automatic setting, we used samples of two linguistic resources: the Swedish FrameNet project (Borin et al., 2010) and the French FrameNet (Candito et al., 2014; Djemaa et al., 2016). We evaluated the generated Swedish and French corpora on a random sample of 100 sentences, from the Swedish FrameNet and the French FrameNet respectively. As PropBank and FrameNet have different annotation styles, we converted the sampled sentences from frame semantics to the semantics used in PropBank.

Table 2 shows the characteristics of the generated PropBanks and the FrameNets used in the evaluations.

DATASET	TOKENS	SENTENCES	PREDICATES	ARGUMENTS
Generated-Swedish	198,008	13,767	14,552	32,659
Generated-French	968,417	47,795	50,091	121,641
SweFN++ (TEST)	1,258	101	101	265
French FrameNet (TEST)	3,606	100	107	227

Table 2: Characteristics of the generated PropBanks used for training the SRL models and the FrameNets used for evaluating the trained models

4.4 Experimental Results

We evaluated the quality of the generated PropBanks in a practical setting as well as the effectiveness of using entity-like LUs in addition to entities. To assess the usefulness of the generated corpora, we first trained a semantic role labeler (Björkelund et al., 2010) on them. We split the generated corpora into 60:20:20 training, development, and testing sets, and we ran a selection process using a greedy forward selection and greedy backward elimination procedure to find the optimal set of features (Johansson and Nugues, 2008; Björkelund et al., 2009). We then used the trained models to automatically parse the test sets described in Sect. 4.3. Table 3 shows the evaluation of the semantic role labeler trained on the generated corpora.

The performance of the semantic role labeler, trained on the generated PropBanks, compares favorably with the automatic evaluations on parallel corpora described in Padó and Lapata (2009). For Swedish, using entity-like LUs, we observe an improvement of the labeled F1-measure by 10%. For French, we do not see the same dramatic increase, which we believe is caused by the large differences in pronoun classification and surface forms between English and French. We believe this discrepancy in improvement stems from projecting entity-like LUs across language groups: while English and Swedish belong to the Germanic branch, French belongs to the Romance group. Although more investigation is needed, these early results suggest that the annotation projection using entity-like LUs is most efficient when applied within a language group.

		LABELED			UNLABELED		
LANGUAGE	LINGUISTIC UNITS	P	R	F1	P	R	F1
Swedish	Entities (Baseline)	79.88	36.89	50.47	93.49	43.17	59.07
	Entities + Unique Tokens	84.82	44.26	58.17	92.67	48.36	63.55
	Entities + Unique Tokens + Pronouns	72.18	52.46	**60.76**	81.58	59.29	**68.67**
French	Entities (Baseline)	68.64	45.21	54.51	75.45	49.70	59.93
	Entities + Unique Tokens	64.03	48.50	55.20	70.36	53.29	**60.65**
	Entities + Unique Tokens + Pronouns	64.31	49.10	**55.69**	69.41	52.99	60.10

Table 3: Evaluation of semantic role labeling on the SweFN++ and French FrameNet corpora.

4.5 Error Analysis

To understand the quality of the generated PropBanks, we conducted an analysis of the predicate and argument errors. We randomly sampled 200 errors, of which 100 errors stemmed from the incorrect projection of argument labels and 100 were incorrect projections of predicates. Tables 4 and 5 show the type of errors for predicates and arguments respectively.

Using loosely parallel corpora, it is no surprise that the largest group of errors in predicate projection stems from sentences expressing differing semantic content. This error comes from sentence pairs, that although they contain the same subset of entities, express differing semantic content. However, as shown in Sect. 4.2, the precision of alignments increases with the number of alignments, leading us to believe that this category of error can be corrected using more data. The second largest error group is formed by different types of parsing errors occurring during the preprocessing stage. Encouragingly, only 6% of predicate projection errors stem from translation shifts, which is a further indication that entities exhibit a constraining property on the types of predicates that can instantiate them, even across languages.

Looking at argument projection errors, we again notice a group of errors stemming from misaligned sentences in loosely parallel corpora, *Differing Semantic Content* and *No Source Equivalent*. Looking beyond, alignment errors due to argument labels being assigned to the wrong token is the single most frequent error. The second largest category of errors is composed of expressions that can not be considered as entities, e.g. *In other words* and *During this time*. Finally, we observed a class of error stemming from entities undergoing a shift in specificity across sentences in two languages. These translation shifts included entities being referred to by their name in one language and by their entity type in the other

language, e.g. *London→the city*.

ERROR CLASS	NUMBER
Differing Semantic Content	66
Parsing Error: Target Syntax	8
Translation Shifts: Predicate Mismatch	6
Parsing Error: Target SRL	5
Parsing Error: Entity Disambiguation	5
Auxiliary Verb	4
Light Verb Constructions	4
No Source Equivalent	1
No Target Equivalent	1
TOTAL	100

Table 4: Error analysis of English→Swedish predicate→verb alignments.

ERROR CLASS	NUMBER
Alignment Error: Non Argument Head	16
Argument is not Entity-like	14
No Source Equivalent	14
Parsing Error: SRL	14
Differing Semantic Content	13
Translation Shift: Argument Entity	12
Parsing Error: Entity Disambiguation	9
Parsing Error: Target Syntax	4
Translation Shift: Argument function	3
Parsing Error: Source Syntax	1
TOTAL	100

Table 5: Error analysis of English→Swedish argument alignments.

5 Conclusion

In this paper, we have described the construction of multilingual PropBanks by aligning loosely parallel corpora using entities. We have trained a semantic role labeler on the generated PropBanks and that we evaluated in a practical setting on frame-annotated corpora. Our results compares favorably to annotation transfer using parallel corpora. In addition, we have extended the entity alignment to include alignment by entity-like linguistic units such as pronouns and dates.

We believe the growing source of loosely parallel corpora and their alignment using entities offers an alternative way to creating multilingual hand-annotated corpora. By performing a semantic projection on loosely parallel corpora, in our case multiple language editions of Wikipedia, we have presented an alternative approach to using parallel corpora. We believe our approach can be extended beyond encyclopedias to similar resources, such as news articles in multiple languages describing the same events.

One future improvement could be to leverage ontologies that categorize entities into types. We believe that such ontologies would prove useful in adjusting the specificity of entities in order to handle some translation shifts across languages. In addition, our current method of forming predicate→verb alignments could be extended by including information about the entity type.

While projecting pronouns from English to Swedish showed an improvement, we did not observe the same improvement when projecting from English to French. Therefore, an additional avenue of investigation could compare the performance of annotation projection within versus across language groups. In addition, a coreference solver could provide an alternative means of resolving pronominal mentions to entities.

Acknowledgements

This research was supported by Vetenskapsrådet under grant 621-2010-4800, and the *Det digitaliserade samhället* and eSSENCE programs.

References

Alan Akbik, Laura Chiticariu, Marina Danilevsky, Yunyao Li, Shivakumar Vaithyanathan, and Huaiyu Zhu. 2015. Generating high quality proposition banks for multilingual semantic role labeling. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 397–407.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Roberto Basili, Diego De Cao, Danilo Croce, Bonaventura Coppola, and Alessandro Moschitti. 2009. Cross-language frame semantics transfer in bilingual corpora. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 332–345. Springer.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 43–48. Association for Computational Linguistics.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 33–36. Association for Computational Linguistics.

Bernd Bohnet and Jonas Kuhn. 2012. The best of both worlds: a graph-based completion model for transition-based parsers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 77–87. Association for Computational Linguistics.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465. Association for Computational Linguistics.

Lars Borin, Dana Dannélls, Markus Forsberg, Maria Toporowska Gronostaj, and Dimitrios Kokkinakis. 2010. The past meets the present in swedish framenet+. In *14th EURALEX international congress*.

Marie Candito, Benoît Crabbé, and Pascal Denis. 2010. Statistical french dependency parsing: treebank conversion and first results. In *Seventh International Conference on Language Resources and Evaluation-LREC 2010*, pages 1840–1847. European Language Resources Association (ELRA).

Marie Candito, Pascal Amsili, Lucie Barque, Farah Benamara, Gal De Chalendar, Marianne Djemaa, Pauline Haas, Richard Huyghe, Yvette Yannick Mathieu, Philippe Muller, Benot Sagot, and Laure Vieu. 2014. Developing a french framenet: Methodology and first results. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may. European Language Resources Association (ELRA).

Jinho D. Choi. 2012. *Optimization of Natural Language Processing Components for Robustness and Scalability*. Ph.D. thesis, University of Colorado Boulder.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2010. Semantic role labeling for open information extraction. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, FAM-LbR '10, pages 52–60.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB'99)*, pages 77–86.

Marianne Djemaa, Marie Candito, Philippe Muller, and Laure Vieu. 2016. Corpus annotation within the french framenet: a domain-by-domain methodology. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, may.

Peter Exner, Marcus Klang, and Pierre Nugues. 2015. A distant supervision approach to semantic role labeling. In *Fourth Joint Conference on Lexical and Computational Semantics (* SEM 2015)*.

Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM.

Raphael Hoffmann, Congle Zhang, and Daniel S Weld. 2010. Learning 5000 relational extractors. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 286–295.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic-semantic analysis with propbank and nombank. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 183–187. Association for Computational Linguistics.

Atif Khan, Naomie Salim, and Yogan Jaya Kumar. 2015. A framework for multi-document abstractive summarization based on semantic role labelling. *Applied Soft Computing*, 30:737–747.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, pages 2216–2219.

Robert Östling. 2013. Stagger: An open-source part of speech tagger for swedish. *Northern European Journal of Language Technology (NEJLT)*, 3:1–18.

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36(1):307–340.

Sebastian Padó. 2007. *Cross-lingual annotation projection models for role-semantic information*. Ph.D. thesis, Saarland University.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Alexandre Rafalovitch and Robert Dale. 2009. United nations general assembly resolutions: A Six-Language parallel corpus. In *Proceedings of the MT Summit XII*, pages 292–299. International Association of Machine Translation, August.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 12–21, Prague, June.

Lonneke Van der Plas, Paola Merlo, and James Henderson. 2011. Scaling up automatic cross-lingual semantic role annotation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 299–304. Association for Computational Linguistics.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*, pages 1–8. Association for Computational Linguistics.

Siamese Convolutional Networks for Cognate Identification

Taraka Rama
Department of Linguistics
University of Tübingen, Germany
taraka-rama.kasicheyanula@uni-tuebingen.de

Abstract

In this paper, we present phoneme level Siamese convolutional networks for the task of pair-wise cognate identification. We represent a word as a two-dimensional matrix and employ a siamese convolutional network for learning deep representations. We present siamese architectures that jointly learn phoneme level feature representations and language relatedness from raw words for cognate identification. Compared to previous works, we train and test on larger and realistic datasets; and, show that siamese architectures consistently perform better than traditional linear classifier approach.

1 Introduction

Cognates are words that are known to have descended from a common ancestral language. In historical linguistics, identification of cognates is an important step for positing relationships between languages. An example of cognates are German *Fuß* and English *foot* whereas, Hindi *chakra* and English *wheel* are cognates that can be traced back to the Proto-Indo-European $*k^wek^wlo-$ and do not exhibit similarity on surface.

In NLP, automatic identification of cognates is associated with the task of determining if two words are descended from a common ancestor or not. In NLP, word similarity measures based on number of shared bi-grams, minimum-edit-distance, and length of longest common subsequence are supplied as features for a linear classifier or a sequence labeler on a set of labeled positive and negative examples; and then employ the trained classifier to classify new word pairs. The features for a classifier consist of string similarity scores (Hauer and Kondrak, 2011; Inkpen et al., 2005).

It has to be noted that the Indo-European dating studies (Bouckaert et al., 2012; Chang et al., 2015; Rama, 2016) employ human expert cognacy judgments for inferring phylogeny and internal dates of a well-studied language family. Therefore, there is a need for developing automated cognate identification methods that can be applied to those families of the world that are not as well-studied as Indo-European language family.

The supervised approaches (Kondrak, 2009; Bergsma and Kondrak, 2007) employ orthographic similarities and character alignments as features for training classifiers. In this work, we show how convolutional networks can be employed to extract phonetic features for the purpose of cognate identification. We also include a neural network approach to integrate language features for jointly training the neural networks. To the best of our knowledge, this work is the first to apply convolutional networks (CNN) for the purpose of cognate identification.

The work is organized as follows. In section 2, we define the task of cognate identification. In section 3, we motivate and describe convolutional network architectures for cognate identification. In section 4, we describe the related work for cognate identification. We present the experimental setup in section 5 and results in section 6. Finally, we present our conclusions in section 7.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1018–1027, Osaka, Japan, December 11-17 2016.

2 Cognate detection

In this paper, we work with Swadesh lists (Swadesh, 1952) that are composed of meanings which are supposed to be resistant to lexical replacement and borrowing.

Meaning	Swedish	English	German
foot	fut (B)	fut (B)	fus (B)
belly	mag3 (N)	bEli (B)	baux (B)
to sew	si (F)	s3u (F)	nE3n (B)

Table 1: A excerpt of Swadesh list from Indo-European Lexical database for Swedish, German, and English for three meanings "foot", "bell", and "to sew". The lexical items are transcribed in ASJP alphabet which is given in table 2. The cognate class labels, indicated in parentheses, do not carry additional information across meanings.

Table 1 shows the cognate class of each lexical item. Within a meaning, if two lexical items belong to a same cognate class, then they are cognates otherwise, they are treated as non-cognates. For example, all word pairs in meaning "foot" belong to the same cognate class "B" and are cognates whereas, the word pairs for English and German are cognate in meaning class "belly" and are not cognate in the meaning class "to sew". The task at hand is to correctly identify if two words from different languages belonging to a meaning class is cognate or not.

3 Convolutional Networks

In this section, we briefly describe some past work that uses CNNs for NLP tasks such as text classification and part-of-speech tagging. Then, we motivate the use of CNNs for cognate identification task.

The supervised approaches to cognate identification supply string similarity or phonetic similarity scores as features which might not capture all the information in two words. Character alignments extracted from minimum-edit-distance are used to train a linear classifier; and, the alignment features are further augmented by the context to capture processes of sound correspondences between two words (Bergsma and Kondrak, 2007; Ciobanu and Dinu, 2014). In a recent paper, Ciobanu and Dinu (2014) use character alignments from word pairs (extracted from a etymological dictionary) as features to train and test SVM classifiers. This method seems to require thousands of word pairs; and, might not be practically feasible in a low-data scenario. The approach of Bergsma and Kondrak (2007) which learns the alignment weights of characters requires monolingual corpora for source and target languages which is not available for many of the world's languages.

In this context, CNNs can be an alternative way to avoid explicit feature engineering through similarity computation and can extract relevant features from a raw word pair. Also, CNNs do not require explicit character alignment since the weights for non-monotonic shared features between two words can be learned through back-propagation.

3.1 CNNs in NLP

Collobert et al. (2011) proposed ConvNets for NLP tasks in 2011 and have been applied for sentence classification (Kim, 2014; Johnson and Zhang, 2015; Kalchbrenner et al., 2014; Zhang et al., 2015), part-of-speech tagging (dos Santos and Gatti, 2014), and information retrieval (Shen et al., 2014).

Santos and Zadrozny (2014) use character embeddings in conjunction with word embeddings to train a convolutional architecture for the classification of short texts. The authors find that their architecture performs better than the systems reported in Socher et al. (2013). In a recent work, Zhang et al. (2015) treat documents as a sequence of characters and transform each document into a sequence of one-hot character vectors. The authors designed and trained two nine layer convolutional networks for the purpose of text classification. The authors report competitive or state-of-the art performance on a wide range of text classification datasets.

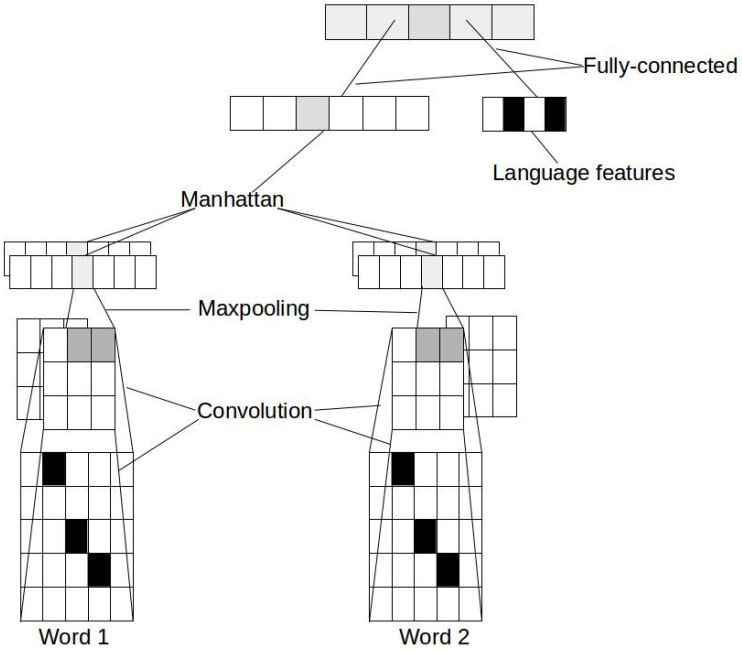


Figure 1: Illustration of Manhattan Siamese Convolutional network. We show the language features as a separate vector. Hot cells are shown in black whereas, real-valued cells are shown in grayscale.

3.2 Siamese Manhattan CNNs

Formally, we define the supervised problem setting where each training example x_i consists of two words x_{ia}, x_{ib} and a label $y_i \in \{0, 1\}$. Each phoneme $x_{iap} \in \mathbb{R}^k$ is a k-dimensional vector. A word is zero-padded or clipped at a pre-determined length n when necessary. A word x_{ia} of length n is represented as:

$$x_{ia} = x_{ia1} \oplus x_{ia2} \oplus \ldots \oplus x_{ian} \tag{1}$$

where, $\oplus$ is a concatenation operator. A convolution operation has a filter $W \in \mathbb{R}^{hm}$ where $h \leq k$ and $m < n$. The window size m defines the size of the filter. The feature map $C \in \mathbb{R}^{pq}$ where, $p = k - h + 1, q = n - m + 1$ is formed by convoluting the filter W with word x_{ia}. A max-pooling operation takes as input $C \in \mathbb{R}^{pq}$ feature map and applies the $max(C_{s \times t})$ to generate a feature $\hat{C} \in \mathbb{R}^{\lfloor p/s \rfloor \lfloor q/t \rfloor}$. The features generated by multiple filters are passed to a sigmoid function $\frac{1}{(1+exp(-x))}$ that computes the probabilities for y_i.

In the original siamese architecture proposed by Chopra et al. (2005), the weights are tied for each input x_{ia}, x_{ib}. The ℓ_2-norm (D) between the representations R_{ia}, R_{ib} computed using the shared convolutional networks of x_{ia}, x_{ib} and the label y_i is used to train a contrastive loss function $y_i \cdot D + (1 - y_i) \cdot max\{0, m - D\}$ where, m is a constant that can be tuned during training.

In this paper, we extend the siamese architecture to include an element-wise absolute difference layer which can then be stacked with multiple fully-connected layers. The final layer would be a sigmoid layer for binary classes. The idea behind this step is to push the CNNs to learn the phonological differences during training. The absolute difference ($-$) operation resembles ℓ_1 norm and is defined as

$$M_{iab} = |R_{ia} - R_{ib}| \tag{2}$$

where, $M_{iab} \in \mathbb{R}^r$ and r is the length of the representation vector at the end of convolutional layer. Hence, we call this architecture as Manhattan CNN. Parts of this architecture is shown in figure 1.

3.3 Phoneme encodings

Santos and Zadrozny (2014) train character embeddings for boosting their short text classification system based on CNNs. However, the cognate identification task typically deals with short word lists ($\sim$ 200) and short words ($\sim$ 5). However, many of the languages such as those studied in this paper do not have enough corpora to train character embeddings. Due to these reasons, we use 1-hot and hand-crafted phoneme encodings to train our convolutional networks.

1-hot phoneme CNN In this representation, each phoneme p is represented as 1-hot vector $\in \mathbb{R}^{|P|}$ where, P is the set of phonemes in a language family. Each word is either zero-padded to attain a length of n or clipped if the length exceeds a fixed length. We use the phonetic alphabet developed by Brown et al. (2008)[1] – for computerized historical linguistics – in our experiments. The ASJP alphabet and its phonetic properties are given in table 2. Word delimiters are represented by $\mathbf{0}$ vectors. We refer this architecture as CharCNN.

Phonetic features CNN In this representation, we encode each phoneme p as a $1/0$ vector of phonetic features. The description of phonetic properties of each phoneme is given in table 2. The features are ordered as they appear in the description of the alphabet in Brown et al. (2008). The first motivation behind this approach is to test if we can use the phonetic information (that is available with the word lists) for cognate identification. The second motivation is to test if CNNs can directly learn the patterns of sound change from underlying phonetic representations for the purpose of cognate identification. We refer this architecture as PhoneticCNN.

Features	p	b	f	v	m	8	4	t	d	s	z	c	n	S	Z	C	j	T	5	k	g	x	N	q	G	X	7	h	l	L	w	y	r	!	V
Voiced	0	1	0	1	1	1	1	0	1	0	1	1	1	0	1	0	1	1	0	0	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1
Labial	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Dental	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Alveolar	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Palatal/Post-alveolar	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Velar	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0
Uvular	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
Glottal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
Stop	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0
Fricative	1	1	1	1	0	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0
Affricate	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nasal	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Click	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Approximant	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0
Lateral	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
Rhotic	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Table 2: The ASJP alphabet is given in columns $2-35$ and the phonetic value of each symbol in the ASJP alphabet. Each phoneme is a multi-hot vector of fixed dimension 16.

3.4 Language features

One major limitation of previous work in cognate identification is that the weight training of word similarity features is not performed jointly with language relatedness information. We present an architecture to learn the phonological similarity jointly with weighted language relatedness. We extend the Manhattan architecture to include language relatedness information during training.

Some languages share more cognate pairs than other language pairs due to genetic relatedness. We can train the model to learn language relatedness jointly with phonological relatedness by representing the languages as 2-hot vector. Formally, two words x_{ia}, x_{ib} belong to different languages $l_a, l_b \in$ language set L is represented as 2-hot vector $\in \mathbb{R}^{|L|}$ which is concatenated with the learned representation M_{iab}. The concatenated vector is then passed to a fully-connected layer whose output is then passed to a sigmoid layer. All our models are trained with binary cross-entropy loss function defined as $-(y_i \cdot log(s_i) + (1-y_i) \cdot log(s_i))$ where s_i is the score for an instance i at the final sigmoid layer. The architecture with language features and the fully connected layer is shown in figure 1.

[1]Known as Automated Similarity Judgment Program; `asjp.clld.org`. The website provides 40 length word lists for more than 4000 of the world's languages and lists of length 100 for some languages. Very few word lists have cognate judgments such as Mayan language family which we include in this work.

We observe that including language relatedness (phylogeny) information seems to be quite challenging. For instance, the work of Bouchard-Côté et al. (2013) uses the inferred phylogeny of Austronesian languages (Greenhill and Gray, 2009) and do not infer the phylogeny themselves. In the case of Indo-European, Bouckaert et al. (2012) infer a Indo-European phylogeny from the cognacy information encoded for 200-word Swadesh lists and do not infer the cognacy judgments jointly with phylogeny.[2]

Using the Indo-European phylogeny information given in Glottolog (Nordhoff and Hammarström, 2011) can be circular since the cognacy judgments used by Bouckaert et al. (2012) are also used by human experts to derive the phylogeny information given in Glottolog. Therefore, we include the language information that is available with the word lists and hypothesize that a fully connected neural network layer can learn the weights of the language features jointly with the phonological representations generated by siamese CNNs through back-propagation.

4 Related work

The past work on cognate identification is mostly based on supervised approaches such as (Hauer and Kondrak, 2011; Bergsma and Kondrak, 2007; Inkpen et al., 2005) and graphical model approaches (Bouchard-Côté et al., 2013). In a different line of work, Kondrak (2000) and List (2012) employ linguistically motivated phoneme correspondence weights for computing the similarity between word pairs.

Inkpen et al. (2005) test the efficacy of different machine learning algorithms to determine if a pair of words are cognates or not. They use various orthographic similarity measures as features for the machine learning algorithms. They train and test their models on word pairs extracted from parallel texts and English-French cognate list; and find that there is no single machine learning algorithm that is good at both the datasets.

Hauer and Kondrak (2011) motivate a SVM classifier for the purpose of clustering word pairs within a meaning. They supply string similarity measures as features for their SVM classifier and then use the trained model to score the extracted word pairs from the testing part of their data. In this paper, we compare our neural network models against their classifier.

Ciobanu and Dinu (2014) test if character alignments extracted from Longest Common Subsequence alignments can be employed for the purpose of pair-wise cognate detection. They train a binary SVM classifier using the multi-gram character alignments as features for four pairs of Romance languages: Romanian-French, Romanian-Italian, Romanian-Spanish, and Romanian-Portuguese. They find that the SVM classifier trained on character alignments performs better than the orthographic similarity measures such as Edit distance, Longest Common Subsequence Ratio, and number of common bigrams.

Bouchard-Côté et al. (2013) employ a graphical model to reconstruct the word forms in Proto-Austronesian using Swadesh lists. They find that the inferred proto-forms largely agree with the reconstructed proto-forms. However, their method requires cognate information and the phylogeny of the language family to be known beforehand. In this article, we also experiment with a subset of Austronesian language family.

5 Experiments

5.1 Hyperparameters and training

The number of feature maps in a convolutional layer is fixed at 10. The architecture features a max-pooling layer that halves the output of the previous convolutional layer. We used the dropout technique with 0.5 probability (Srivastava et al., 2014) to prevent a fully-connected layer from over-fitting. A fully connected layer is trained with ReLU non-linearity ($max(0, x)$). The filter width m is fixed at 2 for 1-hot phoneme CNNs and 3 for phonetic feature CNNs. The filter length h is fixed as the size of $|P|$ for 1-hot phoneme CNNs and 2 for phonetic feature CNNs. The word length parameter n is fixed at 10. We used adadelta optimizer (Zeiler, 2012) with learning rate of 1.0, $\rho = 0.95$, and $\epsilon = 10^{-6}$. We fixed the mini-batch size to 128 in all our experiments. Both our architectures are relatively shallow – only

[2]The Indo-European work also includes higher level subgrouping information as priors to infer the divergence ages along the root and internal nodes of the phylogeny.

three layers – as compared to the text classification architecture of Zhang et al. (2015). We trained all our networks using Keras (Chollet, 2015) and Tensorflow (Abadi et al., 2016).

5.2 Datasets

We evaluate the performance of phoneme CNNs on three different language families: Austronesian, Indo-European, and Mayan.

Austronesian The Austronesian Basic Vocabulary Database[3] has word lists for 210 concepts in 378 languages. The database also has a cognacy judgment for each word. However, the database is not in an uniform transcription. Hence, we semi-automatically processed the words and converted a subset of 100 languages into uniform ASJP alphabet. We extracted a total of $525,941$ word pairs from the processed data of which $167,676$ are cognates.

Indo-European The second dataset comes from the Indo-European Lexical database which was originally created by Dyen et al. (1992) and curated by Michael Dunn.[4] The database is transcribed in a mix of International Phonetic Alphabet (IPA) and Romanized IPA. The database has word lists for 207 concepts in 139 languages. We extracted word lists for only those languages which are in phonemic transcription in more than 80% of the concepts. This filtering step leaves us with a total of $326,758$ word pairs for 52 languages of which $83,403$ are cognates.

Mayan The third dataset comes from the Mayan language family (Wichmann and Holman, 2013) that is spoken in Meso-America. This dataset has word lists in ASJP format for 100 concepts in 30 languages. We extracted $63,028$ word pairs from the dataset out of which $22,756$ are cognates.

Family	Training		Testing		$\|P\|$	$\|L\|$	Avg. # Cognate Classes
	Non-Cognates	Cognates	Non-Cognates	Cognates			
Austronesian	$244,978$	$125,018$	$113,287$	$42,658$	35	100	22.095
Indo-European	$162,818$	$62,120$	$80,537$	$21,283$	38	52	12.21
Mayan	$17,740$	$10,482$	$8,047$	$4,297$	33	30	8.58

Table 3: The number of positive and negative examples in training and testing datasets is given for each family. The size of the alphabet ($|P|$), number of languages ($|L|$) and, the average number of cognate classes per concept for each family.

5.3 Evaluation metrics

The performance of the baseline and the different CNN models is evaluated using Accuracy (ACC) and F-score. Given W word pairs, Accuracy is defined as the number of word pairs that have been assigned the correct labels (both cognate and non-cognate) divided by W. The F-score is defined as the harmonic mean of the Precision (P) and Recall (R) ($\frac{2PR}{P+R}$).

5.4 Baseline

We compare the performance of CNNs against the SVM classifier system trained on the following features from Hauer and Kondrak (2011). We used a linear kernel and optimized the SVM hyperparameter (C) through ten-fold cross-validation and grid search on the training data.

- Edit distance.
- Common number of bigrams.
- Length of longest common prefix.
- Lengths of both the words.
- Absolute difference between lengths of words.

[3] `http://language.psy.auckland.ac.nz/austronesian/` (Greenhill et al., 2008). We accessed the database on 09-12-2015.

[4] `http://ielex.mpi.nl/`

6 Results

For each family, we train our models on word pairs extracted from $\sim 70\%$ of the meanings and test on the remaining meanings. The details of the training and testing datasets are given in table 3. The results of our experiments are given in table 4.

Systems	Indo-European		Austronesian		Mayan	
	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy
Baseline	80.1	78.92	77.1	76.54	81.3	80.96
PhoneticCNN	85.8	**86.6**	77.6	79.24	85.4	85.56
PhoneticCNN + Langs.	**86.1**	86.42	78.3	79.8	86.2	86.23
CharCNN	84.6	85.05	79.1	80.11	86.3	86.4
CharCNN + Langs.	85.7	86.03	**80.3**	**80.94**	**87.5**	**87.5**

Table 4: Accuracies and F-scores of different CNN models against the system of Hauer and Kondrak (2011). CNNs with language features are denoted with a suffix "+ Langs.".

All the CNN models perform better than the baseline across all the language families. The PhoneticCNNs perform better than the CharCNN only on the Indo-European language family. In the case of Austronesian language family, joint training of language features improve the performance over baseline. This is reasonable since the Austronesian language family is spread over a wide range of geographical area spreading from Madagascar to Hawaii. The joint training of language features also improves the accuracy and F-score for Mayan language family.

CharCNN performs the best on the Mayan language family. One reason for this could be that the Mayan language family is a geographically proximal family and does not exhibit great amount of phonological divergence. Moreover, the Mayan language family shows less number of average cognate classes per concept as compared to Austronesian or Indo-European (cf. table 3) which can interpreted as a measure of genetic closeness within a family. In the case of Indo-European, the phonetic CNNs trained jointly with language information perform the best.

6.1 Do CNNs work with small training sets?

Zhang et al. (2015) note that CNNs require large amount of data for training. We test this hypothesis by training our CNNs on a smaller subset of 20 concepts. The results of our experiments are given in table 5.

Systems	Indo-European		Austronesian		Mayan	
	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy
Baseline	81.8	81.05	**77.9**	**77.7**	80.5	80.02
PhoneticCNN	83	**84**	73.6	75.86	84.6	84.64
PhoneticCNN + Langs.	**83**	83.78	73.1	75.82	84.1	84.25
CharCNN	79.6	81.62	74.3	76.69	**85.6**	**85.55**
CharCNN + Langs.	80.9	82.61	76.0	77.84	81.2	81.36

Table 5: Accuracies and F-scores of different CNN models trained on 20 meanings in the training data.

In the case of Indo-European and Mayan, the CNNs perform better than the baseline whereas for Austronesian the CNNs do not outperform the baseline system. The results for Indo-European and Mayan (cf. table 5) are similar to that of the results reported in table 4. That is, the CharCNN system performs the best for Mayan language family, while the PhoneticCNN system performs the best for the Indo-European language family. Surprisingly, for the Austronesian family, the baseline system performs better at F-score than the top-performing system for this language family in table 4, namely the CharCNN (with language features); the Accuracy measure of the Baseline system is also higher, but the difference is not statistically significant. The reason for this could be that there is not enough information in the 20 meanings to learn phonological similarity for 100 languages.

The results for Mayan family suggests that the CharCNN can be used with small datasets for a closely related language family. We believe that this is an important result due to the abundance of small number of language families in the world.

To support our claim, we cite family size numbers from Glottolog[5] which show that there are about 50 language families of size between 10 and 100. Due to this reason, we claim that a cognate identification system that can perform well on geographically proximal, closely related languages is useful for identifying cognates, which, in turn, can be used for inferring phylogenies of under-studied language families.

7 Conclusion

In this article, we proposed siamese CNNs for cognate identification and compared it against a SVM classifier trained on orthographic similarities. Our results suggest that CharCNNs and PhoneticCNNs can be used for the purpose of cognate identification. Our results on Mayan language families suggest that CNNs can be applied for NLP tasks in closely related languages or varieties. The language features improve the performance of CNNs across all the language families.

The performance of CharCNNs suggest that deep learning can be applied for small datasets (language families). Many deep learning systems reported in the NLP literature require huge amount of training data. Here, we show that handcrafted embedding and 1-hot encodings can learn useful representations from raw words for capturing phonological similarities between a word pair.

In the future, we hope to apply CNNs for more language families of the world for the purpose of cognate identification and phylogenetic inference.

Acknowledgements

I thank the reviewers for the useful comments which helped improve the paper. This work has been supported by the ERC Advanded Grant 324246 EVOLAEMP, which is gratefully acknowledged. I thank Simon Greenhill for the permission to use the Austronesian data in the experiments. The data for the experiments was processed by Johann-Mattis List and Pavel Sofroniev. I thank Çağrı Çöltekin for the comments on the initial draft that helped improved the paper.

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Shane Bergsma and Grzegorz Kondrak. 2007. Alignment-based discriminative string similarity. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 656.

Alexandre Bouchard-Côté, David Hall, Thomas L. Griffiths, and Dan Klein. 2013. Automated reconstruction of ancient languages using probabilistic models of sound change. *Proceedings of the National Academy of Sciences*, 110(11):4224–4229.

Remco Bouckaert, Philippe Lemey, Michael Dunn, Simon J. Greenhill, Alexander V. Alekseyenko, Alexei J. Drummond, Russell D. Gray, Marc A. Suchard, and Quentin D. Atkinson. 2012. Mapping the origins and expansion of the Indo-European language family. *Science*, 337(6097):957–960.

Cecil H. Brown, Eric W. Holman, Søren Wichmann, and Viveka Velupillai. 2008. Automated classification of the world's languages: A description of the method and preliminary results. *Sprachtypologie und Universalienforschung*, 61(4):285–308.

Will Chang, Chundra Cathcart, David Hall, and Andrew Garrett. 2015. Ancestry-constrained phylogenetic analysis supports the Indo-European steppe hypothesis. *Language*, 91(1):194–244.

François Chollet. 2015. Keras. *GitHub repository: https://github. com/fchollet/keras*.

[5] `http://glottolog.org/glottolog/family`. Accessed on 15-07-2016.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE.

Alina Maria Ciobanu and Liviu P Dinu. 2014. Automatic detection of cognates using orthographic alignment. In *ACL (2)*, pages 99–105.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78.

Isidore Dyen, Joseph B. Kruskal, and Paul Black. 1992. An Indo-European classification: A lexicostatistical experiment. *Transactions of the American Philosophical Society*, 82(5):1–132.

Simon J. Greenhill and Russell D. Gray. 2009. Austronesian language phylogenies: Myths and misconceptions about Bayesian computational methods. *Austronesian Historical Linguistics and Culture History: A Festschrift for Robert Blust*, pages 375–397.

Simon J. Greenhill, Robert Blust, and Russell D. Gray. 2008. The Austronesian basic vocabulary database: from bioinformatics to lexomics. *Evolutionary Bioinformatics Online*, 4:271–283.

Bradley Hauer and Grzegorz Kondrak. 2011. Clustering semantically equivalent words into cognate sets in multilingual lists. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 865–873, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Diana Inkpen, Oana Frunza, and Grzegorz Kondrak. 2005. Automatic identification of cognates and false friends in French and English. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 251–257.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 103–112.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, June.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

Grzegorz Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 288–295.

Grzegorz Kondrak. 2009. Identification of cognates and recurrent sound correspondences in word lists. *Traitement Automatique des Langues et Langues Anciennes*, 50(2):201–235, October.

Johann-Mattis List. 2012. LexStat: Automatic detection of cognates in multilingual wordlists. In *Proceedings of the EACL 2012 Joint Workshop of LINGVIS & UNCLH*, pages 117–125, Avignon, France, April. Association for Computational Linguistics.

Sebastian Nordhoff and Harald Hammarström. 2011. Glottolog/Langdoc: Defining dialects, languages, and language families as collections of resources. In *Proceedings of the First International Workshop on Linked Science*, volume 783.

Taraka Rama. 2016. Ancestry sampling for indo-european phylogeny and dates.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Morris Swadesh. 1952. Lexico-statistic dating of prehistoric ethnic contacts: with special reference to North American Indians and Eskimos. *Proceedings of the American philosophical society*, 96(4):452–463.

Søren Wichmann and Eric W Holman. 2013. Languages with longer words have more lexical change. In *Approaches to Measuring Linguistic Differences*, pages 249–281. Mouton de Gruyter.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

Exploring Differential Topic Models for Comparative Summarization of Scientific Papers

Lei He[*#], Wei Li[*], Hai Zhuge[*#]
[#] System Analytics Research Institute, Aston University, Birmingham, UK
[*] Key Lab of Intelligent Information Processing, ICT, CAS, University of Chinese Academy of Sciences, Beijing, China
hel2@aston.ac.uk,weili.ict.kg@gmail.com,zhuge@ict.ac.cn

Abstract

This paper investigates differential topic models (*dTM*) for summarizing the differences among document groups. Starting from a simple probabilistic generative model, we propose *dTM-SAGE* that explicitly models the deviations on group-specific word distributions to indicate how words are used differentially across different document groups from a background word distribution. It is more effective to capture unique characteristics for comparing document groups. To generate *dTM*-based comparative summaries, we propose two sentence scoring methods for measuring the sentence discriminative capacity. Experimental results on scientific papers dataset show that our *dTM*-based comparative summarization methods significantly outperform the generic baselines and the state-of-the-art comparative summarization methods under ROUGE metrics.

1 Introduction

Today, the interconnected nature of real-world applications brings more cross-field research problems leading to a much closer relationship between research areas. Real-world challenges require researchers to quickly get acquainted with knowledge in other areas. For example, imagine a researcher who is familiar with topic models wants to extend her research to opinion summarization. She would be more interested in finding out the current development of sentiment analysis and how topic models can be used in sentiment analysis, rather than the common background knowledge such as topic models and basic NLP technologies. Such a real-world demand encourages the study of multi-document comparative summarization for scientific papers in multiple subject areas. This paper presents the initial study on this problem.

Comparative summarization aims at summarizing the differences among document groups (Wang et al., 2012). The core is to compare different topics and find unique characteristics for each document group. The main motivation of this paper is to apply *dTM* to comparative summarization and to model the group-specific topics to capture the unique word usage for characterising documents in the same group. To our best knowledge, there is no previous study providing in-depth model analysis and detailed experimental results on *dTM* applied for comparative summarization.

We first propose a probabilistic generative model *dTM-Dirichlet* to model the group-specific word distributions to capture the unique word usage for each document group. However, *dTM-Dirichlet* is not a truly differential topic model and it suffers from the problems of high inference cost, over-parameterization and lack of sparsity. Evolving from the idea of *SAGE* (Eisenstein et al., 2011), we develop *dTM-SAGE* to make the word probability distributions for each document group to share a common background word distribution and explicitly models how words are used differently in each group from the background word distribution.

Our main contributions include the following two points: (1) we propose *dTM* to capture unique characteristics of each document group in the application background of comparative summarization for cross-area scientific papers; and (2) we propose two sentence scoring methods to measure the sen-

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1028–1038, Osaka, Japan, December 11-17 2016.

tence discriminative capacity and a greedy sentence selection method to automatically generate summary for *dTM*-based comparative summarization.

2 Related Work

Multi-document Summarization. Existing multi-document summarization can be either extractive or abstractive (Sekine and Nobata, 2003). Our work focuses on the extractive techniques which involve in assigning saliency scores to sentences and extracting high-scored sentences in a greedy manner to construct a summary (Wan et al., 2007; Cai et al., 2010; Gupta and Lehal, 2010; Celikyilmaz and Hakkani-Tur, 2011). Graph-based ranking techniques such as TextRank (Mihalcea and Tarau, 2004) and LexPageRank (ErKan and Radev, 2004) have been widely used in extractive summarization. A bigram based supervised method was proposed for extractive summarization in ILP framework (Li et al., 2013; Li, 2015). Jha et al. (2015) proposed an extractive algorithm that combines a content model with a discourse model to generate coherent summaries for scientific articles. A multi-dimensional summarization methodology was proposed to transform the paradigm of traditional summarization research through multi-disciplinary fundamental exploration on semantics, dimension, knowledge, computing and cyber-physical society (Zhuge, 2016).

Comparative Summarization. Unlike the generic summarization that summarizes the common information in document collection, the comparative summarization aims to summarize the differences among document groups. Wang et al. (2012) proposed a discriminative sentence selection method to generate summary by selecting sentences in a greedy manner to minimize the generalized variance of a covariance matrix using a multivariate normal model. Shen and Li (2010) proposed a method by building the sentence graph for each document group and extracting a complementary minimum dominating set on each graph to form a discriminative summary.

Update Summarization. The most similar task to comparative summarization is update summarization, which aims to detect and summarize novel information in a document set B under the assumption that users have already learnt the documents in set A, where documents in A chronologically precede the documents in B. The update summarization has been well studied. Most existing methods solve it as a redundancy removal problem by adding functionality to remove redundant sentences using filtering rules (Fisher and Roark, 2008), Maximal Marginal Relevance (Boudin et al., 2008), or graph-based algorithms (Shen and Li, 2010; Li et al., 2008).

More related to this paper is the work of a topic-model based update summarization approach *DualSum* (Delort and Alfonseca, 2012), which learns a general background distribution across the corpus and a document-specific distribution for each document, but also learns two collection-specific distributions for each pair of update collection and base collection: the joint topic distribution and the update topic distribution. This paper revises *DualSum* as a baseline for evaluation in Section 5.2.

Topic Models for Documents Comparison. The other type of related work is the comparison of documents. Most existing studies for this goal focus on topic models to discover common and specific themes among document collections, referred to as cross-collection topic models (Paul, 2009). This idea was first explored with an initial topic model *PLSI* (Zhai et al., 2004), and later improved with *LDA* topic model (Blei, 2012; Pual, 2009) which inspires our *dTM-Dirichlet*. There are a number of real-world applications extending cross-collection topic models in different scenarios (Ahmed and Xing, 2010; Li et al., 2011). For example, Paul and Girju (2009) employed cross-collection *LDA* (*cc-LDA*) for cross-cultural analysis of blogs and forums and later they proposed a two-dimensional topic-aspect model (*TAM*) to jointly discover topics and aspects in scientific literature (Paul and Girju, 2010). The common idea behind these cross-collection topic models is that using latent topics capture the common and unique word usage among document collections. Cross-collection topic models neglect the correlations between each collection-specific topic and the common background topic, thus make it insufficient to capture differential word usage. More importantly, the correlations are the essence of the differential topic models.

3 Differential Topic Models

In this section, the differential topic models are explored for comparative summarization. We first develop a simple probabilistic generative model, *dTM-Dirichlet*. Evolved from *dTM-Dirichlet*, *dTM-SAGE* is developed by modelling the correlations as additive relation between the group-specific devi-

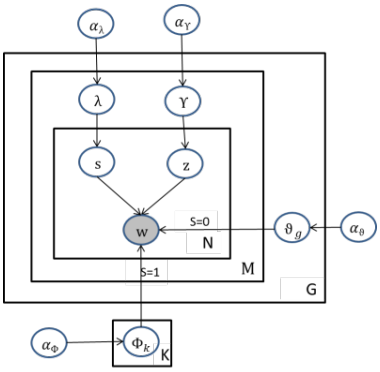


Fig.1 dTM-Dirichlet Model Graph Representation.

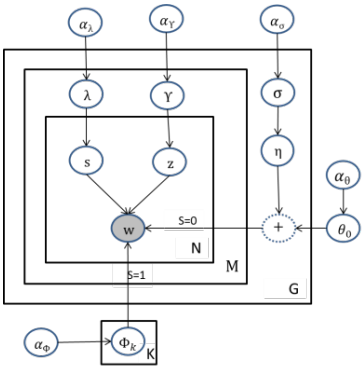


Fig.2 dTM-SAGE Model Graph Representation.

ations and a background word distribution, which enables to capture more salient group-specific words and bypass the problems of high inference cost, over-parameterization and lack of sparsity.

To illustrate *dTM*, we first define some notations to express a document corpus *C*. Let *G* be the number of groups in the corpus, M_g be the number of papers in group g and $N_{g,m}$ be the number of words in paper *m*. A word $w_{g,m,n}$ representing the n^{th} word in paper *m* of group g is a discrete observed variable, defined to be an item in the vocabulary list of the whole corpus.

3.1 *dTM-Dirichlet* Model

dTM-Dirichlet model is a simplified version of cross-collection LDA (*ccLDA*) (Paul and Girju, 2009) for comparing multiple text collections. *dTM-Dirichlet* builds two types of word model. One is for each document group g, in which there is a group-specific content word model ϑ_g that emits discriminative words for the group. The other type is a superset of group-independent word models φ_k that generates either background words shared by all document groups or salient words occurring in several documents of different groups. Reconsidering the example in section 1, the group-independent word model represents two classes of words, i.e. the background words like topic model that are shared by almost all papers; and the salient words like NP chunk and dependency parsing that only occur in several papers of different groups.

We focus on the group-specific word model for comparative summarization. Since background words and salient words provide no group-specific knowledge, they are not distinguished in *dTM-Dirichlet*. Following probabilistic topic models, we assume that word models φ_k and ϑ_g are multinomial distributions over words, drawn from uniform *Dirichlet* distribution (*Dir*) with priors α_φ and α_ϑ.

As shown in Fig. 1, dTM-Dirichlet associates each document a topic distribution $\gamma \sim Dir\ (\alpha_\gamma)$, and the topic assignment variable *z* for each word in the document thus can be multinomially sampled from γ. Besides a topic variable *z*, each word is also assigned with a binary variable *s* that indicates whether the word is a group-independent topic word (*s*=1) or a group-specific content word (*s*=0). Each document has a group-specific word controller $\lambda \sim Beta(\alpha_\lambda)$, which reflects the proportion of group-specific content in a document. s is sampled from a Bernoulli test with the probability of λ.

Formally, the generative process of dTM-Dirichlet model for a corpus *C* divided into *G* document groups is shown in Table 1. When $s_{g,m,n} = 1$, the sample of the group-independent topic word is identical to *LDA*. When $s_{g,m,n} = 0$, the sample of the word $w_{g,m,n}$ is independent from the document's topic distribution $\gamma_{g,m}$ and directly drawn from the group-specific content word distribution θ_g.

dTM-Dirichlet models group-specific word distributions to capture the differential lexicon usage of document groups. However, *dTM-Dirichlet* is not a truly differential topic model, which requires the development of *dTM-SAGE* for comparative summarization.

3.2 *dTM-SAGE* Model

When generating topics for multiple document collections, *LDA*-style generative models associate a multinomial distribution with each document group, which is the same as how we model the group-specific content words in *dTM-Dirichlet* model.

In contrast, *SAGE* (Sparse Additive Generative model) (Eisenstein et al., 2011) provides an alternative way to *LDA* by endowing each document group with a model of the deviation in log-frequency

The generative process of dTM-Dirichlet	The generative process of dTM-SAGE
1. For each topic k, where $1 \le k \le K$ a. Draw $\Phi_k \sim Dir(\alpha_\Phi)$ 2. For each document group *g*, where $1 \le g \le G$ a. Draw $\theta_g \sim Dir(\alpha_\theta)$ b. For each document *m* in group *g*, where $1 \le m \le M_g$ 1) Draw $\lambda_{g,m} \sim Beta(\alpha_\lambda)$ 2) Draw $\gamma_{g,m} \sim Dir(\alpha_\gamma)$ 3) For each word *n*, where $1 \le n \le N_{g,m}$ a) Draw $s_{g,m,n} \sim Bern(\lambda_{g,m})$ b) If $s_{g,m,n} = 1$ (a group-independent topic word) A. Draw a topic assignment $z_{g,m,n} \sim \gamma_{g,m}$ B. Draw a word $w_{g,m,n} \sim \Phi_{z_{g,m,n}}$ c) If $s_{g,m,n} = 0$ (a group-specific content word) A. Draw $w_{g,m,n} \sim \theta_g$	1. Draw $\theta_0 \sim Dir(\alpha_\theta)$ 2. For each topic *k*, where $1 \le k \le K$ a. Draw $\Phi_k \sim Dir(\alpha_\Phi)$ 3. For each document group *g*, where $1 \le g \le G$ a. For each term *v*, where $1 \le v \le V$ 1) Draw $\sigma_{g,v} \sim Exponential(\alpha_\sigma)$ 2) Draw $\eta_{g,v} \sim N(0, \sigma_{g,v})$ b. Set $\boldsymbol{\vartheta_g} \propto exp(\boldsymbol{\vartheta_0} + \boldsymbol{\eta_g})$ c. For each document *m* in group *g*, where $1 \le m \le M_g$ 1) Draw $\lambda_{g,m} \sim Dir(\alpha_\lambda)$ 2) Draw $\gamma_{g,m} \sim Dir(\alpha_\gamma)$ 3) For each word *n*, where $1 \le n \le N_{g,m}$ a) Draw $s_{g,m,n} \sim Bern(\lambda_{g,m})$ b) If $s_{g,m,n} = 1$, Draw $z_{g,m,n} \sim \gamma_{g,m}$, Draw $w_{g,m,n} \sim \Phi_{z_{g,m,n}}$ c) If $s_{g,m,n} = 0$, Draw $w_{g,m,n} \sim \theta_g$

Table 1: The generation process of *dTM-Dirichlet* and *dTM-SAGE*.

from a constant background distribution, which brings three advantages: First, a sparsity-inducing prior can be applied to limit the number of terms whose probability diverges from the background term frequencies. Second, multi-facets latent variables can be easily combined by adding each facet component together to reduce the inference cost. Third, it is redundant to learn unique probabilities for high-frequency background words of each group. Modelling the deviation of each group-specific word distribution cancels the relearn process for the background words.

We propose *dTM-SAGE* which explicitly models the deviation in log-frequency of each group-specific word distribution from a background lexical distribution. *dTM-SAGE* also builds word models for group-independent topic words and group-specific content words. The group-independent topic words consist of background topic words and salient topic words.

dTM-SAGE models two types of group-independent words separately: as shown in Fig. 2, the salient topic words captured by φ_k and the background topic words captured by ϑ_0. The word models φ_k and ϑ_0 are multinomial distributions drawn from uniform Dirichlet prior with parameter α_φ and α_ϑ. To enable ϑ_0 to capture real background topic words shared by all document groups, we replace the constant background distribution in *SAGE* with a latent distribution learnt by MAP estimation using a Newton optimization.

The major difference between *dTM-SAGE* and *dTM-Dirichlet* is how the group-specific topics are generated. In Fig.2, each document group g has a group component vector η_g representing the deviations in log-frequencies from the background distribution ϑ_0. The group-specific topic is represented by log frequency deviations rather than word probabilities. Given the background distribution ϑ_0 and the group component vector η_g, the group-specific topic distribution ϑ_g for each word in a document in the group g, denoted by $\boldsymbol{\vartheta_g} \propto exp(\boldsymbol{\vartheta_0} + \boldsymbol{\eta_g})$, is computed by Equation (1):

$$p(w \mid \theta_0, \eta_g) \;/\; \sum_v \exp(\theta_{0,v}, \eta_{g,v}) \tag{1}$$

In Equation (1), g indexes the group component vector and v indexes the term in the corpus vocabulary. Following *SAGE*, we ignore covariance between terms. For each term v, $\eta_{g,v}$ is drawn from a zero-mean Gaussian distribution $N(0, \sigma_{g,v})$, where the variance $\sigma_{g,v}$ is drawn from the Exponential distribution parameterized by α_σ. The compound model $\int N(\eta; 0, \sigma) Exponential(\sigma; \alpha_\sigma) d\sigma$ is equivalent to a zero-mean Laplace prior on η inducing sparsity and meanwhile permitting large degrees of deviations. In *dTM-SAGE*, we treat the variance σ as a latent variable and develop variational inference on it, which is the same as *SAGE*. The remaining part of *dTM-SAGE* is the same as *dTM-Dirichlet* model. Formally, the generative process of *dTM-SAGE* is shown in Table 1. See Appendix A for inference details of $\boldsymbol{\vartheta_0}$ and $\boldsymbol{\eta_g}$.

4 Comparative Summary Generation

To summarize differences among document groups, we rely on group-specific topics ϑ_g to select most discriminative sentences for summary generation. This section introduces the sentence scoring and the sentence selection techniques developed for *dTM*-based comparative summarization.

4.1 Sentence Scoring

Both *dTM-Dirichlet* and *dTM-SAGE* model the group-specific word distributions ϑ_g to capture the unique content in each document group. For *dTM-SAGE*, we can also get a corpus background topic distribution ϑ_0 that reflects the common themes shared by all groups. To measure the sentence discriminative capacity, we develop two sentence scoring methods: one is based on the word discriminative scores and the other is measured by the difference of the probabilities that a sentence is generated from a group-specific topic distribution and the background topic distribution.

First, given a set of group-specific word distributions ϑ_g ($1 \le g \le$ G), we define the *w*ord *d*iscriminative score $DSW(v, g)$ of a term v to a group g as $DSW(v,g) = \sum_{g' \neq g}(\vartheta_{g,v} - \vartheta_{g',v}) / \sqrt{\sum_g \vartheta_{g,v}^2 + \epsilon}$, where ϵ is a small number (set to 0.05) to avoid the error of division by zero. Larger value of the word discriminative score indicates more discriminative ability the word has. The intuition is that a word more likely to occur in a particular group and less likely to occur in other groups tends to be more discriminative. The *d*iscriminative *c*apacity of a sentence s to a group g $DCS\_dsw(s, g)$ is the average over the word discriminative scores:

$$DCS\_dsw(s, g) = \sum_{w \in s} DSW(w, g) \,/\, len(s) \tag{2}$$

The other method to measure the discriminative capacity of a sentence relies on the likelihood that the sentence is generated from a group-specific distribution and the background topic distribution. Its design is motivated by the idea that a word is more discriminative if it occurs more often in a group-specific topic and occurs rarely in the shared background topic. Given a topic-word distribution ϑ, the probability of a sentence *s* generated from ϑ:

$$\log P(s \mid \theta) = \sum_{w \in s} \log \theta_w \tag{3}$$

Given a set of group-specific word distributions ϑ_g ($1 \le g \le$ G) and the background topic distribution ϑ_0, the *d*iscriminative *c*apacity of a sentence s to a group g is defined as the *d*ifference of sentence *g*enerative *p*robabilities $DCS\_dgp(s, g)$:

$$DCS\_dgp(s, g) = u \log P(s \mid \theta_g) - (1 - u) \log P(s \mid \theta_0) \tag{4}$$

where u is a balance factor trading off between group-specific words and background words.

4.2 Sentence Selection

To select discriminative sentences to form group summary, we use different sentence selection methods according to sentence scoring techniques.

For the sentence scoring based on the word discriminative scores, we first rank the sentences according to the sentence discriminative capacity score DCS_dsw. Then we select a sentence with the highest score if it satisfies the redundancy constraint that indicated by a cosine similarity threshold (empirically set to 0.8).

For the scoring based on difference sentences generative probabilities, suppose we have a set of candidate sentences S to form a summary for group g and we want to select k sentences denoted as S_k. A greedy sentence selection schema is proposed to build S_k by iteratively choosing a j$^{\text{th}}$ sentence that currently has the maximum sentence discriminative capacity score DCS_dgp:

$$s_j^* = \arg\max_{s_j \in S \setminus S_{j-1}} DCS\_dgp(s, g) \tag{5}$$

In order to discourage redundancy, after select one sentence, we update the group-specific topic distribution ϑ_g by setting $\vartheta_{g,w} \propto \vartheta_{g,w}^2$ for each word w in the selected sentence s_j^*. Sentences are selected in this manner until reaching the summary limit.

5 Experiments and Results

5.1 Data Collection and Annotation

Comparative summarization is not a new task. However, to our best knowledge there is no public benchmark data set available. For collecting experiment data, we choose three tasks in NLP: summarization (*SUMMA*), sentiment analysis (*SA*) and geographical NLP tasks (*GEO*) to form three document groups. To make different groups share more salient themes, we focus on papers using probabilistic

Group	Keywords		|D|	|S|
	Title	Plain Text		
SUMMA	summarization	topic model	35	6636
SA	Sentiment	topic model	45	10239
GEO	N/A	topic model, geographical	49	8249

Table 2: General Information of the Dataset

Measures	LDA	SAGE	Variant of DualSum	dTM-Dirichlet	dTM-SAGE
Perplexity	2218.37	2177.29	*1564.04	2052.78	**1891.10**
C_A (Wiki)	0.098	0.143	0.130	0.138	**0.147**
C_V (Wiki)	0.321	0.334	0.344	**0.360**	0.355
C_UCI (Wiki)	-2.116	-1.917	-1.272	-1.495	**-0.905**
C_UCI (Intra)	-0.895	-0.849	-0.662	-0.661	**-0.608**

Table 3: Comparisons of Perplexity and Topic Coherences for Different Topic Models.

topic models. We collect 129 papers in total for the three groups from ACL Anthology Searchbench providing full-text search for 28,000 papers in the ACL Anthology. For each group, we search with two types of keyword filters: plain text filter and title filter. Table 2 shows the general information of each document group, including the keywords, the number of documents |D| and the number of sentences |S|. To pre-process the dataset, we exclude all tables, figures and formulas, remove stop words, perform stemming with Porter Stemmer, and prune words less than 5 times across the corpus. There are 3720 tokens after pre-processing.

We hire three PhD students in Aston University to annotate the dataset. After reading papers in each group, each annotator is asked to first pick out all discriminative sentences in each paper and then write reference summaries delivering the major differences for each group. Additional instructions are given to annotators: Each reference summary should be no more than 300 words; and the discriminative sentences should enable the judgment of which group the paper belongs to. Equipped with the annotated dataset, two parts of evaluations are performed: evaluation of differential topic models and evaluation of the summarization methods.

5.2 Evaluations on *dTM*

In this section, we compare *dTM-Dirichlet* and *dTM-SAGE* with other three topic models in terms of model perplexity and topic coherences: (1) the standard *LDA* topic model, which we run across the corpus and perform Newton optimization to update hyper-parameters; (2) *SAGE*, which a sparse additive generative model proposed in (Eisenstein et al., 2011), and the non-parametric Jeffrey's prior make it parameter-free; (3) the variant of *DualSum*, which is proposed for update summarization (Delort and Alfonseca, 2012) and revised to perform comparative summarization by replacing pairs of collection-specific distributions with group-specific distributions. We implement the variant of *DualSum*, *dTM-Dirichlet* and *dTM-SAGE* models. Experimental settings are detailed below.

Settings for the variant of *DualSum*. The dirichlet priors for word distributions are empirically set to 0.1 and $\alpha_\lambda = (2.0,2.0,1.0)$ to encourage more words generated from the group-specific distributions and document-specific distributions.

Settings for *dTM-Dirichlet*. The dirichlet priors for word distributions α_Φ and α_θ are set to 0.1. For other papramenters, we set the number of group-independent topics $K = 20$, the prior for the topic distribution $\alpha_\gamma = 50/K$, and the prior for the group-specific word controller $\alpha_\lambda = 2.0$. *Beta*(2.0, 2.0) yields equal probabilities that words sampling from the group-specific distribution and the group-independent distributions.

Settings for *dTM-SAGE*. Parameters are set the same as those in dTM-Dirichlet: $\alpha_\Phi = \alpha_\theta = 0.1$, $K = 20$, $\alpha_\gamma = 50/K$ and $\alpha_\lambda = 2.0$. The variational distribution of the variance $\boldsymbol{\sigma}$ is $Gamma(\tilde{\alpha}, \tilde{b})$ which is initialized as $\tilde{\alpha} = 10.0$ and $\tilde{b} = 5.0$. The initialization for $\boldsymbol{\vartheta_0}$ and $\boldsymbol{\eta}$ are from the Uniform distribution $U(0,1)$ and the Normal distribution $N(0,0.5)$ respectively.

First, we investigate the model perplexity. Perplexity is a general measure for evaluating the generative ability of a probabilistic topic model. We compute the perplexity on a held-out test set, 20% of the original dataset. Note that we calculate the perplexity for all models except the variant of *DualSum*, since it models the document-specific distribution for each document and thus there is no natural way to assign probability to new document. For the variant of *DualSum*, we train the model on the whole dataset and report the results on the test set, though it by no means can reflect the generalization capacity of the model.

Perplexity results are shown in the first row in Table 3, from which we can see that the perplexity scores decrease by 7% and 13% respectively between *dTM-Dirichlet* and standard *LDA* and between *dTM-SAGE* and standard *SAGE*. The better results of differential topic models over the standard ones are due to the discrimination between group-specific topics and group-independent topics. Both *SAGE* methods outperform their counterparts of the Dirichlet-multinomial, because the sparsity-inducing prior enables *SAGE* to control sparsity adaptively without over-fitting (Eisenstein et al., 2011).

SAGE	dTM-Dirichlet	dTM-SAGE
sentence, topic, query document, summary, word, generative, model, vertice, distribution,	sentence, summary, document, topic, rouge, extract, score, select, multi, system	sentence, rouge, ilp, duc, tac, summary, timeline, lexrank, redundant, mead

Table 4: Top 10 Words for the Group *SUMMA*.

	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-SU4	Precision
Baselines					
Centroid	0.23084	0.01867	0.21739	0.05672	0.383
LexPageRank	0.25334	0.02092	0.23822	0.06767	0.417
MMR	0.28272	0.02817	0.26333	0.08094	0.433
State-of-the-arts					
DSS	0.30898	0.03766	0.29346	0.09239	0.600
CDS	0.31749	0.03717	0.29047	0.09340	0.549
TM-based Methods					
Basic LDA (dsw)	0.29812	0.03625	0.27940	0.08865	0.517
Variant of DualSum (dsw)	0.37445	0.04584	0.34542	0.11245	0.650
dTM-Dirichlet (dsw)	0.33024	0.06047	0.31388	0.12363	0.700
dTM-SAGE (dsw)	0.39173	0.06800	0.35764	0.12716	0.717
dTM-SAGE (dgp)	**0.42266**	**0.08801**	**0.38519**	**0.16205**	**0.750**

Table 5: Comparison of Rouge Scores (F-score) and Precision.

Summary by dTM-Dirichlet.

①Most of the existing multi-document summarization methods decompose the documents into sentences and work directly in the sentence space using a term-sentence matrix.

②Bayesian sentences-based topic model, every sentences in a document is assumed to be associated to a unique latent topic.

③While previous MDS systems have focused primarily on salience and coverage but not coherence, G-Flow generates an ordered summary by jointly optimizing coherence and salience.

④Markov Random Walk Model (MRW) Graphs methods have been successfully applied to weighting sentences for generic and query-focused summarization.

⑤The topic distributions are used to get the sentence scores and rank sentences.

Summary by dTM-SAGE.

①In recent years, three major techniques have emerged to perform multi-document summarization: graph-based methds such as LexRank, Biased-LexRank for query-focused summarization, language models such as KLSum and variants based on topic models, such as BayeSum and TopicSum.

② Bayesian Query-Focused Summarization, we present BayeSum (Bayesian summarization), a model for sentence extraction in query-focused summarization.

③Sentence Selection Strategy, The task of timeline summarization aims to produce a summary for each time and the generated summary should meet criteria such as relevance, coverage and coherence.

④Models that use more structure in the representation of documents have also been proposed for generating more coherent and less redundant summaries, such as HierSUM and TTM.

⑤In generating a summary, we use MMR (Maximal Marginal Relevance for multi-document) to avoid redundancy in a summary.

Table 6: Comparison of 5-Sentence Summary Generated by *dTM-Dirichlet* and *dTM-SAGE*.

To check the quality of the generated group-specific topics, we investigate various topic coherence measures. The intuition behind the topic coherence measures is that words clustering into a single topic tend to co-occur in the same document. It has been previously verified that topic coherence score is highly correlated with human-judged topic coherence in many works. We rely on Palmetto library (Röder et al., 2015), an online open source implementation, which offers a framework to calculate many coherence measures within a reference corpus of the English Wikipedia.

In our experiment, we compare three widely-used coherence scores over the five topic models: (1) *C_A*, which is the pairwise comparison of the top words based on a context window of size 5, and proposed in (Aletras and Stevenson, 2013); (2) *C_V*, which is a one-set segmentation of the top words based on a sliding window of size 110, and proposed in (Röder et al., 2015); (3) *C_UCI*, which is the pointwise mutual information (PMI) of all word pairs of the top words based on a sliding window of size 10, and proposed in (Newman et al., 2010).

We focus on the group-specific topics. For each group-specific topic-word distribution we get a word list containing the top-20 words and calculate the coherence scores for each word list. The topic coherence results shown in Table 3 are the average coherence scores of the three group word lists. The coherence scores are calculated within two reference corpus: the English Wikipedia (*Wiki*) and the original dataset (*Intra*). Table 4 shows the top 10 words selected by *SAGE*, *dTM-SAGE* and *dTM-Dirichlet* for the group *SUMMA*. Main observations found in Table 3 include:

(1) The three differential topic models generally perform much better than the standard *LDA* and *SAGE* models on all coherence measures, which shows the superiority of our *dTM* models by distinguishing group-specific words and group-independent words;

(2) *dTM-SAGE* consistently performs the best among all the five models in terms of *C_A* and *C_UCI* with the significant increase at least by 6.5% over *dTM-Dirichlet* and 8.2% over the variant of *DualSum*, which shows the advantage of *dTM-SAGE* in accurately ranking the group-specific words due to the essence of the differential word model;

(3) *dTM-Dirichlet* outperforms the variant of *DualSum* with *C_A* and *C_V*, however, it performs nearly the same or even worse when measured with *C_UCI*.

As shown in Table 4, words selected by *dTM-SAGE* (like rouge, lexrank, redundant) are more informative and discriminative than words selected by *SAGE* and *dTM-Dirichlet*.

5.3 Evaluations on Summarization

To evaluate the quality of the generated summaries, we compare our *dTM*-based comparative summarization methods with five other typical methods under ROUGE metrics (Lin and Hovy, 2003). Further, to check the discriminative ability of the comparative summaries, following the evaluation method of (Wang et al., 2012), we investigate the precision of the discriminative sentence selection.

In our experiment, we implement three types of summarization methods: (1) Generic baseline methods, including the centroid-based method (Radev et al. 2004), the graph-based method LexPag-

eRank (Radev et al., 2004) and the MMR-based method (Carbonell and Goldstein, 1998); (2) State-of-the-art comparative summarization methods, including the discriminative sentence selection (*DSS*) method (Wang et al., 2012) and the complementary dominating set (*CDS*) method (Shen and Li, 2010); (3) TM-based comparative summarization methods, which combine four different TMs with two sentence scoring strategies DCS_dsw and DCS_dgp defined in section 4.1, including the basic $LDA\ (dsw)$, the variant of $DualSum\ (dsw)$, *dTM-Dirichlet* (dsw), *dTM-SAGE* (dsw) and *dTM-SAGE* (dgp). For each group, we select 20 sentences to form the final summary.

First, we examine the precision of the discriminative sentence selection. For each group we have 20 sentences in a summary and count how many sentences belong to the annotated discriminative sentence set. Comparisons of the precision results of discriminative sentence selection by different methods are shown at the last column in Table 5. From the precision results, we find that (1) our *dTM*-based comparative summarization methods can select over 70% discriminative sentences, which significantly outperform the state-of-the-art methods with a nearly 20% increase on the precision score; (2) All generic summarization methods perform rather worse due to different concerns on summarization resulting in the lack of discriminative ability of summaries.

We use ROUGE-1.5.5 toolkit to evaluate the quality of generated summaries by comparing them with human-written reference summaries. In our experiment, we limit the length of all summaries to 250 words and report the average ROUGE scores (F-Scores) on various summarization methods in Table 5. According to the results, we observe that: (1) our *dTM*-based comparative summarization methods perform much better (paired t-test with $p<0.05$) than all the baselines, which demonstrates that targeting at a different goal for summarizing the general information among document groups, generic summarization methods are less applicable for comparative summarization, though by removing redundancy, MMR performs better than the other two baselines but still lags behind other summarization methods specifically proposed for comparative summarization; (2) our *dTM*-based comparative summarization methods significantly outperform (paired t-test with $p<0.05$) the other two state-of-the-art comparative summarization methods, which shows that summarizing differences by extracting group-specific topics is more effective than directly summarizing at the sentence level; (3) Both *dTM-SAGE* methods achieve better ROUGE scores than *dTM-Dirichlet*, which is ascribe to the advantage of a differential word model contributing to more informative and discriminative group-specific topics (discussed in section 5.2); and, (4) For *dTM-SAGE*, the greedy sentence selection schema based on DCS_dgp is more effective than simply ranking sentence with DCS_dsw.

5.4 Summary Example

We show an example of the summary generated for the group *SUMMA* by our *dTM-SAGE* and *dTM-Dirichlet* in Table 6. Looking into the summaries, we find that all sentences in both summaries are related to summarization but different in the degree of their discriminative ability. Apparently, the summary generated by *dTM-SAGE* is more specific and unique to summarization, while the summary generated by *dTM-Dirichlet* still contains some general information about topic models in sentence ② and ⑤. Another observation is that the summary of *dTM-SAGE* tends to contain more salient group-specific terms that may not occur in most of group documents but still possess high discrimination, like '*query-focused*', '*MMR*' and '*HierSUM*'. In contrast, the summary by *dTM-Dirichlet* covers more background group-specific words, like '*summarization*' and '*MDS*'. Although these background group-specific terms are discriminative for the group, they are relatively less informative than the salient terms for the purpose of summarization.

6 Conclusions

This paper studies the differential topic models for comparative summarization on cross-area scientific papers. A differential topic model *dTM-SAGE* is proposed to capture the unique characteristics of each document group and generate coherent group-specific topics. A greedy sentence selection method with two sentence discriminative capacity scoring schemas is designed to automatically generate summary for *dTM*-based comparative summarization methods, which achieve significant improvements with various ROUGE metrics. The analysis on experiment results shows that the summaries generated by our *dTM-SAGE* method can cover major differences for each group.

Acknowledgements

Research was partially supported by National Science Foundation of China and National Science Foundation of Jiangsu (No. BK20140895).

Reference

Ahmed, A., and Xing, E. P. 2010. Staying informed: supervised and semi-supervised multi-view topical analysis of ideological perspective. In *Proceedings of EMNLP*, 1140-1150.

Aletras, N., and Stevenson, M. 2013. Evaluating topic coherence using distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics*, 13-22.

Boudin, F., and El-Bèze, M. 2008. A scalable MMR approach to sentence scoring for multi-document update summarization. In *Proceedings of COLING*, 23-26.

Blei, D. M. 2012. Probabilistic topic models. *Communications of the ACM*, 55(4), 77-84.

Cai, X., Li, W., Ouyang, Y., and Yan, H. 2010. Simultaneous ranking and clustering of sentences: a reinforcement approach to multi-document summarization. In *Proceedings of COLING*, 134-142.

Celikyilmaz, A., and Hakkani-Tur, D. 2010. A hybrid hierarchical model for multi-document summarization. In *Proceedings of ACL*, 815-824.

Celikyilmaz, A., and Hakkani-Tür, D. 2011. Discovery of topically coherent sentences for extractive summarization. In *Proceedings of ACL*, 491-499.

Delort, J. Y., and Alfonseca, E. 2012. DualSum: a topic-model based approach for update summarization. In *Proceedings of the European Chapter of ACL*, 214-223.

Erkan, G., and Radev, D. R. 2004. LexPageRank: Prestige in Multi-Document Text Summarization. In *Proceedings of EMNLP*, Vol. 4, 365-371.

Eisenstein, J., Ahmed, A., and Xing, E. P. 2011. Sparse additive generative models of text. In *Proceedings of ICML*, 1041-1048.

Fisher, S., and Roark, B. 2008. Query-focused supervised sentence ranking for update summaries. In *Proceedings of the TAC*.

Gupta, V., and Lehal, G. S. 2010. A survey of text summarization extractive techniques. *Journal of Emerging Technologies in Web Intelligence*, 2(3), 258-268.

Jha, R., Coke, R. and Radev, D. 2015. Surveyor: A system for generating coherent survey articles for scientific topics. In *Proceedings of AAAI*, 2167-2173.

Lin, C. Y., and Hovy, E. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *HLT-NAACL*, 71-78.

Li, W., Wei, F., Lu, Q., and PNR, Y. H. 2008. Ranking sentences with positive and negative reinforcement for query-oriented update summarization. In *Proceedings of COLING*, 489-496.

Li, P., Wang, Y., Gao, W., and Jiang, J. 2011. Generating aspect-oriented multi-document summarization with event-aspect model. In *Proceedings of EMNLP*, 1137-1146.

Li, C., Qian, X., and Liu, Y. 2013. Using Supervised Bigram-based ILP for Extractive Summarization. In *Proceedings of ACL*, 1004-1013.

Li, W. 2015. Abstractive Multi-document Summarization with Semantic Information Extraction. In *Proceedings of EMNLP*,1908-1913.

Mihalcea, R., and Tarau, P. 2004. TextRank: Bringing order into texts. In *Proceedings of EMNLP, 404-411*.

Newman, D., Lau, J. H., Grieser, K., and Baldwin, T. 2010. Automatic evaluation of topic coherence. In *Proceedings of NAACL*, 100-108.

Paul, M. 2009. Cross-collection topic models: Automatically comparing and contrasting text. *Master's thesis, University of Illinois Urbana Champaign, IL, USA.*

Paul, M., and Girju, R. 2009. Cross-cultural analysis of blogs and forums with mixed-collection topic models. In *Proceedings of EMNLP*, 1408-1417.

Paul, M., and Girju, R. 2010. A two-dimensional topic-aspect model for discovering multi-faceted topics. In *Proceedings of AAAI*, 545-550.

Radev, D. R., Jing, H., and Budzikowska, M. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the NAACL-ANLP*, 21-30.

Radev, D. R., Jing, H., Styś, M., and Tam, D. 2004. Centroid-based summarization of multiple documents. *Information Processing & Management*, *40*(6), 919-938.

Röder, M., Both, A., and Hinneburg, A. 2015. Exploring the space of topic coherence measures. In *Proceedings of the 8th ACM international conference on Web search and data mining*, 399-408. ACM.

Sekine, S., and Nobata, C. 2003. A survey for multi-document summarization. In *Proceedings of the HLT-NAACL*, Vol.5, 65-72.

Shen, C., and Li, T. 2010. Multi-document summarization via the minimum dominating set. In *Proceedings of COLING*, 984-992.

Wan, X., Yang, J., and Xiao, J. 2007. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of ACL*, Vol.7, 552-559.

Wang, D., Zhu, S., Li, T., and Gong, Y. 2012. Comparative document summarization via discriminative sentence selection. *ACM Transactions on Knowledge Discovery from Data* (*TKDD*), 6(3), 12.

Zhai, C., Velivelli, A., and Yu, B. 2004. A cross-collection mixture model for comparative text mining. In *Proceedings of ACM SIGKDD*, 743-748. ACM.

Zhuge, H. 2016. Multi-Dimensional Summarization in Cyber-Physical Society, Morgan Kaufmann.

Appendix A. Variational Inference of ϑ_0 and η_g

Generally, we take MAP (maximum a posterior) estimation for the background word distribution ϑ_0 and the group component vectors η and develop variational inference techniques for all other variables.

In dTM-SAGE, the lower bound L with regarding to ϑ_0, η and σ is:

$$L = \log P(\theta_0 \mid \alpha_\theta) + \sum_g \sum_m \sum_n E_Q[\log P(w_{g,m,n} \mid s_{g,m,n} = 0, \theta_0, \boldsymbol{\eta_g})]$$

$$+\sum_g E_Q[\log P(\boldsymbol{\eta_g} \mid 0, \boldsymbol{\sigma_g})] + \sum_g E_Q[\log P(\boldsymbol{\sigma_g} \mid \alpha_\sigma)] - \sum_g E_Q[\log Q(\boldsymbol{\sigma_g})]$$

Maximize L with respect to $\boldsymbol{\vartheta_0}$:

$$L(\theta_0) = \sum_v (\alpha_g^v - 1) * \log \vartheta_0^v + \sum_g \sum_m \sum_n \tilde{\lambda}_{gmn}^0 * \{\vartheta_0^{w_{g,m,n}} - \log(\sum_{v'} \exp(\eta_g^{v'} + \theta_0^{v'}))\}$$

By Assuming $T(v) = \frac{\exp(\eta_g^v + \theta_0^v)}{\sum_j^V \exp(\eta_g^j + \theta_0^j)}$, taking derivatives with respect to ϑ_0^v:

$$\frac{\partial L}{\partial \vartheta_0^v} = \frac{\alpha_g^v - 1}{\vartheta_0^v} + \sum_g \sum_m \sum_n \tilde{\lambda}_{gmn}^0 * \{I(w_{g,m,n} = v) * (1 - T(v)) + I(w_{g,m,n} \neq v) * (-T(v))\}$$

We use Newton-Raphson method to optimize $\boldsymbol{\vartheta_0}$. First, we derive the Hessian matrix by setting:

$$H_{vv}(\theta_0) = \frac{\partial^2 L}{\partial \vartheta_0^{v2}} = -\frac{\alpha_g^v - 1}{(\vartheta_0^v)^2} + \sum_g \sum_m \sum_n \tilde{\lambda}_{gmn}^0 * (T(v)^2 - T(v))$$

$$H_{vv'}(\theta_0) = \frac{\partial^2 L}{\partial \vartheta_0^v \partial \vartheta_0^{v'}} = \sum_g \sum_m \sum_n \tilde{\lambda}_{gmn}^0 * T(v)T(v')$$

After getting Hessian matrix, we invert it with Sherman-Morrison formula and compute the Newton step: $\Delta\boldsymbol{\vartheta_0} = H^{-1}(\boldsymbol{\vartheta_0})\nabla_{\boldsymbol{\vartheta_0}} L(\boldsymbol{\vartheta_0})$. Same procedure on η:

$$\frac{\partial L}{\partial \eta_g^v} = -\eta_g^v * [(\tilde{a}_g^v - 1)\tilde{b}_g^v]^{-1} + \sum_m \sum_n \tilde{\lambda}_{gmn}^0 * \{I(w_{g,m,n} = v) * (1 - T(v)) + I(w_{g,m,n} \neq v) * (-T(v))\}$$

$$H_{vv}(\boldsymbol{\eta}_g) = \frac{\partial^2 L}{\partial \eta_g^{v2}} = -[(\tilde{a}_g^v - 1)\tilde{b}_g^v]^{-1} + \sum_g \sum_m \sum_n \tilde{\lambda}_{gmn}^0 * (T(v)^2 - T(v))$$

$$H_{vv'}(\boldsymbol{\eta}_g) = \frac{\partial^2 L}{\partial \eta_g^v \partial \eta_g^{v'}} = \sum_g \sum_m \sum_n \tilde{\lambda}_{gmn}^0 * T(v)T(v')$$

Bridging the gap between extractive and abstractive summaries: Creation and evaluation of coherent extracts from heterogeneous sources

Darina Benikova[§†], **Margot Mieskes**[§*], **Christian M. Meyer**[§‡], **Iryna Gurevych**[§‡]
[§]Research Training Group AIPHES
[†]Language Technology Lab, University of Duisburg-Essen
[*]Information Science, University of Applied Sciences, Darmstadt
[‡]Ubiquitous Knowledge Processing (UKP) Lab, Technische Universität Darmstadt
https://www.aiphes.tu-darmstadt.de

Abstract

Coherent extracts are a novel type of summary combining the advantages of manually created abstractive summaries, which are fluent but difficult to evaluate, and low-quality automatically created extractive summaries, which lack coherence and structure. We use a corpus of heterogeneous documents to address the issue that information seekers usually face – a variety of different types of information sources. We directly extract information from these, but minimally redact and meaningfully order it to form a coherent text. Our qualitative and quantitative evaluations show that quantitative results are not sufficient to judge the quality of a summary and that other quality criteria, such as coherence, should also be taken into account. We find that our manually created corpus is of high quality and that it has the potential to bridge the gap between reference corpora of abstracts and automatic methods producing extracts. Our corpus is available to the research community for further development.

1 Introduction

The sheer amount of information available via news agencies, social media, scientific publications, etc. overwhelm information seekers. Not only the information overload itself, but also the heterogeneity of the data poses an obstacle to the aggregation and assessment of information, as text structures, styles, and forms of information presentation vary across different genres and domains of text. Therefore, automatic summarization of multiple heterogeneous sources is a key research demand.

Previous efforts such as DUC[1], TAC[2], and MultiLing[3] have mostly focused on homogeneous data. Heterogeneous data poses many new challenges to Multi-Document Summarization (MDS) that have not been extensively covered so far, such as abstracting from deviating text structures, maintaining topic diversity, resolving potentially opposing opinions, and adapting a text for different target audiences. In our work, we aim at creating a novel MDS corpus of heterogeneous sources.

Besides the lack of heterogeneous MDS corpora, we consider the type of summaries available in MDS corpora in general as an even more severe problem: Existing corpora contain primarily *abstractive summaries*, whereas automatic systems typically utilize extractive methods yielding *extractive summaries* (i.e., bags of sentences taken from the source documents). Producing a coherent, human-understandable text, however, requires steps beyond mere extraction. These steps include, among others, compression, co-reference resolution, paraphrasing, and content structuring. Though there are multiple works striving for abstractive summaries, it is very hard to evaluate them, since current MDS corpora do not explicitly address these tasks.

Although ROUGE is the primary evaluation metric used so far, it only compares n-gram overlap between at least one model summary and a reference summary and thus gives a mere *quantitative* impression of the lexical coverage, leaving the coherence of a summary largely unconsidered.

[1] http://duc.nist.gov
[2] http://www.nist.gov/tac
[3] http://multiling.iit.demokritos.gr

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1039–1050, Osaka, Japan, December 11-17 2016.

Corpus	Summary type	Genre	Lang.	Topics	Doc/Topic	Summary size
DUC (2001–2003)	Abstracts	News	en	30–60	10	50–400 words
DUC (2004)	Abstracts	News	en, ar	50	10	10–100 words
DUC (2005–2007)	Abstracts	News	en	50	≥ 25	250 words
TAC (2008–2011)	Abstracts	News, blogs	en	44	10	100 words
TAC (2014)	Abstracts	Scientific	en	50	10	250 words
MultiLing (2011; 2013)	Abstracts	News	7	10	10	240–250 words
MultiLing (2015)	Abstracts	News	10	15	10	240–250 words
Loupy et al. (2010)	Abstracts	News	fr	20	20	≈ 200 words
Hirao et al. (2004)	Abstracts	News	ja	N/A	N/A	5–10 % characters
Ulrich et al. (2008)	Abstracts & Extracts	e-Mails	en	30	≈ 11	250 words†
Goldstein et al. (2000a)	Extracts	News	en	25	10	≥ 10 sentences†
Zechner (2002)	Extracts	Trans. speech	en	23	N/A	N/A
Carenini et al. (2007)	Extracts	e-Mails	en	20	≥ 4	30 % sentences
Nakano et al. (2010)	Extracts	Heterogeneous	en	24	352	$\approx$ 2,560 characters
Lloret et al. (2013)	Extracts	Heterogeneous	en	310	10	100–200 words
Our work	Coherent Extracts	Heterogeneous	de	10	4–14	≈ 500 words†

Table 1: Comparison of manual multi-document corpora († = no predefined summary size)

In contrast, there are corpora containing only extracts, which are suitable for evaluating extractive MDS systems. However, this yields an overly artificial evaluation, since neither their creation guidelines nor the evaluation metrics employed take coherence, fluency, and organization of the summary into account. We argue that such bag-of-sentences-based extracts are of limited use for real users, who prefer a coherent, meaningful text over an unordered collection of sentences.

In order to bridge the gap between the system results (primarily extractive) and the reference data (primarily abstractive), we need high-quality corpora that allow for training and evaluation of automatic systems and their intermediate steps, such as the identification of important nuggets or the removal of redundancy. To this end, we introduce and analyze a novel type of reference summary: *coherent extracts*. Coherent extracts are based on information explicitly taken from the source documents allowing to create a straight-forward evaluation setup based on the selection of important content. At the same time, coherent extracts have a meaningful order and are redacted to ensure a coherent text, which is substantially more helpful for humans than purely extractive summaries. Our vision is to use the data collected during structuring and redaction in order to research improved extractive MDS systems.

In this paper, we introduce our idea of coherent extracts and we create and evaluate a novel MDS corpus of coherent extracts from heterogeneous source documents. In Section 2 we discuss previous work in manual summarization and corpus construction. We describe coherent extracts and our summarization workflow in Section 3 and we evaluate our work in Section 4. Finally, we conclude the paper in Section 5.

2 Related work

Our work is inspired by two main aspects of multi-document summarization: the manual creation of MDS corpora and the evaluation of the resulting summaries. There are two main approaches to summarization: abstractive and extractive. Humans typically produce abstractive summaries by paraphrasing and condensing information from the sources. Automatic systems primarily generate extractive summaries by copying information from the sources without modifying and ordering them. For evaluating an automatic system, however, we require gold standard summaries that fit well to the system output, cf. (Endres-Niggemeyer, 2012). This major discrepancy between the human-generated abstractive gold standard and the system-generated extractive summaries has largely been neglected in previous work.

2.1 Existing multi-document summarization corpora

Endres-Niggemeyer (2012) states that the process of summarization includes three subtasks: *analysis of the input information*, *performance of the core summarization task (condensation, abstraction)* and *representation of the results in an appropriate form*. Schlesinger et al. (2003, p. 46) observe that MDS "lacks complementary documentation of procedures and methodologies for human performance".

Table 1 gives an overview over various MDS corpora in comparison to our work. Most of them are based on homogeneous data such as news (DUC, TAC, MultiLing, Goldstein et al. (2000a), Hirao et al. (2004)), e-mails (Ulrich et al., 2008), and transcribed speech (Zechner, 2002). Lloret et al. (2013) take the top ten results from a regular web search engine as the source documents for their summaries, but do not specify the types of documents used. Therefore, it is unclear whether their data is heterogeneous or not. To the best of our knowledge, only Nakano et al. (2010) specifically use a collection of heteregeneous documents and discuss the corresponding challenges, such as subjectivity and contradictions.

Although the majority of the works presented in Table 1 mention guidelines for producing the summaries, most of them neither describe these guidelines in detail, nor divide the summarization process into clear-cut subtasks. Zechner (2002) gives descriptions for individual annotation tasks, but they are not necessarily subsequently applied. Additionally, their process is targeted towards the summarization of transcribed speech, which requires additional tasks, such as the annotation of topic boundaries, which are not necessary in thematically clustered document collections.

Lloret et al. (2013) use crowdsourcing and therefore the instructions are very limited. Their results show that summaries created via crowdsourcing are unsuitable for reference or gold standard summaries. Although the crowd authors are able to create summaries very quickly, if high-quality results are a priority, it is still advisable to employ trained annotators, despite the slower progress. Hirao et al. (2004) and Nakano et al. (2010) put an extraction task before the actual summarization, but there was no further division of the summarization process. They allow the editing of the text extracts without limitation, thus their summaries cannot be regarded as purely extractive. However, as they do neither provide any instructions on the summary creation process nor evaluate the coherence of the results, it is unknown whether their results are coherent summaries. Hirao et al. (2004) realize a three-step annotation process for producing summaries: (1) extracting important sentences, (2) minimizing redundant sentences, and (3) rewriting the result of the previous step in order to match the size constraint. Their goal is a summary that consists of a collection of informative sentences from the source text.

Another issue in the creation of summaries is coherence and cohesion. None of the methods mentioned above specifically addresses this. Although Zechner (2002, p. 456) states that the "overall goal [...] was to create a concise and readable summary", they do not specify how they reach the goal of coherent summaries. Neither Hirao et al. (2004) nor Zechner (2002) explicitly perform co-reference resolution, although the latter states that it would "certainly be desirable, for the sake of increased coherence and readability, to employ a well-working anaphora resolution component" (ibid., p. 451). Lloret et al. (2013, p. 346) observed that summaries based on crowdsourcing have a poor quality, as they "frequently present lost anaphoric and pronominal references". To our knowledge, none of the mentioned works address this issue in their evaluation.

2.2 Previous summary evaluation

Summaries are primarily evaluated intrinsically, either based on *qualitative* or *quantitative* measures. Early DUC evaluations (Over, 2001; Over and Liggett, 2002; Over and Yen, 2003) used Likert scales, which are psychometric response scales that are often used in questionnaires, for summary evaluation. This evaluation focuses on *qualitative* aspects of a summary. To perform a *quantitative* evaluation of the summaries, model units (Over and Liggett, 2002; Over and Yen, 2003) were used. These are manually annotated semantic units in human summaries, which are compared to automatic summaries according to the frequency of occurrence. Lloret et al. (2013) uses the overall quality for evaluation, based on a five-point Likert-type scale. Later on, Nenkova et al. (2007) introduced the Pyramid method. So-called *Summary Content Units (SCUs)*, which are semantically motivated, subsentential units, serve as the basis for the evaluation. They are variable in length, but not bigger than a sentential clause (Nenkova et al., 2007), and are closely related to model units. In both cases, system summaries are evaluated based on the content overlap rankings. A widely used *quantitative*, automatic evaluation metric is ROUGE (Lin, 2004), which has also been used to evaluate manual summaries, for instance, by Nakano et al. (2010), Lloret et al. (2013), and in DUC evaluations (up to 2007).

In our work, we focus on manual qualitative evaluation using various aspects of text quality, similar to

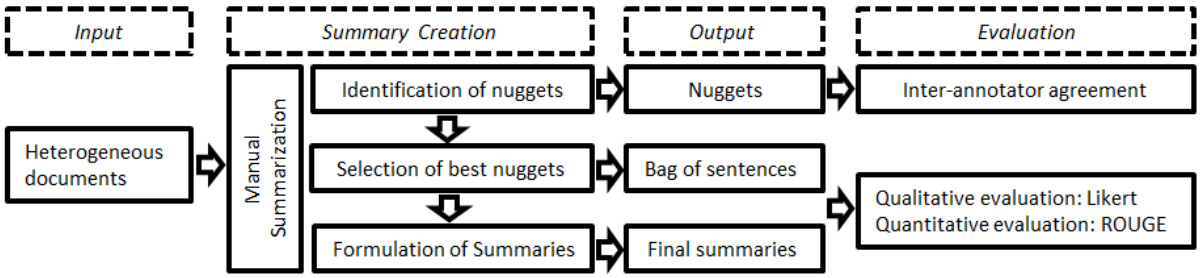


Figure 1: Process of creation and evaluation of coherent extracts and the intermediate processing steps.

the evaluation approach taken in the early DUC evaluations (Over, 2001; Over and Liggett, 2002; Over and Yen, 2003). More specifically, we focus on coherence in summary evaluation and statistically prove that this feature, which is very important for human understanding of text, has not only been neglected in the creation, but also in the evaluation process. In particular, we focus on coherence in our evaluation, which has previously been neglected in the creation process. Our results indicate that this feature is very important for human understanding and should therefore gain more importance.

3 Coherent Extracts

We propose *coherent extracts* as a new type of reference summary, bridging the gap between human-optimized abstracts and (automatic) extracts lacking coherence and fluency. Coherent extracts consist of important nuggets extracted from (multiple) source documents, which are meaningfully ordered and minimally redacted to ensure a coherent text that is close to a completely manually written text.

To substantiate this idea, we create a novel MDS corpus of coherent extracts, which we make freely available.[4] Figure 1 shows an overview of our corpus creation and evaluation procedure discussed in the remaining paper. First, we introduce the *Input* in form of the source document collection and topic selection procedure (Section 3.1). Our MDS process consists of three steps: In the first step the most important units are selected (so-called *Identification of Nuggets*). Next, the selected nuggets are clustered into groups with similar content and for each cluster the best nugget is selected (*Selection of best nuggets*). Finally, during the *Formulation of Summaries*, the best nuggets are co-reference resolved, grammaticalized and sorted coherently. Details for each step are given in Section 3.2 and in Meyer et al. (2016). In Section 4, we describe the *Evaluation* of our work using inter-annotator agreement measures, automatic summarization scoring with ROUGE, and human judgments based on Likert scales.

3.1 Heterogeneous Document Collection

Deutscher Bildungsserver[5] (DBS) is a large web portal that collects links to educational resources in German. As a publicly funded project, it fulfills an important information need of different stakeholders, including teachers, students, parents, politicians, and educational researchers. DBS contains links to highly heterogeneous text types, such as interviews, book reviews, teaching material, NGO profiles, newspaper and scientific articles. Although all topics focus on the educational domain, they vary enormously in terms of audience and specialized subject area (e.g., educational reforms, social impact, educational research, political decisions). Domain experts organize the links according to different topics, different audiences (teachers, students, parents, etc.) and strive for introducing each topic with a brief overview text. However, writing a manual overview for each topic is hardly feasible, as there are hundreds of topics, which are continuously updated by the editors. As we consider the processing of heterogeneous documents as a particularly important research challenge, the DBS makes an excellent basis and use case for research in automatic MDS of heterogeneous sources.

For our corpus, we select 10 topics from DBS and crawl their linked pages yielding between 4 and 14 documents per topic. Table 2 shows the properties of each topic and its documents. Our choice

[4] https://github.com/AIPHES/DBS
[5] http://www.bildungsserver.de

Topic	Source documents #	Sentences	Tokens	Genres News-like	Other	Summaries #	∅ Sentences	∅ Tokens
1. Preventing violence	10	1,446	12,276	2	8	3	39 ± 22	625 ± 343
2. Environmental education	10	184	2,446	2	8	3	12 ± 7	216 ± 37
3. Healthy Nutrition	13	894	6,551	2	11	4	23 ± 14	431 ± 278
4. Lateral entry teachers	5	134	1,950	4	1	4	20 ± 11	482 ± 266
5. Reform of the dual education system	10	432	5,972	2	8	3	26 ± 17	485 ± 263
6. Mediation	7	253	2,651	1	6	3	30 ± 13	453 ± 187
7. German Qualifications Framework	4	127	2,089	3	1	3	16 ± 3	310 ± 71
8. Reading	12	1,270	20,212	9	3	4	78 ± 56	1509 ± 1056
9. Short-term secondary school diploma	14	522	5,275	6	8	3	28 ± 18	576 ± 393
10. Right-wing extremism and racism	7	176	1,933	2	5	3	18 ± 7	336 ± 93
Average per topic	9.2	544	6,136	3.3	5.9	3.3	30 ± 20	566 ± 378
Total sum	92	5,438	61,355	33	59	33	987	18,694

Table 2: Overview of the topics, documents, genres, and summaries in our MDS corpus

of topic and document sizes is guided by the parameters of similar corpora (e.g., the DUC corpora) and by practical constraints of the DBS portal. The genre columns highlight the heterogeneity of the individual topics and the corpus overall. Although there is at least one news-like article in each topic, the source documents are highly heterogeneous, covering brief introductory texts, syllabi, term definitions, presentations of interest groups, and many other text genres.

Once we identify the source documents, we shorten those with more than 40 pages and topics with more than 30,000 tokens in order to reduce the annotation effort. We do not remove documents in order to conserve the full diversity of subtopics and genres. Conversely, we define a minimum of 3 documents and 1,500 tokens to ensure that there is enough information to summarize. We use the boilerplate removal tool by Habernal et al. (2016) to remove HTML markup and non-content (e.g., advertisement, navigation menu).

3.2 Creating Coherent Extracts

We divide the task of creating coherent extracts into three subsequent steps, which in turn consist of further substeps which we explain in an extensive, publicly available annotation guidebook. These detailed instructions ensure a reliable, reproducible annotation workflow and allow us to produce informative, non-redundant, grammatically correct coherent extracts.

Step 1: Identification of nuggets We first instruct the annotators to read each text *before* annotating, to ensure that they understand the full context of the topic. Afterwards, they identify important *nuggets* in the source documents. Our notion of nuggets includes a grammatical and a semantic constraint. The grammatical constraint is that a nugget has to contain at least one verb and one corresponding noun and can maximally span over one sentence. The minimum restriction is based on the assumption that this would facilitate the reformulation of nuggets to grammatically correct sentences as part of the third annotation step (see below). Our semantic constraint of nuggets is similar to previous work on *model units* (Over and Liggett, 2002), *factoids* (Halteren and Teufel, 2003), and *SCUs* (Nenkova et al., 2007) in that they should be important in the context of the given topic.

Step 2: Selection of best nuggets *Redundancy* is a more severe issue in MDS (Goldstein et al., 2000b; Goldstein et al., 2000a) compared to single-document summarization, as recurring facts are more likely to occur in multiple different documents about the same topic than within a single one. Thus, we ask the annotators to group the selected nuggets from step 1 if they "convey the same meaning using different wording" (Bhagat and Hovy, 2013, p. 463). For selecting the *best nugget* of a group, the annotators are required to prefer declarative, objective, and co-reference-free nuggets wherever possible.

Step 3: Formulation of summaries We divide the formulation of the coherent extract into several substeps: First, we ask the annotators to resolve co-references in the best nuggets in order to prevent references to context which are not part of any nugget. This is an important requirement for producing

coherent extracts, although it has been largely ignored in previous works.

Subsequently, the annotators reformulate the nuggets into full, grammatically correct sentences, which form the basis for our coherent extract. The annotators perform as few changes as necessary to prevent them from adding additional semantic content. Typical transformations are the introduction of function words or clarifying indirect speech (e.g., explicitly adding phrases, such as *"according to John Doe"*).

Once the best nuggets are transformed into grammatically correct sentences, the annotators order them in a meaningful way to structure the extract. Depending on the topic, annotators may choose a topic-based or argumentation-based information flow. Based on this structure, the annotators formulate the final extracts. They may add discourse connectives and conjunctions, such as *after*, *however*, or *as well as*. They may re-introduce co-references, if the referring object, person or event is available in the context, to create a fluent and readable text that closely resembles manually written texts. We do not restrict the length of the resulting summaries.

Annotation It is important to note that each annotation step builds upon the results of the previous steps (e.g., the structuring of the sentences builds upon the set of reformulated best nuggets). To implement this annotation setup, we use the MDS*Writer* tool recently introduced by Meyer et al. (2016).

Our annotation team consists of four German native speakers, who are graduate students in linguistics or computational linguistics. We trained the annotators using two smaller topics with 2 and 5 documents. For the actual corpus creation, we assign at least two annotators to each topic and we instruct them to finish all annotation steps of a specific topic before moving to the next one.

Final Corpus Our final corpus consists of 92 input documents with over 61,000 tokens and 33 multi-document summaries with over 18,000 tokens (566 per summary on average). As we did not restrict the length of the summaries, their texts range from 6 to 127 sentences and from 173 to 2,494 tokens. The average token compression rate is 12.2 %. During the summarization process, the annotators identify between 15 and 228 nuggets per topic (48 on average, 1,596 in total), which they process in the later steps in order to remove redundancy and co-references, formulate complete sentences and ultimately the coherent extracts. Our corpus is freely available from our GitHub project page (`https://github.com/AIPHES/DBS`).

4 Evaluation

We evaluate our corpus creation procedure in three experiments: In Section 4.1, we measure the inter-annotator agreement to assess the quality of the identification of nuggets. In Section 4.2, we conduct a quantitative evaluation of our coherent extracts in comparison to bag-of-sentence-based and automatically created summaries using ROUGE. In Section 4.3, we present results on a qualitative evaluation of the three summary types using a range of quality criteria judged on a five-point Likert scale. Finally, Section 4.4 compares the results of the qualitative analysis with the results from the quantitative analysis.

4.1 Inter-Annotator Agreement of the Nugget Identification Step

To ensure the reproducibility of our corpus creation procedure, we first assess how well the annotators agree on the results of the nugget identification step. The agreement scores reveal to what extent the annotation experiment can be repeated with different annotators or data. Furthermore, they yield insight into the degree of subjectivity of the task. Besides the average proportion of observed agreement (A_{O}) of nuggets jointly identified by two annotators, we use the standard metrics from content analysis, namely Fleiss' κ and Krippendorff's α, as defined by Artstein and Poesio (2008).

We first model the nugget identification step as a coding task (i.e., assigning categories to predefined units) by aggregating each annotator's nuggets to the levels of sentences and paragraphs. That is, a sentence or paragraph is considered important, if it contains at least one nugget. This setup allows us to compute the usual A_{O}, κ and α agreement scores. Although κ has been used in some previous works, it is not well-suited for this setup, as it has no explicit notion of missing values. Rather, it assumes that all annotators processed the entire corpus, which is neither true for our work nor for many other previous

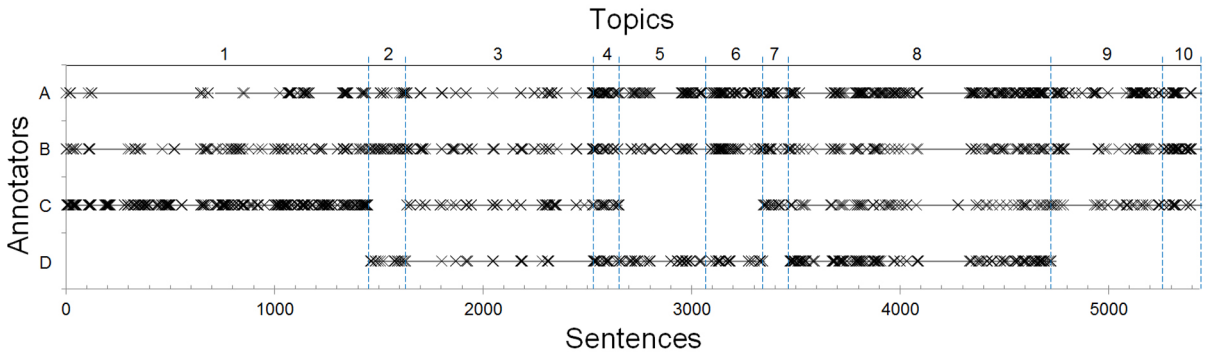


Figure 2: Plot marking all sentences, in which an annotator identified at least one nugget

works, such as Zechner (2002). We therefore recommend to always report Krippendorff's α as well, which uses a similar scale, but explicitly deals with missing annotations.

Since our nugget identification procedure is not limited to selecting entire sentences but to mark spans of varying lengths starting at arbitrary positions, we also model the nugget identification step as a unitizing task (i.e., finding the margins of the unit to be annotated) in order to take the inter-annotator agreement on the nugget boundaries into consideration. Krippendorff's α_{U} is a suitable metric for such setups (Krippendorff, 1995), as it uses the same scale as the regular α and therefore allows for comparison. All measures were calculated using DKPro Statistics (Meyer et al., 2014).[6]

Figure 2 illustrates the resulting annotation. Every $\times$ marks a sentence (bottom x-axis; pooled over all topics indicated by the top x-axis) containing at least one nugget as annotated by one of the four annotators (y-axis). The gray line indicates sentences, which do not contain any nugget, although they have been assigned to a particular annotator. Empty space thus counts towards the missing values mentioned above – i.e., sentences that have not been assigned to the corresponding annotator. The plot shows that the annotators largely vary in the number of nuggets they consider important: Annotator C identified more nuggets in the sentences with low index, while annotator A more extensively marked nuggets in sentences with higher index (Note that the annotators did not perform the task strictly in order of this sentence index). When zooming in, we can observe a fair overlap of sentences considered important or unimportant by a majority of annotators, despite the fact that annotators rarely fully agree on what is important. There are, however, sentences that are very consistently considered unimportant for writing a summary, for example, in topic 8 around sentence index 4200, which is a scientific article about reading with overly detailed information and a list of references.

Table 3 shows the inter-annotator agreement scores in the coding and unitizing setups. Our annotators agree that 85 % of the sentences either contain an important nugget or are not important at all. This is substantially higher than the 56–62 % reported by Goldstein et al. (2000a) and in line with what could be observed in Figure 2. Although agreement scores for nugget identification are rarely reported, we find that our κ score of 0.23 exceeds the $\kappa = 0.13$ reported by Zechner (2002). It is not surprising to find rather low overall κ and α scores, since judging importance is known to be a subjective task. Additionally, we face a highly skewed class distribution between important and unimportant sentences. Disagreement on the smaller class has a high impact on the final κ and α scores. For the unitizing setup, the discrepancy between observed agreement and statistically corrected agreement becomes extreme: Given the large parts of the documents for which no annotator identified a nugget, the observed unitizing disagreement is only 1 %, whereas α_{U} is only 0.19. Comparing α and α_{U} is highly interesting, because the small difference indicates that the annotators mostly agree on a nugget's boundaries. The main decision is therefore *what* to judge as important, not the specific nugget boundaries.

Table 3 also shows the inter-annotator agreement scores individually for each topic. We find a higher agreement for topics with less sentences and tokens (e.g., topic 4), where the annotators mostly agree on the important nuggets. Instead, we observe much lower agreement scores for large topics, in which

[6] https://dkpro.github.io/dkpro-statistics/

IAA setup / Topic ID		1	2	3	4	5	6	7	8	9	10	Overall
Paragraph level	A_O	0.92	0.91	0.97	0.86	0.92	0.89	0.93	0.91	0.95	0.93	0.93
	κ	0.17	0.30	0.34	0.43	0.33	0.39	0.38	0.44	0.41	0.49	0.35
	α	0.17	0.31	0.34	0.43	0.33	0.39	0.38	0.44	0.41	0.49	0.34
Sentence level	A_O	0.82	0.79	0.93	0.74	0.86	0.75	0.75	0.84	0.86	0.81	0.85
	κ	0.10	0.16	0.28	0.29	0.31	0.25	0.18	0.19	0.22	0.31	0.23
	α	0.10	0.16	0.28	0.29	0.31	0.25	0.18	0.19	0.23	0.31	0.22
Nugget level	A_O	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	α_U	0.01	0.07	0.25	0.43	0.17	0.21	0.04	0.16	0.13	0.19	0.19

Table 3: Inter-annotator agreement for our 10 topics at the paragraph, sentence, and nugget levels

α	A	B	C	D
A	—	0.26	0.17	0.28
B	0.26	—	0.17	0.28
C	0.17	0.17	—	0.14
D	0.28	0.28	0.14	—

α_U	A	B	C	D
A	—	0.29	0.18	0.24
B	0.29	—	0.15	0.22
C	0.18	0.15	—	0.05
D	0.24	0.22	0.05	—

Table 4: Pairwise inter-annotator agreement using sentence-based α (left) and nugget-based α_U (right)

the annotators put different foci or in which they identified an overly high or low number of nuggets. Especially for topic 1, the α_U drops extremely, because of the different number of identified nuggets: 49 by annotator A, 76 by B, and 228 by C. Moreover, we find that the specificity of the topic has a large impact on the inter-annotator agreement. Broad topics such as *Reading* (topic 8) tend to have lower agreement than specific topics (e.g., topic 5, *Reform of the dual education system*) with a very clear focus.

In Table 4, we additionally assessed the agreement for each pair of annotators individually. We find the sentence-based α to range from 0.14 to 0.28 and the nugget-based α_U to range from 0.05 to 0.29 across the annotator pairs. Annotator C achieved consistently lower agreement with the others. When looking into the data, we note that this annotator persistently chose more nuggets per topic (58 on average) than the other annotators (45 on average). The qualitative evaluation of the annotator's coherent extracts revealed high quality. Given that our agreement figures are clearly higher than reported for previous corpora, we consider our identified nuggets reliable and our procedure reproducible. However, we also note that the identification of important nuggets is a fairly subjective task. Thus, producing *at least* 3 reference summaries is essential to cover the different notions of what is important.

4.2 Quantitative Evaluation

Quantitative evaluations based on ROUGE have been used both for the evaluation of automatic systems (e.g., in the context of DUC) and for manual summaries, for instance by Nakano et al. (2010) and Lloret et al. (2013). In our evaluation, we compare results on three different summary types: First, we create automatic summaries using various standard summarization methods, including TextRank (Mihalcea and Tarau, 2004), LexRank (Erkan and Radev, 2004), and LSA (Steinberger and Ježek, 2004) as implemented in the SUMY[7] tool.[8] Second, we use *bag of sentences (BoS)*, which are an intermediate result of our manual summarization process, namely the reformulated best nuggets (see Section 3.2). These BoS essentially correspond to purely extractive MDS summaries, as they contain full sentences, which then serve as the basis for coherent extracts. They are already ordered in a meaningful way and annotators are required to not add content, but to only add (for example) connectives in order to create a coherent text (see Section 3.2 for details).[9] Third, we use the coherent extracts, which are the final results of our summarization process. In order to evaluate automatic summaries and BoS we use the manual summaries as references. For the manual summaries, we take individual manual summaries and use the remaining

[7] `https://github.com/miso-belica/sumy`

[8] We used standard settings, except for the length, which we wanted to keep comparable across all summary types.

[9] Examples for the various summary types (automatic, BoS and manual) are available for download.

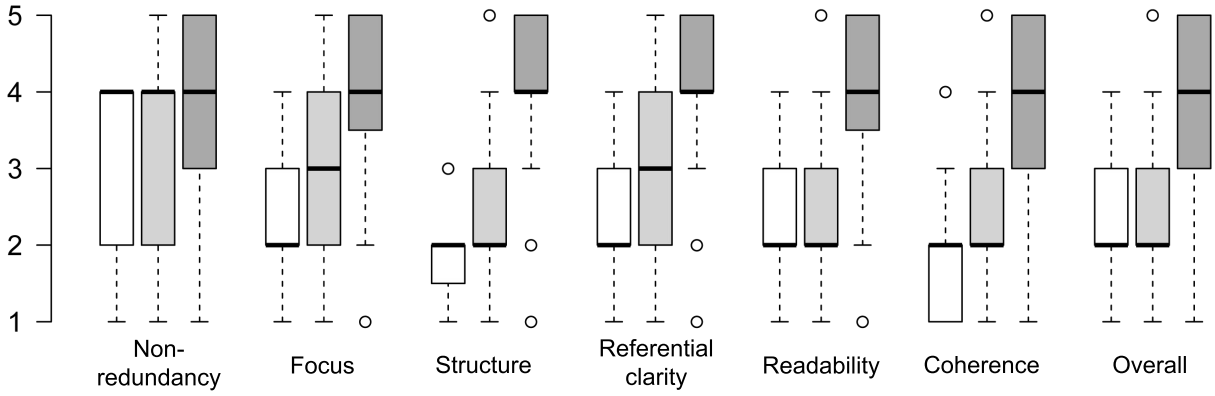


Figure 3: Boxplot of our qualitative judgments of automatic summarization systems (white), bags of sentences (light gray), and coherent extracts (dark gray)

manual summaries as references. The motivation for the latter step is to gain an intuition about the upper bound for the ROUGE scores, but also about the deviation based on individual differences.

We compute the ROUGE-1 recall metric (R-1) for each summary of the three types, in which we use the coherent extracts as peers. For BoS and coherent extracts, we remove the summary of the same author from the set of peers. Our coherent extracts achieve R-1 = 0.42, which is very similar to the R-1 = 0.43 reported by Nakano et al. (2010), but considerably higher than the R-1 = 0.35 discussed by Lloret et al. (2013) for their crowdsourcing setup. BoS achieve the best results with R-1 = 0.49, followed by the coherent extracts (R-1 = 0.42) and then the automatic summaries ($0.32 \leq$ R-1 ≤ 0.35). The scores of automatic methods are comparable to Lloret et al. (2013), but significantly lower than the BoS of coherent extracts. BoS, which are the basis for the coherent extracts, achieve the best results, as they vary less. For the coherent extracts, we asked our annotators to minimally revise the text for readability, fluency, and coherence, which causes a slightly lower n-gram overlap and thus lower R-1 score.

4.3 Qualitative Evaluation

We complement our ROUGE-based evaluation by manually rating the three different summary types on a five-point Likert scale. Our quality criteria are based on early DUC evaluations, but slightly extended for our purposes: We judge *non-redundancy*, *focus*, *structure*, *referential clarity*, *readability*, *coherence*, *length*, *grammaticality*, *spelling*, *layout*, and *overall quality* for 25 automatically produced summaries (as introduced in the previous section) and 16 BoS and coherent extracts from our corpus. We asked six human raters, of mixed gender and age, with different academic degrees in linguistics or computational linguistics, to participate in the evaluation. To prevent a bias, we randomly shuffled the summaries, such that the participants did not know how a summary was created.

Figure 3 shows a boxplot of the average scores for each summary type according to the seven most relevant quality criteria. *Non-redundancy* achieves good results across all summary types. This means that the automatic approaches as well as the two manual approaches (BoS and coherent extracts) manage to compile a mostly redundancy-free summary. However, both automatic approaches and the BoS-based approach fail to achieve high ratings for the other six quality criteria. The criteria *structure* and *readability* vary the most with coherent extracts achieving significantly higher scores than automatic summaries and BoS. *Referential clarity* is – at least for the manual summaries – the most consistently marked evaluation criterion. Additionally, we see that both BoS and automatic methods achieve considerably lower results than the coherent extracts. Especially for the automatic methods, the distribution of values is broadest, whereas for the BoS the distribution is more even. For *focus*, *structure*, and *referential clarity*, we observe a clear pattern with the lowest scores for automatic summaries, slightly better results for BoS, and the best scores for our coherent extracts. In terms of *readability* and *coherence*, both BoS and automatic summaries have similarly low scores, whereas the coherent extracts clearly achieve adequate results. The *overall* score reflects the results on individual quality criteria, in that automatic methods achieve the lowest scores, BoS are slightly better and coherent extracts are by far the best.

4.4 Discussion

Comparing the qualitative and the quantitative evaluation results, we find substantial differences between the automatic summaries, BoS, and coherent extracts. According to ROUGE, the BoS are better than coherent extracts, whereas the manual judgments suggest the opposite. This indicates that even high-quality, redundancy-free BoS with reasonable ROUGE scores are still not good enough to provide reasonable summaries for humans, as they lack in coherence, structure, and focus. We compute the correlation between ROUGE and all criteria judged by human raters, but no rater showed significant results, except for one. This annotator achieved very high scores on the qualitative evaluation, which also shows a significant correlation with ROUGE. It has been shown in the past by Liu and Liu (2010) that ROUGE does not reflect human judgements. Our results support this. But in addition, we show that: (a) BoS as mostly produced by automatic systems are not considered high-quality by humans, (b) although they do achieve very good ROUGE scores and (c) that quality criteria such as *coherence* and *structure*, which are important factors in human judgements, are not covered by a state-of-the-art ROUGE-based evaluation.

The coherent extracts we propose, take these quality criteria into account, as they achieve overall high scores in the manual evaluation. At the same time, they are not restricted to a ROUGE-based evaluation as is the case for abstractive summaries. Rather they allow for evaluating the identification of nuggets (i.e., content selection), redundancy removal, and the formulation of a coherent and fluent summary separately using the intermediate results collected during our corpus creation process. We render this highly important for taking extractive summarization methods to the next level.

5 Conclusion and Further Work

We introduced coherent extracts as a novel type of summary and presented a corresponding MDS corpus based on heterogeneous sources from the educational domain. To compile this corpus, we defined a workflow addressing the identification of important nuggets, the removal of redundancy, and the formulation of a coherent text in separate steps. For each step, we collect the individual results, which allows us to evaluate the corpus creation steps individually as well as their overall quality. We used inter-annotator agreement measures and ROUGE to quantitatively analyze the corpus and we relied on the human judgments for 11 quality criteria on a five-point Likert scale to assess the quality of our corpus and the value of coherent extracts. Our results show that ROUGE overestimates the quality of purely extractive summaries, which especially lack coherence, readability, and structure.

At the same time, our corpus provides data for evaluating automatic MDS systems beyond the typical ROUGE-based setup, because we have an explicit notion of important elements in the input documents and we can trace which sentence of the summary belongs to which source document. Though the latter is also possible with purely extractive summaries, they are not well-suited for human use and thus yield a rather artificial evaluation setup. At the same time, our detailed workflow allows researchers working on MDS systems to reconstruct how humans go beyond mere BoS-type summaries and incorporate this into automatic systems and evaluate the resulting quality. To this end, coherent extracts bridge an important gap between fluent, but difficult-to-evaluate abstracts and low-quality extracts lacking coherence and structure.

In future work, we plan to increase the size of the corpus, such that we can investigate the effects of certain summarizers, topics, genres, or even multiple languages. To complement our human judgments, we also consider to use the Pyramid method (Nenkova et al., 2007) in order to understand different means of summary evaluation for MDS of heterogeneous document collections. Furthermore, we plan to enrich our corpus with abstractive multi-document summaries written by experts. On the one hand this would give us the possibility to compute a qualitative comparison with the new kind of summary described in this paper, on the other hand it would provide data for more accurate evaluation and training.

Acknowledgments

This work has been supported by the DFG-funded research training group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES, GRK 1994/1). We would like to thank our annotators for their valuable contribution.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Rahul Bhagat and Eduard Hovy. 2013. What Is a Paraphrase? *Computational Linguistics*, 39(3):463–472.

Giuseppe Carenini, Raymond T. Ng, and Xiaodong Zhou. 2007. Summarizing email conversations with clue words. In *Proceedings of the 16th International World Wide Web Conference (WWW)*, pages 91–100, Banff, AB, Canada.

Brigitte Endres-Niggemeyer. 2012. *Summarizing Information*. Springer.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22(1):457–479.

Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Jamie Callan. 2000a. Creating and evaluating multi-document sentence extract summaries. In *Proceedings of the Ninth International Conference on Information and Knowledge Management (CIKM)*, pages 165–172, McLean, VA, USA.

Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. 2000b. Multi-Document Summarization By Sentence Extraction. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic Summarization*, pages 40–48, Seattle, WA, USA.

Ivan Habernal, Omnia Zayed, and Iryna Gurevych. 2016. C4Corpus: Multilingual Web-size corpus with free license. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*, pages 914–922, Portorož, Slovenia.

Hans van Halteren and Simone Teufel. 2003. Examining the consensus between human summaries: initial experiments with factoid analysis. In *Proceedings of the 2003 HLT-NAACL Workshop on Text Summarization*, pages 57–64, Edmonton, AB, Canada.

Tsutomu Hirao, Takahiro Fukusima, Manabu Okumura, Chikashi Nobata, and Hidetsugu Nanba. 2004. Corpus and Evaluation Measures for Multiple Document Summarization with Multiple Sources. In *Proceedings of the 20th international conference on Computational Linguistics (COLING)*, pages 535–541, Geneva, Switzerland.

Klaus Krippendorff. 1995. On the reliability of unitizing contiguous data. *Sociological Methodology*, 25:47–76.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL-2004*, pages 25–26, Barcelona, Spain.

Feifan Liu and Yang Liu. 2010. Exploring correlation between ROUGE and human evaluation on meeting summaries. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(1):187–196.

Elena Lloret, Laura Plaza, and Ahmet Ake. 2013. Analyzing the capabilities of crowdsourcing services for text summarization. *Language Resoures and Evaluation*, 47(2):337–369.

Claude de Loupy, Marie Guégan, Christelle Ayache, Somara Seng, and Juan-Manuel Torres Moreno. 2010. A French Human Reference Corpus for Multi-Document Summarization and Sentence Compression. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC)*, pages 3113–3118, Valletta, Malta.

Christian M. Meyer, Margot Mieskes, Christian Stab, and Iryna Gurevych. 2014. DKPro Agreement: An Open-Source Java Library for Measuring Inter-Rater Agreement. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING): System Demonstrations*, pages 105–109, Dublin, Ireland.

Christian M. Meyer, Darina Benikova, Margot Mieskes, and Iryna Gurevych. 2016. MDS*Writer*: Annotation tool for creating high-quality multi-document summarization corpora. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL): System Demonstrations*, pages 97–102, Berlin, Germany.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain.

Masahiro Nakano, Hideyuki Shibuki, Rintaro Miyazaki, Madoka Ishioroshi, Koichi Kaneko, and Tatsunori Mori. 2010. Construction of Text Summarization Corpus for the Credibility of Information on the Web. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC)*, pages 3125–3131, Valletta, Malta.

Ani Nenkova, Rebecca Passonneau, and Kathleen Mckeown. 2007. The Pyramid Method: Incorporating Human Content Selection Variation in Summarization Evaluation. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(2):4.

Paul Over and Walter Liggett. 2002. Introduction to DUC: An Intrinsic Evaluation of Generic News Text Summarization Systems. In *Proceedings of DUC 2002*, Philadelphia, PA, USA.

Paul Over and James Yen. 2003. An Introduction to DUC 2003 Intrinsic Evaluation of Generic News Text Summarization Systems. In *Proceedings of DUC 2003*, Edmonton, AB, Canada.

Paul Over. 2001. Introduction to DUC-2001: an Intrinsic Evaluaton of Generic News Text Summarization Systems. In *Proceedings of DUC 2001*, New Orleans, LA, USA.

Judith D. Schlesinger, John M. Conroy, Mary Ellen Okurowski, and Dianne P. O'Leary. 2003. Machine and Human Performance for Single and Multidocument Summarization. *IEEE Intelligent Systems*, 18(1):46–54.

Josef Steinberger and Karel Ježek. 2004. Using latent semantic analysis in text summarization and summary evaluation. In *Proceedings of the 5th International Conference on Information Systems Implementation and Modelling (ISIM)*, pages 93–100, Rožnov pod Radhoštěm, Czech Republic.

Jan Ulrich, Gabriel Murray, and Giuseppe Carenini. 2008. A Publicly Available Annotated Corpus for Supervised Email Summarization. In *Enhanced Messaging: Papers from the 2008 AAAI Workshop*, Technical Report WS-08-04, pages 77–82. Menlo Park, CA: AAAI Press.

Klaus Zechner. 2002. Automatic Summarization of Open-Domain Multiparty Dialogues in Diverse Genres. *Computational Linguistics*, 28(4):447–485.

Chinese Poetry Generation with Planning based Neural Network

Zhe Wang[†], Wei He[‡], Hua Wu[‡], Haiyang Wu[‡], Wei Li[‡], Haifeng Wang[‡], Enhong Chen[†]
[†]University of Science and Technology of China, Hefei, China
[‡]Baidu Inc., Beijing, China
xiaose@mail.ustc.edu.cn, cheneh@ustc.edu.cn
{hewei06, wu_hua, wuhaiyang, liwei08, wanghaifeng}@baidu.com

Abstract

Chinese poetry generation is a very challenging task in natural language processing. In this paper, we propose a novel two-stage poetry generating method which first plans the sub-topics of the poem according to the user's writing intent, and then generates each line of the poem sequentially, using a modified recurrent neural network encoder-decoder framework. The proposed planning-based method can ensure that the generated poem is coherent and semantically consistent with the user's intent. A comprehensive evaluation with human judgments demonstrates that our proposed approach outperforms the state-of-the-art poetry generating methods and the poem quality is somehow comparable to human poets.

1 Introduction

The classical Chinese poetry is a great and important heritage of Chinese culture. During the history of more than two thousand years, millions of beautiful poems are written to praise heroic characters, beautiful scenery, love, friendship, etc. There are different kinds of Chinese classical poetry, such as Tang poetry and Song iambics. Each type of poetry has to follow some specific structural, rhythmical and tonal patterns. Table 1 shows an example of quatrain which was one of the most popular genres of poetry in China. The principles of a quatrain include: The poem consists of four lines and each line has five or seven characters; every character has a particular tone, Ping (the level tone) or Ze (the downward tone); the last character of the second and last line in a quatrain must belong to the same rhyme category (Wang, 2002). With such strict restrictions, the well-written quatrain is full of rhythmic beauty.

In recent years, the research of automatic poetry generation has received great attention. Most approaches employ rules or templates (Tosa et al., 2008; Wu et al., 2009; Netzer et al., 2009; Oliveira, 2009; Oliveira, 2012), genetic algorithms (Manurung, 2004; Zhou et al., 2010; Manurung et al., 2012), summarization methods (Yan et al., 2013) and statistical machine translation methods (Jiang and Zhou, 2008; He et al., 2012) to generate poems. More recently, deep learning methods have emerged as a promising discipline, which considers the poetry generation as a sequence-to-sequence generation problem (Zhang and Lapata, 2014; Wang et al., 2016; Yi et al., 2016). These methods usually generate the first line by selecting one line from the dataset of poems according to the user's writing intents (usually a set of keywords), and the other three lines are generated based on the first line and the previous lines. The user's writing intent can only affect the first line, and the rest three lines may have no association with the main topic of the poem, which may lead to semantic inconsistency when generating poems. In addition, topics of poems are usually represented by the words from the collected poems in the training corpus. But as we know, the words used in poems, especially poems written in ancient time, are different from modern languages. As a consequence, the existing methods may fail to generate meaningful poems if a user wants to write a poem for a modern term (e.g., Barack Obama).

In this paper, we propose a novel poetry generating method which generates poems in a two-stage procedure: the contents of poems ("what to say") are first explicitly planned, and then surface realization ("how to say") is conducted. Given a user's writing intent which can be a set of keywords, a sentence or

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1051–1060, Osaka, Japan, December 11-17 2016.

静夜思	Thoughts in a Still Night
床前明月**光**， (P P Z Z P)	The luminous moonshine before my bed,
疑是地上**霜**。 (* Z Z P P)	Is thought to be the frost fallen on the ground.
举头望明月， (* Z P P Z)	I lift my head to gaze at the cliff moon,
低头思故**乡**。 (P P Z Z P)	And then bow down to muse on my distant home.

Table 1: An example of Tang poetry. The tone is shown at the end of each line. P represents the level-tone, and Z represents the downward-tone; * indicates that the tone can be either. The rhyming characters are in boldface.

even a document described by natural language, the first step is to determine a sequence of sub-topics for the poem using a poem planning model, with each line represented by a sub-topic. The poem planning model decomposes the user's writing intent into a series of sub-topics, and each sub-topic is related to the main topic and represents an aspect of the writing intent. Then the poem is generated line by line, and each line is generated according to the corresponding sub-topic and the preceding generated lines, using a recurrent neural network based encoder-decoder model (RNN enc-dec). We modify the RNN enc-dec framework to support encoding of both sub-topics and the preceding lines. The planning based mechanism has two advantages compared to the previous methods. First, every line of the generated poem has a closer connection to user's writing intent. Second, the poem planning model can learn from extra knowledge source besides the poem data, such as large-scale web data or knowledge extracted from encyclopedias. As a consequence, it can bridge the modern concepts and the set of words covered by ancient poems. Take the term "Barack Obama" as the example: using the knowledge from encyclopedias, the poem planning model can extend the user's query, Barack Obama, to a series of sub-topics such as outstanding, power, etc., therefore ensuring semantic consistency in the generated poems.

The contribution of this paper is two-fold. First, we propose a planning-based poetry generating framework, which explicitly plans the sub-topic of each line. Second, we use a modified RNN encoder-decoder framework, which supports encoding of both sub-topics and the preceding lines, to generate the poem line by line.

The rest of this paper is organized as follows. Section 2 describes some previous work on poetry generation and compares our work with previous methods. Section 3 describes our planning based poetry generation framework. We introduce the datasets and experimental results in Section 4. Section 5 concludes the paper.

2 Related Work

Poetry generation is a challenging task in NLP. Oliveira (2009; 2012) proposed a Spanish poem generation method based on semantic and grammar templates. Netzer et al. (2009) employed a method based on word association measures. Tosa et al. (2008) and Wu et al. (2009) used a phrase search approach for Japanese poem generation. Greene et al. (2010) applied statistical methods to analyze, generate and translate rhythmic poetry. Colton et al. (2012) described a corpus-based poetry generation system that uses templates to construct poems according to the given constrains. Yan et al. (2013) considered the poetry generation as an optimization problem based on a summarization framework with several constraints. Manurung (2004; 2012) and Zhou et al. (2010) used genetic algorithms for generating poems. An important approach to poem generation is based on statistical machine translation (SMT). Jiang and Zhou (2008) used an SMT-based model in generating Chinese couplets which can be regarded as simplified regulated verses with only two lines. The first line is regarded as the source language and translated into the second line. He et al. (2012) extended this method to generate quatrains by translating the previous line to the next line sequentially.

Recently, deep learning methods achieve great success in poem generation. Zhang and Lapata (2014) proposed a quatrain generation model based on recurrent neural network (RNN). The approach generates the first line from the given keywords with a recurrent neural network language model (RNNLM) (Mikolov et al., 2010) and then the subsequent lines are generated sequentially by accumulating the status of the lines that have been generated so far. Wang et al. (2016) generated the Chinese Song iambics using an end-to-end neural machine translation model. The iambic is generated by translating the pre-

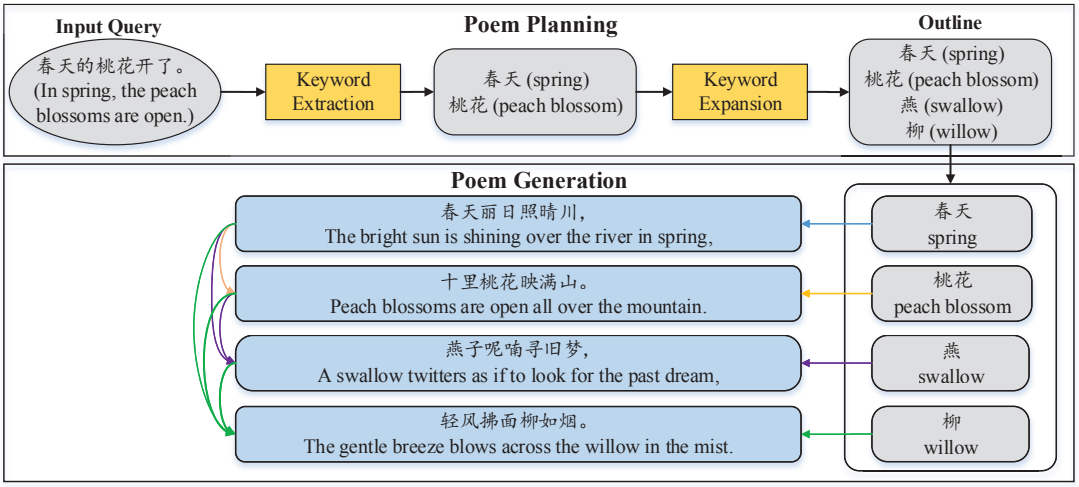


Figure 1: Illustration of the planning based poetry generation framework.

vious line into the next line sequentially. This procedure is similar to SMT, but the semantic relevance between sentences is better. Wang et al. (2016) did not consider the generation of the first line. Therefore, the first line is provided by users and must be a well-written sentence of the poem. Yi et al. (2016) extended this approach to generate Chinese quatrains. The problem of generating the first line is resolved by a separate neural machine translation (NMT) model which takes one keyword as input and translates it into the first line. Marjan Ghazvininejad and Knight (2016) proposed a poetry generation algorithm that first generates the rhyme words related to the given keyword and then generated the whole poem according to the rhyme words with an encoder-decoder model (Sutskever et al., 2014).

Our work differs from the previous methods as follows. First, we don't constrain the user's input. It can be some keywords, phrases, sentences or even documents. The previous methods can only support some keywords or must provide the first line. Second, we use planning-based method to determine the topic of the poem according to the user's input, with each line having one specific sub-topic, which guarantees that the generated poem is coherent and well organized, therefore avoiding the problem of the previous method that only the first line is guaranteed to be related to the user's intent while the next lines may be irrelevant with the intention due to the coherent decay problem (He et al., 2012; Zhang and Lapata, 2014; Wang et al., 2016; Yi et al., 2016). Third, the rhythm or tone in (Zhou et al., 2010; Yan et al., 2013; Zhang and Lapata, 2014; Yi et al., 2016; Marjan Ghazvininejad and Knight, 2016) is controlled by rules or extra structures, while our model can automatically learn constrains from the training corpus. Finally, our poem generation model has a simpler structure compared with those in (Zhang and Lapata, 2014; Yi et al., 2016).

3 Approaches

3.1 Overview

Inspired by the observation that a human poet shall make an outline first before writing a poem, we propose a planning-based poetry generation approach (PPG) that first generates an outline according to the user's writing intent and then generates the poem. Our PPG system takes user's writing intent as input which can be a word, a sentence or a document, and then generates a poem in two stages: Poem Planning and Poem Generation. The two-stage procedure of PPG is illustrated in Figure 1.

Suppose we are writing a poem that consists of N lines with l_i representing the i-th line of the poem. In the Poem Planning stage, the input query is transformed into N keywords $(k_1, k_2, ..., k_N)$, where k_i is the i-th keyword that represents the sub-topic for the i-th line. In the Poem Generation stage, l_i is generated by taking k_i and $l_{1:i-1}$ as input, where $l_{1:i-1}$ is a sequence concatenated by all the lines generated previously, from l_1 to l_{i-1}. Then the poem can be generated sequentially, and each line is

generated according to one sub-topic and all the preceding lines.

3.2 Poem Planning

3.2.1 Keyword Extraction

The user's input writing intent can be represented as a sequence of words. There is an assumption in the Poem Planning stage that the number of keywords extracted from the input query Q must be equal to the number of lines N in the poem, which can ensure each line takes just one keyword as the sub-topic. If the user's input query Q is too long, we need to extract the most important N words and keep the original order as the keywords sequence to satisfy the requirement.

We use TextRank algorithm (Mihalcea and Tarau, 2004) to evaluate the importance of words. It is a graph-based ranking algorithm based on PageRank (Brin and Page, 1998). Each candidate word is represented by a vertex in the graph and edges are added between two words according to their co-occurrence; the edge weight is set according to the total count of co-occurrence strength of the two words. The TextRank score $S(V_i)$ is initialized to a default value (e.g. 1.0) and computed iteratively until convergence according to the following equation:

$$S(V_i) = (1-d) + d \sum_{V_j \in E(V_i)} \frac{w_{ji}}{\sum_{V_k \in E(V_j)} w_{jk}} S(V_j), \tag{1}$$

where w_{ij} is the weight of the edge between node V_j and V_i, $E(V_i)$ is the set of vertices connected with V_i, and d is a damping factor that usually set to 0.85 (Brin and Page, 1998), and the initial score of $S(V_i)$ is set to 1.0.

3.2.2 Keyword Expansion

If the user's input query Q is too short to extract enough keywords, we need to expand some new keywords until the requirement of keywords number is satisfied. We use two different methods for keywords expansion.

RNNLM-based method. We use a Recurrent Neural Network Language Model (RNNLM) (Mikolov et al., 2010) to predict the subsequent keywords according to the preceding sequence of keywords: $k_i = \arg\max_k P(k|k_{1:i-1})$, where k_i is the i-th keyword and $k_{1:i-1}$ is the preceding keywords sequence.

The training of RNNLM needs a training set consisting of keyword sequences extracted from poems, with one keyword representing the sub-topic of one line. We automatically generate the training corpus from the collected poems. Specifically, given a poem consisting of N lines, we first rank the words in each line according to the TextRank scores computed on the poem corpus. Then the word with the highest TextRank score is selected as the keyword for the line. In this way, we can extract a keyword sequence for every poem, and generate a training corpus for the RNNLM based keywords predicting model.

Knowledge-based method. The above RNNLM-based method is only suitable for generating sub-topics for those covering by the collected poems. This method does not work when the user's query contains out-of-domain keywords, for example, a named entity not covered by the training corpus.

To solve this problem, we propose a knowledge-based method that employs extra sources of knowledge to generate sub-topics. The extra knowledge sources can be used include encyclopedias, suggestions of search engines, lexical databases (e.g. WordNet), etc. Given a keyword k_i, the key idea of the method is to find some words that can best describe or interpret k_i. In this paper, we use the encyclopedia entries as the source of knowledge to expand new keywords from k_i. We retrieve those satisfying all the following conditions as candidate keywords: (1) the word is in the window of $[-5, 5]$ around k_i; (2) the part-of-speech of the word is adjective or noun; (3) the word is covered by the vocabulary of the poem corpus. Then the candidate words with the highest TextRank score are selected as the keywords.

3.3 Poem Generation

In the Poem Generation stage, the poem is generated line by line. Each line is generated by taking the keyword specified by the Poem Planning model and all the preceding text as input. This procedure can be

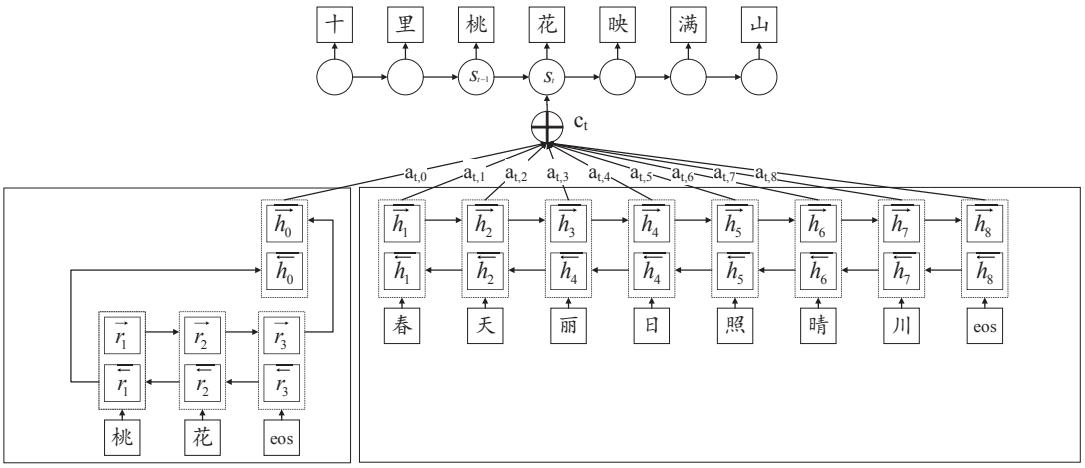


Figure 2: An illustration of poem generation model.

considered as a sequence-to-sequence mapping problem with a slight difference that the input consists of two different kinds of sequences: the keyword specified by the Poem Planning model and the previously generated text of the poem. We modify the framework of an attention based RNN encoder-decoder (RNN enc-dec) (Bahdanau et al., 2014) to support multiple sequences as input.

Given a keyword $\mathbf{k}$ which has T_k characters, i.e. $\mathbf{k} = \{a_1, a_2, ..., a_{T_k}\}$, and the preceding text $\mathbf{x}$ which has T_x characters, i.e. $\mathbf{x} = \{x_1, x_2, ..., x_{T_x}\}$, we first encode $\mathbf{k}$ into a sequence of hidden states $[r_1 : r_{T_k}]$, and $\mathbf{x}$ into $[h_1 : h_{T_x}]$, with bi-directional Gated Recurrent Unit (GRU) (Cho et al., 2014) models. Then we integrate $[r_1 : r_{T_k}]$ into a vector r_c by concatenating the last forward state and the first backward state of $[r_1 : r_{T_k}]$, where

$$r_c = \begin{bmatrix} \overrightarrow{r_{T_k}} \\ \overleftarrow{r_1} \end{bmatrix}. \tag{2}$$

We set $h_0 = r_c$, then the sequence of vectors $\mathbf{h} = [h_0 : h_{T_x}]$ represents the semantics of both $\mathbf{k}$ and $\mathbf{x}$, as illustrated in Figure 2. Notice that when we are generating the first line, the length of the preceding text is zero, i.e. $T_x = 0$, then the vector sequence $\mathbf{h}$ only contains one vector, i.e. $\mathbf{h} = [h_0]$, therefore, the first line is actually generated from the first keyword.

For the decoder, we use another GRU which maintains an internal status vector s_t, and for each generation step t, the most probable output y_t is generated based on s_t, context vector c_t and previous generated output y_{t-1}. This can be formulated as follows:

$$y_t = \arg\max_y P(y|s_t, c_t, y_{t-1}). \tag{3}$$

After each prediction, s_t is updated by

$$s_t = f(s_{t-1}, c_{t-1}, y_{t-1}). \tag{4}$$

$f(\cdot)$ is an activation function of GRU and c_t is recomputed at each step by the alignment model:

$$c_t = \sum_{j=1}^{T_h} a_{tj} h_j. \tag{5}$$

h_j is the j-th hidden state in the encoder's output. The weight a_{tj} is computed by

$$a_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_h} \exp(e_{tk})}, \tag{6}$$

where

$$e_{tj} = v_a^T \tanh(W_a s_{t-1} + U_a h_j). \tag{7}$$

e_{tj} is the attention score on h_j at time step t. The probability of the next word y_t can be defined as:

$$P(y_t|y_1, ..., y_{t-1}, \mathbf{x}, \mathbf{k}) = g(s_t, y_{t-1}, c_t), \tag{8}$$

where $g(\cdot)$ is a nonlinear function that outputs the probability of y_t.

The parameters of the poem generation model are trained to maximize the log-likelihood of the training corpus:

$$\arg\max \sum_{n=1}^{N} logP(\mathbf{y_n}|\mathbf{x_n}, \mathbf{k_n}). \tag{9}$$

4 Experiments

4.1 Dataset

In this paper, we focus on the generation of Chinese quatrain which has 4 lines and each line has the same length of 5 or 7 characters. We collected 76,859 quatrains from the Internet and randomly chose 2,000 poems for validation, 2,000 poems for testing, and the rest for training.

All the poems in the training set are first segmented into words using a CRF based word segmentation system. Then we calculate the TextRank score for every word. The word with the highest TextRank score is selected as the keyword for the line. In this way, we can extract a sequence of 4 keywords for every quatrain. From the training corpus of poems, we extracted 72,859 keyword sequences, which is used to train the RNN language model for keyword expansion (see section 3.2.2). For knowledge-based expansion, we use Baidu Baike[1] and Wikipedia as the extra sources of knowledge.

After extracting four keywords from the lines of a quatrain, we generate four triples composed of (the keyword, the preceding text, the current line), for every poem. Take the poem in Table 1 as example, the generated triples are shown in Table 2. All the triples are used for training the RNN enc-dec model proposed in section 3.3.

Keyword	The Preceding Text	Current Line
床	–	床前明月光
霜	床前明月光	疑是地上霜
明月	床前明月光; 疑是地上霜	举头望明月
故乡	床前明月光; 疑是地上霜; 举头望明月	低头思故乡

Table 2: Training triples extracted from the quatrain in Table 1.

4.2 Training

For the proposed attention based RNN enc-dec model, we chose the 6,000 most frequently used characters as the vocabulary for both source and target sides. The word embedding dimensionality is 512 and initialized by word2vec (Mikolov et al., 2013). The recurrent hidden layers of the decoder and two encoders contained 512 hidden units. Parameters of our model were randomly initialized over a uniform distribution with support [-0.08,0.08]. The model was trained with the AdaDelta algorithm (Zeiler, 2012), where the minibatch was set to be 128. The final model is selected according to the perplexity on the validation set.

4.3 Evaluation

4.3.1 Evaluation Metrics

It is well known that accurate evaluation of text generation system is difficult, such as the poetry generation and dialog response generation (Zhang and Lapata, 2014; Schatzmann et al., 2005; Mou et al., 2016). There are thousands of ways to generate an appropriate and relative poem or dialog response given a specific topic, the limited references are impossible to cover all the correct results. Liu et al. (2016) has recently shown that the overlap-based automatic evaluation metrics adapted for dialog responses, such

[1] A collaborative online encyclopedia provided by Chinese search engine Baidu: `http://baike.baidu.com`.

Poeticness	Does the poem follow the rhyme and tone requirements ?
Fluency	Does the poem read smoothly and fluently?
Coherence	Is the poem coherent across lines?
Meaning	Does the poem have a certain meaning and artistic conception?

Table 3: Evaluation standards in human judgement.

Models	**Poeticness**		**Fluency**		**Coherence**		**Meaning**		**Average**	
	5-char	**7-char**	**5-char**	**7-char**	**5-char**	**7-char**	**5-char**	**7-char**	**5-char**	**7-char**
SMT	3.25	3.22	2.81	2.48	3.01	3.16	2.78	2.45	2.96	2.83
RNNLM	2.67	2.55	3.13	3.42	3.21	3.44	2.90	3.08	2.98	3.12
RNNPG	3.85	3.52	3.61	3.02	3.43	3.25	3.22	2.68	3.53	3.12
ANMT	**4.34**	4.04	**4.61**	4.45	4.05	4.01	4.09	4.04	4.27	4.14
PPG	4.11	**4.15**	4.58	**4.56***	**4.29***	**4.49****	**4.46****	**4.51****	**4.36****	**4.43****

Table 4: Human evaluation results of all the systems. Diacritics ** ($p < 0.01$) and * ($p < 0.05$) indicate that our model (PPG) is significantly better than all other systems.

as BLEU and METEOR, have little correlation with human evaluation. Therefore, we carry out a human study to evaluate the poem generation models. Following (He et al., 2012; Yan et al., 2013; Zhang and Lapata, 2014), we use four evaluation standards for human evaluators to judge the poems: "Poeticness", "Fluency", "Coherence", "Meaning". The detailed illustration can be seen in Table 3. The score of each aspect ranges from 1 to 5 with the higher score the better. Each system generates twenty 5-character quatrains and twenty 7-character quatrains. All the generated poems are evaluated by 5 experts and the rating scores are averaged as the final score.

4.3.2 Baselines

We implemented several poetry generation methods as baselines and employed the same pre-processing method for all the methods.

SMT. A Chinese poetry generation method based on Statistical Machine Translation (He et al., 2012). A poem is generated iteratively by "translating" the previous line into the next line.

RNNLM. A method for generating textual sequences (Graves, 2013), which is proposed by Mikolov et al. (2010). The lines of a poem are concatenated together as a character sequence which is used to train the RNNLM.

RNNPG. In the approach of RNN-based Poem Generator (Zhang and Lapata, 2014), the first line is generated by a standard RNNLM and then all the other lines are generated iteratively based on a context vector encoded from the previous lines.

ANMT. The Attention based Neural Machine Translation method. It considers the problem as a machine translation task, which is similar to the traditional SMT approach. The main difference is that in ANMT, the machine translation system is a standard attention based RNN enc-dec framework (Bahdanau et al., 2014).

4.3.3 Results

The results of the human evaluation are shown in Table 4. We can see that our proposed method, Planning based Poetry Generation (PPG), outperforms all baseline models in average scores. The results are consistent with both settings of 5-character and 7-character poem generations.

The poems generated by SMT are better in Poeticness than RNNLM, which demonstrates that the translation based method can better capture the mapping relation between two adjacent lines. ANMT is a strong baseline which performs better than SMT, RNNLM and RNNPG, but lower than our approach. Both ANMT and PPG use the attention based enc-dec framework. The main difference is that our method defines the sub-topics for each line before generating the poem. The ANMT method just translates the preceding text into the next line. Without the guide of sub-topics, the system tends to generate more general but less meaningful results. In contrast, our approach explicitly considers the keywords, which

	Wrongly Identified MP as HP	Cannot Distinguish	Successfully Identified HP as HP
Normal Group	38.6%	11.3%	50.1%
Expert Group	6.3%	10.0%	83.7%

Table 5: Blind test to distinguish Human-written Poems (HP) from Machine-generated Poems (MP).

秋夕湖上 By a Lake at Autumn Sunset 一夜秋凉雨湿衣， A cold autumn rain wetted my clothes last night, 西窗独坐对夕晖。 And I sit alone by the window and enjoy the sunset. 湖波荡漾千山色， With mountain scenery mirrored on the rippling lake, 山鸟徘徊万籁微。 A silence prevails over all except the hovering birds.	秋夕湖上 By a Lake at Autumn Sunset 荻花风里桂花浮， The wind blows reeds with osmanthus flying, 恨竹生云翠欲流。 And the bamboos under clouds are so green as if to flow down. 谁拂半湖新镜面， The misty rain ripples the smooth surface of lake, 飞来烟雨暮天愁。 And I feel blue at sunset .

Table 6: A pair of poems selected from the blind test. The left one is a machine-generated poem, and the right one is written by Shaoti Ge, a poet lived in the Song Dynasty.

has better controls of the sub-topic for every line. From the results of the human evaluation, it can be seen that the proposed method obtained very close performances in Poeticness and Fluency compared with ANMT but much higher Coherence and Meaning scores, which verified the effectiveness of the sub-topic prediction model.

4.4 Automatic Generation vs. Human Poet

We conducted an interesting evaluation that directly compares our automatic poem generation system with human poets, which is similar to the Turing Test (Turing, 1950). We randomly selected twenty poems from the test set, which are written by ancient Chinese poets. We used the titles of these poems as the input and generated 20 poems by our automatic generation system. Therefore, the machine-generated poems were under the same subject with human-written poems. Then we asked some human evaluators to distinguish the human-written poems from machine-generated ones. We had 40 evaluators in total. All of them were well-educated and had Bachelor or higher degree. Four of them were professional in Chinese literature and were assigned to the Expert Group. The other thirty-six evaluators were assigned to the Normal Group. In the blind test, we showed a pair of poems and their title to the evaluator at each time, and the evaluator was asked to choose from three options: (1) poem A is written by the human; (2) poem B is written by the human; (3) cannot distinguish which one is written by the human.

The evaluation results are shown in Table 5. We can see that 49.9% of the machine-generated poems are wrongly identified as the human-written poems or cannot be distinguished by the normal evaluators. But for expert evaluators, this number drops to 16.3%. We can draw two conclusions from the result: (1) under the standard of normal users, the quality of our machine-generated poems is very close to human poets; (2) but from the view of professional experts, the machine-generated poems still have some obvious shortages comparing to human-written poems. Table 6 gives an example for a pair of poems selected from our blind test.

4.5 Generation Examples

Besides the ancient poems in Table 6, our method can generate poems based any modern terms. Table 7 shows some examples. The title of the left poem in Table 7 is 啤酒 *(beer)*, the keywords given by our poem planning model are 啤酒 *(beer)*, 香醇 *(aroma)*, 清爽 *(cool)* and 醉 *(drunk)*. The title of the right one is a named entity 冰心 *(Xin Bing)*, who was a famous writer. The poem planning system generates three keywords besides 冰心 *(Xin Bing)*: 春水 *(spring river)*, 繁星 *(stars)* and 往事 *(the past)*, which are all related to the writer's works.

啤酒	冰心
Beer	Xin Bing
今宵啤酒两三缸， I drink glasses of beer tonight, 杯底香醇琥珀光。 With the bottom of the glass full of aroma and amber light. 清爽金风凉透骨， Feeling cold as the autumn wind blows, 醉看明月挂西窗。 I get drunk and enjoy the moon in sight by the west window.	一片冰心向月明， I open up my pure heart to the moon, 千山春水共潮生。 With the spring river flowing past mountains. 繁星闪烁天涯路， Although my future is illuminated by stars, 往事萦怀梦里行。 The past still lingers in my dream.

Table 7: Examples of poems generated from titles of modern concepts.

5 Conclusion and Future Work

In this paper, we proposed a novel two-stage poetry generation method which first explicitly decomposes the user's writing intent into a series of sub-topics, and then generates a poem iteratively using a modified attention based RNN encoder-decoder framework. The modified RNN enc-dec model has two encoders that can encode both the sub-topic and the preceding text. The evaluation by human experts shows that our approach outperforms all the baseline models and the poem quality is somehow comparable to human poets. We have also demonstrated that using encyclopedias as an extra source of knowledge, our approach can expand users' input into appropriate sub-topics for poem generation. In the future, we will investigate more methods for topic planning, such as PLSA, LDA or word2vec. We will also apply our approach to other forms of literary genres e.g. Song iambics, Yuan Qu etc., or poems in other languages.

6 Acknowledgments

This research was supported by the National Basic Research Program of China (973 program No. 2014CB340505), the National Key Research and Development Program of China (Grant No. 2016YFB1000904), the National Science Foundation for Distinguished Young Scholars of China (Grant No. 61325010) and the Fundamental Research Funds for the Central Universities of China (Grant No. WK2350000001). We would like to thank Xuan Liu, Qi Liu, Tong Xu, Linli Xu, Biao Chang and the anonymous reviewers for their insightful comments and suggestions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30:107–117.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Simon Colton, Jacob Goodwin, and Tony Veale. 2012. Full-face poetry generation. In *ICCC*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *EMNLP*.

Jing He, Ming Zhou, and Long Jiang. 2012. Generating chinese classical poems with statistical machine translation models. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Long Jiang and Ming Zhou. 2008. Generating chinese couplets using a statistical mt approach. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 377–384. Association for Computational Linguistics.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Ruli Manurung, Graeme Ritchie, and Henry Thompson. 2012. Using genetic algorithms to create meaningful poetic text. *Journal of Experimental & Theoretical Artificial Intelligence*, 24(1):43–64.

Hisar Manurung. 2004. An evolutionary algorithm approach to poetry generation.

Yejin Choi Marjan Ghazvininejad, Xing Shi and Kevin Knight. 2016. Generating topical poetry. In *EMNLP*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *EMNLP*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, volume 2, page 3.

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

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings the 26th International Conference on Computational Linguistics*.

Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad. 2009. Gaiku: Generating haiku with word associations norms. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 32–39. Association for Computational Linguistics.

Hugo Gonçalo Oliveira. 2009. Automatic generation of poetry: an overview. *Universidade de Coimbra*.

Hugo Gonçalo Oliveira. 2012. Poetryme: a versatile platform for poetry generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1:21.

Jost Schatzmann, Kallirroi Georgila, and Steve Young. 2005. Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *6th SIGdial Workshop on DISCOURSE and DIALOGUE*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Naoko Tosa, Hideto Obara, and Michihiko Minoh. 2008. Hitch haiku: An interactive supporting system for composing haiku poem. In *Entertainment Computing-ICEC 2008*, pages 209–216. Springer.

Alan M Turing. 1950. Computing machinery and intelligence. *Mind*, 59(236):433–460.

Qixin Wang, Tianyi Luo, Dong Wang, and Chao Xing. 2016. Chinese song iambics generation with neural attention-based model. *CoRR*, abs/1604.06274.

Li Wang. 2002. A summary of rhyming constraints of chinese poems.

Xiaofeng Wu, Naoko Tosa, and Ryohei Nakatsu. 2009. New hitch haiku: An interactive renku poem composition supporting tool applied for sightseeing navigation system. In *Entertainment Computing–ICEC 2009*, pages 191–196. Springer.

Rui Yan, Han Jiang, Mirella Lapata, Shou-De Lin, Xueqiang Lv, and Xiaoming Li. 2013. i, poet: Automatic chinese poetry composition through a generative summarization framework under constrained optimization. In *IJCAI*.

Xiaoyuan Yi, Ruoyu Li, and Maosong Sun. 2016. Generating chinese classical poems with rnn encoder-decoder. *CoRR*, abs/1604.01537.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, Doha, Qatar, October. Association for Computational Linguistics.

Cheng-Le Zhou, Wei You, and Xiaojun Ding. 2010. Genetic algorithm and its implementation of automatic generation of chinese songci. *Journal of Software*, 21(3):427–437.

Predicting sentential semantic compatibility for aggregation in text-to-text generation

Victor Chenal
School of Computer Science
McGill University
victor.chenal@mail.mcgill.ca

Jackie Chi Kit Cheung
School of Computer Science
McGill University
jcheung@cs.mcgill.ca

Abstract

We examine the task of aggregation in the context of text-to-text generation. We introduce a new aggregation task which frames the process as grouping input sentence fragments into clusters that are to be expressed as a single output sentence. We extract datasets for this task from a corpus using an automatic extraction process. Based on the results of a user study, we develop two gold-standard clusterings and corresponding evaluation methods for each dataset. We present a hierarchical clustering framework for predicting aggregation decisions on this task, which outperforms several baselines and can serve as a reference in future work.

1 Introduction

In text-to-text generation, existing sentence compression and sentence fusion systems assume that the input takes the form of one or more sentences, and the output is one suitably modified sentence (Barzilay et al., 1999; Knight and Marcu, 2000; Marsi and Krahmer, 2005; Filippova, 2010; Thadani and McKeown, 2013, for example). This line of work has led to new, knowledge-lean methods for sentence generation for applications such as text simplification and automatic summarization.

The next step in expanding the scope of text-to-text generation is to relax assumptions about the forms of the input and output, with the eventual goal of generating entire passages or documents in an abstractive manner. We focus in this paper on aggregation, the task of determining what input units belong in the same output sentence. Aggregation is an important step in micro-planning in the traditional data-to-text NLG pipeline (Reiter et al., 2000). However, it has been little examined within the context of text-to-text generation outside of restricted syntactic contexts, such as entity-driven noun phrase rewriting (Nenkova, 2008).

In this paper, we propose a new aggregation task suited for text-to-text generation. The goal of the task is to aggregate content into sentence-sized units by taking sentence fragments (i.e., meaningful bits of text) as input, and outputting clusters of these fragments. Figure 1 shows an example of such a clustering. It can be viewed as a preceding step for sentence fusion or other text-to-text generation methods.

Inspired by surface realization tasks (Belz et al., 2011), we use pre-existing sentences as a source of data in order to cheaply and automatically generate datasets with different granularities. We conduct a user study to confirm the validity of these datasets and design two gold-standard clusterings. We use these gold standards to define two methods of evaluation, and introduce two baselines for the task.

Then, we propose a simple clustering model for this task based on logistic regression and hierarchical clustering. All variants of the model are shown to outperform both baselines on our datasets, but an analysis of the results identifies shortcomings in the clustering that could be overcome in future models.

2 Related work

Microplanning, or sentence planning, encompasses the tasks of natural language generation (NLG) that occur after the general organization of arguments and before the syntactic realization of phrases (Hovy

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1061–1070, Osaka, Japan, December 11-17 2016.

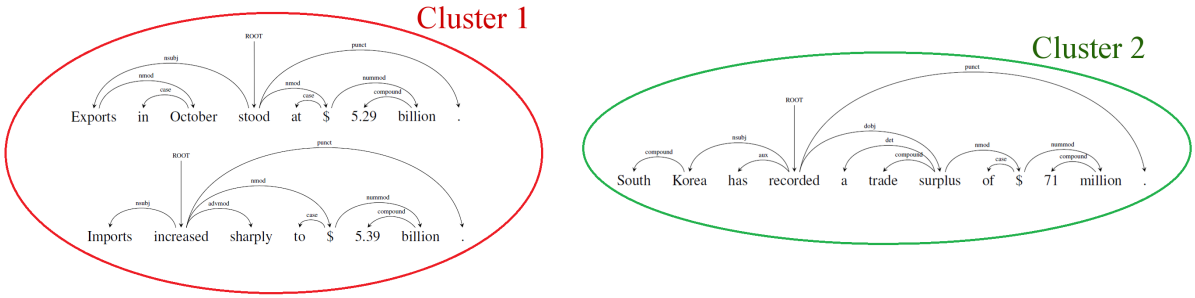


Figure 1: An example of clustering 3 sentence fragments into 2 clusters

and Wanner, 1996). Traditionally, a microplanner uses abstract domain-specific representations to produce linguistic representations. These representations are often used in domain-specific or user-specific generation systems (Mellish et al., 1998; Langkilde, 2000; Walker et al., 2002; Stent et al., 2002) and need to be adapted when changing domains (Stent et al., 2004; Walker et al., 2007). Previous work considered sentence planning to be a rigid succession of three distinct tasks: lexical choice, aggregation and referring expression generation (Reiter et al., 2000). However, more recent approaches have integrated them into a joint system (Angeli et al., 2010; Konstas and Lapata, 2013; Kondadadi et al., 2013).

Microplanning has not been considered relevant as an independent step in most recent work on text-to-text generation, because many microplanning decisions can be directly derived from the input text. For example, sentence fusion and sentence compression systems rely directly on surface text and make assumptions about the similarity of input sentences with limited rewording. Sentence compression, which consists of taking one sentence and removing redundant parts has been heavily studied (Knight and Marcu, 2000; McDonald, 2006; Galley and McKeown, 2007; Cohn and Lapata, 2008; Clarke and Lapata, 2008). Similarly, a large body of work exists in sentence fusion (Barzilay et al., 1999; Barzilay and McKeown, 2005; Marsi and Krahmer, 2005; Filippova, 2010; Thadani and McKeown, 2013).

However, relying so heavily on the input text has limited the scope of text-to-text systems. In terms of aggregation, most current systems in sentence fusion focus on fusing very similar input sentences, as determined by lexical overlap, though several recent approaches expand fusion to more disparate ones (Elsner and Santhanam, 2011; Cheung and Penn, 2014). Our work in this paper can be seen as a systematic investigation to determine what content is semantically compatible at a sentence-level, in order to generate inputs to such systems.

We focus on predicting in this paper on *what* content should be expressed in the same sentence. Other work has examined the issue of *how* the content should be expressed syntactically, such as by applying hand-crafted rules (Pan and Shaw, 2004). White and Howcroft (2015), for example, show that rules for aggregation for clause combination can automatically be learned.

3 Clustering task

3.1 Task definition

The task consists of clustering sentence fragments (see next subsection) into clusters of fragments that can be merged into a single sentence. Figure 1 shows an example of clustering. In this instance, cluster 1 fragments can be combined into the sentence "*Exports in October stood at $5.29 billion while imports increased sharply to $5.39 billion.*". This particular resulting sentence is an example and others can be generated from the same cluster.

The structure of our task can be defined as follows. Given a set of N sentence fragments $\mathbf{x} = \{x_1, x_2, ..., x_N\}$, we compute $\mathbf{y} = \{y_1, y_2, ..., y_k\}$ with $y_i \subseteq \mathbf{x}$, a partitioning of $\mathbf{x}$ into k clusters. The performance of the clustering $\mathbf{y}$ can then be measured by comparing it to a partitioning $\mathbf{c} = \{c_1, c_2, ..., c_m\}$ of $\mathbf{x}$ into m gold-standard clusters (see Section 5.2).

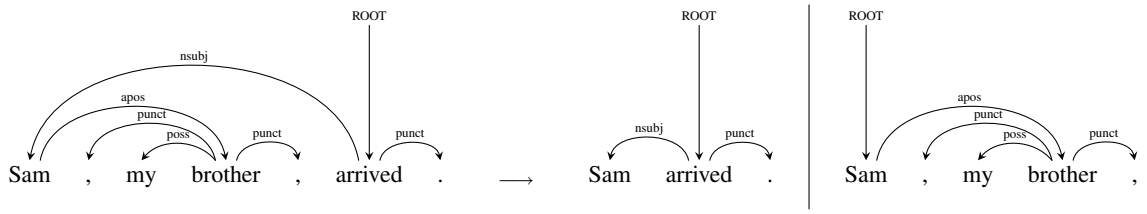


Figure 2: Example of two fragments (right) being extracted from one sentence (left)

3.2 Sentence fragment definition

Sentence fragments are defined as parts of sentences that are semantically consistent and that could be considered as grammatical sentences on their own provided small adjustments such as modifying determiners or punctuation. Sentence fragments are therefore very close to clauses, and a sentence in a text can contain one or multiple fragments. For example, the sentence *"Barack Obama, president of the United States, was born in Hawaii"* contains two fragments: *"Barack Obama (is) president of the United States"* and *"Barack Obama was born in Hawaii"*.

Fragments, as opposed to clauses, can have different levels of granularity while retaining their defining properties. For instance, the sentence *"Bell, based in Los Angeles, makes and distributes electronic, computer and building products"* can be split into two fragments: *"Bell (is) based in Los Angeles"* and *Bell makes and distributes electronic, computer and building products*. The latter fragment could then be further divided into *"Bell makes electronic, computer and building products"* and *"Bell distributes electronic, computer and building products"*. Similarly, these two fragments could then be divided into three subfragments each by removing the conjunction *"and"*.

4 Dataset generation

4.1 Splitting sentences

Sentence fragments are extracted by splitting corpus sentences on specific dependencies. The subtree resulting from such a dependency is extracted and the head word (or compound words) are copied to the new fragment. Figure 2 shows an example of extraction[1].

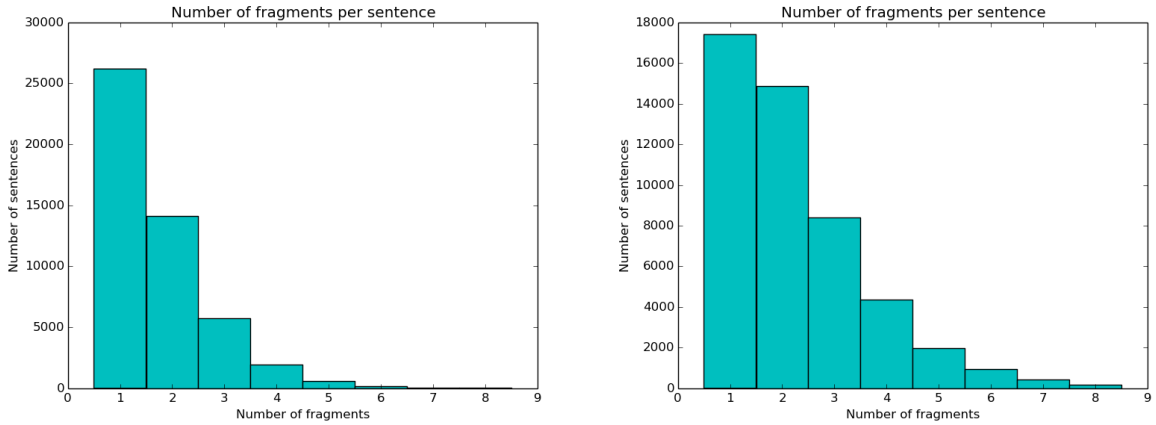
Several types of dependencies can be used to extract such fragments: appositional modifiers (e.g. *"Sam, my brother, arrived"*), adjectival and verbal modifiers (e.g. *"Pierre Vinken, 61 years old,..."*) and relative clauses (e.g. *"South Korea's economic boom, which began in 1986,..."*). To ensure that adjectival and verbal modifiers could be considered as a sentence on their own once extracted, only the subtrees with size at least four are extracted. These rules were designed to make fragments similar to propositions and ensure that they can be interpreted as sentences.

Previous examples show that conjunctions can also be used to extract fragments (see Section3.2). However, extracting conjunctions involves a different splitting mechanism: a fragment is not removed from the original sentence but the sentence is rather copied in two separate instances corresponding to each element from the conjunction (e.g. *"they either ski or snowboard"* is split into *they ski* and *they snowboard*). This different mechanism and the imperfections in the dependency trees make splitting on conjunctions a complex task and can lead to fragments that are not consistent. As a result, we generate two datasets of different granularities from the original corpus: the first one does not involve any action on conjunctions while the second one contains fragments extracted from conjunctions. These datasets will be referred to as *KeepConj* and *SplitConj*, respectively.

4.2 Creation of the datasets

We used the Penn Treebank Wall Street Journal corpus (Marcus et al., 1993). Manually-annotated constituent trees were converted into dependency trees using the the Stanford CoreNLP framework (Manning

[1]Technically, in this example, the first extracted fragment should be post-processed from "*Sam, my brother,*" to "*Sam is my brother*" (and the corresponding dependency tree) to represent a new individual sentence. However, all features used in our models (see Section 6.1) are indifferent to these modifications so the extracted fragments are not modified.



(a) KeepConj

(b) SplitConj

Figure 3: Distribution of the number of fragments per sentence

et al., 2014). Two datasets were created by extracting fragments from all sentences in all documents following the process described in the previous subsection. Each dataset was divided into a training set containing 2000 documents, a development set containing 254 documents and a test set containing 200 documents.

From the 48810 original sentences in the corpus, 84051 fragments were extracted into the *KeepConj* dataset and 111593 fragments were extracted into the *SplitConj* dataset.

Figure 3 shows that the generated datasets contain a lot of singletons, i.e. sentences from which one fragment has been extracted. The less fine-grained the dataset is, the more pronounced the imbalance. In our later experiments, we will define a singleton-only baseline which will form a non-trivial baseline to beat due to this distribution (see Section 7.1).

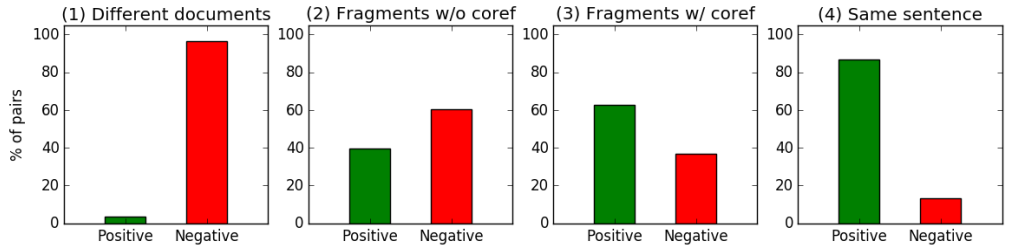
5 Establishing the Gold Standard

5.1 Validation by user study

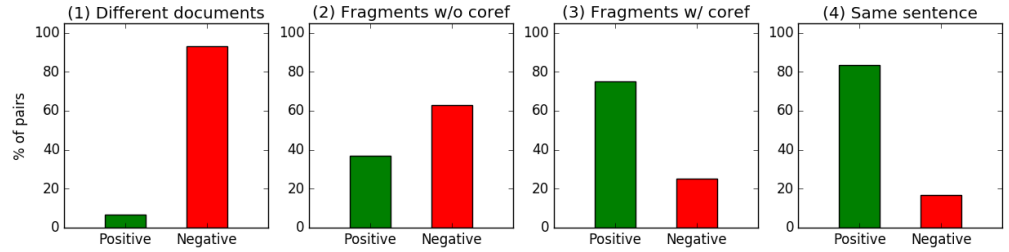
In order to validate our automatic dataset extraction process (i.e. ensuring that fragments extracted from a given sentence could be combined back) and establish a gold-standard clustering to compare the results of the task with, we conducted a user study. Participants, all native English speakers, were presented with pairs of sentence fragments and were asked to decide whether the fragments were mergeable (positive pair) or not (negative pair). Mergeable fragments are defined as fragments that can be combined into a single consistent and grammatical sentence without having a significant impact on their structure. Participants were shown detailed instructions describing the task, the allowed mechanisms for merging fragments as well as examples of positive and negative cases. Each participant was presented with 60 pairs of fragments: 10 pairs originated from the same original sentence, 10 pairs originated from different documents, 20 pairs originated from different sentences within a same paragraph and 20 pairs originated from different paragraphs within a same document. Three separate annotations were recorded for each form of 60 pairs of fragments. A total of 18 participants annotated 360 pairs, 180 pairs per dataset.

The results of the study are shown in Figure 4. The study confirms the validity of the automatic dataset generation process: fragments coming from the same original sentence are mostly considered mergeable by human annotators. A qualitative analysis of the cases when such pairs were labelled as negative show that they either come from an error in the fragment extraction process (that can come from the conversion from constituent trees to dependency trees) or a situation when three or more fragments were extracted from a sentence.

Inter-agreement between participants was measured using Fleiss' kappa (Fleiss, 1971). Agreement is high when fragments come from the same sentence or when they come from different documents. These categories correspond to "easy" cases, respectively positive and negative most of the time. On the other



(a) KeepConj



(b) SplitConj

Figure 4: Results of the user study. Pairs are divided into four categories by source: (1) fragments that come from different documents; (2) fragments that come from different sentences within a same document and that do not contain coreferent entity mentions; (3) fragments that come from different sentences within a same document and that contain coreferent entity mentions; (4) fragments that come from the same sentence.

Pairs considered	Fleiss' κ
All pairs	0.401
(1) Different documents	0.641
(2) Fragments w/o coref	0.261
(3) Fragments w/ coref	0.252
(4) Same sentence	0.731

(a) KeepConj

Pairs considered	Fleiss' κ
All pairs	0.461
(1) Different documents	0.636
(2) Fragments w/o coref	0.358
(3) Fragments w/ coref	0.317
(4) Same sentence	0.727

(b) SplitConj

Table 1: Inter-agreement among human annotators. Pairs are divided into four categories by source, as seen in Figure 4.

hand, agreement in other instances is lower, suggesting that deciding if two fragments are mergeable is a difficult task. However, results show that fragments that contain coreferent entity mentions are more likely to be mergeable (see Figure 4). This result is consistent across datasets.

The study shows that although fragments extracted from the same sentence are indeed mergeable, a significant number of positive cases exist where fragments originate from different sentences within a given document. Most of these cases correspond to fragments that contain coreferent entity mentions, yet not all of these fragments are mergeable. A chi-squared test showed that for both datasets, the proportion of positive pairs among fragments containing coreferent entity mentions (see category (3) in Figure 4) is significantly greater ($p < 0.05$) than the proportion of positive pairs among fragments that originated from the same document but do not contain such mentions (see category (2) in Figure 4). Based on these observations, we will define two different gold-standard clusterings: one optimistic clustering and one pessimistic clustering.

5.2 Evaluation measures

Based on the results from the user study, we define two gold-standards that act as lower and upper bounds on the clustering performance. The first one consists of using the original sentences from the corpus as gold-standard. This means that fragments should be clustered together if and only if they were extracted

from the same sentence. The second gold-standard consists of using fragments that contain coreferent entity mentions in addition to the original sentences. This means that fragments should be clustered together if and only if they were extracted from the same sentence or if they that contain coreferent entity mentions. The first gold-standard evaluation acts as a lower bound on the performance: it restricts the gold-standard to one specific clustering corresponding to the original corpus. On the other hand, the second gold-standard evaluation acts as an upper-bound on the performance: clustering of fragments that contain coreferent entity mentions is always considered to be positive while the user study showed that this is an oversimplification.

To evaluate the performance of a system clustering against one of the gold-standards, the measures of purity and collocation are used. Consider items $x_1, ..., x_N$ to be clustered, $y_1, ..., y_k$ the k system clusters outputted by the algorithm and $c_1, ..., c_m$ the m gold-standard clusters. Purity measures the proportion of items that belong to the same gold-standard cluster in a system cluster, while collocation measures the proportion of items that belong to the same system cluster in a gold-standard cluster:

$$\textit{Purity} = \frac{1}{N}\sum_{j=1}^{k}\max_{i\in[\![1,m]\!]}|y_j \cap c_i| \qquad \textit{Collocation} = \frac{1}{N}\sum_{j=1}^{m}\max_{i\in[\![1,k]\!]}|c_j \cap y_i|$$

When all items x_i are clustered by the system as singletons we have $\textit{Purity} = 1$, while we have $\textit{Collocation} = 1$ when all items x_i are clustered by the system as one cluster. To evaluate the trade-off between purity (that favours a large number of clusters) and collocation (that favours a small number of clusters), we use the $F1$ measure: $\textit{F1} = \frac{2\cdot\textit{Purity}\cdot\textit{Collocation}}{\textit{Purity}+\textit{Collocation}}$.

The mergeability relation in the second gold-standard is not considered to be transitive (i.e. if fragments A and B contain coreferent entity mentions and fragment C originates from the same sentence as fragment B, then fragments A and C should not be clustered together without fragment B); therefore, the second gold-standard only has an impact on purity, and not collocation. Specifically, when computing purity, a gold-standard cluster is the union of the fragments originating from a corpus sentence and all fragments that contain coreferent entity mentions with said sentence. However, when computing collocation, a gold-standard cluster is only the union of the fragments originating from a corpus sentence. This ensures that clustering the fragments back into the original corpus has a collocation score of 1.

We will refer to the metrics associated with the first gold-standard as "$\textit{Metric}^{\textit{LOW}}$" (e.g. $\textit{Purity}^{\textit{LOW}}$) and the metrics associated with the second gold-standard as "$\textit{Metric}^{\textit{UP}}$" (e.g. $\textit{Purity}^{\textit{UP}}$).

6 Hierarchical clustering model

We design a model that uses a logistic regression classifier to learn a similarity measure between two fragments. Based on this similarity function, we perform different methods of hierarchical clustering.

6.1 Learning of the similarity measure

Given a pair of fragments, we train a logistic regression classifier that returns the probability that the pair is positive, which we interpret as the similarity ϕ between the two fragments.

Given two items, x_i and x_j, we compute their joint feature representation $\Psi(x_i, x_j)$. We used a total of 27 pairwise features, detailed in Table 2.

For each document in the training set, each possible positive pair of fragments, i.e. its joint feature representation, is fed to the classifier with a positive label and a number of negative pairs less or equal than the number of positive pairs is fed to the classifier with a negative label. The number of negative cases considered per document is limited to account for the high number of singletons in the corpus and reduce training time.

After training, the similarity function ϕ is defined as $\phi(x_i, x_j) = \sigma_w(\Psi(x_i, x_j))$ where σ_w is the logit function learned by the classifier.

6.2 Variants of the model

Given a set of items $x_1, ..., x_N$ to be clustered, we compute the $N \times N$ similarity matrix K such that $K_{ij} = \phi(x_i, x_j)$. Each line (or column) of K corresponds to a cluster. We then perform bottom-up

Category	Example of feature	Value in example (see Figure 2)
Words in common	number of words in common	1 ("*Sam*")
	weight (*tf-idf*) of words in common	0
	weight (*tf-idf*) of nouns in common	0
	share of common words in longer fragment	0.33
Verb presence	share of fragments containing a verb other than "be"	0.5
Length	difference in length between fragments	1
Semantic roots	share of roots that are a verb other than "be"	0.5
	same root	0

Table 2: Examples of pairwise features used to learn the similarity function ϕ between the fragments "Sam arrived" and "Sam, my brother"

hierarchical clustering on K: we start with each item as a singleton cluster and successively merge clusters that have the highest similarity (clusters corresponding to the line and column of the highest value in K). We stop when a threshold similarity value has been reached. Values in K are updated after each step using one of these methods:

- **Single** linkage: the similarity between two clusters is the maximum similarity between two elements of these clusters
- **Complete** linkage: the similarity between two clusters is the minimum similarity between two elements of these clusters
- **Average** linkage: the similarity between two clusters is the average similarity between two elements of these clusters

In addition to these static methods, we also implement a dynamic **iterative** clustering method: after each step, we recompute new similarity scores by concatenating the fragments in each cluster.

7 Experiments and analysis

7.1 Baselines

We implement two baselines to compare the results of our model with. These baselines rely on specific aspects of the corpus and the fragment extraction process.
SINGLETON: Due to the large number of sentences containing only one fragment in the original corpus, the first implemented baseline consists of clustering each fragment into a separate cluster. This singleton baseline performs particularly well on coarse-grained dataset *KeepConj*.
SAMEROOT: We implemented a second baseline which consists of clustering together fragments that have the same semantic root (node(s) in the dependency tree that have no governor). This "same root" baseline performs well on both datasets.

7.2 Results

Table 3 shows the comparisons between the performances of the baselines and four variants of the hierarchical clustering model. The stopping threshold for each model has been tuned on the development set.

Results show that all variants significantly[2] outperform the baselines on both datasets ($p < 0.01$). The hierarchical clustering method has an inconsistent impact across datasets. The dynamic iterative aggregation performs best on dataset *KeepConj* ($p < 0.05$) while the average-link and complete-link aggregations perform better than other methods on dataset *SplitConj* ($p < 0.05$).

[2]All significance tests were performed using a two-tailed paired sample t-test on F1-scores at the document level.

Model	$Purity^{LOW}$	$Collocation^{LOW}$	$F1^{LOW}$	$Purity^{UP}$	$Collocation^{UP}$	$F1^{UP}$
Singleton baseline	**1.00**	0.588	0.741	**1.00**	0.588	0.741
"Same root" baseline	0.829	0.651	0.729	0.876	0.651	0.747
Single-link	0.900	0.736	0.801	0.961	0.736	0.833
Complete-link	0.895	0.747	0.814	0.953	0.747	0.837
Average-link	0.893	0.754	0.818	0.956	0.754	0.843
Iterative	0.881	**0.778**	**0.827**	0.948	**0.778**	**0.855**

(a) KeepConj

Model	$Purity^{LOW}$	$Collocation^{LOW}$	$F1^{LOW}$	$Purity^{UP}$	$Collocation^{UP}$	$F1^{UP}$
Singleton baseline	**1.00**	0.446	0.617	**1.00**	0.446	0.617
"Same root" baseline	0.841	0.683	0.754	0.884	0.683	0.770
Single-link	0.868	**0.745**	0.802	0.940	**0.745**	0.831
Complete-link	0.920	0.723	0.810	0.970	0.723	0.829
Average-link	0.909	0.733	**0.811**	0.966	0.733	**0.833**
Iterative	0.894	0.731	0.804	0.961	0.731	0.830

(b) SplitConj

Table 3: Evaluation results of the baselines and variants of the model against both gold-standard clusterings

7.3 Analysis

Results show that our model consistently favours purity over collocation, generating a higher number of clusters compared to the gold-standards. Examples in this section were generated by the *iterative* hierarchical clustering model on dataset *KeepConj* but are representative of the behaviour of all variants of the model.

This higher number is mainly due to the fact that our model fails to cluster together some long fragments that share a small number of common words. For example, the sentence "*Influential members of the House Ways and Means Committee introduced legislation that would restrict how the new savings-and-loan bailout agency can raise capital, creating another potential obstacle to the government's sale of sick thrifts.*" is divided during extraction into the fragments "*Influential members of the House Ways and Means Committee introduced legislation*" and "*legislation would restrict how the new savings-and-loan bailout agency can raise capital, creating another potential obstacle to the government's sale of sick thrifts*". These fragments are then rendered as singletons by the model.

On the other hand, shorter fragments are more easily clustered. For instance, fragments "*Another $20 billion would be raised through Treasury bonds*" and "*Treasury bonds pay lower interest rates*" are clustered back into the sentence "*Another $20 billion would be raised through Treasury bonds, which pay lower interest rates.*".

Our model clusters also show instances of coreferent fragments clustering, as evidenced by the significant increase in purity when the second gold-standard is used ($Purity^{UP}$). Fragments "*thrifts are sold, until the assets can be sold separately*" and "*the assets the RTC holds*" are clustered together. While they originate from different sentences, the fragments contain coreferent entity mentions ("*the assets*") and can be merged into the sentence "*Thrifts are sold, until the assets the RTC holds can be sold separately.*".

8 Conclusion

This paper introduced an aggregation task suited for text-to-text generation based on clustering sentence fragments. We presented evaluation metrics that were designed using human-annotated results. These results show that most mergeable pairs of fragments either originate from the same original sentence or contain coreferent entity mentions. We designed a hierarchical clustering model that uses a logistic regression classifier to learn a similarity measure between fragments. Its results were shown to outper-

form simple baselines and can be used as a reference for the task. Given that the task relies on shallow semantic analysis and the evaluation metrics only require coreference resolution, it is highly adaptable. Generating datasets from multi-document corpora can be the object of future work. Other clustering models, using neural networks to learn a similarity measure for instance, can also be considered. Integration of other aspects that were considered outside of the scope of our task, for example discourse relations analysis or information aggregation is another possible future avenue.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512. Association for Computational Linguistics.

Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Regina Barzilay, Kathleen R McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 550–557. Association for Computational Linguistics.

Anja Belz, Michael White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the 13th European workshop on natural language generation*, pages 217–226. Association for Computational Linguistics.

Jackie Chi Kit Cheung and Gerald Penn. 2014. Unsupervised sentence enhancement for automatic summarization. In *EMNLP*, pages 775–786.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 137–144. Association for Computational Linguistics.

Micha Elsner and Deepak Santhanam. 2011. Learning to fuse disparate sentences. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 54–63. Association for Computational Linguistics.

Katja Filippova. 2010. Multi-sentence compression: finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 322–330. Association for Computational Linguistics.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378–382.

Michel Galley and Kathleen McKeown. 2007. Lexicalized markov grammars for sentence compression. In *HLT-NAACL*, pages 180–187.

Eduard H Hovy and Leo Wanner. 1996. Managing sentence planning requirements. In *Proceedings, ECAI-96 Workshop on Gaps and Bridges: New Directions in Planning and Natural Language Generation*, pages 53–58.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization-step one: Sentence compression. *AAAI/IAAI*, 2000:703–710.

Ravi Kondadadi, Blake Howald, and Frank Schilder. 2013. A statistical nlg framework for aggregated planning and realization. In *Proceedings of the 51st annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 1406–1415. Association for Computational Linguistics.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *J. Artif. Intell. Res.(JAIR)*, 48:305–346.

Irene Langkilde. 2000. Forest-based statistical sentence generation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 170–177. Association for Computational Linguistics.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Erwin Marsi and Emiel Krahmer. 2005. Explorations in sentence fusion. In *Proceedings of the European Workshop on Natural Language Generation*, pages 109–117. Citeseer.

Ryan T McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Chris Mellish, Alistair Knott, Jon Oberlander, and Mick ODonnell. 1998. Experiments using stochastic search for text planning. In *Proceedings of International Conference on Natural Language Generation*, pages 97–108.

Ani Nenkova. 2008. Entity-driven rewrite for multi-document summarization. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, pages 118–125.

Shimei Pan and James Shaw. 2004. Segue: A hybrid case-based surface natural language generator. In *Natural Language Generation*, pages 130–140. Springer.

Ehud Reiter, Robert Dale, and Zhiwei Feng. 2000. *Building natural language generation systems*, volume 33. MIT Press.

Amanda Stent, Marilyn A Walker, Steve Whittaker, and Preetam Maloor. 2002. User-tailored generation for spoken dialogue: an experiment. In *INTERSPEECH*.

Amanda Stent, Rashmi Prasad, and Marilyn Walker. 2004. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Kapil Thadani and Kathleen McKeown. 2013. Supervised sentence fusion with single-stage inference. In *IJCNLP*, pages 1410–1418.

Marilyn A Walker, Owen C Rambow, and Monica Rogati. 2002. Training a sentence planner for spoken dialogue using boosting. *Computer Speech & Language*, 16(3):409–433.

Marilyn A Walker, Amanda Stent, François Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research*, 30:413–456.

Michael White and David M Howcroft. 2015. Inducing clause-combining rules: A case study with the sparky restaurant corpus. *ENLG 2015*, pages 28–37.

Sequential Clustering and Contextual Importance Measures for Incremental Update Summarization

Markus Zopf, Eneldo Loza Mencía, and Johannes Fürnkranz
Research Training Group AIPHES / Knowledge Engineering Group
Department of Computer Science, Technische Universität Darmstadt
Hochschulstraße 10, 64289 Darmstadt, Germany
{zopf@aiphes,eneldo@ke,juffi@ke}.tu-darmstadt.de

Abstract

Unexpected events such as accidents, natural disasters and terrorist attacks represent an information situation where it is crucial to give users access to important and non-redundant information as early as possible. Previous work uses either a fast but inaccurate pipeline approach or a precise but slow clustering approach. Instead, we propose to use sequential clustering for grouping information so that we are able to publish sentences at each time step. By doing so, we combine the best of both clustering and pipeline approaches and create a fast and precise real-time system. Experiments on the TREC Temporal Summarization 2015 shared task dataset show that our system achieves better results compared to the state-of-the-art.

1 Introduction

Events such as accidents, natural disasters and terrorist attacks provide an important and challenging information problem. Shortly after such an event has occurred, the information situation is usually unclear. Initially, only vague information about the event may become available, for example that an earthquake has occurred, but details such as magnitude, epicenter, and whether a tsunami has to be expected are not known at this early point in time. Such information becomes only available as the event develops over time. In such situations, it is crucial for responders, crisis management organizations, and victims to get information as soon as possible. However, it is impossible for humans to monitor *vast amount of textual information* contained in sources such as news articles, tweets, social media posts, forum discussions, and live blogs. In incremental update summarization (IUS) (McCreadie et al., 2014b) the *detection of important information in a timely manner* is a major challenge. Since the information sources are also *highly redundant*, detecting and filtering redundancy is a second important challenge in IUS.

In the past, two types of systems were used for IUC: pipeline systems (McCreadie et al., 2014a; McCreadie et al., 2015) and ordinary clustering systems (Kedzie et al., 2014; Zhao et al., 2014; Yingzhe Yao and Fan, 2015; McCreadie et al., 2015). Pipeline systems process document or sentence and decide about importance of information and novelty immediately and achieve therefore a good timeliness. However, they publish a large amount of unimportant information (low precision) or miss important information (low recall) since they do not make use of the redundancy of information as signal for importance, which is a usually a very good feature in newswire documents (Nenkova et al., 2006). Traditional clustering systems on the other hand are able to exploit redundancy and therefore produce better summaries according to precision and recall. Since they collect documents (for example all documents within one hour) before they decide whether or not to publish information, timeliness is a major issue of these approaches.

Instead of choosing between one of these architectures, we propose to use sequential clustering (Section 3.1) for incremental update summarization. With a sequential clustering, we are able to combine best of both worlds: clustering for redundancy avoidance to achieve high precision and recall and the ability to publish information at every time step to achieve a good timeliness. Hence, sequential clustering seems to be a natural fit for the task of IUS. In the sequential clustering, we jointly identify importance

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1071–1082, Osaka, Japan, December 11-17 2016.

and redundancy by using contextual importance measures (Section 3.2). They are used to assign utility scores to clusters, sentences, and tokens depending on already selected information. An additional redundancy avoidance methodology, as used by the other systems, becomes therefore obsolete.

We show in an experimental evaluation by using data of the TREC 2015 shared task on temporal summarization (Aslam et al., 2015) that our system `SeqCluSum` is able to outperform state-of-the-art. A substantial improvement is achieved even though the computed scores can only be considered a lower bound on the performance since the original competition used a manual and individual evaluation which cannot be reconstructed without loss of precision.

2 Related Work

Traditional extractive multi-document summarization approaches (Nenkova and McKeown, 2011) extract unmodified sentences from source documents to produce a summary. Graph-based approaches (Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Parveen and Strube, 2015) represent source documents as graph and use algorithms such as HITS (Kleinberg, 1999) and PageRank (Brin and Page, 2012) to find important information. Centroid-based summarization (Carbonell and Goldstein, 1998; Radev et al., 2000) systems estimate sentences importance by computing their centrality in the source documents. Similarly to these systems, our approach extracts unmodified sentences and uses centrality as a signal for importance. The systems above perform a retrospective summarization, meaning that the systems analyze all source documents at once independently from their publication date. In IUS however, this is not possible, since important information has to be published as soon as possible. Furthermore, the mentioned systems create summaries of fixed lengths. In IUS we observe a situation where it is not clear in advance how long a summary has to be for a proper summarization of an event. Standard extractive summarization systems are therefore not suited for IUS.

In update summarization (Dang and Owczarzak, 2008), a summary is presented to the systems in addition to source documents which contain new information. The task is to summarize the source documents without repeating information which is already contained in the summary. Since this task is very similar to extractive summarization, methods, which are successfully applied to extractive, were also applied to update summarization.

Research in incremental update summarization (McCreadie et al., 2014b) is strongly influenced by the TREC Temporal Summarization (TREC-TS) shared task (Guo et al., 2013). The TREC-TS 2014 shared task (Aslam et al., 2014) provided a high-volume, pre-filtered version of the TREC KBA 2014 dataset.[1] The best performing run (Kedzie et al., 2014) in the challenge according to the official target metric uses affinity propagation clustering. Zhao et al. (2014), best performing system according to expected latency gain, applies a query expansion and information retrieval step in addition to a k-means clustering and sentence selection step. McCreadie et al. (2014a) proposes a real-time summarization system which achieved best comprehensiveness scores. They use a processing pipeline to filter out irrelevant documents, to classify sentences according to their relevance, and to filter out redundant sentences.

In the TREC-TS 2015 shared task (Aslam et al., 2015) systems competing in the "summarization only" subtask 3 were provided with a low-volume stream of documents.[2] The best system according to the official evaluation score in 2015 was proposed by Raza et al. (2015). The authors use BM25 for sentence scoring and redundancy avoidance based on cosine similarity values. McCreadie et al. (2015) focuses on entity importance and entity-entity interaction to identify important entities in an event. Kedzie et al. (2015) processes documents in hourly batches and combine salience prediction for sentences with affinity propagation clustering.

Our approach aims to combine the timeliness of a pipeline approach such as McCreadie et al. (2014a) with the importance detection strategy of a clustering approach such as Zhao et al. (2014).

[1] `http://trec-kba.org/kba-stream-corpus-2014.shtml`

[2] `http://dcs.gla.ac.uk/~richardm/TREC-TS-2015RelOnly.aws.list`

3 Approach

In this section, we propose sequential clustering in combination with contextual importance measures for incremental update summarization. The algorithm, `SeqCluSum`, consists of two subsystems, which are alternatingly applied to the documents in the stream. In the first subsystem, a sequential clustering algorithm clusters all sentences in a document according to similarity measures (Section 3.1). The second subsystem, a contextual importance measure, estimates scores for clusters (Section 3.2). Both subsystems work hand in hand in order to satisfy the three objectives to publish (i) *important information* while (ii) *avoiding redundancy* in a (iii) *timely manner*.

To detect important information, we make use of the property that important information occurs frequently in newswire data (Nenkova et al., 2006). We define the utility of a cluster as the sum of the utilities of the contained sentences the size of a cluster is an important factor for the cluster importance. By this definition, larger clusters tend to obtain higher utility scores than smaller clusters. Furthermore, we use normalized temporal TF-IDF scores in the contextual importance measure to estimate the utility of a sentence (cf. Section 3.2). Frequent terms in the input documents tend to obtain higher scores than infrequent terms. A cluster containing sentences with frequent terms will therefore obtain higher scores than a cluster with infrequent terms. We prevent publishing redundant information by allowing at most one update per cluster. This is reasonable since one cluster is assumed to represent one particular information which is not represented by another cluster. If a cluster is selected to publish a sentence, this cluster will be marked as *published* and will not be considered for publishing a sentence again. Since the system processes each document sequentially, it is able to publish sentences after each processing step which enables it to satisfy the third objective of timeliness. The pseudo-code of the complete system is shown in Algorithm 1. Line references are given in parenthesis with respect to this pseudo-code.

Algorithm 1 Sequential Clustering and Contextual Importance Measuring

```
   documents = ordered list of documents, clusters ← ∅, updates ← ∅
 1: for each document ∈ documents do
 2:     document ← preprocess (document)                ▷ remove boilerplate content and stem words
 3:     for each sentence ∈ document do                  ▷ add each new sentences to a clusters
 4:         nearestCluster = arg max_{c∈clusters} similarity(sentence, c)
 5:         if clusters = ∅ or (similarity(sentence, nearestCluster) > Θ) then
 6:             clusters ← clusters ∪ {{sentence}};      ▷ create a new cluster
 7:         else
 8:             nearestCluster ← nearestCluster ∪ sentence   ▷ add to nearest cluster
 9:         end if
10:     end for
11:     for each cluster ∈ clusters do                   ▷ evaluate clusters and generate updates
12:         if cluster is not published and w(cluster) > μ then
13:             bestSentence = arg max_{s∈cluster} v(s)  ▷ find best sentence in cluster
14:             updates ← updates ∪ (document.timestamp, bestSentence)
15:             set cluster to published
16:         end if
17:     end for
18: end for
19: return updates
```

Before we apply our system for incremental update summarization to the document stream, we apply three preprocessing steps. We remove duplicate occurrences of web pages, utilize a boilerplate removal, and stem all words in the source documents with the well-known Porter-stemming algorithm (Porter, 1980).

3.1 Sequential Clustering

For the first part of our proposed incremental update summarizer, we adapt the sequential clustering algorithm (Theodoridis and Koutroumbas, 2009) to IUS. The algorithm iterates over all sentences in the currently processed document (line 3) and searches for the nearest existing cluster for each sentence using a similarity measure (line 5), which is described below. If the similarity to all existing clusters is lower than a fixed threshold Θ or no cluster has been created yet, a new cluster is created and the

sentence is added to the new cluster (line 6). Otherwise, the sentence is added to the nearest existing cluster (line 8).

The similarity measure reduces the similarity between a sentence s and a cluster c to the similarity between the sentence s and the first sentence of the cluster c. This has several advantages in comparison to considering all sentences in the cluster. First, the center of the cluster is fixed when we compare only with the first sentence of the cluster. This prevents the cluster from a topic drift. Adding more and more sentences and also using the newly added sentences in the similarity calculations could instead change the initial notion of the cluster significantly since the center of the cluster could move. We observed that this can be a serious issue which leads to a poor clustering. Second, the approach is computationally efficient in comparison to an approach where we would compute the similarity by considering all sentences in a particular cluster. The number of maximal comparisons is reduced from the number of all sentences in a topic to the number of clusters created for a topic. Third, it emphasizes the notion of a cluster as a set of sentences, each of which contains the same information. The system chose the first sentences of each cluster to be its representative since each first sentence has a special role. This special role derives from the fact that the first sentence of each cluster was the reason why the cluster was created. The sentences did not fit to another cluster and were the reason for creating its own, new cluster.

The actual similarity measure (line 4+5) is based on Cosine similarity based on TF-IDF vectors (Salton and McGill, 1986). To calculate the TF-IDF vectors, we use a background corpus B created from 100,000 randomly sampled English Wikipedia articles in addition to all Wikipedia articles which are longer than 10,000 characters (5,794 documents). We use DKPro Similarity (Bär et al., 2013) to calculate the Cosine similarity. A detailed analysis of similarity measures is given in Section 5.2.

3.2 Measuring Importance

After the sentences in a document are clustered, the system evaluates the current cluster landscape to detect important information. We introduce the utility functions u, v, and w to measure the utility of tokens, sentences, and clusters, respectively. The score of a cluster c is measured with a cluster utility function w and equals to the sum of the scores of all sentences s_i in the cluster. We define $w(c) = \sum_{s_i \in c} v(s_i)$, where v is a utility function for sentences. Summing the scores of all sentences in a cluster addresses the property of the corpus that more important information is repeated more frequently in the source documents since larger clusters obtain higher utility scores (c.f. Section 1). The score of a sentence s is defined as the sum of the weights of the tokens t_i contained in sentence s. Thus, we define $v(s) = \sum_{t_i \in s} u(t_i)$, where u is a utility function for a token. In the following, we define a contextual importance measure to estimate the utility of a single token. Since we summarize an ongoing event and do not know which documents will appear later in the stream, we cannot compute TF-IDF scores over all documents. Nevertheless, we want to estimate how salient a token is relatively to the already observed documents D_τ at time τ. This will provide us with a signal about the relative relevance of the tokens very similarly to the well-known TF-IDF. To measure the importance of a token, we first define a context-free utility function u_{cf} in Equation 1. This gives us an impression on how salient a token is in a document collection D_τ. Since the document collection D_τ changes after each time step τ, the utility scores are constantly updated when new documents are processed. We define

$$u_{cf}(t, D_\tau) = \sum_{D \in D_\tau} \frac{n_D(t)}{|D|} \cdot log\left(\frac{|B|}{n_B(t)}\right) \quad (1)$$

where $n_B(t)$ denotes the number of occurrences of the token t in the background corpus B and $|B|$ refers to the total number of tokens in B. Unknown tokens, which do not appear in the background corpus (i.e. $n_B(t) = 0$), are ignored in the calculation of sentences importance. The contextual similarity measures defined in Equation 2 takes already published tokens into account when estimating the utility of a token. It captures therefore the importance of a token with respect to already published tokens. Hence, it avoids publishing sentences which are similar to already published sentences by discounting the scores for already published tokens and therefore provides redundancy avoidance implicitly. We discount the context-free values by dividing the context-free score $u_{cf}(t, D_\tau)$ by the number of occurrences of token

t in the already published updates. We define $u_c(t, D_\tau)$ in Equation 2, where $n_U(t)$ denotes the number of occurrences of token t in the current list of updates U.

$$u_c\left(t, D_\tau\right) = \frac{v_{cf}\left(t, D_\tau\right)}{1 + n_U(t)} \tag{2}$$

3.3 Publishing an Information Update

A cluster is considered as sufficiently important if a fixed threshold μ is exceeded (line 12). As a consequence, one sentence from the cluster is published (line 14). The threshold can easily be adapted on demand to produce more or less verbose summaries. If a cluster exceeds the threshold, the best sentence according to the sentence utility score v is published. The cluster is marked as *published* in this case (line 15), which means that the cluster will not be selected in the future.

4 Evaluation

In this section, we describe the evaluation of our approach. First, we provide detailed information about the used evaluation data in Section 4.1. The evaluation methodology is described in Section 4.2. We use a trace-driven simulation which simulates the course of events to evaluate our system. Results are presented in Section 6 after a detailed analysis of our system in Section 5.

4.1 Data

As evaluation dataset, we use the TREC-TS-2015RelOnly[3] dataset, which was created by the organizers of the 2015 Temporal Summarization shared task (Aslam et al., 2015). The corpus is a filtered version of the KBA Stream Corpus 2014[4], contains documents from various text genres like mainstream news articles, blogs/microblogs, and forum posts, and is divided into 21 clusters of documents. Each cluster represents one topic and has additional meta-data about the start and the end of the event as well as a query term indicating the topic of the event. Each document has sentence segmentation annotations and is guaranteed to contain at least one relevant sentence for the topic. All documents in the corpus have an associated timestamp which equals the crawling time of the document.

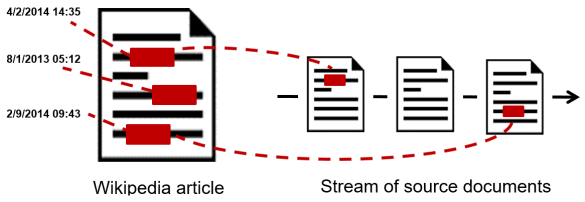


Figure 1: Illustration of the TREC-TS dataset. Information nuggets were retrieved from Wikipedia articles and annotated with timestamps at which this information became publicly available. Sentences in the source documents are matched against the extracted information nuggets.

For the evaluation in the shared task, the organizers extracted time-stamped information nuggets based on the Wikipedia article revision histories of the corresponding events. In a first step, information nuggets were extracted from the revision histories by the track organizers. The nuggets were classified with an importance score of 1, 2, or 3, and tagged with a timestamp, which represents the time when this information became publicly available. In a second step, sentences were pooled from each submission in the TREC-TS 2015 shared task. For each submission, at most 60 sentences were pooled per topic and were manually matched with the extracted information nuggets. A sentence can match multiple nuggets as well as no nugget. Figure 1 illustrates the annotation. Due to the huge annotation effort, only a small set of sentences in the corpus is labeled. This is problematic, since one cannot train, validate, or evaluate a system accurately without additional annotation effort.

[3] http://dcs.gla.ac.uk/~richardm/TREC-TS-2015RelOnly.aws.list
[4] http://trec-kba.org/kba-stream-corpus-2014.shtml

4.2 Methodology

In the task of IUS, the documents have to be processed in temporal order. The result of a system is a list of updates. When a system decides that a sentence from the stream should be published, this particular sentence is marked with the timestamp of the currently processed document according to Algorithm 2.

Algorithm 2 Incremental Update Summarization (IUS)

S = summarizing system; $documents$ = time-ordered list of documents; $updates = \emptyset$
for each $document \in documents$ **do**
  $t \leftarrow document.time$
  $S.process(document)$
  $updates_t \leftarrow S.getUpdates()$
  for each $u \in updates_t$ **do** $updates \leftarrow updates \cup \{(t, u)\}$ **end for**
end for
return $updates$

The systems are not allowed to use information from the future to make this decision. Incorporating information that only became available after the current timestamp is not allowed. One example for such an improper use of information would be the computation of TF-IDF values over the whole corpus since this would provide information about which words will become important in the future.

In our evaluation, we use the same metrics as in the TREC-TS 2015 shared task. The first metric is *normalized expected gain* $n\mathbf{EG}(\mathcal{S})$. This metric is comparable to precision since it measures whether a system publishes updates which are on-topic and novel. It will give a low score to systems which publish irrelevant or redundant information. The second metric $\mathbf{C}(\mathcal{S})$ measures the *comprehensiveness* of a summary. It is similar to recall since it measures how many of the information nuggets that could have been retrieved are covered by the summary. Systems which miss a large amount of important information will obtain a low score according to $\mathrm{C}(\mathcal{S})$. A third metric, the *expected latency* metric E[Latency], measures to which degree the information in the summary is outdated.[5] This metric reflects the requirement that systems are supposed to publish sentences as early as possible to produce the maximal benefit for the users. The later a system identifies important information, the lower its expected latency score. The final measure $\mathcal{H}$ combines the three individually measured properties precision, recall, and timeliness and served as official target measure for the task and is also our main metric to evaluate the systems. It computes the harmonic mean of a latency discounted version of $n\mathbf{EG}(\mathcal{S})$ and a latency discounted version of $\mathbf{C}(\mathcal{S})$. We refer to Aslam et al. (2015) for more detailed information about the metrics.

5 Analysis

In the following, we provide a detailed analysis of our system. We investigate the impact of three parts of our system: the boilerplate removal (Section 5.1), the used similarity measure (Section 5.2), and the clustering (Section 5.3).

5.1 Boilerplate Removal

In this section, we evaluate the boilerplate removal preprocessing step in terms of precision and recall. Although preprocessing is not part of the contribution of this paper, it has an impact on the system performance and is therefore discussed briefly. False positive errors made by the preprocessing cannot be corrected by the subsequent clustering and therefore limits the performance of the overall system irrevocably. It gives also an impression about the dataset itself which is otherwise hard to grasp. For the analysis, we use only non-duplicate documents (i.e. only the first version of each document). Since we do not have gold standard boilerplate removal data for the dataset, we use the nugget annotations instead. Table 1 provides the results of this analysis. Since both boilerplate systems work on the document level, we had to map the result of the boilerplate removal to single sentences. To do this, we used the Jaccard

[5]The name for the metric is a little counter-intuitive, since it used as a discount factor for nugget importance. A value of 1 means that the update is in-line with the Wikipedia article and results in no discount. Values smaller than 1 mean that an update was only published after it was included in the reference. Values bigger than 1 mean that a system published a nugget before it occurred in the Wikipedia article the first time.

Boilerplate system	Similarity	Thr	Max length	tp	tn	fp (critical)	fn
-	-	-	5	294,993	8,472	180	474,017
-	-	-	6	341,444	8,396	256	427,566
-	-	-	7	372,331	8,272	380	396,679
-	-	-	8	394,354	8,112	540	374,656
-	-	-	9	416,763	7,908	744	352,247
-	-	-	10	436,579	7,641	1,011	332,431
Kohlschütter et al. (2010)	Jaccard	0.6	5	684,059	4,266	4,386	84,951
Kohlschütter et al. (2010)	Jaccard	0.8	5	714,653	3,194	5,458	54,357
Kohlschütter et al. (2010)	Cosine	0.6	5	639,374	5,706	2,946	129,636
Kohlschütter et al. (2010)	Cosine	0.8	5	695,350	3,859	4,793	73,660
Habernal et al. (2016)	Jaccard	0.4	5	682,985	4,392	4,260	86,025
Habernal et al. (2016)	Jaccard	0.6	5	721,033	3,187	5,465	47,977
Habernal et al. (2016)	Cosine	0.6	5	686,671	4,340	4,312	82,339
Habernal et al. (2016)	Cosine	0.8	5	729,665	2,820	5,832	39,345

Table 1: We report the performance according to true positives (tp), true negatives (tn), false positives (fp), and false negatives (fn) of the length cutoff baseline and two boilerplate removal system. False positives (falsely pruned valuable sentences) are critical, since the pruned sentences are not processed by the following sequential clustering.

and the Cosine similarity. We tagged the sentence as boilerplate if it was less similar to the most similar sentence in the text retrieved by the boilerplate system according to a threshold and had a maximum length of 5. We observe that we can discard a lot of sentences with only a lengths criterion without too many false positives. Both boilerplate removal systems, Boilerpipe (Kohlschütter et al., 2010) and the tool used in Habernal et al. (2016), do not perform very well in combination with the two similarity measures since both generate a large number of false positives.

5.2 Similarity Measures

One key component of summarization systems are similarity measures which estimate the semantic similarity of two texts or sentences. We therefore performed experiments with different measures in the TREC-TS dataset. We evaluate the similarity measure by comparing the score of similar sentences, which contain exactly the same nuggets and for dissimilar sentences which do not share any nuggets. We define sets of sentence $ss_1, \dots ss_L$ according to the information nuggets contained in the sentences. Since each sentence can contain multiple nuggets, we first define nugget sets $ns_1, \dots, ns_M$ which represent all sets of nuggets similarly to label powersets in multi-label classification. Sentences in a particular sentence set ss_i contain exactly the same information nuggets, i.e. each sentence is contained in exactly one sentence set. For sets $ss_1, \dots ss_L$, and sentences $S = \{s_1, \dots, s_K\}$ we define the sim score of a similarity measure σ as the average similarity of all similar sentences

$$\mathrm{sim}(\sigma) = \sum_{l=1}^{L} \frac{1}{|ss_l|} \sum_{l_1,l_2=1,l_1<l_2}^{|ss_l|} \sigma(s_{l_1}, s_{l_2}) \tag{3}$$

and the dissim score of a similarity measure σ as the average similarity of all dissimilar sentence

$$\mathrm{dissim}(\sigma) = \sum_{l_1,l_2=1,l_1<l_2}^{L} \frac{1}{|ss_{l_1}| \cdot |ss_{l_2}|} \sum_{\substack{s_1 \in ss_{l_1} \\ s_2 \in ss_{l_2}}} \sigma(s_1, s_2). \tag{4}$$

An appropriate similarity measure can perform both, assigning high scores to similar sentences and low scores to dissimilar sentences. Therefore, we measure both by computing $\mathrm{diff}(\sigma) = \mathrm{sim}(\sigma) - \mathrm{dissim}(\sigma)$. We report the scores of various similarity measures in Table 2.

We observe that the TF-IDF-based Cosine similarity with $\log +1$ IDF weighting and L2 normalization performs best.[6] For details regarding the similarity measures, we refer to Bär et al. (2013). Cosine similarity based on the sum of GloVe (Pennington et al., 2014) word embeddings to represent a sentences

[6]The results provided in the table were computed for the complete dataset. Results on the splitted datasets, which were used for the evaluation of the system, confirmed the results.

		TF-IDF-based Cosine			Jaccard-n		Word embedding-based Cosine	
	random	log+1 L1	log+1 L2	binar L2	$n=1$	$n=2$	unweighted	weighted
$\mathrm{sim}(\sigma)$	0.50	0.16	0.41	0.40	0.29	0.19	0.79	0.69
$\mathrm{dissim}(\sigma)$	0.50	0.01	0.15	0.20	0.10	0.01	0.70	0.52
$\mathrm{diff}(\sigma)$	0.00	0.15	**0.26**	0.20	0.19	0.18	0.09	0.17

Table 2: Results of $\mathrm{sim}(\sigma)$ and $\mathrm{dissim}(\sigma)$ for different similarity measures.

yields the lowest results.[7] A weighted version, where we multiply the word vectors with the corresponding IDF score achieves better results than the unweighted Cosine based on the word embeddings.

5.3 Clustering

In this section, we evaluate the cluster landscape by computing cluster purity (Manning et al., 2008) and nugget concentration. Cluster purity measures how many different nugget sets ($ns_1, \ldots, ns_M$) are contained in the clusters. According to the underling idea that one cluster represents one nugget set, one cluster should only contain sentences that belong to the same information nuggets. With the definition of a cluster as a set of sentences (i.e. $c \subset S$), the cluster purity of the clusters $C = \{c_1, \ldots c_I\}$, sentence sets $ss_1, \ldots ss_L$, and K sentences (Manning et al., 2008) is defined as

$$\mathrm{purity}(C) = \frac{1}{K} \sum_{i=1}^{I} \max_l |c_i \cap ss_l| \tag{5}$$

Similarity to purity, we also define a purity purity^+ where we do not consider the sentence set which contains all sentences without any nuggets as majority class in $\max_l |c_i \cap ss_l|$. This provides a better insight into the mixture of non-empty nugget sets. In a perfect clustering, we get a score of 1, i.e. that all clusters contain only sentence which belong to a particular nugget set. Since a purity of 1 is easy to achieve with K clusters, we introduce the term nugget concentration (nc), which is defined as

$$\mathrm{nc} = \frac{1}{L} \sum_{l=1}^{L} \frac{\max_i |c_i \cap ss_l|}{|ss_l|} \tag{6}$$

Boilerplate system	Θ	# cluster	purity	purity^+	nc	nc (after bpr)
max length 8	1.0	1	0.4874	0.1779	0.5134	1.0000
max length 8	0.9	563	0.5759	0.4310	0.4780	0.8914
max length 8	0.8	800	0.6261	0.5151	0.4745	0.8791
max length 8	0.7	893	0.6764	0.6191	0.4717	0.8670
max length 8	0.6	950	0.7259	0.7019	0.4718	0.8648
Kohlschütter et al. (2010) + Cosine < 0.6	1.0	1	0.4466	0.1923	0.4729	1.0000
Kohlschütter et al. (2010) + Cosine < 0.6	0.9	229	0.5328	0.4336	0.4473	0.9038
Kohlschütter et al. (2010) + Cosine < 0.6	0.8	390	0.5784	0.5186	0.4402	0.8871
Kohlschütter et al. (2010) + Cosine < 0.6	0.7	534	0.6613	0.6316	0.4422	0.8780
Kohlschütter et al. (2010) + Cosine < 0.6	0.6	618	0.7360	0.7361	0.4404	0.8693
Habernal et al. (2016) + Cosine < 0.8	1.0	1	0.4406	0.1815	0.3853	1.0000
Habernal et al. (2016) + Cosine < 0.8	0.9	107	0.5299	0.4294	0.3671	0.9291
Habernal et al. (2016) + Cosine < 0.8	0.8	189	0.6058	0.5519	0.3675	0.9253
Habernal et al. (2016) + Cosine < 0.8	0.7	254	0.7078	0.6909	0.3663	0.9195
Habernal et al. (2016) + Cosine < 0.8	0.6	323	0.7826	0.7900	0.3558	0.9219

Table 3: Results of the cluster evaluation for difference boilerplate versions and difference values of Θ.

We use for the analysis the TF-IDF-based Cosine similarity with $log + 1$ IDF weighting and L2 normalization (according to Section 5.2). Since the boilerplate removal can have a significant impact, we report the results based on different boilerplate removal versions. In Table 3 we see that more clusters (due to a lower Θ) lead to a higher purity as expected. Nugget concentration decreases, because it becomes more likely that one nugget set is distributed across multiple clusters. If we only consider

[7] We used pre-calculated word embeddings from `http://nlp.stanford.edu/projects/glove`

sentences which passed the boilerplate removal step (column nc (after bpr)) we see that the system performs best with the stricter boilerplate removal. This can be explained due to the fact that the nugget sets are smaller and a distribution across multiple clusters is less likely. If we consider all sentences in the data, the system based on the first boilerplate removal performs best (column nc). A strict boilerplate removal with many false positives harms the overall performance of the systems. However, processing fewer sentences in the rather computational expensive clustering (compared to the boilerplate removal) leads to better computational performance. A better boilerplate removal would therefore be desirable to improve the performance in IUS.

6 Results

We present in Table 4 the evaluation scores of our system as well as the official evaluation scores of subtask 3 of the TREC-TS 2015 challenge. The results for our systems are computed with the original evaluation script provided by the TREC-TS organizers. The evaluation results of the other systems are taken from the overview paper of the TREC-TS shared task (Aslam et al., 2015). Due to per-task normalization, metric values across the different subtasks at TREC-TS are not comparable. We only provide a lower bound evaluation of our system, generated by only using the annotations provided by the TREC-TS organizers. As described in Section 4.1, only a small subset of sentences in the corpus is annotated. All non-annotated sentences are considered to be unimportant by the evaluation script, which means that valuable sentences might be misinterpreted as noise when using this dataset without additional annotations. This disadvantage does not apply to the reference systems, since at least the top 60 sentences of each reference systems were labeled for each topic. In total, 57.99% of the sentences selected by our system were not annotated. We report this number for a better assessment of the performance of our system. Although our system might get penalized for valuable sentences, we outperform the other approaches. As described in Section 1, we expect that a sequential clustering approach should be able to detect important information very early. The timeliness of our approach (column E[Latency]) confirms this expectation and is an important factor for the superior performance of our system. We also observe that our system is able to generate a reasonable balance of precision ($n\mathbf{EG}(\mathcal{S})$) and recall ($\mathbf{C}(\mathcal{S})$).

System	$\mathcal{H}$	E[Latency]	$F_1(n\mathbf{EG}(\mathcal{S}), \mathbf{C}(\mathcal{S}))$	$n\mathbf{EG}(\mathcal{S})$	$\mathbf{C}(\mathcal{S})$
`SeqCluSum` (lower bound)	**0.1526**	**0.8013**	**0.1842**	0.1485	0.2426
Raza et al. (2015)	0.0853	0.3983	0.1773	0.1840	0.1710
McCreadie et al. (2015)	0.0639	0.5335	0.1189	0.0667	0.5459
McCreadie et al. (2015)	0.0508	0.6741	0.0758	0.0402	0.6590

Table 4: System results sorted by descending $\mathcal{H}$, the main metric used in the TREC-TS shared task. The columns $n\mathbf{EG}(\mathcal{S})$, $\mathbf{C}(\mathcal{S})$, and E[Latency] show the results according to evaluation metrics described in Section 4. Since the main metric is computed by building a harmonic mean of latency discounted versions of $n\mathbf{EG}(\mathcal{S})$ and $\mathbf{C}(\mathcal{S})$ there is no point in achieving high scores in one of these measures while achieving only a low score according to the other metric.

To find reasonable values for the parameters Θ and μ, we manually evaluated our system with a cross-validation. We did this by randomly splitting the topics in two evenly sized sets of topics and used one of the sets as test set while using the other one for parameter optimization. As described in Section 4.1, we did this on the basis of a sparsely labeled dataset. We expect better results if we could use more densely labeled data. We use the Cosine similarity with $\log +1$ IDF weighting and L2 normalization (cf. Section 5.2). We could not determine a superior boilerplate removal in Section 5.1 and Section 5.3 and therefore apply (Kohlschütter et al., 2010) since it is considered to be a well-known standard boilerplate removal system. During the cross-validation, we found $\mathcal{H}$ scores of 0.1664 and 0.1858 in the validation sets which resulted in $\mathcal{H}$ scores of 0.1550 and 0.1501 in the test sets. We report the averaged scores of both test sets in Table 4.

7 Discussion

The sequential clustering depends strongly on the accuracy of the used similarity measure. Since we only use predefined standard similarity measures, we assume that a more sophisticated selection of similarity measures could lead to an improvement of the performance. More abstract semantic methods or additional knowledge resources may model the semantic similarity of words and sentences better. We therefore assume that the performance of the similarity and redundancy detection can be further improved.

Another improvement of the similarity measure would be to utilize the weights of the tokens, which are used during the contextual measuring of importance also in the clustering. If we weight important tokens higher in the similarity measure, we could achieve that the clusters would be more focused on a particular information nugget.

The clustering can also be improved by introducing a re-clustering step, which could split a cluster into multiple clusters when the system detects that one cluster contains too diverse sentences. This is currently prevented by the parameter Θ. By enabling the system to execute a re-clustering step, this parameter could become obsolete. Furthermore, the system could merge multiple clusters to one cluster if it detects that the content of the clusters are more similar as the seed sentence suggested.

Our model relies heavily on centrality as a features to detect important information timely. Kedzie et al. (2015) mentioned that using centrality as signal for importance is problematic in IUS since it requires some time until an important information is central enough in the news stream. However, the burstiness of news streams can diminishes this issue. A system which is independent from the centrality features would nevertheless be able to detect new important information in big data streams earlier than our systems.

To evaluate our system, we used a cross-validation to optimize the parameters μ and Θ in validation sets and applied the values to test sets. This limits the performance of our system, since the topics have different sizes and different amounts of interesting information. Therefore, an extension of our system where the parameters μ and Θ are adaptive could further improve the performance. μ for example, could depend on the time since no update has been published. If there is a long time span with no updates, μ could be lowered to increase the probability that there will be a new message soon.

8 Conclusions

In this paper, we proposed `SeqCluSum`, a system for incremental update summarization based on sequential clustering and contextual importance measures. It combines the strength of the two prior proposed techniques, namely real-time processing pipelines or clustering approaches. The sequential clustering arranges similar sentences together while the contextual importance measures score the clusters and sentences according to their contextual importance. The contextual importance measures estimate the importance of information as well as their novelty jointly. The evaluation shows that even the lower bound scores of our system outperforms previous state-of-the-art systems. We obtain better latency and higher $\mathcal{H}$ scores, which means that our system detects important information earlier and with a better trade-off between precision and recall. We therefore conclude that sequential clustering in combination with contextual importance measuring is well-suited for the task of IUS. Furthermore, our system does not use the provided query and is hence able to detect important information without using query expansion, which was often used by previous systems to improve recall. We do not use any handcrafted knowledge bases to get additional domain-knowledge, which is another advantage of our system.

Acknowledgments

This work has been supported by the German Research Foundation (DFG) as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1.

References

Javed Aslam, Fernando Diaz, Richard McCreadie, and Tetsuya Sakai. 2014. TREC 2014 temporal summarization track overview. In *Proceedings of The Twenty-Fourth Text REtrieval Conference (TREC 2014)*. National Institute of Standards and Technology (NIST), November.

Javed Aslam, Fernando Diaz, Matthew Ekstrand-Abueg, Richard McCreadie, Virgil Pavlu, and Tetsuya Sakai. 2015. TREC 2015 temporal summarization track overview. In *Proceedings of The Twenty-Fourth Text REtrieval Conference (TREC 2015)*. National Institute of Standards and Technology (NIST), November.

Daniel Bär, Torsten Zesch, and Iryna Gurevych. 2013. DKPro Similarity: An open source framework for text similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 121–126, Sofia, Bulgaria, August. Association for Computational Linguistics.

Sergey Brin and Lawrence Page. 2012. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18):3825–3833.

Jaime Carbonell and Jade Goldstein. 1998. The use of DKPro Similarity, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR 1998)*, pages 335–336. ACM.

Hoa Trang Dang and Karolina Owczarzak. 2008. Overview of the TAC 2008 update summarization task. In *Proceedings of the First Text Analysis Conference (TAC 2008)*.

Günes Erkan and Dragomir R Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Qi Guo, Fernando Diaz, and Elad Yom-Tov. 2013. Updating users about time critical events. In *Proceedings of the 35th European Conference on Advances in Information Retrieval (ECIR 2013)*, pages 483–494. Springer Berlin Heidelberg.

Ivan Habernal, Omnia Zayed, and Iryna Gurevych. 2016. C4Corpus: Multilingual web-size corpus with free license. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 914–922. European Language Resources Association (ELRA), May.

Chris Kedzie, Kathleen McKeown, and Fernando Diaz. 2014. Summarizing disasters over time. In *Proceedings of the Twenty-Third Text REtrieval Conference*.

Chris Kedzie, Kathleen McKeown, and Fernando Diaz. 2015. Predicting salient updates for disaster summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL 2015)*, pages 1608–1617, July.

Jon M Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632.

Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, pages 441–450.

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

Richard McCreadie, Romain Deveaud, M-Dyaa Albakour, Stuart Mackie, Nut Limsopatham, Craig Macdonald, Iadh Ounis, Thibaut Thonet, and Bekir Taner Dinçer. 2014a. University of Glasgow at TREC 2014: Experiments with terrier in contextual suggestion, temporal summarisation and web tracks. In *Proceedings of the Twenty-Third Text REtrieval Conference (TREC 2014)*. National Institute of Standards and Technology (NIST), November.

Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2014b. Incremental update summarization: Adaptive sentence selection based on prevalence and novelty. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 301–310. ACM.

Richard McCreadie, Stuart Mackie, Jarana Manotumruksa, Graham McDonald, Saúl Vargas, M-Dyaa Albakour, Craig Macdonald, and Iadh Ounis. 2015. University of glasgow at TREC 2015: Experiments with terrier in contextual suggestion, temporal summarisation and dynamic domain tracks. In *Proceedings of The Twenty-Fourth Text REtrieval Conference (TREC 2015)*. National Institute of Standards and Technology (NIST), November.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 404–411. Association for Computational Linguistics, July.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2-3):103–233.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 573–580. ACM.

Daraksha Parveen and Michael Strube. 2015. Integrating importance, non-redundancy and coherence in graph-based extractive summarization. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1298–1304. AAAI Press.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543.

Martin Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Dragomir R Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic Summarization*, pages 21–30. Association for Computational Linguistics.

Ahsan Raza, Devin M. Rotondo, and Charles L. A. Clarke. 2015. WaterlooClarke: TREC 2015 temporal summarization track. In *Proceedings of The Twenty-Fourth Text REtrieval Conference (TREC 2015)*. National Institute of Standards and Technology (NIST), November.

Gerard Salton and Michael J McGill. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill.

Sergios Theodoridis and Konstantinos Koutroumbas, 2009. *Pattern Recognition*, chapter 12, pages 633–634. Elsevier Ltd, Oxford.

Zhen Yang Yingzhe Yao and Kefeng Fan. 2015. BJUT at TREC 2015 temporal summarization track. In *Proceedings of The Twenty-Fourth Text REtrieval Conference (TREC 2015)*. National Institute of Standards and Technology (NIST), November.

Yun Zhao, Fei Yao, Huayang Sun, and Zhen Yang. 2014. BJUT at TREC 2014 temporal summarization track. In *Proceedings of the Twenty-Third Text REtrieval Conference*.

Natural Language Generation through Character-Based RNNs with Finite-State Prior Knowledge

Raghav Goyal*
Twenty Billion Neurons
Berlin, Germany
raghav.goyal@twentybn.com

Marc Dymetman
Xerox Research Centre Europe
Grenoble, France
marc.dymetman@xerox.com

Eric Gaussier
LIG, Uni. Grenoble Alpes
Grenoble, France
eric.gaussier@imag.fr

Abstract

Recently Wen et al. (2015) have proposed a Recurrent Neural Network (RNN) approach to the generation of utterances from dialog acts, and shown that although their model requires less effort to develop than a rule-based system, it is able to improve certain aspects of the utterances, in particular their naturalness. However their system employs generation at the word-level, which requires one to pre-process the data by substituting named entities with placeholders. This pre-processing prevents the model from handling some contextual effects and from managing multiple occurrences of the same attribute.

Our approach uses a character-level model, which unlike the word-level model makes it possible to learn to "copy" information from the dialog act to the target without having to pre-process the input. In order to avoid generating non-words and inventing information not present in the input, we propose a method for incorporating prior knowledge into the RNN in the form of a weighted finite-state automaton over character sequences. Automatic and human evaluations show improved performance over baselines on several evaluation criteria.

1 Introduction

Rule-based Natural Language Generation systems (Reiter and Dale, 2000) have been quite successful but they suffer from some limitations. They require extensive human effort and tend to produce fixed, repetitive outputs, which do not closely match human-like utterances. For this reason, there has been much interest recently in developing NLG systems which are, at least partially, able to learn from human produced data (Langkilde and Knight, 1998; Belz, 2008).

In the last couple of years, Neural Network (NN) based approaches have gained enormous popularity within statistical NLP generally, with applications to Machine Translation (Sutskever et al., 2014), Conversation Modelling (Vinyals and Le, 2015) and Parsing (Tai et al., 2015), to cite only a few. In particular, architectures based on Recurrent Neural Network (RNN) such as LSTMs (Hochreiter and Schmidhuber, 1997) and GRUs (Cho et al., 2014) have been succesfully used in Language Modelling tasks due to their ability to model sequential information with long-range dependencies.

An RNN-based approach to NLG has been recently proposed by Wen et al. (2015) in the context of a dialog system, where the input semantic representation is a Dialog Act (DA). The decoder uses words as the units of generation. This word-based model requires pre-processing the original data by substituting named entities with placeholders, a process called *de-lexicalisation*. This is necessary because a standard word-level RNN is not able to "copy" input entities into the target, but has to learn each correspondence individually, which it can only do with a lot of data.

Such a de-lexicalization approach has the advantage of reducing data sparsity, but it also suffers from various shortcomings: (i) it requires some reliable mechanism for named-entity recognition, (ii) it requires a post-processing "re-lexicalization" step, where the placeholders are replaced by the original named entities, (iii) it is unable to account for subtle morphological or lexical effects that a specific

* Work performed during Raghav Goyal's internship at XRCE in 2016.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1083–1092, Osaka, Japan, December 11-17 2016.

named entity may have on its context,[1] and (iv) it does not address the problem of multi-slots of the same type, as for instance when two restaurants are mentioned in the same input.

In this work, we propose to use a **character-level** model which does not suffer from the same sparsity issues as a word-level model, and therefore does not require de-lexicalisation. We show that this architecture, coupled with a bidirectional encoding and an attention mechanism (Bahdanau et al., 2014), is able to "copy" information from the dialog act into the target realization, and to produce reasonable results.

However, this model has two main defects. The first is that it can produce non valid words, the second that it can invent named entities not present in the dialogue act. In order to improve this, we propose to constrain the generation of characters through a certain weighted finite-state automaton that incorporates **prior knowledge**: (i) about well-formed strings of characters and (ii) about the fact that named entities in the realization originate from character strings present in the input.

The following points summarize the key contributions of this paper:

- We handle the NLG problem through an attention-based character-to-character model, in particular we encode the input semantics as a string of characters. By enabling copying at the character level, this prevents us from having to de-lexicalize some aspects of the input, as Wen et al. (2015) are obliged to do.

- In order to improve the quality of the generated utterances (avoiding the generation of non-words or the hallucination of named entities), we exploit *a priori* knowledge in the form of a weighted finite-state automaton that constrains the generated strings of characters to either conform to a predefined vocabulary of words, or to originate in portions of the semantic input. This automaton is integrated within the RNN by employing a generic "background-adaptor" mechanism, a technique recently proposed in (Dymetman and Xiao, 2016), which we explain briefly in a self-contained way.

We start in section 2 by defining the network architecture we apply to all our models and by explaining the background-adaptor technique. We proceed to give details of our different models in Section 3 and introduce a finite-state background over characters. In Section 4 we describe the experiments, with details about the dataset, implementation and evaluation; we also give examples illustrating differences between the models. In Section 5 we discuss related work and finally conclude in Section 6 with some perspectives.

2 Proposed Approach

2.1 The architecture

2.1.1 Recurrent Neural Network (RNN)

Our *encoder-decoder* RNN is based on LSTMs (Hochreiter and Schmidhuber, 1997), which have been shown to be effective in particular in Machine Translation (Sutskever et al., 2014). Our approach is close to that of (Bahdanau et al., 2014), who relax the constraint of having a limited, fixed, length vector representation of the source sentence by using the encoder to produce *bi-directional* embeddings of words of this sentence along with an *attention* mechanism; this mechanism dynamically weighs the source embeddings at each time step, thus enabling the network to "attend" to some specific parts of the (still accessible) input during generation.

In our case, the input (source) representation of the information to be realised by NLG is in the form of a dialog act, as in (Wen et al., 2015), for example:

Dialog Act: *inform(name='phoenix hotel'; area='civic center'; accepts_credit_cards='yes')*
Realization: *the phoenix hotel is near the civic center and accepts credit card -s .*

[1] For example, number or gender (in some languages) agreements between a restaurant name and the remainder of a sentence (French example: *le ritz (la belle époque) est situé (est située)* ...); or spurious repetitions of articles in hotel names (*the HOTEL_NAME is ...* → *the the renaissance is ...*).

While Wen et al. (2015) handle the dialog act as a binary vector encoding different slot-value pairs (after delexicalization), we instead directly encode it as a sequence of tokens, either at the word or at the character level.

2.1.2 Background-Adaptor RNNs

In certain of our experiments, we exploit a variant of recurrent networks which we will call "Background-Adaptor RNNs", which is based on a recent proposal by Dymetman and Xiao (2016) for incorporating prior knowledge in recurrent networks to help training in the presence of limited data.[2] We now briefly explain this technique, which is applied here for the first time to the problem of NLG and to character-based models.

Abstracting away from details, a standard RNN model defines a conditional probability distribution $p_\theta(x_{t+1}|x_1, x_2, ..., x_t; C)$, where C is the context of the model (for us, this is the input dialog act), where $x_1, x_2, ..., x_t$ are the already generated tokens, and where x_{t+1} is the token being generated; the matrix parameters of the models are denoted by θ. The background-adaptor technique extends this as follows:

$$p_\theta(x_{t+1}|x_1, x_2, ..., x_t; C) \propto \underbrace{a_\theta(x_{t+1}|x_1, x_2, ..., x_t; C)}_{\text{adaptor}} . \underbrace{b(x_{t+1}|x_1, x_2, ..., x_t; C)}_{\text{background}} . \quad (1)$$

The difference is that now p_θ is defined as a combined process, obtained by multiplying an "adaptor" a_θ, which is a standard RNN, with a "background" b, which is an arbitrary, *a priori* defined conditional language model; it is given externally and is fixed during training of the combined process. This process is simply obtained by the product of the adaptor and the background, normalized over the different possible vocabulary symbols x_{t+1} to obtain a probability distribution (as indicated by the proportionality symbol). Overall, the process is still only parametrized by θ, and training with it only requires a small modification of the log-loss for taking into account the background factor.

To provide some intuition, let us note that when the background process is a uniform distribution over the vocabulary, we are back to the usual RNN. The other extreme is when the background process exactly corresponds to the actual observed data, in which case the adaptor only has to learn to produce a (close to) uniform distribution (an easy task). The interesting cases are intermediary situations, where the background process incorporates some prior information (for example a generic language model trained on a large corpus), which the adaptor can leverage in order to more easily adapt to the training data (for example a small in-domain corpus). In our application, the background process will provide some hard or soft constraints that the generated symbols have to conform to, presented in the form of finite-state automata.

3 The Models

Word-based model (WORD)

Our first, word-based, model uses the encoder-decoder architecture with the attention mechanism described above. As mentioned earlier, for a word-based model the data has to be de-lexicalised first[3] i.e. the named entities, such as restaurant names, addresses, telephone numbers, etc., have to be replaced by place-holders such as REST_NAME, TEL_NUMBER, and the like. These entities are then re-lexicalized at the end of the decoding to form the final utterance.

[2]The log-linear RNNs introduced in (Dymetman and Xiao, 2016) go beyond background-adaptor RNNs as described here: we only exploit the part of log-linear RNNs concerned with the distinction between the "background" and the "adaptor", not the aspects having to do with log-linear features. The use of a background for supporting RNN training is also implicit in the semantic parsing paper (Xiao et al., 2016).

[3]To emphasize this point, let us note that, in standard word-based seq2seq RNNs, the input and output vocabularies are totally disjoint. In order to map the input word "Ritz" into the output word "Ritz", the RNN has to learn the mapping from scratch based on training data, and cannot rely on any a priori knowledge about the correspondence; if "Ritz" is rare (or nonexistent) in the training data, the mapping cannot be learnt reliably (or at all). By de-lexicalizing "Ritz" into the generic "HOTEL", both in the input and output, the training is much simplified: the RNN only has to learn to map "HOTEL" to "HOTEL", a much more frequent observation.

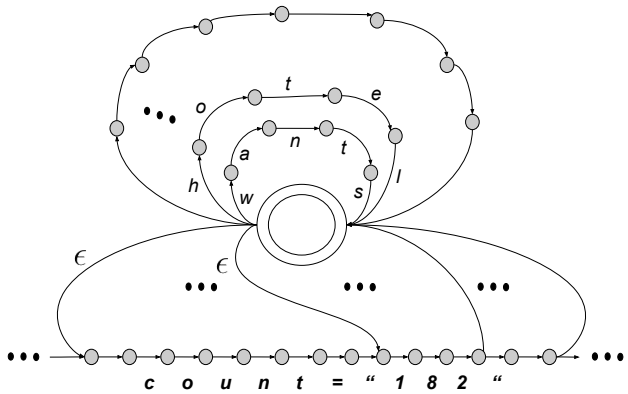


Figure 1: The Background FSA which shows the inclusion of target vocabulary words such as "hotel" and "wants" (top of the figure). Also, it is capable of accepting any substring from the dialog act such as the number "182" (bottom of the figure). The large central state is both initial and final.

Differently from (Wen et al., 2015), who use a one-hot encoding for the input DA, and also add a special gate mechanism to better control the consumption of slots, here we simply treat the DA as a sequence of word tokens.

Character-based model (C)

Our second model is a character-based model in which both the input and the output are character strings. Such models have been used in NMT (Neural MT) to tackle the problem of rare words (Ling et al., 2015; Luong and Manning, 2016). One advantage is that they work over a small vocabulary ($\sim$ 50 symbols), which they have each observed many times; in consequence, they have the ability to learn to map a character onto itself, if the context requires it. This copy mechanism is useful for carrying material from the original unprocessed input to the target, and, perhaps counter-intuitively, is not present in usual word-based RNNs, which have to learn each mapping from scratch (but see fn. 3, as well as the related work section 5 below about augmenting word-level RNNs with a copy ability).

Character-based model(s) with a Finite State Background (C-NWFSA and C-WFSA)

Our last class of models use a background-adaptor approach where the character-based model (adaptor) gets help from a finite-state automaton (FSA) which provides some prior knowledge (background).

The background FSA, in its non-weighted version (C-NWFSA) is illustrated in Figure 1. It accepts: (1) **All common words from the target vocabulary**: these words are all the words (around 500) that may appear in realizations, excluding the named-entities, independently of the DA input; (2) **All substrings from the source dialog act**: the FSA has the freedom to transition to and from any character position in the input DA; this acts as a prior, conditional on the input, which allows the adaptor to copy substrings, in particular named entities, from any part of the DA.

The idea behind this background automaton is that although the character-level model has the capacity to exploit dialog acts without de-lexicalisation (in contrast to the word-level models), nothing prevents it from generating on the one hand non-words, and on the other hand strings of characters that have no evidence in the DA. Through the "intersective" technique of equation (1), the background automaton b constrains the combined process p_θ to only produce common words or substrings of the DA. While the RNN adaptor a_θ works similarly similarly to model (C), the overall training loss depends on b also.

The FSA that we just described is the *unweighted* automaton (C-NWFSA), and its effect is strictly to accept or reject strings of characters. We also consider a *weighted* version (C-WFSA), with the same topology, but where the probabilistic weights on the arcs are parametrized (in a very elementary way) through a single parameter $\alpha \in [0, 1]$, as we now explain.

The total mass of transitions from the start state to one of the "common words" at the top of the figure is α, and the remaining mass $(1 - \alpha)$ is used for (epsilon) transitions to the character representation of

the DA at the bottom of the figure. Between the common words the mass is then distributed uniformly, and for edges from the start state to the DA, the mass is also distributed uniformly onto all the positions of the DA, that is, inversely proportionally to the length of the DA. Once on a position in the DA, the probability of escaping the DA back to the initial state is set to $1 - \alpha$, that of continuing in the DA to α. Overall a higher value of α indicates on the one hand that it is more difficult to "escape" from using common words once using common words, and also that once one is inside the DA, it is more difficult to escape from it back to the common words, which then encourages longer stretches of the DA to be produced once inside it. The way in which we fix α is detailed in section 4.

4 Experiments

4.1 Dataset

The datasets used for our experiments are those made available by Wen et al. (2015), in two domains, *hotel* and *restaurant*. Each set consists of DAs along with their natural language realisations. There are eight different DA types that indicate the communicative intent such as *inform*, *reject*, *confirm* etc. Each DA is a combination of slot-value pairs of the information to be conveyed, with a total of 13 different possible slots such as *name, pricerange, address* etc.

Each domain consists of roughly 5K samples, which we split in the ratio 8:1:1 into training, development and test.[4]

4.2 Implementation

The implementation is done using the Python libraries: Theano (Theano Development Team, 2016) and Lasagne (Dieleman et al., 2015). We implemented the attention mechanism (Bahdanau et al., 2014) by modifying the LSTM class of the Lasagne library. Also, the probability of the combined process c is calculated by element-wise multiplication of individual processes a_θ and b followed by normalization.

The FSA over characters is handled through PyFST[5], a python wrapper for OpenFST (Allauzen et al., 2007). We use this tool to perform the operations of ϵ-removal and determinization (otherwise delicate to program directly), in that order, from an initial non-deterministic weighted or unweighted FSA (which depends in part on the DA input). Finally, the automaton obtained is exported in matrix form to facilitate integration with Theano.

The models are all trained with the same configuration of hyperparameters: the forward and backward RNNs of the bidirectional encoder have 300 hidden units each, similarly the decoder RNN has 300 hidden units. The number of hidden units used in the single layer perceptron for calculating the attention weights is 100. For training, we use SGD together with Adam (Kingma and Ba, 2014) updates, with an initial learning rate of 0.001. A small minibatch of size 20 is chosen due to GPU memory constraints when storing FSA matrices for each sample contained in the minibatch. After the training procedure, beam search is used to sample utterances from the obtained conditional language model. A beam of length 5 is used to obtain the top 5 realisations and the one with highest probability is selected as the prediction.

The α parameter of the weighted FSA is fit to the data by performing a search over the list of values $[0.99, 0.95, 0.9, 0.8, 0.7, ..., 0.1]$. Log-likelihood of the training dataset is used as the optimization criterion, calculated as the sum of log probabilities of all target realizations present in the training set. The maximum likelihood is obtained for $\alpha = 0.9$, both for Hotel and for Restaurant, and this α is used in all our experiments.[6]

Also, the average number of states present in the deterministic weighted FSAs of the training samples is approximately 70K and 120K for the hotel and restaurant domain respectively.

[4]Note that we had to produce our own split, the data provided along with (Wen et al., 2015) does not specify their split, which prevents comparison with their results.

[5]https://github.com/vchahun/pyfst .

[6]As we already mentioned, our approach to fitting the automaton to the data with a single parameter α is simplistic. We could clearly use more parameters, and fit them through some EM procedure.

Model	BLEU	
	Hotel	Restaurant
WORD	0.4495	0.4322
C	0.4200**	0.3699**
C-NWFSA	0.4109**	0.3971**
C-WFSA	**0.4655**	**0.4381**

Table 1: BLEU scores of the models computed on the test set. Statistical significance is calculated using paired bootstrap resampling (Koehn, 2004) $^{**}p < 0.01$.

	Model	Adequacy		Fluency		
		Precision	Recall	No non-words	Non-redundant	Naturalness
Hotel	WORD	0.952*	**0.844**	0.989	0.941*	1.841*
	C	0.844**	0.633**	0.911**	0.974	1.674**
	C-NWFSA	0.844**	0.615**	0.974*	0.974	1.756**
	C-WFSA	**0.978**	0.815	**0.996**	**0.978**	**1.926**
Restaurant	WORD	0.956	**0.793**	0.994	0.976	1.908
	C	0.846**	0.530**	0.926**	**0.988**	1.787**
	C-NWFSA	0.820**	0.609**	0.959**	0.935**	1.731**
	C-WFSA	**0.973**	0.778	**0.997**	0.982	**1.932**

Table 2: Human evaluation of top realisation of the models. Statistical significance is computed through a pairwise difference one-tailed Student's t-test between the model with maximum score against the others. $^{*}p < 0.05$, $^{**}p < 0.01$.

4.3 Evaluation

4.3.1 Automatic Evaluation

Automatic evaluation results are shown in Table 1, using BLEU-4 (Papineni et al., 2002).[7] The results are shown for the four models (WORD, C, C-NWFSA, C-WFSA) described earlier, evaluated over our test sets for Hotel (538 realisations) and Restaurant (520 realisations).

4.3.2 Manual Evaluation

The automatic metrics do not always correlate with human judgement, so we also perform manual evaluation, based on the following adequacy (information conveyed) and fluency (linguistic quality) scales:

1. **Adequacy:**
   - Precision (i.e. "Correctness") [1/0]: all information in the DA is present in the generated utterance (1=yes, 0=no).
   - Recall (i.e. "Completeness") [1/0]: all information in the utterance is present in the DA (1=yes, 0=no).
2. **Fluency:**
   - No non-words [1/0]: all tokens in the utterance are actual words (1=yes, 0=no).
   - Non-redundant [1/0]: there is no repeated information in the utterance (1=yes, 0=no).
   - Natural or "good english" [2/1/0]: the utterance is "good" english (grammatical and natural) (2=good,1=acceptable,0=not acceptable). This is the main linguistic quality measure.

The evaluation was conducted on the full test set, both for Hotel and Restaurant, and is reported in Table 2.

[7] We used the mteval-v13a.pl script with default options from http://www.itl.nist.gov/iad/mig/tests/mt/2009/.

	Selected samples from the Hotel domain
1	inform(name='the inn san francisco';address='943 s van ness ave';phone='4156410188') [C]: the address of the inn san francisco is 943 s van ness ave . their phone number is 4156410188 . [C-NWFSA]: the inn san francisco 's phone number is 4156410188 [C-WFSA]: the address of the inn san francisco is 943 s van ness ave . the phone number is 4156410188 . [WORD]: **the the** inn san francisco 's address is 943 s van ness ave and the phone number is 4156410188 .
2	inform(name='hotel des arts';price_range='moderate') [C]: the hotel des **artea** hotel is in the moderate price range . [C-NWFSA]: the hotel des arts is in the moderate price range . [C-WFSA]: hotel des arts is in the moderate price range . [WORD]: hotel des arts is moderate -ly priced .
	Selected samples from the Restaurant domain
3	inform(name='yummy yummy';price_range=cheap;good_for_meal=dinner) [C]: i have found a restaurant called **mimmmmey** is good for dinner . [C-NWFSA]: **mmy** much is a good restaurant is good for dinner . [C-WFSA]: yummy yummy is cheap and good for dinner . [WORD]: yummy yummy is a cheap dinner restaurant with a cheap price range .
4	inform(name='straits restaurant';price_range=expensive;food=singaporean;good_for_meal=dinner) [C]: straits restaurant is an expensive restaurant that serves singaporean food . [C-NWFSA]: straits restaurant is expensive and serves singaporean food for dinner . [C-WFSA]: straits restaurant is expensive and is good for dinner . [WORD]: straits restaurant is an expensive restaurant that serves singaporean food and is good for dinner .

Table 3: Example realisations of the models. The most probable realization from a beam of length 5 is shown in each case.

4.3.3 Discussion

In terms of BLEU scores, we observe that the character-based model with a weighted finite-state background (C-WFSA) is significantly better than the other character-based models, and slightly better than the (WORD) model.

In terms of manual evaluation, the results are a bit more contrasted. Overall, (C-WFSA) is significantly better than the other two character-based models, apart from the case of "non-redundancy", where it behaves in a quite similar way; this is however not very surprising, because nothing in the FSA background that we presented specifically controls for non-redundancy (the non-repetition of semantic material). (C-WFSA)

All the models are quite deficient in Recall, but (WORD) is a bit better there; again, the FSA background does not control for Recall. On the other dimensions than Recall, (C-WFSA) is slightly better than (WORD), but not always significantly; it is especially good in overall linguistic quality (Naturalness) and in Precision (which is something that the background more directly controls for); interestingly, the unweighted model (C-NWFSA) is significantly worse on precision than the weighted version.

4.4 Illustrations

Examples We show in Table 3 a few examples from the different models. Example (1) shows a case where (C) and (C-WFSA) are both correct, (C-NWFSA) is deficient in recall, and (WORD) after re-lexicalization produces a sequence of two 'the'. Example (2) is a case where (C) invents a named entity. In example (3), (C) also invents a named entity and (C-NWFSA) incompletely copies one. In Example (4), (C) and (C-WFSA) are both deficient in recall, but the other models are good.

Attention Heatmap Because of its independent interest, we also show in Figure 2 an "attention heatmap", here in the case of model (C), for the following example:

```
[DIALOG-ACT]: inform(name='noe 7s nest bed and breakfast';address='1257 guerrero st')
[REFERENCE]: * the address is 1257 guerrero st for noe 7s nest bed and breakfast. #
[REALISATION]: noe 7s nest bed and breakfast is located at 1257 guerrero st .
```

The input DA is written along the columns of the array and the corresponding target realisation is written along the rows. An x^{th} row in the array represents the weights given by the attention mechanism over the character embeddings of the source dialog act at the point where it is generating the y^{th} character of the realisation.

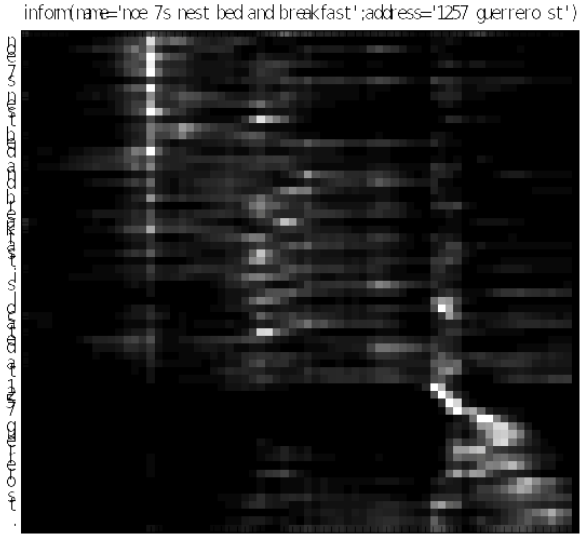


Figure 2: Attention heatmap for a selected example. The x-axis and y-axis denote the input DA and realisation respectively. Each pixel shows the attention weight (white largest) of the x^{th} source character at the point where the y^{th} target character is produced.

We observe a certain "2-block" structure of the attention, with the attention over the hotel slot value when generating the hotel name, over the address slot value when generating its address. The focus of attention is specially strong when generating the four digits 1,2,5,7; one should note that the (C) model has to learn to copy these digits one by one from the input to the target. By contrast, there is very little attention specifically paid to the initial part of the DA "i n f o r m (n a m e", because the little specific information contained there is easy to convey in a more distributed way over the whole input encoding.

5 Related Work

In the field of Neural Machine Translation (NMT), the problem of translating "rare words" such as named entities has recently attracted a fair amount of attention. The main approaches have been ones that either try to augment RNNs with some form of word-copying, or else have, similar to us, some character-level aspects.

Luong et al. (2014) preprocess the data and replace each unknown word in the target sentence by a placeholder token also containing a positional pointer to the corresponding word in the source sentence. Through these pointers, they learn an explicit mapping for copying unknown words from the source to the target. This is a bit similar to the de-lexicalization process of (Wen et al., 2015) in NLG, although the positional aspect might allow to handle several values associated with the same slot type.

Ling et al. (2015) tackle the problem of unseen words by proposing the use of a character-level model instead of a word-level model. As in our case, the limited vocabulary (namely, the different characters) is small enough that the model can learn to copy characters in certain contexts. This model is different from a strict character-level model as it introduces a hierarchy for forming words from characters instead of regarding a sentence as a flat string of characters. Luong and Manning (2016) also propose an hybrid word-character model to handle the rare word problem. For encoding the source sentence, they use a character-level model for rare words and a word-level model for frequent words. Then, for generating a target sentence, they use a word level model to get a first realisation of the target which contains <unk> for rare words and in a second step use a character-level model to generate a realisation of the rare word using contextual information.

Ling et al. (2016), in the context not of NMT this time, but of code generation, introduce a multiple predictor framework, where they can handle jointly the generation of generic code tokens and that of specific tokens for variable names and the like. They choose a character-level softmax for generating generic target tokens along with a pointer network (Vinyals et al., 2015) for copying specific tokens.

We are not aware of any prior work making use of models for *a priori* constraining the string of characters generated by an RNN, as we are doing here.

6 Conclusion

In this work we have proposed a character-level generator which is able to "copy" information from the source dialog act to the target utterance, and which uses original data without requiring pre-processing. By incorporating prior knowledge in the form of a finite-state automaton, exploiting a notion of "background-augmented" RNN, we discourage the character-level model from generating non-existing words or information for which there is no evidence in the input. Overall, the weighted version of the automaton performs much better than the version without prior knowledge, better than the non-weighted version of the automaton, and slightly better than the word-based version that requires de- and re- lexicalisation.

The main areas where our weighted automaton does not perform well (along with all other models, to different extents) are those of "Recall" (expressing all information present in the DA) and "Non-redundancy" (not expressing the same content twice). These are the same areas in which the original model of (Wen et al., 2015) introduced specific machinery, both in terms of a technique for controlling the "consumption" of slots, and of the use of a reranker on top of the operation of the RNN. As a perspective, we could also easily use a reranker, but as a continuation of our overall approach we would preferably incorporate the corresponding constraints as *a priori* knowledge, for instance by intersecting the current automaton with one that (i) forced certain substrings of the DA to appear in the target (improving recall), and (ii) prevented certain substrings to appear twice (improving non-redundancy).[8]

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. Openfst: A general and efficient weighted finite-state transducer library. In *Implementation and Application of Automata*, pages 11–23. Springer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Anja Belz. 2008. Automatic Generation of Weather Forecast Texts Using Comprehensive Probabilistic Generation-Space Models. *Natural Language Engineering*, 14(04):431–455.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, Diogo Moitinho de Almeida, Brian McFee, Hendrik Weideman, Gábor Takács, Peter de Rivaz, Jon Crall, Gregory Sanders, Kashif Rasul, Cong Liu, Geoffrey French, and Jonas Degrave. 2015. Lasagne: First release., August.

Marc Dymetman and Chunyang Xiao. 2016. Log-Linear RNNs: Towards Recurrent Neural Networks with Flexible Prior Knowledge. *arXiv: 1607.02467*, pages 1–22.

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

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*, pages 388–395. Citeseer.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 704–710, Stroudsburg, PA, USA. Association for Computational Linguistics.

[8] One important potential advantage of the background approach over reranking is that it operates more locally and can be used in situations where a reranker would need to filter among a myriad of not-so-good candidates; additionally, the background is exploited by the adaptor when learning the parameters θ. We note also that the background approach does not require b to be realized by an automaton, but can be realized by any predefined process of the form described in (1).

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based Neural Machine Translation. *ICLR'16*, pages 1–11.

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, Andrew Senior, Fumin Wang, and Phil Blunsom. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Association for Computational Linguistics (ACL)*, Berlin, Germany, August.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the Rare Word Problem in Neural Machine Translation. *Arxiv*, pages 1–11.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge.

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

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1556–1566.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. *Neural Information Processing Systems 2015*, pages 1–9.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, September.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings Association For Computational Linguistics*, Berlin, August.

A Hybrid Approach to Generation of Missing Abstracts in Biomedical Literature

Suchet K Chachra
suchet.chachra@gmail.com

Asma Ben Abacha
asma.benabacha@nih.gov

Sonya Shooshan
sonya@nlm.nih.gov

Laritza Rodriguez
laritza.rodriguez@nih.gov

Dina Demner-Fushman
ddemner@mail.nih.gov

U.S. National Library of Medicine, Bethesda, MD, USA

Abstract

Readers usually rely on abstracts to identify relevant medical information from scientific articles. Abstracts are also essential to advanced information retrieval methods. More than 50 thousand scientific publications in PubMed Central lack author-generated abstracts, and the relevancy judgements for these papers have to be based on their titles alone. In this paper, we propose a hybrid summarization technique that aims to select the most pertinent sentences from articles to generate an extractive summary in lieu of a missing abstract. We combine i) health outcome detection, ii) keyphrase extraction, and iii) textual entailment recognition between sentences. We evaluate our hybrid approach and analyze the improvements of multi-factor summarization over techniques that rely on a single method, using a collection of 295 manually generated reference summaries. The obtained results show that the hybrid approach outperforms the baseline techniques with an improvement of 13% in recall and 4% in F1 score.

1 Introduction

PubMed Central[1] (PMC) is a repository of biomedical and life sciences journals supported by the U.S. National Library of Medicine (NLM). PMC provides access to the abstracts as well as the full-text content of biomedical articles. The open-access subset of PubMed Central contains over one million biomedical articles as of Fall 2015, and is widely used as a public resource to discover, read and build upon its vast portfolio of biomedical knowledge. Given the abundance and variety of information available within PMC, many user queries return a multitude of results, which makes it more difficult to identify relevant data. The amount of potentially relevant results often increases further due to information retrieval techniques such as query expansion using synonyms.

Article abstracts are usually considered to be entry points into the full-text. Abstracts often contain key health-outcome data and clinical findings that help identifying relevant data when narrowing down the number of returned results to a select few. The abstracts, however, are missing in 50,000 articles available within the open access subset of PMC. Therefore, for this large set of articles, the only way for the users to judge the relevancy of an article is either through the article title, which is not always reliable, or through the full-text of the paper, which can be time-consuming.

In this paper, we describe a novel hybrid approach that builds upon textual entailment, keyphrase extraction and health outcome detection to generate surrogate abstracts for biomedical articles where none are available. Using a set of 295 documents and manually generated extractive summaries that we make publicly available with this publication, we also analyze how this approach compares to baseline methods relying on a single technique.

2 Related Work

Single and multi-document text summarization of biomedical articles received much attention over the years. Lloret *et al.* developed COMPENDIUM, a text summarization system for generating abstractive and extractive summaries for individual biomedical papers (Lloret et al., 2013). They observed that

[1] http://www.ncbi.nlm.nih.gov/pmc

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1093–1100, Osaka, Japan, December 11-17 2016.

extractive methods are as effective as abstractive summarization or text generation. Kim *et al.* proposed a sub-topic or theme detection method for multi-document clustering and topical summarization of citation data (Kim et al., 2015).

Previous work shows that Recognizing Textual Entailment (RTE) can provide effective information for text summarization. RTE is the task of recognizing an inference relation between two sentences expressing the fact that the meaning of one sentence is entailed by the other (Androutsopoulos and Malakasiotis, 2010; Dagan et al., 2013). In particular, Entailment-based minimum vertex cover method (Gupta et al., 2014) is an RTE method for single document summarization using graph-based algorithms. Textual entailment and logic segmentation based methods also improved performance for single document summarization (Tatar et al., 2008).

Keyword identification methods were also used in single and multiple document summarization and document clustering (Frigui and Nasraoui, 2004; Hammouda et al., 2005), as well as summary generation based on the salience of sentences (Erkan and Radev, 2004). A more detailed survey of summarization methods is presented in (Nenkova and McKeown, 2012).

In this paper, we propose a novel hybrid approach that combines both textual entailment and keyword extraction for the construction of relevant extractive summaries. We particularly show that such combination yields more comprehensive and informative summaries for a variety of documents. Another contribution of our work is a manually created collection of summaries for 295 documents that have no author-generated abstracts. The summaries created by two experts are released with this paper.

3 Methods

In this section, we first describe three baseline methods for abstract generation, where the abstract is generated by combining the top five scoring sentences according to each method, in the order in which they appear in the original text. Second, we describe our method for the recognition of entailment relationships between sentences. Thereafter, we present our hybrid approach that aggregates the best-performing baseline methods and exploits textual entailment relationships in the article full-text to enrich the set of selected sentences.

3.1 Summary Generation based on Health Outcome Identifier (HO)

We used an existing Health Outcome identifier, previously shown to perform well on extracting health outcomes, also called bottom-line, from PubMed abstracts (Demner-Fushman et al., 2006). The health outcome detector employs an ensemble of rule-based, Naïve Bayes, n-gram based, position based, document-length based and semantic classifiers to compute the likelihood scores for each sentence in the article to contain a health outcome. The rule-based classifier analyzes each sentence for existence of cue phrases such as "significantly greater" and "dropout rate". The Naïve Bayes classifier generates a likelihood score based on a bag of words representation of the sentence. The n-gram based classifier looks for uni- and bi-grams that provide a high information gain measure and are strong positive predictors of outcomes such as "superior" and "especially useful". Positional and document length classifiers factor the position of a given sentence in the supplied text and the length of each sentence to provide probability estimates for containing health outcomes. The semantic classifier uses the results of a biomedical concept-extractor that detects presence of biomedical concepts belonging to outcome-related semantic types, such as diseases, symptoms, and medications, within the sentence and concept discovery information from previous sentences to generate a likelihood score. Finally, the probability scores from each classifier are combined to compute the final score $S(x)$ for each sentence x:

$$S(x) = \sum_{k=1}^{n} \alpha_k P_k(x)$$

Where P_1(x), ..., P_n(x) are the probability scores from various classifiers and (α_1, ..., α_n) are the coefficients or weights used to add the likelihood scores.

3.2 Summary Generation based on Keyphrase Extraction

A *keyword* is "a single word that is highly relevant" and a *keyphrase* is "a sequence of two or more words that is considered highly relevant". Our two remaining baseline methods for summary generation are inspired by (Luhn, 1958; Edmundson, 1969) and identify salient sentences to be selected based on detection of keywords or multi-word keyphrases. More precisely, the task is to identify "key sentences" within a given text, defined as the sentences that contain more keyphrases or keywords compared to others. Each method uses a different algorithm for extracting keyphrases from a given text. The extracted keyphrases are then normalized before being used to generate a score for each sentence, using the frequency of contained keywords. The two methods are described below.

3.2.1 Keyphrase Extraction with KEA

The Keyphrase Extraction Algorithm (KEA), developed by (Witten et al., 1999), uses a Naïve Bayes classifier to identify key phrases within text and a discretization scheme developed by (Fayyad and Irani, 1993) based on Minimum Descriptor Length Principle. The algorithm first splits the input text into phrase boundaries, based on punctuation and word boundaries, to look for sequences of words of length up to three to be used as candidate phrases for further examination. Candidate phrases that end in a stopword or occur only once in the text are dropped. Next, the Naïve Bayes model is used on each candidate phrase with feature values t (for $TF \times IDF$) and d (for $distance$) to compute probabilities $P[yes]$ and $P[no]$ that candidate phrase is a keyphrase in the document. The overall probability that the candidate phrase is a keyphrase is then calculated as:

$$p = \frac{P[yes]}{(P[yes] + P[no])}$$

Finally, candidate phrases are ranked according to the above value and the top n keywords are returned, where n is the number of requested keywords. In our experiments, KEA was restricted to output no more than 15 keyphrases per document.

3.2.2 Keyphrase Extraction with Microsoft Text Analytics (MSTA)

The Microsoft Azure Machine Learning suite provides access to Text Analytics web services, which is based on Microsoft Office's sophisticated Natural Language Processing toolkit. MS Text Analytics was used in our experiments to extract keyphrases from the full-text article. For our task, parts of article text were broken into chunks of successive sentences up to 1000 characters long to support the web-service requirement of maximum text length per individual request.

3.2.3 Keyphrase Normalization and Sentence Ranking

Before sentence ranking, a normalization step is performed to remove selected keywords that also occur as complete words within other keyphrases. Acronyms with all uppercase characters are always selected and not filtered during the normalization step. An example of the keyword normalization step is shown below:

$$microCT\ scans,\ microCT \longrightarrow microCT\ scans$$

$$Italian\ Purine\ Clubs,\ Italian\ Purine,\ Purine\ Clubs \longrightarrow Italian\ Purine\ Clubs$$

Once the keyphrase normalization is complete, article sentences are ranked in the order of keyphrase frequency, counting multiple occurrences of the same keyphrase as one. The keyphrase frequency, i.e. the number of keyphrases contained in each sentence is later used to identify target areas of the article text that are relatively more informative compared to others.

3.3 Hybrid Approach Using Baseline Methods and Textual Entailment

Our hybrid approach uses the output of both Health Outcome identification method and KEA keyphrase extraction method to identify a set of candidate sentences, C, as an initial summary. The next factor in the hybrid approach is based on inference relations obtained by recognizing textual entailment between article sentences.

3.3.1 Recognizing Textual Entailment

Textual entailment between two sentences of the same article is recognized using a feature-based classifier. We use a set of similarity measures as learning features. We will refer to it as SimSet in the remainder of the paper.

For a sentence pair (S_1, S_2), three features are computed after stopword removal and word stemming. The first feature is the word overlap between S_1 and S_2. The second feature is the Dice coefficient based on the number of common bigrams. The third feature is the maximum similarity value between five similarity measures: Levenshtein distance, Dice coefficient, Jaccard similarity, Cosine and Word Overlap.

We trained our RTE classifier on the SNLI corpus (Bowman et al., 2015) which contains 570K sentence pairs annotated with three labels: entailment, contradiction and neutral. The authors showed that the size of this corpus allows lexicalized classifiers to outperform some existing sophisticated entailment models. Also, the tested RNN models (a plain RNN and an LSTM RNN) and the feature-rich/lexicalized model show similar performance when trained on the full corpus.

In the scope of our study, we converted the contradiction and neutral labels to the same non-entailment class. Table 1 presents the results of our classifier using the SVM and Logistic Regression algorithms. We apply our RTE method to all possible sentence pairs in each article of our collection.

Classifier	**Accuracy**
SimSet (SVM)	75.86
SimSet (Logistic Regression)	75.64
Lexicalized classifier (Bowman et al., 2015)	75.00

Table 1: 2-class test accuracy on the SNLI corpus for recognizing textual entailment.

3.3.2 Improving Summaries Using Textual Entailment Graph Traversal

The extracted entailment relations are used to generate one or more directed graphs. The vertices V in these directed graphs are the article sentences for which entailment is detected, and the edges E represent directional entailment relations.

The next step involves iterating over each candidate sentence, $C_i \in C \cap V$, involved in at least one entailment relation and selected by the baseline systems. The sub-graph starting at node C_i is then traversed to check if there exists a sentence $V_j \neq Ci$ directly entailed by C_i, or indirectly entailed through a descendant of C_i such that $f(V_j) > f(C_i)$, for a given function f. If such a sentence is found, and has not been previously selected during similar optimization, then V_j is recorded to replace C_i in the final summary.

After the above iteration has been performed for each sentence in $C \cap V$, any unexplored graphs, formed by vertices $V_{rem} \subseteq V - C$ and disjoint from the candidate sentences obtained in earlier steps, are explored beginning from the source node to select additional sentences (one sentence for each disjoint entailment graph). This helps in the enrichment of the final summary by selecting vital hypothesis chains missed by the baseline systems. This allows addressing scenarios where the entailment relations discovered in the article involve other sentences that were not previously selected by the baseline systems, i.e. $C \cap V = \varnothing$. For our various experiments, function f was designed to prefer: i) shorter sentences (Hybrid MinLength), ii) longer, more informative sentences (Hybrid MaxLength) and iii) sentences with higher scores from baseline systems (Hybrid MaxScore).

4 Experiments & Discussion

4.1 Evaluation Dataset

Our experiments were conducted on 295 articles (the evaluation dataset) taken from the open-access subset of PMC. We picked a predetermined number of articles at random from 16 different article types, a classification provided by PMC for each article (e.g., research article, patient's case description or review articles) to ensure a diverse evaluation collection. Table 2 shows the breakup of our evaluation

dataset by article type. We note here that the articles selected for evaluation did not contain author-generated abstracts, so we had to manually generate a reference set of extractive summaries ("Golden Summaries") to be used in the evaluation of the baseline and hybrid system generated summaries.

Article Type	Count
Extended Abstracts	69
Research Articles	48
Review Articles	48
Case Reports	30
Editorials	15
Book Reviews	10
Brief Reports	10
Discussions	10
Letters	10
Meeting Reports	10
News	10
Introduction	5
Obituary	5
Oration	5
Product Reviews	5
Replies	5

Table 2: Article Type counts in the Evaluation Dataset.

4.2 Manual Extraction of Reference Summaries

Two experts, a clinician trained in medical informatics and a medical librarian, were asked to extract "golden summaries" from the articles in the evaluation dataset. Articles of various types were uniformly distributed between the human evaluators. The task was to identify and select key article sentences from the article text. The preferable length in number of sentences for reference summaries was set at 10 sentences, but the system allowed human evaluators to override this limit and adjust it to the minimal length needed to capture all key points of an article. It is important to specify here that manually extracting and compiling reference summaries is highly laborious and required the experts to read the supplied articles, hence being more time-consuming than using author supplied abstracts for evaluations. Figure 1 shows our sentence selection interface for reference summary extraction.

The manually extracted reference summaries are available for download at `https://archive.nlm.nih.gov/ridem/infobot_docs/reference-summaries{.zip,.tar.gz}`

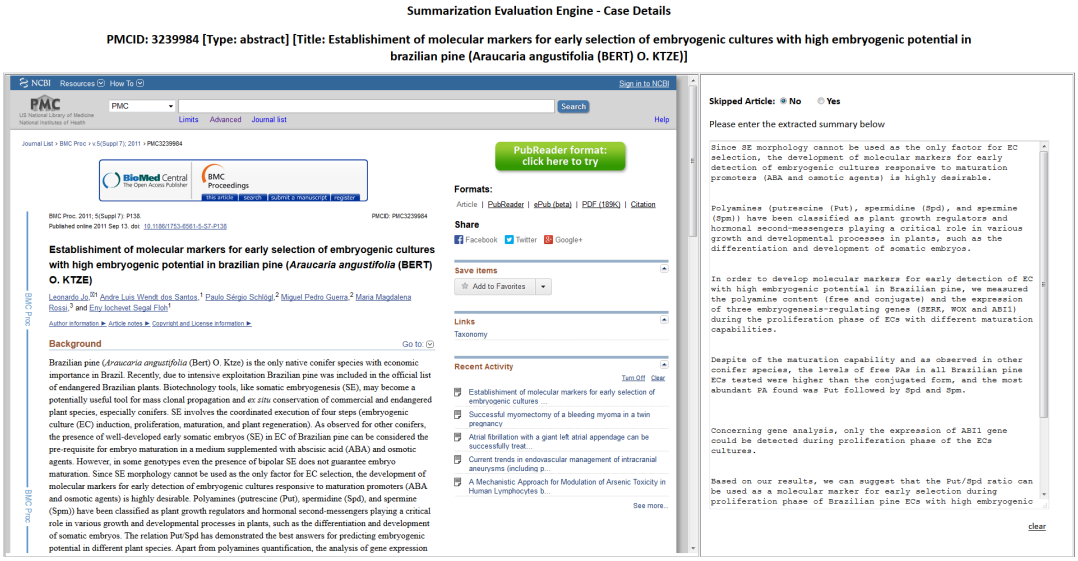
4.3 Judging Baseline System Summaries for Content and Coverage

An additional evaluation task for the human evaluators was to judge the summaries generated by the three baseline systems for content coverage and potential usefulness by rating the baseline summaries on a scale of 1-5 (1=Not at all, 5=Perfect) for the below criteria:

- Is the summary informative?
- Does the summary reflect the most important issues?
- Does the summary capture the bottom-line?

For this task, the generated summaries from three baseline systems were presented to the human evaluators unlabeled and in random order. Table 3 shows for each baseline system the number of articles where the evaluators judged the baseline system summary as acceptable or better (Score, $S_{criterion}$, greater than or equal to 3) for each of the above mentioned criteria. Figure 2 shows our interface for recording such judgements.

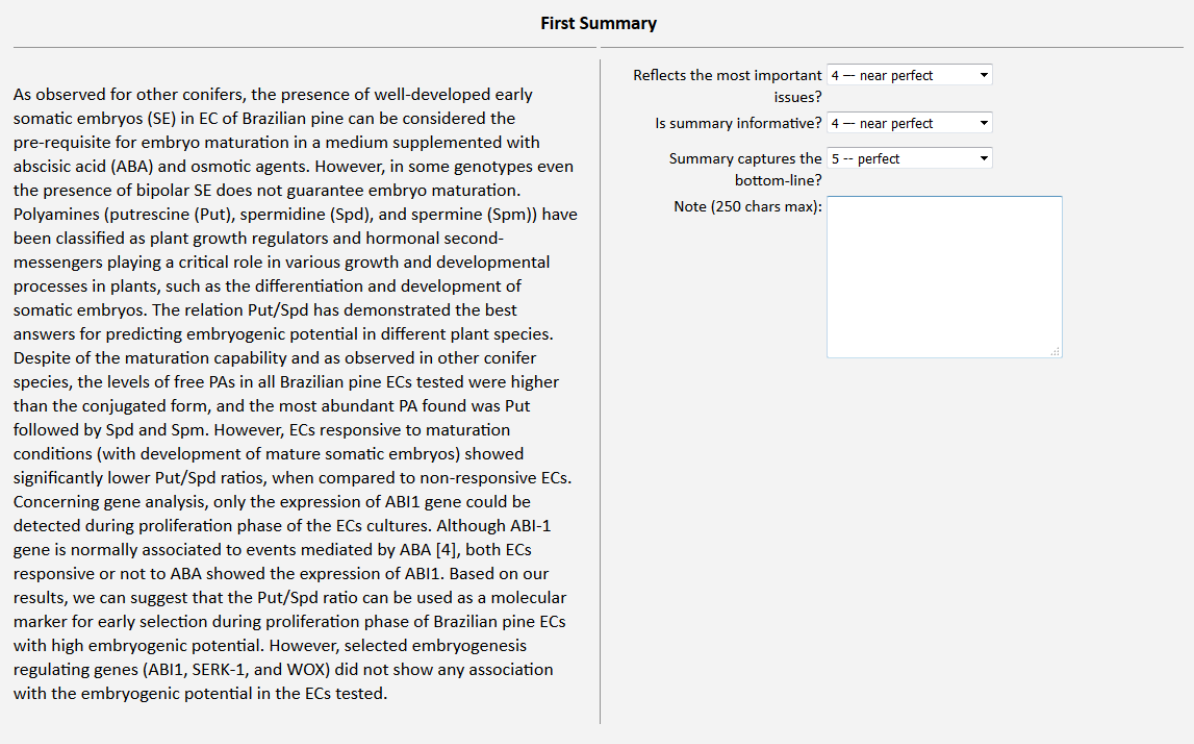
Figure 1: Interface for reference summary sentence selection.



System	Informative ($S_{info} \geq 3$)	Overall Rating ($S_{imp.issues} \geq 3$)	Bottom Line ($S_{bottom} \geq 3$)
HO	242	208	228
KEA	258	228	181
MSTA	244	212	155

Table 3: Manual evaluation of the automatically generated baseline summaries.

Figure 2: Interface for the evaluation of unlabeled baseline summaries.



Based on the preliminary results shown above and the Rouge-2 scores provided in the next section, we decided to base our hybrid approach on the HO and KEA baseline systems and to further improve the combined summaries using Textual Entailment relations recognized in the article text.

4.4 Rouge-2 Evaluation of Generated Baseline and Hybrid Summaries

In addition to manual evaluation of the baseline summaries, we compared them with the hybrid summaries as whole paragraphs to human extracted "golden summaries" using the recall based evaluation metric ROUGE-2 (with stopword removal) for automatic overlap measurement. Table 4 and Table 6 present the Rouge-2 results for the baseline systems and hybrid systems respectively.

System	R (%)	P (%)	F (%)
HO	27.97	27.96	27.13
KEA	28.42	29.44	28.03
MSTA	24.90	20.71	21.99

Table 4: ROUGE-2 evaluation results for summaries generated by baseline systems.

We also tested for, and observed a low overlap between baseline summaries generated by HO and KEA-based systems, which indicates that a hybrid approach could be more comprehensive and likely to outperform individual systems in terms of content recall and coverage. We also observed that in 46 of 130 cases in which an entailment relation was present, our feature-based RTE classifier was able to correctly identify at least one relation involving a sentence that was also selected as prominent by human evaluators. Table 5 presents the summary overlap between baseline systems and inter-annotator agreement.

System	R (%)	P (%)	F (%)
HO Vs. KEA	21.85	19.69	20.21
Inter-Annotator	46.33	42.99	43.47

Table 5: ROUGE-2 results for HO and KEA summary overlap and inter-annotator agreement.

System	R (%)	P (%)	F (%)
Hybrid MinLength	38.82	27.88	31.73
Hybrid MaxLength	41.76	27.41	32.18
Hybrid MaxScore	39.87	27.71	32.88

Table 6: ROUGE-2 evaluation results for summaries generated hybrid systems.

4.5 Discussion

Using a hybrid approach to abstract generation significantly improved the recall while still providing similar precision values, despite the fact that hybrid summaries are generally longer compared to baseline systems. More generally, our experiments show that combining multiple single-factor techniques like keyword extraction, health outcome detection and utilizing semantic relations in text using textual entailment works well for different kinds of articles, and is more likely to outperform traditional baseline approaches for text summarization.

5 Conclusion

We presented a new hybrid approach combining textual entailment and keyword extraction for the summarization of biomedical articles. Our results show that such combination yields substantial improvement in recall while maintaining the precision at the same level. In future work, we plan to incorporate named entity recognition and use the extracted named entities as additional keywords to improve precision and to apply a similar approach to multi-document summarization.

References

Ion Androutsopoulos and Prodromos Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. *J. Artif. Int. Res.*, 38(1):135–187, May.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res. (JAIR)*, 22:457–479.

Usama M. Fayyad and Keki B. Irani. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, pages 1022–1029.

Hichem Frigui and Olfa Nasraoui. 2004. Unsupervised learning of prototypes and attribute weights. *Pattern Recognition*, 37(3):567–581.

Anand Gupta, Manpreet Kaur, Shachar Mirkin, Adarsh Singh, and Aseem Goyal. 2014. Text summarization through entailment-based minimum vertex cover. In *Proceedings of the Third Joint Conference on Lexical and Computational Semantics (*SEM 2014)*, pages 75–80, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Khaled M. Hammouda, Diego N. Matute, and Mohamed S. Kamel. 2005. Corephrase: Keyphrase extraction for document clustering. In *Machine Learning and Data Mining in Pattern Recognition, 4th International Conference, MLDM 2005, Leipzig, Germany, July 9-11, 2005, Proceedings*, pages 265–274.

Sun Kim, Lana Yeganova, and W. John Wilbur. 2015. Summarizing topical contents from pubmed documents using a thematic analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 805–810.

Elena Lloret, María Teresa Romá-Ferri, and Manuel Palomar. 2013. COMPENDIUM: A text summarization system for generating abstracts of research papers. *Data Knowl. Eng.*, 88:164–175.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. In *IBM Journal of research and development 2(2)*, pages159-165.

Harold P. Edmundson. 1969. New Methods in Automatic Extracting. In *Journal of the ACM 16 (2)*, pages 264-285.

Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 43–76. Springer.

Dina Demner-Fushman, Barbara Few, Susan E. Hauser, and George Thoma. 2006. Automatically identifying health outcome information in MEDLINE records.. In *Journal of the American Medical Informatics Association 13(1)*, pages 52-60.

Doina Tatar, Andreea Diana Mihis, and Dana Lupsa. 2008. Text entailment for logical segmentation and summarization. In *Natural Language and Information Systems, 13th International Conference on Applications of Natural Language to Information Systems, NLDB 2008, London, UK, June 24-27, 2008, Proceedings*, pages 233–244.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. KEA: practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM conference on Digital Libraries, August 11-14, 1999, Berkeley, CA, USA*, pages 254–255.

Imitation learning for language generation from unaligned data

Gerasimos Lampouras
Department of Computer Science
University of Sheffield, UK
g.lampouras@sheffield.ac.uk

Andreas Vlachos
Department of Computer Science
University of Sheffield, UK
a.vlachos@sheffield.ac.uk

Abstract

Natural language generation (NLG) is the task of generating natural language from a meaning representation. Rule-based approaches require domain-specific and manually constructed linguistic resources, while most corpus based approaches rely on aligned training data and/or phrase templates. The latter are needed to restrict the search space for the structured prediction task defined by the unaligned datasets. In this work we propose the use of imitation learning for structured prediction which learns an incremental model that handles the large search space while avoiding explicitly enumerating it. We adapted the Locally Optimal Learning to Search (Chang et al., 2015) framework which allows us to train against non-decomposable loss functions such as the BLEU or ROUGE scores while not assuming gold standard alignments. We evaluate our approach on three datasets using both automatic measures and human judgements and achieve results comparable to the state-of-the-art approaches developed for each of them. Furthermore, we performed an analysis of the datasets which examines common issues with NLG evaluation.

1 Introduction

Natural language generation (NLG) is the task of generating natural language from a machine-interpretable meaning representation (MR). Traditionally, NLG systems tend to be rule-based and require domain-specific language resources (i.e. sentence plans, aggregation rules, ordering information, etc.) to be manually authored for a particular task, e.g. weather reporting (Reiter et al., 2005), route navigation (Dale et al., 2003), or ontology descriptions (Bontcheva and Wilks, 2004; Androutsopoulos et al., 2013).

From a machine learning perspective (ML) perspective, NLG is a complex structured prediction task due to its large output space, which is the set of all possible NL utterances. To limit the output space, many ML-based approaches rely on aligned training datasets (Mairesse et al., 2010; Dethlefs et al., 2013) that specify the words that each MR element is responsible for, and phrase templates. While these enable a variety of models to be trained, their applicability is restricted as they need additional manual annotation.

More recently, ML-based approaches focused on learning NLG models from unaligned training data. Dušek and Jurcícek (2015) learn how to incrementally generate deep-syntax dependency trees of candidate sentence plans specifying which MR elements to be mentioned and the overall sentence structure, however the surface realization (i.e. the actual NL word generation) is performed using rules based on the frames of an English-Czech dependency treebank, a resource that may not be available for certain domains or languages. In another approach, Wen et al. (2015) use a Long Short-term Memory (LSTM) network to learn from unaligned data and jointly address sentence planning and surface realization. They augment each cell of the LSTM with a gate that conditions it on the input MR. However the LSTM is trained using cross-entropy at the word level, thus unable to take into account interactions among the word being considered and words to be predicted later in the sentence, an issue which was ameliorated by training a backward LSTM to re-rank the best-scoring outputs generated by the forward one.

In this paper, we propose an ML-based approach that learns sentence planning and surface realization from unaligned training data using the imitation learning Locally Optimal Learning to Search (LOLS)

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1101–1112, Osaka, Japan, December 11-17 2016.

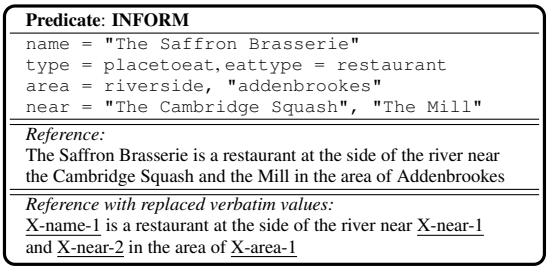


Figure 1: Sample MR and corresponding NL utterance from the BAGEL dataset.

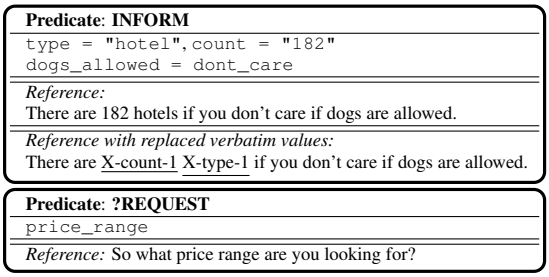


Figure 2: Sample MRs and corresponding NL utterances from the SF datasets.

framework (Chang et al., 2015). Similar to other imitation learning algorithms for structured prediction such as DAGGER (Ross et al., 2011), LOLS reduces structured prediction to classification with greedy inference thus avoiding the enumeration of all outputs, while also ameliorating the issue of error propagation. Unlike other structured prediction frameworks, LOLS is able to learn using non-decomposable loss functions. Thus it is well-suited to learning NLG systems where measures such as BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004) that do not decompose over single words are commonly used for evaluation. Thus we are able to learn the interactions of the word currently being predicted with those to be predicted later in the sentence. We further propose a variant of LOLS using sequence correction and updates (Collins and Roark, 2004), and an exponential decay schedule (Daumé III et al., 2009) . We compare against the systems by Dušek and Jurcícek (2015) and Wen et al. (2015) on their respective datasets (three datasets across two domains) and show that the NLG system proposed achieves comparable results in both automatic and human evaluations.[1]

2 Natural language generation

The NLG process takes as input a meaning representation (MR) consisting of a predicate followed by an unordered set of attributes and corresponding values; the output is a NL sentence. Figures 1 and 2 show samples from the BAGEL (Mairesse et al., 2010) and SF datasets (Wen et al., 2015) that we are considering in this work. The predicates in each MR are utterance-level labels that represent the overall function of the utterance, e.g. whether the utterance should *inform* the user of the attributes and values, or whether it should *request* them from the user. Each attribute may have multiple values, which may be either strings that appear verbatim in the references (e.g. "the Saffron Brasserie", "hotel"), constants from a controlled vocabulary (e.g. `placetoeat`), or boolean (e.g. `yes`, `no`).

We preprocess the MR-NL pairs by replacing verbatim strings with variables ("X-") in the MRs and NLs, which results in the same delexicalized MR paired with multiple NL references (see Figures 1 and 2). Note that the second MR in Figure 2 contains no verbatim strings, and its reference is not modified. Multiple references for the same MR can be beneficial since they capture lexical variation for the same meaning, but also pose a challenge to model learning as they provide ambiguous training signal.

We formulate the generation of a NL utterance from a MR as a sequence of two types of actions, content prediction actions a_c and word prediction actions a_w (Alg. 1). To generate an NL utterance, each content prediction action selects which attribute c should be expressed next; in the case of multiple-valued attributes, one value is chosen based on their order of appearance in the training data. Once the content prediction action sequence is completed, for each selected attribute c, we generate a sequence of words chosen from its corresponding dictionary D_c. This dictionary consists of all the words that we have observed to co-occur with attribute c in the training data. Content and word prediction have special termination actions (END_{attr} and END_{word} respectively). The generation process can stop without exhausting all attributes in the MR thus making it possible to omit information deemed redundant (e.g. that a restaurant is a place to eat). From the action sequence produced, it is straightforward to derive the final sentence by keeping the word prediction actions; the content prediction actions are discarded.

[1] All code is available at `https://github.com/glampouras/JLOLS_NLG`

Algorithm 1: NLG process

Input: meaning representation MR with set of attributes C, attribute dictionaries $D_c, \forall c \in C$
Output: action sequence A

do
  predict attribute $c \in C \cup \{END_{attr}\}$ and append a_c to A_c
  remove c from C
while $a_c \neq END_{attr}$
for a_c *in* A_c **do**
  do
    predict word $w \in D_c \cup \{END_{word}\}$ and append a_w to A_w
  while $a_w \neq END_{word}$
$A = (A_c, A_w)$

Fig. 3 illustrates the action sequence to generate the NL sentence for the MR of Figure 1. The first set of actions is to choose amongst the attributes (`eattype`, `name`, `near`, or the content termination action END_{attr}), and if necessary also choose which value to express (e.g. for `area`, either `riverside` or `X-area-1`). Following this, for each chosen attribute and value we generate an appropriate sequence of word prediction actions (e.g. "at", "the", "side", "of", "the", "river") followed by the END_{word} action denoted by "|" (top part of Fig. 3). Content prediction actions are only used indirectly to generate the sentence; thus different sequences can result in the same sentence, as shown in the bottom part of Fig. 3.

The NLG process defined in this section assumes we train two types of classifiers, one for the content prediction actions and one for word prediction actions for each attribute. If we had alignment information, it would be possible to extract data to train both types of models, either independently (Angeli et al., 2010) or jointly. However, we do not assume access to such information. Instead, we take advantage of the ability of imitation learning algorithms such as LOLS to learn with non-decomposable loss functions by only needing to evaluate complete output predictions instead of individual actions. In NLG's case, this means we do not require explicit supervision for the content and word prediction actions, but only a way to evaluate complete generated sentences against the reference using measures such as BLEU.

3 Locally Optimal Learning to Search

Here we describe how we learn the content and word classifiers we introduced in Section 2. As input to the algorithm (Alg. 2), we assume a set of training instances $\mathcal{S}$ and a loss function ℓ that compares complete NL sequences generated for MRs in $\mathcal{S}$ against references for that MR, e.g. using BLEU or ROUGE. In addition, an expert policy π^{ref} must be specified which will be acting as an oracle during training, returning the best content or word action possible given the words predicted already and the gold standard NL reference for the instance. We describe the expert policy and loss function in detail in separate subsections. Finally, the learning rate β is also part of the input. The output is a learned policy consisting of one classifier for content prediction and one classifier for word prediction for each attribute c, whose label set is the attribute dictionary D_c. These classifiers are learned using a cost-sensitive classification (*CSC*) learning algorithm (*CSCL*). In *CSC* each training example has a vector of misclassification costs, thus rendering some mistakes on some examples more expensive than others.

Before the training iterations begin, a policy π_0 (i.e. content and word classifiers) is initialized on

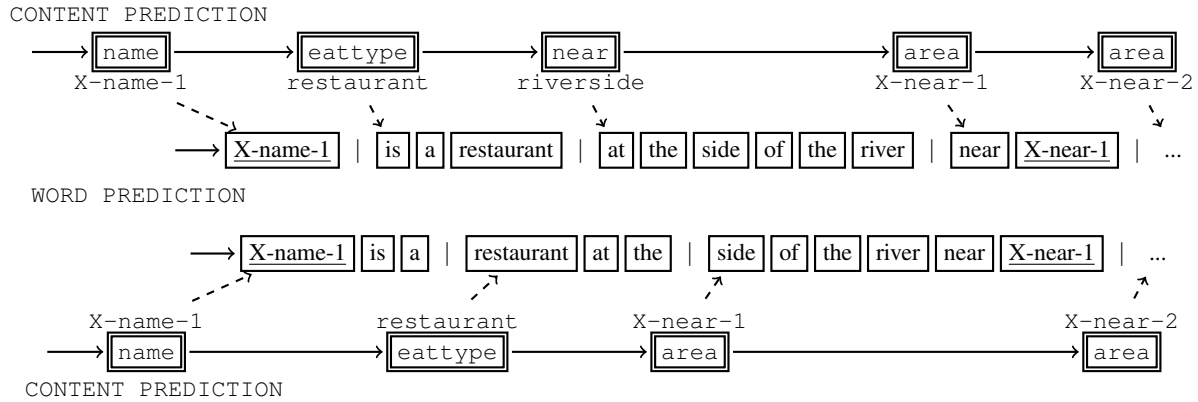


Figure 3: NLG process example with two different content and word prediction action sequences.

Algorithm 2: Locally Optimal Learning to Search (LOLS) for NLG

Input: training instances $\mathcal{S}$, expert policy π^{ref}, loss function ℓ, learning rate β, CSC learner *CSCL*
Output: Learned policy H_N

1 Initialize a policy π_0
2 **for** $i = 0$ **to** N **do**
3 CSC examples $E = \emptyset$
4 $p = (1 - \beta)^i$ // exponential decay
5 **for** s ***in*** S **do**
6 Predict A by executing $\pi_i(s)$ // roll-in
7 **for** *action* a_t ***in*** A **do**
8 Let $\pi^{out} = \pi^{ref}$ with probability p, otherwise $\pi^{out} = \pi_i$
9 **foreach** *possible action* a_t^j **do**
10 $a\prime_{t+1:T} = \pi^{out}(s; a_{1:t-1}, a_t^j)$ // roll-out
11 Assess $c_t^j = \ell(a_{1:t-1}, a_t^j, a\prime_{t+1:T})$
12 Create a feature vector $\Phi_t = f(s, a_{1:t-1})$
13 Add (Φ_t, c_t) to E
14 Set $a^\star = a_t^j$ with minimum c_t^j // find best possible action
15 **if** $a_t \neq a^\star$ **then**
16 evaluate only the next E actions $a_{t+1} \dots a_{t+E}$ **in** A
17 then replace A by executing π^{ref} for actions $a_{1:t+E}$
18 and $\pi_i(s)$ for actions $a_{t+E+1:T}$ // sequence correction
19 Learn $\pi_{i+1} = CSCL(\pi_i, E)$ // update
20 $H_N = avg(\pi_0, \dots, \pi_N)$

the training data. For every instance s of the training data S, the algorithm predicts an action sequence $\{a_0 \dots a_T\}$ (line 6, roll-in) using the learned policy π_i of the previous iteration (if it is the first iteration, this is the initial learned policy π_0), using the NLG process defined in Alg.1. At each timestep t of this sequence, the algorithm considers all actions a_t^j that the (content or word) classifier could take (line 9). In order to estimate the cost of each action a_t^j, the algorithm produces an action sequence $\left\{a_{1:t-1}, a_t^j, a\prime_{t+1:T}\right\}$ (line 10, roll-out) assuming action a_t^j was taken, and assesses it according to the loss function ℓ (line 11). Thus the algorithm learns whether to select an attribute or generate a word by taking the corresponding action and assessing its effect on the quality of the complete NL. A *CSC* training example is generated from these costs (line 12-13), which will be used to update the appropriate classification model in the learned policy. The features Φ_t are extracted from the input MR and all previous actions and the partial NL corresponding to them (line 12) thus capturing (possibly long-range) dependencies between actions. Finally, all the policies trained at the end of each iteration are averaged into a final policy (line 19). The learned policies consisting of classifiers, unlike the expert policy π^{ref} which is available only for the training instances, can generalise to unseen data.

When examining all possible actions, the algorithm generates the rest of the alternative sequence (roll-out) $\{a\prime_{t+1:T}\}$ according to either the expert policy π^{ref} or the learned policy π_i, depending on the probability p. In its original specification (Algorithm 1 by Chang et al. (2015)), LOLS chooses which policy to use for each roll-out separately (π^{out} is set inside the foreach loop) and may use a different policy to perform the roll-out for each possible action, but we use the same policy (expert or learned) for all actions to ensure that their costs are calculated consistently. We update this probability p (line 4) at the start of each training iteration, according to the learning rate β. Chang et al. (2015) set this probability to a fixed value throughout all iterations, but we opted to follow SEARN's exponential decay schedule (Daumé III et al., 2009), to increasingly move away from the expert policy as we found it to work better in initial experiments; we discuss this further in section 4.2. By gradually decreasing the use of π^{ref} in the roll-outs, the costs used to train the classifiers are adjusted to their own predictions, thus teaching them to predict actions jointly.

3.1 Expert policy

The expert policy π^{ref} is invoked when assessing possible actions using roll-outs (line 10). Given the sequence of actions already taken $\{a_0 \dots a_t\}$, it acts as an oracle that returns the best possible action

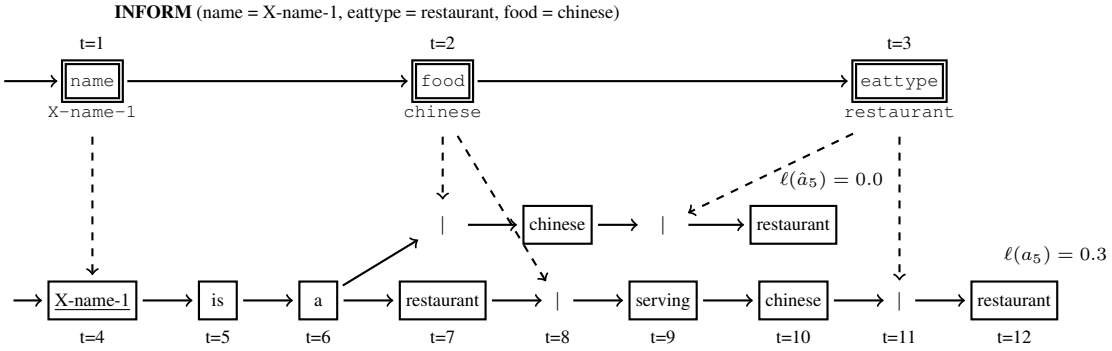


Figure 4: LOLS roll-in (bottom) and roll-out (upper branch) trajectories. Content and word actions are in single and double boxes respectively.

to take, according to the NL reference for the training instance. If a word action is to be predicted by the expert policy we locate the word that makes the sequence (ignoring content actions) best match (i.e. minimize the loss function to) the NL reference. In order to return content actions though, an optimal expert policy would need access to alignments between the MR and the NL reference, which we assume we don't have. Instead, we define a simple heuristic alignment approach that augments the training data with noisy alignments, resulting in a suboptimal expert. LOLS is particularly suitable for this as it can learn with suboptimal expert policies (Chang et al., 2015). This is due to assessing the costs for all possible actions via roll-outs evaluated using the loss function against the (gold) reference, from which the classifier can learn to prefer actions that could contradict those returned by the expert policy. Additionally, in LOLS there is no per-time step mixing of expert and learned policies (i.e. all actions in a single roll-out are predicted by either the expert or the learned policy), which makes designing the expert policy much simpler overall; this is important in tasks like NLG where the optimal action changes drastically when mistakes happen, which is unlike tasks like POS tagging.

The naive alignment used in the expert policy uses the following heuristics. The verbatim values of the MR are simply aligned with their occurrences in the NL reference. The constant values are compared with all unaligned word n-grams in the reference, and aligned with the most similar one according to their character-based Levenshtein distance. For boolean attributes, or attributes with the `dont_care` value, we compare the attribute name with the unaligned n-grams (e.g. aligning `dogs_allowed="yes"` to "allows dogs") instead of the value. Finally, inferred alignments are expanded to unaligned words on the right and left side of the sequence until an already aligned word (or punctuation) is found.

Figure 4 illustrates how examining all possible actions at each time step can overcome the errors in the naive alignments. The content sequence has already been predicted (top of the figure), and at timestep $t = 7$, the classifier returns action $a_7 =$ "restaurant". However, later on the sequence we need to predict words for the attribute `eattype`, for which the classifier (due to the naively aligned training data) is likely to repeat the word "restaurant" leading to an unnatural utterance. So, given this action ($\hat{a}_7 =$ "restaurant"), the resulting sequence has a cost of $\ell(a_7) = 0.3$. By examining all possible actions, the algorithm can surmise that if the classifier stops generating words about the current attribute ($\hat{a}_7 =$ "restaurant"), it would lead to a more natural utterance, and hence a lower loss ($\ell(\hat{a}_7) = 0.0$) which will update the learned policy accordingly. An alternative course of action would be for the classifier to learn to stop generating words about the attribute `eattype` at action $\hat{a}_{12} =$ "restaurant".

3.2 Loss function

In LOLS the loss function is only used to compare complete action sequences against references (line 11). Therefore, it does not need to decompose over individual actions and allows us to use measures like BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004) as loss functions. These measures can consider multiple references for the same MR, which is useful in learning lexical variation, i.e. multiple ways of expressing the same meaning. The loss function is used to assess roll-outs obtained by both learned and expert policies when assessing both action types. In this way, content actions, which are ignored by the loss function, are evaluated based on their impact on word predictions that follow them. When invoking

the expert policy for content actions, the loss function only checks if the reference contains the attributes in the roll-out. We examine the impact of various loss functions in section 4.2.

Additionally, the loss function penalizes actions that are certain to lead to undesirable behavior, i.e. repeating words, producing a termination action without having generated anything, predicting attributes not present in the MR, or attributes whose dictionary does not contain the word(s) that should follow.

3.3 Sequence correction

Another modification we introduced to the original LOLS algorithm is sequence correction in which soon after the first suboptimal action a_t (i.e. not having the minimum cost among those possible) of the roll-in trajectory is encountered (lines 14-18), we attempt to correct the errors encountered so far in the sequence before examining further actions. We allow the algorithm to examine at most E actions ($a_{t+1} \dots a_{t+E}$) following the first suboptimal one (line 16) before re-predicting the roll-in trajectory using the expert policy π^{ref} up to and including the final examined action $a_{1:t+E}$, and using the policy π^{out} to predict the rest of the sequence $a_{t+E:T}$ (lines 17-18). We then proceed with examining the next action a_{t+E+1} of the new corrected sequence. If suboptimal actions are encountered further in the new sequence, sequence correction may again be performed and the sequence re-predicted.

We found this to be helpful in avoiding generating noisy *CSC* training examples. Consider the reference "'X-name-1 is a hotel that allows dogs" and the predicted (incorrect) partial output "X-name-1 is a dog". When assessing the action for the next word, it is likely that predicting "that" or "hotel" will result in lower cost than END_{word} since a loss function such as BLEU or ROUGE would reward the increased word overlap. Assuming that we extract the previous word as a feature in line 12, this would result in learning that predicting "hotel" after "dog" is beneficial. However, this appears to be the case due to the the loss function and the incorrect partial prediction. Correcting the sequence after we encounter the suboptimal word action "dog" to "'X-name-1 is a hotel" will provide the appropriate context for the next action's *CSC*; most of the features Φ_t are extracted from this context. The intuition is similar to the early updates in the incremental perceptron proposed by Collins and Roark (2004).

3.4 Comparison to other Imitation Learning frameworks

In previous subsections, we introduced three modifications to the LOLS framework: 1) the use of SEARN's exponential decay schedule when determining the π^{out} policy, 2) the use of the same policy (expert or learned) for actions in the same step of the sequence, instead of potentially using a different policy for each roll-out, and 3) using sequence correction when encountering suboptimal actions in the sequence.

The main difference of LOLS to other imitation learning frameworks is how the roll-in and roll-out predictions are performed. LOLS performs each roll-in deterministically using the expert policy π^{ref}, while the SEARN and DAGGER (Ross et al., 2011) frameworks roll-in using mixtures of the learned and expert policies; however, partially using the expert policy to roll-in limits those algorithms' capability to learn from the classifiers' mistakes. For roll-out, SEARN stochastically uses a mixture of the learned and expert policies, similarly to LOLS. DAGGER does not employ roll-outs, but uses the expert policy (also referred to as dynamic oracle) to "teach" the correct action to the classifier; this means that it cannot be used to learn using non-decomposable loss functions. Later, a variant of DAGGER, coined V-DAGGER (Vlachos and Clark, 2014), introduced roll-outs to the framework (again using a stochastic mixture of the two policies). AGGREVATE (Ross and Bagnell, 2014) uses a similar roll-in strategy to LOLS, but the expert policy is used exclusively for each roll-out; in the case of NLG, where the expert policy is suboptimal (see section 3.1), this may introduce noise to the classifier.

4 Data and experiments

4.1 Implementation details

We use the adaptive regularization of weight vectors (AROW) algorithm (Crammer et al., 2013) for cost-sensitive classification learning. We train separate content and word classifiers for each predicate, and predicate-attribute combination, respectively. Both types of classifiers (word or content action classifiers) exploit features from the immediately preceding words, attributes and attribute-value pairs, and from

language models estimated on the classifier's corresponding training data. They also consider features based on the input MR, such as which attributes and values have already been mentioned and which have not, and, specifically for the word classifiers, whether a word partly or wholly expresses a value.

During training, we also prevent the classifiers from generating more actions than what has been observed in the training data. Finally, the NL utterances are generated in a "delexicalized" form, but we replace the variables with the corresponding values in post-processing; this is the reverse procedure of replacing verbatim values with variables as shown in Figures 1 and 2. The same system is applied to all datasets discussed in the following sections, and its code will be publicly available upon publication.

4.2 Experiments with the SF datasets

The two SF datasets were created by Wen et al. (2015) and contain MRs about hotels and restaurants respectively.[2] The hotel dataset consists of 5,192 MRs with a single NL reference each. They contain 8 predicates (`inform`, `confirm`, etc.) and 12 attributes, some of them unique to each dataset. Each attribute can take at most one value (or be empty). Following Wen et al. (2015) we partitioned the data into a training, validation, and testing set in a 3:1:1 ratio. We also created multiple references for each validation and test MR as they did, by grouping all delexicalized references that correspond to the same delexicalized MR. We then repopulate the references according to the MR's values.

Figure 5 shows heat maps of BLEU-4 (top set) and ROUGE-4 (bottom set) scores our system achieved on the validation set of the SF hotel dataset for different values of the learning rate β and the sequence correction parameter E; the notation *"std"* denotes that no sequence correction was used in that configuration. The heat maps are also grouped by which loss function was used in each configuration. By comparing the best configurations, we can conclude that using sequence correction leads to slightly better results (a 1.56 difference on average in BLEU). In figure 6 we show how the learned policy is improved with each epoch of LOLS ($\pi_0 \ldots \pi_3$); note that figure 5 shows the best epoch of each configuration.

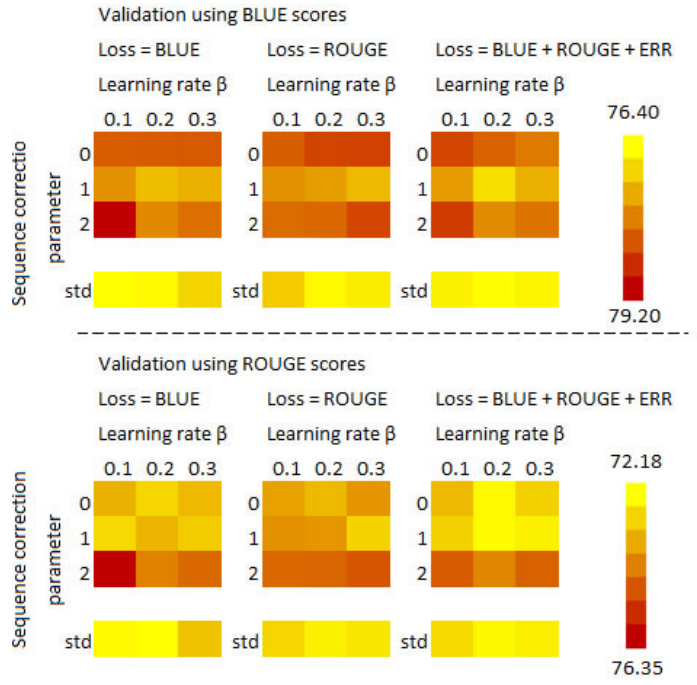


Figure 5: Result heat map for SF hotels across different parameters.

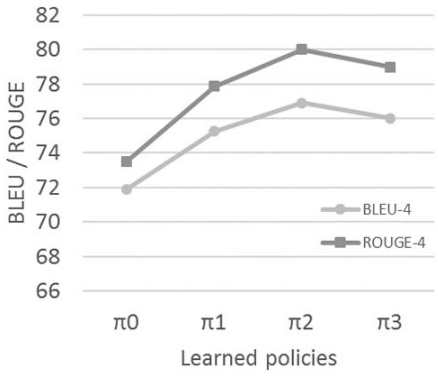


Figure 6: Result heat map for SF hotels across different parameters.

As loss functions to LOLS we tried BLEU-4, ROUGE-4, and the harmonic mean of BLEU-4, ROUGE-4 and ERR(%). In essence, BLEU measures the precision of the sentence's content while ROUGE measures the recall. We opted to use ROUGE-4 rather than other variants of ROUGE since it has been shown to

[2] SF is freely available for download at: https://www.repository.cam.ac.uk/handle/1810/251304

correlate better with human assessments in summarization tasks (Graham, 2015). ERR(%) measures what portion of the MRs is not expressed in the NL sentence; it is defined as the number of attribute-value pairs in the MR that are not expressed in the generated NL sentence to the total number of attribute-value pairs in the MR. We can map the values of the MR to the NL sentence similarly to how we calculate the naive alignments (see section 3.1). By comparing these loss functions on the heat maps of Figure 5, we find that there is very little difference between them (0.12 difference on BLEU, and 0.36 on ROUGE). We decided to use the harmonic mean in our experiments since it captures more information.

Additionally, we tried setting a fixed value to probability p (see line 4 in algorithm 2) as per Chang et al. (2015) instead of using SEARN's exponential decay schedule (Daumé III et al., 2009). When comparing the best configurations on the validation set (with no sequence correction), using a fixed value seems to be slightly worse than using an exponential decay schedule (0.89 BLEU difference in SF hotel).

To assess the impact of naive alignments (see section 3.1) on our system we tried initializing the policy π_0 on random alignments. In this case, the values of the MR are aligned with words in the NL reference randomly. Subsequently, the random alignments are expanded alternatively to unaligned words similarly to naive alignment expansion as described in section 3.1. The best configuration with random alignments, as expected, performs worse to using naive alignments (25.84 difference in BLEU), but LOLS still improves on the initial policy (5.49 improvement on π_0). While all results reported above are on the SF hotel dataset, similar conclusions can be drawn for the other datasets we examined.

	SF RESTAURANT								
	FULL TEST (1040 MRS)			UNIQUE (158 MRS)			NON-OVERLAPPING (13 MRS)		
	BLEU	ROUGE	ERR(%)	BLEU	ROUGE	ERR(%)	BLEU	ROUGE	ERR(%)
WEN	74.50	77.75	2.54	52.97	43.52	6.29	27.04	20.44	10.38
LOLS	66.01	64.56	0.15	49.44	38.52	0.58	28.21	21.47	0.00

	SF HOTEL								
	FULL TEST (1076 MRS)			UNIQUE (96 MRS)			NON-OVERLAPPING (11 MRS)		
	BLEU	ROUGE	ERR(%)	BLEU	ROUGE	ERR(%)	BLEU	ROUGE	ERR(%)
WEN	86.54	84.36	0.88	66.37	56.19	3.99	37.24	27.27	6.82
LOLS	80.00	76.88	0.25	68.65	68.37	0.52	33.31	27.01	3.63

Table 1: Results on the SF datasets.

Finally, we use the test set to compare against the results of Wen et al. (2015), while setting the parameters based on the validation set results. Our test set results are summarized in Table 1; we report BLEU-4, ROUGE-4, and ERR(%). The BLEU-4 and ERR(%) scores of Wen et al. (2015) have been updated after personal communication with them; both are now computed using the same implementation of the measures. The scores are calculated on the full test set, as well as two smaller subsets. The first subset consists exclusively of the unique MRs in the test set (i.e. no MR appears more than once in the unique subset). The results in the unique subset may be more indicative of a system's performance as the distribution of MRs in the test set is unbalanced, e.g. an extreme case in the full test set, is the MR `goodbye()` which appears 176 times. Comparing the two systems, we see that LOLS performs worse in terms of BLEU and ROUGE, but also has lower ERR(%); this indicates that our system tries to include more attributes in the NL sentences. However, when looking at the scores in the unique MR subset, the difference between the two systems narrows, with almost no difference in the SF hotel dataset.

Examining the test set further, we noticed that there is a big overlap between the training, validation and test sets when we consider the MRs in their delexicalized form (see sction 2). Given that both our and Wen et al. (2015)'s models are trained on the delexicalized forms of the training MRs that means that the evaluation is, in part, performed on MRs that have already been encountered during training. In an effort to more accurately assess how the models generalize to unseen input, we define a second subset of the test set, consisting of the unique MRs whose delexicalized version does not appear in either the training nor validation data (i.e. non-overlapping MRs). As it turns out, these are only 11 and 13 non-overlapping MRs in the SF hotel and restaurant datasets respectively. Comparing the two systems in this second subset shows little difference between them; however we cannot draw any conclusions as the subset is very small. We show below some output utterances from Wen et al. (2015)'s system and ours:

MR: inform(type=restaurant,count='239',good_for_meal=dont_care,food=dont_care)

WEN: I have found 239 restaurants serve any meal if the food type is not an issue.

LOLS: There are 239 restaurants if you do not care about the meal and food.

MR: inform(name='kean hotel',has_internet='no')

WEN: We're sorry, the Kean hotel does not have access to the internet.

LOLS: We're sorry, the Kean hotel does not have internet.

4.3 Experiments with the BAGEL dataset

The BAGEL dataset was developed by Mairesse et al. (2010) and contains 202 MRs, each corresponding to 2 NL references.[3] The dataset describes restaurants, and contains one predicate (`inform`) and 8 attributes (see Figure 1). Each attribute can take multiple values, which are either verbatim strings or constants.

	BLEU	ROUGE	NIST	ERR(%)
DUSEK				
Basic perceptron	52.35	46.43	4.777	23.90
+ Diff-tree upd.	56.75	50.82	5.137	25.10
+ Future promise	56.71	48.08	5.441	24.16
LOLS	54.22	47.39	5.547	16.66

Table 2: Results on the BAGEL dataset.

In this section we are comparing against Dušek and Jurcícek (2015), who based their work on the BAGEL dataset. We include results from three variations of their system, the first using only a perceptron to score the output, the second augmented with differing tree updates and the last with future promise estimations. Following their work, we use 10-fold cross validation, with no overlap between MRs in the training set and the test set; we obtained and use the same folds. Their scores have been recalculated using the same implementation of the measures we use. The BAGEL dataset has no separate development set, so we tuned our system's parameters using 10-fold-cross-validation as well.

Table 2 shows our results; we report BLEU-4, ROUGE-4, NIST (Doddington, 2002), and ERR(%). We have achieved better NIST scores than Dušek and Jurcícek (2015)'s, with slightly worse BLEU-4 and ROUGE-4 scores. Our lower ERR(%) suggests that our system includes more information (i.e. attributes-value pairs) into the sentences, which may account for the slightly worse BLEU and ROUGE scores. Another explanation for Dušek and Jurcícek (2015)'s performance is, as stated before, that their surface realization is performed using rules based on the frames of an English-Czech dependency treebank.

4.4 Human evaluation

To better assess the perceived quality of our system's output, we showed pairs of MRs and generated utterances for all datasets to 202 human judges, not involved in the work of this article. They were all fluent, though not native, English speakers. We generated 1076 + 1040 + 202 texts from the MRs of the SF hotel, SF restaurant, and BAGEL data sets using our NLG system, and also obtained the respective texts that Wen et al. (2015) and Dušek and Jurcícek (2015) generated in their work using their best configurations. Each MR and generated utterance was shown to (at least) three human judges, in random order. For each pair, the judges were asked to either score the generated utterance in terms of fluency, and informativeness; a scale from 1 to 6 was used. Each judge scored fluency and informativeness on separate sets of texts, to minimize correlation between the criteria. Among the texts that every judge scored, was a small subset of texts whose quality in regards to each other was known beforehand. We used that subset to filter out unreliable human judges that did not score these texts as expected.

	BAGEL		SF RESTAURANT		SF HOTEL	
	Dušek and Jurcícek (2015) + Future promise	LOLS	Wen et al. (2015)	LOLS	Wen et al. (2015)	LOLS
Fluency	5.15*	4.79*	4.49	4.23	4.41	4.68
Informativeness	4.53*	5.24*	5.29	5.36	5.36	5.19

Table 3: Human evaluation on BAGEL and SF datasets.

[3] BAGEL is freely available for download at: http://farm2.user.srcf.net/research/bagel/

Table 3 shows the results of the human evaluation, averaged over all the texts. In the BAGEL dataset, our system performs slightly worse in terms of fluency, but is perceived as more informative. In the SF restaurant dataset, our system performs worse in terms of fluency and better in informativeness, while in the SF hotel dataset, our system performs better in fluency and worse in informativeness. However, no statistically significant difference was detected in the SF datasets for both of these criteria.[4] Table 4, shows the results if we isolate the human judgements for the subset of non-overlapping MRs (see section 4.2); we see that the differences between the systems are much clearer but again we must state that no safe conclusions can be drawn from that small a subset.

	SF RESTAURANT		SF HOTEL	
	Wen et al. (2015)	LOLS	Wen et al. (2015)	LOLS
Fluency	3.53	3.23	2.38	4.22
Informativeness	4.90	5.23	4.65	4.30

Table 4: Human evaluation on the non-overlapping MRs of the SF datasets.

5 Related work

Apart from the works we compared against, only Konstas and Lapata (2013) and Mei et al. (2016) do not need pre-aligned data. Konstas and Lapata (2013)'s approach incorporates the work of Liang et al. (2009) that learns a generative semi-Markov model to calculate the alignments. We note that this alignment model is developed on the datasets considered, and does not generalize equally well to other datasets (Angeli et al., 2010). On the other hand, the naive alignments we infer are much simpler and we improve them by joint learning of word and content action prediction with respect to the sentence-level evaluation via BLEU and ROUGE. Concurrently, Mei et al. (2016) introduced an encoder-aligner-decoder model to perform content selection and surface realization without pre-aligned data. Their work employs bidirectional LSTM-RNN models, similarly to the work of Wen et al. (2015), and a coarse-to-fine aligner. Unfortunately, they do not report results in the datasets we performed our evaluation on, do not compare against Wen et al. (2015), and their code was unavailable when we were preparing this article.

Imitation learning algorithms for structured prediction have been applied successfully to a variety of tasks, such as dependency parsing (Goldberg and Nivre, 2013) and dynamic feature selection (He et al., 2013). Vlachos and Clark (2014) applied a variant of DAGGER (Ross et al., 2011) to learning a semantic parser from unaligned training examples, which is the reverse task to NLG, i.e. predicting the MR given the NL utterance. To circumvent the lack of alignment information they resorted to defining a randomized expert policy similar to the heuristic one we define, but NLG poses a greater challenge since the output space is all English sentences possible given the vocabulary considered.

Finally, we believe that the main benefit of our imitation learning approach, namely that it is able to learn using a non-decomposable loss function, is orthogonal to using continuous representations such as the hidden state and memory cell in the LSTM of Wen et al. (2015). Recent work by Ranzato et al. (2016) showed how RNNs can be trained at the sequence level (as opposed to the word level) with non-decomposable loss functions in the context of machine translation using imitation learning, and such an approach would also be applicable to NLG.

6 Conclusions

In this work, we adapted the Locally Optimal Learning to Search algorithm (Chang et al., 2015) to learn an NLG system from unaligned training data, focusing on its ability to learn with non-decomposable loss functions and suboptimal expert policies. We compared our approach to two recently proposed approaches for learning NLG from unaligned data on datasets they were developed on (three datasets across two domains) and achieved comparable results in automatic evaluation. We also performed human evaluation which showed that our system performs comparably, though its fluency requires improvement on MRs with multiple attributes.

[4] We performed Analysis of Variance (ANOVA) and post-hoc Tukey tests ($\alpha = 0.05$); * denotes statistical significance.

Acknowledgements

This research is supported by the EPSRC grant Diligent (EP/M005429/1). We would like to thank Tsung-Hsien Wen and Ondřej Dušek for their help on the evaluation, Lucia Specia for her advice on the evaluation, and anyone who participated in the human evaluation. Finally, we would like to thank our anonymous reviewers for their helpful comments that helped improve the article.

References

I. Androutsopoulos, G. Lampouras, and D. Galanis. 2013. Generating natural language descriptions from OWL ontologies: the NaturalOWL system. *Journal of Artificial Intelligence Research*, 48(1):671–715.

G. Angeli, P. Liang, and D. Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pages 502–512, Cambridge, MA.

K. Bontcheva and Y. Wilks. 2004. Automatic report generation from ontologies: the MIAKT approach. In *Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems (NLDB 2004)*, pages 324–335, Manchester, UK.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 111–118. Association for Computational Linguistics.

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

Robert Dale, Sabine Geldof, and Jean-Philippe Prost. 2003. Coral : Using natural language generation for navigational assistance. In Michael J. Oudshoorn, editor, *Twenty-Sixth Australasian Computer Science Conference (ACSC2003)*, volume 16 of *CRPIT*, pages 35–44, Adelaide, Australia. ACS.

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

Nina Dethlefs, Helen Hastie, Heriberto Cuayáhuitl, and Oliver Lemon. 2013. Conditional random fields for responsive surface realisation using global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1254–1263, Sofia, Bulgaria, August. Association for Computational Linguistics.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT 2002, pages 138–145, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ondřej Dušek and Filip Jurcícek. 2015. Training a natural language generator from unaligned data. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 451–461.

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

Yvette Graham. 2015. Re-evaluating automatic summarization with bleu and 192 shades of rouge. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 128–137, Lisbon, Portugal, September. Association for Computational Linguistics.

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

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48:305–346.

P. Liang, M. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the 47th Meeting of the Association of Computational Linguistics and 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL/AFNLP '09)*, pages 91–99, Suntec, Singapore.

C.W. Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of ACL-04 Workshop: Text Summarization Branches Out*.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL 2010, pages 1552–1561, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *Proceedings of NAACL*.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadelphia, PA.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proceedings of the 2016 International Conference on Learning Representations*.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167:137–169.

Stéphane Ross and J. Andrew Bagnell. 2014. Reinforcement and imitation learning via interactive no-regret learning. *CoRR*, abs/1406.5979.

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

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

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, September.

Product Review Summarization by Exploiting Phrase Properties

Naitong Yu, Minlie Huang, Yuanyuan Shi*, Xiaoyan Zhu
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Dept. of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China
*Samsung R&D Institute China - Beijing
ynt12@mails.tsinghua.edu.cn, aihuang@tsinghua.edu.cn
yy.shi@samsung.com, zxy-dcs@tsinghua.edu.cn

Abstract

We propose a phrase-based approach for generating product review summaries. The main idea of our method is to leverage phrase properties to choose a subset of optimal phrases for generating the final summary. Specifically, we exploit two phrase properties, *popularity* and *specificity*. *Popularity* describes how popular the phrase is in the original reviews. *Specificity* describes how descriptive a phrase is in comparison to generic comments. We formalize the phrase selection procedure as an optimization problem and solve it using integer linear programming (ILP). An aspect-based bigram language model is used for generating the final summary with the selected phrases. Experiments show that our summarizer outperforms the other baselines.

1 Introduction

With the growth of the Internet over the decades, e-commerce is becoming more and more popular. Product reviews are helpful for both merchants and customers. Merchants analyze the reviews to get feedback to improve their products. Customers make use of the reviews to get a better understanding of the product. The opinions in the reviews can help them make the final decision. However, the vast availability of such reviews becomes overwhelming to users when there is just too much to digest. Product review summarization is the task to address this problem. It summarizes the large number of reviews and generates a short readable summary which contains the overall rating of the opinions in the reviews.

Traditional extractive summarization has been studied for a long time, such as (Hovy and Lin, 1999; Kupiec et al., 1995; Paice, 1990). Recently, there are also a number of studies on abstractive summarization, such as (Banerjee et al., 2015; Bing et al., 2015; Liu et al., 2015). However, applying traditional summarization methods directly on product reviews doesn't yield satisfying results. This is due to that product review summarization is quite different from traditional extractive summarization. From the perspective of data size, the number of reviews of a product is often much larger than that of traditional data such as news articles. Another important difference is that sentences in product reviews are usually colloquial and contain lots of noises. Directly extractive summaries may contain a large number of undesired information.

A number of researchers have studied the task of review summarization. (Ganesan et al., 2010) proposed a graph-based method for generating ultra concise opinion summaries of products. They used predefined rules for finding valid sub-paths in the graph and converted those sub-paths into sentences. Since the sentence generation was rule-based, their method didn't provide a well-formed grammatical summary. (Gerani et al., 2014) generated product review summaries by using discourse structure. After simplifying the discourse graph, they used a template-based NLG framework to generate natural language summaries. Their summary produced a statistical overview of the product but lacked detailed information. (Ganesan et al., 2012) proposed some heuristic rules to generate phrases, they used a modified mutual information function and an n-gram language model to ensure the representativeness and readability of the phrases. However, their method didn't consider the descriptiveness of the phrases.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,
pages 1113–1124, Osaka, Japan, December 11-17 2016.

We propose a phrase-based approach for generating product review summaries. We provide users with information that cover the most popular opinions in the original reviews targeting at each aspect. We use phrases as the basic unit of our summary, instead of sentences. We adopt the phrase definition in (Lu et al., 2009), that each phrase is composed by a pair of head term and modifier. The head term of a phrase denotes an *aspect* of the product, and the modifier denotes the *opinion* towards the aspect. For example, a phrase about the screen of a cellphone, "stunning [*modifier*] screen [*head*]". Based on the structure of phrases, we define two phrase properties, *popularity* and *specificity*. *Popularity* models how popular the phrase is in the original reviews. *Specificity* models how descriptive the modifier is to the head term. These two properties indicate the most important features of phrases in a good summary. We formalize this problem as an optimization problem and solve it using integer linear programming (ILP). A bigram aspect-based language model is used to order the selected phrases by aspects to form the final summary.

To summarize, our contributions are as follows:

- We propose a phrase-based approach for generating product review summaries. Our method leverages phrase properties, i.e., specificity and popularity, to choose popular and descriptive phrases from the original reviews.
- We formalize the summarization task as an optimization problem, and solve it using integer linear programming (ILP).
- We evaluate our summarization algorithm with both preference evaluation and qualitative evaluation. Our system performs better than other baselines in both evaluations.

The rest of this paper is organized as follows: In Section 2, we will present our phrase-based review summarization algorithm. In Section 3, we will describe the dataset and experiment results. In Section 4, we will describe the related work. In Section 5, we will summarize our work.

2 Summarization Algorithm

Our summarization algorithm takes a set of reviews of one product and a set of aspects as input and generates a summary based on the properties of the phrases extracted from the input reviews. The first step is phrase extraction. Phrases are extracted from the reviews using a given list of aspects. There are various methods for extracting aspects (Hu and Liu, 2004a; Hu and Liu, 2004b; Kim et al., 2011; Lu et al., 2009). In this paper, we do not focus on aspect extraction but consider them as the input. Each of the extracted phrases is tagged with its corresponding sentiment orientation. The second step is optimal phrase selection. We calculate properties of the phrases and select a subset of optimal phrases for constructing the final summary. We formalize this selection problem as an optimization problem and solve it using integer linear programming (ILP). The third step is aspects ordering. We use an aspect-based bigram language model to decide the order of the aspects in the final summary. In the last step, summary generation, phrases are filled into their corresponding aspect placeholders to form the final summary. The summarization framework is shown in Figure 1.

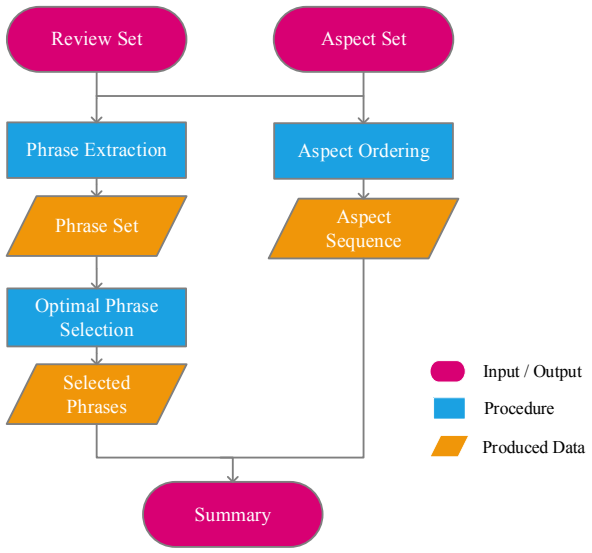


Figure 1: The overall framework of our summarization algorithm.

2.1 Phrase Extraction

In product reviews, most opinions are expressed in concise phrases, such as "camera is excellent" or "stunning screen". We adopt the phrase definition in (Lu et al., 2009), that each phrase can be parsed into a pair of a head term and a modifier. [1]

Aspect. An *aspect* denotes some specific feature of the product. For each aspect, there is a set of *aspect keywords* describing the corresponding aspect.

For example, available aspects of cell phones may include "appearance", "screen", "battery", etc. The aspect keywords set of "appearance" may include "appearance", "design", "surface", etc. Each keyword in the same keywords set is describing the same aspect.

Phrase. A phrase $p = (w_h, w_m)$ is in the form of a pair of a head term w_h and a modifier w_m. The head term is an aspect keyword of the product and the modifier expresses some opinion towards the aspect.

For example, "camera [*head*] is excellent [*modifier*]" and "stunning [*modifier*] screen [*head*]".

Phrase extraction is based on lexical and syntactic rules. First we perform part-of-speech (POS) tagging on the reviews.[2] Then we extract phrases from the reviews with Algorithm 1.

The input of the algorithm is the reviews of one product, denoted as R, and the keywords of all aspects, denoted as K. The output of the algorithm is a list of phrases denoted as P, with the corresponding indexes of the reviews denoted as $Index$, from which each phrase is extracted.

Algorithm 1 Phrase Extraction

Input:
Reviews of one product, $R = \{r_i\}_{i=1}^{n}$
Keywords, $K = \{k_j\}_{j=1}^{m}$
Output:
Phrases P and the corresponding indexes $Index$

```
1:  for each r_i in R do
2:    for each k_j in K do
3:      if ExistKeyword(k_j, r_i) then
4:        w_h ← k_j
5:        pos ← GetPosition(k_j, r_i)
6:        if ExistModifier(r_i, pos) then
7:          w_m ← GetModifier(r_i, pos)
8:          p ← (w_h, w_m)
9:          P ← P + {p}
10:         Index ← Index + {i}
11:       end if
12:     end if
13:   end for
14: end for
```

(1) For each review in R, first we check whether there are any aspect keywords in the review. If any aspect keywords are found (Line 3), then for each keyword found in the review, we set w_h as the aspect word (Line 4).

(2) From the position where we found the aspect word (Line 5), we do a forward and backward search to find the nearest adjective word. If the adjective word is found (Line 6), then set w_m as the adjective word (Line 7). If the adjective word is modified by an adverb, then the adjective word along with the adverb become the modifier w_m. If the adjective word is modified by a negative word, the negative word is also included in the modifier. This is handled by the function $GetModifier(r_i, pos)$. For example, in the phrase "screen/n is/v not/adv very/adv clear/adj", the modifier w_m is "not very clear".

(3) If both the head term w_h and the modifier w_m are found, a phrase $p = (w_h, w_m)$ is extracted, along with the index of the corresponding review (Line 8 - 10).

2.2 Optimal Phrase Selection: Definitions

In this section, we select a subset of optimal phrases from the phrase set. The subset should best represents the overall opinions expressed in the original reviews.

It is intuitive that a phrase is more likely to be included in a summary if it represents most of the users' opinion. For example, if there are 75% of the reviews containing the phrase "camera is excellent" while

[1] We demonstrate our summarization algorithm with running examples in English, but the datasets we use in our experiments are in Chinese.

[2] We use THULAC (`http://thulac.thunlp.org/`) as the POS tagging tool.

there are only 15% of the reviews containing other phrase "photo quality is very bad", then we should choose the former one as a candidate, because it is more popular in the original reviews.

On the other hand, for phrases describing the same aspect, we prefer the one whose modifier describes its head term more descriptive, i.e., the one which is more specific. For example, there are two phrases about the same aspect: "screen is clear" and "screen is good", we prefer "screen is clear" because that the modifier "clear" is more specific and better expresses the characteristic of the aspect "screen" while the modifier "good" is more general and can be used to describe other aspects.

A phrase is considered to be a candidate of the summary if it is popular in the original reviews and its modifier is specific to its head term. To better describe the phrase properties proposed above, we give the definition of *popularity* and *specificity* formally.

Definition 1 (Popularity). For a phrase p, let R_p denote the set of reviews that contain p, let R_{all} denote the set of all reviews. The *popularity* of phrase p is defined as:

$$\text{Popularity}(p) = \frac{|R_p|}{|R_{all}|} \tag{1}$$

where $|R|$ is the size of the review set R.

For a phrase set P, $p_i \in P$, $i = 1, \ldots, n$, let R_{p_i} denote the set of reviews that contain p_i, let R_{all} denote the set of all reviews. $\text{Popularity}(P) = |\cup R_{p_i}|/|R_{all}|$.

Suppose that we want to calculate the popularity of the phrase "long battery". Let's say there are 120 reviews in total and 25 of them contain the phrase "long battery", then the popularity of the phrase is $\text{Popularity}(\text{"long battery"}) = 25/120 = 0.21$.

Definition 2 (Specificity). For a phrase $p = (w_h^p, w_m^p)$, w_h^p denotes the aspect keyword of p, and A_p denotes the aspect that w_h^p belongs to, i.e., $w_h^p \in A_p$. w_m^p denotes the modifier of p. $P_{w_m = w_m^p}$ denotes the set of phrases whose modifier $w_m = w_m^p$, and $P_{w_h \in A_p, w_m = w_m^p}$ denotes the set of phrases whose head term $w_h \in A_p$ and modifier $w_m = w_m^p$. The *specificity* of phrase p is defined as:

$$\text{Specificity}(p) = \frac{|P_{w_h \in A_p, w_m = w_m^p}|}{|P_{w_m = w_m^p}|} \tag{2}$$

For a phrase set P, $p_i \in P, \quad i = 1, \ldots, n$, $\text{Specificity}(P) = \sum_i \text{Specificity}(p_i)$.

For example, suppose that we want to calculate the specificity of the phrase "beautiful design". The head term of this phrase is "design" and it belongs to the aspect *appearance*. The modifier term of this phrase is "beautiful". Let's say that there are 50 phrases whose modifier is "beautiful" in total, and there are 42 phrases whose modifier is "beautiful" and whose head term belong to the aspect *appearance*, then the specificity of the phrase is $\text{Specificity}(\text{"beautiful design"}) = 42/50 = 0.84$.

2.3 Optimal Phrase Selection: Problem Formalization

To select the optimal subset of phrases, we combine popularity and specificity to form an optimization problem. We use an integer linear programming (ILP) library[3] to solve this problem. We maximize $\text{Popularity}(P)$ and $\text{Specificity}(P)$ of a phrase set P together with the following constraints:

- **Length Constraint:** The total length of the summary is no longer than L_s.
- **Aspect Constraint:** For each aspect, the number of phrases in the cluster is no more than L_a.
- **Consistency Constraint:** For phrases in the same aspect, the sentiment orientation of these phrases should agree with each other.

To define the problem formally, let p_i denote the ith phrase in the phrase set P_{all}, and let r_j denote the jth review in the review set R_{all}. Let x_i represent a binary variable, that can take 0 or 1, depending on

[3] `http://sourceforge.net/projects/lpsolve/`

whether the ith phrase is selected for the final summary or not, and let y_j also represent a binary variable, that denotes whether the jth review is selected or not. Let P_{sel} denotes the set of phrases which are selected for the final summary. The objective function can be denoted as:

$$\begin{aligned} F(x_1, \ldots, x_n, y_1, \ldots, y_m) &= \text{Specificity}(P_{sel}) + \text{Popularity}(P_{sel}) \\ &= \sum_i \text{Specificity}(p_i) \cdot x_i + \frac{1}{|R_{all}|} \sum_j y_j \end{aligned} \tag{3}$$

The length constraint can be denoted as:

$$\sum_i l(p_i) \cdot x_i \leq L_s \tag{4}$$

where $l(p_i)$ denotes the length of phrase p_i. This constraint limits the total length of the summary to be no longer than L_s.

The aspect constraint can be denoted as:

$$\sum_{p_i \in P_{A^k}} x_i \leq L_a, \quad \forall A^k \tag{5}$$

where A^k denotes the kth aspect, and P_{A^k} denotes the set of phrases whose head term $w_h \in A^k$. These constraints limit the phrase number of each aspect to be no more than L_a.

The consistency constraint can be denoted as:

$$\left| \sum_{p_i \in P_{A^k}} o(p_i) \cdot x_i \right| = \sum_{p_i \in P_{A^k}} x_i, \quad \forall A^k \tag{6}$$

where $o(p_i)$ denotes the sentiment orientation of phrase p_i. $o(p_i) = 1$ if the sentiment orientation of phrase p_i is positive, and $o(p_i) = -1$ if the sentiment orientation of phrase p_i is negative. These constraints ensure that phrases in the same aspect have the same sentiment orientation.

There are other constraints to ensure the consistency between the phrase set and review set:

$$x_i \cdot Occ_{i,j} \leq y_j, \quad \forall i, j \tag{7}$$

$$\sum_i x_i \cdot Occ_{i,j} \geq y_j, \quad \forall j \tag{8}$$

where $Occ_{i,j}$ is a binary value, $Occ_{i,j} = 1$ if and only if phrase p_i is in review r_j, i.e., $p_i \in r_j$. Equation (7) means that if phrase p_i is selected ($x_i = 1$), then any review r_j that $p_i \in r_j$ is also selected ($y_j = 1$). Equation (8) means that if review r_j is selected ($y_j = 1$), then at least one phrase p_i that $p_i \in r_j$ is selected ($x_i = 1$).

2.4 Aspects Ordering

When writing comments, customers tend to first mention the aspect they care about most. We determine the aspect order by finding the most common aspect sequence in the original reviews. By this means, the generated summary would be more natural and coherent to human-generated reviews. Since we are only interested in the relative order of two adjacent aspects, we use an aspect-based bigram language model that assign probabilities to sequence of aspects.

For each review r, let $\Gamma(r)$ denote a function that maps each review to its corresponding aspects sequence, i.e., $\Gamma(r) = s_r$, where $s_r = [a_1^r, a_2^r, \ldots, a_q^r]$ is a sequence of aspects, and a_i^r is the ith aspects in the review r. Let $S = \{\Gamma(r) | r \in R\}$ denote the set of all aspect sequences for all reviews, let A denote the available aspect set, for any $a_i, a_j \in A$, $s_k \in S$,

$$p(a_i | a_j) = \frac{\sum_k I(a_i a_j \in s_k)}{\sum_k I(a_j \in s_k)} \tag{9}$$

where $I(x) = 1$ if x is true, otherwise $I(x) = 0$. For an arbitrary aspect sequence $s = a_1, a_2, \ldots, a_r$, the probability of the sequence is:

$$p(s) = p(a_1)p(a_2|a_1)\ldots p(a_r|a_{r-1}) \tag{10}$$

The objective sequence is the sequence that has the maximum probability in the language model and each aspect appears and only appears in the sequence once:

$$s^* = \arg\max p(s) \tag{11}$$

where $s = a_1, a_2, \ldots, a_n$, n is the number of aspects, $a_i \neq a_j, \forall i, j, i \neq j$.

2.5 Summary Generation

The summary generation procedure is quite straightforward. We sort the phrases in the optimal phrase set by the order of their corresponding aspects, and the final summary is a sequence of these phrases.

For phrases which have different aspects, the order of them is the same as the order of the corresponding aspects in the aspect sequence. For phrases which have the same aspect, the order is the same as the descending order of the size of the corresponding review set. Specially, if two phrases in the same aspect have the same head term, we merge their modifier term into one. For example, "screen is big" and "screen is clear" can be merged into "screen is big and clear".

Table 1 shows an example summary generated by our system.[4]

屏幕舒服、效果出色，电池耐用、续航长，做工精致，质量很棒，性能优越，速度快，性价比极高，价格适中，像素一般，摄像头很一般，外观漂亮、好看，软件太少、不丰富，界面十分简洁，画质清楚，操作简单、简洁，音效不错，音质特好，内存不够、较小，信号相当不错，通话声音清晰，按钮位置不好，机身重、太大。
Screen is comfortable and display is excellent. Battery is durable and lasts long. Exquisite workmanship and good quality. Performance is superior and fast. Price is cost-effective and affordable. Camera is very general. Appearance is beautiful and nice. Software is not rich. Interface is very simple. Picture is clear and of good quality. Operation is simple. Sound quality is especially good. Memory is not enough. Signal is quite good. Clear voice calls. Button location is not good. Body is heavy and too big.

Table 1: Example summary generated by our system.

3 Experiments

3.1 Data Preparation

Phrase Extraction. We construct our experiment dataset using customer reviews of 10 cellphones. The reviews are collected from `jd.com`, `zol.com` and `weibo.com`. All of the reviews are written in Chinese. We get 33,948 reviews in total. We construct 17 aspects manually, each aspect contains about 6 aspect keywords on average. With these aspect keywords, we perform phrase extraction on the review dataset. Each of the extracted phrases is then tagged with its corresponding sentiment orientation. In total, we have extracted 44,461 phrases, i.e., 1.31 phrases per review on average. 83% of the extracted phrases are positive, while 17% are negative.

Sentiment Detection. We train an SVM classifier to classify the sentiment orientations for the phrases. We use words as features and the corresponding values are binaries which indicate whether the words are present or not. First we drop words with frequency lower than 100, then we rank words by their chi-square values in descending order and choose the first third of all words as the feature words. We use `svmlight`[5] with default parameters to implement our classifier.

We use the review data for cell phones from `jd.com`. Since a single review may contain different sentiment orientations, we split each review into short sentences by a splitting delimiter set, which contains punctuations such as "，。；：！？ ". For each short sentence, we ask two annotators to judge the sentiment

[4]The summary is in Chinese and we translate it into English manually.

[5]`http://svmlight.joachims.org/`

orientation as "positive", "negative" or "neutral". Conflicting results are reviewed by a third annotator. We are only interested in short sentences with opinions, so we drop those tagged with "neutral". We get 182,120 short sentences in total, 72.5% are positive and 27.5% are negative. Since the same opinion word may have different orientations in different aspects, we cluster the short sentences by their aspects and train an SVM classifier for each aspect separately. For each of the 17 aspects, we randomly select 320 short sentences for testing, and the rest are used for training. The overall performance of sentiment classification is shown in Table 2. This result shows that out SVM classifier performs good enough for our review summarization task.

Precision		Recall		F1		Accuracy
Pos	Neg	Pos	Neg	Pos	Neg	All
91.8	91.2	92.3	90.7	92.0	90.9	91.5

Table 2: % of precision, recall, F1 and overall accuracy on sentiment classification

3.2 Baselines

The evaluation of review summarization is a very challenging task. On one hand, to the best of our knowledge, there is no dataset of product reviews with human written summaries. On the other hand, since the amount of product reviews is often large, it is quite difficult to generate human written summaries.

We evaluate the summaries generated by our system (denoted as **ReviewSum**) with a state-of-the-art extractive baseline, a state-of-the-art abstractive baseline and a simplified version of our system. The details of the baselines are described as follows:

1) LexRank: LexRank (Erkan and Radev, 2004) is a graph-based extractive summarization method which computes sentence importance based on the concept of eigenvector centrality in a graph representation of sentences. In the experiment, first we cluster sentences by their aspects. Then for each sentence cluster, LexRank is performed for summary generation. The final summary is generated by putting summaries of different aspects together in the same aspect order of ReviewSum.

2) Opinosis: Opinosis (Ganesan et al., 2010) is a novel graph-based summarization method which generates concise abstractive summaries of highly redundant opinions. In the experiment, for each aspect, we build an Opinosis graph and get the top candidate summaries. The final summary is generated by putting summaries of different aspects together in the same aspect order of ReviewSum.

3) BasicSum: BasicSum is a simplified version of our summarization method. Instead of popularity and specificity, TF-IDF score is used in the objective function. The objective function of BasicSum can be denoted as:

$$F(x_1, \ldots, x_n) = \sum_i \text{tf-idf}(p_i) \cdot x_i \tag{12}$$

where tf-idf(p_i) is the TF-IDF score of phrase p_i, and x_i is a binary value representing whether phrase p_i is selected in the final summary or not.

3.3 Experiments Evaluation

Due to the lack of human written summaries as gold standard, we perform two tasks to evaluate the summaries generated by our system. Task 1 is pairwise user preference evaluation and Task 2 is user scoring evaluation.

In Task 1, we run six pairwise comparisons of four summaries generated by our method and baselines. For each comparison, two summaries of the same product are shown to the annotators in random order. The name of the product and the original reviews are also shown to the annotators. For two summaries S_1 and S_2, annotators need to make a choice in the following three options: 1) Prefer S_1, 2) Prefer S_2, 3) No preference. Note that the exact names of S_1 and S_2 are hidden to annotators.

In order to ensure the quality of the evaluation, annotators are instructed to read the original reviews first before they make their choice. Annotators are specially instructed that their choice should based

on "*overall satisfaction with the information provided by the summary and intuitive feelings about the summary*".

In Task 2, we ask annotators to evaluate four aspects of each summary. The aspects considered during the evaluation include Grammaticality, Non-Redundancy, Consistency and Descriptiveness. Each aspect is rated with a score from 0 (bad) to 10 (excellent). Annotators are instructed to read the summary carefully and rate each aspect with scores matching the quality of the corresponding aspect.

20 annotators participate in Task 1 and Task 2, 10 annotators for each task. All of them are native Chinese speakers with experiences of product review writing. In Task 1, each comparison is evaluated by at least 5 annotators, and more than 300 comparison results are generated. In Task 2, each summary is rated by at least 5 annotators, and more than 160 rated scores are generated.

3.4 Results and Discussions

Sys I	**Sys II**	No pref	Pref Sys I	Pref Sys II	Agreement
BasicSum	LexRank	8%	42%	50%	0.2
BasicSum	Opinosis	2%	22%	76%	0.5
LexRank	Opinosis	12%	32%	56%	0.6
LexRank	ReviewSum	8%	14%	**78%**	0.8
BasicSum	ReviewSum	14%	0%	**86%**	0.8
Opinosis	ReviewSum	8%	18%	**74%**	0.6

Table 3: Results of pairwise comparison preferences. Statistically significant improvements ($p < 0.01$) over the baselines are demonstrated by bold fonts.

Preference Evaluation. Table 3 shows the results of Task 1. The first two columns denote systems compared in each comparison. The following three columns indicate the percentage of preference decisions for each preference category. Statistically significant improvements ($p < 0.01$) of our system over the baselines are demonstrated in bold fonts. The last column indicates the agreement rate of preference comparisons for different systems. Specifically, in our experiments, we treat one pairwise comparison as in agreement if four (out of five) annotators give the same preference decision. Table 4 shows system preference results of each product. System preferences are computed based on the results of pairwise comparison. For each system, the preference is the number of times annotators prefer the system, divided by the total number of comparisons for the system. For example, we have three systems A, B and C. A is preferred over B 10 out of 20 times, and A is preferred over C 25 out of 30 times, then the overall preference of A is $(10 + 25)/(20 + 30) = 70\%$.

Products	BasicSum	LexRank	Opinosis	ReviewSum
Galaxy S5	7%	27%	60%	**87%**
iPhone 5S	20%	40%	53%	**80%**
iPhone 5C	7%	20%	67%	**87%**
Ascend P7	20%	33%	60%	**80%**
Lumia 1320	33%	60%	13%	60%
Sony l36h	33%	20%	40%	**80%**
HTC One	27%	53%	20%	**93%**
LG G2	20%	27%	67%	67%
Galaxy Note 4	13%	33%	67%	**80%**
Galaxy Grand 2	33%	7%	53%	**80%**
Total	21%	32%	50%	**79%**

Table 4: System preference results of each product. Statistically significant improvements ($p < 0.01$) over the baselines are demonstrated by bold fonts.

From Table 3 and Table 4 we can see that our system significantly outperforms BasicSum. The only difference between BasicSum and our system is the objective function. Our system uses popularity and specificity of the phrases while BasicSum uses TF-IDF score. The result shows that popularity and specificity can prominently improve the quality of the summary. In fact, popularity and specificity improve the descriptiveness of the summary significantly, which we will discuss later.

The results in Table 3 and Table 4 show statistically significant improvements in pairwise comparisons of our system over the extractive baseline (LexRank) and the abstractive baseline (Opinosis). Due to the limitations of sentence-based extractive models, summaries produced by LexRank contain long sentences with useless information, while our system produces phrase-based summaries without unwanted information. Opinosis produces much shorter and concise summaries than LexRank, but the grammar of the sentences are not very well. In our method, phrases are generated in a concise manner by joining aspect and opinion together. The generated summaries are clear and well-formatted.

Qualitative Evaluation. Table 5 shows the results of Task 2. In order to avoid scoring varies per person, rating scores are normalized by each annotator, i.e., for each annotator, scores in range $[s_{min}, s_{max}]$ are remapped into range $[0, 10]$. The first column denotes systems in the rating task, the following four columns denote average scores of each system in four different aspects: grammaticality, non-redundancy, consistency and descriptiveness.

Systems	Grammaticality	Non-Redundancy	Consistency	Descriptiveness
BasicSum	6.46	4.89	6.92	3.42
LexRank	4.13	5.31	6.56	5.54
Opinosis	5.83	6.38	7.58	6.17
ReviewSum	**6.87**	6.07	7.41	**8.15**

Table 5: Results of aspects rating scores.

The results in Table 5 show that our system achieves the best score in grammaticality and descriptiveness. This exactly matches what we expect from our method that outputs well-formatted summaries by choosing neat and descriptive phrases. Also, our system is doing better than BasicSum and LexRank in non-redundancy. TF-IDF scores are used in BasicSum for phrase selection, and this will cause phrases with similar opinion words being selected (such as *good*, *very good*, etc.), which results in the increase of redundancy. LexRank extracts sentences directly from reviews, and information redundant may also be included. In consistency, our system performs nearly as good as Opinosis.

4 Related Work

Our work is related to aspect-based opinion summarization, which can be divide into three distinct steps: aspect extraction, sentiment detection and summary generation.

Aspect extraction involves identifying salient aspects within the text to be summarized. (Lu et al., 2009) used shallow parsing to identify aspects for short comments. (Popescu and Etzioni, 2007) used a web-based domain-independent information extraction system to extract aspects from parsed review data. (Hu and Liu, 2004a) and (Hu and Liu, 2004b) used supervised association rule mining-based approach to perform the task of aspect extraction. (Zhuang et al., 2006) used a feature list combining the full cast of all movies and a regular expression set to identify features in movie reviews. (Ku et al., 2006) used paragraph level frequencies as well as document level ones to help identify features.

Sentiment detection is the task of detecting sentiment orientation (positive or negative) on the aspect or feature. (Lu et al., 2009) used a learning-based method for sentiment detection. (Hu and Liu, 2004a; Hu and Liu, 2004b) used an effective method based on WordNet. (Ku et al., 2006) also used a set of positive and negative words from GI and CNSD to predict sentiments of aspects. (Zhuang et al., 2006) used dependency relationships to identify opinions associated with feature words.

Summary generation involves aggregating the results of aspect extraction and sentiment detection and generate the final opinion summary in an effective and easy to understand format. Statistical summary

is the most commonly adopted format, such as (Hu and Liu, 2004a; Hu and Liu, 2004b; Hu and Liu, 2006; Zhuang et al., 2006). (Titov and McDonald, 2008b) used a topic modeling method to provide a word level summary for each topic. (Popescu and Etzioni, 2007) also provided a word level summary by ranking opinion words associated to features and showing the strongest opinionated word for each aspect. (Mei et al., 2007) scored the probability of each sentence using TSM model and generated a sentence level summary. (Ku et al., 2006) used TF-IDF to score sentences and select the most relevant and discriminative sentence to be shown as summary. Besides texts, aggregated ratings can also be shown for summary, such as (Lu et al., 2009). (Ku et al., 2006) and (Mei et al., 2007) showed summary as a timeline with opinion changes over time.

5 Conclusion

In this paper, we propose a phrase-based summarization algorithm for the task of product review summarization. The proposed phrase selection scheme can fully utilize the characteristics of review sentences and capture the main information. We propose two properties of phrases, popularity and specificity, to score the phrase. Integer linear programming (ILP) is used to optimize the objective function. We use an aspect-based bigram language model to determine the aspect order of the candidate phrases to make the generated summaries more fluent and natural. Experimental results show that our system outperforms state-of-the-art systems in most cases. Our proposed summarization algorithm can produce concise, well-formatted and descriptive summaries of product reviews.

Acknowledgements

This work was partly supported by the National Basic Research Program (973 Program) under grant No. 2013CB329403, the National Science Foundation of China under grant No.61272227/61332007. The work was also supported by Tsinghua University Beijing Samsung Telecom R&D Center Joint Laboratory for Intelligent Media Computing.

References

Alexandra Balahur and Andrés Montoyo. 2008. Multilingual feature-driven opinion extraction and summarization from customer reviews. In *NLDB*, volume 5039, pages 345–346. Springer.

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *24th International Joint Conference on Artificial Intelligence (IJCAI). Buenos Aires, Argentina: AAAI press*.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging.

Giuseppe Carenini, Jackie Chi Kit Cheung, and Adam Pauls. 2013. Multi-document summarization of evaluative text. *Computational Intelligence*, 29(4):545–576.

Kushal Dave, Steve Lawrence, and David M Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528. ACM.

Giuseppe Di Fabbrizio, Amanda J Stent, and Robert Gaizauskas. 2014. A hybrid approach to multi-document summarization of opinions in reviews. *INLG 2014*, page 54.

Yajuan Duan, Zhimin Chen, Furu Wei, Ming Zhou, and Heung-Yeung Shum. 2012. Twitter topic summarization by ranking tweets using social influence and content quality. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 763–780.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, pages 457–479.

Dimitrios Galanis, Gerasimos Lampouras, and Ion Androutsopoulos. 2012. Extractive multi-document summarization with integer linear programming and support vector regression. In *COLING*, pages 911–926. Citeseer.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING '10)*.

Kavita Ganesan, ChengXiang Zhai, and Evelyne Viegas. 2012. Micropinion generation: an unsupervised approach to generating ultra-concise summaries of opinions. In *Proceedings of the 21st international conference on World Wide Web*, pages 869–878. ACM.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1602—1613.

Eduard Hovy and Chin-Yew Lin. 1999. Automated text summarization in summarist. In *Advances in Automatic Text Summarization*. MIT Press.

Minqing Hu and Bing Liu. 2004a. Mining opinion features in customer reviews. In *AAAI*, volume 4, pages 755–760.

Minqing Hu and Bing Liu. 2004b. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Minqing Hu and Bing Liu. 2006. Opinion extraction and summarization on the web. In *AAAI*, volume 7, pages 1621–1624.

Hyun Duk Kim, Kavita Ganesan, Parikshit Sondhi, and ChengXiang Zhai. 2011. Comprehensive review of opinion summarization.

Lun-Wei Ku, Yu-Ting Liang, and Hsin-Hsi Chen. 2006. Opinion extraction, summarization and tracking in news and blog corpora. In *AAAI spring symposium: Computational approaches to analyzing weblogs*, volume 100107.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 68–73. ACM.

Kevin Lerman, Sasha Blair-Goldensohn, and Ryan McDonald. 2009. Sentiment summarization: evaluating and learning user preferences. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 514–522. Association for Computational Linguistics.

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In *Proceedings of the 23rd international conference on computational linguistics*, pages 653–661. Association for Computational Linguistics.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086.

Yue Lu, ChengXiang Zhai, and Neel Sundaresan. 2009. Rated aspect summarization of short comments. In *Proceedings of the 18th international conference on World wide web*, pages 131–140. ACM.

Rebecca Mason, Benjamin Gaska, Benjamin Van Durme, Pallavi Choudhury, Ted Hart, Bill Dolan, Kristina Toutanova, and Margaret Mitchell. 2016. Microsummarization of online reviews: An experimental study. Association for the Advancement of Artificial Intelligence.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. 2007. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th international conference on World Wide Web*, pages 171–180. ACM.

Chris D Paice. 1990. Constructing literature abstracts by computer: techniques and prospects. *Information Processing & Management*, 26(1):171–186.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Ana-Maria Popescu and Orena Etzioni. 2007. Extracting product features and opinions from reviews. In *Natural language processing and text mining*, pages 9–28. Springer.

Benjamin Snyder and Regina Barzilay. 2007. Multiple aspect ranking using the good grief algorithm. In *HLT-NAACL*, pages 300–307.

Maosong Sun, Xinxiong Chen, Kaixu Zhang, Zhipeng Guo, and Zhiyuan Liu. 2016. Thulac: An efficient lexical analyzer for chinese.

Ivan Titov and Ryan McDonald. 2008a. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th international conference on World Wide Web*, pages 111–120. ACM.

Ivan Titov and Ryan T McDonald. 2008b. A joint model of text and aspect ratings for sentiment summarization. In *ACL*, volume 8, pages 308–316. Citeseer.

Lu Wang, Hema Raghavan, Claire Cardie, and Vittorio Castelli. 2014. Query-focused opinion summarization for user-generated content. In *COLING*, pages 1660–1669.

Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. 2015. Phrase-based compressive cross-language summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 118–127. Association for Computational Linguistics.

Renxian Zhang, Wenjie Li, and Dehong Gao. 2012. Generating coherent summaries with textual aspects. In *AAAI*.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 43–50. ACM.

Generating Questions and Multiple-Choice Answers using Semantic Analysis of Texts

Jun Araki, Dheeraj Rajagopal, Sreecharan Sankaranarayanan,
Susan Holm, Yukari Yamakawa, Teruko Mitamura
Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
junaraki@cs.cmu.edu, dheeraj@cs.cmu.edu, sree@cmu.edu,
sh4s@andrew.cmu.edu, yukariy@andrew.cmu.edu, teruko@cs.cmu.edu

Abstract

We present a novel approach to automated question generation that improves upon prior work both from a technology perspective and from an assessment perspective. Our system is aimed at engaging language learners by generating multiple-choice questions which utilize specific inference steps over multiple sentences, namely coreference resolution and paraphrase detection. The system also generates correct answers and semantically-motivated phrase-level distractors as answer choices. Evaluation by human annotators indicates that our approach requires a larger number of inference steps, which necessitate deeper semantic understanding of texts than a traditional single-sentence approach.

1 Introduction

Exam questions are an indispensable tool for teachers to assess their students' understanding of material. Thus, automatic question generation from text is a key natural language processing technology to aid teachers in examining learners' reading comprehension. Past studies in education showed that higher-level questions, in contrast to simple factoid questions, have more educational benefits for reading comprehension (Anderson and Biddle, 1975; Andre, 1979; Hamaker, 1986). However, most of existing approaches to question generation have focused on generating questions from a single sentence, relying heavily on syntax and shallow semantics with an emphasis on grammaticality (Mitkov and Ha, 2003; Chen et al., 2009; Heilman and Smith, 2010; Curto et al., 2011; Becker et al., 2012; Lindberg et al., 2013; Mazidi and Nielsen, 2014). A problem with this approach is that the majority of questions generated from single sentences tend to be too specific and low-level to properly measure learners' understandings of the overall contents of text. In other words, what is assessed by such question generation systems ends up essentially being the ability to compare sentences, just requiring learners to find a single sentence that has almost the same surface form as a given interrogative sentence. Results of simple sentence comparisons do little to contribute towards the goal of assessing learners' reading comprehension.

In this work, we propose a question generation approach that engages learners through the use of specific *inference steps* over multiple sentences, namely coreference resolution and paraphrase detection, requiring more semantic understanding of text. We primarily use event and entity coreference as a source of producing questions from multiple sentences. Grounded by the past studies in education, we believe that such high-level questions are more sophisticated and educationally valuable for testing reading comprehension than questions generated by the traditional single-sentence approaches. Our question generation strategy is novel in two ways. First, it employs event and entity coreference between antecedents and referring mentions spanning multiple sentences. This requires learners to resolve the coreference and understand the contents of the text semantically. Second, it makes use of paraphrases when generating questions. The resulting questions are able to check learners' lexical knowledge.

To ensure that the resulting multiple choice question is an adequate test of the learner's reading comprehension, incorrect answers, i.e., distractors are necessary. The distractor candidates must be selected to ensure difficulty (Candidate Selection) and compared against the correct answer to ensure that there is only one correct answer overall and the question remains solvable (Reliability Checking). Most systems

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1125–1136, Osaka, Japan, December 11-17 2016.

currently generate word-level distractors for testing grammar, and are intimately tied to the question generation method. Our distractor generation method is capable of automatically generating phrase-level distractors leveraging event triggers and event-event relations for the purpose of testing reading comprehension. The results of our experiments show that our method produces comparable results across two different methods of question generation, thus remaining robust to different question generation methods.

Our question generation system is aimed at enhancing the reading comprehension ability of language learners, more specifically, students who learn English as a second language (ESL). Therefore, our ultimate goal is to generate multiple-choice questions from plain texts in an arbitrary domain. However, the state-of-the-art in extracting semantic representations of event and entity relations from text does not perform well enough to support our question generation approach. Thus, the evaluation of question generation relying on automated semantic relation extraction is not practical at this time. Therefore, in this work we use texts and expert human annotations from the ProcessBank corpus[1] (Berant et al., 2014) to facilitate our semantics-oriented question generation.

Our contributions are as follows:

1. This is the first work to automatically generate questions from multiple sentences, involving specific inference steps such as coreference resolution and paraphrase detection. Our experiments show that questions generated by our approach require taking a larger number of inference steps while ensuring comparable grammatical correctness and answer existence, as compared to questions generated by a traditional single-sentence approach.
2. We also present a complementary system which generates questions based on patterns extracted from relationships between events and entities in the passage. This approach produces a higher number of questions without the need for any manual intervention for the generation of patterns, although less reliable compared to the previous question generation system.
3. To complement the complex questions generated from multiple sentences, our system also generates phrase level distractors aimed at challenging comprehension by using event-event relation annotations. The results of our experiments show that the quality of the generated distractors remains robust across two different methods of question generation.

2 Related Work

Among a considerable amount of prior work on question generation spanning nearly four decades, we focus mainly on two families: (1) *syntax-based transformation* and (2) *shallow semantics-based transformation*. Syntax-based transformation uses the output of syntactic parsers, and changes syntactic structures of sentences to convert them into interrogative sentences as questions (Wolfe, 1976; Mitkov and Ha, 2003; Judge et al., 2006; Heilman and Smith, 2010; Curto et al., 2011). In contrast, shallow semantics-based transformation uses the output of shallow semantic parsers and additional knowledge bases, and generates questions from sentences based on their semantic structures (Chen et al., 2009; Mannem et al., 2010; Becker et al., 2012; Yao et al., 2012; Mazidi and Nielsen, 2014). Both groups generate questions based on a single sentence of text, relying heavily on its argument structures in question construction and mostly emphasizing grammaticality in evaluation. Labutov et al. (2015) have recently presented an approach to generate high-level questions using ontology-derived templates. Our approach differs from theirs in that we leverage semantic representations of text to inject specific inference steps, such as event and entity coreferences, into the process of answering system-generated questions.

In the context of computer-assisted language learning, intelligent tutoring systems have been traditionally employed for generating questions in various domains (Koedinger et al., 1997; Vanlehn et al., 2005). Some of the research in this area uses questions as a way of engaging students in ongoing dialogue (Graesser et al., 2001; Piwek and Stoyanchev, 2010). Our underlying motivation to enhance the reading comprehension ability of non-native English readers aligns best with the Reader Specific Lexical Practice (REAP) system (Heilman et al., 2006) and the SmartReader system (Azab et al., 2013). Though very promising, state-of-the-art approaches in this area have not dealt with multiple sentences to generate questions, either. Our work is aimed at addressing this challenge.

[1] See Section 3.1 for details of this corpus.

Prior work in distractor generation has been mostly studied on cloze (gap-fill) questions. Distractor generation for cloze questions often comprises two steps. *Candidate Selection* controls the type and difficulty of the items, and is intimately tied to the intent and target audience for the questions. This step may take advantage of available datasets such as the Lang-8 corpus[2] to identify confusion pairs and use the most frequent learner confusions as distractors (Sakaguchi et al., 2013) or the set of English language prepositions for use as distractors in preposition testing tasks (Lee and Seneff, 2007). *Reliability Checking* ensures that the question remains solvable, i.e., there is a total of one correct answer only (Zesch and Melamud, 2014).

In most cases, however, only the question and the correct answer are available, and distractors have to be automatically generated using these as input. Random generation of distractors (Mostow and Jang, 2012), distractors with a corpus frequency comparable to the correct answer (Hoshino and Nakagawa, 2007), morphologically, orthographically or phonetically similar words as distractors (Pino and Eskenazi, 2009), and semantically similar words selected using taxonomies (Hoshino and Nakagawa, 2007; Mitkov et al., 2009), thesauri (Sumita et al., 2005; Smith et al., 2010) or an additional layering of semantic relatedness to the context (sentence or paragraph) (Pino and Eskenazi, 2009; Agarwal et al., 2011; Mostow and Jang, 2012) are all valid strategies for the generation of distractor items. However, most of these methods generate word-level distractors for testing grammar rather than comprehension. Moreover, distractor generation is often tied to the method of generating questions. Our method automatically generates phrase-level distractors for testing reading comprehension, and is decoupled from the method used for question generation.

3 Generating Questions and Multiple-Choice Answers

Our system comprises a question generation component and a distractor generation component. We show a high-level overview of the system in Figure 1. The question generation component first detects question targets, and then generates questions triggered by the targets as correct answers. This component implements two different template-based algorithms, as described in Section 3.2 and Section 3.3. Given a tuple (text, question, answer) as input, the distractor generation component generates n distractors per question. The resulting tuple (question, answer, distractors) forms a multiple-choice question as the final output of our system.

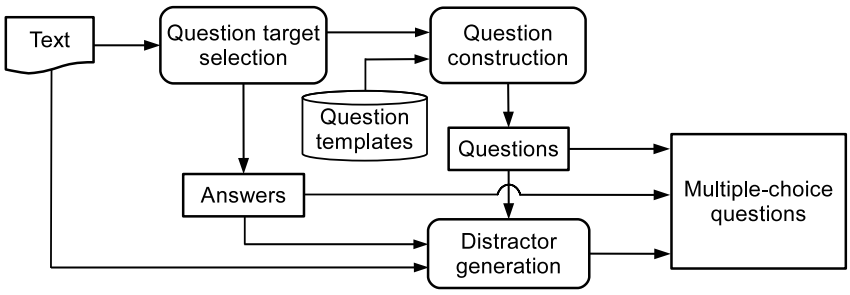


Figure 1: A high-level overview of our multiple-choice question generation system.

3.1 The ProcessBank Corpus

In order to generate questions from multiple sentences using reliable annotation of event and entity coreference, we use the ProcessBank corpus[3], particularly utilizing the expert annotation of events and entities, as described in Section 1. The corpus consists of 200 paragraphs about biological processes, extracted from the high school level textbook *Biology* (Campbell and Reece, 2005). Such textbooks are ideal sources to test learners' reading comprehension from an educational perspective. The corpus includes rich process structures annotated by biologists, shown in Figure 2. More specifically, the expert annotation comprises entity mentions, entity coreference, event triggers (without any event types), arguments (with semantic roles), and event-event relations such as event coreference and causal relations.

[2]A corpus of manually annotated errors by ESL learners, available at `http://cl.naist.jp/nldata/lang-8/`

[3]`http://www-nlp.stanford.edu/software/bioprocess/`

Each event in the corpus represents a biological process. A trigger is defined as a word or a phrase that expresses the process most clearly, and an argument is defined as a phrase denoting entities that participate in the process. The corpus also includes multiple-choice questions per paragraph, created by the biologists. We refer to these expert questions to devise our question templates.

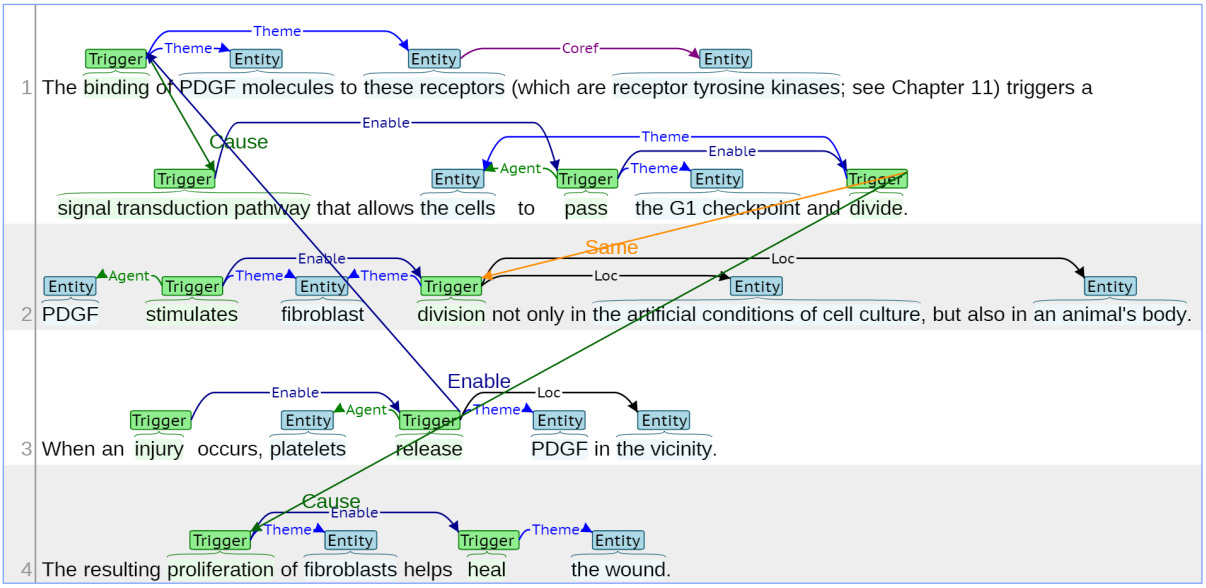


Figure 2: A paragraph with annotation of events, entities and their relations in ProcessBank. A 'Same' link means event coreference, whereas a 'Coref' link means entity coreference.

3.2 Question Generation using Coreferences and Paraphrases

Our first question generation system (QG1) is aimed at generating questions from multiple sentences using three semantic relations: event coreference, entity coreference, and paraphrases. We recognize that one of the key learning points for biology students is the order of biological processes (events) because many of the expert questions in ProcessBank ask about it. Based on this observation, we devise question patterns and templates shown in Table 1. As seen in this table, pattern P2 and P3 focus on the order of biological processes. However, the whole question generation strategy, including P2 and P3, does not rely on any domain-specific knowledge or rules on biology, but instead it makes use of domain-independent semantic relations, such as event coreferences and causal relations. Hence, the question generation strategy is not restricted to the biology domain, and is portable to other domains from an algorithm perspective.

QG1 consists of two parts: **answer generation** and **question construction**. It first finds question patterns applicable to the given text. For example, pattern P3 is applicable to the paragraph of Figure 2 since "divide" (E1) in the first sentence and "division" (E2) in the second sentence are coreferent, and only the former trigger has a 'Cause' relation to "proliferation" (E3). This pattern match means that "proliferation" can be an answer. We then make use of arguments of the answer trigger to generate a more complete answer, obtaining "proliferation of fibroblasts". Next, the algorithm constructs a question given the matched pattern. In the case of the example above, "division" is a nominal trigger. Thus, the algorithm uses question template T5, and creates a question "What is a result of the fibroblast division not only in the artificial conditions of cell culture, but also in an animal's body?" As seen in this example, the algorithm takes advantage of the fact that E2 lacks a piece of information that E1 has in P1, P2 and P3. We only use event coreference E1-E2 where E1 and E2 exist in different sentences, ensuring that questions are generated from multiple sentences.

We also give examples to illustrate how to generate questions based on entity coreferences and paraphrases in Figure 3 and Figure 4, respectively. Pattern P4 applies to Figure 3 because "they" (En1) in the second sentence is coreferent with "a few tumor cells" (En2) in the first sentence, and the former

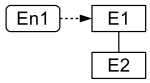
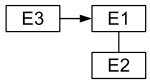
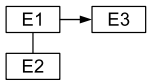
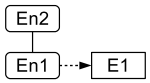
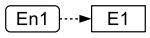
Semantic relation	Question patterns	Answer	Question templates
Event coreference	P1. 	En1	T1. What [*verbal trigger + subsequent arguments*]?
	P2. 	E3	T2. What causes [*nominal trigger + subsequent arguments*]? T3. What makes it happen to [*verbal trigger + subsequent arguments*]? T4. What makes it happen that [*event clause*]?
	P3. 	E3	T5. What is a result of [*nominal trigger + subsequent arguments*]? T6. What happens when [*event clause*]?
Entity coreference	P4. 	En2	T1. What [*verbal trigger + subsequent arguments*]?
Paraphrase	P5. 	En1	

Table 1: Question patterns and templates using event coreference, entity coreference, and paraphrases. In question patterns, En denotes an event trigger, and Enn an entity mention. A straight line denotes a coreference link, a dashed arrow an 'Agent' relation, and a straight arrow a relation which is 'Cause', 'Enable' or 'Result'. An event clause in question templates is defined as a text span including an event trigger and its arguments.

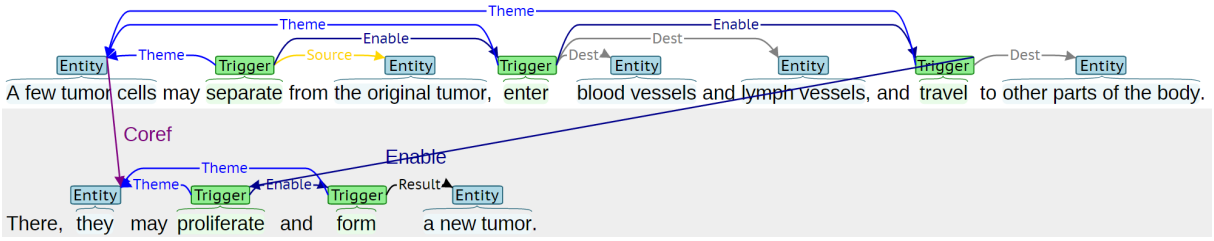


Figure 3: An example text to generate a question using an entity coreference.

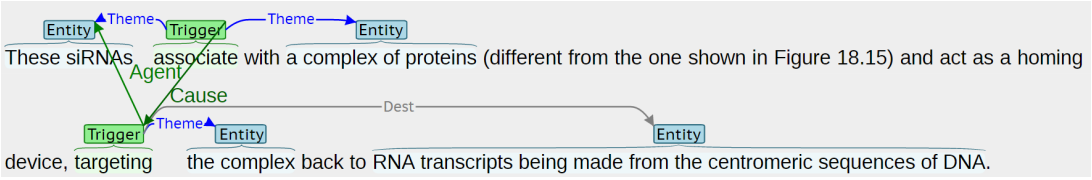


Figure 4: An example text to generate a question using a paraphrase.

entity is an agent of triggers "proliferate" and "form". Given that, "a few tumor cells" can be an answer, and the system can generate a question "What may proliferate and form a new tumor?" Similarly, P5 applies to Figure 4 because "these siRNAs" are an agent of trigger "targeting", and "complex" has a paraphrase "composite". Since "these siRNAs" are coreferent with "siRNAs" in the previous sentence (not appearing in Figure 4), a system can generate an answer "siRNAs" and a question "What targets the composite back to RNA transcripts being made from the centromeric sequences of DNA?" Note that the paraphrase "composite" is inserted into the question.

3.3 Question Generation using Concept Relationships

Our alternate question generation system (QG2) uses question patterns from events and entities in the passage and their relations. In this system, we rely on the fact that we can extract the question patterns based on the relations between entities and events in a passage. The relations between these events and entities span across multiple sentences, and thus most questions are generated from multiple sentences. We first extract generic patterns from the existing questions and relations, and then apply them to unseen passages to generate new questions. Since we rely on generic patterns, a significant number of questions might not have answers in the passage in comparison to QG1. For example, let us consider the relation (*Entity Theme Trigger*), for which one of question patterns is "What happens because of *Entity Trigger*?". A sample question from the passage in Figure 2 is "What happens because of PDFG release?" which does not have a valid answer in the passage. The ambiguity of the *Theme* relation gives rise to question patterns that might or might not have a correct answer in the passage. Alternatively, let us consider the relation (*Trigger_1 Enables Trigger_2*). One possible question can be "*What is enabled by Trigger_1*?" which would have *Trigger_2* as the correct answer. We then apply this pattern to other passages when triggers have the same kind of relations. Patterns from the *Enables* relation tend to have an answer when they are applied to new passages. Table 2 shows a sample of possible relations from the passage shown in Figure 2.

Relation	Entity	Location	Theme	Event Trigger	Event Trigger	Coreference
Enable	–	–	–	injury	release	–
Agent	the cells	–	–	pass	–	–
–	divide	–	–	–	–	division
Theme	–	–	–	binding	PDFG molecules	–

Table 2: A sample of possible relations for the passage in Figure 2.

Some sample questions that can be generated from this passage are shown below.

1. What event enables *Event_Trigger* (division)?
2. What would happen without *Event_Trigger* (the binding) of *Event_Trigger* (PDFG molecules)?
3. What happens after *Event_Trigger* (passage) of a *Entity* (the cells)?

Table 3 shows some of the relations and question templates extracted from the ProcessBank corpus.

Question pattern	Question template
Entity *Result* Trigger	What event should occur before the *Trigger* of *Entity*?
Trigger_1 *Super* Trigger_2	What would happen without *Trigger_1* in *Trigger_2*?
Entity *Theme* Trigger	What would happen without the *Trigger* of *Entity*?
Entity *Location* Trigger	Where was *Trigger* in *Entity*?
Trigger_1 *Cause* Trigger_2	What is caused by *Trigger_1*?

Table 3: Entity-trigger relations as question patterns and their associated question templates.

3.4 Distractor Generation

Our distractor generation method can be described as a sequence of the following steps.

1. **Generation of an event graph using event triggers and event-event relations**. The method first forms an event graph as a basis to find possible distractors. Event triggers act as nodes of the graph whereas event-event relations act as edges of the graph. Coreferent nodes in this initial graph are collapsed together to construct the final event graph.
2. **Mapping a question and its correct answer onto the event graph**. In this step, we identify the nodes (event triggers) corresponding to those present in the question and the correct answer. The identified nodes are not used for the generation of distractors. This ensures the reliability of generated distractors, and serves as our *Reliability Checking* step.
3. **Selection of event triggers for distractor generation**. Nodes (event triggers) in the event graph, which are not the ones identified in the previous step, serve as potential candidates for generating distractor items. While informed selection strategies such as the use of causal relations and information regarding prior and subsequent events could have been used to select distractors, they are effective only if the correct answer is known with complete certainty. To combat the stochasticity around the correctness of the answer generated by our question generation methods, we employ random selection from among the potential candidates. We randomly select three nodes as distractors from the list of nodes identified as potential candidates (after removing the nodes in the question and the correct answer). This serves as our *Candidate Selection* step.
4. **Construction of distractor items**. A distractor item is constructed by selecting the phrase comprised of a selected node (event trigger) and its surrounding entities, provided no comma or period is encountered. We resolve entity and event coreferences associated with terms in this step ito produce the final distractor items.

Let us consider as an example the passage shown in Figure 2. Step 1 generates the event graph obtained from this passage as shown in Figure 5. The nodes "divide" and "division" are coreferent, and therefore have been collapsed into a single node "divide/division" along with their edges. Now consider

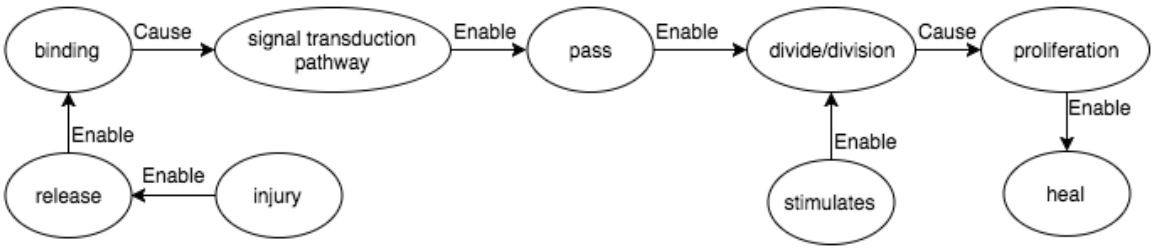


Figure 5: An event graph generated from the passage in Figure 2.

the question and answer we saw associated with this passage in Section 3.2:

- Question: "What is a result of the fibroblast division not only in the artificial conditions of cell culture, but also in an animal's body?"
- Answer: "Proliferation of fibroblasts"

Step 2 recognizes that the nodes "divide/division" and "proliferation" in the event graph are the triggers that the question and answer refer to. We can now use any of the other event triggers from the event graph to generate our distractor items. Since we collapse coreferent nodes, we can guarantee that the nodes selected will not refer to the same triggers as the ones in the question or the answer (*Reliability Checking*). From among the remaining nodes, step 3 randomly selects "binding", "stimulates" and "release" for distractor generation. Lastly, step 4 generates phrases with these triggers and their surrounding entities after entity and event coreference resolution, and constructs the distractor items shown below.

- Distractor 1: binding of PDGF molecules to receptor tyrosine kinases
- Distractor 2: PDGF stimulates fibroblast
- Distractor 3: platelets release PDGF

4 Experiments and Results

To assess the performance of our system, two human annotators evaluate our question generation component and distractor generation component. For a meaningful comparison on question generation, we

use the question generation system by Heilman and Smith (2010) as a baseline. Let MH refer to the baseline. In our experiments, we generate 200 questions from each system, and generate 3 distractors per question.

4.1 Evaluation Criteria for Generated Questions and Distractors

It is important to evaluate generated questions, but this is not straightforward mainly due to the wide variety of acceptable natural language expressions. We use three metrics for the evaluation.

Grammatical correctness judges whether a question is syntactically well-formed. It does not evaluate whether a question is semantically coherent, ignoring the meaning of the question. Our three point scale for this metric is based on the number of grammatical errors.

- 1 (best): The question has no grammatical errors.
- 2: The question has 1 or 2 grammatical errors.
- 3 (worst): The question has 3 or more grammatical errors.

For consistency in counting grammatical errors, we define common grammatical errors in English: spelling errors, run-on sentences, lack of subject-verb agreement, lack of pronoun-antecedent agreement, misplaced modifiers, missing or erroneous quantifiers, prepositions or determiners, erroneous verb forms or nominalization, incorrect word choice, and other errors.

Answer existence identifies whether the answer to a question can be inferred from the passage associated with the question. Note that the answer must be inferred using the passage information only, without relying on external knowledge beyond the passage. Even if a system generates a question while making a specific target its answer, it could be impossible that the target is the answer due to the lack of a valid inference path from the question to the target as its answer. This metric is intended to penalize such questions. Our two-point scale for this metric is:

- 1 (yes): The answer to the question can be inferred from the passage.
- 2 (no): The answer to the question cannot be inferred from the passage.

In addition to answer existence, we also evaluate the correctness of system-generated answers. For this, we use the following three-point scale ratings: correct (1), partially correct (2), and incorrect (3).

Inference steps concern how many semantic relations humans need to understand in order to answer a question. This metric directly evaluates our central idea: inference steps for answering a question. We define the following set of semantic relation types to be considered as inference:

- Event coreference within input text and event coreference between input text and a question.
- Entity coreference within input text and entity coreference between input text and a question.
- Paraphrases in input text and a question.
- Negation, which is a binary relation about logical truthness.

Distractor quality is a rating to measure how appropriate a distractor is for a given question and its correct answer. We set up a three-point scale for rating generated distractors as follows:

- 1 (worst): A distractor is confusing because it overlaps the correct answer partially or completely.
- 2: A distractor can be easily identified as an incorrect answer.
- 3 (best): A distractor can be viable.

As for rating 2, we look for a particular reason for being easily eliminated, such as the distractor is not present in given text.

4.2 Results of Question Generation

We show our results of question generation in Table 4. QG1 achieved more inference steps compared to QG2 by 0.49 and compared to MH by 0.60, while it gained comparable ratings of grammatical correctness and answer existence. We computed the inter-annotator agreement with Cohen's Kappa for each of the criteria mentioned in Section 4.1. Overall, we have a kappa value of 0.55, 0.58 and 0.49 for grammatical correctness, answer existence and inference steps respectively. This result implies moderate agreement. Table 5(a) shows our results for answer correctness. We observe that QG1 tends to generate questions with more incorrect answers than MH.

System	Grammatical correctness			Answer existence			Inference steps		
	Ann 1	Ann 2	Total	Ann 1	Ann 2	Total	Ann 1	Ann 2	Total
QG1	1.52	1.48	1.50	1.17	1.26	1.21	0.80	0.71	**0.76**
QG2	2.13	2.07	2.10	1.58	1.75	1.67	0.31	0.20	0.27
MH	1.42	1.25	**1.34**	1.20	1.14	**1.17**	0.13	0.19	0.16

Table 4: The performance comparison in question generation. Numbers in grammatical correctness and answer existence are average ratings, and lower is better. Numbers in inference steps are average inference steps, and higher is better.

System	Ann 1	Ann 2	Total
QG1	1.35	1.57	1.46
MH	1.08	1.13	1.11

(a) Average ratings of answer correctness in 200 questions. Lower numbers are better. Scores range from 1-3, with 1 a correct answer.

System	Ann 1	Ann 2	Total
QG1	1.98	1.90	1.94
MH	1.93	1.88	1.91

(b) Average ratings of distractors on the same set of 100 questions that the both human evaluators rate as answer existence 1 ("an answer exists"). Higher numbers are better. Scores range from 1-3, with 3 a viable distractor.

Table 5: Results of answer correctness (Table 5(a)) and distractor generation (Table 5(b)).

4.3 Results of Distractor Generation

To make the evaluation of distractor generation reasonable, we need the same set of questions as input for QG1 and MH. Both of the two evaluators rate answer existence 1 on 129 questions generated from QG1 and on 127 questions generated from MH. We randomly select 100 questions from each of the question sets. We show our results of question generation in Table 5(b). Our method achieves an average distractor score of around 2 with a majority of the distractors being rated 2.

5 Discussion

As described in Section 3.2, QG1 attempts to generate questions involving at least one inference step. The average inference step of 0.76 in Table 4 means that the algorithm fails to generate intended questions approximately once out of every four times. A common source of these errors is that some other events can be associated with the event in the question by a different relation (e.g., 'Super'), and they can be an answer to the question. The most common kind of grammatical errors in questions generated by the QG2 system were nominalization ("grow" vs "growth") and verb form ("is uses" instead of "is used"). In terms of inference steps, 'entity coreference' occurred most often. In contrast, QG1 commonly made "determiner" errors in the case of question patterns involving a nominal trigger. For instance, "natural selection" is a noun phrase which does not need an article, but QG1 mistakenly adds "the" in front of it.

The distractor generation component needs to generate distractors as close to a rating of 3 as possible. However, distractors labeled as 2 ("easily eliminated") often occur because they come from events preceding the event described in the question or from events following the results of the events described in the question. A better understanding of cause-and-effect or temporal relations in event graphs might reduce the selection of these easily identified distractors.

We also faced some issues in evaluating system output. The number of grammatical errors assigned can depend on annotators' opinions on how the errors should be corrected. Different corrections could be reached by a different number of steps, changing the evaluation. Thus, differences in opinion on the correct form of the question impacted inter-annotator agreement. Another challenge concerned evaluating the inference steps needed to answer the generated questions. This evaluation requires annotators to identify semantic relations (i.e., event and entity coreferences, negations, and paraphrases) in the texts and count the number of steps needed to answer the question. In some cases, it was not clear how many steps to count, when there were several event mentions in a coreference chain.

6 Conclusion and Future Work

We have presented two different methods that automatically generate questions from multiple sentences. The first method requires learners to take specific inference steps such as event and entity coreferences over multiple sentences. The second method generates questions using patterns extracted from the relations between entities and events in a corpus. Our experiments showed that questions generated by both methods require more inference steps than questions generated by a traditional single-sentence approach. In particular, the first method outperforms the baseline system in terms of the number of inference steps by 0.60 on average, while ensuring comparable grammatical correctness and answer existence. Grounded by past studies in education, we believe that our system-generated questions are more sophisticated and educationally valuable for testing reading comprehension because they require more semantic understanding of text. Our distractor generation approach uses event triggers and event-event relations to generate distractors by building an event graph that lends itself to superior Reliability Checking. We observed that the quality of the generated distractors remains robust across two different methods of question generation.

There are a number of avenues for future work. Although our question generation strategy is domain-independent as described in Section 3.2, some question patterns such as the ones focusing on the order of events (biological processes) might not be useful in other domains. One could explore more domain-adaptable question generation strategies. In another direction, one can extend our system to achieve an end-to-end system which generates multiple-choice questions directly from input text by leveraging automated high-performance semantic parsers, instead of relying on human annotations of a particular corpus. As for distractor generation, one can look at more intelligent ways of generating distractor items from the event graph using causal relations and the knowledge of prior and subsequent events. From a perspective of evaluation, a real user test with non-native English readers is also important. The real-test evaluation will allow us to know how many of the system-generated questions are usable in the real test, obtain the insights of what questions are truly important for language learners' reading comprehension, and capture the rationale behind distractors (e.g., the most confusing distractor patterns) based on an analysis of readers' performance. One could also explore methods to automatically carry out one or more of our evaluation processes. As we argued in this paper, our question generation strategy lends itself to language learners' reading comprehension. In addition, our research can also be useful to the task of creating exam questions and answers, since manual creation is normally quite time-consuming. Automatically generated questions and multiple-choice answers make the creation of exam QAs more efficient.

Acknowledgements

This publication was partly made possible by grant NPRP-08-1337-1-243 from the Qatar National Research Fund (a member of the Qatar Foundation). The statements made herein are solely the responsibility of the authors. Jun Araki is partly supported by an IBM Ph.D. Fellowship and a Funai Overseas Scholarship.

References

Manish Agarwal, Rakshit Shah, and Prashanth Mannem. 2011. Automatic question generation using discourse cues. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–9.

Richard C. Anderson and W. Barry Biddle. 1975. On asking people questions about what they are reading. *Psychology of Learning and Motivation*, 9:90–132.

Thomas Andre. 1979. Does answering higher level questions while reading facilitate productive learning? *Review of Educational Research*, 49(2):280–318.

Mahmoud Azab, Ahmed Salama, Kemal Oflazer, Hideki Shima, Jun Araki, and Teruko Mitamura. 2013. An English reading tool as a NLP showcase. In *Proceedings of IJCNLP 2013: System Demonstrations*, pages 5–8.

Lee Becker, Sumit Basu, and Lucy Vanderwende. 2012. Mind the gap: Learning to choose gaps for question generation. In *Proceedings of NAACL-HLT 2012*, pages 742–751.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of EMNLP 2014*, pages 1499–1510.

Neil Campbell and Jane Reece. 2005. *Biology*. Benjamin Cummings.

Wei Chen, Gregory Aist, and Jack Mostow. 2009. Generating questions automatically from informational text. In *Proceedings of the 2nd Question Generation Workshop*.

Sérgio Curto, Ana Cristina Mendes, and Luísa Coheur. 2011. Exploring linguistically-rich patterns for question generation. In *Proceedings of the UCNLG+Eval: Language Generation and Evaluation Workshop*, pages 33–38.

Arthur C. Graesser, Kurt VanLehn, Carolyn P. Ros, Pamela W. Jordan, and Derek Harter. 2001. Intelligent tutoring systems with conversational dialogue. *AI Magazine*, 22(4):39–51.

Christiaan Hamaker. 1986. The effect of adjunct questions on prose learning. *Review of Educational Research*, 56(2):212–242.

Michael Heilman and Noah A. Smith. 2010. Good question! Statistical ranking for question generation. In *Proceedings of NAACL-HLT 2010*, pages 609–617.

Michael Heilman, Kevyn Collins-thompson, Jamie Callan, and Maxine Eskenazi. 2006. Classroom success of an intelligent tutoring system for lexical practice and reading comprehension. In *Proceedings of the 9th International Conference on Spoken Language Processing*.

Ayako Hoshino and Hiroshi Nakagawa. 2007. Assisting cloze test making with a web application. In *Proceedings of Society for Information Technology and Teacher Education International Conference*.

John Judge, Aoife Cahill, and Josef van Genabith. 2006. QuestionBank: Creating a corpus of parse-annotated questions. In *Proceedings of ACL/COLING 2006*, pages 497–504.

Kenneth R Koedinger, John R Anderson, William H Hadley, and Mary A Mark. 1997. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8:30–43.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *Proceedings of ACL/IJCNLP 2015*, pages 889–898.

John Lee and Stephanie Seneff. 2007. Automatic generation of cloze items for prepositions. In *Proceedings of INTERSPEECH 2007*, pages 2173–2176.

David Lindberg, Fred Popowich, John Nesbit, and Phil Winne. 2013. Generating natural language questions to support learning on-line. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 105–114.

Prashanth Mannem, Rashmi Prasad, and Aravind Joshi. 2010. Question generation from paragraphs at UPenn: QGSTEC system description. In *Proceedings of the 3rd QG Workshop*.

Karen Mazidi and Rodney D. Nielsen. 2014. Linguistic considerations in automatic question generation. In *Proceedings of ACL 2014*, pages 321–326.

Ruslan Mitkov and Le An Ha. 2003. Computer-aided generation of multiple-choice tests. In *Proceedings of the NAACL-HLT 2003 Workshop on Building Educational Applications Using Natural Language Processing*, pages 17–22.

Ruslan Mitkov, Le An Ha, Andrea Varga, and Luz Rello. 2009. Semantic similarity of distractors in multiple-choice tests: extrinsic evaluation. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 49–56.

Jack Mostow and Hyeju Jang. 2012. Generating diagnostic multiple choice comprehension cloze questions. In *Proceedings of the 7th Workshop on Building Educational Applications Using NLP*, pages 136–146.

Juan Pino and Maxine Eskenazi. 2009. Semi-automatic generation of cloze question distractors effect of students' L1. In *Proceedings of SLaTE 2009*, pages 65–68.

Paul Piwek and Svetlana Stoyanchev. 2010. Question generation in the CODA project. In *Proceedings of the 3rd Workshop on Question Generation*, pages 29–34.

Keisuke Sakaguchi, Yuki Arase, and Mamoru Komachi. 2013. Discriminative approach to fill-in-the-blank quiz generation for language learners. In *Proceedings of ACL 2013*, pages 238–242.

Simon Smith, PVS Avinesh, and Adam Kilgarriff. 2010. Gap-fill tests for language learners: Corpus-driven item generation. In *Proceedings of the International Conference on Natural Language Processing 2010*, pages 1–6.

Eiichiro Sumita, Fumiaki Sugaya, and Seiichi Yamamoto. 2005. Measuring non-native speakers' proficiency of English by using a test with automatically-generated fill-in-the-blank questions. In *Proceedings of the 2nd workshop on Building Educational Applications Using NLP*, pages 61–68.

Kurt Vanlehn, Collin Lynch, Kay Schulze, Joel A Shapiro, Robert Shelby, Linwood Taylor, Don Treacy, Anders Weinstein, and Mary Wintersgill. 2005. The Andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education*, 15(3):147–204.

John H. Wolfe. 1976. Automatic question generation from text - an aid to independent study. In *Proceedings of the ACM SIGCSE-SIGCUE Technical Symposium on Computer Science and Education*, pages 104–112.

Xuchen Yao, Gosse Bouma, and Yi Zhang. 2012. Semantics-based question generation and implementation. *Dialogue and Discourse, Special Issue on Question Generation*, 3(2):11–42.

Torsten Zesch and Oren Melamud. 2014. Automatic generation of challenging distractors using context-sensitive inference rules. In *Proceedings of the 9th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 143–148.

Evaluation Strategies for Computational Construction Grammars

Tânia Marques
School of Informatics
University of Edinburgh
Edinburgh EH8 9AB United Kingdom
tmarques@inf.ed.ac.uk

Katrien Beuls
Artificial Intelligence Lab
Vrije Universiteit Brussel
Pleinlaan 2 B-1050 Brussels Belgium
katrien@ai.vub.ac.be

Abstract

Despite the growing number of Computational Construction Grammar implementations, the field is still lacking evaluation methods to compare grammar fragments across different platforms. Moreover, the hand-crafted nature of most grammars requires profiling tools to understand the complex interactions between constructions of different types. This paper presents a number of evaluation measures, partially based on existing measures in the field of semantic parsing, that are especially relevant for reversible grammar formalisms. The measures are tested on a grammar fragment for European Portuguese clitic placement that is currently under development.

1 Introduction

Computational Construction Grammar allows computational linguists to formalize their hypotheses and intuitions about certain linguistic phenomena and explore how these representational choices affect the processing of natural language utterances (Schneider and Tsarfaty, 2013). In this sense, it follows in the footsteps of Generalized Phrase Structure Grammar (GPSG) (Gazdar et al., 1985), Lexical Functional Grammar (Bresnan et al., 2016), Head-Driven Phrase-Structure Grammar (HPSG) (Pollard and Sag, 1994) and Combinatory Categorial Grammar (CCG) (Steedman, 2000). Yet, different from these other approaches, it adheres to the principles of Construction Grammar (CxG) (Goldberg, 2005; Östman and Fried, 2005; Hoffmann and Trousdale, 2013).Therefore, constructions are treated as first-class citizens in the grammatical organisation of a language. They are viewed as learned mappings between form (sounds, morphemes, syntactic categories) and function (semantics, pragmatics, etc.). Because there is no strict separation between the lexicon and the grammar, semi-productive idioms like "X let alone Y" are treated in the same way with lexemes and core syntactic patterns (Fillmore et al., 1988). We can distinguish three main computational frameworks that are currently active within the Construction Grammar community: Embodied Construction Grammar – ECG (Bergen and Chang, 2005), Fluid Construction Grammar – FCG (Steels, 2004; Steels, 2011) and more recently Template Construction Grammar – TCG (Barrès and Lee, 2014).

The evaluation of computational construction grammars is currently not reaching further than proof-of-concept grammar fragments that show how to implement a certain language phenomenon and demonstrate the resulting grammar by means of web demonstrations or its use in a simulation-based robotic environment (Trott et al., 2015). There are two reasons for the lack of widely used evaluation metrics in CxG: (i) Different from data-driven approaches, construction grammars are not built automatically from annotated treebanks and therefore do not reach a wide coverage in the traditional sense. Instead, both ECG and FCG allow the grammar writer to test a number of sentences automatically when loading the grammar fragment and return the average base parse for these (geometric mean of the number of parses per sentence). (ii) Different from syntactic parsers that concentrate on the syntactic accuracy of the syntax trees that their grammars derive, computational construction grammars focus on semantic accuracy as a metric that better meets their objectives. However, semantic accuracy is harder to measure than syntactic accuracy because it requires textual corpora annotated with large formal meaning representations that are agreed upon by different grammar developers.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1137–1146, Osaka, Japan, December 11-17 2016.

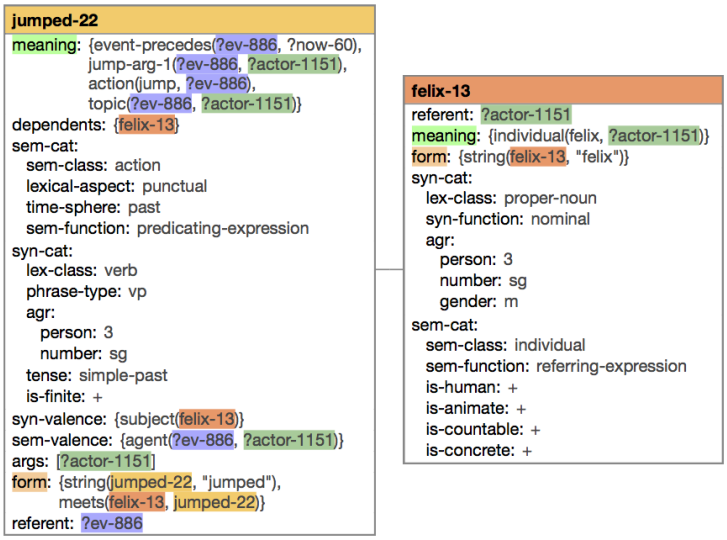


Figure 1: Resulting transient structure after parsing "Felix jumped". Features cut across argument structure, information structure and functional structure.

The field of computational CxG is reaching a certain level of maturity in terms of linguistic phenomena that are treated and publications that accompany grammars (e.g. (van Trijp, 2015) for FCG and (Trott et al., 2015) for ECG), and has clearly demonstrated the feasibility of implementing the constructional approach in a full-fledged computational framework (Schneider and Tsarfaty, 2013). The time has come to evaluate these implementations so that they can be compared within the field and benchmarks can be created. We therefore suggest a number of metrics for parsing (comprehending an utterance into a meaning representation) and production (formulating an utterance from a meaning representation) of a test suite that addresses a specific linguistic phenomenon. Because computational CxG fragments are hand-crafted precision grammars rather than data-driven approaches, broad-coverage newspaper corpora are not interesting test beds for evaluation since they are not constrained towards the phenomena of study. Rather, we suggest to compile a test suite of sentences taken from linguistic research in the area that the grammar is focusing on.

Before explaining the actual metrics we designed, our contribution first argues, in Section 2, for the usefulness of including CxG in computational linguistics. Section 3 then draws parallels between the fields of semantic parsing and computational construction grammar in terms of semantic representations, before Section 4 presents metrics we propose to evaluate grammars in the latter formalism. The case study then shows the use of these metrics in a sample grammar for European Portuguese in Section 5. Finally, Section 6 concludes.

2 Why Constructions?

Computational CxG views production and comprehension in terms of a chain of consecutive operations over a linguistic structure, called the transient structure, a feature structure consisting of units that are made up of non-typed feature-value pairs, which maintains a temporary state of knowledge. A sequence of transient structures on a particular execution chain is called a linguistic pathway (Steels, to appear). One of the characteristics of CxG is its "insistence on simultaneously describing grammatical patterns and the semantic and pragmatic purposes to which they are dedicated" (Fillmore et al., 1988) so transient structures, as well as constructions, need to be able to represent information from a multitude of different perspectives. A construction schema, or in short, a construction, is an abstract schema that can be used to "expand any aspect of a transient structure from any perspective and it can consult any aspect of this transient structure to decide how to do so" (Steels, to appear).

The result of construction application is thus a transient structure from which semantic or formal information can directly be extracted, without additional steps. Figure 1 shows the resulting transient

structure after parsing the intransitive sentence "Felix jumped" with a dependency-style grammar. The meaning features of both units form together the complete understanding of the sentence. Through variable equalities, Felix is linked to the first argument of the jump action and the topic of the sentence. Of course, the meaning predicates in this example are merely illustrative here and are subject to choices that the grammar engineer makes.

Not only the resulting transient structure can be valuable in the evaluation procedure but also the constructions that built the structure should be considered. In the case of "Felix jumped", a two-word utterance, at least five constructions have been at work to construe its interpretation. Two lexical constructions covering the words themselves, one tense construction to situate the event in the past (and indicate that it is not the adjective "jumped"), one argument structure construction linking the Felix to the actor or the jumping event (intransitive) and one information structure construction identifying "Felix" as the topic of the sentence.

3 Semantic Parsing

Semantic parsers – similar to Computational Construction Grammar implementations – are not interested in building well-formed syntactic trees but instead map sentences into formal meaning representations (Mooney, 2007). They are used in domains such as question answering, where natural-language questions are converted into formal queries, and are typically built over databases with unsupervised (e.g. (Poon and Domingos, 2009)) or supervised (e.g. (Berant et al., 2013)) learning algorithms. To go beyond simple query formulation towards complex knowledge extraction, a recent approach by (Parikh et al., 2015) uses distant supervision methods to learn a semantic parser from a database of complex events and unannotated texts.

While testing, the retrieved meanings are compared to a gold standard annotation to calculate the precision and recall of the parser. Three main ways to calculate precision and recall can be distinguished, ranging from less to more fine-grained analyses:

1. The retrieved meaning/formulated query is correct when it matches the gold standard. Recall is then the number of correct meanings divided by the number of sentences. Precision is the number of correct meanings divided by the number of sentences for which the parser produced a meaning. An example of a system that employs this measure is the Cocktail system (Tang and Mooney, 2001).

2. Instead of using a binary measure, one can calculate the overlap in attribute-value pairs between the retrieved meaning and the gold standard. Recall is then the percentage of recovered attribute-value pairs per sentence, averaged over the test set. An example of a semantic parser that calculates this overlap is the (Zettlemoyer and Collins, 2007) parser for the ATIS flight info domain.

3. The Smatch score (Cai and Knight, 2013) measures the overlap between meaning representations while taking into account variable bindings. It is determined by calculating the maximum possible F-scores for alternative bindings. This calculation process can be seen in Table 1 for a small exemplifying sentence "a boy is", with the following gold standard and parsed meanings (in Abstract Meaning Representation):

   gold standard meaning: *instance(?x, boy)* $\wedge$ *attribute(?x, single)* $\wedge$ *instance(?y, be)* $\wedge$ *attribute(?y, currently-being)* $\wedge$ *is(?y, ?x)*

   parsed meaning: *instance(?a, boy)* $\wedge$ *attribute(?b, single)* $\wedge$ *instance(?b, be)* $\wedge$ *is(?a, ?a)*

	Matched	Precision	Recall	F-Score
x = a y = b	2	2/4	2/5	0.44
x = b y = a	1	1/4	1/5	0.22
			S-score:	0.44

Table 1: Calculation of the normal Smatch score.

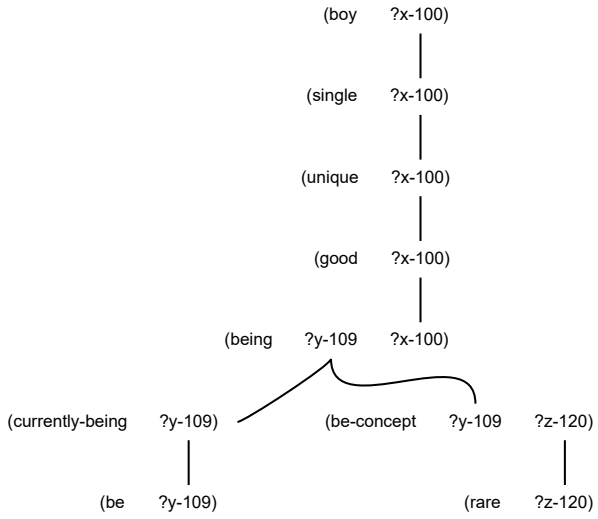


Figure 2: Fully connected meaning after parsing the sentence "A good boy is rare".

Given the existing variables, there are two possible bindings. The *?x* is either bound to *?a* or *?b*, and the same is true for *?y*. There are 5 relations in the gold standard, but only 4 in the generated meaning. Replacing the variables with *?x = ?a; ?y = ?b* gives two matches between the meanings: *instance(?x, boy) = instance(?a, boy)* and *instance(?y, be) = instance(?b, be)*. This makes the precision 2/4 and the recall 2/5, leading to an F-score of 0.44, which is the maximum amongst all the possible bindings. Therefore, 0.44 is also the Smatch score (S-score) for the two meanings.

The S-score is interesting for evaluating construction grammars, because the meaning representation obtained in Fluid Construction Grammar can be directly converted to Abstract Meaning Representation (AMR). AMR has two types of relations: a relation between a concept and a variable (instance and attribute relations); and a relation between two variables (argument relations). The AMR version of the meaning network in Figure 2 would become: *instance(?x, boy)* $\wedge$ *attribute(?x, single)* $\wedge$ *attribute(?x, unique)* $\wedge$ *attribute(?x, good)* $\wedge$ *instance(?y, be)* $\wedge$ *attribute(?y, currently-being)* $\wedge$ *arg(?y, ?x)* $\wedge$ *arg (?y, ?z)* $\wedge$ *instance(?z, rare)*. All the concepts related to the "boy" use the same variable *?x*, and the concept "rare" is connected with the verb "be", which is happening at present and corresponds to the being *?x*, "boy" which is *?z*, "rare". The nodes in the graph are second order logics predications, where properties and relations can be objects as well. For instance, the meaning of "a good boy is rare" could be: *(boy ?x)* $\wedge$ *(single ?x)* $\wedge$ *(unique ?x)* $\wedge$ *(good ?x)* $\wedge$ *(be ?y)* $\wedge$ *(currently-being ?y)* $\wedge$ *(being ?y ?x)* $\wedge$ *(be-concept ?y ?z)* $\wedge$ *(rare ?z)*. All the concepts related to the "boy" use the same variable *?x*, and the concept "rare" is connected with the verb "be", which is happening at present and corresponds to the being *?x*, "boy" which is *?z*, "rare".

4 Proposed Metrics

The remainder of this section explains the two accuracy metrics in more detail. For comprehension, a variant of the Smatch score is proposed that takes into account subparts of the meaning graph. For production, we present the longest common substring measure, with two variants. Finally, a single profiling measure is included to quantify the efficiency of the grammar in comprehension and production.

Reinterpreting Smatch Let us consider that we were evaluating a hypothetical precision grammar with the goal of identifying nouns and their modifiers, but that lacked any constructions for verbs. Then, the S-score of 0.44 (obtained in Table 1), while important for understanding the broad accuracy of the grammar, might not really tell us how the grammar is doing in terms of identifying the nouns. For expressing this in an explicit way, we might also want to have a specific smatch score to calculate the accuracy only for this phenomena. Unfortunately, due to the relational nature of the different parts of the

sentence, it is not possible to separate what is being studied from what is not, since the identification of its relations is also important.

A compromise between what is studied and what we can measure was found for enabling us to calculate a more specific S-score, while keeping the same process of finding the maximum F-score. This is done by annotating the variables in the gold standard that are more important for the phenomena being studied. In this case, it would be *x*. Then, we disregard any relation that does not contain this variable in it, and we accept partial matchings. Even, if *is(?y, ?x)* did not totally match, but the noun was identified in the correct position, it would still be counted. Table 2 shows the calculation for the same sentence ("a boy is"), but considering only the variable *x*. There are three relations to be considered in the gold standard and two in the meaning obtained, regardless of the variable bindings. The matches are still two, but the precision and recall increase, leading to a S-score of 0.8

	Matched	Precision	Recall	F-Score
x = a y = b	2	2/2	2/3	0.8
x = b y = a	1	1/2	1/3	0.4
			S-score:	0.8

Table 2: Calculation of the specific Smatch score.

The S-score obtained for a specific phenomenon should not be presented on its own, because it would lead to an erroneous understanding of the grammar accuracy over the whole corpus. However, together they give a better understanding how the grammar is working. For instance, in this example, the grammar only parses this sentence into its meaning with a 0.44 accuracy, yet the noun identification is mostly correct, which is what actually tell us if the grammar is working for what it was proposed.

Calculating reproducibility The FCG grammar has an additional problem that is usually not faced in semantic parsers: the fact that the full meaning of a sentence is correctly obtained does not necessarily mean that the original utterance can be reproduced. FCG does not work with templates or libraries of sentences, instead the sentence will be reproduced based on the syntactic aspects of the grammar. While word order does not pose any problems in parsing and can indeed help to guide the comprehension process, it can become an issue in production that gets worse when sentence length increases. It is also particularly hard in languages that do not follow a rigid word order, where a different but correct sentence can also be produced without any change in meaning.

Measures that evaluate the correctness of the sentence that is produced are essential to have a complete understanding of the accuracy of FCG due to its bidirectional nature. The most obvious measure would be to compare the sentence obtained with a set of possible acceptable sentences that can be generated back with the same meaning. However, this binary measure does not convey much information. Instead, we propose to use the Longest Common Subsequence (LCS) algorithm to obtain the percentage of the sentence that was correctly generated. LCS is an algorithm that given two sequences, $X = [x_1, x_2, .., x_i]$ and $Y = [y_1, y_2, .., y_j]$, can find the maximum length subsequence between them, defined as a strict increasing sequence of indices of X $[1, 2, .., k]$ such that $x_{ij} = z_j$ (Cormen, 2009). The number of words in the maximum length common subsequence divided by the number of words in the original acceptable sentences, will then gives us an estimation of the percentage of the sentence that the grammar was able to generate back. This measure is not new, and has been previously used to evaluate machine translated texts by (Lin and Och, 2004).

Because the comprehension process might lead to partial meanings, we distinguished between two variants of the LCS measure. One for sentences that were successfully parsed, leading to a fully connected network and one for the unsuccessful ones that might only lead to the production of partial sentences.

Grammar efficiency Construction application may generate a large search. Multiple constructions can expand the same transient structure at a certain time step, or a single construction can expand the transient structure in more than one way. Splits in the search tree can be the result of ambiguities in processing, but

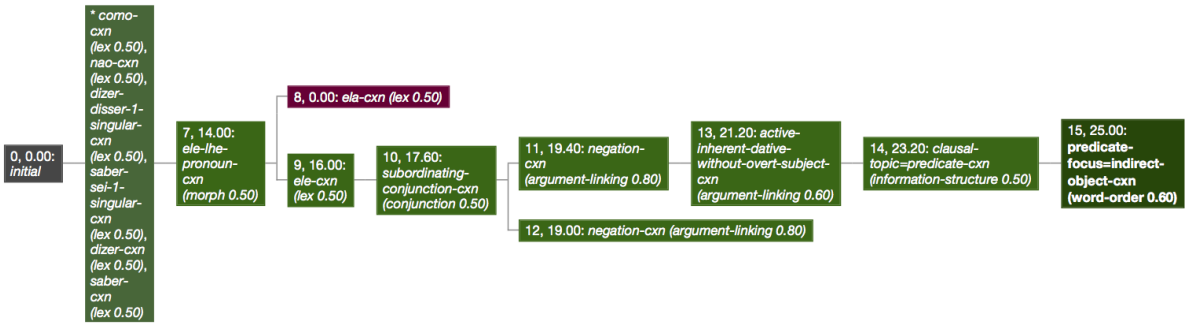


Figure 3: The comprehension process of "não sei como disser-lhe" has an efficiency of 13/14 (not considering the red node, where the second unification step failed).

often originate in sloppy grammar design. Measuring grammar efficiency (at least in terms of additional nodes generated in the search tree) is done by dividing the number of search nodes in the branch that leads to the solution (when all goal tests succeed) by the total number of nodes in the tree. When efficiency equals to 1, no additional nodes are created. The grammar efficiency of the comprehension example in Figure 3 equals 13/15. We see that one of the two additional nodes is red, indicating that it passed the first unification step (conditional part) but failed when unifying the contributing part of the construction. This node do not create additional search since they have no further children. A more informative grammar efficiency measure would perhaps only consider the succeeded nodes. In that case, the example sentence "não sei como disser-lhe" has an efficiency of 13/14 (0.93).

5 Case Study

To better understand how one can interpret the proposed metrics and how they differ from traditional approaches, we include the evaluation of an European Portuguese (EP) grammar fragment that is under development in FCG. It should be noted, however, that this grammar fragment is only here to exemplify how the metrics can be used during grammar developing and evaluation phase. We do not make any claims regarding the grammar itself. The chosen fragment focuses on pronominal clitics. Clitics in EP can be positioned following the verb (enclisis) or preceding the verb (proclisis). Correct clitic placement does not depend on the finiteness of the verb (as in other Romance languages) but is instead determined by the phrasal context. Hence, the coverage of different contexts is an important consideration in the evaluation of such a grammar. Therefore, we built our own test suite by collecting 67 sentences from linguistic research papers dedicated to this phenomenon (Madeira, 1992; Luís et al., 2004; Luís and Otoguro, 2011), and annotated them manually. To create the grammar, lexical constructions were automatically generated from the test suite words based on their grammatical categories, but the core grammatical constructions were hand-crafted considering the linguistic concept being studied, and do not represent necessarily all grammatical intricacies present in the test suite sentences.

The normal S-score gives us an F-score of 0.75 ± 0.21, while the specific S-score returns a slightly higher value of 0.79 ± 0.32. These numbers tell us that the grammar is not covering all necessary aspects to comprehend all the sentences but it is better in the clitics placement than it would seem, as the specific S-score is higher than the general one. To get a better understanding of which parts of the grammar are still not satisfactory, the example sentences were annotated with the proclisis trigger they contain, or else tagged as enclisis. Table 3 shows the average S-scores for seven triggers, together with their standard deviation. Between brackets the number of example sentences is shown. The operator adverb and the relative clauses results exemplify the biggest discrepancies between the two S-scores: 0.62 vs 0.84 and 0.63 vs. 0.80, respectively. This shows that while the grammar fragment is not very good at handling the sentences containing this concepts, the position of the clitics is still correctly identified. Looking into the sentences, we understand that they have more complex verbal forms that are not well processed by the grammar because they were not implemented. The opposite situation does also occur where the general score is higher than the specific. For instance, the enclisis results present a F-score of 0.75 vs 0.73, which

Clitics Position	Triggers	S-score		LCS	
		Normal	Specific	Successful	Unsuccessful
enclisis (28)	none (28)	0.75 ± 0.20	0.73 ± 0.38	0.87 ± 1.43	0.49 ± 0.63
proclisis (39)	negation (6)	0.77 ± 0.25	0.81 ± 0.79	1.00 ± 0.00	N/A
	wh-question (5)	0.88 ± 0.18	0.86 ± 0.28	0.90 ± 0.12	0.33 ± 0.00
	relative clause (7)	0.62 ± 0.08	0.84 ± 0.25	0.43 ± 0.68	0.37 ± 0.10
	fronted focus (8)	0.70 ± 0.22	0.88 ± 0.23	0.84 ± 0.33	0.42 ± 0.01
	operator adverb (3)	0.63 ± 0.23	0.80 ± 0.28	0.40 ± 0.00	0.20 ± 0.00
	undefined-subject (3)	0.89 ± 0.09	0.89 ± 0.16	1.00 ± 0.00	N/A
	downward quantifier (7)	0.78 ± 0.20	0.79 ± 0.30	0.74 ± 0.63	0.14 ± 0.30

Table 3: Accuracy and reproducibility results for the EP grammar case study. All the results are presented by the concepts being studied, the proclisis triggers.

means that although it seems to perform generically well in those sentences, the positioning of the clitics is incorrect more often than predicted by the general score.

Table 3 also includes the Longest Common Substring results. Two scores are kept: the leftmost column includes the LCS for sentences that were produced from meanings that passed all goal tests (and have thus a higher chance of success). The rightmost column shows the scores for productions whose initial meaning networks were extracted from a comprehension process that failed. No score is shown when the case did not happen. The biggest issue in getting the word order right seems to occur in the relative clauses (0.43) and the operator adverbs (0.40). The latter allow multiple grammatically correct word orders in EP, whereas we only include the first solution.

When it comes to grammar efficiency, Figure 4 (on the left) shows the results ordered by sentence length (3–7). We see indeed that the efficiency does not scale well when longer sentences are parsed (and the same goes for production). Less than 20% of all search nodes are used by the branch that leads to the solution. To get an idea of the complexity of the grammar Figure 4 (on the right) plots the number of constructions that is needed to parse a sentence. Sentences of length 3 require on average 9 constructions and the number increases with steps of 2 with every word that is added.

The grammar fragment presented in this section covers a very basic grammatical phenomenon of positioning the clitics correctly. Yet, being basic it has several intrinsic aspects that are fundamental to get right to process it correctly, especially if we want to have grammars that generate grammatically correct sentences. A traditional metric of accuracy or even the normal S-score metric would give an unfair comparison between this fragment grammar and a corpus-based grammar. While, the latter might be better at covering all the sentences provided in a corpus or in a test suite, it might always fail in the processing of this phenomena. While the former grammar not being constructed for a wide coverage

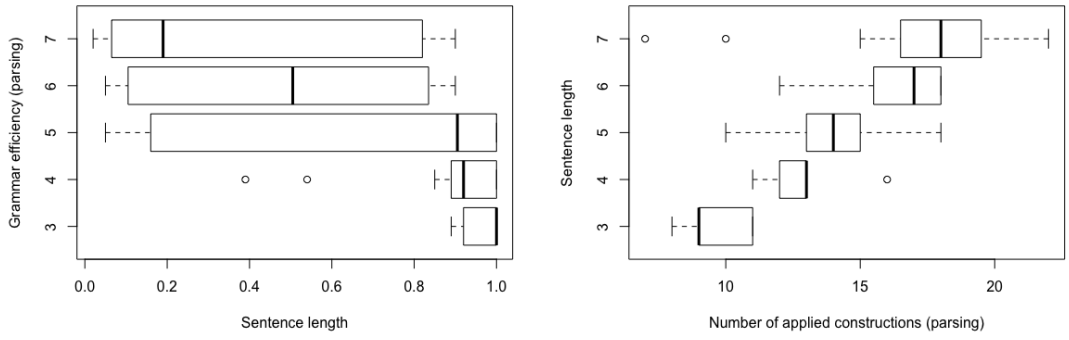


Figure 4: The efficiency and complexity of the EP grammar in function of the sentence length.

might be better at handling this phenomena. The specific s-score when used together provides us with further information to understand how the grammar differs in a more detailed level. Furthermore, this score and the additional profiling metrics for efficiency are very useful for grammar engineers during the grammar developing phase to understand where their grammar falls short and should be further developed. The LCS in FCG provides a further insight into the accuracy of the generated sentences which tells us how well the grammar understands and generates the sentence back.

Using the AMR annotation for evaluating grammar is a relatively costly process. However, it is becoming increasingly necessary to have a semantically annotated corpus in addition to the more traditional syntactical tree-based annotation. Furthermore, there are some strategies that can decrease the burden on the annotators: (1) if only a partial phenomena is being studied, then it might be reasonable to have only partial annotations; (2) it is possible to distribute the phenomena per annotator, thus decreasing the learning curve; (3) it is possible to provide the sentences already generated by the grammars and ask the annotators to fix the errors based on their cognitive understanding of the meaning and/or a set of rules provided.

6 Conclusions

Taking inspiration from existing measures in semantic parsing and machine translation, we proposed two new metrics for evaluating computational Construction Grammar implementations: the S-score, with a variant for focusing on the parts of the meaning graph that are tackled by the grammar fragment; and the LCS score, indicating the reproducibility of the retrieved meaning network. Used in addition to more traditional metrics, these scores give insights about the exact type of phenomena that can be handle by a precision grammar, which is important to distinguish grammars that cover a large number of sentences but invariably fail in processing specific phenomena and grammars that cover a small set of sentences but can deal well with a specific phenomena. Additionally, some profiling measures are also suggested to give an idea of the grammar efficiency and complexity. We hope the proposed metrics help grammar engineers to better understand the complex interactions between the constructions in their grammars and the phenomena being covered by it.

Acknowledgements

The research presented in this paper has been funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 607062 *ESSENCE: Evolution of Shared Semantics in Computational Environments* (`http://www.essence-network.com/`).

References

Victor Barrès and Jinyong Lee. 2014. Template construction grammar: from visual scene description to language comprehension and agrammatism. *Neuroinformatics*, 12(1):181–208.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In Timothy Baldwin and Anna Korhonen, editors, *Conference on Empirical Methods on Natural Language Processing*, pages 1533–1544, Seattle, Washington. The Association for Computational Linguistics.

Benjamin Bergen and Nancy Chang. 2005. Embodied construction grammar in simulation-based language understanding. In Jan-Ola Östman and Mirjam Fried, editors, *Construction grammars: Cognitive grounding and theoretical extensions*, number 3 in Constructional Approaches to Language, pages 147–190. John Benjamins, Amsterdam.

Joan Bresnan, Ash Asudeh, Ida Toivonen, and Stephen Wechsler. 2016. *Lexical-Functional syntax*. Wiley-Blackwell, West-Sussex, England, 2 edition, August.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013*, volume Volume 2: Short papers, pages 748–752, Sofia, Bulgaria, 4-9 August 2013.

Thomas H. Cormen. 2009. *Introduction to algorithms*. MIT Press, Boston, MA.

Charles J Fillmore, Paul Kay, and Mary Catherine O'Connor. 1988. Regularity and idiomaticity in grammatical constructions: The case of let alone. *Language*, 64(3):501–538.

Gerald Gazdar, Ewan Klein, Geoffrey K. Pullum, and Ivan A. Sag. 1985. *Generalized Phrase Structure Grammar*. Basil Blackwell, Oxford.

Adele Goldberg. 2005. *Constructions at work*. Oxford University Press, Oxford.

Thomas Hoffmann and Graeme Trousdale, editors. 2013. *The Oxford handbook of Construction Grammar*. Oxford University Press, Oxford.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 605–612, Barcelona, Spain, July. ACL.

Ana Luís and Ryo Otoguro. 2011. Inflectional morphology and syntax in correspondence: Evidence from european portuguese. In Glyn Hicks Alexandra Galani and George Tsoulas, editors, *Morphology and Its Interfaces*, number 178 in Linguistik Aktuell/Linguistics Today, pages 97–136. John Benjamins, Amsterdam.

Ana Luís, Ryo Otoguro, Miriam Butt, and Tracy Holloway King. 2004. Proclitic contexts in european portuguese and their effect on clitic placement. In Miriam Butt and Tracy Holloway King, editors, *The Proceedings of the LFG'04 Conference*, pages 334–352, Stanford, CA. CSLI Publications.

Ana Maria Madeira. 1992. On clitic placement in European Portuguese. *UCL Working Papers in Linguistics*, 4:95–122.

Raymond J. Mooney. 2007. Learning for semantic parsing. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 4394 of *Lecture Notes in Computer Science*, pages 311–324, Mexico City, Mexico, February 18-24. Springer.

Jan-Ola Östman and Mirjam Fried. 2005. *Construction Grammars: Cognitive grounding and theoretical extensions*, volume 3 of *Constructional Approaches to Language*. John Benjamins, Amsterdam.

Ankur P Parikh, Hoifung Poon, and Kristina Toutanova. 2015. Grounded semantic parsing for complex knowledge extraction. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 756–766, Denver, Colorado, June 5. Association for Computational Linguistics.

Carl Pollard and Ivan A Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press, Chicago.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1*, pages 1–10, Singapore, August 6-7. Association for Computational Linguistics.

Nathan Schneider and Reut Tsarfaty. 2013. Book review: Design patterns in Fluid Construction Grammar. *Computational Linguistics*, 39(2):447–453.

Mark Steedman. 2000. *The syntactic process*. MIT Press, Boston, MA.

Luc Steels. 2004. Constructivist development of grounded construction grammars. In Walter Daelemans, editor, *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 9–19, Barcelona. Association for Computational Linguistics.

Luc Steels, editor. 2011. *Design patterns in Fluid Construction Grammar*. Number 11 in Constructional Approaches to Language. John Benjamins, Amsterdam.

Luc Steels. to appear. The basics of Fluid Construction Grammar. *Constructions and Frames*.

Lappoon R Tang and Raymond J Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning*, pages 466–477, Freiburg, Germany. Springer.

Sean Trott, Aurélien Appriou, Jerome Feldman, and Adam Janin. 2015. Natural language understanding and communication for multi-agent systems. In *Artificial Intelligence for Human-Robot Interaction Papers from the AAAI 2015 Fall Symposium*. AAAI.

Remi van Trijp. 2015. Cognitive vs. generative construction grammar: The case of coercion and argument structure. *Cognitive Linguistics*, 26(4):613–632.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 678–687, Prague, June.

Building a Monolingual Parallel Corpus for Text Simplification Using Sentence Similarity Based on Alignment between Word Embeddings

Tomoyuki Kajiwara
Graduate School of System Design
Tokyo Metropolitan University
Tokyo, Japan
kajiwara-tomoyuki@ed.tmu.ac.jp

Mamoru Komachi
Graduate School of System Design
Tokyo Metropolitan University
Tokyo, Japan
komachi@tmu.ac.jp

Abstract

Methods for text simplification using the framework of statistical machine translation have been extensively studied in recent years. However, building the monolingual parallel corpus necessary for training the model requires costly human annotation. Monolingual parallel corpora for text simplification have therefore been built only for a limited number of languages, such as English and Portuguese. To obviate the need for human annotation, we propose an unsupervised method that automatically builds the monolingual parallel corpus for text simplification using sentence similarity based on word embeddings. For any sentence pair comprising a complex sentence and its simple counterpart, we employ a many-to-one method of aligning each word in the complex sentence with the most similar word in the simple sentence and compute sentence similarity by averaging these word similarities. The experimental results demonstrate the excellent performance of the proposed method in a monolingual parallel corpus construction task for English text simplification. The results also demonstrated the superior accuracy in text simplification that use the framework of statistical machine translation trained using the corpus built by the proposed method to that using the existing corpora.

1 Introduction

Text simplification is the process of rewriting a complex text into a simpler form while preserving its meaning. The purpose of text simplification is to assist the comprehension of readers, especially language learners and children. Recent studies have treated text simplification as a monolingual machine translation problem in which a simple synonymous sentence is generated using the framework of statistical machine translation (Specia, 2010; Zhu et al., 2010; Coster and Kauchak, 2011a; Coster and Kauchak, 2011b; Wubben et al., 2012; Štajner et al., 2015a; Štajner et al., 2015b; Goto et al., 2015). However, unlike statistical machine translation, which uses bilingual parallel corpora, text simplification requires a monolingual parallel corpus for training. While bilingual parallel data are available in large quantities, monolingual parallel data are hard to obtain because simplification of a complex text is not a by-product of other tasks. Monolingual parallel corpora for text simplification are available in only seven languages—English (Zhu et al., 2010; Coster and Kauchak, 2011b; Hwang et al., 2015; Xu et al., 2015), Portuguese (Caseli et al., 2009), Spanish (Bott and Saggion, 2011), Danish (Klerke and Søgaard, 2012), German (Klaper et al., 2013), Italian (Brunato et al., 2015), and Japanese (Goto et al., 2015). In addition, only the English corpora are open to the public. We therefore propose an unsupervised method [1] that automatically builds monolingual parallel corpora for text simplification without using any external resources for computing sentence similarity.

In this study, a monolingual parallel corpus for text simplification is built from a comparable corpus comprising complex and simple texts. This was done in two steps. First, we compute the similarity for all combinations of complex and simple sentences using the alignment between word embeddings. Second, we extract sentence pairs whose similarity exceeded a certain threshold. Figure 1 gives an overview of the method. Monolingual parallel corpus can be used for text simplification in the framework of SMT.

[1] https://github.com/tmu-nlp/sscorpus

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1147–1158, Osaka, Japan, December 11-17 2016.

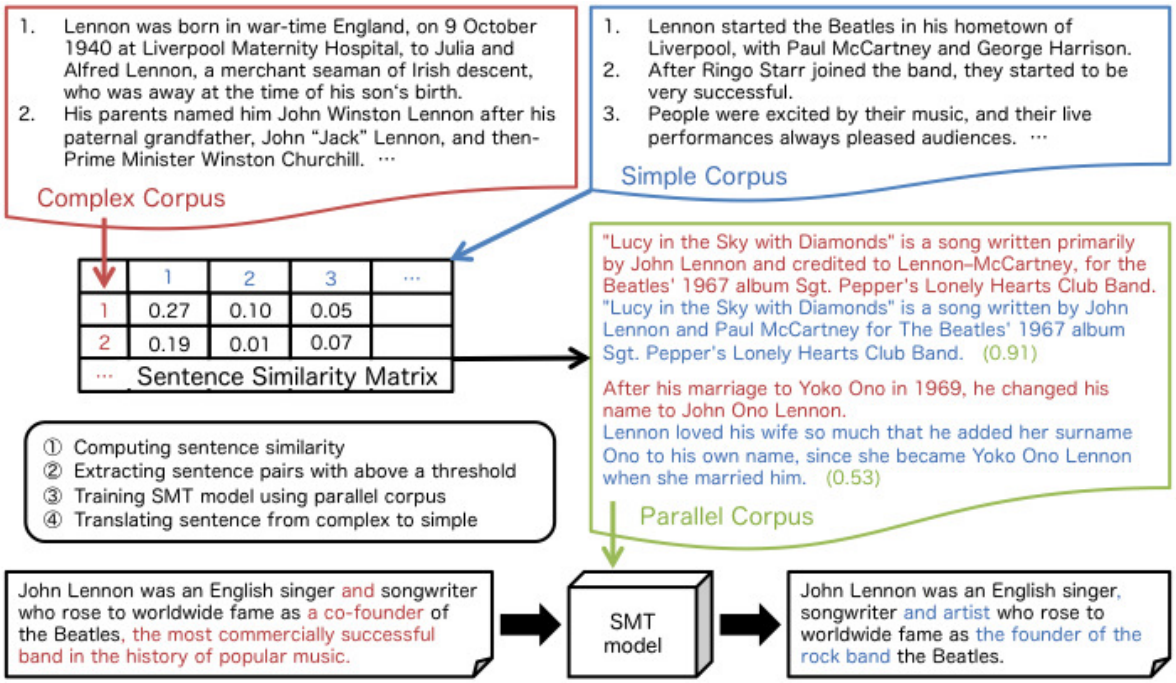


Figure 1: Process flow of building a monolingual parallel corpus and simplifying a sentence using the SMT framework.

We evaluated our proposed method using a benchmarking dataset [2] to construct a corpus for English text simplification. The benchmark dataset contains pairings of complex and simple sentences with a binary label of parallel (the sentence pair is synonymous) or nonparallel (the sentence pair is not synonymous). Intrinsic evaluation using this dataset showed that the proposed method had an improved F1 score. In addition, we built a statistical machine translation model trained on the resulting corpus and compared it with one trained on the existing corpora. Extrinsic evaluation using statistical machine translation for text simplification demonstrated the improved BLEU score of the proposed method.

Our contributions are summarized as follows:

- The proposed method improved the binary classification task between monolingual parallel data and nonparallel data by 3.1 points (0.607 → 0.638), compared with the F1 score from a previous study, and demonstrated high accuracy in building a monolingual parallel corpus for text simplification.
- The SMT-based text simplification model trained using the corpus built by the proposed method had a BLEU score 3.2 points higher (44.3 → 47.5) than an SMT-based text simplification model trained using the state-of-the-art monolingual parallel corpus.
- The proposed method can build a monolingual parallel corpus for text simplification at low cost because it does not require any external resources such as labeled data or dictionaries when computing sentence similarity.

2 Related Work

The statistical machine translation framework has become widely used in text simplification. In English, text simplification using a monolingual parallel corpus extracted from the English Wikipedia and Simple English Wikipedia has been actively studied. Coster and Kauchak (2011b) simplified sentences using the standard phrase-based SMT toolkit Moses (Koehn et al., 2007) and evaluated it using the standard automatic MT evaluation metric BLEU (Papineni et al., 2002). In addition to generic SMT translation models, specialized translation models such as targeting phrasal deletion have been proposed (Zhu et al., 2010; Coster and Kauchak, 2011a; Wubben et al., 2012). These studies reported that models specialized

[2] http://ssli.ee.washington.edu/tial/projects/simplification/

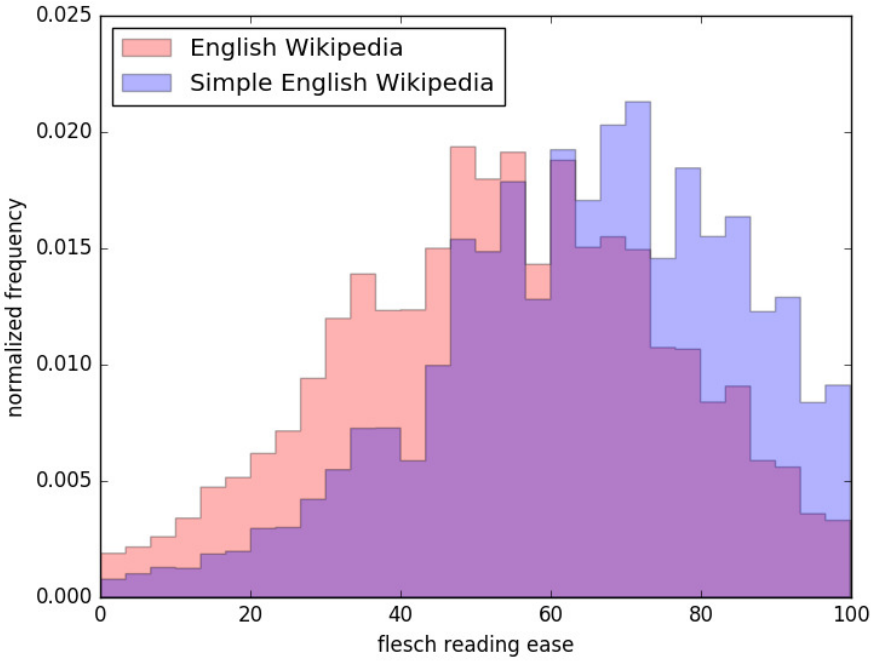


Figure 2: Readability score distribution of English Wikipedia and Simple English Wikipedia. A higher score in Flesch Reading Ease indicates simpler sentences.

in text simplification improved readability and the BLEU score. In languages other than English, text simplification using SMT has been studied for Portuguese (Specia, 2010), Spanish (Štajner et al., 2015b), and Japanese (Goto et al., 2015). We follow these works in applying SMT to text simplification, whilst improving the quality and quantity of the monolingual parallel corpus using an unsupervised method.

Three monolingual parallel corpora for English text simplification have been built from English Wikipedia and Simple English Wikipedia. First, Zhu et al. (2010) [3] pioneered automatic construction of a text simplification corpus using the cosine similarity between sentences represented as TF-IDF vectors. Second, Coster and Kauchak (2011b) [4] extended Zhu et al. (2010)'s work by considering the order of the sentences. However, these methods did not compute similarities between different words. In text simplification, it would be useful to consider similarities between synonymous expressions when computing the similarity between sentences, since concepts are frequently rewritten from a complex to a simpler form. Third, Hwang et al. (2015) [2] computed the similarity between sentences taking account of word-level similarity using the co-occurrence of a headword in a dictionary and its definition sentence. We also consider word-level similarity to compute similarity between sentences but using word embeddings to build a text simplification corpus at low cost without requiring access to external resources.

These text simplification corpora built from English Wikipedia and Simple English Wikipedia received some criticism. Xu et al. (2015) point out that Zhu et al. (2010)'s corpus has 17% of sentence pairs unaligned (two sentences have different meanings or only have partial content overlap) and 33% of sentence pairs become more complex (the simple sentence has the same meaning as the original sentence but is not simpler). However, Simple English Wikipedia contains simpler expressions in general. Figure 2 shows the distribution of the readability scores of Simple English Wikipedia and English Wikipedia. It clearly illustrates that Simple English Wikipedia contains easier sentences than English Wikipedia and supports that it is a good source for text simplification. Štajner et al. (2015a) investigated the quality and quantity of a monolingual parallel corpus using the framework of statistical machine translation and showed that sentence pairs with a moderate level of similarity are effective for training text simplification models. Therefore, we use the sentence similarity method to accurately measure the moderate level of

[3] https://www.ukp.tu-darmstadt.de/data/sentence-simplification/simple-complex-sentence-pairs/

[4] http://www.cs.pomona.edu/~dkauchak/simplification/

similarity.

To address the challenge of computing the similarity between sentences containing different words with similar meanings, many methods have been proposed. In semantic textual similarity task (Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014; Agirre et al., 2015), sentence similarity is computed on the basis of word similarity following the success of word embeddings such as word2vec (Mikolov et al., 2013a). For example, a supervised approach using word embeddings when obtaining a word alignment achieved the best performance in SemEval-2015 Task 2 (Sultan et al., 2015). Word embeddings have also been used in unsupervised sentence similarity metrics (Mikolov et al., 2013b; Song and Roth, 2015; Kusner et al., 2015). These unsupervised sentence similarity metrics can be applied to the automatic construction of a monolingual parallel corpus for text simplification, without requiring the data to be labeled.

3 Sentence Similarity based on Alignment between Word Embeddings

We propose four types of sentence similarity measures for building a monolingual parallel corpus for text simplification, based on alignments between word embeddings that have achieved outstanding performance on different NLP tasks. The methods discussed in Sections 3.1-3.3 are the sentence similarity measures proposed by Song and Roth (2015) for a short text similarity task. The Word Mover's Distance (Kusner et al., 2015) discussed in Section 3.4 is another sentence similarity measure based on alignment between word embeddings that is known to achieve good performance on a document classification task.

3.1 Average Alignment

The sentence similarity $\mathrm{STS}_{\mathrm{ave}}(x, y)$ between sentence x and sentence y is computed by averaging the similarities between all pairs of words taken from the two sentences, as follows:

$$\mathrm{STS}_{\mathrm{ave}}(x, y) = \frac{1}{|\boldsymbol{x}||\boldsymbol{y}|} \sum_{i=1}^{|\boldsymbol{x}|} \sum_{j=1}^{|\boldsymbol{y}|} \phi(x_i, y_j) \tag{1}$$

Here, x_i denotes the i-th word in the sentence x ($\boldsymbol{x} = (x_1, x_2, \ldots, x_{|\boldsymbol{x}|})$), y_j denotes the j-th word in the sentence y ($\boldsymbol{y} = (y_1, y_2, \ldots, y_{|\boldsymbol{y}|})$), and $\phi(x_i, y_j)$ denotes the similarity between words x_i and y_j. We employed the cosine similarity as the word similarity $\phi(x_i, y_j)$.

3.2 Maximum Alignment

Average alignment, discussed in Section 3.1, is an intuitive method. However, it is not possible that all word pairs have a high similarity $\phi(x_i, y_j)$, even when considering synonymous sentence pairs. Moreover, it is often the case that many word similarities $\phi(x_i, y_j)$ are noise and are near to zero. Therefore, we utilize only accurate alignments by computing the sentence similarity $\mathrm{STS}_{\mathrm{asym}}(x, y)$ from the most similar word y_j for each word x_i rather than averaging the word similarities between all pairs. Here $\mathrm{STS}_{\mathrm{asym}}(x, y)$ is an inherently asymmetric score. Therefore, we obtain the symmetric sentence similarity $\mathrm{STS}_{\mathrm{max}}(x, y)$ by averaging the two similarities $\mathrm{STS}_{\mathrm{asym}}(x, y)$ and $\mathrm{STS}_{\mathrm{asym}}(y, x)$ as follows:

$$\mathrm{STS}_{\mathrm{asym}}(x, y) = \frac{1}{|\boldsymbol{x}|} \sum_{i=1}^{|\boldsymbol{x}|} \max_j \phi(x_i, y_j), \quad \mathrm{STS}_{\mathrm{max}}(x, y) = \frac{1}{2}(\mathrm{STS}_{\mathrm{asym}}(x, y) + \mathrm{STS}_{\mathrm{asym}}(y, x)) \tag{2}$$

3.3 Hungarian Alignment

Average alignment and *maximum alignment* can be considered as sentence similarity measures based on many-to-many word alignments and many-to-one word alignments, respectively. However, since these methods compute the word alignments independently, they do not take into account the sentence-level consistency of alignments. To address this lack of global alignment, we represent two sentences x and y as a bipartite graph in which the vertices consist of words that occur in each sentence and the edges reflect their word-level similarity. The graph is then used to define sentence similarity. This bipartite

graph is a weighted complete bipartite graph whose edge is assigned a word similarity $\phi(x_i, y_j)$ as a weight. The one-to-one word alignment that maximizes the sum of the word similarities is obtained by finding the maximum matching of the bipartite graph. This maximum matching problem can be solved using the Hungarian algorithm (Kuhn, 1955). The sentence similarity $\mathrm{STS}_{\mathrm{hun}}(x, y)$ is then computed by selecting a word $h(x_i)$ using the Hungarian algorithm for each word x_i:

$$\mathrm{STS}_{\mathrm{hun}}(x, y) = \frac{1}{\min(|\boldsymbol{x}|, |\boldsymbol{y}|)} \sum_{i=1}^{|\boldsymbol{x}|} \phi(x_i, h(x_i)) \tag{3}$$

3.4 Word Mover's Distance

Word Mover's Distance (Kusner et al., 2015) also considers the global consistency of word alignments when computing sentence similarity based on a many-to-many word alignment. This is a special case of the Earth Mover's Distance (Rubner et al., 1998) which solves the transportation problem of transporting words from sentence x to sentence y.

$$\mathrm{STS}_{\mathrm{wmd}}(x, y) = 1 - \mathrm{WMD}(x, y), \qquad \mathrm{WMD}(x, y) = \min \sum_{u=1}^{n} \sum_{v=1}^{n} \mathcal{A}_{uv} \psi(x_u, y_v) \tag{4}$$

$$\text{subject to} : \sum_{v=1}^{n} \mathcal{A}_{uv} = \frac{1}{|\boldsymbol{x}|} freq(x_u), \qquad \sum_{u=1}^{n} \mathcal{A}_{uv} = \frac{1}{|\boldsymbol{y}|} freq(y_v)$$

Here $\psi(x_u, y_v)$ denotes the dissimilarity (distance) between the two words x_u and y_v. We used the Euclidean distance to denote the word dissimilarity $\psi(x_u, y_v)$. Here, $\mathcal{A}_{uv}$ denotes a weighted matrix of flow from word x_u in the sentence x to word y_v in the sentence y, n denotes the vocabulary size, and $freq(x_u)$ denotes an occurrence frequency of the word x_u in the sentence x.

4 Experiments for Building a Monolingual Parallel Corpus for Text Simplification

We built a monolingual parallel corpus for text simplification by aligning sentences from a comparable corpus using sentence similarity, based on the alignment between word embeddings, and evaluated the effectiveness of the proposed method from the quality of the corpus. First, we evaluated the proposed method in binary classification of a sentence pair as parallel or nonparallel. Next, we built a monolingual parallel corpus for text simplification using the proposed sentence similarity measure, and evaluated it qualitatively. Finally, we trained text simplification models using the SMT framework on our corpus and on existing corpora, to compare their effectiveness.

4.1 Binary Classification between Parallel and Nonparallel Sentences

Hwang et al. (2015) built a benchmark dataset [2] for text simplification extracted from the English Wikipedia and Simple English Wikipedia. They defined four labels: *Good (G) ("The semantics of the sentences completely match, possibly with small omissions.")*, *Good Partial (GP) ("A sentence completely covers the other sentence, but contains an additional clause or phrase that has information which is not contained within the other sentence.")*, *Partial ("The sentences discuss unrelated concepts, but share a short related phrase that does not match considerably.")*, and *Bad ("The sentences discuss unrelated concepts.")*. They annotated 67,853 sentence pairs (277 *G*, 281 *GP*, 117 *Partial*, and 67,178 *Bad*). We classified a sentence pair as parallel or nonparallel using this benchmark dataset to evaluate the sentence similarity measures. We conducted experiments in two settings: a setup (*G vs. O*), in which only sentence pairs labeled *G* were defined as parallel, and the other setup (*G + GP vs. O*), in which sentence pairs labeled either *G* or *GP* were defined as parallel. We evaluated the performance of the binary classification using two measures, the maximum F1 score (MaxF1) and the area under the curve (AUC).

Noise in the word alignment for *average alignment*, *maximum alignment*, and *hungarian alignment* was removed by aligning only those word pairs (x_i, y_j) which had a word similarity $\phi(x_i, y_j) > \theta$. This threshold θ was tuned to maximize MaxF1. We employed 0.89 and 0.95 in the binary classification of *G*

Method		*G vs. O*		*G + GP vs. O*	
		MaxF1	AUC	MaxF1	AUC
Zhu et al.	(Hwang et al., 2015)	0.550	0.509	0.431	0.391
Coster and Kauchak	(Hwang et al., 2015)	0.564	0.495	0.415	0.387
Hwang et al.	(Hwang et al., 2015)	0.712	0.694	0.607	0.529
Additive embeddings		0.691	0.695	0.518	0.487
Average alignment		0.419	0.312	0.391	0.297
Maximum alignment		0.717	0.730	**0.638**	**0.618**
Hungarian alignment		0.524	0.414	0.354	0.275
Word Mover's Distance		**0.724**	**0.738**	0.531	0.499

Table 1: Binary classification accuracy of parallel and nonparallel sentences. *Good (G) vs. Others (O)* is defined for sentence pairs where the label *G* denotes parallel. *G + Good Partial (GP) vs. O* regards the label *GP* as parallel in addition to the label *G*. The labels *G* and *GP* refer to bi- and uni-directional entailment, respectively.

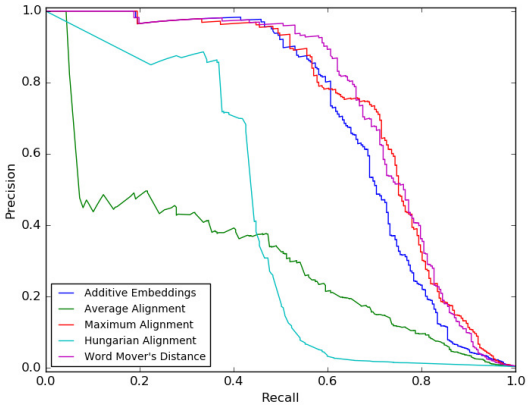


Figure 3: PR curves in binary classification of *G* and *O*.

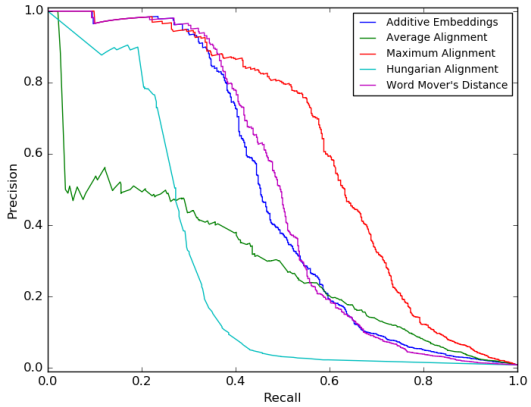


Figure 4: PR curves in binary classification of *G + GP* and *O*.

vs. O and *G + GP vs. O* for *average alignment*, 0.28 and 0.49 in the binary classification of *G vs. O* and *G + GP vs. O* for *maximum alignment*, and 0.98 in the binary classification of *G vs. O* and *G + GP vs. O* for *hungarian alignment*.

Table 1 compares sentence similarity measures in the binary parallel and nonparallel classification task. The top three methods in the upper row are taken from previous studies of monolingual parallel corpus construction for text simplification, and the five methods in the lower rows are the sentence similarity measures based on the word embeddings. *Additive embeddings* provides yet another baseline method, in which sentence embeddings are composed by adding word embeddings without word alignment, and sentence similarity is computed using the cosine similarity between sentence embeddings. We used publicly available [5] pretrained word embeddings to compute sentence similarity. From Table 1, it can be seen that *Word Mover's Distance* performed best in the binary classification task between *G vs. O*, whereas *maximum alignment* performed best in the binary classification task between *G + GP vs. O*.

Figures 3 and 4 show the Precision-Recall curves in the binary classification task between parallel and nonparallel sentences. Figure 4 shows that *maximum alignment* performed better than the other sentence similarity measures based on word embeddings, in the binary classification between *G + GP vs. O*.

Text simplification must take account not only of paraphrases from a complex expression to a simple expression but also of the deletion of unimportant parts of a complex sentence. It is therefore important to include both *G* sentence pairs, where the simple sentence is synonymous with the complex sentence, and *GP* sentence pairs, where the complex sentence entails the simple sentence. For this reason, *maximum alignment*, which performed best in classification between *G + GP vs. O*, was the preferred measure for

[5] https://code.google.com/archive/p/word2vec/

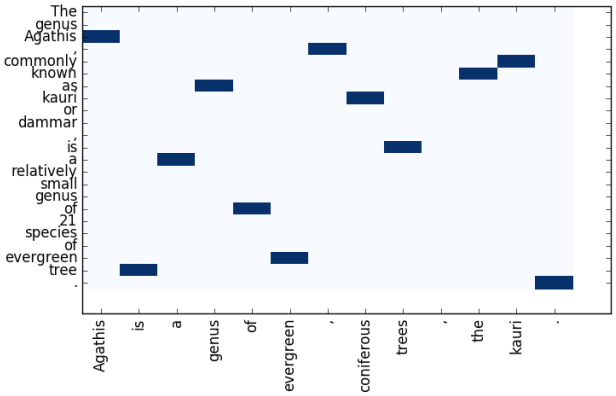


Figure 5: Hungarian word alignment matrix. A vertical axis indicates a sentence from English Wikipedia. A horizontal axis indicates a sentence from Simple English Wikipedia.

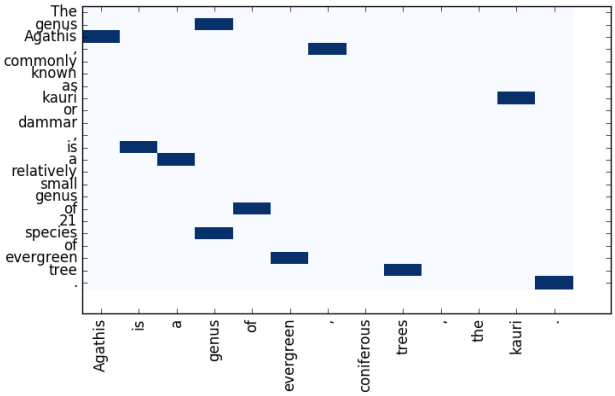


Figure 6: Maximum word alignment matrix. A vertical axis indicates a sentence from English Wikipedia. A horizontal axis indicates a sentence from Simple English Wikipedia.

computing sentence similarity in text simplification.

The experimental results demonstrate the effectiveness of *maximum alignment* in text simplification tasks, but why *maximum alignment* is the best? We present two illustrative figures to explain the reason. First, in *hungarian alignment* (Figure 5), false word alignments such as "as, genus," "tree, is," and "commonly, kauri" are found because of the restriction of one-to-one word alignment on the whole. Second, in *maximum alignment* (Figure 6), correct word alignments such as "genus, genus," "species, genus," "tree, trees," and "kauri, kauri" are found because many-to-one word alignment is searched greedily. It may identify ambiguous pairs such as "genus, genus" and "species, genus," but symmetrization of many-to-one alignment succeeds in reducing this type of noisy alignment. The restriction of *hungarian alignment* is too strict to correctly align content words between the sentences since even function words need to be aligned one-by-one. Also, in text simplification tasks, many-to-one alignment is more appropriate than one-to-one alignment because paraphrase between a phrase and a word occurs frequently.

4.2 Building an English Text Simplification Corpus

We built a monolingual parallel corpus for text simplification from English Wikipedia (normal) [6] and Simple English Wikipedia (simple) [7] using the *maximum alignment* that performed best in the previous experiment. First, we paired articles from the normal and simple editions by an exact match of titles, obtaining 126,725 article pairs. Sentence extraction using WikiExtractor [8] and tokenization using NLTK 3.2.1 [9] gave an average number of words per sentence of 25.1 for the normal articles and 16.9 for the simple articles. The average numbers of sentences per article were 57.7 and 7.65, respectively.

We computed the sentence similarity of all pairings of normal and simple sentences using *maximum alignment*. We based the threshold for word similarity and sentence similarity on the experimental results shown in Table 1. We aligned only those word pairs with a word similarity equal to or greater than 0.49, and aligned only those sentence pairs with a sentence similarity equal to or greater than 0.53. As a result, we obtained 492,993 sentence pairs from 126,725 article pairs.

Table 2 shows examples from the monolingual parallel corpus for text simplification with sentence similarity. We found synonymous expressions (purchased → bought) in sentence pairs with a similarity greater than 0.9 and deletion of unimportant parts of a sentence (such as ...) in sentence pairs with a similarity equal to or greater than 0.7. We also found sentence pairs with only a few words in common with a similarity less than 0.7.

[6] https://dumps.wikimedia.org/enwiki/20160501/
[7] https://dumps.wikimedia.org/simplewiki/20160501/
[8] https://github.com/attardi/wikiextractor/
[9] http://www.nltk.org/

similarity	normal	simple
0.9	Woody Bay Station was **purchased** by the Lynton and Barnstaple Railway Company in 1995 and, after much effort, a short section of railway reopened to passengers in 2004.	Woody Bay Station was **bought** by the Lynton and Barnstaple Railway Company in 1995 and, after much effort, a short section of railway reopened to passengers in 2004.
0.8	This work continued with the 1947 paper "Types of polyploids: their classification and significance", which **detailed a system for the classification of polyploids and** described Stebbins' ideas about the role of paleopolyploidy in angiosperm evolution**, where he argued that chromosome number may be a useful tool for the construction of phylogenies**.	This work continued with the 1947 paper "Types of polyploids: their classification and significance", which described Stebbins' ideas about the role of paleopolyploidy in angiosperm evolution.
0.7	Mir **has been** a significant influence on late 20th-century art, in particular the American abstract expressionist artists **such as Motherwell, Calder, Gorky, Pollock, Matta and Rothko, while his lyrical abstractions and color field paintings were precursors of that style by artists such as Frankenthaler, Olitski and Louis and others**.	Mir **was** a significant influence on late 20th-century art, in particular the American abstract expressionist artists.
0.6	**The couple** has **four children:**	**She** has **two daughters and two sons.**
0.5	Ithaca is **in the rural Finger Lakes region about northwest of New York City; the nearest larger cities, Binghamton and Syracuse, are an hour's drive away by car, Rochester and Scranton are two hours, Buffalo and Albany are three.**	Ithaca is **a city in upstate New York, America.**

Table 2: Examples from our text simplification corpus ranked by similarity.

4.3 English Text Simplification

We trained SMT-based text simplification models using our corpus and existing text simplification corpora (Zhu et al., 2010; Coster and Kauchak, 2011b; Hwang et al., 2015). The results were compared to evaluate the effectiveness of our text simplification corpus. We treated text simplification as a translation problem from the normal sentence to the simple one and modeled it using a phrase-based SMT trained as a log linear model. In each corpus, we randomly sampled 500 sentence pairs for tuning with MERT (Och, 2003) and used the remainder for training. Moses was used as the phrase-based SMT tool. We employed GIZA++ (Och and Ney, 2003) to obtain the word alignment, and KenLM (Heafield, 2011) to build the 5-gram language model from the entire Simple English Wikipedia [7]. As test data, we used 277 sentence pairs labeled *G* and 281 sentence pairs labeled *G + GP* from the Hwang et al. (2015) dataset and evaluated the accuracy using BLEU.

Table 3 shows the number of sentences, range of vocabulary, average number of words per sentence, and BLEU scores of the text simplification models trained on each corpus. The text simplification model trained on our corpus achieved the best BLEU score. To compare the learning curves of our corpus with that from Hwang et al. (2015), we recorded the BLEU scores while changing the corpus size. We discovered that the difference in performance was not only due to the corpus size, as the BLEU scores of the model trained on our corpus remained higher than those of the model trained on the Hwang et al. (2015) corpus at all corpus sizes. The model trained on the Coster and Kauchak (2011b) corpus performed slightly better than that trained on our corpus in 100,000 sentence pairs of a *G* (bi-directional entailment) test set; however, in a *G + GP* (uni-directional entailment) test set that requires more various

Text simplification corpus	#sents.	#vocabulary		Avg. #words per sent.		BLEU	
		normal	simple	normal	simple	*G*	*G + GP*
Baseline (None)						42.1	22.3
Zhu et al.	100,000	173,463	143,030	21.2	17.4	41.8	22.1
Zhu et al. (All)	107,516	181,459	149,643	21.2	17.4	42.0	22.1
Coster and Kauchak	100,000	112,744	102,418	23.7	21.1	43.8	23.4
Coster and Kauchak (All)	136,862	132,567	120,620	23.6	21.1	44.3	23.8
Hwang et al.	100,000	117,474	103,427	25.3	21.2	42.9	22.7
Hwang et al. (G)	154,305	152,419	133,825	25.2	21.2	42.9	22.7
Hwang et al.	200,000	175,416	145,773	25.6	20.5	43.1	22.7
Hwang et al. (G + GP)	284,238	212,138	164,979	26.0	19.8	43.9	23.1
Hwang et al.	300,000	217,699	167,945	26.1	19.7	42.9	22.7
Hwang et al. (All)	391,116	248,510	184,521	26.5	19.4	43.1	22.8
Ours	100,000	122,390	112,670	23.9	21.8	43.2	23.6
Ours	200,000	180,776	151,815	24.7	20.1	45.7	24.8
Ours	300,000	219,628	174,576	25.2	19.0	46.4	25.3
Ours (All)	492,493	274,775	198,043	25.3	17.9	**47.5**	**26.3**

Table 3: SMT-based English text simplification performance. *Baseline* does not do any simplification.

Input	Mozart's Clarinet Concerto and Clarinet Quintet are both in A major, and generally Mozart was more likely to use clarinets in A major than in any other key besides E-flat major.
Reference	Mozart used clarinets in A major often.
Zhu et al.	Mozart's Clarinet Concerto and Clarinet Quintet are both in A major, and generally Mozart **which he** more likely to use clarinets in A major than in any other key besides E-flat major.
Coster and Kauchak	**Mozart was** Clarinet Concerto and Clarinet Quintet are both in A major, and **Mozart used clarinets in A major often**.
Hwang et al.	Mozart's Clarinet Concerto and Clarinet Quintet are both in A major, and generally Mozart was more likely to use clarinets in A major than in any other key besides E-flat major.
Ours	Mozart's Clarinet Concerto and Clarinet Quintet are both in A major, and **Mozart used clarinets in A major often**.

Table 4: Examples of text simplification trained on different text simplification corpora.

substitution and phrasal definition, the model trained on our corpus performed slightly better than their corpus.

Our corpus gave a larger difference in the average number of words between normal and simple sentences than the other corpora, with values closer to the average numbers of words per sentence in the entire Wikipedia (25.1 and 16.9, respectively). This suggests that *maximum alignment* was able to compute sentence similarity more accurately than the other measures regardless of the sentence length.

Table 4 shows examples of text simplification trained on different text simplification corpora. The model trained on our corpus generated a simple sentence that appropriately entailed the reference. The model trained on the Coster and Kauchak (2011b) corpus simplified the input sentence appropriately but also performed incorrect substitutions and generated an ungrammatical sentence. The model trained on the *G + GP* part of the Hwang et al. (2015) corpus did not rewrite the input sentence. The model trained on the Zhu et al. (2010) corpus used almost the same amount of Coster and Kauchak (2011b)'s corpus but performed incorrect substitutions and generated an ungrammatical sentence. Coster and Kauchak (2011b) extended Zhu et al. (2010)'s work by considering the order of sentences. In a Wikipedia article, sentences are arranged in a characteristic order, e.g., a definition sentence appears in the first sentence. Therefore, they may obtain similar sentence pairs effectively. In contrast, our simple proposed method achieved equivalent or higher performance than their method without considering any ordering of sentences.

5 Conclusions

We proposed an unsupervised method for building a monolingual parallel corpus for text simplification. Four types of sentence similarity metric were proposed, based on alignment between word embeddings. Experimental results demonstrated the effectiveness of the sentence similarity measure using many-to-one word alignment to align each word in the complex sentence with the most similar word in the simple sentence. Our proposed method achieved state-of-the-art performance in both an intrinsic evaluation based on classifying sentence pairs from the English Wikipedia and Simple English Wikipedia into a parallel and nonparallel data, and in an extrinsic evaluation in which a complex sentence was translated into a simple sentence.

We successfully built an English monolingual parallel corpus for text simplification from comparable corpus with different levels of difficulty. However, such large-scale comparable corpus is unavailable in many languages. In future work, we will build a monolingual parallel corpus from a raw corpus by combining our sentence similarity measure with a readability assessment. Specifically, we will divide the raw corpus into complex and simple corpora based on readability, and use the paired corpora to align complex sentences with simple ones. This approach should be applicable to any language, and open new opportunities in the field of text simplification.

Acknowledgements

This research was (partly) supported by Grant-in-Aid for Research on Priority Areas, Tokyo Metropolitan University, Research on social bigdata.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 385–393, Montréal, Canada.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic Textual Similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 81–91, Dublin, Ireland.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263, Denver, Colorado, USA.

Stefan Bott and Horacio Saggion. 2011. An Unsupervised Alignment Algorithm for Text Simplification Corpus Construction. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 20–26, Portland, Oregon, USA.

Dominique Brunato, Felice Dell'Orletta, Giulia Venturi, and Simonetta Montemagni. 2015. Design and Annotation of the First Italian Corpus for Text Simplification. In *Proceedings of The 9th Linguistic Annotation Workshop*, pages 31–41, Denver, Colorado, USA.

Helena M. Caseli, Tiago F. Pereira, Lucia Specia, Thiago A.S. Pardo, Caroline Gasperin, and Sandra M. Aluísio. 2009. Building a Brazilian Portuguese Parallel Corpus of Original and Simplified Texts. In *Advances in Computational Linguistics, Research in Computer Science*, pages 59–70, Mexico City, Mexico.

William Coster and David Kauchak. 2011a. Learning to Simplify Sentences Using Wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 1–9, Portland, Oregon, USA.

William Coster and David Kauchak. 2011b. Simple English Wikipedia: A New Text Simplification Task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 665–669, Portland, Oregon, USA.

Isao Goto, Hideki Tanaka, and Tadashi Kumano. 2015. Japanese News Simplification: Task Design, Data Set Construction, and Analysis of Simplified Text. In *Proceedings of MT Summit XV*, pages 17–31, Miami, Florida, USA.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning Sentences from Standard Wikipedia to Simple Wikipedia. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 211–217, Denver, Colorado, USA.

David Klaper, Sarah Ebling, and Martin Volk. 2013. Building a German/Simple German Parallel Corpus for Automatic Text Simplification. In *Proceedings of the Second Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pages 11–19, Sofia, Bulgaria.

Sigrid Klerke and Anders Søgaard. 2012. DSim, a Danish Parallel Corpus for Text Simplification. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, pages 4015–4018, Istanbul, Turkey.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Harold W. Kuhn. 1955. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2:83–97.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From Word Embeddings To Document Distances. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 957–966, Lille, France.

Tomas Mikolov, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at the International Conference on Learning Representations*, Scottsdale, Arizona, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, Lake Tahoe, Nevada, USA.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 1998. A Metric for Distributions with Applications to Image Databases. In *Proceedings of the Sixth International Conference on Computer Vision*, pages 59–66, Washington, DC, USA.

Yangqiu Song and Dan Roth. 2015. Unsupervised Sparse Vector Densification for Short Text Similarity. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1275–1280, Denver, Colorado, USA.

Lucia Specia. 2010. Translating from Complex to Simplified Sentences. *Lecture Notes in Computer Science*, 6001:30–39.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2015. DLS@CU: Sentence Similarity from Word Alignment and Semantic Vector Composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 148–153, Denver, Colorado, USA.

Sanja Štajner, Hannah Bechara, and Horacio Saggion. 2015a. A Deeper Exploration of the Standard PB-SMT Approach to Text Simplification and its Evaluation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 823–828, Beijing, China.

Sanja Štajner, Iacer Calixto, and Horacio Saggion. 2015b. Automatic Text Simplification for Spanish: Comparative Evaluation of Various Simplification Strategies. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pages 618–626, Hissar, Bulgaria.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence Simplification by Monolingual Machine Translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 1015–1024, Jeju Island, Korea.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in Current Text Simplification Research: New Data Can Help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A Monolingual Tree-based Translation Model for Sentence Simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1353–1361, Beijing, China.

Word2Vec vs DBnary: Augmenting METEOR using Vector Representations or Lexical Resources?

Christophe Servan[1,2], Alexandre Bérard[1], Zied Elloumi[1,3], Hervé Blanchon[1] & Laurent Besacier[1]

[1]LIG – Univ. Grenoble Alpes
Domaine Universitaire
38401 St Martin d'Hères, France
firstname.lastname@imag.fr

[2]SYSTRAN
5 Rue Feydeau
75002 Paris, France
firstname.lastname@systran.fr

[3]LNE
29 Avenue Roger Hennequin
78190 Trappes, France
firstname.lastname@lne.fr

Abstract

This paper presents an approach combining lexico-semantic resources and distributed representations of words applied to the evaluation in machine translation (MT). This study is made through the enrichment of a well-known MT evaluation metric: METEOR. This metric enables an approximate match (synonymy or morphological similarity) between an automatic and a reference translation. Our experiments are made in the framework of the *Metrics* task of WMT 2014. We show that distributed representations are a good alternative to lexico-semantic resources for MT evaluation and they can even bring interesting additional information. The augmented versions of METEOR, using vector representations, are made available on our *Github* page.

1 Introduction

Learning vector representations of words using neural networks has generated a strong enthusiasm in the NLP research community. In particular, many contributions were proposed after the work of (Mikolov et al., 2013a; Mikolov et al., 2013b; Mikolov et al., 2013c) on training word embeddings. The main reasons for this strong interest are: the proposal of a simple and efficient neural architecture to learn word vector representations, the availability of an open source tool *Word2Vec*[1] and the rapid structuring of a user community[2]. Later on, several contributions have extended the work of Mikolov on word vectors to phrases (sequences of words) (Mikolov et al., 2013b; Le and Mikolov, 2014a) and to bilingual representations (Luong et al., 2015). All these vector representations capture similarities between words, phrases or sentences at different levels (morphological, semantic).

However, although these representations can be semantically informative, they do not exactly replace fine-grained information available in lexical-semantic resources such as *WordNet* (Fellbaum, 1998), *BabelNet* (Navigli and Ponzetto, 2010), or *DBnary* (Sérasset, 2012). Such lexical resources are also more easily interpretable by humans as shown in (Panchenko, 2016), but their construction is costly while word embeddings can be trained *ad infinitum* on any monolingual or bilingual corpora.

In short, both approaches (lexical resources and word embeddings) have their pros and cons. However, few studies have attempted to compare and combine them. Pioneering work of Faruqui et al. (2014) proposed to refine representations learning using lexical resources. The idea is to force words connected in the lexical network, to have a close representation (for example through a synonymy link). The technique proposed is evaluated on several benchmarks (word similarity, sentiment analysis, finding of synonyms). More recently, Panchenko (2016) and Rothe and Schütze (2015) extended word embeddings to sense embeddings and tried to compare them to lexical synsets.

[1]http://word2vec.googlecode.com/svn/trunk/

[2]https://groups.google.com/d/forum/word2vec-toolkit

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1159–1168, Osaka, Japan, December 11-17 2016.

Contributions: this article attempts to review the contribution of vector representations to measure sentence similarity. We compare them with similarity measures based on lexical resources such as *WordNet* or *DBnary*. Machine Translation (MT) evaluation was identified as a particularly interesting application to investigate, since MT evaluation is still an open problem nowadays. More precisely, we propose to augment a well known MT evaluation metric (METEOR (Banerjee and Lavie, 2005)) which allows an approximate matching (through synonymy or morphological similarity) between MT hypothesis and reference. The augmented versions of METEOR proposed (using word embeddings, lexical resources or both) allow us to objectively compare the contribution of each approach to measure sentence similarity. For this, correlations between METEOR and human judgements (of MT outputs) are measured within the framework of WMT 2014 *Metrics* task. The code of the augmented versions of METEOR is also provided on our *Github* page[3].

Outline: in section 2 (Related Work), we quickly present METEOR, lexical resources and word embeddings. Section 3 presents our propositions to augment METEOR in order to conduct a fair comparison between lexical resources and vector representations respectively. Section 4 presents our experiments made within the framework of WMT 2014, as well as quantitative and qualitative analyses. Finally, section 5 concludes this work and gives some perpectives.

2 Related Work

2.1 An automatic metric for MT evaluation: METEOR

2.1.1 The origins

METEOR was proposed to compensate BLEU's and NIST's weaknesses (Papineni et al., 2002; Doddington, 2002). In short, METEOR was created to better correlate with human judgements by using more than word-to-word alignments between a hypothesis and some references. The alignment is made according to three *modules*: the first stage uses exact match between word surface forms (*Exact* module), the second one compares word stems (*Stems* module) and the third one uses synonyms (*Synonym* module) from a lexical resource such as WordNet (available for English only in METEOR).

One contribution of this paper is to propose an alternative to *Stems* and *Synonym* modules: our proposed add-on will be called *Vectors* module later on.

2.1.2 Recent extensions of METEOR

METEOR-NEXT (Denkowski and Lavie, 2010a) was proposed to better correlate with HTER (Human-targeted Translation Edit Rate – *HTER* (Snover et al., 2006)). HTER is a semi-automatic post-editing based metric, which measures the edit distance between a hypothesis and a reference. METEOR-NEXT proposes to go further than just word-to-word alignment by using phrase-to-phrase alignments. For this, phrase databases were created for several languages like English (Snover et al., 2009), German, French or Czech (Denkowski and Lavie, 2010b). More recently, another version called *METEOR Universal* used bitexts to extract paraphrases (Denkowski and Lavie, 2014).

METEOR was also extended by using Word Sense Disambiguation (WSD) techniques (Apidianaki and Marie, 2015). The authors used *Babelfly* (Moro et al., 2014) for several langage pairs (translation from French, Hindi, German, Czech and Russian to English). A better correlation with human judgement at segment level was observed using WSD in METEOR.

Finally, to extend the use of *Synonym* module to target languages others than English, Elloumi et al. (2015) proposed to replace WordNet by DBnary (Sérasset, 2012). The new target languages equipped with a *Synonym* module were French, German, Spanish, Russian and English.

2.2 Lexical resources

2.2.1 WordNet

WordNet is a well known lexical resource for English. Created at the University of Princeton (Fellbaum, 1998), it is used in several NLP tasks such as Machine Translation, Word Sense Disambiguation,

[3] https://github.com/cservan/METEOR-E

Cross-lingual Information Retrieval, etc. WordNet links nouns, verbs, adjectives and adverbs to a set of synonyms called "synsets". Each synset represents a specific concept.

Synsets are linked to each other according to semantic, conceptual and lexical relations. Words with multiple meanings correspond to multiple synsets and meanings are sorted according to their frequency. WordNet is available in several languages (Arabic, French, etc.) but these versions are not freely available. In METEOR, only English WordNet is used to match hypothesis and reference words according to their meanings. It contains more than 117,000 synsets.

To extract lemmatized forms, METEOR uses a function called *Morphy-7WN1* which firstly checks special cases in an exception list and secondly uses rules to lemmatize words according to their syntactic class.

2.2.2 DBnary

DBnary is a multilingual lexical resource in RDF format (Klyne and Carroll, 2004). This resource has been collected by Sérasset (2012). Lexical data are represented using the LEMON vocabulary (McCrae et al., 2011). Most Part-of-Speech tags are linked with *Olia standards* or *Lexinfo* vocabularies (Chiarcos and Sukhareva, 2015; Cimiano et al., 2011) which makes them reusable in many contexts.

DBnary is downloadable or available online through a SPARQL access point. Lexical data are automatically extracted from Wiktionary, Wikipedia's dictionary for 21 languages[4].

	English	French	Russian	German
Number of entries	620 K	322 K	185 K	104 K
Number of meanings	498 K	416 K	176 K	116 K
Number of synsets	35 K	36 K	31 K	33 K

Table 1: Detail of the data used from DBnary for the languages targeted in this paper.

Among available lexical data, one may find 2.9M lexical entries (with parts-of-speech, canonical form for all of them, along with pronunciations when available and inflected forms for some languages). Lexical entries are subdivided into 2.5M lexical senses (with their definitions and some usage example).

DBnary also contains more than 4.6M translations going from the 21 extracted sources languages to more than 1500 different target languages. Additionally, DBnary contains lexicosemantic relations (syno/anto-nyms, hypo/hypero-nyms, etc.). Table 1 shows the size of the data for languages involved in the experiments later reported in this paper. Additional figures are available on the DBnary public web site[5].

Lemmatized forms for DBnary are based on the *TreeTagger* module (Schmid, 1995), which enables us to find the corresponding synsets.

2.3 Monolingual and bilingual embeddings

2.3.1 Overview

Learning word embeddings is an active research area (Bengio et al., 2003; Turian et al., 2010; Collobert et al., 2011; Huang et al., 2012). The main idea is to learn a word representation according to its context: the surrounding words (Baroni and Zamparelli, 2010). The words are projected on a continuous space and those with similar context should be close in this multi-dimensional space. When word vectors are available, a similarity between two words can be measured by a metric such as a cosine similarity.

Using word-embeddings for machine translation evaluation is appealing since they can be used to compute similarity between words or phrases in the same language (monolingual embeddings capture intrinsically synonymy or morphological closeness) or in two different languages (bilingual embeddings allow to directly compute a distance between two sentences in different languages). We use the *MultiVec* (Bérard et al., 2016) toolkit for computing and managing the continuous representations of texts. It includes word2vec (Mikolov et al., 2013a), paragraph vector (Le and Mikolov, 2014b) and bilingual distributed representations (Luong et al., 2015) features.

[4]Bulgarian, Dutch, English, Finnish, French, German, (Modern) Greek, Indonesian, Italian, Japanese, Latin, Lithuanian, Malagasy, Norwegian, Polish, Portuguese, Russian, Serbo-Croat, Spanish, Swedish and Turkish

[5]`http://kaiko.getalp.org/about-dbnary/`

2.3.2 Use of vector representations in NLP evaluation

Zou et al. (2013) proposed to use bilingual word embeddings to detect similarities for word alignment. This information is used as an additional parameter in a phrase-based machine translation system. (Banchs et al., 2015) proposed to explore a metric funded on latent semantic analysis (Salton et al., 1975) to extract semantic embeddings and measure the similarity between two sentences. Finally, these word embeddings were used to enrich ROUGE, a metric for evaluating automatic summarization (Ng and Abrecht, 2015).

As far as MT evaluation is concerned, Gupta et al. (2015) proposed a metric based on neural network language models jointly with dependency trees to link an hypothesis to a reference. Meanwhile, Vela and Tan (2015) proposed an approach to model document embeddings to predict translation adequacy.

These works are close to ours but they propose metrics which need to be learned and optimized to a specific task or domain. In our work, we use word embeddings trained once and for all on a (large) general corpus. Our detailed methodology to augment METEOR metric is presented in the next section.

3 Augmented METEOR

3.1 Data and protocol

We evaluate our augmented METEOR through WMT14 framework (*metrics* task (Machacek and Bojar, 2014)). This framework enables us to estimate the correlation of proposed evaluation metric with human judgements for several machine translation outputs and several language pairs (English-French, English-German, English-Russian, and vice versa). In our experiments, we use segment level Kendall's τ correlation coefficient, as proposed in WMT14 (based on systems ranking at sentence level by humans, compared to automatic metric ranking).

We augment METEOR in two ways: firstly, we replace the use of lexical resources by the use of word embeddings. In other words, we replace *Stem* and *Synonym* modules by our new *Vector* module. Secondly, we combine lexical resources and word embeddings by using jointly *Stem*, *Synonym* and our *Vector* module.

To summarize, the following variants of METEOR are evaluated:

- *METEOR Baseline*: the METEOR score is estimated using *Exact*, *Stem*, *Synonym* and *Paraphrase* modules for English as a target language and *Exact*, *Stem* and *Paraphrase* modules for other target languages,
- *METEOR DBnary*: similar to *METEOR Baseline* but *Synonym* module is available for any target language since it uses DBnary resource instead of Wordnet,
- *METEOR Vector*: the *Stem* and *Synonym* modules are replaced by the *Vector* module ;
- *METEOR Baseline + Vector*: the *METEOR Baseline* configuration is augmented with the *Vector* module ;
- *METEOR DBnary + Vector*: the *METEOR DBnary* configuration is augmented with the *Vector* module.

3.2 *METEOR DBnary*

As mentioned in section 2.1, the *Synonym* module of METEOR uses WordNet's synsets (117K entries for English). As an alternative, we use another lexical resource: DBnary (Sérasset, 2012), as proposed recently by Elloumi et al. (2015). This allows us to use *Synonym* module for any target language: French, German, Spanish, Russian and English.

More precisely, synonym relations are extracted from DBnary using SPARQL request on the DBnary server[6]. We extract data for English, French, Russian and German languages. The extraction process outputs relations in the following format: $lemma \rightarrow Synonym$. Then, these data are projected to the

[6] http://kaiko.getalp.org/about-dbnary/online-access/

WordNet format used in METEOR code. This process gives an identifier (ID) for each lemma and builds a list of synonym IDs for each lemma such as: $lemma \rightarrow ID\_Syn_1,\ ID\_Syn_2,\ ID\_Syn_3$.

The first two lines of Table 3 compare *METEOR DBnary* and *METEOR Baseline* for several French-English MT systems submitted to WMT14 (Bojar et al., 2014).

METEOR DBnary improved the score by 0.7 points from *METEOR Baseline*. In other words, DBnary seems to match more synonyms than WordNet, despite the fact that WordNet is 3.3 time bigger than DBnary in English. This could be due to the fact that WordNet has only 4 morpho-syntactic categories (Noun, Verbs, Adjectives and Adverbs) while DBnary has more morpho-syntactic categories.

3.3 *METEOR Vector*

As mentioned in section 2.3.2, we propose to replace lexical resources by word embeddings. Word embeddings capture the context of the words. Consequently, similar word vectors may correspond to synonyms or morphological variants (see section 2.3).

Language	Corpora	# of lines	# of source words	# of target words
French–English	Europarl V7 + news commentary V10	2.2 M	67.2 M	60.7 M
German–English	Europarl V7 + news commentary V10	2.1 M	57.2 M	59.7 M
Russian–English	Common Crawl + news commentary V10 + Yandex	2.0 M	47.2 M	50.3 M

Table 2: Bilingual corpora used to train the word embeddings for each language pair.

In our *Vector* module, the matching between two words is done using a similarity score derived from the cosine similarity. If the similarity score is higher than a threshold, the words are considered as matched (potential synonyms or morphological proximity). In our experiments, we evaluate using: (a) a default threshold fixed to 0.80 (b) an oracle threshold obtained empirically on the WMT14 data set (Machacek and Bojar, 2014).

Table 2 summarizes data used to train monolingual word embeddings and bilingual word embeddings. These word embeddings were trained with a CBOW model, a vector size of 50 and a windows size ± 5 words, thanks to the MultiVec toolkit (Bérard et al., 2016).

Metrics	Systems: online A	online B	online C	rbmt 1	rbmt 4
METEOR Baseline	36.33	36.71	31.19	33.00	31.65
METEOR DBnary	36.93	37.33	32.01	33.69	32.42
METEOR Vector	37.00	37.34	31.87	33.67	32.34
METEOR Baseline + Vector	37.08	37.40	31.96	33.75	32.45
METEOR DBnary + Vector	37.53	37.88	32.60	34.32	33.05

Table 3: METEOR scores (all configurations) on the *newstest* corpus of the WMT14 translation evaluation task from French to English.

The results presented in table 3 show that word embeddings (*Vector* module) can efficiently replace lexical resources (*Synonym* and *Stem* modules) to match words in the translation hypothesis with those in the reference. In addition, their combination shows a good potential to match even more words between hypothesis and reference. In the next section, we evaluate if the proposed versions of *augmented* METEOR better correlate with human judgements.

4 Correlations of Augmented METEOR with Human Judgements

4.1 Results of different METEOR configurations

In these experiments, we present results obtained with the *Vector* module based on two threshold values: a default one (0.80) and an oracle one which maximizes the correlation with human judgement.

Table 4 presents the correlation scores obtained within the framework of WMT14 metrics task (Machacek and Bojar, 2014)[7]. The evaluation is done according to several translation tasks: from English to French (en–fr), German (en–de) and Russian (en–ru), and vice versa. French, German and Russian

[7]For better readability, we do not add standard deviations in the tables. These numbers will be, however, provided in supplementary material put on the paper web page (`https://github.com/cservan/METEOR-E/paper`).

as target languages represent a growing difficulty due to their morphology. English as target language allows to compare the lexical databases (Wordnet *vs* DBnary).

To English. Firstly, when the translation direction is to English, we can observe that *METEOR Baseline* and *METEOR Vector* get equivalent results in average. *METEOR DBnary* also obtains similar results to *METEOR Baseline*. When we combine WordNet lexical resource and word embeddings (*METEOR Baseline + Vector*), the reference score is increased by $0,005$ points. If the combination is done with DBnary's lexical data (*METEOR DBnary + Vector*), the improvement is similar.

For *Vector* module, optimization of the threshold slightly improves the average correlation. Combination of *METEOR Baseline + Vector* or *METEOR DBnary + Vector* improves by $0,002$ points when the threshold is optimized.

From English. Secondly, when the translation direction is from English, we can observe an improvement of the correlation score obtained with *METEOR DBnary*, compared with *METEOR Baseline*. This is due to the fact that for French, German and Russian as target languages, *METEOR Baseline* does not use any *Synonym* module. Our *METEOR Vector* with the default threshold also gets better correlation scores compared to *METEOR DBnary* ($+0.003$ points in average). The combinations *METEOR Baseline + Vector* and *METEOR DBnary + Vector* further improve correlations with human judgements ($+0.001$ points in average). Finally, when we use an oracle threshold for *Vector* module, improvements are bigger and can reach 0.013 points in average, compared to *METEOR Baseline*.

Language pairs	fr–en		de–en		ru–en		Average	
Metric	Threshold	τ	Threshold	τ	Threshold	τ	Threshold	τ
METEOR Baseline	–	0.406	–	0.334	–	0.329	–	0.356
METEOR DBnary	–	0.408	–	0.334	–	0.328	–	0.357
METEOR Vector	0.80	0.407	0.80	0.332	0.80	0.328	0.80	0.356
METEOR Baseline + Vector	0.80	0.407	0.80	0.343	0.80	0.332	0.80	0.361
METEOR DBnary + Vector	0.80	0.407	0.80	0.337	0.80	0.338	0.80	0.361
METEOR Vector	0.89	0.411	0.78	0.333	0.80	0.328	0.82	0.357
METEOR Baseline + Vector	0.73	0.412	0.80	0.343	0.88	0.333	0.80	0.363
METEOR DBnary + Vector	0.73	0.413	0.79	0.338	0.80	0.338	0.77	0.363
Language pairs	en-fr		en-de		en-ru		Average	
Metric	Threshold	τ	Threshold	τ	Threshold	τ	Threshold	τ
METEOR Baseline	–	0.280	–	0.238	–	0.427	–	0.315
METEOR DBnary	–	0.284	–	0.239	–	0.435	–	0.319
METEOR Vector	0.80	0.290	0.80	0.241	0.80	0.436	0.80	0.322
METEOR Baseline + Vector	0.80	0.288	0.80	0.241	0.80	0.440	0.80	0.323
METEOR DBnary + Vector	0.80	0.289	0.80	0.242	0.80	0.439	0.80	0.323
METEOR Vector	0.72	0.295	0.79	0.241	0.72	0.439	0.74	0.325
METEOR Baseline + Vector	0.86	0.296	0.79	0.242	0.79	0.445	0.81	0.328
METEOR DBnary + Vector	0.88	0.294	0.75	0.245	0.79	0.443	0.81	0.327

Table 4: Correlation score at segment level between several METEOR configurations and human judgements (WMT14 framework). Scores obtained with the *Vector* module are presented firstly with the default threshold (0.80) and secondly with the oracle threshold (under the dashed line).

4.2 Investigating more embeddings configurations

In the previous section, *METEOR Vector* used a simple and monolingual word embedding configuration. This section investigates more configurations (monolingual and bilingual) to improve METEOR.

In this experiment, we focus only on *METEOR Vector*. Indeed, the *monolingual (baseline)* shown in table 6 corresponds to the line *METEOR Vector* in Table 4. Firstly, we propose to train our embeddings on bitexts (Table 2) using *bivec* approach (Luong et al., 2015). We also try to train pseudo-bilingual embeddings on a pseudo bitext with target language text and POS tags (see an example in Table 5). The main idea is to strongly link words with their syntactic class when learning word embeddings. We

madam president , on a point of order .	$\Leftrightarrow$	NOUN NOUN PUNCT ADP DET NOUN ADP NOUN PUNCT

Table 5: Example of bitext where the target side is replaced by POS.

call this kind of model *pseudo-bilingual with POS*. In the same way, we train bilingual models called *pseudo-bilingual with lemmas*, where the POS tags are replaced by lemmas. In addition, we also learn word embeddings with lemmas only and bilingual models with lemmas only.

Models:	*monolingual (baseline)*	*bilingual*	*pseudo-bilingual with POS*	*pseudo-bilingual with lemmas*	*monolingual (lemmas)*	*bilingual (lemmas)*
To English	0.356	0.354	0.355	0.354	0.357	0.357
From English	0.322	0.322	0.320	0.325	0.324	0.318

Table 6: Average correlation score at segment level for *METEOR Vector* with several training configurations of word embeddings with the default threshold (0.80).

In the Table 6, we compare several training configuration of the word embeddings through the same protocol as previous section (only average correlations are reported while the detailed results will be provided as supplementary material on the paper web page). When we observe the average results, the *bilingual* embeddings seem not to be as efficient as the monolingual baseline. The pseudo-bilingual approaches with POS and Lemmas obtained slightly the same results as the monolingual baseline regarding all the configurations we have. Finally, the monolingual model learned on lemmas (instead of words) tends to be slightly better when the translation direction is to English. However, this trend should be confirmed in a future investigation.

4.3 Discussion

The correlation scores obtained with the enriched metric tend to suggest that distributed representations are as powerful as lexico-semantic resources for automatic MT evaluation. Furthermore, vector representations can bring additional information, and they are definitely useful when no lexical resource is available in the target language.

Considering the average correlation scores obtained, the configurations *METEOR Vector* and *METEOR DBnary* are comparable, except on German language, for which *METEOR Vector* obtained a better correlation score. On the other hand, when we combine lexical data with *Vector* module (*METEOR DBnary + Vector*), we observe a small increase of the correlation score, in particular when threshold is tuned, which suggests a tunable version of METEOR.

Finally, several embeddings variants were trained but it seems that monolingual models are efficient enough for the specific task (MT evaluation) considered here.

4.3.1 Examples

To illustrate the word matching obtained by our versions of METEOR, we analyze two examples from the evaluation data set. In these examples, we present the alignments obtained with *METEOR DBnary + Vector*.

hypothesis:	je pense qu' il est concevable que ces données soient employées pour le bénéfice mutuel .
Reference:	j' estime qu' il est concevable que ces données soient utilisées dans leur intérêt mutuel .

Table 7: First example from the system *rbmt 1* evaluated with the combination *METEOR DBnary + Vector*. The relations detected with the lexical resource DBnary are framed in continuous line while those obtained thanks to the distributed representations are framed in dotted line.

The example presented in table 7 shows *rbmt 1* system output submitted during the WMT14 translation task. *METEOR baseline* found only alignments for words with the same surface forms (*"qu' ", "il", "est"*, etc. – these forms are found identical thanks to the *Exact* module and are not highlighted here). The *Synonym* module based on DBnary makes it possible to find a correspondence between words *"employées" – "utilisées"* and *"pour" – "dans"*. Lastly, *Vector* module indicates that words *"pense"* and *"estime"* are contextually closed, just as the words *"je"* and *"j"'*. When the example is only evaluated with *METEOR Vector*, words *"employées"* and *"utilisées"* are also paired with the default threshold (0.80). On the other hand, the words *"bénéfice"* and *"intérêt"* are paired by the module *Vector* only if the decision threshold is lowered to 0.75.

In the second example presented in table 8, the hypothesis is provided by *rbmt 4* system. As in the previous example, the correspondences found with *Synonym* module based on DBnary (framed by one

hypothesis:	le créateur de SAS disait il faisait un genre du feuilleton géopolitique .
Reference:	le père de SAS disait faire un genre de feuilleton géopolitique .

Table 8: Another example scored with the combination *METEOR DBnary + Vector*.

continuous line) are supplemented by those found by *Vector* module (dotted line): *Synonym* module found *"créateur"* – *"père"* and *"faisait"* – *"faire"*; while *"du"* and *"de"* are aligned thanks to *Vector* module.

These examples illustrate the complementarity between lexical resources and word embeddings for sentence similarity detection. Word vectors can enable to match important words (like *"pense"* and *"estime"* in our first example), but also empty words (like *"du"* et *"de"* in our second example).

4.3.2 Limitations of Word Embeddings

So far, we did not deal with Out-Of-Vocabulary (OOV) words in *METEOR Vector*. By OOV we mean words that do not have a vector representation because they were not found in the training corpus for word embeddings. In that case, no matching can occur between the word in the hypothesis and words in reference. Consequently, it might be interesting to carefully select the training corpus for word vectors so that it will be close enough to the machine translation outputs to evaluate. This could be considered in future works.

5 Conclusion and Perspectives

In this paper, we proposed to compare text similarity measures based on vector representations with similarity measures based on lexico-semantic resources. Our work was applied to machine translation evaluation and we extended an existing evaluation metric called METEOR. Our experiments have shown that word vector representations can be useful when no lexical resource is available in the target language. Moreover, it seems that these representations can bring complementary information in addition to lexical resources (experiments done for French, English, German and Russian as target languages).

Our future works on this topic will focus on the use of phrase embeddings to complement the *Paraphrase* module of METEOR. We also plan to introduce a *syntax flavor* in our *Vector* module by weighting the cosine distances differently according to the morpho-syntactic category of the words. Finally, we will study the adaptation of our approach to other metrics such as TER-Plus, for instance.

The tool, the data and the models presented in this paper will be put online[8] to facilitate reproducibility of the experiments we carried out.

Acknowledgements

This work was supported by the KEHATH project funded by the French National Agency for Research (ANR) under the grant number ANR-14-CE24-0016-03.

References

Marianna Apidianaki and Benjamin Marie. 2015. METEOR-WSD: Improved Sense Matching in MT Evaluation. In *the Proceedings of the 9th Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST'9)*.

Raphael E. Banchs, Luis F. D'Haro, and Haizhou Li. 2015. Adequacy–Fluency Metrics: Evaluating MT in the Continuous Space Model Framework. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):472–482, March.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43th Annual Meeting of the Association of Computational Linguistics*.

[8] `https://github.com/cservan/METEOR-E`

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

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

Alexandre Bérard, Christophe Servan, Olivier Pietquin, and Laurent Besacier. 2016. MultiVec: a Multilingual and Multilevel Representation Learning Toolkit for NLP. In *The 10th edition of the Language Resources and Evaluation Conference (LREC 2016)*.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 Workshop on Statistical Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Christian Chiarcos and Maria Sukhareva. 2015. Olia – ontologies of linguistic annotation. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):379–386.

P. Cimiano, P. Buitelaar, J. McCrae, and M. Sintek. 2011. Lexinfo: A declarative model for the lexicon-ontology interface. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(1):29 – 51.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Michael Denkowski and Alon Lavie. 2010a. Extending the meteor machine translation evaluation metric to the phrase level. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Michael Denkowski and Alon Lavie. 2010b. METEOR-NEXT and the METEOR paraphrase tables: Improved evaluation support for five target languages. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*.

Michael Denkowski and Alon Lavie. 2014. METEOR Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*.

Zied Elloumi, Hervé Blanchon, Gilles Serasset, and Laurent Besacier. 2015. METEOR for Multiple Target Languages using DBnary. In *Proceedings of MT Summit 2015*.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2014. Retrofitting word vectors to semantic lexicons. *CoRR*.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MA: MIT Press.

Rohit Gupta, Constantin Orasan, and Josef Van Genabith. 2015. Machine Translation Evaluation using Recurrent Neural Networks. In *Proceedings Workshop on Machine Translation (WMT), Metrics Shared Task*.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*.

Graham Klyne and Jeremy J. Carroll. 2004. Resource Description Framework (RDF): Concepts and Abstract Syntax. Technical report.

Quoc V. Le and Tomas Mikolov. 2014a. Distributed Representations of Sentences and Documents. In *Proceedings of The 31st International Conference on Machine Learning*.

Quoc V. Le and Tomas Mikolov. 2014b. Distributed Representations of Sentences and Documents. In *Proceedings of the 31th International Conference on Machine Learning (ICML'14)*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*.

Matous Machacek and Ondrej Bojar. 2014. Results of the WMT14 Metrics Shared Task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

John McCrae, Dennis Spohr, and Philipp Cimiano, 2011. *Linking Lexical Resources and Ontologies on the Semantic Web with Lemon*, pages 245–259. Springer Berlin Heidelberg, Berlin, Heidelberg.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *The Workshop Proceedings of the International Conference on Learning Representations (ICLR) 2013*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

Roberto Navigli and Simone Paolo Ponzetto. 2010. Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*.

Jun-Ping Ng and Viktoria Abrecht. 2015. Better Summarization Evaluation with Word Embeddings for ROUGE. In *In The Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Alexander Panchenko. 2016. Best of both worlds: Making word sense embeddings interpretable. In *the 10th edition of the Language Resources and Evaluation Conference (LREC 2016)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL*.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

G. Salton, A. Wong, and C. S. Yang. 1975. A Vector Space Model for Automatic Indexing. *ACM*, 18(11):613–620, November.

Helmut Schmid. 1995. Treetagger| a language independent part-of-speech tagger. *Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart*, 43:28.

Gilles Sérasset. 2012. Dbnary: Wiktionary as a lemon-based multilingual lexical resource in rdf. *Semantic Web Journal-Special issue on Multilingual Linked Open Data*, 6(4):355–361.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*.

Matthew Snover, Nitin Madnani, Bonnie J Dorr, and Richard Schwartz. 2009. Fluency, adequacy, or HTER?: exploring different human judgments with a tunable MT metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*.

Mihaela Vela and Liling Tan. 2015. Predicting Machine Translation Adequacy with Document Embeddings. In *Proceedings Workshop on Machine Translation (WMT), Metrics Shared Task*.

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. 2013. Bilingual Word Embeddings for Phrase-Based Machine Translation. In *EMNLP*.

Broad Twitter Corpus: A Diverse Named Entity Recognition Resource

Leon Derczynski
Dept. of Computer Science
University of Sheffield
S1 4DP, UK
leon.d@shef.ac.uk

Kalina Bontcheva
Dept. of Computer Science
University of Sheffield
S1 4DP, UK
k.bontcheva@shef.ac.uk

Ian Roberts
Dept. of Computer Science
University of Sheffield
S1 4DP, UK
i.roberts@shef.ac.uk

Abstract

One of the main obstacles, hampering method development and comparative evaluation of named entity recognition in social media, is the lack of a sizeable, diverse, high quality annotated corpus, analogous to the CoNLL'2003 news dataset. For instance, the biggest Ritter tweet corpus is only 45 000 tokens – a mere 15% the size of CoNLL'2003. Another major shortcoming is the lack of temporal, geographic, and author diversity. This paper introduces the Broad Twitter Corpus (BTC), which is not only significantly bigger, but sampled across different regions, temporal periods, and types of Twitter users. The gold-standard named entity annotations are made by a combination of NLP experts and crowd workers, which enables us to harness crowd recall while maintaining high quality. We also measure the entity drift observed in our dataset (i.e. how entity representation varies over time), and compare to newswire. The corpus is released openly, including source text and intermediate annotations.

1 Introduction

Businesses, governments, and communities increasingly need real-time information from dynamic, large-volume media data streams, such as blogs, Facebook, and Twitter. In particular, the automatic detection of mentions of people, organizations, locations, and other entities (i.e. Named Entity Recognition) is a key step in numerous social media analysis applications, e.g. competitor and brand monitoring (Mostafa, 2013), debate and election analysis (Mascaro and Goggins, 2012; Tumasjan et al., 2010), disaster response (Kedzie et al., 2015; Neubig et al., 2011), and health- and well-being applications (Coppersmith et al., 2014; Choudhury et al., 2013).

NER methods (typically trained on longer texts, such as news articles), have been shown to perform poorly on shorter and noisier social media content (Ritter et al., 2011). Therefore, recent Twitter NER work (Ritter et al., 2011; Liu et al., 2011; Derczynski et al., 2015) has focused on improving the state-of-the-art, through new methods. The challenges come from named entities (NEs) typically being out-of-vocabulary (OOV) as compared to the training newswire data; the shorter context; and lack of sufficiently large NE annotated social media datasets.

In more detail, Table 1 shows that there are less than 90 thousand tokens of publicly available NE-annotated tweet datasets, and even those have shortcomings in terms of annotation methodology (e.g. singly annotated), low inter-annotator agreement, and stripping of important entity-bearing hashtags and user mentions. At the same time, Ritter et al. (2011) demonstrated that blending gold-standard newswire training data with social media training data - in order to increase the size of the dataset and better generalize - leads to worse performance. Lastly, as this paper demonstrates, existing NER datasets suffer from poor temporal diversity, making the trained NER models sensitive to the entity drift phenomenon.

We address these challenges through building a sizeable manually-annotated corpus – the Broad Twitter Corpus (BTC). In order to maximize diversity, the BTC is stratified for time, including social media posts from a six-year period, drawn from different parts of year, days of the month, and times of the day. Secondly, it is drawn from different places, accounting for different variants of English and the

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1169–1179, Osaka, Japan, December 11-17 2016.

Corpus	Tokens	Entity schema	Annotator type	Annotator qty.	Notes
Finin et al. (2010)	7K	PLO (3)	Crowd only	Multiple	Low IAA (Fromreide, 2014)
Ritter et al. (2011)	46K	Freebase (10)	Expert	Single	IAA unavailable
Liu et al. (2011)	12K	PLO (3) + Product	?	?	Private corpus
Rowe et al. (2013)	29K	PLO (3) + Misc	Expert	Multiple	No hashtags/usernames
Broad Twitter Corpus	165K	PLO (3)	Expert + Crowd	Multiple	Source JSON available

Table 1: Characteristics of openly-available social media NER corpora. PLO standards for Person, Location, Organization NE types.

different entities found in various regions of the English-speaking world. Finally, it is partially socially segmented, including reactions to news stories, non-professional content, and text from the "twitterati".

The corpus is made freely available in various formats; the source text is included under Twitter's revised 2015 licensing guidelines, as are the intermediate annotations.

2 Corpus Construction

The goal of the corpus is to provide a representative example of named entities in social media. Social media is said to contain more variance than some other text types, like newswire. It certainly is authored by a broader demographic than newswire (Eisenstein, 2013), and contains a variety of styles and formality registers, unlike other user-generated content such as Youtube comments or SMS (Hu et al., 2013). Changes in general discourse subject are also said to present more quickly in social media, creating topic shifts of both greater magnitude and higher frequency than other text types. We focus on Twitter, using this as the "model organism" of social media text (Tufekci, 2014), to assemble a corpus that is capable of catching these variances.

2.1 Annotation Scheme

The BTC corpus is divided into segments, where the documents within each segment share a common theme (see Table 2). The documents consist of the social media message – i.e. tweet – complete with its JSON metadata. The text of the tweet is then annotated with into sentences and tokens, using the TwitIE tokenizer (Bontcheva et al., 2013).

Tokenisation presents some issues in tweets. Classic schemas like PTB do not work well with constructs like smilies or URLs. To address this, we use the TwitIE tokeniser (Bontcheva et al., 2013) which is roughly based on TweetMotif and the twokeniser tool (O'Connor et al., 2010). Of note, we separate the preceding symbol in mentions and hashtags (the @ or # characters) as a distinct token, but still include this in entity spans.

The main question was which entity classes should be covered in the corpus. Some Twitter corpora have used ten top-level Freebase categories (Bollacker et al., 2008; Ritter et al., 2011)[1] or have included products (see Table 1). For expert-based annotation methodologies Hovy (2010) recommend at most ten, ideally seven, categories. In crowdsourcing, successful tasks tend to present even fewer choices – in most cases between two and five categories (Sabou et al., 2014).

Therefore, we chose to focus on the three most widely used entity categories: Person, Location, and Organization (Table 1). Apart from being well understood by annotators, these three categories offer straightforward mapping to the other existing Twitter datasets, so all could be used in combination, as training data, if needed. An initial pilot (Bontcheva et al., 2014a) also included a fourth class, "Product", but crowd workers struggled to annotate these correctly and with good recall, so they were dropped.

For polysemous entities, our guidelines instructed annotators to assign the entity class that corresponds to the correct entity class in the given context. For example, in "*We're driving to Manchester*", Manchester is a location, but in "*Manchester are in the final tonight*", it is a sports club – an organization.

Special attention is given to username mentions. Where other corpora have blocked these out (Rowe et al., 2013) or classified them universally as person (Ritter et al., 2011), our approach is to treat these as named entities of any potential class. For example, the account belonging to Manchester United football club would be labeled as an organization.

[1]Categories used: company, facility, geo-location, movie, music artist, person, product, sports team, TV show and other.

Section	Region	Collection period	Description	Annotators	# Tweets
A	UK	2012.01	General collection	Expert	1000
B	UK	2012.01-02	Non-directed tweets	Expert	2000
E	Global	2014.07	Related to MH17 disaster	Crowd & expert	200
F	Stratified	2009-2014	Twitterati	Crowd & expert	2000
G	Stratified	2011-2014	Mainstream news	Crowd & expert	2351
H	Non-UK	2014	General collection	Crowd & expert	2000

Table 2: Segments of the corpus. A region of "stratified" indicates that data was taken from six regions in the English-speaking world that had a sufficient crowdsourcer workforce to ensure annotator diversity.

2.2 Corpus Diversity

In order to maximize diversity of English social media content, our methodology samples tweets along three dimensions: spatial, temporal and social.

The spatial dimension aims to account for linguistic variation in social media across English-speaking countries. In particular, the discussed entities vary from one region to the next, e.g *Justin Trudeau* in Canada vs. *Theresa May* – in the UK. For these reasons, data is collected from the USA, the UK, New Zealand, Ireland, Canada and Australia.[2]

The temporal dimension is key for being able to make models more resilient towards temporal entity drift (Masud et al., 2010; Magdy and Elsayed, 2016), i.e. the change in entities mentioned at different time periods. For example, *George Bush* or *Pamela Anderson* in the 1990s vs Angelina Jolie and Tiger Woods in 2010, or Santa near Christmas and Guy Faulks in UK tweets in the early winter.

We aim to capture this drift in the BTC by collecting posts over a number of years,[3] as well as being taken from different times of years, days in the month, and times of day. In contrast, previous social media NE corpora were gathered during narrow contiguous time periods (Ritter et al., 2011).

The final, social dimension again aims to account for variation in linguistic styles across different kinds of Twitter users. For instance, verbal communication behaviors such as g-dropping are often copied into typed social media messages (Eisenstein et al., 2010). To try to capture these, the corpus collects data from different segments, explicitly taking in content from well-known public figures, news outlets, well-known social media figures, plus a large volume of randomly-selected posts.

2.3 Corpus Segmentation

The dataset is organized into multiple segments (Table 2) to reflect the diversity criteria and annotation approach (expert vs. crowd). English tweets were filtered using langid.py (Lui and Baldwin, 2012).

Segment A comprises a random sample of UK tweets, collected after New Year, annotated by multiple NLP experts. This data was used for calibration, so includes both expert input and crowd correction.[4]

Segment B is similar to segment A. In this segment we focused on non-directed tweets – i.e. those that are not private replies and so do not begin with a username mention. These were found to be more likely to contain a named entity based on sampling the Ritter et al. (2011) corpus.

Segment E is a small sample focused on a specific event, the crash of flight MH17 over Ukraine. It contains commentary from different places and social levels, contributing different kinds of language and reactions. The entities here have a wide range and are generally from outside the English-speaking world, often instead being Ukrainian, Dutch or Malaysian. This provides needed diversity in names, as well as examples of L2 English language use around NEs (i.e., from non native speakers).

Segment F comprises content from popular individuals that provide twitter-based commentary – the "twitterati". These are stratified across the six English-speaking regions listed above, as well as a general global section. Authors are from the worlds of celebrity, music, politics, sports, journalism; they are a

[2]While there are other countries with English as a first language, such as Botswana and Singapore, we were constrained by the number of local crowd workers we could access (see Section 3.1).

[3]Taken from an archive of Twitter "garden hose" data, a fair 10% sample (Kergl et al., 2014).

[4]The crowd correction is via feedback from its use as gold test data; see Section 3.1.

Annotator group	Recall over final annotations	F1 Agreement
Expert	0.309	0.835
Crowd	0.837	0.350

Table 3: Comparison of expert and crowd annotators over segment A, for calibration. Note higher recall but lower agreement in crowd. Agreement is F1 lenient with micro-averaging.

mix of both principals in the fields and also commentators. This content is reasonably unique to social media, often being too low-impact, speculative or controversial to reach mainstream news.

Segment G contains, in contrast, tweets from mainstream news in the six English-speaking regions, e.g. CNN in the US, SMH in Australia, RTE in Ireland, CBC in Canada and so on. Many local outlets that do not have an international edition are also included.

Segment H is the most varied. To balance the UK bias of segment B, this segment excludes tweets of UK origin (according to the Twitter metadata). The segment is stratified for month of year, time of day, and day of week, giving an even spread over many temporal cycle types in the collection period.

3 Annotation Process

To make annotation scalable and of high quality, while ensuring sufficient annotator variety, corpus annotation was carried out using a mix of NLP experts and paid-for crowdsourcing. The annotation process follows general best practices in crowdsourced corpus annotation (Callison-Burch and Dredze, 2010; Alonso and Lease, 2011; Sabou et al., 2014). For example, task design is kept clean (a critical factor, more important than e.g. interface language – (Khanna et al., 2010)), and the process developed over pilot and refinement iterations.

Tasks were built in GATE and jobs automatically managed through through Crowdflower (Bontcheva et al., 2014b). First segments were entirely annotated and adjudicated by experts as calibration. To maximize annotator focus, images attached to tweets or featuring in content (e.g. news stories) linked to from tweets are shown alongside the task, for worker priming (Morris et al., 2012).

A majority of crowd workers use crowd work as their primary income. We calculate task mean completion times and reward work so that it pays *above* the minimum wage in our country.

3.1 Annotator recall

To annotate text with diverse content, annotators with diverse knowledge are needed. Typical named entity cues, such as capital first letters, are often missing in social media; extra knowledge is one way of compensating for these. We introduce the concept of "annotator recall", which describes the ability of annotators to identify NE mentions, partly based on their own knowledge. The breadth of social media content makes annotation harder for experts. For example, none of the experts we asked during annotation (or the attendees of talks we gave on the topic) could initially explain the entity *KKTNY* in the text "*KKTNY in 45mins!!!!!*", without referring to external resources. However, the crowd, being more diverse, was sometimes able to identify and ground this expression.[5]

To assess annotator recall, we compared expert and crowd annotation over segment A. First, this segment was annotated by a mixture of expert annotators, with each document doubly-annotated and results expert adjudicated. Crowd annotators were then also asked to annotate this data using the same guidelines, and the results compared, shown in Table 3.

In general, the crowd found more entities than experts (Table 3). That is to say, crowd recall over the oracle annotation of the data was higher. However, agreement was lower in the crowd. This was handled by including human expert adjudication over all segments. Further, to improve corpus-level knowledge diversity, each worker was limited to a maximum of 50 tasks.[6] Details of the adjudication process are given in Section 3.2.

[5]The entity colloquially refers to a television program, "Kim and Kourtney Take New York"; from Ritter et al. (2011) data.

[6]Each task involves annotating five tweets for one entity type; each tweet was annotated by five to seven different workers.

Agreement level	IAA (max-recall vs. expert)	IAA (naïve)
Whole document	0.839	n/a
Person entity	0.920	0.799
Location entity	0.963	0.861
Organization entity	0.936	0.954
All entities	**0.940**	**0.877**

Table 4: Inter-annotator agreement breakdown.

	Feature	Count
Dataset	Documents	9 551
	Tokens	165 739
Entities	Person	5 271
	Location	3 114
	Organization	3 732
	Total	12 117

Table 5: Corpus size statistics

After merging in these crowd annotations, a random set of 100 documents were taken from this segment and used as gold test data in later tasks. Gold test data is used to assess performance of individual workers online, by being seeded among real annotation tasks. Performance on gold test data estimates the quality of an annotator's work. Annotators are made aware of results of this process and can highlight incorrect gold test data. The gold test data also provides qualification training for workers; they must perform well on gold test data before accessing real tasks. This gold test set was used throughout the remainder of the corpus annotation, providing quality control and crowd annotator training.

Spatial variation also played a role in crowd annotator recall. Based on the crowd vs. expert comparison data, there were multiple cases where the annotation was deemed "easy" by experts, but the crowd would still miss it. Investigation suggested that annotators based in the same country as the origin tweet had better recall on these entities. Conversely, annotators working on documents from other countries had lower recall. As matched geographic contexts tend to produce better results, during annotation, the geographically stratified parts of corpora were issued only to crowd workers in the same region, in order to maximize recall and local knowledge.

3.2 Adjudication

Adjudication is an important step in refining annotator data. In the case of crowdsourced annotations, further economies of scale can be afforded by automating adjudication; tools already exist for this, such as MACE (Hovy et al., 2013). However, auto-adjudication is poorly equipped to handle exceptional circumstances; further, it is hard to judge its impact without human intervention. The construction of the BTC involved a combination of automatic and human adjudication.

Primarily, we found there were problems with recall. Often, only a single crowd worker or expert would annotate a given (correct) entity. Under traditional agreement-based measures, this singleton annotation would be in the minority, and so likely removed in a typical adjudication step. Given our and others' experiences with annotator recall and diversity (Balog et al., 2012; Difallah et al., 2013), we find this method inappropriate. Further, this annotator behavior has implications for inter-annotator agreement (IAA); perfectly adequate aggregate annotations will nevertheless have reduced agreement. Therefore, naïve IAA measures (such as Fleiss' κ) risk under-reporting multiple annotator performance in high-diversity text. Also, while IAA is sometimes used as a correlate of task understanding, here it also reflects annotator world knowledge.

Workers are continually monitored using 100 documents from segment A, which removes workers having too low performance on reference tasks. This ensures annotator quality while simultaneously permitting variance in annotator knowledge.

To manage annotator sparsity (see Section 3.1), we experimented with a simple automatic adjudication rule. Any time a span is annotated by a worker, that span is placed in the final set. Adjacent annotations of the same type are concatenated. This is the "max-recall" method. We apply max-recall to all original expert and crowd annotations; then, an expert adjudicator evaluates annotator max-recall output to provide a final gold version. Hapax legomena are not unusual, given the diverse subject matter, though only retained after best-effort verification by the expert. Agreement figures are given in Table 4, as well as standard annotator performance.

In our experiments, we measured naïve IAA as the proportion of annotators agreeing on each token of an entity. This gives a per-entity agreement figure. The per-entity is then micro-averaged across the entire corpus, to show the level of annotator agreement, with some ability to account for variance in

entity bounds. This method does not give generous score, but fits our scenario of having many annotators, which makes pairwise comparison awkward. Indeed, even under this measurement regime, IAA levels were respectable.

3.3 Results

In total, the final corpus contained 9 551 documents. In these, there were 165K tokens and 12K entity mentions. Details are given in Table 5; the comparison of this dataset to other social media named entity corpora is in Table 1. Over the annotation process, we collected 125K annotations from 755 workers. These comprised eight experts and a total of 747 crowd workers. Inter-annotator agreement was 0.94, with max-recall autoadjudication followed by a final expert confirmation adjudication step.

4 Drift Analysis

We have described three dimensions for variation: temporal, spatial, and social. This section examines the difference in entities found across the corpus.

Year	1996	2009	2010	2011	2012	2013	2014
Our corpus	0	3	5	127	2414	275	6022
Ritter (2011)	0	0	6902	0	0	0	0
CoNLL'03	1358	0	0	0	0	0	0

Table 6: Volume of tweets in corpus by year; darker background means higher volume.

4.1 Data Selection Comparison

The selection of texts affects the kinds of entities and entity contexts that will be found. Getting a good range is important, as discussed in Section 2. The corpus aims to establish a good spatial, temporal and social range of NEs. This section describes the variation present in the corpus along each of these dimensions, and compares it to two other major NER datasets: the CoNLL 2003 shared task data (Tjong Kim Sang and Meulder, 2003), and Ritter et al. (2011)'s NER data.[7] The other social media corpora mentioned earlier are distributed as plain text, without timestamps or messages IDs, which precludes their analysis.

The temporal distribution is described and compared to the CoNLL2003 and Ritter datasets by year (Table 6), month (Table 7), day of month (Table 8), day of week (Table 9), and time of day (Table 10).[8] Temporal data is based on the local time. Note that both the Ritter and CoNLL data are confined to narrow ranges, the former having been collected in a short period on one day, and the latter having training data from a few days in the summer of 1996 and test data from later the same year (6/7 December). These two corpora are therefore constrained to anachronistic terminology, with the Ritter data also having socially constrained data (i.e., it is only from people active during one Friday evening – September 17, 2010).

4.2 Spatial Entity Diversity

Different regions are likely to mention different entities. The corpus contains two segments that are subdivided into various country-specific strata. To measure the spatial diversity of named entities, we compare the density of entity mentions that are specific to each region, against the rest of the segment.

[7]Many thanks to Alan Ritter for providing references to the original tweet IDs, which allowed this metadata to be captured.
[8]No time of day data was present in the CoNLL dataset.

Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Our corpus	2308	68	502	862	1074	1056	1321	850	342	419	23	21
Ritter (2011)	0	0	0	0	0	0	0	0	6902	0	0	0
CoNLL'03	0	0	0	0	1	0	0	1131	1	0	1	224

Table 7: Volume of tweets in corpus by month; darker background means higher volume.

Day of month	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Our corpus	559	162	187	163	186	191	541	174	187	197	200	545	201	165	274	265	905	276	254	253	203	170	667	304	217	273	303	250	207	230	137
Ritter (2011)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6902	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CoNLL'03	1	0	0	0	8	165	51	0	0	0	0	0	0	0	0	0	0	0	0	0	7	125	103	66	85	111	121	168	130	107	110

Table 8: Volume of tweets in corpus by day of month; darker background means higher volume.

Weekday	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Our corpus	999	1115	2062	2016	1019	1027	608
Ritter (2011)	0	0	0	0	6902	0	0
CoNLL'03	111	122	174	262	376	227	86

Table 9: Volume of tweets in corpus by day of week.

Time of day	morning	afternoon	evening	night
Our corpus	1873	3299	2156	1518
Ritter (2011)	0	0	6902	0

Table 10: Volume of tweets by time of day.

Novel entities are calculated as the proportion of all entities that occur uniquely in one region. This is based on the assumption that if entities are distributed evenly across the world, i.e. region has no effect, then entity mentions sampled from any given region will roughly match the total. Conversely, if entity mentions do vary by region, then the proportion of novel entities in any coherent region will be higher. For calibration, we also include the global portion of segment F and all of the global segment H. Singleton mentions are removed to smooth the data. Results are given in Table 11, showing high regional variance. Most regions' surface forms are novel, i.e. unique to that region; the U.S. may have the lowest proportion of novel NEs because U.S. users make up a large proportion of global users. The "global" category has the largest size. This is required to give a broad enough picture of globally common entities to make inter-regional comparisons meaningful. The side effect a disproportionate novel part.

5 Entities in Social Media

Having examined diversity in the underlying text, we next analyze characteristics of entities. We qualitatively examine surface forms, and compare entity distribution in social media to that in newswire.

5.1 Common Surface Forms

Table 12 presents the most frequent surface forms in our corpus and also in the CoNLL'03 NER annotated data. The latter comes from news, based on the RCV1 corpus, which is largely US-based newswire from the 1990s (Rose et al., 2002) written by white working-age men (Eisenstein, 2013).

Temporal concept drift (Masud et al., 2010) is evident here. For example, the most frequently-mentioned person entities have different surface forms, while referring to the same concept. The lexical representation of "the President of the US" has changed from *Clinton* to *Obama*. Similarly, the leader of Russia is present but with a different word; *Yeltsin* in the older newswire, *Putin* in modern social media.

The top locations mentioned remain largely the same level and granularity, being countries that are major actors on the global scale or in the Anglosphere. Extra presence in the social media data of items like *Dublin*, *Ontario* and *Melbourne* may be attributable to our sampling, which more evenly distributed across English speaking nations than a population-weighted or social media presence-weighted approach.

We also see that celebrity figures, such as *@justinbieber* and *Kate Middleton*, are more prevalent in the social media top ranking – as are journalists, such as *@timhudak* and *David Speers*. However, the CoNLL data does contain a large number of sportsmen, as it is rich in cricket reportage; e.g. *Wasim*

[9]Segment E has Russian and Ukrainian media coverage, promoting mentions of this entity. This excludes RT as "retweet".

Region	Distinct surface forms	Forms only in this category	% Novel
Australia	116	95	81.90
Canada	109	94	86.24
Ireland	105	83	79.05
New Zealand	58	52	89.66
United Kingdom	203	144	70.94
United States	135	84	62.22
Global	628	582	92.68

Table 11: Proportions of entities specific to individual regions on social media.

Person		Location		Organization	
Our corpus	**CoNLL'03**	**Our corpus**	**CoNLL'03**	**Our corpus**	**CoNLL'03**
Obama	Clinton	UK	U.S.	Independent	Reuters
President Obama	Yeltsin	Ukraine	Germany	Irish News	U.N.
Kate Middleton	Arafat	US	Australia	RT *(Russia Today)*[9]	CHICAGO
@justinbieber	Lebed	London	France	Twitter	NEW YORK
Putin	Dole	Canada	England	Facebook	OSCE
JudithCollins	Wasim Akram	Iraq	Russia	Malaysia Airlines	NATO
@SimoLove	Waqar Younis	Russia	Britain	BBC	Interfax
Prince William	Mushtaq Ahmed	Australia	Italy	YouTube	EU
James Foley	Dutroux	Irish	China	Reuters	Barcelona
Harper	Croft	Gaza	LONDON	twitter	KDP
David Cameron	Netanyahu	U.S.	Spain	Liverpool	DETROIT
Cameron	Bill Clinton	Dublin	Japan	Labour	USDA
Princess Beatrice	Aamir Sohail	NZ	Sweden	CNN	PUK
Tony Abbott	Mullally	Ireland	Israel	Arsenal	MINNESOTA
Kate	Mother Teresa	Ontario	Pakistan	WorldCup	BOSTON
@timhudak	Wang	Melbourne	NEW YORK	Apple	ST LOUIS
@RossMarowits	Saeed Anwar	USA	Iraq	UN	PHILADELPHIA
NICOLA	Moin Khan	China	Belgium	Guardian	TORONTO
@David_Speers	Salim Malik	Syria	London	Google	Surrey
Zara Phillips	Rubin	Scotland	United States	EU	European Union

Table 12: Top 20 most frequent surface forms of each entity type, in the CoNLL'03 data (newswire from RCV1) and from our Twitter data.

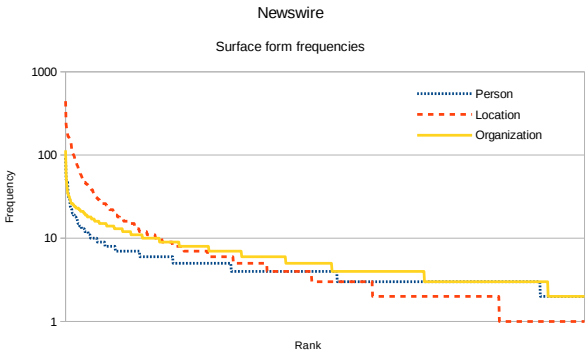


Figure 1: Frequency-Rank curve for entities in CoNLL'03 data.

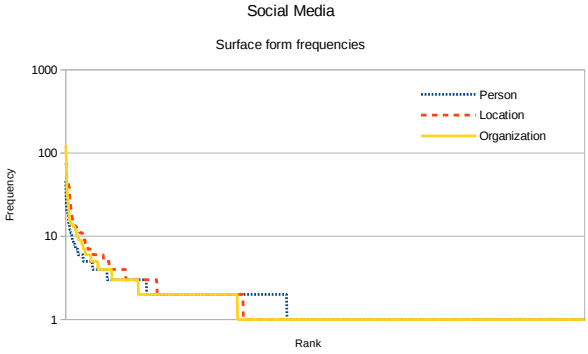


Figure 2: Frequency-Rank curve for entities in the Broad Twitter Corpus.

Akram and *Moin Khan*. This may be attributable to the specific social stratification of the newswire subsample found in the CoNLL data; i.e., it covers little outside of global and business affairs, and only one sport. Alternatively, a cricket championship may have been underway during the narrow date range from which this data was sampled. In any event, one would not expect these names to occur in modern datasets. Further, context around cricketers probably makes poor training examples for general Person entities. In contrast, social media data focuses on a broad range of popular figures and even personal names not found in the public sphere (like *NICOLA*).

Regarding organizations, both datasets are rich in sports clubs, with the typical location/organization metonymy found in that regard (e.g. *DETROIT* can refer to a place or sports club). The CoNLL data is rich in major league baseball data, but poor in its editorial reportage, explaining the high incidence of U.S. sport club mentions but with no frequent baseball personalities. BTC data includes large social media and other internet-based enterprises among its most frequently mentioned organizations. Both datasets also frequently discuss news outlets, possibly due to self-promotion in their own coverage.

5.2 Surface Form Distribution

The frequencies of entity mentions can be used to describe the corpus. We measure frequency-rank curves over the Broad Twitter Corpus and compare them to newswire data (the CoNLL'03 dataset). Frequency rank curves for all three entity types are shown in Figures 1 and 2.

	Social media			Newswire		
	Person	**Location**	**Organization**	**Person**	**Location**	**Organization**
Total weight	5 271	3 114	3 732	10 053	10 572	9 268
h-index	11	15	14	18	44	23
Tail weight	5 048	2 547	3 263	9 399	6 098	8 425
Head proportion	4.23%	18.21%	12.57%	6.51%	42.32%	9.10%

Table 13: Measuring the contribution of the head of entity surface form frequency/rank curve.

In a corpus covering a very narrow topic range, one might expect a small range of entities to be mentioned frequently, and comprise the majority of entity mentions in that corpus. This could be called head-heavy, as the entity mention frequency mass is concentrated in the head of the curve, not the tail. To measure the head-heaviness of these distributions, we first determine the h-index for each entity class (i.e. the lowest entity frequency that is larger than the number of times it is seen) and use this to bisect the curve into head and tail. The frequency mass of the head is then measured as a proportion of the whole. Results are given in Table 13, without disambiguating metonymies. Note that, while in general the head makes up for fewer of the entities in social media when compared to newswire, this is not the case for organization entities. These are slightly more diverse in newswire, with the mass of the head making up only 9.1% of all mentions, compared to 12.57% in tweets. This may be due to the focus on major league baseball teams in the news data.

6 Conclusion

Social media is an important resource and presents many challenges, though the paucity and bias of existing datasets hampers NER research in this text type. This paper presents a large, high-quality corpus for social media that addresses these problems, and demonstrated the breadth and quality of the annotated data. The dataset can be freely downloaded from `http://www.gate.ac.uk/wiki/broad-twitter-corpus.html`, including both all original text (under Twitter license) and also tools for reconstructing the annotated corpus.

Acknowledgements

We are grateful to all our annotators, including among others Dominic Rout, Diana Maynard, Konstantin Yershov, Marta Sabou, Roland Roller, Michał Łukasik, Johann Petrak, Patrick Paroubek, Nattapong Sanchan, Gerhard Wohlgennant, Adam Funk, Amel Fraisse, Arno Scharl, and our many crowd workers across the globe. Alan Ritter of Ohio State University gave us excellent help with references to the original JSON for the 2011 corpus, pivotal to our analyses. Graham Dove of Aarhus University gave tips for our visualisations, which benefit both us and hopefully the reader. This research was supported by the EU under FP7 grant No. 611223, PHEME, and by the UK EPSRC under the CHIST-ERA scheme from grant No. EP/K017896/1, uComp.

References

Omar Alonso and Matthew Lease. 2011. Crowdsourcing for information retrieval: principles, methods, and applications. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1299–1300. ACM.

Krisztian Balog, Yi Fang, Maarten de Rijke, Pavel Serdyukov, and Luo Si. 2012. Expertise retrieval. *Foundations and Trends in Information Retrieval*, 6(2–3):127–256.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Kalina Bontcheva, Leon Derczynski, Adam Funk, Mark A. Greenwood, Diana Maynard, and Niraj Aswani. 2013. TwitIE: An Open-Source Information Extraction Pipeline for Microblog Text. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*. Association for Computational Linguistics.

Kalina Bontcheva, Leon Derczynski, and Ian Roberts. 2014a. Crowdsourcing named entity recognition and entity linking corpora. *The Handbook of Linguistic Annotation (to appear)*.

Kalina Bontcheva, Ian Roberts, Leon Derczynski, and Dominic Rout. 2014b. The GATE Crowdsourcing Plugin: Crowdsourcing Annotated Corpora Made Easy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Association for Computational Linguistics.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 1–12.

Munmun De Choudhury, Scott Counts, Eric Horvitz, and Michael Gamon. 2013. Predicting depression via social media. In *Proceedings of ICWSM*. AAAI, July.

Glen Coppersmith, Mark Dredze, and Craig Harman. 2014. Quantifying mental health signals in Twitter. In *Associational for Computational Linguistics Workshop of Computational Linguistics and Clinical Psychology*.

Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, and Kalina Bontcheva. 2015. Analysis of named entity recognition and linking for tweets. *Information Processing and Management*, 51:32–49.

Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. 2013. Pick-a-crowd: tell me what you like, and I'll tell you what to do. In *Proceedings of the 22nd international conference on World Wide Web*, pages 367–374. ACM.

Jacob Eisenstein, Brendan O'Connor, Noah Smith, and Eric P. Xing. 2010. Discovering sociolinguistic associations with structured sparsity. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proceedings of NAACL-HLT*, pages 359–369.

Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in Twitter data with crowdsourcing. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 80–88.

Hege Fromreide, Dirk Hovy, and Anders Søgaard. 2014. Crowdsourcing and annotating NER for Twitter #drift. In *Proceedings of LREC*, pages 2544–2547. European Language Resources Association.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proc. of NAACL-HLT*, pages 1120–1130.

Eduard Hovy. 2010. Annotation. In *Tutorial Abstracts of ACL*.

Yuheng Hu, Kartik Talamadupula, Subbarao Kambhampati, et al. 2013. Dude, srsly?: The surprisingly formal nature of Twitter's language. *Proceedings of ICWSM*.

Chris Kedzie, Kathleen McKeown, and Fernando Diaz. 2015. Predicting salient updates for disaster summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1608–1617. Association for Computational Linguistics.

Dennis Kergl, Robert Roedler, and Sebastian Seeber. 2014. On the endogenesis of Twitter's Spritzer and Gardenhose sample streams. In *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*, pages 357–364. IEEE.

Shashank Khanna, Aishwarya Ratan, James Davis, and William Thies. 2010. Evaluating and improving the usability of Mechanical Turk for low-income workers in India. In *Proceedings of the first ACM symposium on computing for development*. ACM.

Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. 2011. Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 359–367.

Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *The 50th Annual Meeting of the Association for Computational Linguistics*, volume 4, pages 25–30. ACL.

Walid Magdy and Tamer Elsayed. 2016. Unsupervised adaptive microblog filtering for broad dynamic topics. *Information Processing & Management*, 52(4):513–528.

Christopher Mascaro and Sean Patrick Goggins. 2012. Twitter as virtual town square: Citizen engagement during a nationally televised republican primary debate. In *APSA 2012 Annual Meeting Paper*.

Mohammad M Masud, Qing Chen, Latifur Khan, Charu Aggarwal, Jing Gao, Jiawei Han, and Bhavani Thuraisingham. 2010. Addressing concept-evolution in concept-drifting data streams. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 929–934. IEEE.

Robert R Morris, Mira Dontcheva, and Elizabeth M Gerber. 2012. Priming for better performance in microtask crowdsourcing environments. *Internet Computing, IEEE*, 16(5):13–19.

Mohamed M. Mostafa. 2013. More than words: Social networks text mining for consumer brand sentiments. *Expert Systems with Applications*, 40(10):4241 – 4251.

Graham Neubig, Yuichiroh Matsubayashi, Masato Hagiwara, and Koji Murakami. 2011. Safety information mining – what can NLP do in a disaster. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 965–973. Asian Federation of Natural Language Processing.

Brendan O'Connor, Michel Krieger, and David Ahn. 2010. TweetMotif: Exploratory Search and Topic Summarization for Twitter. In *Proceedings of the Fourth AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 384–385.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proc. of Empirical Methods for Natural Language Processing (EMNLP)*, Edinburgh, UK.

Tony Rose, Mark Stevenson, and Miles Whitehead. 2002. The Reuters Corpus Volume 1 - from Yesterday's News to Tomorrow's Language Resources. In *LREC*, volume 2, pages 827–832.

Matthew Rowe, Milan Stankovic, Aba Sah Dadzie, B.P. Nunes, and Amparo Elizabeth Cano. 2013. Making sense of microposts (#msm2013): Big things come in small packages. In *Proceedings of the WWW Conference - Workshops*.

Marta Sabou, Kalina Bontcheva, Leon Derczynski, and Arno Scharl. 2014. Corpus annotation through crowdsourcing: Towards best practice guidelines. In *Proceedings of the 9th international conference on language resources and evaluation (LREC14)*, pages 859–866.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

Zeynep Tufekci. 2014. Big questions for social media big data: Representativeness, validity and other methodological pitfalls. In *Proceedings of the 8th International AAAI Conference on Weblogs and Social Media*.

Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner, and Isabell M. Welpe. 2010. Predicting elections with Twitter: What 140 characters reveal about political sentiment. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pages 178–185.

Semantic overfitting: what 'world' do we consider when evaluating disambiguation of text?

Filip Ilievski, Marten Postma, Piek Vossen
Vrije Universiteit Amsterdam
De Boelelaan 1105, 1081 HV Amsterdam
The Netherlands
{f.ilievski, m.c.postma, piek.vossen}@vu.nl

Abstract

Semantic text processing faces the challenge of defining the relation between lexical expressions and the world to which they make reference within a period of time. It is unclear whether the current test sets used to evaluate disambiguation tasks are representative for the full complexity considering this time-anchored relation, resulting in semantic overfitting to a specific period and the frequent phenomena within. We conceptualize and formalize a set of metrics which evaluate this complexity of datasets. We provide evidence for their applicability on five different disambiguation tasks. To challenge semantic overfitting of disambiguation systems, we propose a time-based, metric-aware method for developing datasets in a systematic and semi-automated manner, as well as an event-based QA task.

1 Introduction

Semantic processing defines a relation between natural language and a representation of a world it refers to. A challenging property of natural language is the time-bound complex interaction between lexical expressions and world meanings. We use *meaning* in this paper as an umbrella term for both concepts and (event and entity) instances, and *lexical expression* as a common term for both lemmas and surface forms. We can define this interaction as a set of relations, both sense relations and referential relations, that exists within a language community in a certain period of time, e.g. one or a few generations. The people belonging to these generations share one language system that changes relatively slowly but during their lives there are many rapidly changing situations in the world that make certain meanings and expressions dominant and others not. Likewise, we expect that a generation uses a certain set of lexical expressions out of the available set in relation to a set of meanings that balances the trade-off between learning many expressions and resolving extreme ambiguity of a small set of expressions.

The task of interpreting lexical expressions as meanings, known as disambiguation, has been addressed by the NLP community following a "divide & conquer" strategy that mostly ignores this complex time-bound relation. Over the years, this resulted in numerous separate disambiguation tasks each with a specific set of datasets restricted to a small bandwidth with respect to the dynamics of the world and the large scope of the possible meanings that lexical expressions can have. By dividing the problem into different tasks on relatively small datasets, researchers can focus on specific subproblems and have their efforts evaluated in a straightforward manner. Datasets have been developed independently for each task, intended as a test bench to evaluate the accuracy and applicability of the proposed systems. Official evaluation scripts have been created for most datasets to enable a fair comparison across systems.

The downside of this practice is that task integration is discouraged, systems tend to be optimized on the few datasets available for each task, and the dependencies of ambiguities across tasks in relation to the time-bound contextual realities are not considered. As a result, there is little awareness of the overall complexity of the task, given language as a system of expressions and the possible interpretations given the changing world over longer periods of time. Systems are thus encouraged to strongly overfit on a single task, a single dataset, and a specific 'piece' of the world at a specific moment in time.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1180–1191, Osaka, Japan, December 11-17 2016.

Each text forms a unique semantic puzzle of expressions and meanings in which ambiguity is limited within the specific time-bound context, but is extreme without considering this context. The main question we thus put forward and address in this paper is how to enhance disambiguation tasks to cover the full complexity of the time-bound interaction between lexical expressions and meanings (in the broad sense of the word as defined here). We therefore first propose a number of metrics that formally quantity the complexity of this relation and apply this to a wide range of available datasets for a broad range of semantic tasks. Secondly, we provide evidence for the limitations of the current tasks and, thirdly, we present a proposal to improve these tasks in the hope that we challenge future research to address these limitations.

The paper is structured as follows. We motivate the importance and relevance of this temporal interaction for both concept- and instance-based disambiguation tasks in Section 2. Following up on previous research (Section 3), we define a model of the complex interaction (Section 4), and we conceptualize and formalize a collection of metrics in a generic manner (Section 5). Moreover, we apply these metrics to quantify aspects of existing evaluation sets (Section 6). In Section 7, we propose two approaches for creating metric-aware test sets that include a temporal dimension. The paper is concluded in Section 8.

2 Temporal Aspect of the Disambiguation Task

We live in a dynamic and rapidly changing world: some companies expand their offices all around the globe, while others collapse; people become celebrities overnight and are forgotten only several years afterwards. Similarly, a whole range of mainstream technological concepts of today's world have only been known since the last few decades. These observations have a big impact on the dynamics of a language system, since the relation between language expressions and meanings follows the changes in the world. To some extent this is reflected in new expressions and new meanings but most strongly this is reflected in the distributional usage of expressions and their dominant meaning.

For instance, the dominant meaning of the terms *mobile*, *cell*, and *phone* is the same for the contemporary, especially young, generations: mobile phone. On the other hand, older generations also remember different dominant concepts from the 80s and 90s: *mobile* being typically a decoration hanging from the ceiling, *cell* usually being a unit in a prison or body tissue, while *phone* referring to the static devices found at home or on the streets. The dominant meanings of the 80s and 90s have been replaced by new dominant meanings, whereas the younger generation may have lost certain meanings such as the decoration. Similarly, football fans remember two different superstar *Ronaldo* players which have been dominant one after the other: the Brazillian striker and the Portuguese Ballon d'Or award winner.

What is shown by these examples is that not only new meanings appear and old meanings become obsolete but that, more strongly, the usage distribution of competing meanings changes over time. As the mobile phone gains popularity and the mobile decoration gets replaced by others, people refer to the mobile phone more often than the traditional mobile decoration. Hence, in a later point of time, the most commonly used meaning for *mobile* changes, even though both meanings are still possible. Similarly for the *Ronaldo* case: in 2016 one can still refer to both players, but the dominant meaning is now the Portuguese player.

We also observe a relation between the variety of lexical expressions used to refer to a meaning, and its dominance of usage. As the mobile phone gained popularity, its set of associated expressions expanded from only *mobile phone* to also: *mobile*, *phone*, *cell phone*, and *cell*. On the other hand, when referring to a prison cell without a specific context, one should nowadays explicitly use the full expression *prison cell* instead of just *cell*.

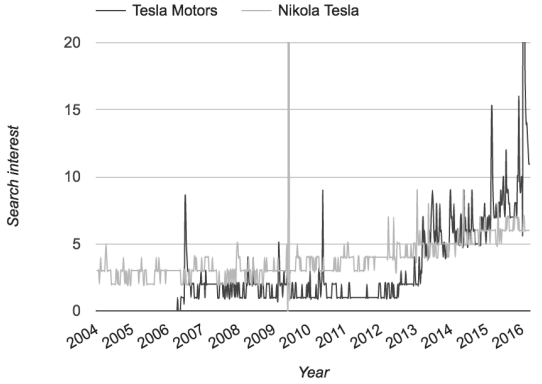
To measure the usage distribution of competing meanings, we could use online resources that track these distributions over time, such as Google Trends[1] and Wikipedia Views.[2] We present the usage distribution for instances denoted by *Tesla* in Figure 1a, and for concepts expressed with the expression *cloud* in Figure 1b. These plots demonstrate the ways in which the distribution of usage changes both for instances and concepts as a function of the temporal dimension. As we discussed in Section 1, the notion

[1] https://www.google.com/trends/

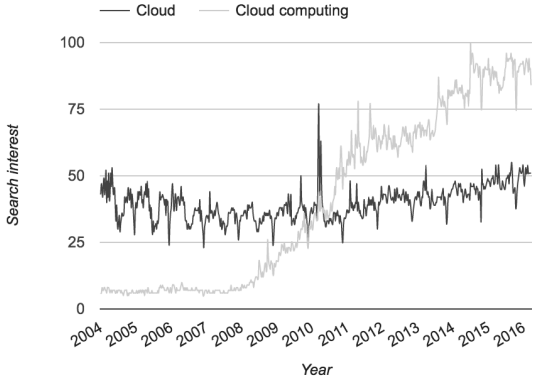
[2] http://stats.grok.se/

Figure 1: Usage distribution for ambiguous concepts and instances based on Google Trends data.

(a) Usage of two dominant meanings for *Tesla*. The black line depicts the usage of *Tesla Motors Company*, while the gray line represents *Nikola Tesla*.



(b) Usage of two dominant meanings for *cloud*. The black line depicts the usage of the clouds in the sky, natural objects placed in the atmosphere. The gray line stands for the modern meaning of cloud as an Internet-based type of computing.



of time and its role in this mapping between expressions and meanings has not been taken into account in the creation of existing disambiguation datasets. This observation points to a serious weakness in the representativeness of existing datasets for the full complexity of the disambiguation task. Consequently, systems are not encouraged to focus on the temporal aspect of the task but in reality the same language system is still used for many different situations within a changing world. While this works for humans, this is not yet solved for machines.

3 Related Work

The three problems enumerated in Section 1 have been addressed to some extent in past work.

Several approaches have attempted to resolve pairs of disambiguation tasks jointly. Examples include: combined Entity Linking (EL) and Word Sense Disambiguation (WSD) (Hulpuş et al., 2015; Moro et al., 2014), combined event and entity coreference (EvC and EnC) (Lee et al., 2012) and resolving WSD and Semantic Role Labeling (SRL) together (Che and Liu, 2010). Although some task combinations are well-supported by multi-task datasets, such as CoNLL 2011 and 2012 for joint coreference (Pradhan et al., 2011; Pradhan et al., 2012), and Moro and Navigli (2015) for WSD and EL, still many multi-task systems have to be evaluated on separate datasets. Notable efforts to create multi-task annotated corpora are the AMR Bank (Banarescu et al., 2013) and the MEANTIME corpus (Minard et al., 2016a).

Properties of existing datasets have been examined for individual tasks. For WSD, the correct sense of a lemma is shown to often coincide with the most frequent sense (Preiss, 2006) or the predominant sense (McCarthy et al., 2004). In the case of McCarthy et al. (2004), the predominant sense is deliberately adapted with respect to the topic of the text. Our work differs from McCarthy et al. (2004) because they do not consider the temporal dimension. As a response to sense-skewed datasets, Vossen et al. (2013) created a balanced sense corpus in the DutchSemCor project in which each sense gets an equal number of examples. Similarly, Van Erp et al. (2016) conclude that EL datasets contain very little referential ambiguity. Evaluation is focused on well-known entities, i.e. entities with high PageRank (Page et al., 1999) values. Additionally, the authors observe a considerable overlap of entities across datasets, even for pairs of datasets that represent entirely different topics. Cybulska and Vossen (2014) and Guha et al. (2015) both stress the low ambiguity in the current datasets for the tasks of EvC and EnC, respectively. Motivated by these findings, Guha et al. (2015) created a new dataset (QuizBowl), while Cybulska and Vossen (2014) extended the existing dataset ECB to ECB+, both efforts resulting in notably greater ambiguity and temporal diversity. As far as we are aware, no existing disambiguation dataset has included the temporal dependency of ambiguity, variance, or dominance.

The problem of overfitting to a limited set of test data has been of central interest to the body of

work focusing on domain adaptation (Daume III, 2007; Carpuat et al., 2013; Jiang and Zhai, 2007). By evaluating on a different domain than the training one, these efforts have provided valuable insights into system performance. However, to our knowledge, this research has also not addressed the temporal aspect of the task.

We therefore propose to take this a step further and examine system performance with respect to a set of metrics, applicable over disambiguation tasks, thus setting the stage for creation of metric-aware datasets. We expect that these metrics show reduced complexity within well-defined temporal and topical boundaries and increased complexity across these boundaries. More extensive datasets than existing single- and multi-task datasets, driven by metrics on ambiguity, variance, dominance and time, would challenge semantic overfitting.

4 Semiotic Generation and Context Model

We want to model the relation between expressions and meanings in the world within a generation that shares the same language system, as well as the fluctuation in usage of expressions and meanings over time within this generation. We therefore assume that for each language community at a specific time, there exist a set of meanings M in the world and a set of lexical expressions L in a language. The relation between these sets is many-to-many: each lexical expression L_i can refer to multiple meanings $M_1, M_2, ...$ (ambiguity) and each meaning M_j can be verbalized through multiple lexical expressions $L_1, L_2, ...$ (variance). As we discuss in Section 2, the sets of M, L, their relations, and especially the distributions of these relations, are dynamic, i.e. they can change over time. We denominate this model "Semiotic Generation and Context Model", because it captures the distribution changes in the semiotic relation between meanings and lexical expressions, given the context of the changes in the world and within the language system of a generation.

In practice, we study available proxies of the world at a moment in time and of the language of a generation which capture this relation at a given time snapshot: lexical resources are considered as a proxy of the language system of a generation and the dataset is considered as a proxy for the world at a particular moment in time creating a specific context. We analyze the time-anchored interaction between M and L in the datasets proxy and measure this against their interaction in the resources proxy to provide insight on how representative the datasets are for the task. Note that the proxies of datasets and resources cover only a subset of the language used within a generation, and (consequently) only a subset of all possible meanings. While not ideal, this is the best we have because there is no way to capture all language used within a generation nor possibly list every possible meaning, especially considering that we can always create new meanings, e.g. by inventing some non-real world ones.

5 Methodology

Based on the Semiotic Generation and Context Model, we now define and formalize a number of metrics that qualify datasets for disambiguation tasks. In this Section, we describe these metrics and explain the tasks we focus on. Furthermore, we enumerate the design choices that guide our pick of datasets and we elaborate on the datasets we analyze.

5.1 Metrics

Mean Observed Ambiguity (MOA)
We define observed ambiguity of an expression as the cardinality of the set of meanings it refers to within a dataset (O_{L_i}). For example, the expression *horse* has 4 meanings in WordNet but only the chess meaning occurs in the dataset, resulting in an observed ambiguity of 1. The Mean Observed Ambiguity (MOA) of a dataset is then the average of the individual observed ambiguity values.

Mean Observed Variance (MOV)
We define observed variance of a meaning as the cardinality of the set of lexical expressions that express it within a dataset (O_{M_j}). The chess meaning of *horse* also has *knight* as a synonym but only *horse* occurs in the dataset, hence an observed variation of 1. The Mean Observed Variance (MOV) of a dataset is then the average of the individual observed variance values.

Mean Observed Dominance of Ambiguity (MODA)
We define dominance of ambiguity as a frequency distribution of the dominant meaning of a lexical expression. For example, *horse* occurs 100 times in the data and in 80 cases it has the chess meaning: the dominance score is 0.8. The Mean Observed Dominance of Ambiguity (MODA) of a dataset is the average dominance of all observed expressions.

Mean Observed Dominance of Variance (MODV)
We define the notion of dominance of variance, as a frequency distribution of the dominant lexical expression referring to a meaning. If *horse* is used 60 times and *knight* 40 times for the same meaning then the observed dominance of variance is 0.6. The Mean Observed Dominance of Variance (MODV) of a dataset is then the average dominance computed over all observed meanings.

Entropy of the Meanings (Normalized) of a Lexical Expression (EMNLE)
We define an alternative notion of dominance, based on entropy, in order to consider the distribution of the less dominant classes in a dataset. We introduce $p(M_j|L_i)$: a conditional probability of a meaning M_j based on the occurrence of a lexical expression L_i. We compute this probability using the formula $p(M_j|L_i) = \frac{p(M_j,L_i)}{p(L_i)}$, a ratio between the number of common occurrences of M_j and L_i, and on the other hand, occurrences of L_i alone. We combine the individual conditional probabilities for L_i in a single information theory metric of entropy, $H(O_{L_i})$:

$$H(O_{L_i}) = \frac{-\sum_{j=1}^{n} p(M_j|L_i) log_2 p(M_j|L_i)}{log_2(n)} \tag{1}$$

For example, given 100 occurrences of the lexical expression *horse*, where 80 occurrences refer to the the chess meaning and 20 to the animal meaning, the entropy of the expression *horse* would be 0.72. To compute a single entropy (EMNLE) value over all lexical expressions in a dataset, we average over the individual entropy values:

$$EMNLE(O_L, R_L) = \frac{1}{n}\sum_{i=1}^{n} H(O_{L_i}, R_{L_i}) \tag{2}$$

Entropy of the Lexical Expressions (Normalized) of a Meaning (ELENM)
We introduce $p(L_i|M_j)$: a conditional probability of a lexical expression L_i based on the occurrence of a meaning M_j. We compute this probability using the formula $p(L_i|M_j) = \frac{p(L_i,M_j)}{p(M_j)}$, a ratio between the number of common occurrences of M_j and L_i, and on the other hand, occurrences of M_j alone. We combine the individual conditional probabilities for M_j in a single information theory metric of entropy, $H(O_{M_j})$:

$$H(O_{M_j}) = \frac{-\sum_{i=1}^{n} p(L_i|M_j) log_2 p(L_i|M_j)}{log_2(n)} \tag{3}$$

Suppose the meaning of horse as a chess piece is expressed 60 times by the lexical expression *horse* and 40 times by *knight*, then the entropy of the chess piece meaning of *horse* is 0.97. To compute a single entropy (ELENM) value over all meanings in a dataset, we average over the individual entropy values:

$$ELENM(O_M, R_M) = \frac{1}{n}\sum_{j=1}^{n} H(O_{M_j}, R_{M_j}) \tag{4}$$

Relation between Observed and Resource Ambiguity (RORA)
We define resource ambiguity of a lexical expression as the cardinality of the set of meanings that it can refer to according to a lexical resource (R_{L_i}). Then we define the ratio between observed and resource ambiguity for a lexical expression as:

$$ratio_{amb}(O_{L_i}, R_{L_i}) = \frac{|\{M_j : M_j \in O_{L_i}\}|}{|\{M_j : M_j \in R_{L_i}\}|} \tag{5}$$

In the case that only 1 out of 4 resource meanings is observed in the dataset, for example only the chess meaning of *horse*, this would lead to a $ratio_{amb}$ value of 0.25. To compute the RORA value of a dataset, we average over the individual ratios:

$$RORA(O_L, R_L) = \frac{1}{n}\sum_{i=1}^{n} ratio_{amb}(O_{L_i}, R_{L_i}) \tag{6}$$

Relation between Observed and Resource Variance (RORV)

We define resource variance of a meaning as the cardinality of the set of lexical expressions which can verbalize it (R_{M_j}). Then we define the ratio between observed and resource variance for a given meaning:

$$ratio_{var}(O_{M_j}, R_{M_j}) = \frac{|\{L_i : L_i \in O_{M_j}\}|}{|\{L_i : L_i \in R_{M_j}\}|} \tag{7}$$

Suppose that the expressions *horse* and *knight* can refer to the meaning of chess piece according to a resource, but only the expression *horse* refers to it in a particular dataset, this would lead to a $ratio_{var}$ value of 0.5. To compute the RORV value of a dataset, we average over the individual ratios:

$$RORV(O_M, R_M) = \frac{1}{n}\sum_{i=1}^{n} ratio_{var}(O_{M_j}, R_{M_j}) \tag{8}$$

Average Time-anchored Rank (ATR)

Since the relevance of meanings is not constant over time, we define the popularity of a meaning in a point of time, $popularity_{M_j}(t)$. A lexical expression can potentially denote multiple meanings, each characterized with a certain degree of time-anchored popularity. Likewise, we order the list of candidate meanings for a given lexical expression based on their popularity at the moment of publishing of the dataset document. For example, if the dataset covers news about a chess tournament, we will see a temporal peak for the chess meaning of *horse* relative to the other meanings. The popularity rank of each meaning, including the correct gold standard meaning, is its position in this ordered list. By averaging over the ranks of all golden candidates we can compute the Average Time-anchored Rank of the golden candidates in a dataset, which gives an indication about the relation between the relative temporal popularity of a meaning and the probability that it is the correct interpretation of an expression, varying from stable to extremely dynamic relations. An ATR rank of a dataset close to 1 indicates a strong bias towards the popular meanings at the time of creation of the dataset.

Average Time-anchored Relative Frequency of Usage (ATRFU)

The potential bias of meaning dominance with respect to its temporal popularity can alternatively be assessed through its frequency of usage at a point of time. We denote the usage of a meaning with U_{M_j}. For a given lexical expression, we compute the relative temporal frequency of usage (FU) of the golden meaning relative to the frequency of usage of all candidate meanings:

$$FU_{M_j}(t) = \frac{U_{M_j}(t)}{\sum_{i=1}^{n} U_{M_i}(t)} \tag{9}$$

The average relative frequency of usage at a given time point (ATRFU) is an average of the frequency values of all gold standard meanings in a dataset. We introduce this metric in order to gain insights into the popularity difference between the competitive meanings at a given time period. This metric would allow us, for instance, to detect that in July 2014 *the United States men's national soccer team* was much more popular than *the women's national soccer team*, while *Tesla Motors* was only slightly more popular than *Nikola Tesla* in May 2015.

Dataset Time Range (DTR)

We define DTR as a time interval between the earliest and the latest published document of a dataset:

$$DTR = [min(date_{doc}), max(date_{doc})] \tag{10}$$

where $date_{doc}$ is the publishing date of a document. For instance, the DTR of the MEANTIME (Minard et al., 2016b) dataset is $[2004, 2011]$.

5.2 Tasks

We demonstrate the applicability of the metrics defined in Section 5.1 on a selection of disambiguation tasks. We cover both concept-oriented tasks (WSD and SRL), as well as instance-based tasks (EL, EnC, and EvC).[3] In Table 1, we specify the model components per disambiguation task, enabling the metrics to be computed. The metrics concerning lexical resources (WordNet (Fellbaum, 1998) for WSD, and PropBank (Kingsbury and Palmer, 2002) for SRL) are only computed for the concept-oriented tasks. Whereas lexical resources, such as WordNet and PropBank, can be seen as reasonable proxies for most of the expressions and concepts known to a generation, it is more difficult to consider databases of instances, such as DBpedia,[4] to approximate all the possible instances that expressions, e.g. *Ronaldo*, can refer to. This is especially the case for events, e.g. the goals *Ronaldo* scored, or the *Ronaldo* t-shirts being sold in a fan shop. There is hardly any registry of real world events independent of the mentions of events in text. Likewise, we only find a few *Ronaldo* entities in DBpedia. Despite its impressive size, DBpedia only covers a very small subset of all instances in the world.

Task	Lexical expression	Meaning	Resource
WSD	lemma	sense	WordNet
SRL	predicate mention	predicate	PropBank
EL	entity mention	entity	DBpedia
EnC	entity mention	entity	DBpedia
EvC	event mention	event	/

Table 1: Task specification of model components.

5.3 Datasets

The choice of datasets conforms to the following rationale. We consider test datasets with running text in English,[5] because we assume that they are the most natural instantiations of the interaction between lexical expressions and meanings and tend to report on the changes in the world. Moreover, such datasets lend themselves better for joint tasks. Finally, we favor publicly available datasets which are commonly used in recent research.

The chosen datasets per disambiguation task are as follows.

WSD The following datasets were taken into consideration: Senseval–2 (**SE2 AW**): All-Words task (Palmer et al., 2001) ; Senseval-3 (**SE3 task 1**): Task 1: The English all-words task (Snyder and Palmer, 2004) ; SemEval-2007 (**SE7 task 17**): Task-17: English Lexical Sample, SRL and All Words (Pradhan et al., 2007) ; SemEval–2010 (**SE10 task 17**): Task 17: All-Words Word Sense Disambiguation on a Specific Domain (Agirre et al., 2010); SemEval–2013 (**SE13 task 12**): Task 12: Multilingual Word Sense Disambiguation (Navigli et al., 2013). The number of test items per competition ranges from roughly 500 to 2500 instances. All most frequent sense baselines are around 65%, except for **SE10 task 17**, in which the focus was on domain-specific WSD, resulting in a most frequent sense baseline of 55%.

SRL For Semantic Role Labelling, we selected the CoNLL-2004 Shared Task: Semantic Role Labeling (**CoNLL04**) (Carreras and Màrquez, 2004). In total, 9,598 arguments were annotated for 855 different verbs.

EL We consider the following datasets: AIDA-YAGO2 (**AIDA test B**) (Hoffart et al., 2011), **WES2015** (Waitelonis et al., 2015), and **MEANTIME** (Minard et al., 2016b). We analyze the commonly used test B collection from the AIDA-YAGO2 dataset, which contains 5,616 entity expressions in 231 documents. WES2015 contains 13,651 expressions in 331 documents about science, while the MEANTIME corpus consists of 120 documents regarding four topics, with 2,750 entity mentions in total.

EnC Guha et al. (2015) created a dataset, QuizBow, for nominal coreference, containing 9,471 mentions in 400 documents. The data annotated comes from a game called quiz bowl.[6]

[3] Note that in the case of SRL we focus on the expression-to-meaning mapping of predicates and do not analyze roles.

[4] `http://dbpedia.org`

[5] Our analysis in this paper is performed on 13 English datasets. The metrics we define in Section 5.1 can easily be applied to many other languages. Namely, the resource-dependent metrics (RORA and RORV) can be applied to the wide range of languages in which DBpedia/WordNet/PropBank are available (for an illustration, DBpedia is currently available in 125 languages). Furthermore, all other metrics rely solely on the annotated textual content within a corpus, which makes them applicable for any language.

[6] `https://en.wikipedia.org/wiki/Quiz_bowl`

EvC we consider three event coreference corpora: EventCorefBank (**ECB**) (Lee et al., 2012), **ECB+** (Cybulska and Vossen, 2014), and EventNuggets (**TAC KBP '15**) (Mitamura et al., 2015). ECB contains 480 documents spread over 43 topics, while its extension ECB+ contains an additional 502 documents spread over the same set of topics. The training corpus of TAC KBP '15 contains 7,478 event coreference chains (hoppers).[7]

6 Analysis

In this Section, we study to what extent datasets cover the complexity of the disambiguation task.[8]

Task	Dataset	MOA	MOV	MODA	MODV	EMNLE	ELENM
WSD	SE2 AW	1.20	1.06	0.94	0.98	0.13	0.05
	SE3 task 1	1.21	1.05	0.94	0.98	0.13	0.04
	SE7 task 17	1.14	1.04	0.95	0.98	0.10	0.03
	SE10 task 17	1.25	1.06	0.93	0.98	0.13	0.05
	SE13 task 12	1.10	1.06	0.97	0.98	0.14	0.05
SRL	CoNLL04	1.20	1.00	0.96	1.00	0.09	0.00
EL	AIDA test B	1.09	1.35	0.98	0.91	0.05	0.22
	WES2015	1.06	1.33	0.97	0.88	0.05	0.21
	MEANTIME	1.19	4.63	0.98	0.64	0.04	0.55
EnC	QuizBowl	1.59	1.80	0.92	0.74	0.13	0.46
EvC	ECB	1.61	3.87	0.89	0.61	0.19	0.65
	ECB+	2.09	3.40	0.85	0.66	0.27	0.57
	TAC KBP '15	4.97	1.22	0.69	0.94	0.47	0.12

Table 2: Observed ambiguity, variance and dominance.

According to Table 2, high complexity in both directions, i.e. high ambiguity and variance, is rare, though the extent of this complexity varies per task. The datasets evaluating WSD, SRL, and EL almost have a 1-to-1 mapping between lexical expressions and meanings, while coreference datasets have higher ambiguity and variance. This can be due to the following reasons: 1. Some of the coreference datasets deliberately focus on increasing ambiguity. 2. An inherent property of coreference seems to be high variance. Similarly, our dominance metrics (MODA/MODV and EMNLE/ELENM) demonstrate a strong bias in our datasets: typically, for any of the datasets, approximately 90% of the occurrences belong to the dominant class on average.

Task	Dataset	ATR	ATRFU
EL	WES2015	1.92	0.53
EL	MEANTIME	1.51	0.51

Table 3: ATR and ATRFU values of the datasets.

Task	Dataset	RORA	RORV
WSD	SE2 AW	0.26	0.38
	SE3 task 1	0.23	0.37
	SE7 task 17	0.20	0.36
	SE10 task 17	0.25	0.40
	SE13 task 12	0.26	0.40
SRL	CoNLL04	0.63	1.00

Table 4: RORA and RORV values of the datasets.

Concerning the concept-oriented tasks, Table 4 shows a notable difference in the complexity of the interaction between the proxies of datasets and resources.[9] Between 74 and 80% of the resource ambiguity per expression is not represented in the datasets, whereas this is the case for 60-64% of the resource

[7] We were unable to obtain the test data for the TAC KBP '15 dataset, hence our analysis is performed on the training data.

[8] The metrics and the analyses of the datasets can be found at `https://github.com/cltl/SemanticOverfitting`.

[9] While computing RORA and RORV, we ignore cases with resource ambiguity and variance of 1.

variance per concept. This is an indication of strong semantic overfitting of the data to a small selection that is not representative for the full potential of expressions and meanings. Furthermore, we observe that this representativeness is relatively constant across concept datasets, which in part can be explained by the fact that the WSD and SRL datasets mainly stem from the same time period (Figure 2), and even from the same corpus (Hovy and Søgaard, 2015). One could argue that the data is correctly representing the natural complexity of a specific time period and genre but it does not challenge systems to be able to shift from one situation to another. We also note a temporal discrepancy between the concept- and instance-based datasets, with the instance-based systems being evaluated on more recent data.

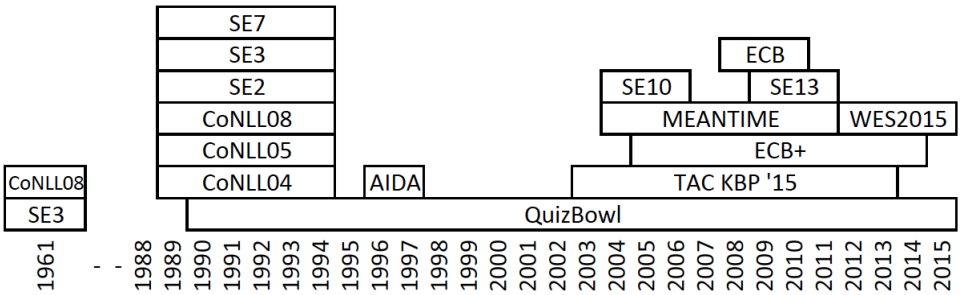


Figure 2: DTR values of the datasets

To understand further the time-bound interaction in our datasets, we study them together with time-bound resources. While our lexical resources and instance knowledge sources contain very little temporal information, we rely on query monitoring websites (Wikiviews and GoogleTrends) to get an indication of the usage of a meaning over time. In Table 3, we show the temporal popularity of entities among their candidates in our datasets according to our web sources.[10] We note a correspondence between the dominance of entities in datasets and their frequency of usage at that time, which exposes a bias of existing datasets towards the most popular entities at the time of their creation.

Our analysis reveals that the existing disambiguation datasets show a notable bias with respect to the aspects of ambiguity, variance, dominance, and time, thus exposing a strong semantic overfitting to a specific part of the world, while largely ignoring long tail phenomena. Typically this is the part of the world that is best known within the context of a generation at a moment of time. This implies that our datasets have a strong bias towards meanings that are popular at that particular moment in time and do not represent the temporal relativity of this bias. Although our metrics provide us with a valuable set of insights into the evaluation datasets, complementary statistical measures should be introduced in the future to capture individual distinctions blurred by averaging over a dataset. These could measure the distribution of ambiguity and variance, their relation to dataset size, and outliers.

7 Proposal for improving evaluation

The direct contribution of our work lies in metric-based evaluation of datasets and resources for systems, which helps interpreting their ability to cope with alterations of ambiguity, variance, dominance, and time. Provided that a collection of multi-task annotated data is available at a central place, our metrics could be applied to output a dataset following certain criteria, e.g. a test set annotated with WSD and EL, whose ambiguity and variance are both between 1.2 and 1.4, and whose documents have been created in the 90s. The practical obstacle is the availability of input data, which can be addressed by the following (semi)automatic expansion method: 1. Collect annotated data and split the data according to time periods. 2. Collect annotated expressions from the data with their dominant meanings. 3. Retrieve

[10]Due to the non-availability of information for the other tasks, we only analyze the temporal dominance for the EL task, even though the set of represented entities in DBpedia is not complete (as discussed in Section 5.3). In our analysis, we only consider ambiguous expressions that can denote more than one entity candidate. The candidates were obtained from the Wikipedia disambiguation pages. From Wikiviews, the month of the document creation time was used for the dominance information.

new documents using unsupervised techniques in which these expressions occur with evidence for usage in other meanings than the dominant one in the existing datasets. Evidence can come from meta data, unsupervised clustering, and temporal and topical distance from annotated data. 4. Fix alternative meanings for all tokens in the new texts (one meaning-per-document), if necessary applying additional disambiguation tools. Add this data as silver data to the collection. 5. If necessary, re-annotate silver data manually or add annotations for other tasks.[11] 6. Spread documents over different time periods for both annotated gold data and silver data to obtain sufficient variation in time-bound contexts. Provided that this acquisition procedure is successful, selecting a dataset would require almost no effort, which enables creation of many, well-motivated datasets. Consequently, the dynamic nature of this process would challenge semantic overfitting.

In case the proposed data acquisition procedure proves too hard or too laborious to realize, we propose an alternative, namely an event-based Question Answering (QA) task, extensively elaborated on in Postma et al. (2016), whose dataset would contain documents gathered in a smart automatic way. The data acquisition procedure for this dataset is driven by multiple confusion factors: ambiguity, dominance, variance, time, location, and topic. This data would reflect a high degree of ambiguity and variance and would capture a wide range of small real-world phenomena. In order to perform on this task with a good accuracy, the systems will be required to exhibit a deeper semantic understanding of the linguistic tail of the disambiguation tasks we analyze in this paper. However, the only task that will explicitly be evaluated is the QA task itself, which means that the annotation task would be largely reduced to the components necessary for the questions and answers.

The resulting time-aware evaluation datasets, originating from both the annotation-based and QA-based approaches, allow the community to test understanding of language originating from different generations and communities, and a community's language usage in relation to different world contexts. It would also assess to what extent a disambiguation system can adapt to language use from another time slice than the one trained on, with potentially new meanings and expressions, and certainly a different distribution of the expression-meaning relation. We believe this challenges semantic overfitting to one single part and time of the world, and will inspire systems to be more robust towards aspects of ambiguity, variance, and dominance, as well as their temporal dependency.

8 Conclusion

We qualified and quantified the relation between expressions and meanings in the world for a generation sharing a language system, as well as the fluctuation in usage of expressions and meanings over time. We proposed the Semiotic Generation and Context Model, which captures the distribution changes in the semiotic relation given the context of the changing world. We apply it to address three key problems concerning semantic overfitting of datasets. We conceptualize and formalize generic metrics which evaluate aspects of datasets and provide evidence for their applicability on popular datasets with running text from five disambiguation tasks. We observe that existing disambiguation datasets show a notable bias with respect to aspects of ambiguity, variance, dominance, and time, thus exposing a strong semantic overfitting to a very limited, and within that, popular part of the world. Finally, we propose a time-based, metric-aware approach to create datasets in a systematic and semi-automated way as well as an event-based QA task. Both approaches will result in datasets that would challenge semantic overfitting of disambiguation systems.

References

Eneko Agirre, Oier López de Lacalle, Christiane Fellbaum, Shu-Kai Hsieh, Maurizio Tesconi, Monica Monachini, Piek Vossen, and Roxanne Segers. 2010. SemEval-2010 Task 17: All-Words Word Sense Disambiguation on a Specific Domain. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval-2010)*, pages 75–80, Uppsala, Sweden, July. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp

[11] Excessive labor could be avoided by prioritizing relevant expressions, e.g. according to the metrics.

Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop & Interoperability with Discourse*, pages 178–186.

Marine Carpuat, Hal Daumé III, Katharine Henry, Ann Irvine, Jagadeesh Jagarlamudi, and Rachel Rudinger. 2013. Sensespotting: Never let your parallel data tie you to an old domain. In *Proceedings of the Association for Computational Linguistics (ACL)*. Citeseer.

Xavier Carreras and Lluís Màrquez, 2004. *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, chapter Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling.

Wanxiang Che and Ting Liu. 2010. Jointly modeling WSD and SRL with Markov logic. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 161–169. Association for Computational Linguistics.

Agata Cybulska and Piek Vossen. 2014. Using a sledgehammer to crack a nut? lexical diversity and event coreference resolution. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 4545–4552.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, pages 256–263, Prague, Czech Republic. Association for Computational Linguistics.

Christiane Fellbaum. 1998. Wordnet: an electronic lexical database. *MIT Press, Cambridge MA*, 1:998.

Anupam Guha, Mohit Iyyer, Danny Bouman, Jordan Boyd-Graber, and Jordan Boyd. 2015. Removing the training wheels: A coreference dataset that entertains humans and challenges computers. In *North American Association for Computational Linguistics (NAACL)*.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordin, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust Disambiguation of Named Entities. In *Conference on Empirical Methods in Natural Language Processing*, EMNLP.

Dirk Hovy and Anders Søgaard. 2015. Tagging performance correlates with author age. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 483–488.

Ioana Hulpuş, Narumol Prangnawarat, and Conor Hayes. 2015. Path-based semantic relatedness on linked data and its use to word and entity disambiguation. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 442–457. Springer.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, volume 7, pages 264–271.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to Propbank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012)*, pages 489–500. Association for Computational Linguistics.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL 2004)*, page 279. Association for Computational Linguistics.

Anne-Lyse Minard, Manuela Speranza, Ruben Urizar, Begona Altuna, Marieke van Erp, Anneleen Schoen, and Chantal van Son. 2016a. MEANTIME, the newsreader multilingual event and time corpus. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

Anne-Lyse Minard, Manuela Speranza, Ruben Urizar, Begona Altuna, Marieke van Erp, Anneleen Schoen, and Chantal van Son. 2016b. Meantime, the newsreader multilingual event and time corpus. *Proceedings of LREC2016*.

Teruko Mitamura, Zhengzhong Liu, and Eduard Hovy. 2015. Overview of tac-kbp 2015 event nugget track. In *Text Analysis Conference*.

Andrea Moro and Roberto Navigli. 2015. Semeval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval-2015)*, pages 288–297.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics (TACL)*, 2:231–244.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. SemEval-2013 Task 12: Multilingual Word Sense Disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 222–231, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: bringing order to the web.

Martha Palmer, Christiane Fellbaum, Scott Cotton, Lauren Delfs, and Hoa Trang Dang. 2001. English Tasks: All-Words and Verb Lexical Sample. In *Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems (SENSEVAL-2)*, pages 21–24, Toulouse, France, July. Association for Computational Linguistics.

Marten Postma, Filip Ilievski, Piek Vossen, and Marieke van Erp. 2016. Moving away from semantic overfitting in disambiguation datasets. In *Proceedings of the EMNLP Workshop on Uphill Battles in Language Processing*.

Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. SemEval-2007 Task-17: English Lexical Sample, SRL and All Words. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 87–92, Prague, Czech Republic, June. Association for Computational Linguistics.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. CoNLL-2011 Shared Task: Modeling unrestricted coreference in OntoNotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL 20111)-Shared Task*, pages 1–27. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 Shared Task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on Empirical Methods on Natural Language Processing (EMNLP) and on Natural Language Learning (EMNLP-CoNLL 2012)-Shared Task*, pages 1–40. Association for Computational Linguistics.

Judita Preiss. 2006. A detailed comparison of WSD systems: an analysis of the system answers for the Senseval-2 English all words task. *Natural Language Engineering*, 12(03):209–228.

Benjamin Snyder and Martha Palmer. 2004. The English All-Words Task. In Rada Mihalcea and Phil Edmonds, editors, *SensEval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text.*, pages 41–43, Barcelona, Spain, July. Association for Computational Linguistics.

Marieke van Erp, Pablo Mendes, Heiko Paulheim, Filip Ilievski, Julien Plu, Giuseppe Rizzo, and Jorg Waiterlonis. 2016. Evaluating entity linking: An analysis of current benchmark datasets and a roadmap for doing a better job. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

Piek Vossen, Ruben Izquierdo, and Atilla Görög. 2013. DutchSemCor: in quest of the ideal sense-tagged corpus. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*, pages 710–718. INCOMA Ltd. Shoumen, Bulgaria.

Jörg Waitelonis, Claudia Exeler, and Harald Sack. 2015. Linked Data Enabled Generalized Vector Space Model to Improve Document Retrieval. In *Proceedings of NLP & DBpedia 2015 workshop in conjunction with 14th International Semantic Web Conference (ISWC2015), CEUR Workshop Proceedings*.

Extraction of Keywords of Novelties From Patent Claims

Shoko Suzuki
IBM Research - Tokyo
19-21 Nihonbashi, Hakozaki-cho,
Chuo-ku, Tokyo, Japan
e30126@jp.ibm.com

Hiromichi Takatsuka
IBM Japan
19-21 Nihonbashi, Hakozaki-cho,
Chuo-ku, Tokyo, Japan
e50508@jp.ibm.com

Abstract

There are growing needs for patent analysis using Natural Language Processing (NLP)-based approaches. Although NLP-based approaches can extract various information from patents, there are very few approaches proposed to extract those parts what inventors regard as novel or having an inventive step compared to all existing works ever. To extract such parts is difficult even for human annotators except for well-trained experts. This causes many difficulties in analyzing patents. We propose a novel approach to automatically extract such keywords that relate to novelties or inventive steps from patent claims using the structure of the claims. In addition, we also propose a new framework of evaluating our approach. The experiments show that our approach outperforms the existing keyword extraction methods significantly in many technical fields.

1 Introduction

Recently there are growing needs for analyzing patents. Many companies want to analyze large amount of patents for various purposes like patent retrieval or analyzing technical trends, etc. For searching and analyzing large amount of patents, NLP-based approaches are adequate, and many approaches are developed (for example (Abbas et al., 2014)). Several keyword extraction methods are proposed in the context of patent retrieval or information extraction from patents. Most of them use traditional unsupervised approaches like BM25 (Robertson and Zaragoza, 2009) or supervised approaches like CRF (Lafferty et al., 2001). While BM25 tends to extract keywords that are characteristic to each patent, CRF is applied to extract various kinds of Named Entities such as technologies, effects, and attributes. (for example in (NTCIR, 2010),(Nishiyama et al., 2010)).

However, considering the original purpose of submitting patents, patents must contain rich information not limited to the above examples. Especially, every patent must contain what the inventors think as novel or having an inventive step compared to the all existing works ever. There is no doubt that extracting such keywords is quite important and applicable to all other patent analysis like patent retrieval or analyzing technical trend, etc. To our knowledge there are very few works that explicitly try to extract those keywords that relate to the novelties or the inventive steps of each patent (we will call these keywords as keywords of novelties in this paper).

In general patent retrieval task, various kind of weights are calculated for keywords/keyphrases. But these weights don't necessarily reflect the degree of the novelties. Several approaches seem to extract novel parts of each patent implicitly, but they don't go so far as to extract keywords of novelties. Besides, in patent retrieval, similar patents often have different surface expressions especially on the novel parts, resulting in the situation that the performance in patent retrieval task is not necessarily related to the performance of extracting the novel part. This means that the extracting keywords of novelties cannot be evaluated directly in patent retrieval task.

In this paper, we propose a new approach of extracting keywords of novelties from patent claims. Among various parts in patent applications, patent claims are the most crucial parts that define the scope

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1192–1200, Osaka, Japan, December 11-17 2016.

of protection and contain all of the important parts of the invention. Since patent claims are written in a specific manner, usual NLP approaches may fail to extract useful information. Therefore, we assume several underlying structural rules in patent claims and utilize these assumptions in extracting keywords.

In addition, we also propose a new framework of evaluating our approach.

In section 2, we first explain patent claim structures and related work to extract those structures. Then we briefly introduce some approaches of keyword extraction from patents. In section 3 we propose a novel approach to extract keywords of novelties. And in section 4, we introduce a new framework of evaluating the performance of our approach. Section 5 shows the experimental setting and the evaluation results of our proposed approach compared to other keyword extraction approaches. Section 6 describes some concluding remarks and future application using our approach.

2 Related Work

Several attempts are made to implicitly extract keywords of novelties for each patent, mainly in the context of patent retrieval. Patent retrieval is a task to extract similar patents when a target patent is given.(Patent retrieval includes those concepts such as invalidity search and prior art search.) Among naive approaches that use common techniques in information retrieval (IR), some utilize the features that are specific to patent claims, which result in better performance than using usual IR techniques.

Although each patent claim is a plain text without apparent sections, it is built under some rules. There is no doubt that utilizing these rules achieves better performance in extracting information from patent claims.

2.1 Patent Claim Structures

Patent claims have specific structures, and it is difficult to understand the contents except for experts. Each patent has one or more claims. In case a patent has multiple claims, these claims are divided into two kinds of claims; independent claims and dependent claims. Each claim can be decomposed to several structural elements. There are also several types for claim structure like Jepson type or Markush type. For example, Jepson type has two parts in a claim, preamble and body. In addition, each claim must declare a subject matter, which corresponds to a noun phrase in a specific position of the claim.

To extract those structures automatically, several approaches exist. For example, Sheremetyeva et al. have decomposed each claim to structural elements using POS-tags (Sheremetyeva et al., 1996), Parapatics et al. have categorized claims into several types using cue phrases (Parapatics and Dittenbach, 2009), and Shinmori et al. have proposed to apply Rhetorical Structure Theory (RST) for parsing structural elements (Shinmori et al., 2003).

2.2 Keyword Extraction from Patent Documents

Keyword extraction is an important task for every kind of documents, and many approaches are suggested. The most known approaches are tf-idf, BM25 and TextRank. These approaches are also applicable to patent claims, but these are not designed to extract keywords of novelties, and tend to achieve low performance in various patent analysis.

For example, TextRank approach is applied (Verma and Varma, 2011) to the whole patent application or to a specific part of a patent like patent claim, abstract, or detailed description in order to extract important keywords for invalidity search. But they did not intend to extract such keywords of novelties. Besides, they did not utilize claim structure.

Word age of each term is introduced to measure the degree of the novelties for each patent (Hasan et al., 2009) in order to score novelty of each patent. Word age represents how long the term exists in a corpus (time span from the time when it first appeared to the time when the target patent submitted), and it represents novelty to some extent. But it is obvious that word age highly depends on the corpus, technical field or surface expressions, which is not related to what inventors think as novel part. Moreover, inventors don't necessarily use new keywords to describe the novel part of of their inventions.

Other than those approaches that directly extract important keywords from patent claims, there are some approaches that patent-specific keyword extraction are used.

Shinmori et al.(2003) proposed an approach to extract such keywords/keyphrases that are representative in each structural element in patent claims by using some morphological patterns after decomposion of claims to structural elements(Shinmori et al., 2003).

Takaki et al.(2004) first decomposed each claim into structural elements, and calculated the importance of each element by a measure of how each term in the element is locally distributed in those elements (Takaki et al., 2004). Their approach seems to grasp one side of patent claim structure, and their approach is effective in patent retrieval task to some extent.

Lin et al.(2010) tried more systematic approach utilizing patent claim structure for patent retrieval (Lin et al., 2010). They built a claim tree by extracting the relations between independent/dependent claims and the structural elements for all patents, then search the similarities of the claim trees. They extracted relevant keywords of each structural element from patent specifications (not from claims) using term frequency in the specifications and calculating mutual information of terms selected from claim and specification.

These patent-specific methods extract keywords from various aspects, but none of them directly tried to extract keywords of novelties for each patent.

There are other approaches to extract keywords from patent claims using claim structures, but the focuses are only limited to discriminations like preambles/bodies (Mase et al., 2005) or the subject matters.

3 Extraction of Keywords Related to Novelty from Patent Claims

Compared to the related work, our approach is built under some assumptions of claim structures. Using these assumptions we propose a new approach to extract keywords of novelties from patent claims.

3.1 Assumptions of Claim Structures

One main reason that a patent is constructed from multiple claims is that an applicant want to protect the scope as broad as possible after registration by a patent office. Patent offices in many countries examine patent claims, and if one idea already exists in prior art or easily imaginable from prior art, they refuse the claim. However, if there is a claim that describes a bit smaller scope of the rejected claim, and the scope is invalidated by any prior art, this claim is granted. Therefore, applicants tend to submit multiple claims in one patent with some hierarchical structures of dependencies. In general, a dependent claim describes smaller scope than that of the ”parent” claim. This means that a dependent claim focuses on an important part of the parent claim to secure this part. In many cases, this important part (that the applicant wants wider scope as possible) describes the novelties or the inventive steps that the inventor wants to claim, or describes the area the patent is applicable.

Besides, each claim usually has multiple structural elements. Since a claim must contain every element that is necessary, part of the elements are necessary to construct the subject matter but not directly related to the novelties or the inventive steps of the patent. This also suggests that elements are related to each other somehow. These relations might represent some process flows or adding functions/features to other elements. Then we can interpret these relations as some kind of hierarchies. Therefore, just like dependent claim structures, we can rebuild structural elements in a claim into some hierarchical structures.

Figure 1 shows an example of hierarchical structures. The element 1-4 is in the claim 1, and the dependent claim 2 and claim 3 depends on the element 2 and 4 respectively.

Now there is another rule in writing a claim; there must be no unnecessary element appeared in a claim. This is because an unnecessary element narrows the scope of the patent, and applicants never want this situation. Suppose that a claim has an additional new element which depends on an key element containing novelties of the patent (this new element might use the output of the key element's process or add some features to the key element), then the claim has only a limited scope compared to the claim without the additional element. This means that a properly written patent claims contain only a few key elements and these key elements should be placed in a lower level of the hierarchical structure.

Based on these observations, we assume two things;

- the keywords of novelties tend to exist in the elements that the dependent claims depend on.

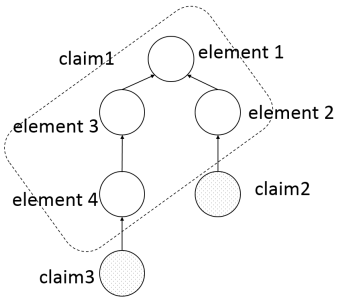


Figure 1: Example of Claim Structure

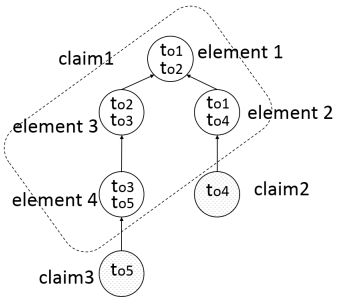


Figure 2: with Overlapping terms

- the keywords of novelties tend to exist in a lower level of the element hierarchical structure.

3.2 Outline of New Approach to Extract Keywords

In this subsection, we describe the outline of our proposed approach to extract keywords of novelties in the first claim using patent claim structures under the assumptions in the previous section. (The extension from the first claim to all independent claims is straightforward.) The notation is listed in Table 1.

Notation	Description
$e(w)$	the first element in $\{e_i\}$ where term w appears
$d(e_i)$	the depth of the element e_i
t_o	overlapping term that connects element-element dependency or element-dependent claim dependency
$ET(e_i)$	map from child element e_i to t_o
$DT(c_i)$	map from dependent claim c_i to t_o
$parent(e_i)$	the parent element of e_i
T_o	the set of all t_o
$ncl(t_o)$	the number of dependent claims $DT^{-1}(t_o)$
$Mod(k)$	map to $\{t_o\}$ that k modifies

Table 1: Notation

Algorithm 1 extracting element dependency structure

$d(e_1) \leftarrow 0$
for $i = 2$ **to** $i = |\{e_i\}|$ **do**
  search $e(w)$ in $\{e_j | j \leq i\}$ for $\forall w \in e_i$
  $d(e_i) \leftarrow -1$
  $d(e_i) \leftarrow \max_{w \in e_i} d(e(w)) + 1$
  if $d(e_i) \neq 0$ **then**
    $t_o \leftarrow \arg\max_{w \in e_i} d(e(w))$
    $parent(e_i) \leftarrow e(t_o)$
    $ET(e_i) \leftarrow t_o$
  end if
end for

Step1:extracting element structure: The first claim of a patent is parsed so that each structural element is decomposed, and dependencies within the elements are extracted. Dependencies between the elements are extracted in the following way. First, the decomposition of each element is done by using cue phrases (Shinmori et al., 2003) or using line breaks. Then, the depth for each element is calculated by following the procedure in Algorithm 1. Note that $\{e_i\}$ is a set of elements sequentially derived from the first claim. Those terms that appear in the subject matter are removed from this analysis. Figure 2 represents the example of dependencies using overlapping terms t_o. The term t_{o3} first appears in e_3, then in e_4. Therefore $parent(e_4) = e_3$, and $ET(e_4) = t_{o3}$. Other types of structure extraction is also applicable.

Step2:extracting claim dependency structure: The claims that depend on the first claim are parsed. The procedure to attach depth for all dependent claims is the same as that of step 1. This means that overlapping term t_o is attached for each dependent claim c_i, i.e. $DT(c_i) \leftarrow t_o$. In figure 2, t_{o5} first appeared in e_4. t_{o5} also appears in the claim 3, then $DT(c_3) = t_{o5}$.

Step3:calculating the score of the representative keywords: For every overlapping term t_o stored in the set T_o, the score s_o is calculated using the depth of element $e(t_o)$ and the number of dependent claims $ncl(t_o)$ attached to t_o. The definition of score s_o is explained in the next subsection 3.3.

Step4:calculating the score of the keywords of novelties: Keywords k modifying each t_o are searched in the first claim. Modification of t_o by k is defined by satisfing at least one of the following conditions;

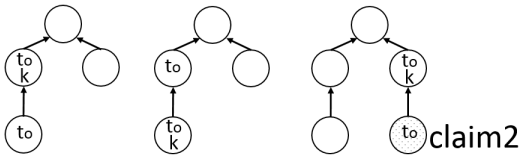


Figure 3: Examples of Modification: k modifying t_o

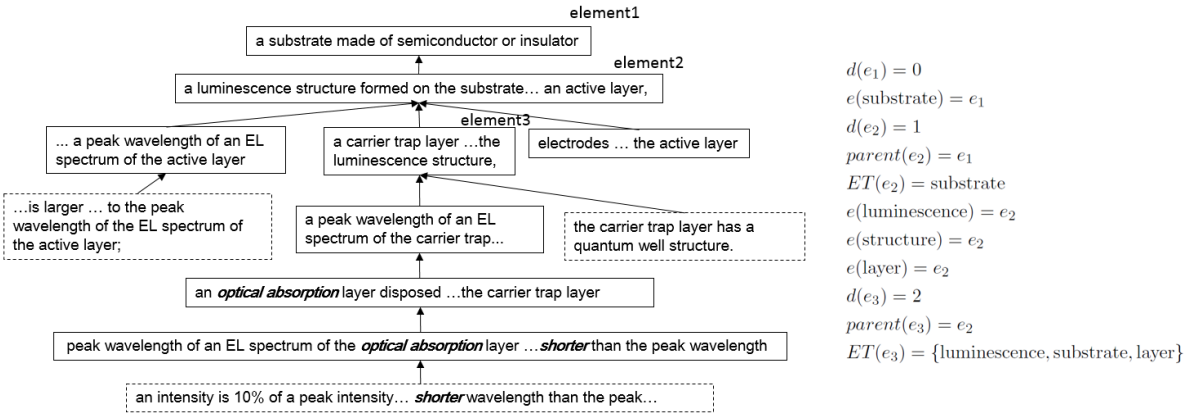


Figure 4: Actual Example of Claim Structure

1. k appears in the element $ET^{-1}(t_o)$.
2. k appears in the element $e(t_o)$.

t_o can be also regarded as k. A keyword k modifying t_o is represented as $t_o \in Mod(k)$ and $k \in Mod^{-1}(t_o)$. Note that the mapping function Mod and Mod^{-1} is many-to-many mapping. Figure 3 shows examples of modifications. In every example in figure 3, k modifies t_o. Then the score $S(k)$ for each $k \in \{k| \cup_{t_o \in T_o} Mod^{-1}(t_o)\}$ is calculated using the definition in the next subsection 3.3.

The actual example of claim structures is shown in figure 4. The boxes enclosed by solid line represent elements in the first claim and those of dashed line represent dependent claims. The bold italic keywords are the keywords of novelties annotated automatically in the framework introduced in the section 4. The right side of figure 4 shows the actual procedure of Algorithm 1 for the first 3 elements.

3.3 Definition of Scores

During the whole process, the locality score for each term w is calculated by

$$loc(w) = \frac{|\{e_i\}|}{|\{e_i | e_i \ni w\}|}, \tag{1}$$

that is, locality represents how much w is localized among all elements.

The score for overlapping terms is defined as

$$s_{o1}(t_o) = d(e(t_o)) * (ncl(t_o) + 1) \tag{2}$$

$$s_{o2}(t_o) = d(e(t_o)) * (log(ncl(t_o) + 1) + 1), \tag{3}$$

and using the score of overlapping terms t_o, the score of k that modifies t_o is defined in the equation

$$S_l(k) = \max_{t_o \in Mod(k)} loc(k) * s_{ol}(t_o) \tag{4}$$

with $l = 1, 2$ and s_{ol} represents eq.(2) or eq.(3), respectively. Note that $S_l(k) = 0$ if $Mod(k) = \emptyset$.

4 Evaluation Method Using Rejected and Granted Patents

In the previous section, we proposed a new approach of how to calculate the scores for each term k. This score $S_l(k)$ represents the degree of novelties of term k in each patent. But the evaluation of this approach is another difficult task. To evaluate the performance of extracting such keywords, we need annotated data. But manual annotation needs domain knowledge of each technical field and expertise in reading patent claims. Therefore, preparing large amount of manual annotations for various technical fields is quite difficult. Since extracting the keywords of novelties is not a conventional task in patent analysis, there is no commonly available shared task set for evaluation. In this section, we propose a general framework to obtain annotated data automatically from the examination result of Patent Offices.

4.1 Process of Patent Examination

Patent Offices in some countries publish all examination processes of each patent. Basically, a submitted patent follows a process like this; **1**. A patent is examined and if an insufficient part exists, the examiner reject the patent. **2**. The applicant may withdraw submission, or modify the claim to overcome the reason of the rejection. **3**. After repeated examination and possible rejection/modification, a patent is decided to be granted or rejected.

One of the major and critical reasons of the rejection is the existence of prior art that invalidates the submitted patent. In other words, if this type of rejection is overcome by modifications of the claims, this modified part must contain the keywords of novelties.

4.2 Framework of Evaluation

Based on the assumption, we build a framework to evaluate the extraction of keywords of novelties for each patent. Each patent document used in this framework must be rejected only by the reason of existence of prior art and then be granted after modifications. For each patent satisfying the above condition, two types of claims are extracted for each patent; the claims before the patent is examined and the claims after the patent is granted. Then those keywords, which appear in the granted first claim but not in the rejected first claim, are extracted. Such keywords are regarded as the positive-labeled set of keywords of novelties. The reason of extracting keywords only from the first claim is, the correspondence between the two types is clear. (For example, the third claim in the rejected patent corresponds to the second claim in the granted patent, which is difficult to guess. But usually, the first claim of the rejected patent is corrected by adding keywords from the proceeding dependent claims or from the description in the application.)

In order to examine the extracted set of keywords are truly positive, we randomly pickup and check some of the granted first claims manually by referencing the arguments in response to the notices of reasons for rejection (i.e. Office Actions). In the arguments, applicants explain how the corrected claims differ from the prior art. In this preliminary analysis, the average f-measure of 26 patents becomes 0.8339. This result ensures that our proposed framework is adequate for preparing positive-labeled set of keywords of novelties at least as an approximation.

The positive-labeled set is compared to those keywords extracted by various approaches for evaluation.

5 Experiments

In this section, we evaluate our approach proposed in section 3 using the framework proposed in section 4.

5.1 Corpus

The Japanese patents submitted during Jun 1 to March 31 in 2005 are used in the evaluation. We selected patents for each technical field corresponding to International Patent Classification (IPC) from A to H. The IPC Section title and the definition list is shown in table 2 (More detailed explanation can be found in (WIPO Guide, 2016)). For calculating document frequency in BM25 (one of the baseline approaches we applied), we use other corpus containing the patent data submitted before Jun 1, 2005. This corpus contains around 3 million patents.

Section	Definition
A	HUMAN NECESSITIES
B	PERFORMING OPERATIONS; TRANSPORTIN
C	CHEMISTRY; METALLURG
D	TEXTILES; PAPER
E	FIXED CONSTRUCTIONS
F	MECHANICAL ENGINEERING; LIGHTING; HEATING; WEAPONS; BLASTING
G	PHYSICS
H	ELECTRICITY

Table 2: IPC Section Title

Approach	Definition
Proposed1	$S_1(k)$ in eq.(4)
Proposed2	$S_2(k)$ in eq.(4)
BM25	BM25 score $BM(w)$
BM25perEle	$S_{base2}(w)$ in eq.(7)
Locality	$loc(w)$ in eq.(1)
Loc*Ele1	$\sum_{e_i \ni w} loc(w) * \tilde{S}_{e1}(e_i)$
Loc*Ele2	$\sum_{e_i \ni w} loc(w) * \tilde{S}_{e2}(e_i)$

Table 3: Definition of Approaches

5.2 Approaches to be Evaluated

As a baseline, we try BM25, the traditional keyword extraction approach. Besides, we also try several naive approaches that are easily conceivable from the related work in section 2. One is using locality as a score defined in eq.(1). This is similar to Takaki(2004)'s approach while they also used element-wise score as well. Here we prepare two types of element-wise score referring to Takaki's approach.

$$\tilde{S}_{e1}(e_i) = \frac{1}{|e_i|} \sum_{w \in e_i} loc(w) \tag{5}$$

$$\tilde{S}_{e2}(e_i) = \frac{1}{log(|e_i|+1)} \sum_{w \in e_i} loc(w) \tag{6}$$

Moreover we prepare one simple extension of BM25:

$$S_{base2}(w) = \max_{e_i \ni w} \tilde{S}_{e2}(e_i) * BM_i(w) \tag{7}$$

where $BM_i(w)$ represents the value of BM25 of w regarding the element e_i as a document.

The table 3 shows the definition of scores of tested approaches. **Proposed1** and **Proposed2** are our new proposed approaches introduced in section 3. **BM25** is the baseline approach. **BM25perEle**, **Loc*Ele1** and **Loc*Ele2** are the approaches easily conceivable from the related work.

5.3 Evaluation Results

We evaluate the performance of keyword extraction using each score by Mean Average Precision (MAP) which is often used to evaluate the performance of information retrieval. Since several types of scores like our proposed approaches or locality tend to have the same value for multiple terms, we calculate all the orders for those tie ranks and averaged the MAP value of each order.

The result is in the table 4. This shows that in every technical field our proposed approaches **Proposed1**, **Proposed2** outperform the baselines using BM25 or those approaches that are easily conceivable from the relate work. Especially the approach **Proposed2** significantly outperforms the baselines in most of the tehcnical fields except for IPC=C (chemistry). One reason for relatively low performance of our proposed approach in the field of chemistry is, that patent claims in the field are often the type of Markush Claim which doesn't fit our current structure parsing method in step1 of subsection 3.2 (The current method fits Jepson Claim better).

6 Conclusion

We propose a new approach to extract keywords of novelties from patent claims. It is a challenging task partly because the problem setting itself is rather unique, and partly because there has been no framework to evaluate the performance directly. We show that the existing keyword extraction techniques don't work well when analyzing patent claims, since patent claims have a specific format that a non-expert finds difficult. Although analyzing patent claims is quite important in many industries, understanding the contents of claims using NLP techniques is still in a developing phase.

Approach	MAP							
	IPC=A	**IPC=B**	**IPC=C**	**IPC=D**	**IPC=E**	**IPC=F**	**IPC=G**	**IPC=H**
Proposed1	0.5947	0.5526	0.4874	0.5338	0.5685	0.5783	0.5825	0.5520
Proposed2	**0.6109**	**0.5741**	**0.4925**	**0.5471**	**0.5880**	**0.6066**	**0.5984**	**0.5779**
BM25	0.3751	0.3329	0.3855	0.3464	0.3373	0.3033	0.3394	0.3333
BM25perEle	0.4106	0.3689	0.4042	0.3709	0.3579	0.3355	0.3825	0.3708
Locality	0.4997	0.4579	0.4475	0.4458	0.4534	0.4390	0.4847	0.4613
Loc*Ele1	0.5155	0.4736	0.4753	0.4739	0.4685	0.4597	0.5082	0.4820
Loc*Ele2	0.5376	0.4925	0.4578	0.4635	0.4795	0.4806	0.5239	0.4943
number of documents	356	501	278	198	370	351	338	501

Table 4: MAP result

While there are works on extracting novelties from trend analysis of news, SNS or other documents, these are mainly aiming to extract only keywords from those documents with brand new contents. But in patent claims, novelties are often represented in general expressions. This is one main reason that we focus on extracting what inventors think as novel for every patent using the structures of patent claims.

In this paper we apply only preliminary approaches, but additional attempts such as using key phrases, using dependencies between terms, or using other fields in patent specifications will surely increase the performance. Moreover, since annotated data is available using our proposed framework, supervised approaches are applicable. Combining features like eq.(4) with other features derived from patent claim structures, supervised approaches may give us some knowledges on what kind of structure may be effective for extracting keywords of novelties.

A simple expansion of the method in step 1 of subsection 3.2 to Markush Claim will also increase the performance in the field of chemistry.

Future work also includes applying our approach to patent retrieval, patent summarization, or detecting important technologies.

References

Akihiro Shinmori, Manabu Okumura, Yuzo Marukawa, and Makoto Iwayama. 2003. *Patent Claim Processing for Readability: Structure Analysis and Term Explanation*. Proceedings of the ACL-2003 Workshop on Patent Corpus Processing, 20:56-65.

Assad Abbas, Limin Zhang, and Samee U. Khan. 2014. *A literature review on the state-of-the-art in patent analysis*. World Patent Information, 37:3-13.

Fu-ren Lin and Feng-mei Huang. 2010. *The Study of Patent Prior Art Retrieval Using Claim Structure and Link Analysis*. Pacific Asia Conference on Information Systems:9-12.

Hidetsugu Nanba, Atsushi Fujii, Makoto Iwayama, and Taiichi Hashimoto. 2010. *Overview of the Patent Mining Task at the NTCIR-8 Workshop*. NTCIR-8 Workshop Meeting.

Hisao Mase, Tadataka Matsubayashi, Yuichi Ogawa, Makoto Iwayama, and Tadaaki Oshio. 2005. *Proposal of Two-stage Patent Retrieval Method Considering the Claim Structure*. ACM Transactions on Asian Language Information Processing, 4(2):190-206.

John D. Lafferty, Andrew McCallum and Fernando C. N. Pereira. 2001. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*. Proceedings of the 18th International Conference on Machine Learning:282-289.

Manisha Verma and Vasudeva Varma. 2011. *Applying Key Phrase Extraction to Aid Invalidity Search*. Proceedings of the 13th International Conference on Artificial Intelligence and Law:249-255.

Mohammad Al Hasan, W. Scott Spangler, Thomas Griffin, and Alfredo Alba. 2009. *COA: Finding Novel Patents Through Text Analysis*. Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining:1175-1184.

Peter Parapatics and Michael Dittenbach. 2009. *Patent Claim Decomposition for Improved Information Extraction*. Proceedings of the 2nd International Workshop on Patent Information Retrieval:33-36.

Risa Nishiyama, Yuta Tsuboi, Yuya Unno and Hironori Takeuchi. 2010. *Feature-Rich Information Extraction for the Technical Trend-Map Creation*. Proceedings of NTCIR Workshop 8 Meeting.

Stephen Robertson and Hugo Zaragoza. 2009. *The Probabilistic Relevance Framework: BM25 and Beyond*. Foundations and Trends in Information Retrieval, 3(4):333-389.

Svetlana Sheremetyeva, Sergei Nirenburg, and Irene Nirenburg. 1996. *Generating patent claims from interactive input*. Proceedings of the 8th. International Workshop on Natural Language Generation:61-70.

Toru Takaki, Atsushi Fujii, and Tetsuya Ishikawa. 2004. *Associative Document Retrieval by Query Subtopic Analysis and Its Application to Invalidity Patent Search*. Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management:399-405.

World Intellectual Property Organization. *INTERNATIONAL PATENT CLASSIFICATION (Version 2016) GUIDE*. `http://www.wipo.int/export/sites/www/classifications/ipc/en/guide/guide_ipc.pdf`.

Leveraging Multilingual Training for Limited Resource Event Extraction

Andrew Hsi **Yiming Yang** **Jaime Carbonell** **Ruochen Xu**
Carnegie Mellon University
Pittsburgh, PA 15213 USA
{ahsi, yiming, jgc, ruochenx}@cs.cmu.edu

Abstract

Event extraction has become one of the most important topics in information extraction, but to date, there is very limited work on leveraging cross-lingual training to boost performance. We propose a new event extraction approach that trains on multiple languages using a combination of both language-dependent and language-independent features, with particular focus on the case where target domain training data is of very limited size. We show empirically that multilingual training can boost performance for the tasks of event trigger extraction and event argument extraction on the Chinese ACE 2005 dataset.

1 Introduction

Traditionally, event extraction has focused on monolingual training – typically English (Grishman et al., 2005; Ji and Grishman, 2008; Gupta and Ji, 2009; Liao and Grishman, 2010; Liao and Grishman, 2011; Li et al., 2013; Bronstein et al., 2015), and occasionally Chinese or other languages (Chen and Ji, 2009b; Piskorski et al., 2011; Li et al., 2012; Chen and Ng, 2012; Chen and Ng, 2014). However, apart from a few isolated studies (Chen and Ji, 2009a; Piskorski et al., 2011), to date there is very little work leveraging cross-lingual information for event extraction. Cross-lingual approaches have proven useful for many other tasks in natural language processing (NLP), including part-of-speech (POS) tagging (Snyder et al., 2009; Cohen et al., 2011), dependency parsing (Zeman and Resnik, 2008; Cohen et al., 2011; McDonald et al., 2011; Ammar et al., 2016), and named entity recognition (Richman and Schone, 2008).

An important issue in event extraction is that the amount of available training data is often insufficient or unbalanced across domains and/or languages. Event extraction training datasets typically contain merely a few hundreds of documents, owing to the complexity and high costs of human annotation. This issue is even more severe for new event types in new languages. This provides strong motivation to leverage existing language resources for event extraction in new languages. This problem is closely related to low-resource NLP, which has been gathering increased interest among researchers (Garrette et al., 2013; Miao et al., 2013; Duong et al., 2014; Duong et al., 2015).

In this paper, we propose a novel approach for cross-lingual event extraction, which trains on multiple languages and leverages both language-dependent and language-independent features in order to boost performance. Using such a system we aim to jointly leverage available multilingual resources (annotated data and induced features) to overcome the annotation-scarcity issue in the target language of interest. Empirically we show that our approach can substantially improve the performance of monolingual systems for the task of Chinese event argument extraction. Our approach is novel compared to existing work in that we have no reliance on using either high quality machine translation or manually aligned documents, which may be unavailable for a given target language.

The rest of the paper is organized as follows. Section 2 introduces relevant terminology used in the event extraction field. Section 3 describes some related work on event extraction and cross-lingual NLP. Section 4 details our event extraction system and the types of features we use. In Section 5, we describe

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1201–1210, Osaka, Japan, December 11-17 2016.

our experimental setup and discuss results for both cross-lingual event trigger extraction and cross-lingual event argument extraction. We conclude in Section 6 with some ideas for future work.

2 Terminology and Task Definitions

We will begin by briefly introducing the basic terminology used in the event extraction field and the task definitions by the Automatic Content Extraction (ACE) Evaluation program[1] conducted by the National Institute of Science and Technology (NIST). The ACE program focused on entity detection, relation detection, and event detection – among these, we focus in this paper specifically on the event detection task, which consists of event trigger extraction and event argument extraction.

- An *event* is something that happens at a particular time and place, often involving one or more people. Examples include births, attacks, and arrests.
- An *event mention* is a particular textual occurrence of an event. A text may contain several different mentions that all refer to the same physical event.
- An *event trigger* is the specific word in a sentence that indicates the existence of an event.
- An *event argument* is an entity fulfilling a specific role within the event. The set of permissible roles depends on the event type. For example, the Attacker role would be valid for a Conflict.Attack event, while the same role would be invalid for an event of type Business.Declare-Bankruptcy. Additionally, all event types in ACE include roles for Time and Place.
- Lastly, an *event argument mention* is a particular textual occurrence of an event argument.

The *event trigger extraction task* is to identify all of the event triggers contained within a set of documents. The *event argument extraction* task is to identify all of the event arguments contained within a set of documents. In most cases, the event trigger extraction step is conducted first to identify the event mentions, and then event argument extraction is performed on top of this to identify the particular entities fulfilling argument roles for these event mentions.

3 Related Work

A variety of machine learning methods have been used for event extraction in the past, including pipelines of classifiers (Grishman et al., 2005; Ji and Grishman, 2008; Liao and Grishman, 2011), joint inference models (Li et al., 2013; Li and Ji, 2014; Yang and Mitchell, 2016), and neural networks (Nguyen and Grishman, 2015; Chen et al., 2015) – the vast majority of which focus solely on the English monolingual training scenario. A subset of the event extraction literature has considered the study of Chinese event extraction (Chen and Ji, 2009b; Li et al., 2012; Chen and Ng, 2012; Chen and Ng, 2014). However, most of these works also focus solely on the monolingual case, and do not leverage any additional training data from other languages.

The most related work to our approach is that of Chen and Ji (2009a). In their model, they designed a co-training approach to augment a small Chinese training corpus with additional examples from an unlabeled corpus. Given a parallel corpus of English-Chinese documents and a monolingual English event extraction system (trained on annotated English documents), they used the system to predict the event labels on the English part of the parallel documents and project the predicted labels to the Chinese part of the parallel corpus based on gold standard alignments. The Chinese system is then trained using a combination of the originally annotated Chinese document and the parallel texts with the projected labels. This approach offered slight improvements in the event trigger extraction task and the event argument extraction task (see Section 2 for definitions), but relies on having in-domain parallel texts either aligned by humans or by high quality machine translation models between the source and target languages. In contrast, our proposed approach has no such limitation, and hence is easier to apply to any target language of interest.

[1] http://www.itl.nist.gov/iad/mig/tests/ace/

Another related work is that of Piskorski et al. (2011), who use cross-lingual information to refine the results of event extraction. In particular, they run several monolingual event extraction systems independently, translate the extracted argument fillers into English, and merge together argument fillers across documents. Using this cross-document information fusion, they find improved performance over monolingual systems. However, this work relies on having documents across multiple languages that describe the exact same event, which is an unrealistic case in practice. Additionally, they also rely on having high quality machine translation in order to translate the argument output of each monolingual system into English.

There does exist some prior work on the broader field of cross-lingual information extraction. Riloff et al. (2002) start with English annotated source texts, create a parallel corpus via machine translation, and project the annotations via alignments. The projected annotations are then used to conduct training in the target language. Sudo et al. (2004) presents an approach for extracted patterns in a source language and translating these patterns for use on a target language. However, these works are limited to entity extraction, whereas our focus is on event extraction. Furthermore, both works rely on having high-quality machine translation output.

Beyond information extraction, cross-lingual training has offered benefits for a variety of tasks. McDonald et al. (2011) use a delexicalized English parser to seed a lexicalized parser in the target language, and then iteratively improve upon this model via constraint driven learning. Duong et al. (2014) develop a POS tagger for low resource languages by first projecting predicted English POS tags across parallel data to obtain target language training data, and then further augment this with a small amount of annotated data in the target language. Ammar et al. (2016) developed a language universal dependency parser by using language-independent features to create a general model, and fine-tuning the resulting model with language-specific features and embeddings. Similarly to our model, this method has no requirement about the availability of alignments and parallel text.

4 System Description

To date, there exist only a handful of languages that have ACE-style event annotations, yet this leaves a vast number of languages in which people have no capacity to conduct event extraction. This problem is compounded by the difficulty of event extraction annotation. Annotation of documents for event extraction is a very labor-intensive, costly task – even the standard benchmark dataset of ACE 2005 only contains several hundred annotated English documents. It is inconceivable to believe that we will ever have similar datasets for every language of interest.

A natural effort therefore, is to leverage as much information as we can from existing "high-resource" event-extraction languages, along with whatever limited training data we may have for the target language. To accomplish this, we create a standard pipeline-of-classifiers approach to event extraction, and then augment this model with multilingual features.

The overall system architecture may be seen in Figure 1. We begin by preprocessing the data to obtain tokenizations, POS tags, and dependency parses. We then extract our trigger features, and run a multi-class classifier to predict the trigger labels. We subsequently extract our argument features using the original preprocessed data in combination with the system predicted trigger labels, and run a second multi-class classifier to make predictions on the argument roles.

4.1 Trigger Prediction

We begin by describing our trigger prediction component. For the task of event trigger prediction, we train a multi-class logistic regression classifier using LIBLINEAR (Fan et al., 2008). For each word, we make a prediction on the event trigger type – one of 33 given types from the ACE ontology, or the NONE category to represent when a word does not trigger an ACE event.

The trigger system uses a variety of monolingual features, seen in Table 1. For the majority of the features, we use binary indicator functions to represent whether the feature is either active or inactive for the particular data instance. For the word embedding features, we use the real-valued vectors directly for representing each word.

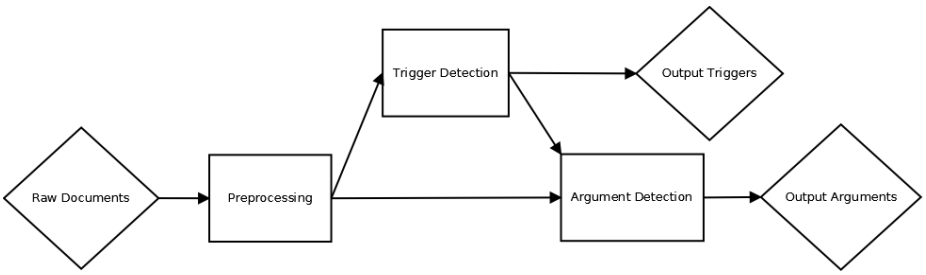


Figure 1: Architecture for our event extraction system. The argument component relies on the predictions from the trigger component.

Event Trigger Extraction Features
Lexical features (e.g. words and lemmas within a context window)
Length of the current word
Language-specific POS tags within a context window
Universal POS tags within a context window
Word embedding vector for current word
Dependent/Governor information from dependency parsing
Bilingual dictionary word pairs

Table 1: Features used in the Event Trigger Extraction component

Multilingual training is leveraged via the use of four types of features: 1.) Universal POS Tags (Petrov et al., 2012), 2.) Universal Dependencies (McDonald et al., 2013), 3.) limited bilingual dictionaries, and 4.) aligned multilingual word embeddings. The Universal POS tags and Universal Dependencies allow us to use a single set of tags for both languages, which thereby enables the use of English training data directly in our model. The bilingual dictionary provides a limited number of translations between words, and may be used both directly in the model and for aligning word embeddings. The aligned word embeddings similarly allow us to directly use English training data, as each component in the vector is aligned to semantically match those of the target word embeddings.

To obtain aligned word embeddings, we first start with monolingual word embeddings. We obtain monolingual texts for both English and the target language from Wikipedia, and independently train word embeddings for each language using word2vec (Mikolov et al., 2013). These monolingual embeddings are then aligned by solving a regression problem.

Let $D = (x_i, z_i)_{i=1}^{n}$ represent a limited bilingual dictionary between the two languages, where $x_i \in \mathbb{R}^{d_1}$ is the word embedding of word i in English and $z_i \in \mathbb{R}^{d_2}$ is the word embedding of its translation in the target language. Our regression problem is to find a transformation matrix W minimizing the following objective function:

$$\min_{W \in \mathbb{R}^{d2 \times d1}} \sum_{i=1}^{n} \|Wx_i - z_i\|^2 + \lambda \|W\|_F^2 \tag{1}$$

The first term of the objective function serves to ensure that the projected vectors in English closely match those of their translations in the target language. The second term is a regularization term to avoid overfitting. This problem has a closed form solution, which is given by:

$$W^* = ZX^T(XX^T + \lambda I)^{-1} \tag{2}$$

where $X = [x_1, x_2, ..., x_n] \in \mathbb{R}^{d_1 \times n}$ and $Z = [z_1, z_2, ..., z_n] \in \mathbb{R}^{d_2 \times n}$

The resulting aligned embeddings may then be used in our feature set. Using these aligned embeddings is preferable over just using the direct dictionary translations, as many words in both the source and target language may not appear in the bilingual dictionary. In our approach, once a mapping is found between two embedded spaces, we may project any word into this shared space.

4.2 Argument Prediction

We now describe the argument prediction component of our system. For this component, we require a few additional fields of information: 1.) event trigger words, and 2.) entity mentions. Event trigger words and entity mentions may be provided either as gold information or extracted automatically using machine learning approaches.

In each sentence, for each (trigger word, entity mention) pair, we make a prediction on the argument role (if any) between the trigger and the entity. The ACE ontology contains 35 different argument types, and we also include the NONE label to indicate when there is no relationship between the trigger and the entity. As in the previous case of trigger extraction, we once again accomplish this by training a multi-class logistic regression classifier using LIBLINEAR.

Event Argument Extraction Features
Lexical features about the entity phrase
Lexical features for individual words in the entity phrase
Entity type, subtype
Event type and subtype of trigger word
Existence of any other candidate entities in the same sentence
Distance between the trigger and entity
Dependent/Governor information from dependency parsing
Bilingual dictionary word pairs

Table 2: Features used in the Event Trigger Extraction component

Features for the event argument extraction component may be seen in Table 2. Multilingual information is leveraged in a similar way to the trigger prediction component, using Universal POS tags, Universal Dependencies, and any available bilingual dictionaries to learn from English training data.

5 Experimental Setup

To test our approach, we conduct experiments on two separate tasks: event trigger extraction and event argument extraction. We begin by describing our experimental setup and metrics, and subsequently show empirical results on the two tasks.

5.1 Dataset

We conduct experiments on the ACE 2005 dataset, the most dominating benchmark dataset for event trigger extraction and event argument extraction. The English and Chinese portions of ACE each contain several hundred documents annotated with gold standard entity and event information. We preprocess the raw text of each document using Stanford CoreNLP (Manning et al., 2014). We split the Chinese portion into 10 folds, and perform cross-validation. In the ACE collection, the number of labeled Chinese documents is approximately the same as the number of English documents, so to simulate a low resource scenario, we select just one training fold for each round of cross-validation. We use another fold for parameter tuning, and use the remaining folds in each round for testing. We use CC-CEDICT[2] as our bilingual dictionary between English and Chinese.

For our baseline system, we use just the Chinese data for training, and only the monolingual features. Our cross-lingual system uses the entire set of features, and additionally incorporates the entire English portion of ACE 2005 for training.

5.2 Metrics

We report both micro-averaged and macro-averaged precision, recall, and F1. Typically the event extraction community reports micro-averaged results, which give the overall performance after pooling all the labels together. However, we argue that only presenting this single perspective provides a skewed

[2]downloadable from https://cc-cedict.org/wiki/start

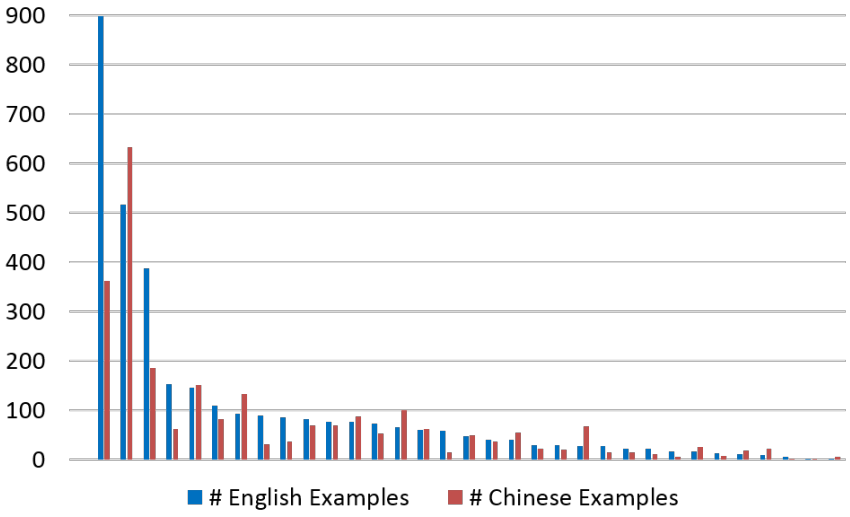


Figure 2: Distribution of Trigger Types in ACE 2005. Each bar represents one of the event types found in ACE 2005, and the height of the bar represents the number of event mentions for said class.

Trigger Types (frequency)	
Conflict.Attack (1252)	Movement.Transport (607)
Life.Die (488)	Personnel.End-Position (196)
Contact.Meet (190)	Personnel.Elect (143)
Transaction.Transfer-Money (140)	Life.Injure (116)
Justice.Charge-Indict (107)	Contact.Phone-Write (107)
Transaction.Transfer-Ownership (101)	Justice.Trial-Hearing (100)
Justice.Sentence (94)	Personnel.Start-Position (92)
Justice.Arrest-Jail (88)	Justice.Convict (75)
Conflict.Demonstrate (72)	Life.Marry (58)
Justice.Sue (55)	Life.Be-Born (46)
Business.Declare-Bankruptcy (40)	Justice.Appeal (39)
Business.Start-Org (38)	Justice.Release-Parole (35)
Business.End-Org (33)	Life.Divorce (28)
Justice.Fine (28)	Justice.Execute (20)
Business.Merge-Org (18)	Personnel.Nominate (11)
Justice.Acquit (7)	Justice.Extradite (3)
Justice.Pardon (2)	

Table 3: Counts of trigger types in English ACE 2005.

view of the system performance, as this type of measure is highly favorable to the majority class labels. This is particularly an issue for the ACE 2005 collection, as the distribution over both trigger types and argument roles is highly skewed, as seen in Figures 2 and 3.

Table 3 shows the specific counts for each trigger type in the English portion of ACE 2005. The trigger type "Conflict.Attack" occurs far more frequently in the texts than any of the others – more than twice that of the second most common type. On the other extreme, the highly infrequent types only occur very rarely in the text. Analysis of the argument counts (Table 4) shows a similar situation. While not as badly skewed as in the trigger case, there is still noticeable disparity between the most frequent and least frequent classes. Some of this may be attributed to the fact that ACE includes several argument types that correspond to different varieties of "Time", but even if we ignore the "Time"-type arguments, there are still nine argument classes with less than 100 examples each.

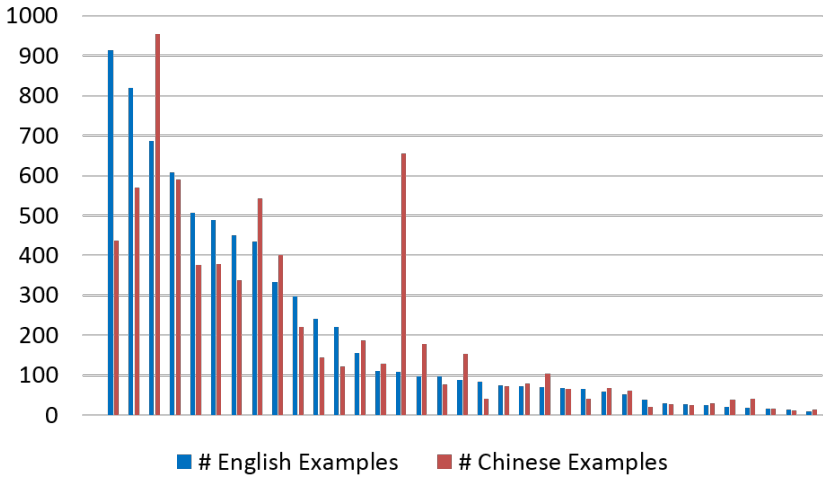


Figure 3: Distribution of Argument Roles in ACE 2005. Each bar represents one of the argument roles found in ACE 2005, and the height of the bar represents the number of argument mentions for said class.

Argument Roles (frequency)		
Person (1064)	Place (891)	Time-Within (699)
Entity (685)	Attacker (564)	Target (535)
Victim (529)	Destination (462)	Agent (368)
Defendant (359)	Crime (245)	Instrument (244)
Origin (160)	Artifact (131)	Position (111)
Giver (108)	Recipient (107)	Adjudicator (106)
Org (105)	Buyer (79)	Vehicle (78)
Money (75)	Sentence (74)	Plaintiff (72)
Time-Holds (68)	Time-Starting (52)	Beneficiary (42)
Seller (37)	Prosecutor (29)	Time-After (24)
Time-Before (20)	Time-Ending (19)	Time-At-End (15)
Time-At-Beginning (15)	Price (8)	

Table 4: Counts of the argument roles in English ACE 2005.

5.3 Event Trigger Extraction results

Results for trigger extraction may be seen in Table 5. The cross-lingual approach shows improved performance on both the micro-averaged and macro-averaged F1 metrics, demonstrating the success of incorporating multilingual training. On the macro-averaged metric, we see an improvement of 10.7%, and on the micro-averaged metric, an improvement of 3.9%. We find these improvements on F1 to be significant under a t-test with α=0.01.

As one would expect, the macro-averaged scores are noticeably lower than the micro-averaged scores, which suggests that the rare classes for event triggers suffer from worse performance than the frequent classes. Note that the difference in performance between the two approaches is larger on the macro-average metric. This suggests that the addition of multilingual training is playing a key role to improve performance on these particular rare classes.

	Macro-Average			Micro-Average		
	Precision	Recall	F1	Precision	Recall	F1
Monolingual approach	0.421	0.183	0.233	**0.646**	0.271	0.381
Cross-lingual approach	**0.443**	**0.209**	**0.258**	0.635	**0.288**	**0.396**

Table 5: Results of Trigger Extraction Task on Chinese ACE 2005

5.4 Event Argument Extraction results

Results for argument extraction may be seen in Tables 6 and 7. Table 6 shows the optimal performance obtained by using the gold event triggers as input, and Table 7 shows the more realistic scenario of using the system predicted triggers as input. In both cases, we utilize the existing gold entity information provided by the ACE collection.

We see even larger boosts in macro-average performance from the argument extraction component than from the trigger extraction component – using gold triggers we get a 34.8% improvement on macro F1, and using system predicted triggers we get a 28.2% improvement on macro F1. On micro-average metrics, we find a smaller, but still meaningful boost in performance: 3.2% improvement on micro F1 when using gold triggers as input, and 5.7% improvement on micro F1 when using the system predicted triggers. We find all of our argument results to show significant improvements on F1 over the monolingual equivalents under a t-test with α=0.01.

We suspect that the noticeably larger gains in argument macro-average performance compared to trigger performance may be due to the more semantic nature of the task. Trigger words are primarily dependent on lexical information, whereas arguments rely more heavily on deeper semantic information such as that provided by dependency parsing. Information leveraged from sources like Universal Dependencies is therefore likely to have a greater effect in the argument extraction setting, and in particular on the rare classes that do not have enough data to perform well under monolingual training.

	Macro-Average			Micro-Average		
	Precision	Recall	F1	Precision	Recall	F1
Monolingual approach	0.510	0.189	0.250	**0.744**	0.336	0.462
Cross-lingual approach	**0.556**	**0.267**	**0.337**	0.731	**0.355**	**0.477**

Table 6: Results of Argument Extraction Task on Chinese ACE 2005, using gold trigger labels as input

	Macro-Average			Micro-Average		
	Precision	Recall	F1	Precision	Recall	F1
Monolingual approach	0.400	0.080	0.124	**0.651**	0.140	0.230
Cross-lingual approach	**0.422**	**0.105**	**0.159**	**0.651**	**0.150**	**0.243**

Table 7: Results of Argument Extraction Task on Chinese ACE 2005, using system predicted trigger labels as input

6 Conclusion

In this paper, we proposed a cross-lingual approach to event extraction that leverages both language-dependent and language-independent features to train with multiple languages. Motivated by the necessity of developing new techniques for expansion of event extraction to new languages, and inspired by the recent success stories of cross-lingual NLP, we developed an approach which allows us to incorporate any additional training data from other languages, while also maximally utilizing whatever monolingual data we have available. Our experimental results show improved performance for event trigger extraction and event argument extraction with multilingual training, under both the macro-averaged and micro-averaged metrics. These are very encouraging numbers, and we believe this indicates event extraction to be a promising direction for future cross-lingual researchers to explore.

There are several natural extensions to this work. One interesting direction of research would be to explore an actual (rather than simulated) low-resource language, where the target language may not only have little (or no) training data, but may not even have available tools for preprocessing tasks (POS tagging, entity recognition, parsing, etc.). This is an important area of research, as the majority of the world's languages do not have such tools available. A second direction of promising research is consider the case of not just leveraging a single source language, but to rather include multiple source languages.

A final interesting direction is to adapt recent neural methods for cross-lingual NLP, such as those by Ammar et al. (2016). By using a more sophisticated machine learning approach, we may be able to improve our multilingual efforts even further than our current approach.

Acknowledgements

This research was supported in part by DARPA grant FA8750-12-2-0342 funded under the DEFT program.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many languages, one parser. In *TACL*.

Ofer Bronstein, Ido Dagan, Qi Li, Heng Ji, and Anette Frank. 2015. Seed-based event trigger labeling: How far can event descriptions get us? In *ACL*.

Zheng Chen and Heng Ji. 2009a. Can one language bootstrap the other: A case study on event extraction. In *NAACL HLT Workshop on Semi-supervised Learning for Natural Language Processing*.

Zheng Chen and Heng Ji. 2009b. Language specific issue and feature exploration in chinese event extraction. In *HLT-NAACL*.

Chen Chen and Vincent Ng. 2012. Joint modeling for chinese event extraction with rich linguistic features. In *COLING*.

Chen Chen and Vincent Ng. 2014. Sinocoreferencer: An end-to-end chinese event coreference resolver. In *LREC*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL*.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *EMNLP*.

Long Duong, Trevor Cohn, Karin Verspoor, Steven Bird, and Paul Cook. 2014. What can we get from 1000 tokens? a case study of multilingual pos tagging for resource-poor languages. In *EMNLP*.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. A neural network model for low-resource universal dependency parsing. In *EMNLP*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. In *JMLR*.

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-world semi-supervised learning of pos-taggers for low-resource languages. In *ACL*.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nyus english ace 2005 system description. In *Proc. ACE 2005 Evaluation Workshop*.

Prashant Gupta and Heng Ji. 2009. Predicting unknown time arguments based on cross-event propagation. In *ACL-IJCNLP*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*.

Qi Li and Heng Ji. 2014. Constructing information networks using one single model. In *EMNLP*.

Peifeng Li, Guodong Zhou, Qiaoming Zhu, and Libin Hou. 2012. Employing compositional semantics and discourse consistency in chinese event extraction. In *EMNLP*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL*.

Shasha Liao and Ralph Grishman. 2010. Filtered ranking for bootstrapping in event extraction. In *ACL*.

Shasha Liao and Ralph Grishman. 2011. Acquiring topic features to improve event extraction: in pre-selected and balanced collections. In *RANLP*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL*.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *EMNLP*.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *ACL*.

Yajie Miao, Florian Metze, and Shourabh Rawat. 2013. Deep maxout networks for low-resource speech recognition. In *IEEE Workshop on Automatic Speech Recognition and Understanding*.

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

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *ACL*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *LREC*.

Jakub Piskorski, Jenya Belayeva, and Martin Atkinson. 2011. Exploring the usefulness of cross-lingual information fusion for refining real-time news event extraction: A preliminary study. In *RANLP*.

Alexander E. Richman and Patrick Schone. 2008. Mining wiki resources for multilingual named entity recognition. In *ACL*.

Ellen Riloff, Charles Schafer, and David Yarowski. 2002. Inducing information extraction systems for new languages via cross-language projection. In *COLING*.

Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. 2009. Adding more languages improves unsupervised multilingual part-of-speech tagging: A bayesian non-parametric approach. In *NAACL*.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2004. Cross-lingual information extraction system evaluation. In *COLING*.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. In *NAACL*.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *IJCNLP*.

LILI: A Simple Language Independent Approach for Language Identification

Mohamed Al-Badrashiny and Mona Diab
Department of Computer Science, The George Washington University
{badrashiny,mtdiab}@gwu.edu

Abstract

We introduce a generic Language Independent Framework for Linguistic Code Switch Point Detection. The system uses the word length, character level (1, 2, 3, 4, and 5)-grams and word level unigram language models to train a conditional random fields (CRF) model for classifying input words into various languages. We test our proposed framework and compare it to the state-of-the-art published systems on standard data sets from several language pairs: English-Spanish, Nepali-English, English-Hindi, Arabizi (Refers to Arabic written using the Latin/Roman script)-English, Arabic-Engari (Refers to English written using Arabic script), Modern Standard Arabic(MSA)-Egyptian, Levantine-MSA, Gulf-MSA, one more English-Spanish, and one more MSA-EGY. The overall weighted average F-score of each language pair are 96.4%, 97.3%, 98.0%, 97.0%, 98.9%, 86.3%, 88.2%, 90.6%, 95.2%, and 85.0% respectively. The results show that our approach despite its simplicity, either outperforms or performs at comparable levels to state-of-the-art published systems.

1 Introduction

Linguistic Code Switching (LCS) is a common practice among multilingual speakers in which they switch between their common languages in written and spoken communication. In Spanish-English for example: "She told me that mi esposo looks like un buen hombre." ("She told me that my husband looks like a good man"). In this work we care about detecting LCS points as they occur intra-sententially where words from more than one language are mixed in the same utterance. LCS is observed on all levels of linguistic representation, and especially pervasive in social media. LCS poses a significant challenge to NLP, hence detecting LCS points is a very important task for many downstream applications. In this paper we address this challenge using a generic simple language independent approach. We illustrate our approach on several language pairs utilizing publicly available data sets and comparing our performance against state-of-the-art sophisticated systems tailored to the problem of LCS point detection (LCSPD). Furthermore, we show the robustness of our approach on the most challenging problem of language variety code switching where the code switching is happening between a standard and dialect, namely we illustrate our performance on Modern Standard Arabic (MSA) mixed with Egyptian Dialectal Arabic data (EGY).

2 Related Work

Several systems have recently addressed the problem of LCSPD in written text both within language varieties and across different language pairs. Relevant work on the problem of LCSPD among different language pairs can be summarized in the following works.

3ARRIB (Al-Badrashiny et al., 2014; Eskander et al., 2014) addresses the challenge of how to distinguish between Arabic words written using Roman script (Arabizi) and actual English words in the same context/utterance. The assumption in this framework is that the script is Latin for all words. It trains a finite state transducer (FST) to learn the mapping between the Roman form of the Arabizi words and

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1211–1219, Osaka, Japan, December 11-17 2016.

their Arabic form. It uses the resulting FST to find all possible Arabic candidates for each word in the input text. These candidates are filtered using MADAMIRA (Pasha et al., 2014), a state-of-the-art morphological analyzer and POS disambiguation tool, to filter out non-Arabic solutions. Finally, it leverages a decision tree that is trained on language model probabilities of both the Arabic and Romanized forms to render the final decision for each word in context as either being Arabic or English.

Bar and Dershowitz (2014) addresses the challenge for Spanish-English LCSPD. The authors use several features to train a sequential Support Vector Machines (SVM) classifier. The used features include previous and following two words, substrings of 1-3 character ngrams from the beginning and end of each word thereby modeling prefix and suffix information, a boolean feature indicating whether the first letter is capitalized or not, and 3-gram character and word ngram language models trained over large corpora of English and Spanish, respectively.

Barman et al. (2014) present systems for both Nepali-English and Spanish-English LCSPD. The script for both language pairs is Latin based, i.e. Nepali-English is written in Latin script, and Spanish-English is written in Latin script. The authors carry out several experiments using different approaches including dictionary-based methods, linear kernel SVMs, and a k-nearest neighbor approach. The best setup they found is the SVM-based one that uses character n-gram, binary features indicate whether the word is in a language specific dictionary of the most frequent 5000 words they have constructed, length of the word, previous and next words, 3 boolean features for capitalization to check if the first letter is capitalized, if any letter is capitalized, or if all the letters are capitalized.

The approach presented by King et al. (2014) utilizes character n-gram probabilities, lexical probabilities, word label transition probabilities and existing named entity recognition tools within a Markov model framework.

Jain and Bhat (2014) use a CRF based token level language identification system that uses a set of easily computable features (Ex. isNum, isPunc, etc.). Their analysis showed that the most important features are the word n-gram posterior probabilities and word morphology.

Lin et al. (2014) use a CRF model that relies on character n-grams probabilities (tri and quad grams), prefixes, suffixes, unicode page of the first character, capitalization case, alphanumeric case, and tweet-level language ID predictions from two off-the-shelf language identifiers: cld2[1] and ldig.[2] They increase the size of the training data using a semi supervised CRF autoencoder approach (Ammar et al., 2014) coupled with unsupervised word embeddings.

MSR-India (Chittaranjan et al., 2014) uses character n-grams to train a maximum entropy classifier that identifies whether a word is language1 or language2. The resultant labels are then used together with word length, existence of special characters in the word, current, previous and next words to train a CRF model that predicts the token level classes of words in a given sentence/tweet.

On the other hand, for within language varieties, AIDA (Elfardy et al., 2014) and AIDA2 (Al-Badrashiny et al., 2015) are the best published systems attacking this problem in Arabic. In this context, the problem of LCSPD is more complicated than mixing two very different languages since in the case of varieties of the same language, the two varieties typically share a common space of cognates and often faux amis, where there are homographs but the words have very different semantic meanings, hence adding another layer of complexity to the problem. In this set up the assumed script is Arabic script. AIDA (Elfardy et al., 2014) uses a weakly supervised rule based approach that relies on a language model to tag each word in the given sentence. Then it uses the LM decision for each word in the given sentence/tweet and combine it with other morphological information to decide upon the final class of each word. AIDA2 (Al-Badrashiny et al., 2015) uses a complex system that is based on a mix of language dependent and machine learning components to detect the linguistic code switch between the modern standard Arabic (MSA) and Egyptian dialect (EGY) that are both written using Arabic script. It uses MADAMIRA (Pasha et al., 2014) to find the POS tag, prefix, lemma, suffix, for each word in the input text. Then it models these features together with other features including word level language model probabilities in a series of classifiers where it combines them in a classifier ensemble approach to

[1] https://code.google.com/p/cld2/

[2] https://github.com/shuyo/ldig

find the best tag for each word.

We compare our system to all of the above systems in addition to some other baselines.

3 Approach

In this paper, we present a very simple language independent framework called LILI to address the challenge of linguistic code switch point detection (LCSPD) when it occurs using the same script for the mixed languages in the utterance. Our framework is mainly based on the assumption that each language has its own character pattern behaviors and combinations relating to the underlying phonology, phonetics, and morphology of each language independently. Accordingly, the manner of articulation constrains the possible phonemic/morphemic combinations in a language. For example in Arabic, it is hard to find a word that have the "th" sound followed by an "s" sound, while it is possible in English as in the word "thus". Historically, the famous Arab lexicographer Al-Farahidi (718 - 786 CE) noticed this phenomenon (where certain sound sequences are allowed while others are not in the language) and devised a method by which he can distinguish Arabic words from foreign ones on the basis of the possible sequences of letters in Arabic (Ahmad, 2003). Though having closed set of impossible sequences of letters in each language could help in distinguishing between languages within an utterance, but in reality it is hard to find such sets for all languages. Hence, we believe that building a character level n-gram language model for the target language to maximize the probabilities of the possible patterns and suppress the probabilities of the impossible ones should provide an approximation to such a closed set rule-based list. Not to mention that producing such a list by hand is quite laborious and error prone as a process.

Accordingly, we propose a supervised learning framework to address the challenge of LCSPD. We assume the presence of annotated code switched training data where each token is annotated as either Lang1 or Lang2. We create a sequence model using Conditional Random Fields (CRF++) tool (Sha and Pereira, 2003). For each word in the training data, we create a feature vector comprising word length, character sequence level probabilities, and unigram word level probabilities. Once we derive the learning model, we apply to input text to identify Lang1 tokens vs. Lang2 tokens in context. For the character sequence level probabilities, we build (1, 2, 3, 4, and 5)-gram character language models (CLMs) using the SRILM tool (Stolcke, 2002) for each of the two languages presented in the training data using the annotated words. For example, if the training data contains the two languages "lang1" and "lang2", we use all words that have the "lang1" tags to build (1, 2, 3, 4, and 5)-grams CLMs for "lang1" and the same for "lang2". We apply all of the created CLMs to each word in the training data to find their character sequence probabilities in each language in the training data. To increase the difference between the feature vectors of the words related to "lang1" and those related to "lang2", we use a word level unigram LM for each of the two languages in the training data. Then we apply the unigram LMs to each word in the training data to find their word level probabilities in each language in the training data, i.e. checking whether it pertains to the language or not by virtue of having a higher probability in the corresponding LM than words not in the language.

4 Experimental Setup

4.1 Data

We evaluate our proposed framework on different language pairs exhibiting code switching. We use the training and test data sets provided by the shared task for "Language Identification in Code-Switched Data" [ShTk] in 2014 and 2016 (Solorio et al., 2014; Molina et al., 2016). The ShTk-2014 datasets includes English-Spanish, English-Nepali, Modern standard Arabic (MSA)-Egyptian Arabic (EGY), and English-Mandarin[3], while the ShTk-2016 datasets includes English-Spanish, and, MSA-EGY. In addition to these languages, we evaluate our system on MSA-Levantine (LEV), MSA-Gulf, Arabizi-English, Arabic-Engari, and English-Hindi datasets[4].

[3]Unfortunately, we did not manage to get English-Mandarin datasets from the organizers but we got the rest of them.

[4]Nepali, Arabizi, and Hindi are written using Roman script. Engari is written using Arabic script

- MSA-LEV and MSA-Gulf: These datasets are collected from online newspaper commentary and Twitter by Cotterell and Callison-Burch (2014). The datasets are annotated for sentence level. We re-annotated the data for token level using the guidelines provided by Diab et al. (2016). We then split the data into training and test (80% for training and 20% for testing);

- Arabizi-English: We use the same training and test sets used by 3ARRIB. The data sets are created by the Linguistic Data Consortium from SMS/Chat corpus (Bies et al., 2014; LDC, 2014a; LDC, 2014b; LDC, 2014c);

- Arabic-Engari: Same as the MSA-EGY data sets. But we re-annotated the data to tag all English words that are written in Arabic script. MSA and EGY words are both tagged as Arabic words;

- English-Hindi: It consists of 728 and 376 sentences for training and test sets, respectively, collected from Twitter and Facebook. This dataset is part of a corpus that is used for POS-tagging experiment in code-switched data (Jamatia et al., 2015).

Table 1 shows the distribution of each language in the training and test sets. The lang1, lang2 labels refer to the two languages addressed in the dataset name, for example for the language pair English-Spanish, lang1 is English and lang2 is Spanish, in that order[5].

All Language-Pairs	Training-Set		Test-Set	
	lang1	lang2	lang1	lang2
English-Spanish-2014	77,101	33,099	7,424	5,278
Nepali-English-2014	60,493	44,111	12,286	17,216
MSA-EGY-2014	79,059	16,291	57,740	21,871
Arabizi-English-2014	93,402	11,122	27,308	1,903
Arabic-Engari	439,875	1,282	79,611	433
English-Hindi	6,562	5,526	8,676	378
LEV-MSA	44,694	11,522	11,524	2,265
Gulf-MSA	57,718	8,655	15,400	1,409
English-Spanish-2016	77,101	33,099	32,442	123,973
MSA-EGY-2016	79,059	16,291	5,804	9,630

Table 1: Language distribution (words/language) in the training and test data sets for all language-pairs

In addition to the training data described in table 1, we used the following datasets to improve the word level LMs of the English, Spanish and Arabic langauges:

- English Gigaword (LDC, 2003b): To build the unigram word level LM for the English part in English-Spanish, English-Nepali, and English-Hindi language-pairs;

- Spanish Gigaword (LDC, 2009): To build the unigram word level LM for the Spanish part in English-Spanish language-pair;

- Arabic Gigaword (LDC, 2003a): To build the unigram word level LM for the MSA part in MSA-EGY, LEV-MSA, and Gulf-MSA language-pairs;

- Egyptian discussion forums (LDC, 2012): To build the unigram word level LM for the EGY part in MSA-EGY language-pairs. It is also used in addition to the Arabic Gigaword to build the LM for the Arabic part in the Arabic-Engari language pair.

4.2 Baselines

We evaluate our approach against the best published results using the same training and test sets. The baselines include:

[5]The ShTk-2014 has a different naming convention for the Nepali-English, however we opt for changing the naming to indicate the majority class is Nepali as in lang1 and English is the minority class language lang2

- Majority: In this baseline, for each word in the test set, we check the most frequent tag for that word in the training set and assign it to that word. If the word is not in the training set, we give it the most frequent language tag observed overall in the training data;
- TAU: The results on the English-Spanish dataset obtained by Bar and Dershowitz (2014);
- DCU-UVT: The results on the English-Spanish and English-Nepali datasets obtained by Barman et al. (2014);
- CMU: The results on the English-Spanish, English-Nepali, and MSA-EGY datasets obtained by Lin et al. (2014);
- IUCL: The results on the English-Spanish, English-Nepali, and MSA-EGY datasets obtained by King et al. (2014);
- IIIT: The results on the English-Spanish, English-Nepali, and MSA-EGY datasets obtained by Jain and Bhat (2014);
- MSR-India: The results on the English-Spanish, English-Nepali, and MSA-EGY datasets obtained by Chittaranjan et al. (2014);
- AIDA: The results on the MSA-EGY dataset obtained by our contribution in the ShTk-2014 (Elfardy et al., 2014);
- AIDA2: The results on the MSA-EGY dataset obtained by our previous publication about AIDA2 system (Al-Badrashiny et al., 2015);
- 3ARRIB: The results on the Arabizi-English dataset obtained by Eskander et al. (2014);
- IIIT Hyderabad and HHU-UH-G: The best results on the English-Spanish-2016 and MSA-EGY-2016 respectively[6].

5 Evaluation

Table 2 summarizes the published results by all baselines systems on the English-Spanish-2014, English-Nepali-2014, Arabizi-English, and MSA-EGY-2014 datasets. The table shows that TAU is the best published system on the English-Spanish-2014 data, DCU-UVT is the best published system on the English-Nepali-2014 data, 3ARRIB is the best published system on the Arabizi-English, and AIDA2 is the best published system on the MSA-EGY-2014 data.

Table 3 shows the results of our system on all language-pairs compared to the best published results from table 2 and the majority baseline. Lang1 indicates the majority class as per the training data, while lang2 indicates the minority class in the training data. The results show that LILI yields competitive results compared to all published state-of-the-art systems. The overall weighted average F-score for LILI is higher than all the majority baselines. It is also either higher than the published state-of-the-art systems except in the MSA-EGY compared to AIDA2 and HHU-UH-G, or very close to the best published results as in the English-Spanish-2016 dataset (LILI got 95.2% compared to 96% of IIIT Hyderabad with only 0.8% difference). Arabic language pairs; MSA-EGY, Gulf-MSA, and LEV-MSA are the most challenging ones. Because unlike the other languages, the words in each of these pairs do not create disjoint sets, as mentioned earlier, there is significant overlap hence they share significant character and word patterns. This issue is even worse because the native Arabic speakers do not write the short vowels (also know as diacritics) while they are able to reconstruct them while reading without any problem. The MSA shares many words with the other Arabic dialects but with different diacritics. For example, the words (yalEabuwn) in MSA and (yilEabuwn) in Gulf (Both mean they are playing)

[6]We are unaware of the citations of these 2 papers since they are not published yet while writing this paper. We got the systems names and their results from the shared task website `http://care4lang1.seas.gwu.edu/cs2/call.html`

Language-Pairs	System	lang1	lang2	Avg-F
English-Spanish-2014	CMU	93.30%	93.60%	93.42%
English-Spanish-2014	DCU-UVT	93.60%	92.70%	93.23%
English-Spanish-2014	**TAU**	**95.20%**	**95.20%**	**95.20%**
English-Spanish-2014	IUCL	94.10%	93.20%	93.73%
English-Spanish-2014	IIIT	92.90%	92.00%	92.53%
English-Spanish-2014	MSR-India	94.20%	93.80%	94.03%
English-Nepali-2014	CMU	91.40%	93.20%	92.45%
English-Nepali-2014	**DCU-UVT**	**97.40%**	**96.50%**	**96.87%**
English-Nepali-2014	IUCL	80.80%	87.10%	84.48%
English-Nepali-2014	IIIT	96.90%	94.30%	95.38%
English-Nepali-2014	MSR-India	96.90%	94.80%	95.67%
Arabizi-English	**3ARRIB**	**97.40%**	**75.80%**	**95.99%**
MSA-EGY-2014	CMU	89.90%	81.10%	87.48%
MSA-EGY-2014	IUCL	81.10%	59.50%	75.17%
MSA-EGY-2014	IIIT	86.20%	52.90%	77.05%
MSA-EGY-2014	MSR-India	86.00%	56.40%	77.87%
MSA-EGY-2014	AIDA	89.40%	76.00%	85.72%
MSA-EGY-2014	**AIDA2**	**92.90%**	**82.90%**	**90.15%**

Table 2: Summary results of all published systems that use English-Spanish-2014, English-Nepali-2014, Arabizi-English, and MSA-EGY-2014 datasets. For each group, the F-score is presented for lang1 and lang2 followed by the weighted average F-score for both languages.

All	LILI			Best Published Results				Majority Baseline		
Language-Pairs	lang1	lang2	Avg-F	System	lang1	lang2	Avg-F	lang1	lang2	Avg-F
English-Spanish-2014	97.2%	95.3%	**96.4%**	TAU	95.2%	95.2%	95.2%	92.8%	88.0%	90.8%
Nepali-English-2014	97.6%	97.0%	**97.3%**	DCU-UVT	97.4%	96.5%	96.9%	92.8%	94.0%	93.3%
English-Hindi	98.8%	80.4%	**98.0%**	NA	NA	NA	NA	98.4%	64.9%	97.0%
Arabizi-English	98.3%	77.9%	**97.0%**	3ARRIB	97.4%	75.8%	96.0%	86.9%	36.5%	83.6%
Arabic-Engari	99.1%	64.2%	**98.9%**	NA	NA	NA	NA	95.0%	62.4%	94.8%
MSA-EGY-2014	86.0%	87.2%	86.3%	AIDA2	92.9%	82.9%	**90.2%**	70.9%	63.7%	68.9%
LEV-MSA	93.6%	61.0%	**88.2%**	NA	NA	NA	NA	90.1%	25.1%	79.4%
Gulf-MSA	95.5%	36.6%	**90.6%**	NA	NA	NA	NA	94.6%	13.6%	87.8%
English-Spanish-2016	88.6%	96.9%	95.2%	IIIT Hyderabad	92.3%	96.9%	**96.0%**	77.4%	84.1%	82.7%
MSA-EGY-2016	82.0%	86.8%	85.0%	HHU-UH-G	85.4%	90.4%	**88.5%**	59.6%	47.4%	52.0%

Table 3: Summary results of our system performance on all language-pairs compared to the best published results and Majority baselines. For each group, the F-score is presented for lang1 and lang2 followed by the weighted average F-score for both languages. There are no published systems for English Hindi, Arabic-Engari, LEV-MSA, and Gulf-MSA hence the NA (not available).

become the same after removing the short vowels (ylEbwn). Hence, modeling more nuanced features is needed such as POS tags and morphological information, which is the case in the AIDA2 system. But despite the simplicity of the presented approach in this paper, in the MSA-EGY-2014 and MSA-EGY-2016 datasets, our yielded weighted average F-scores (87.2% and 85.0%) are not far from the AIDA2 (90.15%) and HHU-UH-G (88.5%) scores. Furthermore, It worth mentioning that running AIDA2 on the MSA-EGY-2016 gives 85.2% weighted average F-score; which is lower than LILI.

In a supervised framework, minority class detection is the more challenging task. We compare our performance to the published systems where available as well as the majority baseline. Our results are comparable to the published state-of-the-art systems, even significantly better in the MSA-EGY-2014 dataset. Moreover, our results outperform the majority baseline in all language pairs.

We can notice from table 2 that neither of the baselines published systems achieve the best results across the different language pairs. For example the DCU-UVT system achieved best result on Nepali-EN-2014 but it did not even achieve the second best on Spanish-EN-2014. Although the MSR-India has higher result than the DCU-UVT on the Spanish-EN-2014, it has lower results than it on the Nepali-EN-2014 data. This shows that despite the simplicity of our approach, it is generic and outperforms the states of the art systems or competitively compared to them across all the language pairs.

The above results show that the proposed approach is working well on the binary problems when we know beforehand the word is either lang1 or lang2. However, in real code switching scenario we do not know what the other language variety could be in the data. To see how robust our approach is, we tested our approach on a multi-class model. We trained a single multi-class CRF model using a combined data from our different training sets and tested it on the corresponding combined test sets. Unfortunately, we didn't manage to create a CRF model using all our training data. The datasets we managed to use are the English-Spanish-2014, Nepali-English-2014, English-Hindi, Arabizi-English, and MSA-EGY-2014. Table 4 shows the F-score results of our system on all languages using the multi-class model. We didn't find any published systems that conducted the same experiment to compare our results to. Therefore, table 4 only compares our results to the majority baseline and the binary model in table 3. The results show that LILI outperforms the majority baseline on all languages. The F-scores on all languages are comparable to the F-scores from the binary-classes case except on Hindi, where the multi-class model is much lower than the binary one. However, the overall weighted average F-score is high (91.0%). This shows that LILI is able to perform in a good way on the real code switching problem.

Language	LILI-Multi-Class-Model	LILI-Binary-Model	Majority Baseline
Nepali	97.4%	97.6%	90.0%
Hindi	62.3%	80.4%	30.7%
Arabizi	95.6%	98.3%	80.7%
English	96.2%	96.4%	93.4%
MSA	85.5%	86.0%	70.0%
Spanish	94.3%	95.3%	79.1%
Engari	64.0%	64.2%	56.5%
EGY	87.7%	87.2%	63.0%
Avg-F	91.0%	91.2%	77.8%

Table 4: The F-score of our system compared to the majority baseline on all languages using a single multi-class model. The last row shows the weighted average F-score for all languages.The number in the English row of LILI-Multi-Class-Model is the weighted average F-score of the English label obtained by LILI in table 3 on English-Spanish-2014, Nepali-English-2014, English-Hindi, and Arabizi-English datasets.

Finally, the simplicity of our system made it very fast. It can process up to 20,000 words/sec; which renders it very efficient and amenable to large scale processing. We compare our system's speed to our previously published tools; AIDA2 and 3ARRIB. AIDA2 processes 1000 words/sec and 3ARRIB processes 49 words/sec. Hence LILI is orders of magnitude faster than both systems with a relatively minor drop in performance compared to AIDA2 if we consider overall F1-score, or better performance if we care about detecting the minority class, and better performance than 3ARRIB overall.

6 Conclusion

In this paper, we introduced a simple yet powerful framework for the linguistic code switch point detection problem. The solution is language independent, thus it can work with any language-pair. The framework is based on the idea that each language has its own phonological system that control which set of sounds can occur together. Our assumption was that this is sufficient to distinguish between languages used in the same utterance. The results show that our simple approach outperforms or at least performs competitively compared to state-of-the-art complex machinery systems when evaluated against standard data sets with the added advantage of speeds that could be scaled up to big data levels.

References

Al-Khalil Ibn Ahmad. 2003. *Kitab al-?Ain*. Edited by Abdulhameed Hindawy(2003), Dar Al-kotob Alilmiyah, Beirut, Lebanon, first edition.

Mohamed Al-Badrashiny, Ramy Eskander, Nizar Habash, and Owen Rambow. 2014. Automatic Transliteration of

Romanized Dialectal Arabic. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 30–38, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Mohamed Al-Badrashiny, Heba Elfardy, and Mona Diab. 2015. Aida2: A hybrid approach for token and sentence level dialect identification in arabic. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 42–51, Beijing, China, July. Association for Computational Linguistics.

Waleed Ammar, Chris Dyer, and Noah A Smith. 2014. Conditional random field autoencoders for unsupervised structured prediction. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3311–3319. Curran Associates, Inc.

Kfir Bar and Nachum Dershowitz. 2014. The tel aviv university system for the code-switching workshop shared task. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 139–143, Doha, Qatar, October. Association for Computational Linguistics.

Utsab Barman, Joachim Wagner, Grzegorz Chrupała, and Jennifer Foster. 2014. Dcu-uvt: Word-level language classification with code-mixed data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 127–132, Doha, Qatar, October. Association for Computational Linguistics.

Ann Bies, Zhiyi Song, Mohamed Maamouri, Stephen Grimes, Haejoong Lee, Jonathan Wright, Stephanie Strassel, Nizar Habash, Ramy Eskander, and Owen Rabmow. 2014. Transliteration of Arabizi into Arabic Orthography: Developing a Parallel Annotated Arabizi-Arabic Script SMS/Chat Corpus. In *Arabic Natural Language Processing Workshop, EMNLP*, Doha, Qatar.

Gokul Chittaranjan, Yogarshi Vyas, Kalika Bali, and Monojit Choudhury, 2014. *Proceedings of the First Workshop on Computational Approaches to Code Switching*, chapter Word-level Language Identification using CRF: Code-switching Shared Task Report of MSR India System, pages 73–79. Association for Computational Linguistics.

Ryan Cotterell and Chris Callison-Burch. 2014. A Multi-Dialect, Multi-Genre Corpus of Informal Written Arabic. In *LREC*.

Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Nada AlMarwani, and Mohamed Al-Badrashiny. 2016. Creating a large multi-layered representational repository of linguistic code switched arabic data. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Heba Elfardy, Mohamed Al-Badrashiny, and Mona Diab, 2014. *AIDA: Identifying Code Switching in Informal Arabic Text*, chapter Proceedings of the First Workshop on Computational Approaches to Code Switching, pages 94–101. Association for Computational Linguistics.

Ramy Eskander, Mohamed Al-Badrashiny, Nizar Habash, and Owen Rambow. 2014. Foreign words and the automatic processing of arabic social media text written in roman script. *In Proceedings of the First Workshop on Computational Approaches to Code-Switching. EMNLP 2014, Conference on Empirical Methods in Natural Language Processing, October, 2014, Doha, Qatar*.

Naman Jain and Ahmad Riyaz Bhat, 2014. *Proceedings of the First Workshop on Computational Approaches to Code Switching*, chapter Language Identification in Code-Switching Scenario, pages 87–93. Association for Computational Linguistics.

Anupam Jamatia, Björn Gambäck, and Amitava Das. 2015. Part-of-speech tagging for code-mixed english-hindi twitter and facebook chat messages. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 239–248, Hissar, Bulgaria, September. INCOMA Ltd. Shoumen, BULGARIA.

Levi King, Eric Baucom, Timur Gilmanov, Sandra Kübler, Daniel Whyatt, Wolfgang Maier, and Paul Rodrigues. 2014. The iucl+ system: Word-level language identification via extended markov models. In *Proceedings of the First Workshop on Computational Approaches to Code-Switching*. EMNLP 2014, Conference on Empirical Methods in Natural Language Processing, Doha, Qatar.

LDC. 2003a. Arabic Gigaword Fifth Edition LDC2011T11. Linguistic Data Consortium.

LDC. 2003b. English Gigaword LDC2003T05. Linguistic Data Consortium.

LDC. 2009. Spanish Gigaword Second Edition LDC2009T21. Linguistic Data Consortium.

LDC. 2012. BOLT - Phase 1 Discussion Forums Source Data R4 LDC2012E54. Linguistic Data Consortium.

LDC. 2014a. BOLT Phase 2 SMS and Chat Arabic DevTest Data – Source Annotation, Transliteration and Translation. LDC catalog number LDC2014E28.

LDC. 2014b. BOLT Phase 2 SMS and Chat Arabic Training Data – Source Annotation, Transliteration and Translation R1. LDC catalog number LDC2014E48.

LDC. 2014c. BOLT Program: Romanized Arabic (Arabizi) to Arabic Transliteration and Normalization Guidelines. Version 3. Linguistic Data Consortium.

Chu-Cheng Lin, Waleed Ammar, Lori Levin, and Chris Dyer, 2014. *Proceedings of the First Workshop on Computational Approaches to Code Switching*, chapter The CMU Submission for the Shared Task on Language Identification in Code-Switched Data, pages 80–86. Association for Computational Linguistics.

Giovanni Molina, Fahad AlGhamdi, Mahmoud Ghoneim, Abdelati Hawwari, Nicolas Rey-Villamizar, Mona Diab, and Thamar Solorio. 2016. Overview for the second shared task on language identification in code-switched data. In *Proceedings of The EMNLP 2016 Second Workshop on Computational Approaches to Linguistic Code Switching (CALCS)*.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan M. Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *Proceedings of LREC*, Reykjavik, Iceland.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology-NAACL*, pages 213–220, Edmonton, Canada.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steve Bethard, Mona Diab, Mahmoud Gonheim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirshberg, Alison Chang, , and Pascale Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code-Switching. EMNLP 2014, Conference on Empirical Methods in Natural Language Processing, October, 2014, Doha, Qatar*.

Andreas Stolcke. 2002. Srilm an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*.

High Accuracy Rule-based Question Classification using Question Syntax and Semantics

Harish Tayyar Madabushi
School of Computer Science,
University of Birmingham,
Birmingham, United Kingdom.
H.T.Madabushi@cs.bham.ac.uk

Mark Lee
School of Computer Science,
University of Birmingham,
Birmingham, United Kingdom.
M.G.Lee@cs.bham.ac.uk

Abstract

We present in this paper a purely rule-based system for Question Classification which we divide into two parts: The first is the extraction of relevant words from a question by use of its structure, and the second is the classification of questions based on rules that associate these words to Concepts. We achieve an accuracy of *97.2%*, close to a 6 point improvement over the previous State of the Art of 91.6%. Additionally, we believe that machine learning algorithms can be applied on top of this method to further improve accuracy.

1 Introduction and Motivation

Question Answering (QA) is a task in Natural Language Processing (NLP) that requires the system to provide concise answers to Natural Language questions. Interest in QA has grown dramatically over the past couple of years, in part due to advances in NLP and Machine Learning, that have allowed for significant improvements in QA systems, and in part due to its increased accessibility to the general public via smart-phone applications such as Siri and Google Now.

An important element of QA is Question Classification (QC), which is the task of classifying a question based on the expected answer. As an example, the question "Who is the prime minister?" could be assigned the class "person", whereas the question "Where is the prime minister?" could belong to the class "location". Since the task involves identifying the type of answer, it is sometimes referred to as Answer Type Classification. While there do exist QA Systems that do not make use of QC, QC has been shown to significantly improve the performance of QA systems (Hovy et al., 2001).

A priori knowledge of the kind of information that a QA system is required to extract allows for the exploitation of predefined patterns and improved feature selection. For example, consider a QA system provided with the information that the question "How long is the term of office of the Prime Minister?" requires, as an answer, a "number that represents a duration". Such a system could dramatically reduce its search space, in that it could focus on numbers. The design of a QA system's search and information extraction components determines the classes that a QC should use. Despite this dependence on how question classes are used, there are some common question classes that are widely accepted as useful. The rules that govern and classes contained in a given question classification are based on the specific *Question Taxonomy* chosen.

2 Related Work

Work on QC, as in most NLP tasks, can be broadly divided into three categories: **a)** those that make use of machine learning, **b)** those that rely purely on rules, and **c)** those that are a hybrid of the two. With the increased popularity and success of machine learning techniques, most recent work on QC has been limited to methods that make use of machine learning. While there continues to be some exploration into semantic information contained in sentences, such information is often converted into features.

While there are several Question Taxonomies that are available for use in training and testing QC systems, the most popular is the one introduced by Li and Roth (2002). This is because of the 5,500

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1220–1230, Osaka, Japan, December 11-17 2016.

training questions and corresponding classification they provide, in addition to the classification of the 500 TREC 10 (Voorhees, 2001) questions. Their classification is a two level system which contains a coarse and a fine level of classification for each question. Table 1 lists the classification introduced by them. In this paper, we refer to a specific classes in the following format: coarse:fine. For example, the class animal, contained in the coarse class ENTY, will be referred to as enty:animal.

Coarse	Fine
ABBR	abbreviation, expansion
DESC	definition, description, manner, reason
ENTY	animal, body, color, creation, currency, disease, event, food, instrument, language, letter, other, plant, product, religion, sport, substance, symbol, technique, term, vehicle, word
HUM	description, group, individual, title
LOC	city, country, mountain, other, state
NUM	code, count, date, distance, money, order, other, percent, percent, period, speed, temperature, size, weight

Table 1: Question Taxonomy introduced by Li and Roth (2002).

The original method proposed by Li and Roth (2002), relies on machine learning and first classifies questions into coarse classes, before then using the coarse class as a feature in fine grained classification. They also report their results for both the coarse and fine classes. We, however, focus our efforts on fine grained classification.

Metzler and Croft (2005) provide a detailed analysis of statistical methods of QC prior to 2005, while dismissing rule-based systems as "cumbersome and inflexible", and a more recent survey by Loni (2011) details QC methods using more recent Machine Learning techniques. Work on QC over the last couple of years has involved either reducing the number of features (Pota et al., 2016; Pota et al., 2015), focusing on specific domains (Feng et al., 2015) or using new methods in machine learning such as Convolutional Neural Networks (Kim, 2014) and Skip-Thought Vectors (Kiros et al., 2015).

The previous State of the Art in fine grained classification of Li and Roth (2002)'s data is 91.6% and was achieved by Van-Tu and Anh-Cuong (2016), who base their work on using semantic features in a linear SVM. Of specific relevance to our work is the work by Silva et al. (2011), who first extract headwords, before then mapping these headwords into various categories using WordNet (Miller, 1995) to achieve an accuracy of 90.8%. Previous work by (Huang et al., 2008), which also makes use of both headwords and WordNet, while using slightly different methods, achieves an accuracy of 89.2%.

3 Concepts as a Theoretical Framework for Question Classification

Concepts are generalisations or abstractions that allow the use of previous experience in new situations. For example, questions such as "Who is the actor who ...?", of the form "Who $auxiliary\_verb$ $(determiner)^*$ *Concept:Occupation* who ...?", can be classified under the class hum:person, if we had information about the Concept "occupation", because this would enable us to map all questions that use any occupation in this particular pattern to this QC. Similarly, information about the Concept "meaning" would enable us to create a rule to classify questions such as "What is the meaning of the word ...?", and "What does the word ... mean?" to the question class desc:definition. As can be seen from the latter example, Concepts need not always be associated with nouns.

3.1 Implementing Concepts using Types

As described in the previous section, it is useful to define Concepts as sets of words and to this end, we require a method of generating a large number of words that belong to a particular Concept. To achieve this, we make use of Types (Tayyar Madabushi et al., 2016), which provide a way of defining sections of an ontology to belong to a given *Type*. While the authors use Types to identify classes of nouns that can be compared when measuring the semantic similarity between two sentences, we use Types to define Concepts. In this work, we modify the definition of types by making use of WordNet hyponyms: W_1 is considered a hyponym of W_2 if $\forall e \in W_1$, e is an instance of W_2.

A Type consists of a set of WordNet synsets or words S, and represents the set of words whose lemmas belongs to the union of the set S, and in the case of synsets, the set containing the hyponym closure of the synsets in S, and in the case of words, those words. As an example, all words whose lemmas belong to the hyponym closure of the synset 'occupation.n.01', such as bookkeeping, acting, and ministry belong to the the Type 'occupation.n.01'. It is interesting to note that this one definition provides us conceptual information on 283 lemmas (the size of the hyponym closure of occupation.n.01).

We use types to create a rough approximation of Concepts. We achieve this by manually picking specific synsets within WordNet and associating them and all their hyponyms to a particular QC based on where in a question they appear. Revisiting the first example in Section 3, the Concept "occupation" is defined by creating a Type that includes the word occupation and all hyponyms of the synset 'occupation.n.01'. Similarly, the synsets 'people.n.01', 'organization.n.01', 'university.n.01', 'company.n.04', 'social_group.n.01', and all of their hyponyms are assigned to the Question Class "Human Group". Some words, such as the word "mean" discussed in the second example in Section 3, belong to a particular Type while their hyponyms do not (in the case of "mean", "aim", "drive", and "spell" are hyponyms, which do not imply that a question belongs to the definition class the same way the word "mean" does), and in such cases, we add just the word and not its hyponyms.

The manual process of creating Types is done by looking at all hyponyms of the synset entity.n.01 and assigning them to a Type *iff* that synset and all its hyponyms represent the same Concept. This sometimes leads to instances wherein the same word is part of different Types because of its different word sense. In such cases, Types are redefined using less general synsets.

Not all of the Types we define are directly associated with a Question Class. For example, we define the Type $people\_from$, consisting of 'inhabitant.n.01' and all its hyponyms which enables us to identify the class enty:termeq (i.e. equivalent term). We do this by checking to see if the question asks us what people from a particular place call something, by use of the rule "What $auxiliary\_verb$ $people\_from$ call $word$?". As an example, the question "What do Italians call Noodles?" matches this rule and belongs to the QC enty:termeq. We also define groups of verbs as belonging to certain Types, such as the Type of verbs that can only be performed by a person (e.g. sing, invent) and the Type of words that require us to perform a possessive or a prepositional *roll* (Section 6.1).

4 System Overview

The system presented in this work consists of three parts: **a)** extracting a Question's Syntactic Map (defined in Section 5.1), **b)** identifying the headword, of the noun phrase in the question, while handling Entity Identification and phrase detection, and **c)** using rules to map words at different positions in the Syntactic Map to identify the QC. These are further broken down into the following steps (programmatically, methods):

Syntactic Map Extraction	*Question Rewrite*	Rewrites questions that are in non-standard form.
	Parse Tree Analysis	Extract structure information from the question using Constituency-based parse trees
Word, Phrase and Entity Extraction	*Headword Extraction*	Extract headwords from noun phrases in the question using **a)** Possessive Unrolling **b)** Preposition Rolling **c)** Entity Identification
	Verb, Wh-word and Adjective Extraction	Extract the Auxiliary and Major Verbs, the Wh-word and all adjectives from the question.
Rule-based Classification	*Match Rules based on the Question Syntax and Word Type*	Using a hierarchy of syntactic positions in a question, iteratively check to see if there exists a rule for mapping the word at that position to a QC.

For example, given the question "Name of actress from England in the movie 'The Titanic' is what?", our system identifies its QC as follows: We first identify that this question is not in a form that we can analyse to extract the Syntactic Map and rewrite it as "What is the name of the actress from England in the movie 'The Titanic'?" (Section 5.2). The question's parse tree is then analysed to generate the Question's Syntactic Map (Section 5.1). We then identify the headword to be the noun *actress* using prepositional

rolling (Section 6.1). At this stage, we have established that the question's wh-word is "What", auxiliary verb is "is", and headword is "actress". We check for the existence of a rule that classifies this question by iterating through these elements in a predefined order (Section 7.2). This results in the word "actress" matching the rule : *'occupation.n.01' and its hyponyms in SQ-NNP when the wh-word is 'what' indicate that the question class is* ***hum:ind***, so enabling us to classify the question as hum:ind.

4.1 Methodology

To avoid bias, we use the 5,500 questions and their respective question classes provided as training data by Li and Roth (2002) for exploration and rule discovery, and ensure that the 500 TREC questions, which consist of the test set, are not observed during the creation of rules (although the system is, at regular intervals, tested on this set to ensure progress). Once we complete the analysis of a question's parse tree, not all words in the question are of further relevance to the task of QC. However, so as to maximise the number of words that we have rules for, we try to create rules for all words that appear in training set.

5 Syntactic Maps

Previous work that has made use of parse trees includes that by Silva et al. (2011), who used Collin's Rules (Collins, 1999) to extract headwords and work by Shen and Lapata (2007) who made use of FrameNet (Baker et al., 1998). Unlike these works, we first extract, what we call, a Question's *Syntactic Map*, before creating rules that depend on the position of words in this Map.

A Syntactic Map (SM), unlike a parse tree, is a *fixed* structure that we fill in with information from a question's parse tree and can contain empty or "None" elements. It is a generic template for all the different kinds of questions that we can classify, and any question that we cannot convert to a Syntactic Map, cannot be classified using our system. Crucially, the SM contains the following five elements of a question: **a)** the question's wh-word **b)** the noun phrase (if any) contained in the WHNP sub-tree and its internal phrase structure, and from the SQ sub-tree of the parse tree: **c)** the Auxiliary Verb (AVP) **d)** the noun phrase (if any) and its internal phrase structure, and **e)** the Main Verb (MVP) (if any). Noun phrases including possessives, and prepositional phrases are extracted into similar fixed structures. Programmatically, a SM is a class (object-oriented programming), as are the constituent noun phrases, prepositional phrases, and verbs. The generic structure of a SM, along with the structure of its constituents is shown in Table 2.

Syntactic Map		
WH Word		What/Name/Who/...
WHNP		
	NNP	Noun Phrase in WHNP
SQ		
	AVP	Axillary Verb of SQ
	NNP	Noun Phrase in SQ
	MVP	First Main Verb of SQ

Constituent Noun Phrase		Constituent Prepositional Phrase		Constituent Verb	
JJ	Adjective	PP	Prepositional word		
NN	Noun	NN	Attached Noun Phrase		
PRP	Preposition	VP	Attached Verb Phrase	VB	Verb
POS	Possessive	CPP	Attached Prepositional Phrase		
TJJ	Trailing Adjective				

Table 2: The fixed structure of a Syntactic Map (left), and the constituent phrase structures (right).

In the question "How much does the President get paid ?", it is the adverb "much" that allows us to infer that the expected answer is a number and additionally, the word "paid" allows us to infer that the number, in fact, represents money hence resulting in the question class num:money.

In the questions "What is a golf ball made of ?" and "What does gringo mean ?" the verbs after the noun (the first Main Verb or MVP) provide us with important clues on which question class these questions belong to (in this case enty:substance and desc:def). It is for this reason that we move beyond conventional headword extraction and focus on populating Syntactic Maps, which capture more information about the question. Although Silva et al. (2011) consider words other than nouns, they do so only when the questions contain certain exact phrases.

5.1 Syntactic Map Extraction

The first step in SM extraction is the extraction of the "WHNP" and "SQ" sections of a question from its constituent parse tree, which we generating using the Stanford CoreNLP toolkit Manning et al. (2014).

The WHNP sub-tree represents the Wh-noun Phrase and the SQ sub-tree the main clause of a wh-question. In cases where there is neither (e.g. Name the highest mountain.), we use the first noun phrase as the SQ sub-tree. From the WHNP and the SQ sections of the parse tree, we extract the various elements of the SM as shown in Table 2. This requires the parsing of noun, prepositional, possessive and verb phrases. Due to space constraints, we only provide an overview of each of these below. Additionally, extracting each of these elements is done recursively as sentences often contain possessive phrases or prepositional phrases within one another. Table 3 illustrates one such scenario in which a question has two recursive possessive phrases.

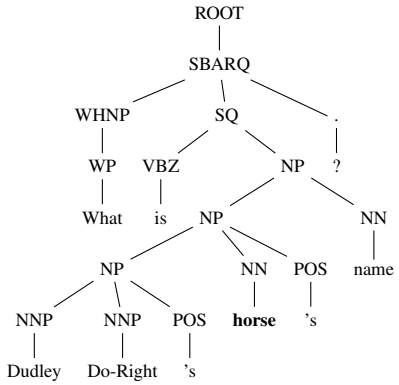
Parse Tree	Extracted Structure
	WH Word: What WHNP NNP: None SQ AVP: ['is'] NNP: (Possessive)Dudley Do-Right (Possessive)horse name MVP

Table 3: The Parse Tree and Extracted SM of a Question Consisting of a Nested Structure.

We make the conscious decision of stopping the SM extraction process after reaching the first main verb. This is because we observed that there were very few questions that require structural information beyond this point.

Our method of analysing noun phrases handles the extraction of adjectives, possessive phrases, prepositions and trailing adjectives but ignores all determiners. Prior to analysing parse trees of noun phrases, we first modify certain parse tree patterns that noun phrases occur in. The resultant Constituency-based parse trees are not always valid but greatly simplify the analysis of noun phrases. Two examples of the modifications we perform to noun phrase sub-trees are illustrated in Table 4

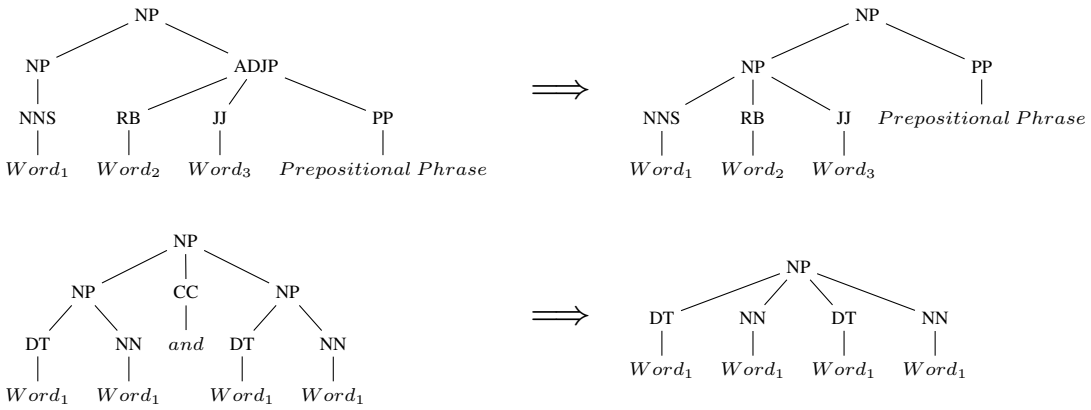


Table 4: Some of the Parse Tree Modifications that are Performed on Noun Phrases.

This simplification process leaves us with the task of extracting information from noun phrases that belong to a much smaller set of sub-tree patterns. Some of the more common noun phrase patterns are illustrated in Table 5. Possessive phrases are treated as nouns that must have, attached to them, yet another noun. When we identify a preposition phrase or a verb phrase, that sub-tree is passed to either the preposition or verb analysis method respectively.

Similarly, we extract information from prepositional sub-trees based on their structure, which nearly

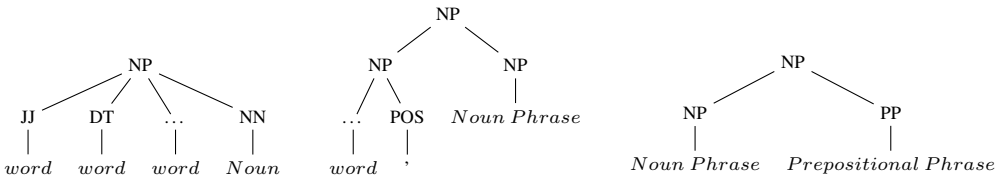


Table 5: Some Common Sub-tree Patterns that Noun Phrases occur in.

always belong to one of the following three patterns: A preposition phrase with one child that is the preposition and the other that is one of either a noun phrase, verb phrase or another prepositional phrase (e.g. "name of the prime minister of U.K."). These patterns are illustrated in Table 6. Just as in the case of noun phrases, we pass on any sub-trees of phrases that are of a different kind to the appropriate analysis module, which enables us to generate a recursive SM.

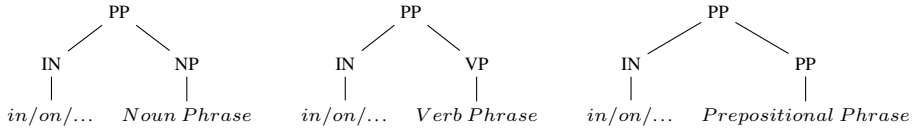


Table 6: Some Common Sub-Tree Patterns that Prepositional Phrases occur in.

5.2 Question Rewrites

There are some questions that do not belong to the standard structure of questions such as "A corgi is a kind of what?" and "In 139 the papal court was forced to move from Rome to where?". We identify several of these structures and create rewrite rules (e.g *x is/was y in/of what z?*) to rewrite these questions to a form that we can parse. We use regular expressions instead of parse tree analysis as these structures are very easy to identify and so the overhead of parsing is not justified. Using these rules the above two questions will be rewritten as "What is a corgi a kind of?" and "To where was the papal court forced to move from Rome in 139?".

6 Concept Identification

In this section, we provide details on methods we use for identifying relevant Concepts, which we extract by analysing the SM.

6.1 Preposition Rolling and Possessive Unrolling

Rolling and Unrolling refer to the selective moving forward through a preposition, or backwards through a possessive noun. Consider the question "What is the quantity of American soldiers still unaccounted for from the Vietnam war?" from which we extract *quantity(PP) of PP-NN:(JJ)American soldiers*, and the question "What are the different types of plastic?" from which we extract *(JJ)different types(PP) of PP-NN: plastic*. In the second instance, we must *roll* through the preposition to reach the relevant word "plastic", whereas, in the first instance, we must not, so identifying "quantity'.

Similarly, consider the question "What game's board shows the territories of Irkutsk, Yakutsk and Kamchatka?" from which we extract the noun phrase *(Possessive)game board*, and the question "Name Alvin's brothers." from which we extract *(Possessive)Alvin brothers*. In the first instance we need to *unroll* through the possessive to reach the relevant word "game", whereas in the second case we must not. We call this selective process of moving forward through a preposition "Rolling", and the process of selectively moving backwards through a possessive "Unrolling". Rolling and Unrolling are achieved through a list of rules that depend on the Type of the target and source of the Roll or Unroll.

6.2 Headword and Phrase Extraction

Consider the question "What mystery writer penned '...the glory that was Greece, and the grandeur that was Rome'?". The relevant noun phrase that we extract from the SM is "mystery writer" and the head of

this noun phrase is "writer", the last noun in the noun phrase. This is often the case, and some previous works have used only this to identify the head of a noun phrase (Metzler and Croft, 2005). Unfortunately, this is not always the case, and does not always provide the word that is most useful for QC. For example, the noun phrase extracted from "What crop failure caused the Irish Famine?" is "crop failure" and the relevant noun is "crop". Although it can be argued that the head noun in this phrase is "failure", qualified by "crop", this would not aid us in classification, as "crops" are a form of food and the expected Question Class is enty:food, while "failure" is a very different Concept.

We automatically identifying the head noun by identifying *Verb Nouns* and *Descriptive Nouns* starting at the right of the noun phrase and ignoring such nouns. We define Verb Nouns as nouns that have a more common verb form (e.g. fail) or verbs that are "acts", which we identify by parsing the definition of the verb. Similarly, we define Descriptive Nouns as nouns that belong to a Type we define as descriptive which includes, for example, hyponyms of the synyset 'digit.n.01'.

6.3 Entity Identification

Let us now consider the question "What is bipolar disorder?". The correct Question Class for this question is desc:definition, however, it is easy to miss-classify this question as belonging to the class enty:dismed (entity, disease or medicine), because the word "bipolar" is tagged as an adjective. To get around this we require a method of identifying that "bipolar disorder" must be considered as a single entity.

Even in instances wherein it is relatively easy to identify an entity, as in the case of phrases that consist of consecutive nouns, it is important to be able to convert these phrases to a form that appears in WordNet. For example, the phrase "equity securities" can be identified as a single entity, however, it is listed in WordNet under the entry "Shares".

We identify these phrases using a method called Wikification (Mihalcea and Csomai, 2007), which is the process of linking words and phrases in a piece of text to titles of Wikipedia entries. The intuition behind this is that a phrase that appears as a Wikipedia Article title must be important enough to be considered as a single Entity. We base our method of Wikification on the original, while replacing the process of keyword identification with SM and that of Word Sense Disambiguation with the method detailed in Section 7.1. For example, there is an article on Wikipedia titled "Bipolar Disorder" on Wikipedia and the Wikified term for "equity securities" is "Shares".

7 Question Classification using Syntactic Maps

Once we have the SM of a question, we use rules to identify the relevant QC. However, before we can match appropriate words, we require a way of identifying the correct sense of a word.

7.1 Word Sense Disambiguation

SMs often provide us with a single word that represents the object that the question expects as an answer. The question "What album put The Beatles on the cover of Time in 1967 ?", for example, requires that the answer consists of an "album". However, it is unclear whether album refers to "one or more recordings issued together" or "a book of blank pages with pockets or envelopes". Huang et al. (2008) address this problem by use of the Lesk Algorithm (Lesk, 1986).

Our use of SM allows for implicit Word Sense Disambiguation as it is rare for the same word to appear at the same syntactic location but in different senses. When this does happen however, we identify the sense of a word based on the Types of the surrounding elements of the SM. For example, "How much does it cost to fly to Japan?" and "How much does a plane weigh?" both have the word "much" at the same position and so require us to identify the Types of associated words (i.e. "cost" and "weigh") to be able to disambiguate the relevant Concept.

7.2 Mapping Question Classes

The intuition behind the mapping process is that words or phrases at certain positions in the SM trigger certain Concepts, which gives away the question class. To this end, we use Types defined for each different position in the SM to map questions to question classes. For example, the word "do" appearing as the

Data: Syntactic Map, Type Definitions, Classes associated with Type Definitions.
Result: Question Class

if *Preposition Rolling Possible* **then**
    Perform Preposition Roll
if *Possessive Unrolling Possible* **then**
    Perform Possessive Unroll
Initialise $head\_noun\_class$ to None ; /* $head\_noun\_class$ is a Tuple Consisting of the Major and Minor Question Type */
$head\_noun \leftarrow$ Extract Head Noun from Syntactic Map ;
$head\_noun\_adjectives \leftarrow$ Extract Head Noun adjectives from Syntactic Map ;
for *reversed(* $head\_noun\_adjectives$ *)* **do**
    if $adjective$ *has Type Defined* **then**
        $head\_noun\_class \leftarrow$ Class associated with Type;
if $head\_noun\_class$ *is None* **then**
    if $head\_noun$ *has Type Defined* **then**
        $head\_noun\_class \leftarrow$ Class associated with Type;

if $head\_noun\_class[0]$ == *"ABBR"* **then**
    if $head\_noun$ *is an Abbreviation* **then**
        return *('ABBR', 'exp')*
    return $head\_noun\_class$

if *All of the following elements in the Syntactic Map are Empty: WHNP-NNP, SQ-MVP,* $head\_noun\_adjectives$ **then**
    if *There has been no Rolling or Unrolling* **then**
        if *AVP is one of "is", "are", "was", "were"* **then**
            if *WH_Word is "What"* **then**
                return *('DESC', 'def')*
            if *WH_Word is "Who"* **then**
                return *('HUM', 'desc')*

for *reversed(* $head\_noun\_adjectives$ *)* **do**
    if $adjective$ *has WSD Type Defined* **then**
        return *Class associated with WSD Type;*

$wh\_word \leftarrow$ Extract What Word from Syntactic Map ;
if wh_word == *"define"* **then**
    if $head\_noun\_class[0]$ == *"DESC"* **then**
        return $head\_noun\_class$
    return *("DESC", "def")*
if wh_word == *"how"* **then**
    if $head\_noun\_class[0]$ == *"DESC"* **then**
        return $head\_noun\_class$
    return *("DESC", "manner")*
/* Similar restrictions are imposed on other possible wh_words (i.e. ``where'', ``whose'', ``describe'', ``when'', ``why'', ``name'', and ``what'') */
$main\_verb \leftarrow$ Extract Main Verb from Syntactic Map ;
$auxiliary\_verb \leftarrow$ Extract Auxiliary Verb from Syntactic Map ;
for *verb in [main_verb, auxiliary_verb* **do**
    if *verb has Type Defined* **then**
        return *Class associated with Type;*
    if *verb has WSD Type Defined* **then**
        return *Class associated with WSD Type;*

if $head\_noun\_class$ *is None* **then**
    return *("ENTY", "other")*
return $head\_noun\_class$

Algorithm 1: A Simplified Algorithm showing the Mapping of the Syntactic Map to Question Classes

auxiliary verb is handled differently from when it appears as the main verb in the SM. The order in which different sections of the SM are considered determines which word is finally used during classification.

There are some special words, such as "much", "do", "name" and "call", that require more complex classification rules. The adjective "much" for example could indicate the class num:money or num:weight depending on whether the other sections of the SM contain the Type "money" or the Type "weight". As in the case of WSD, we define disambiguation rules for each such word.

Algorithm 1, while not exhaustive in listing the mapping rules (due to space constraints), provides a simplified overview of the mapping of Semantic Maps to Question Classes. It takes as input the SM,

the Type definitions and associated Question Classes and returns a tuple consisting of the Major and Minor question classes. Just over 230 Type definitions and 10 special Word Sense Disambiguation definitions cover the entire test set, and at the time of writing, these have been expanded to around 600 Type definitions and 70 WSD definitions.

8 Results

We achieve an accuracy of 97.2% on the TREC 10 dataset which translates to an incorrect tagging of 14 of the 500 questions in the dataset. This is close to a 6 point improvement over the previous state of the art of 91.6% (Van-Tu and Anh-Cuong, 2016). We list our accuracy against that of various other works that have reported results on the TREC 10 dataset in Table 7.

Study	Classifier	Accuracy	
		Coarse	Fine
This Work	**None**	**-**	**97.2%**
Van-Tu and Anh-Cuong (2016)	Linear SVM	95.2%	91.6%
Pota et al. (2016; Pota et al. (2015)	Linear SVM	89.6%	82.0%
Kim (2014)	Convolutional Neural Networks	93.6%	-
Kiros et al. (2015)	Skip-Thought Vectors	91.8%	-
Silva et al. (2011)	Linear SVM	95.0%	90.8%
Loni et al. (2011)	Linear SVM	93.6%	89.0%
Merkel and Klakow (2007)	Language Modelling	-	80.8%
Li and Roth (2006)	SNoW	-	89.3%
Li and Roth (2002)	SNoW	91.0%	84.2%

Table 7: Results Achieved by this Work alongside some other Works that use the same Dataset.

8.1 Error Analysis

Table 8 provides a list of some of the questions that we misclassify along with the reason for this. One of the advantages of a purely rule-based system is the ability to pinpoint the exact reason for an incorrect classification.

Question	**Correct Class**	**Classified As**	**Reason**
What are the twin cities?	LOC city	DESC def	We classify both these as definitions because we (correctly) identify "twin cities" and "speed of light" as entities. The presence of the word "the" however requires information about the entity instead of a definition for the entity - a rule that requires to be added.
What is the speed of light?	NUM speed	DESC def	
What is compounded interest?	DESC def	DESC desc	Our Wikification system fails to identify "compounded interest" to be the same as the entity "compound interest".
What is the spirometer test?	DESC def	ENTY instru	The word "test", has a natural verb form so forcing the system to identify "spirometer" as the head noun. Some modifications to the function identifying Verb Nouns are required to rectify this.

Table 8: An analysis of some of the questions that we fail to classify correctly.

9 Conclusion and Future Work

We presented a purely rule-based system for QC which exploits decades of research into the structure of language and Concepts. Although this method has focused on a particular type of questions, we believe that a similar method can be applied to classifying questions of a different type, and we intend to extend our work to include those datasets. We also note that these are a common and important kind of questions, which are similar to those handled by most modern smartphone interactive systems such as Google Now (Ristovski, 2016).

Finally, we intend to implement a QA system that leverages QC to explore the true impact of high-accuracy question classification. We also intend to make this system available through a simple Application Programming Interface (API) [1] so other QA systems can benefit from this work.

[1] API available at: `http://www.harishmadabushi.com/research/questionclassification/`

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 86–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing.* Ph.D. thesis, University. of Pennsylvania.

Guangyu Feng, Kun Xiong, Yang Tang, Anqi Cui, Jing Bai, Hang Li, Qiang Yang, and Ming Li. 2015. Question classification by approximating semantics. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 407–417, New York, NY, USA. ACM.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhiheng Huang, Marcus Thint, and Zengchang Qin. 2008. Question classification using head words and their hypernyms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 927–936, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *CoRR*, abs/1506.06726.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, pages 24–26, New York, NY, USA. ACM.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xin Li and Dan Roth. 2006. Learning question classifiers: The role of semantic information. *Nat. Lang. Eng.*, 12(3):229–249, September.

Babak Loni, Gijs van Tulder, Pascal Wiggers, David M. J. Tax, and Marco Loog, 2011. *Question Classification by Weighted Combination of Lexical, Syntactic and Semantic Features*, pages 243–250. Springer Berlin Heidelberg, Berlin, Heidelberg.

Babak Loni. 2011. A survey of state-of-the-art methods on question classification. journal article uuid:8e57caa8-04fc-4fe2-b668-20767ab3db92, Delft University of Technology, Mekelweg 2, 2628 CD Delft, Netherlands, June.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Andreas Merkel and Dietrich Klakow. 2007. Improved methods for language model based question classification. In *INTERSPEECH*, pages 322–325.

Donald Metzler and W. Bruce Croft. 2005. Analysis of statistical question classification for fact-based questions. *Information Retrieval*, 8(3):481–504.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 233–242, New York, NY, USA. ACM.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November.

M. Pota, A. Fuggi, M. Esposito, and G. D. Pietro. 2015. Extracting compact sets of features for question classification in cognitive systems: A comparative study. In *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pages 551–556, November.

Marco Pota, Massimo Esposito, and Giuseppe De Pietro, 2016. *A Forward-Selection Algorithm for SVM-Based Question Classification in Cognitive Systems*, pages 587–598. Springer International Publishing, Cham.

Kristijan Ristovski. 2016. A complete list of google now commands. *http://ok-google.io/*. Retrieved: June 2016.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, page 12–21.

João Silva, Luísa Coheur, Ana Cristina Mendes, and Andreas Wichert. 2011. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154.

Harish Tayyar Madabushi, Mark Buhagiar, and Mark Lee. 2016. UoB-UK at SemEval-2016 Task 1: A Flexible and Extendable System for Semantic Text Similarity using Types, Surprise and Phrase Linking. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 680–685, San Diego, California, June. Association for Computational Linguistics.

Nguyen Van-Tu and Le Anh-Cuong. 2016. Improving question classification by feature extraction and selection. *Indian Journal of Science and Technology*, 9(17).

Ellen M. Voorhees. 2001. Question answering in TREC. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, pages 535–537, New York, NY, USA. ACM.

Incorporating Label Dependency for Answer Quality Tagging in Community Question Answering via CNN-LSTM-CRF

Yang Xiang, Xiaoqiang Zhou[1], Qingcai Chen[2], Zhihui Zheng, Buzhou Tang, Xiaolong Wang, and Yang Qin
Intelligent Computation Research Center,
Harbin Institute of Technology Shenzhen Graduate School, China
{xiangyang.hitsz, xiaoqiang.jeseph,qingcai.chen,
zhihui.zchina,tangbuzhou}@gmail.com
wangxl@insun.hit.edu.cn, csyqin@hitsz.edu.cn

Abstract

In community question answering (cQA), the quality of answers are determined by the matching degree between question-answer pairs and the correlation among the answers. In this paper, we show that the dependency between the answer quality labels also plays a pivotal role. To validate the effectiveness of label dependency, we propose two neural network-based models, with different combination modes of Convolutional Neural Networks, Long Short Term Memory and Conditional Random Fields. Extensive experiments are taken on the dataset released by the SemEval-2015 cQA shared task. The first model is a stacked ensemble of the networks. It achieves 58.96% on macro averaged F_1, which improves the state-of-the-art neural network-based method by 2.82% and outperforms the Top-1 system in the shared task by 1.77%. The second is a simple attention-based model whose input is the connection of the question and its corresponding answers. It produces promising results with 58.29% on overall F_1 and gains the best performance on the *Good* and *Bad* categories.

1 Introduction

Community question answering (cQA) provide abundant human-to-human questions and answers, behaving as a good resource for building automatic question answering systems. For example, many users ask about "How to build your body like a model?" in Yahoo Answers[3] and also there are many answerers who are willing to share their experiences. Answer quality tagging refers to automatically identifying whether an answer is good, so as to collect high quality question answer pairs.

The task is challenging in that one should develop useful features that can effectively bridge the semantic gap between the question and answer (QA) pair. The matching degree on content is the primary factor that determines how good an answer is. It can be measured through a series of syntactic and semantic features such as overlapped linguistic elements (Berger et al., 2000; Agichtein et al., 2008; Punyakanok et al., 2004). For example, if the question and answer have several words shared, the answer is likely to be good. However, the content similarity performs well only when both the question and answer are well-structured and overlapped, which are rare to see in cQA sentences.

Many previous studies discover specific features based on the characteristics of cQA systems. For instance, opinion leader or experts would probably contribute more good answers than bad, and the number of thumbs-up towards an answer may reflect how much the users accept it (Agichtein et al.,

[1] The first two authors have equal contributions.
[2] Corresponding author
[3] https://answers.yahoo.com/question/index?qid=20070918225025AA5Jz0G

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1231–1241, Osaka, Japan, December 11-17 2016.

2008). The inefficiency of textual similarity can be partly complemented with the help of these specific features (Hu et al., 2013; Tran et al., 2015). But the definition and validation of them need too much laborious cost and might not be appropriate when transferring to newly built systems.

In this paper, we focus on a novel concept of contextual feature, the *label dependency*, which we think plays a pivotal role when predicting the answer quality in cQA. Intuitively, being a rational answerer, when we decide to answer a question, we face the following situations:

- If it is a new question (without an answer), just answer it.
- If it has been answered but all the existed answers are bad, we provide our good answer.
- If it has been answered and some of the existed answers have already satisfied the asker's information needs, we can choose to answer it from a distinct perspective.
- Other actions.

A typical example for cQA is shown in Figure 1 in which A4 and A5 follow several bad or potentially good comments. Although A4 and A5 are both good answers, they provide alternative useful information. Based on the above assumptions, we say that there are possible soft constraints between the quality tags of the answers (i.e. a good answer follows several bad ones). The contextual dependency relation can be affected by the relations among the answer contents as well as the interactions between answer quality tags.

> **Q**: Where can i buy globe roam sim here in qatar?Can anyone tell me where can i buy globe roaming sim? thanks! if your selling globe roaming sim.
>
> **A1**: vivo bonito, did you just cut and paste that from the Globe Website? (Bad)
>
> **A2**: i am not working to any either of the smart or globe.. and that was my opinion when i bought one family sim pack... to where, at my disappointments.. were not all true upon on the run...(sigh)hell network, a misguiding false adverts. (Bad)
>
> **A3**: i heard that globe's signal is not good here and besides calling thru roaming will be more expensive. anyway, in the souq ull find sim roaming for smart, dont know about globe. (Potential)
>
> **A4**: you can go to filipino souq you cab get it for QR 25. (Good)
>
> **A5**: ... available at designated retail outlets in the Philippines. u would easily find it while processing some of ur papers at POEA Ortigas. (Good)

Figure 1: Example for cQA thread.

To efficiently model the contextual information, we propose two neural network-based models with different combination modes of Convolutional Neural Networks (CNN), Long Short Term Memory (LSTM) and Conditional Random Fields (CRF). The first model (ARC-I) is a stacked ensemble of the above networks, which can be seen as a combination of RCNN (Zhou et al., 2015) and LSTM-CRF (Huang et al., 2015). In ARC-I, LSTM is applied on the sequence of encoded QA matching pairs, with CRF on the final layer to memorize transition probabilities over the tag sequence. The main improvement from Zhou et al. (2015) is the addition of backward LSTM and CRF. And the difference from Huang et al. (2015) is that we adopt the LSTM-CRF model in comment-level (actually sentence-level within this paper) sequence tagging, with the help of CNN sentence modelling (Kim, 2014).

ARC-II is a novel and much simpler model, with the integration of the attention mechanism. In ARC-II, the question and its answers are linearly connected in a sequence and encoded by CNN. An attention-based LSTM is then applied on the encoded sequence. Through attention, the model learns how much the question and the context affect the predicting of the current answer. And similar to ARC-I, a CRF layer is appended at last. Using a simple attention function (described in Section 3), ARC-II reduces the size of the parameter space and trains faster than ARC-I.

We carried out extensive experiments on the dataset released by the SemEval-2015 cQA shared task. By adding bidirectional LSTM (Bi-LSTM) and CRF, we achieve 58.96% on macro averaged F_1 (by ARC-I) for answer quality tagging, which improves the state-of-the-art neural network-based method by 2.82% and outperforms the task winner by 1.77%. ARC-II produces promising results with 58.29% on overall F_1 and gains the best performance on the *Good* and *Bad* categories. The contributions of this

work can be summarized as: 1) experiments show that encoding the label dependency is powerful in answer quality tagging; 2) the proposed models achieve state-of-the-art result and considerable improvements on individual categories; 3) as far as we know, this is the first work using the LSTM-CRF architecture on sentence-level sequence tagging.

The roadmap of this paper is: in Section 2 we briefly introduce the literature. The proposed models are detailedly described in Section 3. Experiments are arranged in Section 4. Discussion and conclusion are at last.

2 Related Work

2.1 Answer Quality Tagging

In the literature, methods for answer quality tagging in cQA can be roughly divided into the following four groups: sparse feature-based methods, translation models, parsing trees, and deep CNNs. Sparse feature-based methods are the mostly widely applied and have the longest research duration. Early studies such as Agichtein et al. (2008) and Suryanoto et al. (2009), and later studies such as Yih et al. (2013) and Hou et al. (2015) all achieved not bad results using simple classifiers with manually constructed sparse features. However, feature engineering is time consuming and has low extensibility to other domains. Translation models relied on large-scale of training pairs and can effectively bridge the semantic gap in many cases (Berger et al., 2000; Riezler et al., 2007; Surdeanu et al., 2008). One difficulty is that large parallel training dataset is hard to obtain. The similarity computed from parsing trees is a direct criterion to measure the semantic correlation between two sentences. Typical works are tree edit distance (Punyakanok et al., 2004; Yao et al., 2013) and convolutional tree kernels (Severyn and Moschitti, 2013). But one of the drawbacks of parsing tree-based methods is that most existed parsers perform badly in low-quality sentences (i.e. spoken style language in cQA or daily dialogues).

In the research field of deep learning, Deep Belief Networks (DBN) can also be classified into sparse feature-based methods since they mainly incorporated sparse encodings for multiple features (Wang et al., 2010). The issue of discontinuous word features are later tackled by CNN (Kim, 2014; Hu et al., 2014), following some applications for answer selection in cQA (Yu et al., 2014; Qiu and Huang, 2015; Shen et al., 2015). Zhou et al. (2015)'s work modelled the content correlations using LSTM along QA sequences but they ignored the constraints among quality tags. Their method is insufficient in context learning also due to the neglect of sequence-level dependency modelling. Joty et al. (2015) proposed a graph-cut approach on the judgement of good or bad answers and gains improvement from the baselines. Joty et al. (2016) is an improved version of Joty et al. (2015) in which the authors introduced jointly learning approaches to capture global dependency. The above two works validated the importance of label dependency but they divided the tagging task into two subtasks and built their models in a distinct way.

2.2 Combining Deep Neural Networks and Graphical Models

Wöllmer et al. (2011) was one of the earliest studies that combine neural networks and graphical models. They appended an LSTM layer on top of a Hidden Markov Model in automatic speech recognition. However, the LSTM they applied was only a shallow architecture. Only in recent two or three years did some researchers begin to explore the combination of deep neural networks and graphical models. Huang et al. (2015) proposed the LSTM-CRF model for POS, chunking and NER, and produced state-of-the-art (or close to) accuracies. Lample et al. (2016) applied character and word embeddings in LSTM-CRF and generated good results on NER for four languages. Ma et al. (2016) added a CNN layer on word and character embedding and outperformed previous works in POS tagging and NER.

As far as we know, all the related studies were settled on word-level tasks. In most cases, sentence-level sequence labelling is quite distinct from the word-level in that the dependencies between adjacent sentences are not hard. So that the constraints for tags are weak than word-level tasks, demanding additional information to reinforce the constraints (i.e. sentence meanings).

3 Approach

3.1 ARC-I

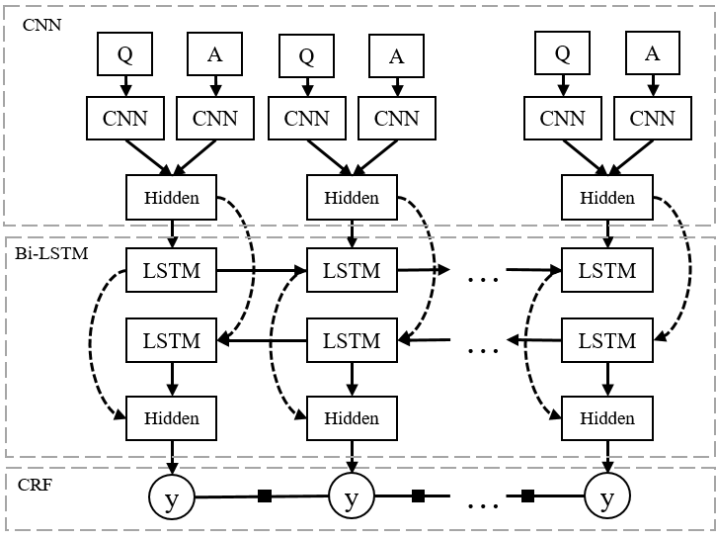


Figure 2: The overview of ARC-I.

The overview of ARC-I is shown in Figure 2. From input to output, the stack of networks are CNN, Bi-LSTM, and CRF. In more detail, in the CNN layer, each QA pair is encoded into a fixed length vector by parallel CNNs together with a following fully connected layer (denoted as *Hidden* in the figures). A Bi-LSTM layer is put next to learn the correlations along the encoded sequence. A fully connected layer and softmax are appended later to generated the predicted tags for the neural network. And at last the cost of the whole network is adjusted using the transition probabilities by a CRF layer over the generated tags.

We incorporate forward linear chain CRF. It jointly models the generative probability (from the output of Bi-LSTM) and the transition probability between adjacent tags from a global perspective, and thus automatically learns the content correlation and constraints among labels.

3.2 ARC-II

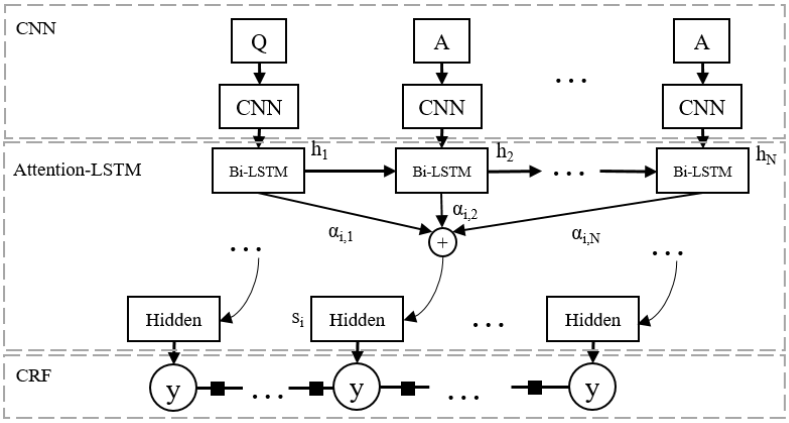


Figure 3: The overview of ARC-II. $\alpha_{i,j}$ stands for the attention weight of the *i*th unit focused on the *j*th encoded element.

Encoding the sequence of QA pairs may introduce extra parameters, the optimization of which can cause much training time. A concise way is to directly learn from the sequence composed of the question and its answers. The model is depicted in Figure 3. In the CNN layer, each question/answer is encoded using a single CNN. In the LSTM layer, an attention based Bi-LSTM is applied to learn the context information along the sequence. Attention mechanism can tackle the bias problem of RNNs (i.e. LSTM/GRU) through computing a weighted distribution of the encoded elements at each time step (Bahdanau et al.,

2015). The distribution reflects the correlations between the current answer and its context. Similar to ARC-I, we also put a CRF layer at last to learn the transitions. To simplify the network, we used a simple form of attention: the attention vector is compromised by the similarities between the current answer and the contextual sentences (the question and the answers).

There are also some variants based on the proposed architectures. By removing backward LSTM or CRF, we validated the contribution of each module to the final result. By replacing CRF with the previously predicted label (denoted as *LP* in the experiment), we tested the superiority of modelling the tag sequence softly over hard encoding. In the following subsections, we explain each applied module in detail.

3.3 CNN

We did not explore deep on sentence modelling such as tuning the depth or dimensions of the nodes, instead we simply adopt the architecture from Kim (2014). The input for CNN is the distributed representation of a sentence, by mapping each word index into its embeddings which were pre-trained in an unsupervised way. Each question or answer is taken as a sentence but not a paragraph due to its length limit (<100 words) in the dataset. To simplify calculation, each sentence is padded to the same length *n* with zero vectors.

We denote the *k*-dimensional embedding for word *j* in a sentence as $z_j \in \mathbb{R}^k$, and thus the sentence can be represented by:

$$z_{1:n} = z_1 \oplus z_2 \oplus ... \oplus z_n \quad (1)$$

where $\oplus$ is the concatenation operator and $z_{i:h}$ stands for $\{z_i, z_{i+1}, \ldots, z_{i+h-1}\}$, i=0,1,…,*n*-*h*+1. The basic CNN operations for sentence modelling include convolution and max-pooling. Convolution is achieved by applying a fixed length sliding window (filter) $w^m \in \mathbb{R}^{h \times k}$ on each word position *i*, such that *n*-*h*+1 convolutional units are generated by:

$$c_i^m = \sigma(w^m \cdot z_{i:i+h} + b^m),\ i = 0,1,...,n-h+1 \quad (2)$$

where σ is the activation function such as Sigmoid or ReLU (Dahl et al., 2013) and b^m is the bias factor for the *m*th layer.

Max-pooling is more popular than mean-pooling in sentence modelling due to the characteristics of natural language. A *d*-max-pooling is to select the maximum unit in every adjacent *d* convolutional units.

$$c_i^m = max(c_j^m, c_{j+1}^m, ..., c_{j+d-1}^m),\ j = 0, d, 2d, ... \quad (3)$$

Following Kim (2014), we applied three convolution filters with lengths *3*, *4* and *5* (with the embedding dimension as the width of the filter), each of which is followed by a max pooling layer to select the most effective structures. Feature maps of size 100 are applied to learn the representations from multiple perspectives.The flattened output vectors for each filter are concatenated as the output of the CNN layer.

3.4 LSTM

LSTM can learn long-distance dependencies through the additions of the *input gate*, *forget gate*, *output gate* and *memory cell* into each recurrent unit. The architecture employed in this paper is analogous to the one introduced by Graves et al. (2013). The input to each LSTM unit is x_t and the output is h_t. The output can be computed through Equations 4-8.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad (4)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \quad (5)$$

$$c_t = f_t c_{t-1} + i_t \tau(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (6)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad (7)$$

$$h_t = o_t \theta(c_t) \quad (8)$$

where τ and θ are usually set as the *tanh* function. *W*s and *b*s are weights and biases for each gate. The gate functions can decide what should be passed or retained, thus control whether certain information can be propagated or overwritten across the recurrent network.

In our models, LSTM is applied over the sequence generated by CNN. To bidirectional sequence dependencies, a backward LSTM is arranged over the reverse sequence. We denote the encoded result by forward LSTM as $\vec{h}_{enc}$, backward LSTM as $\overleftarrow{h}_{enc}$ and the concatenation of them as $h_{enc} = [\vec{h}_{enc}, \overleftarrow{h}_{enc}]$.

3.5 Attention-LSTM

Attention mechanism is firstly introduced by Bahdanau et al. (2015) in machine translation. By applying attention in an encoder-decoder framework, the model can figure out the contributions of the encoded elements to the generation of the current unit, using an automatic alignment model. Attention is later popularized in other tasks, such as QA (Hermann et al., 2015) and relation extraction (Liu et al., 2016).

Assume the encoded sequence (question+answers) by LSTM is $h_{enc} \in \mathbb{R}^{(n+1)\times d}$ where d is the output dimension of LSTM. For the ith answer (encoded as s_i), the context vector c_i is computed as a weighted sum of the annotations $\{h_j\}$ (h_j stands for the jth element in h_{enc}):

$$c_i = \sum_{j=1}^{n+1} \alpha_{ij} h_j \tag{9}$$

and the weight α_{ij} is computed by

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n+1} \exp(e_{ik})} \tag{10}$$

where

$$e_{ij} = s_i^T \cdot h_j \tag{11}$$

In ARC-II, we degenerate Eq. 11 into simply computing the similarity between sentences, that is $e_{ij} = h_i^T \cdot h_j$. Thus, the attention context for an answer derives from the similarity with the question as well as the associations with other answers. The correlation with previous answers reflect the information of dependency while the correlation with later answers perhaps convey the information of comments.

3.6 CRF

CRF has been shown superior in many sequence labelling tasks (Lafferty et al., 2001;Sha and Pereira, 2003; Quattoni et al.,2004). Given an observation sequence $X=\{x_1, x_2, \ldots, x_n\}$, CRF jointly models the probability of the entire sequence of labels $Y=\{ y_1, y_2, \ldots, y_n \}$ by using the discriminative probability to y_i given x_i and the transition probability between adjacent labels. The original probability model of CRF is written as:

$$p(Y \mid X; W, b) = \frac{\prod_{i=1}^{n} \psi_i(y_{i-1}, y_i, X)}{\sum_{y' \in \mathcal{Y}(X)} \prod_{i=1}^{n} \psi_i(y'_{i-1}, y'_i, X)} \tag{12}$$

where $\psi_i(y', y, x_i) = \exp(W^T_{y',y} x_i + b_{y',y})$ are potential functions, and $\mathcal{Y}(X)$ denotes the set of possible label sequences given X.

In this work, the observed variable is the encoded sequence (QA sequence) generated by Bi-LSTM. A CRF layer is followed to capture the dependencies between adjacent labels via a state transition matrix. Therefore, Equation 12 is transformed into a simpler form by replacing the potential functions by the outputs of Bi-LSTM. Formally, we denote the outputs of Bi-LSTM as $M=\{m_1, m_2, \ldots, m_n\}$ so as to differentiate from the input of the whole network X (the original word ids). By applying softmax, we obtain the predicted score for each answer on each category.

$$P(y_i = j \mid m_i) = \frac{e^{m_i^T w_j + b_j}}{\sum_k e^{m_i^T w_k + b_k}}, \; j,k = 0,1,2 \tag{13}$$

By adding the transition probability from state y_{i-1} to y_i, the probability of the sequence M is:

$$S(M, Y) = \sum_{i=1}^{n} P(y_i = j \mid m_i) + \sum_{i=1}^{n} T(y_i = j \mid y_{i-1} = k) \tag{14}$$

So the probability for the sequence Y can be yielded by applying a softmax over all possible tag sequences:

$$p(Y \mid M) = \frac{e^{s(M,Y)}}{\sum_{y' \in \mathcal{Y}(M)} e^{s(M,y')}} \tag{15}$$

3.7 Training

We pre-trained the word embeddings with word2vec (Miklov et al., 2013) using the continuous bag-of-words model and the embedding dimension is 100. To achieve fixed length sentence representations (100 in this work), we padded each sentence with zero vectors. The embeddings were not fine-tuned during training because of performance decline in the experiments. We add a dropout layer after CNN with the dropout rate 0.1. The parameters are optimized using AdaDelta (Zeiler, 2012) and the learning rate is initialized as 0.01. The settings for CNN are the same as those by Kim (2014). Following Zhou et al. (2015), we set the dimension of the gates as 360. However, we did not follow the settings of CNN in Zhou et al. (2015)'s work since we found that the settings by Kim (2014) can produce better results when adding Bi-LSTM and CRF. We train the network using a complete end-to-end process. The implementation is under the help of Theano (Al-Rfou et al., 2016) and the tagger codebase[4].

4 Experiments

4.1 Experimental Settings

We carried out the experiments on the dataset released by SemEval-2015 cQA shared task (Nakov et al., 2015). The statistics of the dataset are listed in Table 1. The questions and answers are crawled from Qatar Living Forum. The word embeddings are pre-trained on the untagged cQA data[5]. Each answer is manually tagged as *Good*, *Bad* or *Potential* according to its quality in which *Potential* means the answer is potentially useful to the questioner.

The evaluation metrics include macro averaged precision, recall and F_1 over the target categories, denote as *Prec.*, *Recall*, and F_1 in this section. We also reported the F_1s on individual categories to see whether the models work well on certain labels. The models are trained on the training set, tuned on the development set and tested on the testing set.

	No. of Threads (Q)	No. of Answers (A)	Avg. Length
Train	2600	16541	6.36
Dev	300	1645	5.48
Test	329	1976	6.01

Table 1. Statistical data for SemEval-2015 cQA dataset.

We compare our models with five baselines: 1) The Top-1 system in the shared task which includes almost all the previous popular features, such as translation models and word-based topic models (Tran et al., 2015). This is the state-of-the-art work in the literature. 2) The Top-2 system in the shared task which employs multiple syntactic and semantic features, with ensemble learning as its classification schema (Hou et al., 2015). 3) The state-of-the-art neural network-based system that has an analogous architecture as the proposed models, differing in an insufficient modelling of answer correlation and label dependency (Zhou et al., 2015). 4) A global inference model using graph-cut algorithm but only on binary classification (*Good* or *Bad*) (Joty et al., 2015). 5) An improved version of (Joty et al., 2015) which jointly models the dependencies between tags and learns comment-level classifiers. The main idea of the latter two methods is to divide the tagging task into two subtasks: comment-level tagging and pairwise similarity measuring. However, they still depend on laborious feature engineering.

Further, to validate the role of each component, we compare several sub-networks based on the proposed architectures. The sub-networks are denoted as CNN+LSTM (without backward LSTM and CRF), CNN+LSTM+CRF (without backward LSTM), CNN+Bi-LSTM (without CRF), and CNN+Bi-LSTM+CRF (with all components). To test the importance of CRF in modelling label dependency, we

[4] https://github.com/glample/tagger
[5] http://alt.qcri.org/semeval2015/task3/index.php?id=data-and-tools

replace CRF with the previously predicted label (LP) which is encoded using one-hot format and initialized with zero vectors. The naming rule for LP-based models is similar to those for CRF. The parameters were trained on the training set, and tuned on the development set. Finally, the optimal set of parameters were tested on the testing set.

4.2 Multiclass Tagging

ARC-I The results for multiclass (*Good*, *Bad* and *Potential*) tagging are shown in Table 2[6]. From the fourth column, we see that the best F_1 value is achieved by ARC-I with CNN+Bi-LSTM+CRF, which outperforms the state-of-the-art (Tran et al., 2015) by 1.77% and the best neural network-based method on this dataset (Zhou et al. 2015) by 2.82%. Meanwhile, the optimal values on precision and recall are also generated by this setting. The results imply the importance of both Bi-LSTM (for content dependency) and CRF (for label dependency).

We also notice that the addition of LP gains improvement for both the two variants (LSTM vs. Bi-LSTM), but in varying degrees. Without backward LSTM, LP enhance the baseline by 1.63%, while on the other side, the improvement is about 0.2%.One possible explanation is that backward LSTM plays a more important part than the constraint from the previous label. The F_1 values by both the settings with CRF outperform the baselines heavily, and are also better than other settings without label dependency, indicating the effectiveness of applying CRF. It is also remarkable that the optimal overall precision overpasses 60%, which is a significant improvement over the baselines (2.5%). A good precision is helpful in many real world applications such as QA knowledge base building.

Methods	**Prec.**	**Recall**	**F_1**	**F-Good**	**F-Bad**	**F-Pot.**
Baselines						
(Tran et al., 2015)	57.31	57.20	57.19	78.96	78.24	14.36
(Hou et al., 2015)	57.83	56.82	56.41	76.52	74.32	18.41
(Zhou et al., 2015)	56.41	56.16	56.14	77.31	75.88	15.22
ARC-I						
CNN+LSTM	55.74	55.86	55.75	78.29	77.66	11.30
CNN+LSTM+LP	57.26	57.72	57.32	76.37	77.32	18.28
CNN+LSTM+CRF	58.97	58.41	58.61	78.91	77.06	**19.87**
CNN+Bi-LSTM	57.61	56.98	56.75	79.13	78.44	12.70
CNN+Bi-LSTM+LP	57.22	56.84	56.96	78.23	76.11	16.17
CNN+Bi-LSTM+CRF	**60.33**	**58.86**	**58.96**	79.80	78.63	18.46
ARC-II						
CNN+LSTM	58.45	57.07	56.48	79.84	78.84	10.76
CNN+LSTM+LP	56.34	56.67	55.23	**81.71**	**79.95**	4.04
CNN+LSTM+CRF	59.62	57.98	57.92	81.29	79.07	13.39
CNN+Bi-LSTM	56.47	56.06	55.43	80.63	77.68	7.96
CNN+Bi-LSTM+LP	56.89	56.73	56.39	80.78	78.11	10.28
CNN+Bi-LSTM+CRF	59.83	58.41	58.29	81.22	79.60	14.05

Table 2: Experiment results by the proposed models and baselines. The optimal value for each column is marked bold. F-Pot stands for the F_1 value for the *Potential* category.

ARC-II ARC-II has considerable improvement from the baselines. We notice that with the full network components, it outperforms (Zhou et al., 2015) by 2.15% and (Tran et al. 2015) by 1.10%. Similar trends in the addition of LP or CRF also indicate the importance of label dependency and the superiority of CRF to LP. We also test the model without the attention part, but got very bad results, regardless of

[6] The result files can be downloaded from https://github.com/o0laika0o/CNN-LSTM-CRF-for-cQA-answer-tagging

the addition of backward LSTM. It is mainly due to the loss of the question information from the sequence. We think using attention over the post sequence is a neat way to capture relations from both the question and the context.

A statistical test (*t*-test) is further conducted to evaluate the significance of the improvements by the CRF-based methods (compared with the existed state-of-the-art by Tran et al. (2015)). The results show that the improvement of ARC-I with Bi-LSTM and CRF is mildly significant (p<0.08), and all the other three CRF-based results are statistically significant (p<0.05).

4.3 Binary Classification

Methods	Prec.	Recall	F_1	Acc
(Joty et al., 2015)	78.30	82.93	80.55	79.80
(Joty et al., 2016)	77.3	**86.2**	81.5	80.5
ARC-I	81.54	81.29	81.28	81.33
ARC-II	**82.29**	82.22	**82.22**	**82.24**

Table 3: Results for binary classification.

The last two competitors' methods are binary classification, which removes the *Potential* class from the original target category set. Determining *Good* or *Bad* is a more direct way for the answer selection task and is much closer to real world applications. Joty et al. (2015) and Joty et al. (2016) gain improvements from baselines and the latter one is the state-of-the-art work on binary classification.

We carried out experiments on binary classification and show the results in Table 3. It is notable that we have much better precision and accuracy than their methods. And ARC-II gets the best F_1, which outperforms (Joty et al., 2016) by 0.72%. We may say that ARC-II is more appropriate for binary classification albeit it is inferior in predicting the *Potential* class (seen from Table 2, even 4.04% in F_1 for *Potential* while around 80% for *Good* and *Bad*). From this perspective, although the overall F_1 for multiclass tagging is not the best, ARC-II has its unique advantage.

5 Discussion

We can draw a conclusion that backward LSTM and CRF are both good contributors and complement each other in determining the quality tags and CRF plays a more important part. The backward LSTM intends to capture the comments to the previous answers (i.e. a negative comment perhaps follows a bad answer) and we notice that it improves the original model by 1% in ARC-I. However, in ARC-II, the improvement is not as obvious as the former, mainly due to that the attention mechanism already takes the future steps into account. The employment of CRF brings over 2% promotion from the baselines and can be seen as the most vital factor (the role of LSTM is not so significant which can be drawn from Zhou et al. (2015)). For ARC-II, we also tried other forms of attention such as using more parameters to replace the element-wise dot in this work, but failed to get better results. We think a possible explanation is that the explosive parameters cannot be effectively trained given the current size of the training corpus.

From Table 2, we also recognize that the addition of LP is helpful in some cases but in other cases the improvement is not obvious or even LP does bad to the result (CNN+LSTM+LP for ARC-II). The comparisons between methods with LP and CRF prove that soft constraints are more powerful than hard constraints. Moreover, it is noticed that in most cases, label dependency can boost the performance on *Potential*, which indicates that this class rely more on the contextual information.

6 Conclusion

This paper introduces two models for answer quality tagging in cQA, one with a hierarchical architecture from input to output, and the other with attention mechanism integrated. Through the combination of CNN, Bi-LSTM and CRF, we focus on the modelling of context information, including content correlation and label dependency. Experiments show that we achieve the state-of-the-art overall precision, recall, F_1, as well as the best performance on individual classes. Through the comparisons on label dependency, we discover that CRF is superior to others by learning global constraints. Future development may rise from the import of extra features, such as the user metadata, and a thorough pre-processing

towards the noisy input. With adjustment to the architectural elements or training procedures, we believe the models can be incrementally improved further.

Acknowledgement

This paper is supported in part by National 863 Program of China (2015AA015405), National Natural Science Foundation of China (61402128, 61473101 and 61272383), and Strategic Emerging Industry Development Special Funds of Shenzhen (JCYJ20140417172417105 and JCYJ20140508-161040764). We thank the reviewers for their constructive suggestions on this paper.

Reference

Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. 2008. Finding High-quality Content in Social Media. In Proceedings of the 2008 International Conference on Web Search and Data Mining, pages 183–194. ACM.

Rami Al-Rfou, Guillaume Alain, et al. Theano: A Python Framework for Fast Computation of Mathematical Expressions. 2016. arXiv preprint arXiv:1605.02688. MLA.

Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv preprint arXiv:1409.0473.

Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu Mittal. 2000. Bridging the Lexical Chasm: Statistical Approaches to Answer-finding. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 192–199. ACM.

George E Dahl, Tara N Sainath, and Geoffrey E Hinton. 2013. Improving Deep Neural Networks for LVCSR using Rectified Linear Units and Dropout. In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 8609–8613. IEEE.

Alan Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid Speech Recognition with Deep Bidirectional LSTM. In Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on, pages 273–278. IEEE.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, et al. 2015. Teaching Machines to Read and Comprehend. Advances in Neural Information Processing Systems. 2015: 1693-1701.

Yongshuai Hou, Cong Tan, Xiaolong Wang, Yaoyun Zhang, Jun Xu, and Qingcai Chen. 2015. Hitszicrc: Exploiting Classification Approach for Answer Selection in Community Question Answering. In Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval, volume 15, pages 196–202.

Haifeng Hu, Bingquan Liu, Baoxun Wang, Ming Liu, and Xiaolong Wang. 2013. Multimodal DBN for Predicting High-quality Answers in cQA Portals. In Proceedings of ACL, pages 843–847.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In Proceedings of Advances in Neural Information Processing Systems, pages 2042–2050.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. Computer Science.

Shafiq Joty, Alberto Barrón-Cedeno, Giovanni Martino Da San, et al. 2015. Global Thread-level Inference for Comment Classification in Community Question Answering. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). 2015, 15.

Shafiq Joty, Lluıs, and Preslav Nakov. Joint Learning with Global Inference for Comment Classification in Community Question Answering. Proceedings of NAACL-HLT. 2016.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:1408.5882.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In Proceedings of ICML.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. arXiv preprint arXiv:1603.01360

Yankai Liu, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, Maosong Sun. 2016. Neural Relation Extraction with Selective Attention over Instances. To be appeared in Proceedings of ACL 2016.

Xuezhe Ma and Eduard Hovy. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. 2016. arXiv preprint arXiv:1603.01354.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.

Preslav Nakov, Llu´ ıs Marquez, Walid Magdy, Alessandro Moschitti, James Glass, and Bilal Randeree. 2015. SemEval-2015 task 3: Answer Selection in Community Question Answering. SemEval-2015, page 269.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2004. Mapping Dependencies Trees: An Application to Question Answering. In Proceedings of AI&Math 2004, pages 1–10.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional Neural Tensor Network Architecture for Community based Question Answering. In Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI), pages 1305–1311.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional Random Fields for Object Recognition. Advances in Neural Information Processing Systems. 2004.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical Machine Translation for Query Expansion in Answer Retrieval. In Annual Meeting-Association for Computational Linguistics, volume 45, page 464.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic Feature Engineering for Answer Selection and Extraction. In EMNLP, pages 458–467.

Fei Sha and Fernando Pereira. Shallow Parsing with Conditional Random Fields. In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1. Association for Computational Linguistics, 2003.

Yikang Shen, Wenge Rong, Zhiwei Sun, Yuanxin Ouyang, and Zhang Xiong. 2015. Question/answer Matching for cQA System via Combining Lexical and Sequential Information. In Twenty-Ninth AAAI Conference on Artificial Intelligence.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2008. Learning to Rank Answers on Large Online QA Collections. In ACL, volume 8, pages 719–727.

Maggy Anastasia Suryanto, Ee Peng Lim, Aixin Sun, and Roger HL Chiang. 2009. Quality-aware Collaborative Question Answering: Methods and Evaluation. In Proceedings of the second ACM International Conference on Web Search and Data Mining, pages 142–151. ACM.

Quan Hung Tran, Vu Tran, Tu Vu, Minh Nguyen, and Son Bao Pham. 2015. Jaist: Combining Multiple Features for Answer Selection in Community Question Answering. In Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval, volume 15, pages 215–219.

Baoxun Wang, Xiaolong Wang, Chengjie Sun, Bingquan Liu, and Lin Sun. 2010. Modelling Semantic Relevance for Question-answer Pairs in Web Social Communities. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1230–1238. ACL.

Martin Wöllmer, Erick Marchi, Stefano Squartini, and Bjö rn Schuller. 2011. Multi-stream LSTM-HMM Decoding and Histogram Equalization for Noise Robust Keyword Spotting. Cognitive Neurodynamics, 2011, 5(3): 253-264.

Xuchen Yao, Benjamin Van Durme, Chris CallisonBurch, and Peter Clark. 2013. Answer Extraction as Sequence Tagging with Tree Edit Distance. In HLTNAACL, pages 858–867. Citeseer.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question Answering using Enhanced Lexical Semantic Models.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. arXiv preprint arXiv:1412.1632.

Matthew D Zeiler. 2012. Adadelta: An Adaptive Learning Rate Method. arXiv preprint arXiv:1212.5701.

Xiaoqiang Zhou, Baotian Hu, Qingcai Chen, Buzhou Tang, and Xiaolong Wang. 2015. Answer Sequence Learning with Neural Networks for Answer Selection in Community Question Answering. arXiv preprint arXiv:1506.06490.

Semantically Motivated Hebrew Verb-Noun Multi-Word Expressions Identification

Chaya Liebeskind, Yaakov HaCohen-Kerner
Department of Computer Science, Jerusalem College of Technology, Lev Academic Center
21 Havaad Haleumi St., P.O.B. 16031
9116001 Jerusalem, Israel
liebchaya@gmail.com, kerner@jct.ac.il

Abstract

Identification of Multi-Word Expressions (MWEs) lies at the heart of many natural language processing applications. In this research, we deal with a particular type of Hebrew MWEs, Verb-Noun MWEs (VN-MWEs), which combine a verb and a noun with or without other words. Most prior work on MWEs classification focused on linguistic and statistical information. In this paper, we claim that it is essential to utilize semantic information. To this end, we propose a semantically motivated indicator for classifying VN-MWE and define features that are related to various semantic spaces and combine them as features in a supervised classification framework. We empirically demonstrate that our semantic feature set yields better performance than the common linguistic and statistical feature sets and that combining semantic features contributes to the VN-MWEs identification task.

1 introduction

Multi-word expressions (MWE) were defined by Sag et al. (2002) as "idiosyncratic interpretations that cross word boundaries (or spaces)" while Bouamor et al. (2012) defined a MWE as "a combination of words for which syntactic or semantic properties of the whole expression can not be obtained from its parts".

Jackendoff (1997) claimed that that the frequency of MWEs in a speaker's lexicon is of the same order of single words. Due to their relative high frequency and complexity, MWEs require high-quality treatment in many applications in natural language processing (NLP) such as data mining, machine translation (MT), information retrieval, natural language understanding, natural language generation, question answering (QA), text summarization, and word sense disambiguation (WSD).

The aim of this work is to explore Hebrew Verb-Noun MWEs (VN-MWEs). VN-MWEs are MWEs whose constitutents include a verb and a noun. The motivation of this research is to enable automatic identification of VN-MWEs for various NLP tasks such as MT, QA, and WSD, and to classify collocations that include verbs as VN-MWEs or non-VN-MWEs.

Most prior efforts to automatically classify MWEs focused on three approaches: (1) Statistical approaches, either frequency-based or co-occurrence-based (Dias et al., 1999; Deane, 2005; Pecina and Schlesinger, 2006). (2) Linguistic approaches that are based on NLP tools, such as taggers and parsers (Al-Haj, 2009; Bejcek et al., 2013; Green et al., 2013). (3) Hybrid approaches which combine statistical and linguistic approaches (Baldwin, 2005; Boulaknadel et al., 2008; Farahmand and Nivre, 2015).

In this paper, we claim that on top of standard linguistic and statistical metrics, MWE identification methods can greatly benefit from exploiting semantically motivated cues. For example, when a VN-MWE is highly idiomatic, the semantics of the verb and the noun are not likely to overlap.

The contribution of this paper is, in a first step, to combine semantic features in the framework of supervised MWE classification. We suggest a simple semantically-motivated indicator that helps to

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1242–1253, Osaka, Japan, December 11-17 2016.

detect VN-MWEs. Then, we define semantic features that implement our indicator in various semantic spaces and integrate them within our Machine Learning (ML) classification algorithm.

We show that combining semantic features improves the accuracy and F-score results of VN-MWE classification. Moreover, our analysis reveals that the semantic feature set yields better results than each one of the two other approaches, the statistical and the linguistic.

The rest of this paper is organized as follows: Section 2 introduces relevant background about MWEs in Hebrew and identification of MWEs using semantic features. Section 3 presents the linguistic, statistical, and semantic feature sets that were applied for the supervised VN-MWEs classification task. Section 4 introduces the experimental setting, the experimental results for nine ML methods, and their analysis. Finally, Section 5 summarizes the main findings and suggests future directions.

2 Background

2.1 MWEs in Hebrew

Al-Haj (2009) presented an architecture for lexical representation of MWEs written in Hebrew and a specification of the integration of MWEs into a morphological processor of Hebrew. He also introduced a system that extracts noun compounds from Hebrew raw text based on their idiosyncratic morphological and syntactic features. A support vector machine (SVM) classifier using these features identified noun-noun constructs with an accuracy of over 80%.

Al-Haj and Wintner (2010) created for each noun-noun construction, a vector of the 16 features: 12 linguistically-motivated features and 4 collocation measures. Their dataset includes 463 instances, of which 205 are noun compounds (positive examples) and 258 negative. They applied LIBSVM classifier (Chang and Lin, 2001) with a radial basis function kernel. The best combination of features yielded an accuracy of 80.77% and F-score of 78.85, representing a reduction of over one third in classification error rate compared with the baseline.

Tsvetkov and Wintner (2012) proposed a methodology for extracting MWEs in Hebrew-English corpora. MWEs of various types are extracted along with their translations, from small, word-aligned parallel corpora. They focused on misalignments, which typically indicate expressions in the source language that are translated to the target in a non-compositional way. They implemented a simple algorithm that proposes MWE candidates based on such misalignments, relying on 1:1 alignments as anchors that delimit the search space. Evaluation of the algorithm's quality demonstrates significant improvements over Naive alignment-based methods.

Tsvetkov and Wintner (2014) proposed a framework for identifying MWEs in texts using multiple sources of linguistic information. Their system enables identification of MWEs of various types and multiple syntactic constructions. Their methodology is unsupervised and language-independent; it requires relatively few language resources and is thus suitable for a large number of languages. They applied four ML methods. The system was tested on three languages: Hebrew, French, and English. Applying the Bayesian Network ML method on a combination of linguistically motivated features and feature interdependencies reflecting domain knowledge yielded the best results (Hebrew: accuracy of 76.82% and F-score of 0.77; French: accuracy of 79.04% F-score of 0.778; and English: accuracy of 83.52% and F-score of 0.835).

Sheinfux et al. (2015) introduced different types of verbal MWEs in Modern Hebrew. In addition, they proposed an analysis of these MWEs in the framework of HPSG, and they incorporated this analysis into HeGram, a deep linguistic processing grammar of Modern Hebrew. Their analysis covers various MWE types, including challenging phenomena such as (possessive) co-indexation and internal modification. The HeGram grammar produced two analyses for most MWEs, corresponding to their idiomatic and literal readings.

Liebeskind and HaCohen-Kerner (2016) presented a lexical resource containing 505 Verb-Noun MWEs (VN-MWEs) in Hebrew. These VN-MWEs (247 bigrams and 258 trigrams) were manually collected from five web resources and annotated. Following Al-Haj (2009), the authors classified the linguistic properties of these VN-MWEs along 3 dimensions: morphological, syntactic, and semantic. The major findings are: (1) the main characteristic properties of VN-MWEs are the semantic properties

of non-compositionality and lexical fixedness; (2) High degrees of idiomaticity (92%) and lexical fixedness (94%) were found for the VN-MWEs; (3) 82% of the VN-MWEs do not allow any changes in the constituent order; and (4) 87% have a non-compositional syntax.

2.2 Identification of MWEs using Semantic Features

Katz and Giesbrecht (2006) applied latent semantic analysis (LSA) vectors to distinguish compositional from non-compositional uses of German expressions. The LSA vectors of compositional and non-compositional meaning were constructed from a training set of example sentences. Afterwards, a simple nearest neighbor algorithm was applied on the LSA vectors of the tested MWEs. The LSA-based classifier obtained an average accuracy of 72%, which outperformed the simple maximum-likelihood baseline with accuracy of 58%.

Sporleder and Li (2009) proposed supervised and unsupervised methods to distinguish literal from non-literal usages of idiomatic expressions by measuring the semantic relatedness of an expression's component words to nearby words in the text. Their assumption was that if an expression is used literally, but not idiomatically, its component words will be related semantically to a few words in the surrounding discourse. If one or more of the expression's components were sufficiently related to enough surrounding words, the usage was classified as literal, otherwise as idiomatic. The supervised classifier method (90% F-score on literal uses) was better than the lexical chain classifier methods (60% F-score).

Biemann and Giesbrecht (2011) provided an overview of the shared task at the ACL-HLT 2011 DiSCo (Distributional Semantics and Compositionality) workshop. The authors described the motivation for the shared task, the acquisition of datasets, the evaluation methodology, and the results of participating systems. The evaluation shows that most systems outperformed simple baselines, yet have difficulties in reliably assigning a compositionality score that closely matches the gold standard. Generally, approaches based on word space models performed slightly better than approaches relying merely on statistical association measures.

Guevara (2011) proposed and evaluates a framework that models the semantic compositionality in computational linguistics based on the combination of distributional semantics and supervised ML. The applied method, Partial Least Squares (PLS) Regression, outperformed all the competing models in the reported experiments with Adjective-Noun (AN) pairs extracted from the BNC.

Salehi et al. (2015) introduced the first attempt to use word embeddings to predict the compositionality of MWEs. They considered both single- and multi-prototype word embeddings. Experimental results showed that, in combination with a back-off method based on string similarity, word embeddings are superior to, or competitive with state-of-the-art methods over 3 standard compositionality datasets ((1) English noun compounds ("ENCs"); (2) English verb particle constructions ("EVPCs"); and (3) German noun compounds ("GNCs")).

3 Supervised VN-MWEs Classification

In the previous section, we discussed linguistic properties of Hebrew VN-MWEs that may help in distinguishing coincidental word combinations from collocations. We next define them and describe how to incorporate these properties as features within a ML framework for classifying candidate VN-MWEs.

3.1 Feature Sets

We next detail how the semantic properties of VN-MWEs, as well as the linguistic and statistical properties found useful in prior work, are encoded as features. Then, in Section 4, we describe the supervised ML model and our feature analysis procedure. There are 206 features in our model, divided into 3 sets: linguistic, statistical and semantic. We defined the sets as Al-Haj and Wintner (2010) and Liebeskind and HaCohen-Kerner (2016) did. However, we note that semantic information is often defined as a sub-type of linguistic information and it might be more accurate to contrast morpho-syntactic information (i.e., parts-of-speech and syntactic parses) with semantic information.

3.1.1 Linguistic features

Most of our linguistic features are based on information extracted from a Part-Of-Speech (POS) tagger for the Hebrew language (Adler, 2007). Our linguistic features encode both morphological and syntactical properties of VN-MWEs. For each candidate VN-MWE, we compute counts that reflect the reasonableness of the candidate to represent at least one of its linguistic properties. We focus on the linguistic properties that Liebeskind and HaCohen-Kerner (2016) recognized as notable. Our linguistic properties include two families of properties: morphological and syntactical.

Partial Inflection Following Al-Haj and Wintner (2010), for each VN-MWE candidate, the following 8 features are defined: the number of occurrences of the candidate in which both constituents are in singular, the number of occurrences in which both constituents are in plural, the number of occurrences in which the verb is in singular and the noun is in plural, the number of occurrences in which the noun is in singular and the verb is in plural, the number of occurrences of the verb in plural, the number of occurrences of the verb in singular, the number of occurrences of the noun in plural and the number of occurrences of the noun in singular. Two additional features that we calculate are the number of verb suffixes, which indicate a conjugation of grammatical tense, possession or direct objects, as well as the number of noun suffixes, which indicate nouns number and gender (*ildi*[1] (my child), *ild**inw*** (our children), *ild**h*** (a girl)).

Syntactic Fixedness VN-MWEs are expected to appear in restricted syntactic forms. Fazly and Stevenson (2006) suggested that to quantify the syntactic fixedness of a VN-MWE candidate, we need to: (i) identify relevant syntactic patterns and (ii) translate the frequency distribution of the candidate in the identified patterns into a measure of syntactic fixedness. Following this approach, we define syntactic patterns and clues as features in our supervised framework.

We use the most frequent POS patterns found in Liebeskind and HaCohen-Kerner (2016)'s VN-MWEs lexical resource as relevant syntactic patterns and count the number of occurrences of the candidate in each of these patterns (7 features).

Since prepositions and definite articles frequently appear in these patterns, we counted the number of occurrences of the candidate in which it includes an article, a pronoun, a particle, a conjunction, an auxiliary or a negation (6 features). Then, considering the fact that some of these POS are often Hebrew prefixes, we also encoded prefixes' occurrences. The features that we calculate are the number of occurrences of verb and noun prefixes (2 features), the number of occurrences of prefixes which start with a certain frequent *formative letter* (7 features for each POS, verb and noun), the number of occurrences of a certain frequent prefix (36 features for each POS). Our Hebrew stopword list also include some particles. Therefore, we calculated an additional feature of the number of stopwords in the candidate VN-MWE (1 feature).

The syntactic property of compositionality is encoded by the difference between the number of occurrences of the candidate constituents with and without a *slot* (1 feature). The syntactic property of a number of syntactic structures that permit a change in the order of constituents is encoded by the difference between the number of occurrences of the candidate constituents in their original order and the number of occurrences of the candidate constituent in a reversed order (1 feature).

3.1.2 Statistical features

We define some statistical features based on frequency and co-occurrence affinity. Each of these features is separately calculated for two candidate representations: surface and lemmatized. First, we compute the raw frequency of the VN-MWE candidate and the raw frequency of its verb and noun constituents (6 features). Then, we utilize features that represent known association measures: Log-likelihood, Total mutual information, Pointwise mutual information and Poisson-Stirling measure. We calculate them for bigrams and trigrams separately (16 features). Finally, we define four statistical features based on two non-parametric methods, which does not make the independence assumption and allows scores to be

[1]To facilitate readability we use a transliteration of Hebrew using Roman characters; the letters used, in Hebrew lexicographic order, are abgdhwzxTiklmns`pcqršt.

compared across n-grams of different length: Mutual Expectation (ME) (Dias et al., 1999) and Mutual Rank Ratio (MRR) (Deane, 2005) (4 features).

In addition, we calculate the number of words, number of characters and the average number of characters per word (3 features) for each candidate in its base form.

3.1.3 Semantic features

Liebeskind and HaCohen-Kerner (2016) observed that the most characteristic properties of VN-MWEs are the semantic properties of compositionality and lexical fixedness. To encode this property, we represent the meaning of the candidate's constituents by vectors in the same semantic space. Due to the idiomaticity of VN-MWEs, we expect the similarity of vectors of words in a non-VN-MWE to be greater than the similarity of vectors of words in a VN-MWE. For example, the VN-MWE *to eat one's hat* vs. the non-VN-MWE *to eat an apple*. We expect the vectors of *eat* and *apple*, which share a common context, to be closer than the vectors of *eat* and *hat* in a representative semantic space.

We construct semantic features from the following five different semantic spaces:

(1) Hyperspace Analogue to Language (HAL) (Lund and Burgess, 1996): The algorithm computes a word-by-word matrix, using a 10-word reading frame that moves incrementally through a corpus of text. The algorithm considers context only as the words that immediately surround a given word. Any time two words are simultaneously in the frame, the association between them is increased, that is, the corresponding cell in the matrix is incremented. The amount by which the association is incremented varies inversely with the distance between the two words in the frame; closer neighboring words are thought to reflect more of the focus word's semantics and so are weighted higher. The algorithm also records word-ordering information by treating the co-occurrence differently based on whether the neighboring words appeared before or after the focus word.

(2) Correlated Occurrence Analogue to Lexical Semantics (COALS) (Rohde et al., 2006): The algorithm constructs a word-by-word matrix where each element in the matrix represents how frequently word_i occurs with word_j in a certain window. The matrix is then normalized by correlation, and any negative values are set to zero and all other values are replaced by its square root. Then, optionally, the Singular Value Decomposition (SVD) is used to reduce the word co-occurrence matrix.

(3) Random Indexing (RI) (Sahlgren, 2005): The algorithm uses statistical approximations of the full word co-occurrence data to achieve dimensionality reduction. RI represents co-occurrence through index vectors. Each word is assigned a high-dimensional, random vector that is known as its index vector. These index vectors are very sparse, which ensures that the chance of any two arbitrary index vectors having an overlapping meaning is very low. Word semantics are calculated for each word by keeping a running sum of all of the index vectors for the words that co-occur.

(4) Reflective Random Indexing (RRI) (Cohen et al., 2010): The algorithm is a second-order iterative extension to the RI method. Reflective random indexing adds another cycle by restarting the construction of the term vectors using the basis of document vectors, and then creating the document vectors again using the term vectors. Such retraining has been found to improve the ability of RI to make indirect inferences, drawing meaningful associations between terms that do not occur together in any document.

(5) Word Embeddings (Mikolov et al., 2013): Word embedding is the collective name for neural-network based approaches in which words are embedded into a low dimensional space. In word embedding models, the contexts of each word are modeled by a d-dimensional vector of real numbers. The vector are meaningless on their own, but semantically similar words have similar vectors, and vector similarities are easy to compute.

Each of these five semantic spaces is generated for two word representations: surface and lemmatized. We use different measures to compute the similarity between two vectors. For the first four semantic spaces[2], we calculate Cosine similarity, Lin similarity, Euclidean distance, Pearson correlation, average common feature rank, Jaccard index, Tanimoto coefficient, and Spearman rank correlation. For the fifth semantic space[3], we calculate cosine similarity, euclidean distance, and manhattan distance.

[2]implemented by the S-Space Package `https://github.com/fozziethebeat/S-Space`

[3]implemented by the deeplearning4j word2vec package `http://deeplearning4j.org/word2vec`

Some of the measures did not yield a valid score for all the examples in our dataset. As a result, the total number of semantic features is 62.

An additional semantic feature, which measures lexical fixedness, counts the number of occurrences of the VN-MWE candidate in the Bible. MWEs from the Bible are citations that tend to be fixed, replacing any of their constituents by a semantically similar word generally results in an invalid or a literal expression.

We note that corpus-based statistics are used to calculate some of the linguistic features (e.g., the features which encode the Partial Inflection property). Additionally, some of the statistical features, such as Mutual Expectation (ME) and Mutual Rank Ratio (MRR), capture the semantic behavior of VN-MWEs.

4 Evaluation and Analysis

4.1 Experimental setting

Following Al-Haj and Wintner (2010), we used four of the MILA knowledge center[4]corpora: the Knesset corpus, which contains the Israeli parliament proceedings from 2004-2005; the Haaretz corpus that contains articles from the Haaretz newspaper from 1991; TheMarker corpus, which contains financial articles from the TheMarker newspaper from 2002; and the Arutz 7 corpus, which contains newswire articles from 2001-2006. From the morphologically disambiguated version of the corpora (Itai and Wintner, 2008; Yona and Wintner, 2008; Bar-haim et al., 2008), we extracted all word bigrams and trigrams that include a verb and a noun.

To evaluate our proposed supervised model, we constructed a labeled dataset. We selected all the word bigrams and trigrams that occur at least 25 times in the corpora. These candidates were annotated by two annotators, who were asked to classify them as a VN-MWE or a non-VN-MWE. We evaluated the inter-annotator agreement and observed a Kappa (Cohen, 1960) value of 0.59, which is considered as moderate (Landis and Koch, 1977). Thus, we considered a candidate as a VN-MWE or not only if both annotators agreed on its classification. This reduced the labeled data to 553 instances, of which 306 are VN-MWEs (256 bigrams and 50 trigrams) and 247 are non-VN-MWEs (157 bigrams and 90 trigrams).

4.2 Application of nine Machine Learning methods

We combined the features in a supervised classification framework using nine ML methods: Random Forest, Decision Tree, Bagging, Adaboost, Bayes Network, Supported Vector Machine (SVM), Logistic Regression and Multilayered Perceptron. The accuracy rate of each ML method was estimated by a 10-fold cross-validation test. We ran these ML methods by the WEKA platform (Witten and Frank, 2005; Hall et al., 2009) using the default parameters. Table 1 shows the performances of the different ML methods on the full feature set of 206 features, as described above. The best ML method was Random Forest. Therefore, we have performed further experiments using only this method. These experiments are presented in the next sub-section.

4.3 Further experimental results using the random forest method

In this research, we defined three types of feature sets (Section 3): linguistic, statistic and semantic. The classification results of the Random Forest algorithm (the best ML method in Table 1) on each of the sets are presented in the left side of Table 2. The semantic feature set yielded the best accuracy result (77.4%). The advantage of the semantic feature set over the linguistic and statistical feature sets is notable (3.5% and 5% respectively) and is statistically significant according to the McNamar test (McNemar, 1947) for the statistical feature set (p=0.017). The advantage is also almost statistically significant at level 0.05 for the linguistic feature set (p=0.056).

A hybrid approach, which combines the linguistic and statistical information, is commonly used in MWE extraction. Therefore, we investigated different combinations of feature sets. The results of our exploration are presented in the right side of Table 2. The best results were obtained using all the three sub-feature sets. However, the contribution of the linguistic feature set was negligible (80.47% vs.

[4]http://www.mila.cs.technion.ac.il/resources/

#	ML Method	Accuracy (%)	F-Measure
1	Random Forest	**80.47**	**0.795**
2	Decision Tree (J48)	71.25	0.708
3	Bagging	78.84	0.78
4	AdaBoost (M1)	74.32	0.736
5	Bayes Network	69.80	0.703
6	Logistic Regression	70.52	0.706
7	Multilayered Pereceptron	68.72	0.686
8	SVM (SMO)	76.13	0.76
9	SVM (LibSVM)	63.11	0.488

Table 1: Comparison of results obtained by nine ML methods

Feature Set	Accuracy (%)	F-Measure	Feature Sets	Accuracy (%)	F-Measure
Linguistic	73.96	0.721	Linguistic & Semantic	77.4	0.765
Statistical	72.51	0.712	Linguistic & Statistic	78.84	0.777
Semantic	**77.4**	**0.767**	Semantic & Statistic	**80.29**	**0.796**
			All	**80.47**	**0.796**

Table 2: Comparison of results for different combinations of feature sets

80.29%). As was found in previous studies (Justeson and Katz, 1995; Pecina, 2010), the approach of combining linguistic and statistical features works efficiently. Yet, combining linguistic and semantic features did not yield any improvement over using only the semantic feature set.

For each of the above feature set configurations, we tried to filter out non-relevant features using two well-known feature selection methods: Information gain (InfoGain, IG) (Hunt et al., 1966) and Correlation-based Feature Subset (CFS) (Hall, 1998). The use of these two feature selection methods did not improve the accuracy of any configuration. However, we used the information obtained by the IG selection method to better understand which features have more influence on the classification accuracy. Table 3 presents the features, which were selected by the IG method for the three feature sets (the number in parentheses is the feature rank). The linguistic properties of partial inflection and constituent order were found as important properties for distinguishing MWEs from non-MWEs. The two non-parametric statistical features, Mutual Expectation (ME) and Mutual Rank Ratio (MRR), outperformed other baseline association measures. The good performance of the algorithm using the semantic features is due to the combination of various semantic spaces and vector comparison measures.

Table 4 shows the features that were selected by the IG method for the different combinations of feature sets. For each feature sub-set of a combined configuration, Table 4 details how many and which of its selected features were also selected by the IG measure when each set was tested as a standalone feature set (see Table 3). While the same semantic features were selected for the standalone (Semantic) and combined configurations (Linguistic & Semantic and Semantic & Statistical), different linguistic and statistical features were selected by each of the configurations that include them.

We further analyze our suggested semantic feature set by comparing the performance of the different semantic spaces. Table 5 shows the classification results of the Random Forest algorithm on the various sub-sets of the semantic features. The semantic features that were constructed by the HAL semantic space outperformed the other semantic representations. The advantage of the HAL semantic space might be due to its sensitivity to word-ordering. This sensitivity enables the representation to model the important constituent order linguistic property of VN-MWE. The low performance of the word embedding space could be explained either by its low number of features or by the fact that these vector were constructed without any task-depended training.

Finally, we investigated the False Positive (FP) and False Negative (FN) classifications of our suggested semantic feature set. We found that some of the FPs were due to light verbs, such as ***lqbl*** *mid`* (to

Feature set	# of feat.	Feature list
Linguistic	10	SINGUALR_VERB_PLURAL_NOUN (1), COUNJUNCTION (2), CONSTITUENT_ORDER (5). PREFIX_VERB (*wmš* (4), starts with *m* (7), *kšb* (9)), PREFIX_NOUN (*mh* (3), *š* (8), *kšm* (10))
Statistical	5	TRIGRAMSPMI (1), MUTUALRANKRATIO (2), NOUNLFREQUENCY (3), MUTUALSCORE (4), TRIGRAMSLL (5)
Semantic	23	**COAL**: PEARSON (1), COSINE (5), AVERAGE_COMMON_FEATURE_RANK (ACFR) (13), **COAL_LEMMA**: PEARSON (2), COSINE (3), TANIMOTO (4), ACFR (10)
		HAL: EUCLIDEAN (8), ACFR (11), **HAL_LEMMA**: ACFR (6), LIN (17), TANIMOTO (19), PEARSON (22), COSINE (23)
		RI: EUCLIDEAN (7), **RI_LEMMA**: ACFR (12), TANIMOTO (15), LIN (16)
		RRI: EUCLIDEAN (9), **RRI_LEMMA**: SPEARMAN (18)
		Word Embeddings: EUCLIDEAN (20), COSINE (21), **Word Embeddings_LEMMA**: MANHATTAN (14)

Table 3: InfoGain feature selection of the linguistic, statistic and semantic feature sets

get information) and ***`wšh*** *`bwdh* (to make a work). The general meaning of the light verbs decreased the vectors comparison score and candidates with light verbs were wrongly classified as VN-MWEs. This might be because light verbs have little semantic content of their own and they are used in combination with various nouns. Thus, the semantic similarity between the light verb and a specific noun was relatively low. A possible solution to the light verb issue is to use a directional inclusion-based measure to compute the similarity between two vectors (Weeds and Weir, 2003; Clarke, 2009; Kotlerman et al., 2010).

Another interesting finding is that domain-specific VN-MWEs were often misclassified as FNs. VN-MWEs like *lhqim mmšlh* (to establish a government) and *lgbš `mdh* (to form an opinion) were wrongly classified as non-MWEs since they frequently co-occur in our political domain, so their semantic vectors are rather close.

5 Conclusions and Future Work

We presented a supervised classification model for identification of Hebrew VN-MWEs. Our semantic feature set yields better performance than the common linguistic and statistical feature sets and that combining semantic features contributes to the Hebrew VN-MWEs identification task.

Most previous related studies apply only one ML method. An exception was the study of Tsvetkov and Wintner (2014), which applied 4 ML methods. In this research, we applied 9 ML methods. Moreover, we have performed further experiments using only the Random Forest method, which has been found as the best ML method for our task. Our experiment over a manually labeled dataset showed that the semantic feature set outperforms the statistical and linguistic feature sets and that combining semantic features with the two other feature sets further improved the performance (especially with the statistical set).

In future work, we would like to investigate more sophisticated models for representing the semantic meaning of VN-MWEs. For example, we plan to extend the single-word vector representation to learn larger semantic composition representations (Baroni and Zamparelli, 2010; Grefenstette and Sadrzadeh, 2011; Socher et al., 2012). We also plan to investigate directional inclusion-based similarity measures for computing vector similarity.

In addition, we plan to adopt our model to under-resourced languages, many of them are found in the developing world where we lack the linguistic information.

Feature set	# of feat.	Sub-feature set	Overlapping features	Additional features in top10
Linguistic & Semantic	33	Linguistic	CONSTITUENT_ORDER (1/10)	None
		Semantic	all (23/23)	None
Linguistic & Statistical	15	Linguistic	CONSTITUENT_ORDER (1/10)	POSPATTERN (verb + preposition + noun), SINGULAR_VERB, PREFIX_NOUN (*wšb*, *kl*), PLURAL_VERB_SINGULAR_NOUN, PREFIX_VERB (b), PREFIX_NOUN_NUM
		Statistical	MUTUALRANKRATIO (1/5)	MUTUALSCORELEMMA, BIGRAMSTMI
Semantic & Statistical	28	Semantic	all (23/23)	None
		Statistical	None (0/5)	FREQUENCY

Table 4: IG selection results for the different combinations of feature sets

Semantic Space	# of feat.	Accuracy (%)	F-Measure	ROC Area
COAL	13	73.59	0.731	0.8
HAL	14	**75.04**	**0.738**	**0.82**
RI	14	72.87	0.717	0.782
RRI	15	72.87	0.716	0.781
Word Embedding	6	64.56	0.635	0.69

Table 5: Comparison of the results obtained by different semantic sub-spaces

Acknowledgements

We would like to express our deep gratitude to Avital Day, our research assistant, for her help in programming and carrying out the research experiments. We would also like to acknowledge the networking support by the COST Action IC1207: PARSEME: PARSing and Multi-word Expressions. This work was partially funded by an internal research grant from Jerusalem College of Technology, Lev Academic Center.

References

Meni Adler. 2007. *Hebrew Morphological Disambiguation: An Unsupervised Stochastic Word-based Approach*. Ph.D. thesis, Ben-Gurion University of the Negev.

Hassan Al-Haj and Shuly Wintner. 2010. Identifying multi-word expressions by leveraging morphological and syntactic idiosyncrasy. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 10–18, Beijing, China, August. Coling 2010 Organizing Committee.

Hassan Al-Haj. 2009. *Hebrew multiword expressions: Linguistic properties, lexical representation, morphological processing, and automatic acquisition*. Ph.D. thesis, University of Haifa.

Timothy Baldwin. 2005. Deep lexical acquisition of verb-particle constructions. *Comput. Speech Lang.*, 19(4):398–414, October.

Roy Bar-haim, Khalil Sima'an, and Yoad Winter. 2008. Part-of-speech tagging of modern hebrew text. *Natural Language Engineering*, 14(2):223–251, April.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

Eduard Bejcek, Pavel Stranák, and Pavel Pecina. 2013. Syntactic identification of occurrences of multiword expressions in text using a lexicon with dependency structures. In *Proceedings of the 9th Workshop on Multiword Expressions*, pages 106–115.

Chris Biemann and Eugenie Giesbrecht. 2011. Distributional semantics and compositionality 2011: Shared task description and results. In *Proceedings of the workshop on distributional semantics and compositionality*, pages 21–28. Association for Computational Linguistics.

Dhouha Bouamor, Nasredine Semmar, and Pierre Zweigenbaum. 2012. Identifying bilingual multi-word expressions for statistical machine translation. In *LREC*, pages 674–679.

Siham Boulaknadel, Be'atrice Daille, and Driss Aboutajdine. 2008. A multi-word term extraction program for arabic language. In *LREC*, pages 1485–1488. European Language Resources Association.

Daoud Clarke. 2009. Context-theoretic semantics for natural language: an overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 112–119. Association for Computational Linguistics.

Trevor Cohen, Roger Schvaneveldt, and Dominic Widdows. 2010. Reflective random indexing and indirect inference: A scalable method for discovery of implicit connections. *Journal of biomedical informatics*, 43(2):240–256.

J. Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37.

Paul Deane. 2005. A nonparametric method for extraction of candidate phrasal terms. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 605–613. Association for Computational Linguistics.

Gaël Dias, Sylvie Guilloré, and JG Pereira Lopes. 1999. Language independent automatic acquisition of rigid multiword units from unrestricted text corpora. *Traitement Automatique des Langues Naturelles, Institut dEtudes Scientifiques, Cargèse, France*, pages 333–339.

Meghdad Farahmand and Joakim Nivre. 2015. Modeling the statistical idiosyncrasy of multiword expressions. In *Proceedings of NAACL-HLT*, pages 34–38.

Afsaneh Fazly and Suzanne Stevenson. 2006. Automatically constructing a lexicon of verb phrase idiomatic combinations. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 337–344, Trento Italy.

Spence Green, Marie-Catherine de Marneffe, and Christopher D. Manning. 2013. Parsing models for identifying multiword expressions. *Computational Linguistics*, 39(1):195–227, March.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404. Association for Computational Linguistics.

Emiliano Guevara. 2011. Computing semantic compositionality in distributional semantics. In *Proceedings of the Ninth International Conference on Computational Semantics*, IWCS '11, pages 135–144, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November.

M. Hall. 1998. *Correlation-based feature subset selection for machine learning*. Ph.D. thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand.

E. B. Hunt, J. Marin, and P. J. Stone. 1966. *Experiments in Induction*. Academic Press, New York.

Alon Itai and Shuly Wintner. 2008. Language resources for hebrew. *Language Resources and Evaluation*, 42(1):75–98.

Ray Jackendoff. 1997. *The architecture of the language faculty*. Number 28. MIT Press.

John S. Justeson and Slava M. Katz. 1995. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(1):9–27.

Graham Katz and Eugenie Giesbrecht. 2006. Automatic identification of non-compositional multi-word expressions using latent semantic analysis. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 12–19. Association for Computational Linguistics.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359–389.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Chaya Liebeskind and Yaakov HaCohen-Kerner. 2016. A lexical resource of hebrew verb-noun multi-word expressions. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, LREC'16, pages 522–527, Portoroz, Slovenia, may. European Language Resources Association (ELRA).

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Pavel Pecina and Pavel Schlesinger. 2006. Combining association measures for collocation extraction. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 651–658, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pavel Pecina. 2010. Lexical association measures and collocation extraction. *Language Resources and Evaluation*, 44:137–158.

Douglas L. T. Rohde, Laura M. Gonnerman, and David C. Plaut. 2006. An improved model of semantic similarity based on lexical co-occurence. *Communications of the ACM*, 8:627–633.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann A. Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. In *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing '02, pages 1–15, London, UK, UK. Springer-Verlag.

Magnus Sahlgren. 2005. An introduction to random indexing. In *Methods and applications of semantic indexing workshop at the 7th international conference on terminology and knowledge engineering, TKE*, volume 5.

Bahar Salehi, Paul Cook, and Timothy Baldwin. 2015. A word embedding approach to predicting the compositionality of multiword expressions. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 977–983.

Livnat Herzig Sheinfux, Tali Arad Greshler, Nurit Melnik, and Shuly Wintner. 2015. Hebrew verbal multi-word expressions. In *Proceedings of the 22nd international conference on Head-Driven Phrase Structure Grammar (HPSG 2015)*, pages 123–136.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Caroline Sporleder and Linlin Li. 2009. Unsupervised recognition of literal and non-literal use of idiomatic expressions. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 754–762. Association for Computational Linguistics.

Yulia Tsvetkov and Shuly Wintner. 2012. Extraction of multi-word expressions from small parallel corpora. *Natural Language Engineering*, 18(04):549–573.

Yulia Tsvetkov and Shuly Wintner. 2014. Identification of multiword expressions by combining multiple linguistic information sources. *Computational Linguistics*, 40(2):449–468.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 81–88. Association for Computational Linguistics.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Shlomo Yona and Shuly Wintner. 2008. A finite-state morphological grammar of hebrew. *Natural Language Engineering*, 14:173–190, 4.

Semantic Relation Classification via Hierarchical Recurrent Neural Network with Attention

Minguang Xiao **Cong Liu***
School of Data and Computer Science, Sun Yat-sen University
xiaomg@mail2.sysu.edu.cn, liucong3@mail.sysu.edu.cn

Abstract

Semantic relation classification remains a challenge in natural language processing. In this paper, we introduce a hierarchical recurrent neural network that is capable of extracting information from raw sentences for relation classification. Our model has several distinctive features: (1) Each sentence is divided into three context subsequences according to two annotated nominals, which allows the model to encode each context subsequence independently so as to selectively focus as on the important context information; (2) The hierarchical model consists of two recurrent neural networks (RNNs): the first one learns context representations of the three context subsequences respectively, and the second one computes semantic composition of these three representations and produces a sentence representation for the relationship classification of the two nominals. (3) The attention mechanism is adopted in both RNNs to encourage the model to concentrate on the important information when learning the sentence representations. Experimental results on the SemEval-2010 Task 8 dataset demonstrate that our model is comparable to the state-of-the-art without using any hand-crafted features.

1 Introduction

Semantic relation classification is an important task in natural language processing, which has attracted great attention in recent years. The goal is to identify the semantic relationship between a pair of nominals marked in a sentence. For instance, in the sentence "The software $[\text{company}]_{e_1}$ addressed the problem with the $[\text{publication}]_{e_2}$ of a fix on Saturday", the marked nominals of *company* and *publication* are of relationship *Product-Producer(e_2, e_1)*. Most conventional models focus on machine learning and feature design, which have been shown to obtain performance improvements (Kambhatla, 2004; Tratz and Hovy, 2010; Rink and Harabagiu, 2010).

Recently, neural network approaches have been widely used for relation classification, which aim at reducing the need of hand-crafted features. These approaches are broadly divided into two categories: one explores the effectiveness of using dependency paths and its attached subtrees between two nominals, and various neural networks are adopted to model the shortest dependency paths and dependency subtrees (Xu et al., 2015a; Xu et al., 2015b; Liu et al., 2015); the other exploits deep neural networks to learn syntactic and semantic features from raw sentences (Zeng et al., 2014; Dos Santos et al., 2015; Zhang et al., 2015), which has been proved effective, but inevitably suffers from irrelevant parts. Our paper introduces an attentive neural network that selectively focuses on useful information on raw sentences.

Context information of the annotated nominals has been widely believed to be useful for relation classification (Zhang et al., 2015; Thang Vu et al., 2016). In this work, we further explore the effectiveness of context information around the annotated nominals in a sentence. In our model, a sentence with two marked nominals is divided into three context subsequences according to two marked nominals: the left context subsequence, the middle context subsequence and the right context subsequence. This method is similar to Pei et al. (2015) and Thang Vu et al. (2016), which have showed that contextual information is effectively obtained by deep learning techniques. Instead of combining the middle context

*Cong Liu is the corresponding author.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1254–1263, Osaka, Japan, December 11-17 2016.

subsequence with the left and right context subsequences, respectively, as in (Thang Vu et al., 2016), we propose to learn context representations via recurrent neural networks that work on each context subsequence independently. For example, the sentence "The software $[\text{company}]_{e_1}$ addressed the problem with the $[\text{publication}]_{e_2}$ of a fix on Saturday" is split into three subsequences: " The software", "addressed the problem with the" and "of a fix on Saturday". And the marked nominals of $[\text{company}]_{e_1}$ and $[\text{publication}]_{e_2}$ are not included in any context subsequence. As a result, the sentence is divided into five parts: three context subsequences and two annotated nominals. Our sentence representations are leant hierarchically from context subsequences to sentences using a hierarchical recurrent neural network, which firstly learns the context representation of each context subsequence independently, and then encodes the semantics of context subsequences into a sentence representation for the relation classification. Furthermore, we introduce the attention mechanism (Bahdanau, 2014; Rush, 2015; Rocktäschel et al., 2016) that encourages the model to focus on the important information. Experimental results demonstrate that our model is comparable to the state-of-the-art with a single model that works on the raw sentences.

In the rest of this paper, we review recurrent neural networks in Section 2. We provide details about our model in Section 3. Section 4 presents our experiments and their results. Finally, we make a conclusion in Section 5.

2 Recurrent Neural Networks

Recurrent neural networks (RNNs) (Elman, 1990; Mikolov et al., 2010) project a sequence of inputs $x_1, \ldots, x_T$ to a sequence of outputs $y_1, \ldots, y_T$ via an affine transformation followed by a non-linear function. At timestep *t*, a standard RNN computes the new hidden vector as

$$h_t = f(Wx_t + Uh_{t-1} + b) \tag{1}$$

where *W* is trained matrix transforming the current input x_t into the current state linearly, *U* is also trained matrix connecting the previous state h_{t-1} with the current state, and *b* is a bias term, and *f* is a non-linear function (e.g., tanh).

However, RNNs with the above form may suffer from gradient *exploding* or *vanishing* problem (Bengio et al., 1994; Hochreiter, 1997) during training when it is trained with the backpropagation through time algorithm (Rumelhart et al., 1986; Werbos, 1990; Williams and Zipser, 1995). To address this problem, long short-term memory network (LSTM) was proposed in (Hochreiter and Schmidhuber, 1997) where the architecture of a standard RNN was modified to avoid vanishing or exploding gradients. Many LSTM variants have been proposed, and here we adopt the version of Zaremba and Sutskever (2014a).

The LSTM model comprises a memory cell that can store information over a long period of time, and three gates that allow it to control the flow of information into and out of the cell: input gate, forget gate, and output gate. Concretely, the LSTM unit at time step *t* encompasses a collection of vectors: an input gate i_t, a forget gate f_t, an output gate o_t, a memory cell c_t, and a hidden state h_t. The unit accepts an input vector x_t, the previous hidden state h_{t-1}, and the memory cell c_{t-1} and computes the new vectors using the following equations:

$$\begin{aligned}
i_t &= \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}) \\
f_t &= \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) \\
o_t &= \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}) \\
u_t &= tanh(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}) \\
c_t &= i_t \odot u_t + f_t \odot c_{t-1} \\
h_t &= o_t \odot tanh(c_t)
\end{aligned} \tag{2}$$

where σ denotes the element-wise application of the logistic function, $\odot$ denotes the element-wise multiplication of two vectors, *W* and *U* are weight matrices, and *b* are bias vectors.

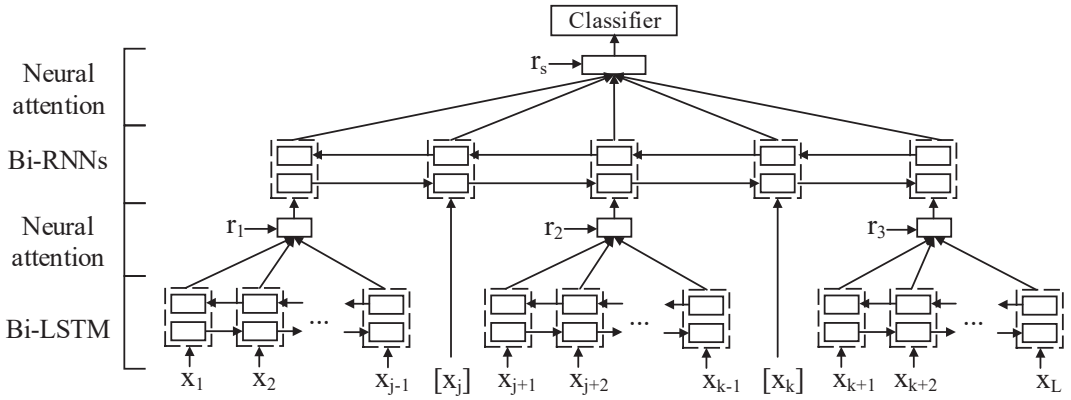


Figure 1: The architecture of our model. Given a sentence consisting of *L* words, it is divided into the left context subsequence $[x_1, \ldots, x_{j-1}]$, the middle context subsequence $[x_{j+1}, \ldots, x_{k-1}]$ and the right context subsequence $[x_{k+1}, \ldots, x_L]$. $[x_j]$ and $[x_k]$ represent the marked nominals e_1 and e_2 respectively.

3 Model

In this section, we introduce the proposed neural model that learns distributed representations from raw sentences. These representations serving as features are further used for relation classification. An overview of our model is shown in Figure 1.

Given a sentence with two annotated nominals, the sentence is firstly divided into five parts (three context subsequences and two annotated nominals) based on the two marked nominals (Section 3.1). Next, the model computes the distributed representations for the context subsequences using a bidirectional LSTM that works on word vectors (Section 3.2). Lastly, these distributed context representations are further encoded into a sentence representation via a bidirectional RNN (Section 3.3). Furthermore, we extend this model with a neural attention that encourages the model to focus on important information.

3.1 Context Subsequences

In most cases different contexts have different functions for the meaning of sentences. Some recent work fell into the idea that the middle context contains the most relevant information for relation classification, combining the middle context with the left and right context respectively (Zhang et al., 2015; Thang Vu et al., 2016). We instead model each context part independently, which allows the model to automatically identify contexts that contain useful information.

Given a sentence *s* and its annotated nominals e_1 and e_2, the sentence first is split into five parts according to the two annotated nominals: the left context subsequence, entity e_1, the middle context subsequence, entity e_2 and the right context subsequence. Preprocessing the sentence in such a way allows the model to encode each context subsequence independently.

3.2 Context Subsequence Composition

3.2.1 Word Encoder

A bidirectional LSTM (Bi-LSTM) (Graves and Schmidhuber, 2005; Graves et al., 2013) is applied to independently encoding each of the three context subsequences. A bidirectional LSTM consists of two LSTMs: the forward and backward LSTMs. They are run in parallel: the forward LSTM inputs the words from x_1 to x_T, and the backward LSTM inputs in an reverse order from x_T back to x_1. At time step *t*, we obtain the hidden state (denoted as h_t) of the bidirectional LSTM by concatenating the forward hidden state (denoted as $\overrightarrow{h_t}$) and the backward one (denoted as $\overleftarrow{h_t}$), i.e., $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$. Bi-LSTM can summarize the information from the whole context subsequence centered around words, which let the model understand the meaning of words comprehensively.

Given a sentence *s* divided into the left context subsequence c_1, the middle context subsequence c_2 and the right context subsequence c_3, we assume that the sentence *s* contains *L* words and the context subsequence c_i has T_i words, where *i* $\in [1, 3]$. The input to Bi-LSTM is a context subsequence c_i: [x_{i1}, ..., x_{iT_i}] where x_{it} is the word vector for word w_{it}. At time step *t*, the encoder produces a hidden state h_{it} which gathers the information of the whole context subsequence c_i centered around w_{it}. The equations are following:

$$\overrightarrow{h_{it}} = \overrightarrow{LSTM}(x_{it}) \tag{3}$$

$$\overleftarrow{h_{it}} = \overleftarrow{LSTM}(x_{it}) \tag{4}$$

$$h_{it} = concat(\overrightarrow{h_{it}}, \overleftarrow{h_{it}}) \tag{5}$$

where *concat* is concatenation function, i.e., h_{it} = [$\overrightarrow{h_{it}}$; $\overleftarrow{h_{it}}$].

Note that our model encodes the left, middle and right context subsequence independently but with one Bi-LSTM. [1]

3.2.2 Word-level Attention

Due to the fact that raw sentences contain more information than the shortest dependency paths, there may be some irrelevant information in raw sentences. For concentrating on these words that are important to predict the relationship of entities, it can be a good strategy to pay more attention on these words. To encourage such behavior, this paper introduces a word-level attention mechanism. The attention mechanism enables the model to differently attend over the hidden vectors of Bi-LSTM along a context subsequence, and produces a weighted representation m_i of them as follows:

$$\begin{aligned} z_{it} &= tanh(W^{(w)} h_{it} + W^{(c)} r_i + b) \\ \alpha_{it} &= \frac{exp(v_z^\top z_{it})}{\sum_{j=1}^{T_i} exp(v_z^\top z_{ij})} \\ m_i &= \sum_{t=1}^{T_i} \alpha_{it} h_{it} \end{aligned} \tag{6}$$

where $W^{(w)}$ and $W^{(c)}$ are weight matrices, *b* is a bias vector, v_z is a weight vector and $v_z^\top$ is its transformation, and r_i is an external context vector that is randomly initialized and jointly optimized during training.

The attention representation z_{it} corresponding to the *t*-th word w_{it} in the context subsequence c_i is computed via a non-linear combination of the hidden state h_{it} and the external context vector r_i. The attention weight α_{it} for the *t*-th word w_{it} in the context subsequence c_i is a probability that is the normalized weight of z_{it} (parameterized by v_z) through a softmax layer, reflecting the importance of the *t*-th word w_{it} with respect to the meaning of the context subsequence c_i in classifying the relationship of two entities. The external context vector r_i not only represents the high-level meaning of the context subsequence c_i, but also allows the model to identify that the word w_{it} is in the context subsequence c_i.

3.3 Sentence Composition

After establishing an attention-based Bi-LSTM (Section 3.2) to capture the meaning of three context subsequences, resulting in three context representations, there is one difficulty that how to further obtain the semantic composition of these context representations plus two representations of marked nominals. Note that there are five semantic representations. The most common approach is that a multilayer perceptron (MLP) is adopted to take these representations as input and compute semantic compositionality

[1] We adopt Bi-LSTM to encode each context subsequence separately even if it contains few words, such as one word.

for them. In this work, we adopt a Bi-RNN to integrate syntactics and semantics of three context subsequences and two annotated nominals into sentence representation *s*, which is further fed into a classifier for relation classification. We propose to learn sentence representations via Bi-RNNs for two reasons: (1) a sentence containing two annotated nominals divided into three context subsequences that are ordered as in the sentence, can be treated as a short sequence that consists of five tokens; (2) recurrent neural networks are competent enough to model the semantics of these context subsequences and their inherent relations, which is important to obtain the semantic meaning of the sentence. The experimental results demonstrate that Bi-RNNs significantly outperform MLP.

Let Y be a matrix containing five column vectors [m_1, m_{e_1}, m_2, m_{e_2}, m_3], where m_i ($i \in [1, 3]$) is the representation of the context subsequence c_i, and m_{e_1} and m_{e_2} are the representations of annotated nominals e_1 and e_2. [2] To obtain compositional vector representations for sentences, we iterate the following sequence of equations:

$$\overrightarrow{h_j} = \overrightarrow{RNN}(y_j) \tag{7}$$

$$\overleftarrow{h_j} = \overleftarrow{RNN}(y_j) \tag{8}$$

$$h_j = concat(\overrightarrow{h_j}, \overleftarrow{h_j}) \tag{9}$$

where $y_j \in Y$ (j $\in [1, 5]$) is the *j*-th column vector in Y.

Note that the sentence only contains five elements, our model do not make any assumptions about the type of RNNs used in this subsection. But as far as comparison goes, LSTMs performs better than the standard RNNs.

To selectively focus on the important context subsequences, it is an alternative solution to applying neural attention to the hidden vectors of the above Bi-RNNs, similar to Subsection 3.2.2. We also make further extensions such as average pooling and max pooling.

3.4 Training

A fully connected softmax layer is used as classifier for classification. It produces the probability distribution *p* over relation types conditioned on the sentence representation *s*:

$$p = softmax(W^{(s)}s + b^{(s)}) \tag{10}$$

The training objective is to minimize the cross-entropy error between the ground truth and predicted label. The parameters of our model are optimized using AdaGrad (Duchi et al., 2011) with a learning rate of 0.01, a mini-batch size of 5 and a L_2 regularization coefficient of 10^{-6}. The details are described further in Section 4.2.

4 Experiments and Evaluation

4.1 Dataset

In our experiments, we evaluate our model on the SemEval-2010 Task 8 dataset (Hendrickx et al., 2010), which is one of the most widely used benchmarks for relation classification. The dataset contains 10,717 annotated sentences divided into 8,000 sentences for training and 2717 for testing. Each sentence is annotated with each of nine different relationship and an artificial relation *Other*, and each relationship has two direction except for the undirected relation *Other*. The nine directed relations are *Cause-Effect, Instrument-Agency, Product-Producer, Content-Container, Entity-Origin, Entity-Destination, Component-Whole, Member-Collection, and Message-Topic*.

The official evaluation metric is the macro-averaged F1-score (excluding *Other*), and takes into consideration the directionality. We use the official scorer to test the model performance.

[2] We use an additional *tanh* layer to map the word vectors of annotated nominals e_1 and e_2 to the dimensionality of the hidden size of the Bi-LSTM.

4.2 Implementation

We tune the hyperparameters for our model using 5-fold cross-validation. We pretrain 200-dimensional word embeddings using *word2vec* (Mikolov et al., 2013) on the English Wikipedia corpus, and randomly initialize other hyperparameters. We set the LSTM dimension to be 200. We apply dropout only on the word embeddings and outputs of LSTM as in (Zaremba et al., 2014b), and the dropout rate is 0.2.

To enable a direct comparison with the previous work, we use the same features: position features, WordNet hypernyms and NER. WordNet hypernyms and NER were obtained using the tool of Ciaramita and Al-tun (2006). [3]

4.3 Results

Model	features	F1
Bi-LSTM + MLP	-	82.43
	+ all features	83.30
Bi-LSTM + Bi-RNN	-	82.67
	+ all features	82.92
Bi-LSTM + Bi-LSTM	-	83.90
	+ all features	84.27

(a) The effect of neural network architectures.

Method	features	F1
Concatenation	-	79.66
	+ all features	80.56
Average	-	79.91
	+ all features	81.39
Max-Pooling	-	81.67
	+ all features	82.48
Attention	-	83.90
	+ all features	84.27

(b) Comparison of different methods

Table 1: (a) F1-scores on the test data for various neural network architectures. We also test these models with three features of position features, WordNet and NER. (b) The comparison of different methods on SemEval-2010 Task 8 test set. Here the neural network architecture is the combination of two Bi-LSTMs.

4.3.1 The Effect of Different Components

The effect of neural network architectures We first analyze the effect of different neural network architectures of the combinations of Bi-LSTM with MLP, a standard Bi-RNN and Bi-LSTM separately. Here we apply neural attention to the hidden states of RNNs (Bi-LSTM and Bi-RNN). To ensure the number of parameters comparable, we adopt a two-layer full-connected neural network with the hidden size of 600 dimension and a non-linear function of *tanh* to serve as MLP. And the hidden size of the standard RNN is 350-dimensional. From Table 1a, we find that both the combinations of Bi-LSTM with Bi-RNN and Bi-LSTM outperform the combination of Bi-LSTM and MLP without any features. In particular, the combination of Bi-LSTM and Bi-LSTM achieves the best result 83.90% without any feature, and its F1-score is about 1.5% higher than the model of Bi-LSTM+MLP. The results indicate that the neural architecture of two Bi-LSTMs effectively captures semantic meanings of these context subsequences and their inherent relations, and obtains more robust sentence representations for relation classification. In this paper, we tackle the relation classification task using the combination of two Bi-LSTMs.

The comparison of different methods Table 1b shows experiments for our model with various methods for the hidden vectors of Bi-LSTMs. We begin with the model using the concatenation of the final state of forward and backward LSTMs. And then we replace concatenation operation with average pooling, max-pooling and neural attention respectively. Not surprisingly, processing the hidden vectors of Bi-LSTMs via neural attention achieves the best result, which gives an improvement of 2.23 percentage points in F1-score over max-pooling. We suspect that this is due to the attention model being run in a more focused way that makes it easier to capture large important information from contexts. We also consider the impact of features for these methods. Results in Table 1b show that by adding features the F1-scores of all methods improve, which hints that three features are useful for relation classification.

[3] sourceforge.net/projects/supersensetag/

Model	Feature Set	F1
SVM	+POS, WordNet, prefixes and other morphological features, dependency parse, Levin classes, PropBank, FanmeNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2
MV-RNN	- +POS, NER, WordNet	79.1 82.4
FCM	- +dependency parsing, NER	80.6 83.0
CNN	- +position features, words around nominals, WordNet	69.7 82.7
BLSTM	- +POS, NER, WordNet, position features, dependency feature, relative-dependency feature	82.7 84.3
DepNN	+WordNet +NER	83.0 83.6
SDP-LSTM	- +POS embeddings, WordNet embeddings, grammar relation embeddings	82.4 83.7
depLCNN + NS	- +WordNet, words around nominals	84.0 **85.6**
Our model	- +position features, WordNet, NER	83.9 84.3

Table 2: Experimental results of our model against other models.

4.3.2 Comparison with State-of-the-art Models

Table 2 compares our model with several start-of-art models. The SVM model (Rink and Harabagiu, 2010) is used for relation classification by combining lexical and semantic features. It extracts these hand-crafted features from sentences with the use of many external resources. Socher et al. (2012) extend the recursive neural networks with matrix-vector spaces (MV-RNN), and use MV-RNN to learn representations along the constituency tree for relation classification. Yu et al. (2014) propose factor-based compositional embedding models (FCM) for relation classification. It learns representations for the substructures of an annotated sentence, which are further used for classification. Zeng et al. (2014) exploit a convolutional neural network (CNN) to extract lexical and sentence level features for relation classification. And they design position features to specify the target nouns in the sentence, which leads to better performance for their model. CR-CNN outperforms the state-of-art by using a new ranking loss function and omitting the representation of the *Other* class for diminishing its effect, as proposed by (Dos Santos et al., 2015). Zhang et al. (2015) utilized bidirectional LSTMs (BLSTM) to capture the sentence level features and concatenated them and lexical level features to form the finally feature vector for relation classification. Liu et al. (2015) design a dependency-based framework (DepNN) to learn semantic representations of the augmented dependency paths that are the combination of the shortest dependency paths and their dependency subtrees. Xu et al. (2015a) build multiple LSTMs to model the different channels of word vectors, POS, grammatical relations, and WordNet along the shortest dependency paths and achieves an F1-score of 83.7 (SDP-LSTM). Xu et al. (2015b) propose to learn a robust representation using a convolutional neural network that works on the dependency path between subjects and objects, and propose a negative sampling strategy (NS) to address the relation directionality (DepLCNN). Thang Vu et al. (2016) design extended middle context and present a new context representation for convolutional neural networks for relation classification (ER-CNN). And they also propose connectionist bi-directional recurrent neural networks (R-RNN) that adds a connection to the hidden states of bi-directional recurrent neural networks.

We observe in Table 2 that our model is comparable to the state-of-the-art (previous best result is 84.0% obtained by depLCNN + NS) without any features, whereas depLCNN works on the shortest dependency

Model	Feature Set	F1
CR-CNN	-	82.8
	+position features	84.1
R-RNN	+position features, position indicators, entity flag	83.4
ER-CNN	+position features, extended middle context	84.2
ER-CNN + R-RNN	+all features, voting scheme	84.9
Our model	-	84.1
	+position features	84.5

Table 3: Comparison of ranking models (no lexical features).

paths, which consist of most relevant information and avoid negative effect from irrelevant parts in the sentences. This result suggests that our model automatically focuses on important information related to determining the relationship of two entities. The F1-score is improved by adding three features but not as obvious as in (Zeng et al., 2014; Xu et al., 2015b) (CNN, depLCNN). We argue that this is due to Bi-LSTMs being able to learn position information on sequences and lexical features leading to overfitting as in (Yu et al., 2014; Liu et al., 2015).

4.3.3 Comparison of ranking models

For fair comparison, we also replace the softmax layer with a ranking layer to train our model, as proposed in (Dos Santos et al., 2015). We use training settings following Thang Vu et al. (2016). More details about ranking layer are described in (Dos Santos et al., 2015; Thang Vu et al., 2016).

From Table 3, we observe that our model outperform the state-of-the-art without any feature, whereas previous work's best reported performance is 83.9% in ER-CNN using word embeddings of size 400. Combining ER-CNN and R-RNN using a voting scheme achieves a state-of-the-art result of 84.9 in F1-score, which is presented by (Thang Vu et al., 2016). But our model reaches a new state-of-the-art result with a single model when position features are added, and outperforms the model of ER-CNN that learns context representations for two contexts of the combinations of the middle context with the left and right context respectively.

5 Conclusion

In this work, we introduce a hierarchical recurrent neural network model that learns useful features from raw sentences for relation classification. We further extend the model with neural attention at two different levels that provides significant improvements over the concatenation, average pooling and max-pooling. Our model shows comparable performance to the state-of-the-art on the SemEval-2010 Task 8 dataset without using any costly hand-crafted features. In addition, the models presented here are general hierarchical models, and are therefore suitable for hierarchical structures, such as paragraphs and documents.

Acknowledgements

This work was partially supported by National Science Foundation of China (grant 61472459). We thank the anonymous reviewers for their insightful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks 5(2):157–166*.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research, 12:2121–2159*.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science, 14(2):179–211*.

Alex Graves, Navdeep Jaitly, and A-R Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. *In IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU). pages 273–278*.

A. Graves and J. Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Net- works, 18(5):602–610*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, PreslavNakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. *In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 33–38. Association for Computational Linguistics*.

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

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6(02):107–116*.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. *In Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions, page 22. Association for Computational Linguistics*.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, Houfeng Wang. 2015. A Dependency-Based Neural Network for Relation Classification. *InProceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), pages 285–290. Association for Computational Linguistics*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan "Honza" Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. *In Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010), pages 1045–1048*.

Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *In Proceedings of the Workshop at ICLR*.

Cícero Nogueira Dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 626–634. Association for Computational Linguistics*.

Wenzhe Pei; Tao Ge; Baobao Chang. 2015. An Effective Neural Network Model for Graph-based Dependency Parsing. *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 313–322. Association for Computational Linguistics*.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. *In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 256–259. Association for Computational Linguistics*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Phil Blunsom. 2016. Reasoning about Entailment with Neural Attention. *In Proceedings of ICLR2016*.

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

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 379–389. Association for Computational Linguistics*.

M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Sig- nal Processing, 45(11):2673–2681*.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. *In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1201–1211. Association for Computational Linguistics*.

Ngoc Thang Vu, and Heike Adel and Pankaj Gupta and Hinrich Schütze. 2016. Combining Recurrent and Convolutional Neural Networks for Relation Classification. *In Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Stephen Tratz and Eduard Hovy. 2010. Isi: automatic classification of relations between nominals using a maximum entropy classifier. *In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 222–225. Association for Computational Linguistics*.

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

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

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, Zhi Jin. 2015a. Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Paths. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1785–1794. Association for Computational Linguistics*.

Kun Xu, Yansong Feng, Songfang Huang and Dongyan Zhao. 2015b. Semantic Relation Classification via Convolutional Neural Networks with Simple Negative Sampling. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 536–540. Association for Computational Linguistics*.

Mo Yu, Matthew Gormley, and Mark Dredze. 2014. Factor-based compositional embedding models. *In Proceedings of the NIPS Workshop on Learning Semantics*.

Wojciech Zaremba and Ilya Sutskever. 2014a. Learning to execute. *arXiv preprint arXiv:1410.4615*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014b. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. *In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 2335–2344*.

Shu Zhang, Dequan Zheng, Xinchen Hu, Ming Yang. 2015. Bidirectional Long Short-Term Memory Networks for Relation Classification. *In Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

A Unified Architecture for Semantic Role Labeling and Relation Classification

Jiang Guo[♪], **Wanxiang Che**[♪], **Haifeng Wang**[♩], **Ting Liu**[♪] and **Jun Xu**[♪]
[♪]Center for Social Computing and Information Retrieval, Harbin Institute of Technology, China
[♩]Baidu Inc., China
{jguo, car, tliu, jxu}@ir.hit.edu.cn
wanghaifeng@baidu.com

Abstract

This paper describes a unified neural architecture for identifying and classifying multi-typed semantic relations between words in a sentence. We investigate two typical and well-studied tasks: *semantic role labeling* (SRL) which identifies the relations between predicates and arguments, and *relation classification* (RC) which focuses on the relation between two entities or nominals. While mostly studied separately in prior work, we show that the two tasks can be effectively connected and modeled using a general architecture. Experiments on CoNLL-2009 benchmark datasets show that our SRL models significantly outperform state-of-the-art approaches. Our RC models also yield competitive performance with the best published records. Furthermore, we show that the two tasks can be trained jointly with multi-task learning, resulting in additive significant improvements for SRL.

1 Introduction

Semantic relation identification and classification are important problems towards the understanding of natural language sentences. Multi-typed semantic relations have been defined between two terms in a sentence in natural language processing (NLP) to promote various applications. For instance, the task of *Semantic Role Labeling* (SRL) defines shallow semantic dependencies between arguments and predicates, identifying the semantic roles, e.g., *who did what to whom, where, when, and how*. SRL has been a long-standing and challenging problem in NLP, primarily because it is strongly dependent on rich contextual and syntactical features used by the underlying classifiers (Gildea and Jurafsky, 2002). Another instance is *Relation Classification* (RC) which assigns sentences with two marked entities (or nominals) to a predefined set of relations (Hendrickx et al., 2010). Compared with SRL, relations defined in RC express much deeper semantics. Figure 1 shows example annotations of SRL and RC respectively.

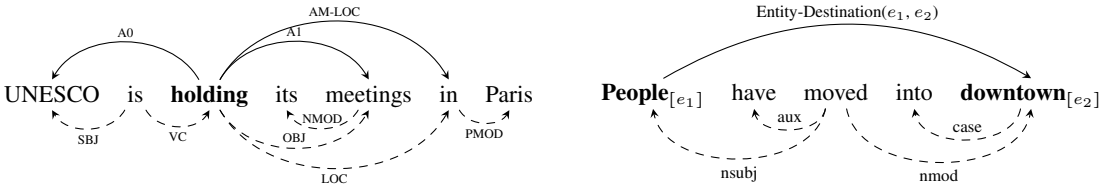
These two problems are typically studied separately in different communities. Hence the connections between them are neglected, both in data resources and approaches. In this paper, we show that SRL and RC have a lot of common ground and can be modeled with a unified model. We start by looking into the key features which have been proven dominant in both SRL and RC.

- Contextual features. Words within a proper window size of the target words are important for most statistical models of various NLP tasks, such as Part-of-Speech tagging, Named Entity Recognition and Parsing. They are also important for identifying the semantic relatedness between two terms in a sentence. Consider the RC example in Figure 1(b), the context word "*moved*" is a strong indicator for classifying the relation of ⟨**People**, **downtown**⟩ as *Entity-Destination*. However, most of the conventional approaches in SRL and RC only considers local context features through feature engineering, which might be incomplete.

- Syntactical features. Both state-of-the-art SRL and RC systems employ the syntactic path between the two target terms as an important feature. Figure 1 shows the dependency parses for

† Corresponding author: Wanxiang Che

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1264–1274, Osaka, Japan, December 11-17 2016.



(a) Semantic Role Labeling. (b) Relation Classification.

Figure 1: Examples of *semantic role labeling* (a) and *relation classification* (b).

the two sentences. For example, the dependency path between "meetings" and the predicate "**holding**" ("holdings" $\xrightarrow{OBJ}$ "meetings") strongly indicates an A1 relation (*patient* role). Early approaches built in discrete feature space are not capable of utilizing the *word path* features which are extremely sparse. Fortunately, recent progress in distributed representations and deep neural networks provides a promising solution for this problem.

- Lexical semantic features. Lexical properties of a word (e.g., the identity of a word, its lemma, its morphological features) are important for semantic tasks. Particularly in tasks like relation classification, it is often impossible to determine the relation without the semantic ground of the target words. Therefore, previous approaches have been using lexical features like word embeddings, lemmas, WordNet, etc.

This paper describes a unified neural model for SRL and RC that effectively utilizes the three kinds of features above. Our model captures global contextual features and syntactic path features by using bidirectional long short-term memory (LSTM)-based recurrent neural networks. We especially focus on SRL which, in our opinion, is more complicated and difficult. SRL is a structure prediction task with certain structural constraints. To this end, an additional post-inference procedure based on integer linear programming (ILP) is applied to SRL, in order to meet the constraints. Furthermore, our unified model successfully connects SRL and RC, presenting the possibility of *multi-task learning*. We show that the SRL performance can be significantly improved through knowledge transfer from RC.

We conduct experiments on the CoNLL-2009 shared task datasets for SRL, and the SemEval-2010 Task 8 dataset for RC. On SRL, our models significantly outperform previous approaches in various languages. On RC, our model also obtains performance competitive to the state-of-the-art.[1]

Our primary original contributions include:

- We propose a unified model for SRL and RC, which effectively captures global contextual features, syntactical features and lexical semantic features.
- We show that SRL can be significantly improved by jointly training with RC, reaching new state-of-the-art performance.

2 Related Work

The present work ties together several strands of previous studies.

Semantic Role Labeling A great deal of previous SRL research has been dedicated to designing rich and expressive features, pioneered by Gildea and Jurafsky (2002). For instance, the top performing system on the CoNLL-2009 shared task employs over 50 language-specific feature templates (Che et al., 2009). These features mostly involve the predicate, the candidate argument, their contexts and the syntactic path between them (Surdeanu et al., 2003; Xue and Palmer, 2004; Pradhan et al., 2005). Besides, higher-order features involving several arguments or multiple predicates have also been explored (Toutanova et al., 2008; Martins and Almeida, 2014; Yang and Zong, 2014).

[1]Our code is available at: `https://github.com/jiangfeng1124/nnsrl-rc`.

Several approaches have been studied to alleviate the intensive feature engineering in SRL and get better generalization. Moschitti et al. (2008) introduce different kinds of tree kernels for capturing the structural similarity of syntactic trees. While attractive in automatic feature learning, the kernel-based approaches typically suffer from high computational cost. Lei et al. (2015) instead use low-rank tensors for automatic feature composition based on four kinds of basic feature sets. However, tensor-based approaches cannot well generalize the high-sparsity structural features like syntactic path. Besides, they still need a relatively small amount of feature engineering to make use of the local contexts. Another line of research focuses on neural models (Collobert et al., 2011; Zhou and Xu, 2015; FitzGerald et al., 2015), which have shown great effectiveness in automatic feature learning on a variety of NLP tasks. Most recently, Roth and Lapata (2016) employ LSTM-based recurrent neural networks to obtain the representations of syntactic path features, which is similar to our work. Aside from the distributed path features, they also use a set of binary input feature sets from Anders et al. (2010). In contrast to these prior work, our model jointly leverages both global contexts and syntactic path features using bidirectional LSTMs.

Relation Classification Early research on RC has also been relying heavily on human-engineered features (Rink and Harabagiu, 2010). Recent years have seen a great deal of work on using neural networks to alleviate the intensive engineering on contextual and syntactic features. For example, Socher et al. (2012) propose recursive neural networks for modeling the syntactic paths between the two entities whose relation is to be determined. Zeng et al. (2014) use convolutional neural network for learning sentence-level features of contexts and obtain good performance even without using syntactic features. Later approaches have used more sophisticated models for better handling long-term dependencies, such as sequential LSTMs and tree LSTMs (Liu et al., 2015; Xu et al., 2015b; Miwa and Bansal, 2016). In addition, Yu et al. (2014) and (2015) investigate tensor-based approaches for learning the combination of embedding features and lexicalized sparse features.

Therefore, despite that relation classification has mostly been studied separately from SRL, they have a substantial amount of commonalities. It inspires us to develop a potentially unified architecture to take advantage of the progress in each research direction.

Multi-task Learning There has been a line of research on joint modeling pipelined NLP tasks, such as word segmentation, POS tagging, parsing and semantic role labeling (Hatori et al., 2012; Li et al., 2011; Bohnet and Nivre, 2012; Henderson et al., 2013; Lluís et al., 2013). Most multi-task learning or joint training frameworks can be summarized as parameter sharing approaches proposed by Ando and Zhang (2005). In the context of neural modeling for NLP, the most notable work was proposed by Collobert and Weston (2008), which aims at solving multiple NLP tasks within one framework by sharing common word embeddings. This work also inspires us in this study to develop a unified architecture for SRL and RC in prior to joint training.

Recently, the idea of neural multi-task learning was applied to sequence-to-sequence problems with recurrent neural networks. Dong et al. (2015) use multiple decoders in neural machine translation systems that allows translating one source language to many target languages. Luong et al. (2015) study the ensemble of a wide range of tasks (e.g., syntactic parsing, machine translation, image caption, etc.) with multi-task sequence-to-sequence models. Liu et al. (2016) incorporate different kinds of corpus for implicit discourse relation classification using multi-task neural networks. More recently, multi-task learning has also been applied to sentence compression (Klerke et al., 2016) and machine translation quality estimation (Shah and Specia, 2016).

3 Problem Definition

This section gives formal definitions of the two tasks to be investigated: SRL and RC.

3.1 Semantic Role Labeling

We follow the setup of the CoNLL-2009 shared task. Given a sentence $\mathbf{s}$, each token is annotated with a predicated POS tag and predicted word lemma. Some tokens are also marked as predicates. Besides,

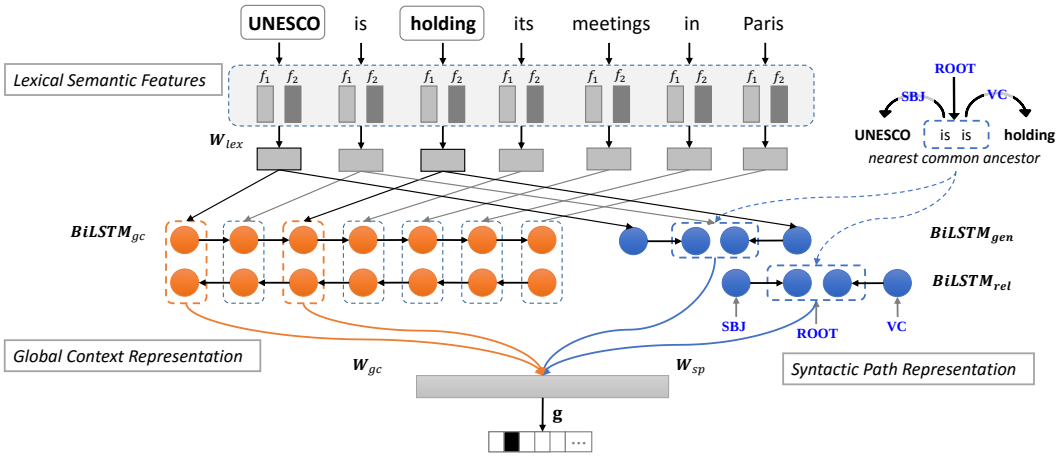


Figure 2: The unified architecture for SRL and RC.

a predicted syntactic dependency tree y_{syn} is also provided (cf. below part of Figure 1(a)). The goal is to determine the semantic dependencies for each predicate p_i (cf. upper part of Figure 1(a)). These dependencies identify the arguments of each predicate and the role labels.

In this work, we focus on the identification and classification of the arguments associated with given predicates. More formally, for each predicate p_i in $\mathbf{s}$, we loop over all the tokens in $\mathbf{s}$ except p_i: $\{w \in \mathbf{s} | w \neq p_i\}$, and determine their role labels. It can be considered as a classification problem with each instance as a word pair $\langle p_i, w \rangle$. We include an additional *NULL* label indicating that a token is not an argument of p_i. To guarantee the resulting semantic dependencies meet certain constraints, we further apply ILP over the output probabilities in each position for post-inference (Section 4.4).

3.2 Relation Classification

As demonstrated in Figure 1, the semantic relations specified in relation classification are totally different from SRL. SRL is more close to the syntactic dependencies while RC is totally semantic. Our setup follows the SemEval-2010 Task 8. Each sentence $\mathbf{s}$ is annotated with a pair of nominals e_1 and e_2, and our goal is to identify the relation between e_1 and e_2. Nine relations are defined in the task, and the directionality of relation between e_1 and e_2 is considered in the evaluation. Relations that do not belong to the nine relations are marked as *Other*.

4 Unified Neural Architecture

As described above, both SRL and RC can be formalized as a classification problem over instances of word pairs within a sentence. We propose a unified neural architecture, as illustrated in Figure 2, for modeling these two tasks. Our architecture includes the following three primary components.

4.1 Lexical Feature Representation

We extract basic lexical features for each token in a sentence. Typical lexical features for SRL and RC include word (or *lemma* when available) and POS tag. For RC, additional features can be used, such as *named entity type* (NE) and WordNet. All these features are then represented as low-dimensional real-valued vectors, i.e., feature embeddings. Word embeddings can be readily pretrained using *word2vec* on a large unlabeled corpus, which have proved helpful in many applications. Next, various feature embeddings are composed through a nonlinear transformation, and thus a token can be represented as:

$$\begin{aligned} &\mathbf{x}_i = \texttt{ReLU}(\mathbf{W}_{lex}\mathbf{\Phi}_i + \mathbf{b}_{lex}), where \\ &\mathbf{\Phi}_i = [\mathbf{w}_i; \mathbf{p}_i] \ \ for \ \ \textbf{SRL}, \ \ \mathbf{\Phi}_i = [\mathbf{w}_i; \mathbf{p}_i; \mathbf{ne}_i; \mathbf{wn}_i] \ \ for \ \ \textbf{RC} \end{aligned} \tag{1}$$

$\mathbf{w}_i$ represents the word or lemma (when available), $\mathbf{p}_i$ represents the POS tag, $\mathbf{ne}_i$ is the named entity, and $\mathbf{wn}_i$ is the WordNet hypernym.

4.2 Global Context Representation

We obtain global context representations of the target words by using bidirectional LSTM-based RNNs. For more computation details of LSTM, we refer the readers to Hochreiter and Schmidhuber (1997). The LSTMs take as input the token representation $\mathbf{x}_i$ in each position. The hidden state vectors of the two directions' LSTM units corresponding to each target word are then concatenated as its global context representation:

$$\mathbf{R}^{gc}_{e_1} = [\overrightarrow{\mathbf{h}}_{e_1}; \overleftarrow{\mathbf{h}}_{e_1}]; \quad \mathbf{R}^{gc}_{e_2} = [\overrightarrow{\mathbf{h}}_{e_2}; \overleftarrow{\mathbf{h}}_{e_2}] \tag{2}$$

Note that an important difference between our model and previous neural models is that we utilize the hidden state vectors of e_1 and e_2 instead of the representation of the whole sentence, which frees us from using position-related features (Zeng et al., 2014; Collobert et al., 2011; dos Santos et al., 2015).

4.3 Syntactic Path Representation

We define the *nearest common ancestor* token of e_1 and e_2 as $nca(e_1, e_2)$. Then the path from e_1, e_2 to $nca(e_1, e_2)$, i.e., $e_1 \rightarrow \ldots \rightarrow nca(e_1, e_2)$ and $nca(e_1, e_2) \leftarrow \ldots \leftarrow e_2$, are also modeled with bidirectional LSTMs, as shown in Figure 2 (right panel). We use two kinds of syntactic paths, including a *generic path* that takes the token representation $\mathbf{x}_i$ as input, and a *relation path* that takes the dependency relations along the path as input (Figure 2). These two paths are modeled with $\mathbf{BiLSTM}_{gen}$ and $\mathbf{BiLSTM}_{rel}$ respectively. The hidden state vectors of the two directions' LSTM units of $nca(e_1, e_2)$ are then concatenated as the syntactic path representation of (e_1, e_2):

$$\mathbf{R}^{gen}_{(e_1,e_2)} = [\overrightarrow{\mathbf{h}}^{gen}_{nca(e_1,e_2)}; \overleftarrow{\mathbf{h}}^{gen}_{nca(e_1,e_2)}]; \quad \mathbf{R}^{rel}_{(e_1,e_2)} = [\overrightarrow{\mathbf{h}}^{rel}_{nca(e_1,e_2)}; \overleftarrow{\mathbf{h}}^{rel}_{nca(e_1,e_2)}] \tag{3}$$

The global context representations and syntactic path representation are then composed through a non-linear layer, resulting in the representation used for final classification.

$$\mathbf{p} = \texttt{ReLU}(\mathbf{W}_{gc}\underbrace{[\mathbf{R}^{gc}_{e_1}; \mathbf{R}^{gc}_{e_2}]}_{\text{Global Context}} + \mathbf{W}_{sp}\underbrace{[\mathbf{R}^{gen}_{(e_1,e_2)}; \mathbf{R}^{rel}_{(e_1,e_2)}]}_{\text{Syntactic Path}} + \mathbf{b}) \tag{4}$$

$$p(c|\mathbf{p}) = \texttt{softmax}(\mathbf{g}_c^\top \mathbf{p} + \mathbf{q}_c) \tag{5}$$

Our model is trained by minimizing the cross-entropy loss: $\mathcal{L}(\theta) = -\sum_{i=0}^{N} \log p(c_i|\mathbf{p}_i)$, where N is number of training instances.

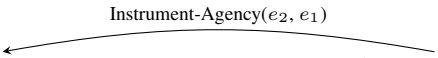
4.4 Post-Inference with Integer Linear Programming for SRL

SRL is a structure prediction problem and the predicted results should satisfy some structural constraints. For instance, some roles only appear once for a predicate in a sentence. Following Punyakanok et al. (2004) and Che et al. (2008), we apply ILP on the probability distributions at each token generated by our model to get the global optimization. We use the three constraints defined in Che et al. (2008):

- C1: Each word should be labeled with one and only one label (including *NULL*).
- C2: Roles with a small probability (smaller than 0.3) should never be labeled (except for *NULL*).
- C3: Some roles (except for *NULL*) usually appear once for a predicate in a sentence. Hence a non-duplicate-roles list is utilized for each language.

5 Multi-task Learning

The commonalities between SRL and RC inspire us to explore their potential mutual benefits. According to the *Shortest Path Hypothesis* (Bunescu and Mooney, 2005), if e_1 and e_2 are two entities mentioned in the same sentence such that they are observed to be in a certain relationship R, they often indicate two arguments of the same predicate or a sequence of predicates. To gain more insights, let's look at the following example in RC:



"The **author**$_{[e_1]}$ of a keygen uses a **disassembler**$_{[e_2]}$ to look at the raw assembly code."

Here, the "Instrument-Agency" relation provides significant evidences that **author** and **disassembler** are two arguments of a certain predicate, most likely with semantic roles A0 (agent) and A1 (patient). Furthermore, given the dependency parse tree, it's easy to find out that their associated predicate is "uses". Therefore, RC is expected to benefit both the identification and classification of semantic roles. Analogously, SRL results of a sentence also have positive impacts to the *identification* of semantic relations between e_1 and e_2, i.e. whether or not a relation exists between e_1 and e_2.

However, the roles defined in SRL can hardly contribute to the *classification* of much more fine-grained relation types in RC. For example, the roles A0, A1 can hardly help us to distinguish between the relation types like *Instrument-Agency*, *Product-Producer*, *Cause-Effect*, etc. Given this intuition, we will mainly focus on improving SRL with RC in this work.

Our proposed unified model allows knowledge transfer across SRL and RC in a natural way through parameter sharing. In this work, we consider two ways of knowledge transfer.

- Cascaded Learning (CAS). Models are trained in a cascaded manner. Specifically, a RC model is trained first, and then the parameters (e.g., word embeddings, network weights) are used to initialize the neural network for training SRL in the second stage.
- Multi-task Learning (MTL). Models are trained jointly in a stochastic manner:
  1. Select a task according to a certain probability distribution (explained below).
  2. Sample a batch of instances from the task, and feed-forward the neural network.
  3. Update the corresponding parameters by back-propagation w.r.t. the instances.
  4. Go to 1.

In multi-task learning, two important factors are taken into account. First, we typically expect the two tasks to converge at a similar rate (Caruana, 1997). We approximately achieve this by using a **weighted task sampling** strategy in step 1. More specifically, we observe that SRL converges about 4 times slower than RC by running them separately, hence we sample from SRL 4 times often than RC during training. Despite the lack of theoretical guarantee, we found it working well in practice. Second, the key for multi-task learning to work is **parameter sharing**. Given the unified architecture, we can share most of the network parameters for knowledge transfer. Note that different dependency parses might be used for SRL and RC in practice. In this work, we use the officially provided predicted parses from CoNLL-2009 shared task in SRL, but adopt Stanford parser (Manning et al., 2014) to obtain parses for sentences in RC. These kinds of parses are quite different in terms of both the head-finding rules and the dependency relations. Therefore, we set the parameters involving dependency path modeling as *task-specific*, i.e., $\mathbf{BiLSTM}_{gen}, \mathbf{BiLSTM}_{rel}$ and $\mathbf{W}_{sp}$ (Figure 2). The output weights ($\mathbf{g}$) are *task-specific* as standard of multi-task learning, in order to handle different set of relations to be classified in SRL and RC.

6 Experiment

In this section, we first describe data and our experimental settings, then the results and analysis.

6.1 Data and Settings

For SRL, we evaluate on the English dataset and other 4 languages (Chinese, Catalan, German and Spanish) in the CoNLL-2009 shared task. We use the official split for training, development and testing. In addition, a subset of the Brown corpus is used as the out-of-domain test set. We use the officially provided predicted POS tags, lemmas and dependency parses as our input. All predicates are given for each sentence during both training and testing. Besides, we neither predict nor use the sense for each predicate, and thus exclude the predicate senses in most of the evaluation. We follow Lei et al. (2015) and combine the predicate sense output of Anders et al. (2010) with our SRL output, to provide results directly comparable to previous published results.

We compare our model to several state-of-the-art systems, primarily including the best performing system in CoNLL-2009 shared task, the most recently proposed PathLSTM model of Roth and Lapata (2016), the neural network model of FitzGerald et al. (2015), and the low-rank tensor model of Lei et al. (2015). We also consider some variants of the above models that use reranking or model ensemble.

For RC, we use the relation classification dataset of the SemEval 2010 task 8. The dataset contains 10,717 annotated sentences, including 8,000 for training and 2,717 for testing. We conduct 5-fold cross-validation to determine the best training iterations, and use the official scoring script for evaluation.

Several competitive models are to be compared, including the top performed system in SemEval 2010 (Rink and Harabagiu, 2010), the Matrix-Vector Recursive Neural Network (MV-RNN) model of Socher et al. (2012), the CNN model of Zeng et al. (2014), the tensor-based model of Yu et al. (2014), the CNN model using ranking loss (dos Santos et al., 2015), and the dependency-based neural network models (Liu et al., 2015; Xu et al., 2015b).

Word embeddings are pretrained using *word2vec* on large-scale unlabeled data. For English, Catalan, German and Spanish, we use the latest Wikipedia data. For Chinese, we obtain the raw text from Xinhua news section (2000–2010) of the fifth edition of Chinese Gigaword (LDC2011T13). The LTP toolkit (Che et al., 2010) is applied to segment Chinese text into words.

We adopt *predicate-wise* training for SRL and *sentence-wise* training for RC, and use stochastic gradient descent for optimization. Initial learning rate is set to $\eta_0 = 0.1$ and updated as $\eta_t = \eta_0/(1+0.1t)$ on each epoch t. Our hyperparameters for the unified model are listed in Table 1. When training RC-only models, the LSTM input/hidden dimension is set to 200, and the dimension of hidden layer is 400.

Dimension of embeddings				Dimension of layers		
word	*POS*	*NE*	*WordNet*	*LSTM input*	*LSTM hidden*	*hidden*
200	25	25	25	100	100	200

Table 1: Hyperparameters settings.

6.2 SRL Results

Table 2 reports the SRL performance on the English dataset. Our supervised models (SUP) outperform the six top performing systems on both in-domain and out-of-domain datasets (the second block), and is comparable to two top systems that use reranking or model ensemble (the third block).

Effect of transfer learning By comparing the cascaded training system (CAS), the multi-task learning system (MTL) with SUP, we can find that the task of RC is significantly helpful for improving SRL models. In particular, MTL consistently works better than CAS. Our best models (MTL) outperform all of the previous systems, and achieve new state-of-the-art SRL results.

Figure 3 shows the learning curves of SUP, CAS and MTL on development data. At early training iterations, CAS is very close to MTL, and improves faster than SUP, indicating that the RC parameters indeed serve as a good initialization for SRL. MTL gradually outperforms CAS as the training converges, which further verifies the advantage of joint training over cascaded training.

Effect of post-inference We further investigate the effect of post-inference with ILP. As shown in Table 3, ILP has a considerable impact on the final SRL performance consistently for all of our models.

Multilingual Results Table 4 shows the results of our SRL-only system (SUP) on other languages in the CoNLL-2009 shared task. Our model outperforms the best performing system on all the four languages we considered, with particularly large gains on Chinese (+6.3 absolute F1-score). Note that our model is also unified for each language, without language-specific tuning of features or hyperparameters.

6.3 RC Results

The only difference of our RC model from the SRL model is at the input layer, where we use two additional features: NE and WordNet. Table 5 shows the RC results on the SemEval 2010 task 8. Our model achieves an F1-score of 83.9%, which is comparable to the top performing systems in previous work. dos Santos et al. (2015) obtain an F1-score of 84.1% by using ranking loss, with special treatment

Model	Excluding predicate senses			Including predicate senses	
	WSJ-dev	WSJ-test	Brown-test	WSJ-test	Brown-test
SUP	82.32	84.06	72.12	87.67	76.56
CAS	83.33	84.73	73.00	88.14	77.15
MTL	**83.51***	**85.04***	**73.22***	**88.37***	**77.34***
CoNLL-2009 1st place	–	82.08	69.84	86.15	74.58
(Roth and Lapata, 2016)	–	–	–	86.7	75.3
(FitzGerald et al., 2015)	82.3	83.6	71.9	87.3	75.2
(Lei et al., 2015)	81.03	82.51	70.77	86.58	75.57
(Roth and Woodsend, 2014)	–	80.87	69.33	85.50	74.67
(Anders et al., 2010)	78.85	81.35	68.34	85.80	73.92
Model + Reranker/Ensemble	WSJ-dev	WSJ-test	Brown-test	WSJ-test	Brown-test
(Roth and Lapata, 2016)+R,E	–	–	–	87.9	76.5
(FitzGerald et al., 2015)+E	83.0	84.3	72.4	87.8	75.5
(Roth and Woodsend, 2014)+R	–	82.10	71.12	86.34	75.88
(Anders et al., 2010)+R	80.50	82.87	70.91	86.86	75.71

Table 2: SRL labeled F1-score of our model variants, with comparison to the state-of-the-art systems on the CoNLL-2009 shared task. Statistical significance (MTL *vs.* SUP) with $p < 0.01$ is marked with *.

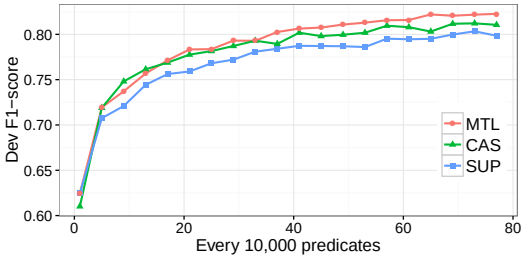


Figure 3: SRL F1-scores on the development data w.r.t. the number of predicates trained.

Model	WSJ-dev	WSJ-test
SUP	82.32	84.06
w/o ILP	81.87	83.53
CAS	83.33	84.73
w/o ILP	82.90	84.40
MTL	83.51	85.04
w/o ILP	83.15	84.75

Table 3: Effect of post-inference, evaluated excluding predicate senses.

to the artificial relation (*Other*). Such task-specific strategy can also be potentially used in our model for further improvements. As discussed in Section 5, to our intuition, knowledge contained in SRL is not supposed to benefit RC. To verify this, we further test on RC with cascaded learning and multi-task learning. We obtain a small degradation in RC performance in both cases (-0.9 for CAS and -0.7 for MTL). Nevertheless, we still expect improvements on joint learning of SRL and *relation extraction* (rather than *classification*), which we leave to future exploration.

7 Conclusion

In this paper, we propose a unified architecture for the task of SRL and RC. We effectively capture the global contextual representation and syntactic path representations using bidirectional LSTM-based recurrent neural networks. By evaluating on benchmark datasets for both SRL and RC, we show that our models outperform or get competitive results with the state-of-the-art systems. Furthermore, we take advantage of our unified model to transfer knowledge across the two tasks using multi-task learning with parameter sharing. Our models obtain new state-of-the-art results for SRL.

Acknowledgements

We are grateful to Tao Lei for providing the outputs of their systems. We thank the anonymous reviewers for their insightful comments and suggestions. This work was supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grant 61300113 and 61370164.

Language	Test set			
	Ours	(Lei et al., 2015)	CoNLL 1st	CoNLL 2nd
Chinese	**75.46**	69.16	68.52	68.71
Catalan	**79.24**	74.67	76.78	74.02
German	**77.41**	76.94	74.65	76.27
Spanish	**79.17**	75.58	77.33	74.01

Table 4: SRL labeled F1-score excluding predicate senses on Chinese, Catalan, German and Spanish. All results are evaluated excluding predicate senses.

Model	**Features**	**F1**
SVM (Rink and Harabagiu, 2010) (Best in SemEval 2010)	POS, prefixes, morphological, WordNet, Levin classes, PropBank, FrameNet, dependency parse, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2
MVRNN (Socher et al., 2012)	syntactic parse	79.1
MVRNN (Socher et al., 2012)	syntactic parse, POS, NER, WordNet	82.4
CNN (Zeng et al., 2014)	position, WordNet	82.7
FCM (Yu et al., 2014)	dependency path, NER	83.0
DepNN (Liu et al., 2015)	dependency parse, NER	83.6
CR-CNN (dos Santos et al., 2015)	position	**84.1**
depLCNN (Xu et al., 2015a)	WordNet, words around nominals	83.7
Ours	dependency path, POS, NER, WordNet	**83.9**
Model + Ensemble/Additional data		
ER-CNN+R-RNN (Vu et al., 2016)	position	84.9
depLCNN+NS (Xu et al., 2015a)	WordNet, words around nominals	85.6

Table 5: Comparison with previously published results for SemEval 2010 Task 8.

References

Björkelund Anders, Bohnet Bernd, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstrations*, pages 33–36, August.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853, December.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proc. of EMNLP-CoNLL*, pages 1455–1465, July.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proc. of EMNLP*, pages 724–731, October.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Wanxiang Che, Zhenghua Li, Yuxuan Hu, Yongqiang Li, Bing Qin, Ting Liu, and Sheng Li. 2008. A cascaded syntactic and semantic dependency parsing system. In *Proc. of CoNLL*, pages 238–242, August.

Wanxiang Che, Zhenghua Li, Yongqiang Li, Yuhang Guo, Bing Qin, and Ting Liu. 2009. Multilingual dependency-based syntactic and semantic parsing. In *Proc. of CoNLL 2009: Shared Task*, pages 49–54, June.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *Coling 2010: Demonstrations*, pages 13–16, August.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of the 25th ICML*, pages 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proc. of the 53rd ACL and the 7th IJCNLP (Volume 1: Long Papers)*, pages 1723–1732, July.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proc. of ACL-IJCNLP*, pages 626–634, July.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proc. of EMNLP*, pages 960–970, September.

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

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *Proc. of ACL*, pages 1045–1053, July.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics*, 39(4):949–998.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proc. of SemEval*, pages 33–38, July.

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

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *Proc. of NAACL*, pages 1528–1533, June.

Tao Lei, Yuan Zhang, Lluís Màrquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-order low-rank tensors for semantic role labeling. In *Proc. of NAACL*, pages 1150–1160, May–June.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for chinese pos tagging and dependency parsing. In *Proc. of EMNLP*, pages 1180–1191, July.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng WANG. 2015. A dependency-based neural network for relation classification. In *Proc. of ACL-IJCNLP*, pages 285–290, July.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Xavier Lluís, Xavier Carreras, and Lluís Màrquez. 2013. Joint arc-factored parsing of syntactic and semantic dependencies. *Transactions of the Association for Computational Linguistics*, 1:219–230.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proc. of 52nd ACL: System Demonstrations*, pages 55–60, June.

André F. T. Martins and Mariana S. C. Almeida. 2014. Priberam: A turbo semantic parser with second order features. In *Proc. of SemEval 2014*, pages 471–476, August.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning*, 60(1-3):11–39, September.

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proc. of COLING*, pages 1346–1352, Aug 23–Aug 27.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proc. of the 5th International Workshop on Semantic Evaluation*, pages 256–259, July.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. *arXiv preprint arXiv:1605.07515*.

Michael Roth and Kristian Woodsend. 2014. Composition of word representations improves semantic role labelling. In *Proc. of EMNLP*, pages 407–413, October.

Kashif Shah and Lucia Specia. 2016. Large-scale multitask learning for machine translation quality estimation. In *Proc. of NAACL*, pages 558–567, June.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proc. of EMNLP-CoNLL*, pages 1201–1211, July.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proc. of ACL*, pages 8–15, July.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191, June.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proc. of NAACL*, pages 534–539, June.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proc. of EMNLP*, pages 536–540, September.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proc. of EMNLP*, pages 1785–1794, September.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proc. of EMNLP*, pages 88–94, July.

Haitong Yang and Chengqing Zong. 2014. Multi-predicate semantic role labeling. In *Proc. of EMNLP*, pages 363–373, October.

Mo Yu, Matthew Gormley, and Mark Dredze. 2014. Factor-based compositional embedding models. In *NIPS Workshop on Learning Semantics*.

Mo Yu, Matthew R. Gormley, and Mark Dredze. 2015. Combining word embeddings and feature embeddings for fine-grained relation extraction. In *Proc. of NAACL*, pages 1374–1379, May–June.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proc. of COLING*, pages 2335–2344, August.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proc. of ACL-IJCNLP*, pages 1127–1137, July.

Facing the most difficult case of Semantic Role Labeling: A collaboration of word embeddings and co-training

Quynh Ngoc Thi Do[1], **Steven Bethard**[2], **Marie-Francine Moens**[1]
[1]Katholieke Universiteit Leuven, Belgium
[2]University of Arizona, United States
quynhngocthi.do@cs.kuleuven.be
bethard@email.arizona.edu
sien.moens@cs.kuleuven.be

Abstract

We present a successful collaboration of word embeddings and co-training to tackle in the most difficult test case of semantic role labeling: predicting out-of-domain and unseen semantic frames. Despite the fact that co-training is a successful traditional semi-supervised method, its application in SRL is very limited. In this work, co-training is used together with word embeddings to improve the performance of a system trained on CoNLL 2009 training dataset. We also introduce a semantic role labeling system with a simple learning architecture and effective inference that is easily adaptable to semi-supervised settings with new training data and/or new features. On the out-of-domain testing set of the standard benchmark CoNLL 2009 data our simple approach achieves high performance and improves state-of-the-art results.

1 Introduction

Semantic role labeling (SRL) is an essential natural language processing (NLP) task that identifies the relations between a predicate and its arguments in a given sentence. Intuitively, it aims at answering the questions of "Who did What to Whom, and How, When and Where?" in text. For example, the processing of the sentence "He bought tons of roses yesterday" should result in the identification of a "buying" event corresponding to the predicate "bought" with three arguments including "he" as the *Agent (A0)*, "tons of roses" as the *Thing being bought (A1)*, and "yesterday" as the *Time (AM-TMP)* arguments. Traditional SRL systems have concentrated on supervised learning from several manually-built semantic corpora, (e.g., FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2005)). One important limitation of supervised approaches is that they depend heavily on the accuracy, coverage and labeling scheme of the labeled corpus. When the training and the testing data are in different domains, the linguistic patterns and their distributions in the testing domain are different from the ones observed in the training data, resulting in a considerable performance drop. Developing more manually-built semantic corpora is expensive and requires huge human efforts. Thus, exploiting large unlabeled datasets by semi-supervised or unsupervised approaches is a promising solution.

Our contribution in this paper is two-fold: First, we introduce a SRL system with a simple learning architecture and effective inference that is easily adaptable to new training data or new features in semi-supervised settings. Second, we present a semi-supervised approach that is a combination of using word embeddings as extra features and using a variant of a co-training algorithm to create new training data facing the most difficult cases of SRL: improving the SRL system trained on a relatively large training dataset (CoNLL 2009) when working with the "out-of-domain" and especially "unseen" semantic frames. Although co-training is a successful traditional semi-supervised method, its application to SRL is very limited. To our knowledge, there has been no successful case of co-training applied to a large amount of training data in the literature.

In this work, we first enrich a traditional feature set of SRL by the distributional word representations induced from a large unlabeled corpus. Then, we divide the feature set into two different sets with one referring to the semantic or meaning information of the argument candidate and one referring to its

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1275–1284, Osaka, Japan, December 11-17 2016.

syntactic information based on the dependency structure. Two local classifiers are then trained, one on each of the two feature sets. Next, we label unannotated data from the same domain as the target data with these classifiers. Finally, a global classifier is trained with selected newly labeled instances and the joint feature set of the two local classifiers. Our experiments show that the combination of using distributional word representations and a co-training strategy effectively improves SRL in the challenging out-of-domain scenario. It outperforms using only word embeddings or co-training especially in unseen frames.

The rest of the paper is structured as follows. Section 2 discusses related works. We describe our SRL system and our methodology in Section 3 and Section 4 respectively. Our experiment is presented in Section 5 and Section 6 and finally we conclude in Section 7.

2 Related Work

In traditional supervised approaches, SRL is modeled as a pipeline of predicate identification, predicate disambiguation, argument identification, and argument classification steps. Hand-engineered linguistically-motivated feature templates represent the semantic structure employed to train classifiers for each step. It is common among the state-of-the-art systems to train a global reranker on top of the local classifiers to improve performance (Toutanova et al., 2005; Björkelund et al., 2010; Roth and Lapata, 2016). SRL models have also been trained using graphical models (Täckström et al., 2015) and neural networks (Collobert et al., 2011; FitzGerald et al., 2015). Some systems have applied a set of structural constraints to the argument classification sub-task, such as avoiding overlapping arguments and repeated core roles, and enforced these constraints with integer linear programming (ILP) (Punyakanok et al., 2008) or a dynamic program (Täckström et al., 2015).

Regarding leveraging unlabeled data, semi-supervised methods have been proposed to reduce human annotation efforts. He and Gildea (2006) investigate the possibility of a weakly supervised approach by using self-training and co-training for unseen frames of SRL. They separate the headword and path as the two views for co-training, but could not show a clear performance improvement. The sources of the problem appeared to be the big gap in performance between the headword and path feature sets and the complexity of the task. Some other works show slight improvements of using co-training for SRL when there is a limited number of labeled data (Lee et al., 2007; Samad Zadeh Kaljahi and Baba, 2011). Fürstenau and Lapata (2012) find novel instances for classifier training based on their similarity to manually labeled seed instances. This strategy is formalized via a graph alignment problem.

Recently, there has been interest in distributional word representations for natural language processing. Such representations are typically learned from a large corpus using neural networks (e.g., Weston et al. (2008)), probabilistic graphical models (e.g., Deschacht et al. (2012)) or term-cooccurrence statistics (e.g., Turney and Pantel (2010)) by capturing the contexts in which the words appear. Often words from the vocabulary or phrases are mapped to vectors of real numbers in a low dimensional continuous space resulting in so-called word embeddings. Deschacht et al. (2012) employ distributed representations for each argument candidate as extra features when training a supervised SRL. Roth and Woodsend (2014) propose to use the compositional representations such as interaction of predicate and argument, dependency path and the full argument span to improve a state-of-the-art SRL system.

3 A Semantic Role Labeling System for Semi-Supervised Approaches

In this section, we introduce a semantic role labeling system designed for semi-supervised settings. The system has a simple training strategy with local classifiers for different steps in SRL pipeline. Instead of training a global reranker on top of the local classifiers to improve performance as in other common pipeline-based state-of-the-art systems like (Toutanova et al., 2005; Björkelund et al., 2010; Roth and Lapata, 2016), we propose a novel joint inference technique that works across the argument identification and argument classification steps. That makes the SRL training simple (no reranker, only local classifiers) and therefore easily adaptable to new training examples or new features. We will show later in the experiment that the joint inference gives us comparable results to a reranker.

Following prior work (Toutanova et al., 2005; Björkelund et al., 2010; Roth and Lapata, 2016), our system consists of four modules: (1) A predicate identification (PI) module which detects whether a

word is a predicate. (2) A predicate disambiguation (PD) module which labels a predicate with a sense, where we train a local classifier for each predicate lemma. (3) An argument identification (AI) module that recognizes argument words, where given a predicate p, each word w_i in p's sentence is assigned a probability $P^{AI}(p, w_i)$ of being p's argument. (4) An argument classification (AC) module that assigns labels to argument words, where given a predicate p and a set of labels $\mathbf{L}$ – PropBank semantic role label set in this work, each word w_i is assigned probabilities $P^{AC}(p, w_i, L_j)$ to receive $L_j \in \mathbf{L}$ as semantic label.

We employ the features proposed by Björkelund et al. (2010) as the basic feature set. All of the local classifiers are trained using L2-regularized logistic regression. For multiclass problems, we use the one-vs-rest strategy.

At inference time, the local classifier predictions are merged using integer linear programming (ILP). In most of the prior work, ILP was only used for AC inference. However, this approach limits the interaction of AI and AC when making decisions. In another approach, Srikumar and Roth (2011) introduce a simple approach to joint inference over AI and AC allowing the two argument sub-tasks to support each other. Their local AC classifier has an empty label which indicates that the candidate is, in fact, not an argument. This forces AC module to learn also the argument identification and is in contrast with our approach in which the tasks of AI and AC classifiers are completely separated leading to a simpler AC learning. Their inference is formularized as an ILP problem that mazimizes the sum of local prediction scores over AI and AC. The authors then enforce consistency constraints between the identifier and the argument classifier predictions – the identifier should predict that a candidate is an argument if and only if the argument classifier does not predict the empty label. In this paper, we propose a novel ILP inference formulation in which the interaction of AI and AC is exploited and emphasized not only in the consistency constraint but also in the objective function.

Joint Argument Inference For each predicate, we perform joint inference over the AI and AC steps for all the words in the sentence. Given a predicate p and set of words in the sentence $w_1, w_2, ..., w_n$, each word is determined as either non-argument or as one of the semantic roles via an ILP formulation. Let $\mathbf{U}$ be the set of binary indicator variables corresponding to the decision whether w_i is a non-argument. Specifically, $u_i = 1$ if w_i is not an argument and $u_i = 0$ otherwise. Let $\mathbf{V}$ be the set of binary indicator variables corresponding to the decision whether w_i is a certain semantic role. Specifically, $v_{ij} = 1$ if w_i is assigned label L_j and $v_{ij} = 0$ otherwise.

A simple joint inference to maximize the sum of local prediction scores over AI and AC (see Srikumar and Roth (2011)) would be[1]:

$$\sum_{i=1}^{n} \left[(1 - P^{AI}(p, w_i)) * u_i + P^{AI}(p, w_i) * (1 - u_i) + \sum_{j=1}^{|\mathbf{L}|} \left(v_{ij} * P^{AC}(p, w_i, L_j) + (1 - v_{ij}) * (1 - P^{AC}(p, w_i, L_j)) \right) \right]$$

In this paper, to exploit more effectively the interaction between AI and AC, we propose to maximize the objective function:

$$\sum_{i=1}^{n} \left\{ (1 - P^{AI}(p, w_i)) * u_i * \lambda + P^{AI}(p, w_i) * \left[\sum_{j=1}^{|\mathbf{L}|} \left(v_{ij} * P^{AC}(p, w_i, L_j) + (1 - v_{ij}) * (1 - P^{AC}(p, w_i, L_j)) \right) \right] \right\}$$

Subject to:

$$\forall i : u_i + \sum_{j=1}^{|L|} v_{ij} = 1 \tag{1}$$

$$\forall j : if\ L_j\ is\ core\ role : \sum_{i=1}^{n} v_{ij} \leq 1 \tag{2}$$

[1]Note that we can not use exactly the same objective function as in Srikumar and Roth (2011) because our AC module does not produce the empty label.

λ is a parameter between 0 and 1 controlling the balance of recall and precision. If λ is small, the importance of predicting correct non-argument words is lowered, so more words are considered as argument candidates leading to the increase of recall and decrease of precision. In contrast, if λ is large, precision goes up and recall goes down.

Constraint 1 forces the system to assign either only one semantic role or a non-argument label to each word. Meanwhile, constraint 2 restricts the core roles (A0, A1, A2, A3, A4, A5, AA) to appear no more than once.

In the experiment, we will compare the performance of our proposed inference to the approaches using a reranker as in Björkelund et al. (2010) and the above simple joint inference of AI and AC.

4 Semi-supervised approach

4.1 Problem

In SRL, classifiers need linguistic clues to make a correct prediction. Two different types of clues are frequently distinguished: (1) Semantic or meaning clues. For example, when classifying arguments for the predicate "sleep", if the classifier sees the word "bed", assigning the role "AM-LOC" to the word seems reasonable. (2) Syntactic or word order clues. In the above example, if the classifier sees a candidate that is a child of the predicate in the dependency tree and has "SUBJECT" as deprel or "NNP" as part of speech (POS) tag, it is likely to be the role "A0". These clues are often helpful, but they often are not specific enough to predict the exact semantic role. For example, given the sentences "we cut the cake on Monday" and "we cut the cake on the table", the words "on" in both sentences have the same POS path to the predicate "cut", but they are two different roles ("AM-TMP" and "AM-LOC" respectively). Traditional approaches for SRL encode linguistic clues as indicator features such as the observed POS tag of a word and the syntactic path to its head. However, their occurrence is often sparse in the training data and in combination with being ambigous signals for semantic roles they do not generalize well across domains. When the target data is in a different domain, its vocabulary differs from the training data, and the classifier may fail to recognize semantic or meaning clues. If the predicate is observed in the training data, then the syntactic patterns may still be useful. However, in the worst case, if the predicate is unseen, both of the clues become weak leading to the most difficult case for SRL.

4.2 Method Description

Motivated by the successes of distributional word representations in capturing word similarity and of co-training in boosting the performance of classification by using two different views which naturally fit the two types of clues discussed above, we propose a combination of these approaches to tackle the problem. Our method, shown in Algorithm 1 and described in detail in Section 4.4, leverages unlabeled data to improve the performance of SRL. We first extend the feature set with distributional representations induced from a large unlabeled corpus. The two modules PI and PD are trained and used to label the predicate and predicate senses in the unlabeled texts. We then divide the feature sets of AI and AC into two intuitive sets, one for the semantic information of the argument candidate, and one for the syntactic information. A co-training strategy is applied twice, once to AI and once to AC. In each process, local classifiers are trained on the labeled data using the two divided feature sets. Then, we label a set of unlabeled data from the target domain using these classifiers. At the final step, selected newly labeled instances are used to train a global classifier with the joint feature set of the two local classifiers.

4.3 Word Embeddings

Instead of using the compositional representations proposed in (Roth and Woodsend, 2014), we follow a simple and natural approach to employ word embeddings: For each *word feature* (e.g., argument word, predicate word, right child word), we map its value to a distributed representation and concatenate the new representation to the original feature vector. For each *word set feature* (e.g., child word set), we sum up the distributional representations of the set elements, and concatenate the obtained vector to the original feature vector. Under this approach, word embeddings can be easily applied to any step of SRL which contains word or word set features. We will show later in the experiments that this simple approach

Table 1: The feature division for Co-training. -,N,V indicates that the feature is not used, used for nominal frames, and used for verbal frames respectively. Note that we omit distributional representation features and feature bigrams.

	Identification		Classification			Identification		Classification	
	Sem	Syn	Sem	Syn		Sem	Syn	Sem	Syn
DeprelPath	-	V,N	-	V	Deprel	V	-	V	-
POSPath	-	V,N	-	V	RightSiblingWord	-	V	-	-
Word	V,N	-	V,N	-	PredParentWord	-	V	-	V
PredParentPOS	-	V	-	V	Position	V,N	-	V,N	-
PredLemmaSense	V	V	V,N	V,N	PredLemma	N	N	V,N	V,N
ChildWordSet	N	-	-	-	RightChildPOS	N	-	V,N	-
PredPOS	N	N	V	V	RightChildWord	N	-	V,N	-
ChildDeprelSet	-	-	V	-	POS	-	-	V	-
LeftSiblingPOS	-	-		V,N	LeftChildPOS	-	-	V	-
PredWord	-	-	V,N	V,N	LeftSiblingWord	-	-		N
LeftChildWord	-	-	N	-	ChildPOSSet	-	-	N	-

gives us comparable results to other state-of-the-art systems. Word embeddings also help to improve the semantic or meaning clues, and can therefore boost the performance of the co-training strategy.

4.4 Co-training

The *co-training* semi-supervised learning paradigm was first proposed by Blum and Mitchell (1998), aiming at exploiting unlabeled data to improve performance given limited training data. The classical algorithm applies when the data can be represented by two or more separate, but redundant "views" such as two disjoint feature subsets. For example, web pages can be described by either the text on the web page or the text from hyperlinks pointing to the web page. The two classifiers trained on two "views" of the data can help each other, by adding one's most confident examples into the other one's training set.

In this work, we apply a variant of co-training to the two argument steps. We propose to divide the SRL feature sets into two views based on the dependency tree. The Sem_View, which emphasizes the semantic or meaning clues (and is related to the headword view of He and Gildea (2006)), consists of all features based on the argument candidate itself and all the words that are lower than the argument candidate on the dependency tree (e.g., left child word, right child word, children word set). The Syn_View, which emphasizes syntactic clues (and is related to the path view of He and Gildea (2006)), consists of all features based on the path from the argument to the predicate and all the words that are equal or higher than the argument candidate on the dependency tree (e.g., left sibling word, right sibling word, parent word). The features referring to the predicate itself are included in both views. More details of the feature division can be found in Table 1.

The details of the co-training method are shown in Algorithm 1. First of all, we extend our feature sets with word embeddings as proposed in Section 4.3. The two predicate modules PI and PD are trained on the original training data, then used to label the unannotated data. After that, we start the co-training process. For each of AI and AC[2], we loop for a number of iterations: the two views are trained on the labeled data using their corresponding feature sets with logistic regression. Both of the two views are used to label the unannotated data. We then select informative examples from the unlabeled dataset to be used as extra training data. Our selection strategy is to select the examples that cause a disagreement between the two views: one view is confident that the example belongs to class c (labeling score is higher than a certain threshold) while the other view is uncertain about this (labeling score is between a lower and upper threshold). This strategy differs from a classical co-training set up where we select the examples where the two local classifiers confidently agree (Blum and Mitchell, 1998). This is motivated by the

[2]Note that the co-training is applied to AI and AC separately. That means the two views of AI interact with each other, and the two views of AC interact with each other, but views from AI do not interact with views from AC.

Algorithm 1: The SRL Co-training Algorithm.

Input : A large collection of labeled sentences $\mathbf{L}$ and one of unlabeled sentences $\mathbf{U}$
Output : A full SRL

Enrich feature sets with word embeddings.
Train PI and PD on $\mathbf{L}$.
Label $\mathbf{U}$ by PI and PD.
Divide the feature sets of AI and AC into semantic/meaning clues (Sem_View) and syntactic/word-order clues (Syn_View).
for *each of the two argument sub-tasks (AI or AC)* **do**
    $\mathbf{V} = \mathbf{U}$
    $n = 0$
    while $\mathbf{V} \neq \emptyset \wedge n < |\mathbf{L}|$ **do**
        Build local classifier A on $\mathbf{L}$ using Sem_View features
        Build local classifier B on $\mathbf{L}$ using Syn_View features
        $n = |\mathbf{L}|$
        for *x in* $\mathbf{V}$ **do**
            Use A and B to label x.
            if *A or B is confident (labeling score* $> t_1$*) that* x *belongs to class* c *and the other is uncertain (*$t_3 <$ *labeling score* $< t_2$*)* **then**
                Add (x, c) to $\mathbf{L}$
            end
            Remove x from $\mathbf{V}$
            if $|\mathbf{L}| \geq n + k$ **then**
                Break the for loop
            end
        end
    end
    Build the global classifier for AI (or AC) on $\mathbf{L}$ using the joint feature set.
end

fact that the training data in SRL is sufficiently large to build a good system, so we only select the most informative new instances: the ones that make a disagreement between the two views. That means we are helping the classifiers to overcome the cases when one of the two clue types is weak which is common in out-of-domain prediction. Intuitively, when one clue type is strong and the other is not good, the strong one can help the other.

To ensure the balance between different classes, for AI, we added the same number of positive and negative examples to the labeled data. For main roles (e.g., A0, A1) in AC, we skip adding an instance to class c if the ratio of the number of instances in c to the rest is increased more than a certain number of times (2.0 in our experiments) since main roles are already the majority in the original training data. The maximum number of instances that could be added to the training data in each iteration is k[3]. The iterations will finish when there is no instance satisfying our selection criteria or there is no unlabeled data left. At the end of the process, a global classifier for each of the argument sub-task modules (AI or AC) is trained on the final labeled training set using the joint feature set of the two views.

Although co-training has reported success in many real-world applications (Blum and Mitchell, 1998; Kiritchenko and Matwin, 2001; Mihalcea, 2004; Javed et al., 2005), its application in SRL is still very limited. It is common in NLP that the two assumptions of co-training do not strictly hold. He and Gildea (2006) discovered that their two co-training views, headword and path, were not balanced. The headword view was more sensitive to new data, and led to a significant drop in precision. The path view was a more accurate and stable indicator for semantic role labeling. By adding distributional word representations to

[3] k is fixed to 50% of the size of the original training data in our experiment.

Table 2: Information about datasets.

	Section	Corpus	Type
Train	02-21	TreeBank	financial news
Dev	24	TreeBank	financial news
Ood	ck01-03	Brown	fiction
UB	ck04-29,cl,cm,cp	Brown	fiction
UW	01	TreeBank	financial news

the feature set, we expect to improve the performance of the headword view when faced with new data, and thereby produce a successful co-training strategy. Also, since it is difficult to make a good prediction with just one of the two different clue types, we build a global classifier for each of AI and AC with a joint feature set at the end of the process.

5 Experiments

We evaluate the performance of our method when training on a large training set and testing on an out-of-domain test set using a collection of unlabeled data. We expect that from the unlabeled data, which is mostly in the same domain as the target data, we can create a number of new training instances that improve system performance on the target domain.

5.1 Settings

We use the same training, development and out-of-domain test set as provided in the CoNLL 2009 shared task (Hajič et al., 2009). We also collect two sets of unlabeled data which are mostly texts in the same domain as the CoNLL 2009 out-of-domain test set. Table 2 shows some information about our datasets. We call the training set Train, development set Dev, out-of-domain test set Ood, unlabeled data from Brown corpus UB and unlabeled data from TreeBank UW. The training data has 39,279 financial sentences with 12,036 predicate words. In the out-of-domain test set, there are 788 different predicate words and 114 of them have never been observed in the training data. The collection of unlabeled data (UB and UW) contains 12,462 fictional and 1,828 financial news sentences.

The prepreprocessing modules including POS tagger, lemmatizer and dependency parser of (Björkelund et al., 2010) are used in our experiments. All the modules are retrained on the CoNLL 2009 training set. We use Word2Vec[4] (**?**) to learn 300-dimensional word representations from unlabeled corpora including Wikipedia[5], Reuters[6], TreeBank[7].

Following the standard setting of the CoNLL 2009 shared task, we skip the predicate identification step when evaluating our models. It is only used to annotate unlabeled data in the co-training experiments. All the methods are evaluated using the official CoNNL 2009 evaluation software.

We set λ, the parameter controlling the balance between precision and recall, to 0.3 in all of our experiments based on tuning that parameter on the Dev set.

We use as a baseline our system trained on the feature sets proposed by Björkelund et al. (2010) (Baseline).

To evaluate the effectiveness of our proposed inference, we replace the inference of the above Baseline model by a simple joint inference maximizing the local prediction scores over AI and AC (see Section 3). We call this model Simple. Furthermore, we also implement NoJoint model, which has the same settings as Baseline, but the joint inference is replaced by the classical predictions of the local modules.

Our three different proposed approaches are evaluated: using only word embeddings as in Section 4.3 (WE), using co-training strategy as in Section 4.4 (CO) but without word embeddings, and using the combination of word embeddings and co-training as in Section 4.4 (WECO). We also compare our

[4]https://code.google.com/archive/p/word2vec/
[5]http://corpus.byu.edu/wiki/
[6]http://about.reuters.com/researchandstandards/corpus/
[7]https://catalog.ldc.upenn.edu/ldc99t42

Table 3: Detailed CoNLL 2009 results on seen semantic frames (seen predicates) and unseen semantic frames (unseen predicates) on the out-of-domain test set. Significant differences (computed using a randomization test; cf. Yeh (2000)) from Baseline in terms of F1-score are marked by asterisks (* $p < 0.05$)

	Seen frames			Unseen frames		
Method	P	R	F1	P	R	F1
Baseline	79.7	72.7	76.1	77.0	67.1	71.7
WE	80.2	73.6	76.7*	78.8	67.6	72.8*
CO	80.3	72.3	76.1	80.8	66.8	73.1*
WECO	80.4	73.7	**76.9***	81.6	67.9	**74.2***

Table 4: CoNLL 2009 results on the out-of-domain test set

	P	R	F1
CoNLL-2009 1st place	-	-	74.6
Björkelund et al. (2010)	77.9	73.6	75.7
Roth and Woodsend (2014)	-	-	75.9
Lei et al. (2015)	-	-	75.6
Täckström et al. (2015)	-	-	75.5
FitzGerald et al. (2015)	-	-	75.9
Roth and Lapata (2016)	79.7	73.6	76.5
NoJoint	75.9	72.8	74.3
Simple	73.3	75.3	74.3
Baseline	79.5	72.3	75.7
WE	80.1	73.1	76.4
CO	80.3	71.9	75.9
WECO	80.5	73.2	**76.7**

results with the current state-of-the-art systems. We tune the thresholds t_1, t_2, and t_3 in Algorithm 1 by measuring performance on Dev set resulting in $(t_1, t_2, t_3) = (0.8, 0.5, 0.3)$.

5.2 Results

From Table 3, we can see that including word embeddings with the capacity of better capturing word similarity already improves the out-of-domain prediction. Using co-training seems to be more useful when working with unseen frames. It can be seen that CO gives high precision which has possibly benefited from the enforced agreement between the two views. Meanwhile, WE appears to be better in improving recall because of the good capacity to capture word similarity. Interestingly, the combination of these two approaches WECO brings us high scores in both precision and recall. It obtains the best results especially for unseen semantic frames which are the most difficult cases for SRL. WECO improves the prediction for unseen frames over the baseline, WE and CO 2.5, 1.4 and 1.1 F1 points respectively.

As can be seen from Table 4, with a simple learning architecture (local learning, no reranker) and an effective inference, our Baseline model already obtains results comparable to those of state-of-the-art systems. It reaches the F1 score of 75.7% which is the same as Björkelund et al. (2010) using a reranker on top of local classifiers with the same feature set as our baseline model. Our Baseline model surpasses both Simple and NoILP models by 1.4 F1 points proving the effectiveness of our ILP objective function. With our best model, WECO, the official CoNLL 2009 F1 score obtained on the out-of-domain test set outperforms a much more complicated system using reranker and neural network proposed in (Roth and Lapata, 2016) by 0.2 F1 points.

Selection strategy To evaluate our selection strategy in the co-training, we also perform an experiment with a classical selection strategy in which we select examples on which the two views agree with high confidence (i.e. larger than t_1 in both views), resulting in a reduction of 0.2 F1 points on the out-of-domain

test set.

6 Discussion

Semi-supervised setting The experiments show the promising application of semi-supervised methods in out-of-domain scenarios. We present the first successful case of using co-training for SRL when using all available training data (and not an artificially limited small subset). These successes encourage us to develop more advanced techniques digging into the collaboration of the two views in SRL. Semi-supervised techniques have the capacity to exploit a huge amount of unlabeled data, which is cheaper than manually-built annotated data. However, the success of these techniques depends on the quality of unlabeled data. If the unlabeled data itself does not contain informative knowledge, then semi-supervised methods will not help in improving the performance.

System architecture As shown before, even with a simple learning technique we still can obtain very strong results with an effective inference. This architecture is beneficial in semi-supervised settings which often require dealing with new training examples and new features. The inference might increase the computational complexity of the prediction, but given the current progress in efficient software and hardware solutions this will not pose serious problems.

7 Conclusion

We have presented a semi-supervised SRL system facing the most difficult case of SRL: predicting out-of-domain and unseen semantic frames. Our system, with a simple learning architecture and effective joint inference, obtains very strong results on the standard benchmark SRL data from the CoNLL 2009 shared task. We propose using a collaboration of word embeddings and a variant of co-training to exploit unlabeled data. Our method leverages the collaborative relationship of the two signal types in SRL (semantic and syntactic clues) to create informative new training examples that help out-of-domain prediction. An experiment exploiting unlabeled data which includes mostly fictional texts from the Brown corpus achieves better performance than state-of-the-art models on the out-of-domain test set of the CoNLL 2009 shared task.

Last but not least, our SRL system is made publicly available at http://liir.cs.kuleuven.be/software.php.

Acknowledgment

This work is funded by the EU ICT FP7 FET project "Machine Understanding for interactive StorytElling" (MUSE) `http://www.muse-project.eu/`.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of ACL 1998*, ACL '98, pages 86–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *COLING 2010: Demonstrations*, pages 33–36, Beijing, China.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT 1998*, pages 92–100, New York, NY, USA. ACM.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Koen Deschacht, Jan De Belder, and Marie-Francine Moens. 2012. The latent words language model. *Computer Speech and Language*, 26(5):384–409, October.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of EMNLP 2015*.

Hagen Fürstenau and Mirella Lapata. 2012. Semi-supervised semantic role labeling via structural alignment. *Computational Linguistics*, 38(1):135–171.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL 2009: Shared Task*, pages 1–18, Stroudsburg, PA, USA. ACL.

Shan He and Daniel Gildea. 2006. Self-training and co-training for semantic role labeling: Primary report. Technical report, Technical Report 891, University of Rochester.

Omar Javed, Saad Ali, and Mubarak Shah. 2005. Online detection and classification of moving objects using progressively improving detectors. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 696–701.

Svetlana Kiritchenko and Stan Matwin. 2001. Email classification with co-training. In *Proceedings of CASCON 2001*, pages 8–. IBM Press.

Joo-Young Lee, Young-In Song, and Hae-Chang Rim. 2007. Investigation of weakly supervised learning for semantic role labeling. In *Proceedings of ALPIT 2007*, pages 165–170, Washington, DC, USA. IEEE Computer Society.

Tao Lei, Yuan Zhang, Lluís Màrquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-order low-rank tensors for semantic role labeling. In *Proceedings of NAACL 2015*, pages 1150–1160, Denver, Colorado, May–June. ACL.

Rada Mihalcea. 2004. Co-training and self-training for word sense disambiguation. In *In Proceedings of CoNLL 2004*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, March.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computation Linguistics*, 34(2):257–287, June.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of ACL 2016*, Berlin, Germany.

Michael Roth and Kristian Woodsend. 2014. Composition of word representations improves semantic role labelling. In *EMNLP 2014*, pages 407–413.

Rasoul Samad Zadeh Kaljahi and Mohd Sapiyan Baba. 2011. Investigation of co-training views and variations for semantic role labeling. In *Proceedings of Workshop on Robust Unsupervised and Semisupervised Methods in Natural Language Processing*, pages 41–49, Hissar, Bulgaria, September.

Vivek Srikumar and Dan Roth. 2011. A joint model for extended semantic role labeling. In *Proceedings of EMNLP 2011*, pages 129–139, Stroudsburg, PA, USA. ACL.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *TACL*, 3:29–41.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL 2005*, pages 589–596, Stroudsburg, PA, USA. ACL.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188, January.

Jason Weston, Frédéric Ratle, and Ronan Collobert. 2008. Deep learning via semi-supervised embedding. In *Proceedings of ICML 2008*, pages 1168–1175, New York, NY, USA. ACM.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of COLING 2000*, COLING '00, pages 947–953, Stroudsburg, PA, USA. ACL.

Predictability of Distributional Semantics in Derivational Word Formation

Sebastian Padó* **Aurélie Herbelot+** **Max Kisselew*** **Jan Šnajder†**
* Institute for Natural Language Processing, University of Stuttgart
{sebastian.pado,max.kisselew}@ims.uni-stuttgart.de
+ Center for Mind/Brain Sciences, University of Trento
{aurelie.herbelot}@unitn.it
† Faculty of Electrical Engineering and Computing, University of Zagreb
jan.snajder@fer.hr

Abstract

Compositional distributional semantic models (CDSMs) have successfully been applied to the task of predicting the meaning of a range of linguistic constructions. Their performance on semi-compositional word formation process of (morphological) derivation, however, has been extremely variable, with no large-scale empirical investigation to date. This paper fills that gap, performing an analysis of CDSM predictions on a large dataset (over 30,000 German derivationally related word pairs). We use linear regression models to analyze CDSM performance and obtain insights into the linguistic factors that influence how predictable the distributional context of a derived word is going to be. We identify various such factors, notably part of speech, argument structure, and semantic regularity.

1 Introduction

Compositional models of distributional semantics, or CDSMs (Mitchell and Lapata, 2010; Erk and Padó, 2008; Baroni et al., 2014; Coecke et al., 2010), have established themselves as a standard tool in computational semantics. Building on traditional distributional semantic models for individual words (Turney and Pantel, 2010), they are generally applied to *compositionally compute phrase meaning* by defining combination operations on the meaning of the phrase's constituents. CDSMs have also been co-opted by the deep learning community for tasks including sentiment analysis (Socher et al., 2013) and machine translation (Hermann and Blunsom, 2014). A more recent development is the use of CDSMs to model meaning-related phenomena above and below syntactic structure; here, the term "composition" is used more generally to apply to processes of meaning combination from multiple linguistic units, e.g., above and below syntactic structure. Above the sentence level, such models attempt to predict the unfolding of discourse (Kiros et al., 2015). Below the word level, CDSMs have been applied to model word formation processes like compounding (*church* + *tower* → *church tower*) and (morphological) derivation (Lazaridou et al., 2013) (*favor* + *able* → *favorable*). More concretely, given a distributional representation of a basis and a derivation *pattern* (typically an affix), the task of the CDSM is to predict a distributional representation of the derived word, without being provided with any additional information. Interest in the use of CDSMs in this context comes from the observation that derived words are often less frequent than their bases (Hay, 2003), and in the extreme case even completely novel; consequently, distributional evidence is often unreliable and sometimes unavailable. This is confirmed by Luong et al. (2013) who compare the performance of different types of word embeddings on a word similarity task and achieve poorer performance on data sets containing rarer and more complex words. Due to the Zipfian distribution there are many more rare than frequent word types in a corpus, which increases the need for methods being able to model derived words.

In this paper, we ask to what extent the application of CDSMs to model derivation is a success story. The record is unclear on this point: Lazaridou et al. (2013), after applying a range of CDSMs to an English derivation dataset, report success, while Kisselew et al. (2015) found very mixed results on German derivation and generally high variance across words and derivation patterns. The analyses in both studies

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1285–1296, Osaka, Japan, December 11-17 2016.

POS + ID	Pattern	Sample word pair	English translation
A → N 16	*+ität*	produktiv → Produktivität	productive → productivity
A → V 04	*(umlaut)*	kurz → kürzen	short → to shorten
N → A 26	*-ung +end*	Einigung → einigend	agreement → agreeing
N → V 07	*be+ +ig*	Ende → beendigen	end → to end
V → N 09	*(null)*	aufatmen → Aufatmen	to breathe → sigh of relief
V → V 14	*auf+*	holen → aufholen	to fetch → to catch up

Table 1: Examples of derivation patterns from DErivBase

were also limited in scope (cf. Section 2.2). Furthermore, from a linguistic point of view, it is not at all obvious that it is reasonable to model derivation as a fully compositional process, as CDSMs do. Indeed, the classic linguistic definition of derivation distinguishes it from inflection by appealing to its *semantic irregularity*: the meaning changes that accompany derivation are not supposed to be completely predictable (Plank, 1981; Laca, 2001; Plag, 2003; Dressler, 2005).

More specifically, our goal is to gain a more precise understanding of the linguistic factors that govern the success or failure of CDSMs to predict distributional vectors for derived words. To this end, we conduct a broad-coverage analysis of the performance of CDSMs on more than 30,000 German derivationally related word pairs instantiating 74 derivation patterns. As a first step, we build *CDSM prediction models* for each of these patterns. The second step is a linear regression analysis with linguistic properties of patterns and word pairs as predictors and the models' performances as dependent variables. We formulate and test a number of hypotheses about the linguistic properties and establish that, notably, derivations that create new argument structure are generally hard to predict – although the difficulty is mediated by the regularity of the semantic shift involved. Subsequently, we exploit the regression results to combine several state-of-the-art CDSMs into an ensemble. Unfortunately, we do not see improvements over the individual models, which we trace back to a lack of complementarity among the CDSMs.

2 Background: Modeling Morphological Derivation

2.1 Derivational Lexicons

Morphological derivation is a word formation process that produces new words and which, at the word surface-level, can be described by means of an orthographic *pattern* applied to basis words. Table 1 shows that in the simplest case (row 1) this means attaching an affix (*+ität*). The other rows show that the pattern can be more complex, involving stem alternation (row 2; note that the infinitive suffix *+en* is inflectional), deletion of previous affixes (row 3), circumfixation (row 4), or no overt changes, i.e., conversion (row 5).[1] Derivation can take place both within parts of speech (row 6) and across parts of speech.

Derivation is a very productive process in many languages, notably Slavic languages. Thus, natural language processing (NLP) for these languages can profit from knowledge about derivational relationships (Green et al., 2004; Szpektor and Dagan, 2008; Padó et al., 2013). Nevertheless, derivation is a relatively understudied phenomenon in NLP, and few lexicons contain derivational information. For English, there are two main resources. CatVar (Habash and Dorr, 2003) is a database that groups 100K words of all parts of speech into 60K *derivational families*, i.e., derivationally related sets of words. The other is CELEX (Baayen et al., 1996), a multi-level lexical database for English, German, and Dutch, which covers about 50K English words and contains derivational information in its morphological annotation. For German, DErivBase (Zeller et al., 2013) is a resource focused on derivation that groups 280K lemmas into 17K derivational families. As opposed to CatVar and CELEX, it also provides explicit information about the applicable derivation pattern at the level of word pairs. The examples in Table 1 are from DErivBase.

[1] We write patterns as sequences of orthographic operations, using '+' for addition and '-' for deletion, and place the operator before or after the affix to distinguish prefixation and suffixation.

2.2 Modeling the Semantics of Derivation with CDSMs

Lazaridou et al. (2013) were the first to predict distributional vectors for derived words using CDSMs and experimented with a range of established CDSMs. In their paper, all models are supervised, i.e., some word pairs for each pattern are used as training instances, and others serve for evaluation. Also, all models assume that the base word (input) $\mathbf{b}$ and derived word (output) $\mathbf{d}$ are represented as vectors in some underlying distributional space. The *simple additive model* predicts the derived word from the base word as $\mathbf{d} = \mathbf{b} + \mathbf{p}$ where $\mathbf{p}$ is a vector representing the semantic shift accompanying the derivation pattern. The *simple multiplicative model*, $\mathbf{d} = \mathbf{b} \odot \mathbf{p}$ is very similar, but uses component-wise multiplication ($\odot$) instead of addition to combine the base and pattern vectors (Mitchell and Lapata, 2010). The third model, the *weighted additive model*, enables a simple reweighting of the contributions of basis and pattern ($\mathbf{d} = \alpha\mathbf{b} + \beta\mathbf{p}$). Finally, the *lexical function model* (Baroni and Zamparelli, 2010) represents the pattern as a matrix P that is multiplied with the basis vector: $\mathbf{d} = \mathrm{P}\mathbf{b}$, essentially modelling derivation as linear mapping. This model is considerably more powerful than the others, however its number of parameters is quadratic in the number of dimensions of the underlying space, whereas the additive and multiplicative models only use a linear number of parameters.

For their empirical evaluation, Lazaridou et al. (2013) considered a dataset of 18 English patterns defined as simple affixes – 4 within-POS (such as *un-*) and 14 across-POS (such as *+ment*) – and found that the lexical function model is among the top performers, followed by the weighted additive and multiplicative models, all substantially better than baseline. From our perspective, their evaluation has a number of limitations, though: they only included "high-quality" vectors (using human judgments of nearest neighbors to determine quality), thereby focusing on a relatively well-behaved subset of the vocabulary and potentially missing out on highly polysemous words. Furthermore, they evaluated mainly by computing mean cosine similarities between predicted and corpus-observed ("gold") vectors for the derived words – this is not highly informative, as the closeness of the prediction to the actual vector is also dependent on the density of the target's neighborhood.[2] A follow-up study on German (Kisselew et al., 2015) attempted to address these limitations by including all word pairs without prefiltering, and introducing a new evaluation metric that measured how often the predicted vector $\mathbf{d}$ was among the five nearest neighbors of the corpus-observed ("gold") vector $\mathbf{d}$. Kisselew et al.'s evaluation obtained fairly different results: the lexical function model performed worse than the simple additive model, and both CDSMs often had problems outperforming the baseline. This study had its own limitations, though, since it considered only 6 derivation patterns, all within-POS. Thus, it remains unclear to what extent the differences between the two studies are due to (a) the language difference, (b) prefiltering word pairs, or (c) the choice of derivation patterns under consideration.

3 Analyzing Models of Morphological Derivation

Given these conflicting results, we propose to empirically investigate the factors that influence how well CDSMs can predict the semantics of derived words. Note that when we talk about 'predictability' of a derived form, we refer to the ability to model an otherwise already established term, for which a distributional analysis can be performed. That is, we investigate to which extent a one-off compositional procedure can capture the meaning of a word, as in a situation where a speaker encounters an existing term for the first time. Further, we assume that the individual items in the observed data will naturally have different frequencies (from rare to highly frequent) and that this will affect both the learning process and the certainty we can have about the meaning of a test vector. We believe this is a realistic setup in terms of modelling first encounters with established derivations, and we therefore make no attempt to control for the frequency of individual word vectors, either at training or test time.

3.1 Overall Workflow

We follow a two-step workflow depicted on the left-hand side of Figure 1. The workflow involves *prediction models* (cf. Section 3.2), i.e., CDSMs that predict a vector for the derived word given the vector for the base word, as well as *analysis models* (cf. Section 3.3), i.e., linear regression models that predict

[2] They also performed a manual judgment study, but only as an additional experiment on "low-quality" word pairs.

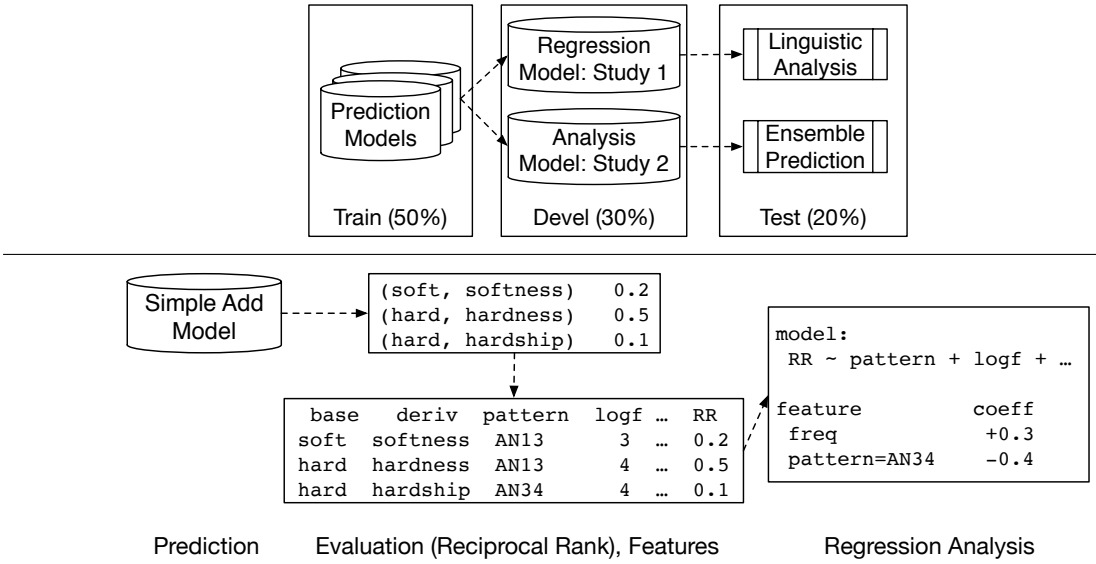


Figure 1: Top: Overall workflow. Below: Toy example

the performance of the CDSMs based on a rich set of linguistic predictors. We build two analysis models, one for linguistic analysis (Experiment 1, Section 4) and one for NLP (Experiment 2, Section 5).

The workflow uses a large set of derivationally related word pairs split into training, development, and test sets (50%, 30%, and 20%, respectively). The splits are stratified by derivation pattern, i.e., each pattern occurs in approximately these ratios in each split. This is a reasonable strategy, assuming that our set of patterns is fairly complete (Zeller et al., 2013) and we can disregard the problem of unseen patterns.

The three sets play different roles. The training set is used to train prediction models. The development set is then used to measure the performance of prediction models on unseen data. These performance numbers are those that the regression model is then trained to predict. Finally, the analysis model itself is evaluated on the test set, that is, on another previously unseen dataset. In this manner, we guarantee that all results we obtain are measured on unseen data and generalize well to novel instances.

We note that the task of the prediction models (constructing the vector for the derived word) incorporates our general assumption that we do not have any information about the derived word. While this is not a reasonable assumption from an NLP point of view, where we would know at least the frequency of the derived word, and may also have a (typically less reliable) distributional vector for it, this "no-knowledge" setup represents, in our opinion, the cleanest setup for an analysis of linguistic factors.

3.2 Prediction (CDSM) Models

Derivationally Related Word Pairs. We draw our derivationally related word pairs from DErivBase[3] (Zeller et al., 2013). As stated above, each word pair is labeled with a derivation pattern, representing the orthographic transformation of the basis word. Since our predictive models are trained for each pattern separately, we ensure that each model will have enough training instances by discarding all patterns with less than 80 word pairs. Out of the 158 patterns in DErivBase, we retain 74 patterns, of which 49 are cross-POS patterns. The 74 patterns cover 30,757 word pairs. Patterns have a median of 194.5 word pairs (min. 83, max. 3028).

Corpus. We derive the frequency counts and the distributional vectors for our analysis from the German web corpus SdeWaC (Faaß and Eckart, 2013), POS-tagged and lemmatized using TreeTagger (Schmid, 1994). Following Kisselew et al. (2015), to mitigate sparsity, for out-of-vocabulary words we back off to the lemmas produced by MATE Tools (Bohnet, 2010), which have higher recall but lower precision than TreeTagger. We also use the MATE dependency analysis to reconstruct lemmas for separated prefix verbs.

[3] http://goo.gl/tiRJy0

Prediction Models. To obtain the vector representations on which we can train our prediction models, we use CBOW, a state-of-the-art predictive distributional semantics space which been shown particularly effective for modelling word similarity and relational knowledge (Mikolov et al., 2013).[4] (Considering the type of semantic space as a parameter is outside the scope of our study.)

As both target and context elements, we use all 280K unique POS-disambiguated lemmas (nouns, adjectives, and verbs) from DErivBase. We use a within-sentence context window of size ± 2 to either side of the target word, 300 context dimensions, negative sampling set to 15, and no hierarchical softmax. On these vector representations, we train four prediction models (cf. Section 2): the simple additive model, the simple multiplicative model, the weighted additive model, and the lexical function model. Each model is trained on each of the 74 patterns separately by minimizing the expected square loss between the predicted and the observed derived vector.[5] For additive models, the shift vector $\mathbf{p}$ is computed as the average of the shift vectors across all word pairs from a single pattern, while the weighted additive model additionally optimizes α and β in a subsequent step. Since the lexical function model is more prone to overfitting due to its many parameters, we train it using ridge regression, employing generalized cross-validation to tune the regularization parameter on the training set. As a fifth, baseline model, we use the identity mapping, which simply predicts the basis vector as the vector of the derived word. Our implementation is based on the DISSECT toolkit (Dinu et al., 2013).

Evaluation. The performance of the CDSMs is measured by how well the predicted vector aligns with the corpus-observed vector for the derived word. More concretely, we quantify the performance on each word pair by *Reciprocal rank (RR)*, that is, 1 divided by the position of the predicted vector in the similarity-ranked list of the observed vector's neighbors. Besides being a well-established evaluation measure in Information Retrieval, RR is also more sensitive than the "Recall out of n" measure used previously (Kisselew et al., 2015), which measures the 0–1 loss and also requires fixing a threshold n. RR also has the advantage of being easily interpretable: a mean reciprocal rank (MRR) of 0.33, e.g., indicates that the correct predicted vector is on average the third-nearest neighbor of the observed vector. The neighbor list for each derived word is POS-specific, that is, it consists of all words in the space that match its part of speech.

3.3 Analysis (Linear Regression) Models

The task of our analysis models is to predict the performance of the CDSM models (measured as reciprocal rank, cf. Section 3.2) at the word pair level, i.e., individual pairs of base and derived words. The goal is to assess which factors have a substantial influence on the prediction of the semantics of derived words. To this end, we use linear regression, which is a well-established analysis method in linguistics and psycholinguistics (Baayen, 2008). Linear regression predicts a dependent variable v as a linear combination of weighted predictors p_i, i.e., $v = \alpha_1 p_1 + \cdots + \alpha_n p_n$. A coefficient α_i can be interpreted as the change in v resulting from a change in the predictor p_i. We use the R statistical environment.

The right-hand side of Figure 1 shows a toy example for a single prediction model (simple additive). We first run the prediction model, then evaluate its reciprocal ranks at the word pair level, then compute features (such as the pattern and the logarithmized frequency of the base). Finally, we perform a regression analysis. It yields the information that higher frequency has a positive impact on performance, while the pattern AN34 has a negative impact.

Our complete set of predictors comprises three classes:

- **Base word level predictors** describe properties of the base word. They include `base_productivity`, the number of derived words known for the base, `base_polysemy`, the number of WordNet synsets, and `base_freq`, its lemma frequency in the SDeWaC corpus.[6] Predictor `base_typicality` is the cosine similarity between the base and the centroid of all bases for the present pattern, as a measure of how semantically typical the base is for the pattern;

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

[5] For the simple additive and multiplicative models, there are analytical solutions.

[6] All numeric variables (predictors and dependent variable) are z-scaled; frequency variables are logarithmized.

	Baseline	Simple Add	Weighted Add	Mult	LexFun
Mean Reciprocal Rank	0.271	0.309	**0.316**	0.272	0.150
# Predictions used by Oracle (Experiment 2)	2139	954	1613	532	913
# Predictions used by Regression-based Ensemble (Experiment 2)	51	2306	3528	190	76

Table 2: Results for individual prediction models on test set

- **Prediction level predictors** describe properties of the vector that the CDSM outputs. Following work on assessing the plausibility of compositionally constructed vectors by Vecchi et al. (2011), we compute the length of the vector (`deriv_norm`) and the similarity of the vector to its nearest neighbors (`deriv_density`), and the similarity between base vector and derived vector (`base_deriv_sim`);
- **Pattern level predictors**. We represent the identity of the pattern, which is admissible since we can assume that the DErivBase patterns cover the (vast) majority of German derivation patterns (Clark, 1973). Unfortunately, this excludes a range of other predictors, such as the parts of speech of the base and derived words, due to their perfect collinearity with the pattern predictor.

The rest of the paper is concerned with performing a regression analysis based on these features. We perform two separate analyses to satisfy two fairly different motivations. The first one is linguistic, namely to understand which *properties of the base and the pattern* make the prediction easy or difficult. This analysis concentrates on one single CDSM, namely the best individual one: if it included multiple CDSMs, the regression model would spend part of its power on predicting the behavior of the (arguably irrelevant) worse CDSMs. Further, this regression model should include only pattern-level and base-level features, since prediction-level features are arguably not linguistic properties. For this approach, see Section 4.

The second motivation comes from NLP and concerns the possibility to define an *ensemble* over several CDSMs that works better than individual CDSMs by employing a regression model to select the best CDSM at the word pair level. This analysis must by definition include the output of multiple prediction models. Furthermore, it should also include the features at the prediction level since they may help distinguish reasonable from unreasonable predictions. We will pursue this approach in Section 5.

4 Experiment 1: Linguistic Analysis

As outlined above, the first task in the linguistic analysis is to select the best individual prediction model for evaluation. The test set evaluation results are shown in the first row of Table 2. As the numbers show, the weighted additive model is the best model, and our analysis will focus on it. We estimate the following linear regression model to predict reciprocal rank (RR) on the development set:

```
RR ~ pattern + base_productivity + base_typicality + base_polysemy + base_freq
```

Applied to the test set, the model achieves a highly significant fit with the data (F=58.32, $p<10^{-12}$, R^2=0.324). Performance is highly variable across patterns and words pairs: results for word pairs span almost the full range of reciprocal ranks between 0 and 1, and the pattern level results range between 0.03 (pattern VV01, *zucken → zuckeln / twitch → saunter*), i.e., predictions are no good, and 0.69 (pattern AN10, *präsent → Präsenz / present → presence*), i.e., most predictions are ranked first or second. Predicted values are not correlated with the residuals ($r < 10^{-6}$). Our further discussion of this regression model is structured along a set of hypotheses we made regarding the influence of particular factors, or more specifically how they translate into distributional behavior.

Training Data and Polysemy. We start by considering the "usual suspects" in data-driven computational linguistics regarding performance, which leads us to three hypotheses. First, *low-frequency bases are hard* due to the limited reliability of the distributional evidence. Second, *atypical bases are hard*, that is,

Predictor	Estimate	LMG score
pattern	(see Table 4)	87.2%
base_productivity	−0.13***	7.6%
base_freq	0.21***	4.1%
base_polysemy	−0.03**	0.8%
base_typicality	0.04***	0.2%

Table 3: Experiment 1: Coefficients, significances, and effect sizes for the predictors

derivations of instances unlike those seen in the training data are difficult to predict. Third, derivation models must account for the selection of individual word senses in derivation: e.g., the verbal base *absetzen* variously means *depose (an official), drop (a load), deduct (an amount)*, but the derived adjective *absetzbar* is only used in the meaning of *deductable*. Since typical distributional models, including ours, do not disambiguate bases by sense, *highly polysemous bases are hard*.

Consider now Table 3, which lists coefficients, significance, and effect sizes for these predictors. Recall that we predict reciprocal rank (RR), that is, positive coefficients indicate better whereas negative coefficients indicate worse performance.[7] The data bears out our hypotheses fairly well: we find positive effects of frequency and of typicality, and negative effects of base productivity and polysemy. The relative importances of these effects is however only weakly indicated by the sizes of the coefficients. Thus, the column LMG provides normalized Lindeman-Merenda-Gold (LMG) scores (Lindeman et al., 1980), a measure of effect size (Grömping, 2012), applied, e.g., by Marelli et al. (2015) in a similar context. These scores indicate what percentage of the variance explained by the model is due to the individual predictor groups. As we see, most variance is accounted for by the `pattern` predictor. `Productivity` and `frequency` account for respectable amounts of variance, while `polysemy` and `typicality` contribute surprisingly little. This finding needs however to be interpreted taking into account that the `pattern` predictor is categorical, and as such "soaks up" all properties at the level of individual patterns, including polysemy. As a matter of fact, the correlation between RR and polysemy at the level of individual word pairs is only weak (ρ=−0.03), while MRR and average polysemy are strongly correlated at the level of derivational patterns (ρ=−0.30).

Since the bulk of the variance is accounted for by the `pattern` predictor, we now turn to formulating hypotheses about derivation patterns.

Within-POS Derivations. We first start out by considering within-POS derivations. While cross-POS derivations are at least partially motivated by the need to change the base's syntactic category, within-POS derivations primarily reflect proper semantic processes, such as polarity and gradation prefixes (*un+*, *über+* for adjectives) or prefix verbs (*hören → aufhören / hear → stop*), which are particularly prominent in German. Such affixes are known to be highly polysemous and hard to characterize both linguistically and computationally (Lechler and Roßdeutscher, 2009; Lehrer, 2009). Thus, we expect that *within-POS derivation is hard to model*.

Table 4 lists all levels of the factor `pattern` that are statistically significant from the grand mean (using contrast coding in the regression model), adopting a threshold of α=0.01. The columns correspond to the parts of speech of the base word, and the rows to the parts of speech of the derived word. Recall that negative coefficients indicate worse performance than average, and positive coefficients better-than-average performance.

The table strongly confirms our hypothesis: all five significant adjective → adjective and all seven verb → verb derivation patterns come with large negative coefficients.

Argument Structure. For cross-POS derivation, we hypothesize that *argument structure* (Grimshaw, 1990) is a major factor, connected to the largest difference to existing applications of CDSMs for phrase meaning: while in phrasal composition the resulting phrase usually shows the same semantic behavior as

[7] We use standard notation for significance (*: $p < 0.05$, **:$p < 0.01$, ***:$p < 0.001$).

Derived POS	Base POS: A			Base POS: N			Base POS: V		
A	AA03	*anti+*	−0.39 ***	NA02	*+isch*	0.52 ***	VA02	*+end*	0.48 ***
	AA07	*ab+*	−0.41 ***	NA05	*-e/en +ig*	−0.19 **	VA03	*+ig*	−0.26 **
	AA13	*nach+*	−0.41 ***	NA25	*-ung +t*	0.67 ***	VA11	*+lich*	−0.36 ***
	AA15	*über+*	−0.43 ***	NA26	*-ung +end*	0.40 **	VA12	*+end*	0.85 ***
	AA17	*vor+*	−0.42 ***	NA27	*+lich*	0.46 ***	VA13	*+t*	1.02 ***
				NA29	*+los*	−0.42 ***			
				NA31	*ge+*	−0.33 ***			
N	AN01	*+e*	−0.39 ***				VN03	*+er*	−0.24 ***
	AN02	*+heit*	0.45 ***				VN07	*+ung*	1.14 ***
	AN03	*+keit*	0.78 ***				VN09	*+en*	0.45 ***
	AN04	*+igkeit*	0.64 ***						
	AN10	*-t +z*	0.75 ***						
	AN11	*+ie*	0.95 ***						
	AN12	*-isch -ik*	0.71 ***						
	AN16	*+ität*	0.64 ***						
	AN17	*(null)*	−0.38 ***						
V	AV01	*+isieren*	0.69 ***	NV09	*(null)*	0.36 ***	VV01	*-en +eln*	−0.45 **
	AV04	*(null)*	0.28 **	NV15	*an+*	−0.31 ***	VV05	*ver+*	−0.26 ***
				NV17	*aus+*	−0.35 ***	VV12	*(stem)*	−0.36 **
				NV20	*ein+*	−0.36 ***	VV13	*an+*	−0.26 ***
				NV22	*ab+*	−0.34 ***	VV22	*ein+*	−0.21 **
							VV27	*vor+*	−0.41 ***
							VV30	*um+*	−0.28 **

Table 4: Experiment 1: Derivation patterns with significant regression model coefficients (α=0.01), cross-classified by base and derived part of speech (*null*: morphologically null derivation; *stem*: anticausative stem change, as in *legen* → *liegen, put* → *lie*)

its head component (an adjective-noun phrase behaves largely like a noun), this is not always the case in derivation. For example, the agentive nominalization pattern *-er* (*laufen* → *Läufer / run* → *runner*) incorporates the agent noun of the verb, which therefore drops out of the context of the derived word. We hypothesize that *argument structure changes are difficult to learn* for the CDSMs we consider.

Looking at Table 4, we see a mixed picture, with easy and difficult patterns. Adjective → noun derivations, which predominantly generate abstract nouns without argument structure (like AN02, *taub* → *Taubheit / deaf* → *deafness*), are overwhelmingly easy to generate. We hypothesize that the deletion of the adjective's argument is not problematic to learn. For verb → noun patterns, the default event nominalization suffix *+ung* (*umleiten* → *Umleitung / redirect* → *redirection*) and stem nominalizations (*fahren* → *Fahren / drive* → *driving*), both of which preserve argument structure, are easy to model. So are the verb → adjective patterns that form present participles (*+end*) and past participles (*+t*). In contrast, the agentive/instrumental nominalization pattern *+er* (*fahren* → *Fahrer / drive* → *driver*), where argument structure changes, is associated with a loss in performance.

We noted that those verb → adjective patterns that form property adjectives (*beachten* → *beachtlich / notice* → *noticeable*) are more challenging to model. This made us aware that difficulties associated with argument structure are mediated by an another important factor, namely *semantic regularity*. The difficulty of such patterns is related to how uniform the semantic shift is among the instances of the pattern, and how well it can be picked up by distributional analysis. As an example of semantically regular shifts, consider the significant adjective → verb patterns (AV01, AV04) which can be paraphrased as "to cause to have property X" (*anonym* → *anonymisieren / anonymous* → *anonymize*). Since there is a direct mapping from the modified head noun of the adjective onto the direct object of the verb, the distributional mapping is relatively easy, even though the shift even involves the creation of new argument structure. In contrast, some verb → adjective patterns like VA11 (*+lich*) involve the introduction of modality, a complex semantic change whose distributional consequences are hard to grasp and which is similar in nature to within-POS derivations (see above).

At the far end of the difficulty scale, we find bad performance for the noun → verb derivations, because these patterns face challenges on both the argument structure and regularity fronts: they generate verbs

from nouns that are only loosely semantically related (Clark and Clark, 1979). An example is NV22 with instances like *Zweig → abzweigen / (tree) branch → branch off*. The only easy noun → verb pattern, NV09, comes with a particularly regular semantic shift, paraphrasable as "to use X as instrument" (*Hammer → hämmern / (a) hammer → (to) hammer*).

Argument structure being a complex phenomenon, we would require additional work to exactly identify which factors play a role in derivational processes, and how those factors interact with distributional models. For instance, certain types of argument deletion/addition can result in shifting lexical items to other sentence constituents (e.g., *X developed Y over ten years* vs. *Y underwent ten years of development*). This kind of effect can, at least in principle, be captured using variable window sizes in a CDSM. Whilst we leave such questions for further research, the present results seem to support the idea that argument structure is a worthwhile aspect to investigate.

5 Experiment 2: Ensemble Prediction

In our second study, we investigate the use of linear regression models to construct an *ensemble* of CDSMs for derivation prediction. Ensemble learning is well established in NLP to capture phenomena that exceed the power of individual models (Dietterich, 2000). In our case, we want to select one vector from among the predictions of multiple CDSMs. We consider two strategies to perform this selection: The *oracle model* compares all prediction models, and simply picks the one with the highest RR. The oracle thus establishes an upper bound that assesses the theoretical benefit of model combination. It achieves an MRR of 0.362 – a modest, but substantial improvement of four and a half points over the best individual model (weighted additive, MRR=0.316, cf. Table 2).

The second strategy is the *regression model* which predicts the CDSMs' expected performances at the word pair level with a linear regression model trained on the development set (cf. Figure 1). As discussed in Section 3.3, our regression model for this purpose differs in two respects from the first study: it includes features for the prediction, and it is trained on the evaluations of all five CDSMs. The provenance of each evaluation result is coded in a new predictor, `cdsm`, with the values `baseline, simple_add, weighted_add, mult, lexfun`. We introduce interactions between `cdsm` and all base-level features to enable the regression model to learn, e.g., that some CDSMs can deal better with low-frequency bases. We estimate the following model on the development set:

```
RR ~ deriv_density + base_deriv_sim + deriv_norm + pattern +
     (base_productivity + base_typicality + base_freq + base_polysemy) * cdsm
```

On the test set, the model achieves a highly significant fit with the data (F=193.5, p$< 10^{-12}$, R^2=0.305), that is, it achieves a similar model fit to the first study.

Unfortunately, the use of this regression model to define an ensemble does not work particularly well: the ensemble yields an MRR of just 0.321, only half a point above the best-performing individual model, weighted additive, with an MRR of 0.316. This is a negative result: our regression models do not directly translate into better predictions for derived word vectors. To understand the reasons for this failure, we perform two analyses. The first one compares how many predictions of each CDSM the oracle and the ensemble selected, as shown in the lower part of Table 2. The oracle selects substantially from all models, while the regression-based ensemble chooses strictly in proportion to the CDSMs' overall performance: The best model (weighted additive) is selected for over 60% of all cases while the lexical function model is almost ignored. This indicates that the regression model is overly dependent on the `cdsm` predictor, while the base-level and pattern-level predictors are not powerful enough to reverse the bias towards higher-MRR models.

Our second analysis follows Surdeanu and Manning (2010), who found that the complementarity between participating models in an ensemble is more important than the exact combination method. To test the amount of complementarity, we computed rank correlations (Spearman's ρ) between the CDSMs' predictions at the word pair level. The results in Table 5 show that the baseline, additive, and multiplicative models are highly correlated (all pairwise ρs larger than 0.84). Only the lexical function model behaves substantially differently (pairwise ρ less than 0.34). This would make it a good candidate

	Baseline	Simple add	Weighted add	Multiplicative
Simple add	0.923			
Weighted add	0.840	0.930		
Multiplicative	0.967	0.929	0.853	
Lexical function	0.281	0.310	0.333	0.304

Table 5: Experiment 2: Correlations among CDSMs at the word level (Spearman's ρ)

for complementary predictions (as its selection by the oracle also witnesses) – however, its overall bad performance (MRR=0.150) drastically reduces its chance to be picked by the ensemble.

6 Conclusions

In this paper, we presented the first analysis of CDSMs on derivational phenomena that is both detailed and broad-coverage. Our main premise was that the linguistic features of individual lexical items, as well as the nature of the derivation pattern, would affect the extent to which the derived form could be predicted. This led us to establish relationships between linguistic properties and distributional behavior of words, a central topic in distributional semantics that seems to have received very little attention.

To quantify these relationships, we built a linear regression model with CDSM performance as dependent variable and linguistic features as predictors. An effect size analysis showed that the base term's productivity and frequency influence difficulty, but that the derivation pattern has a much larger effect. By analyzing patterns, we found that the three main factors for bad performance were: modifications of argument structure, semantic irregularity, and within-POS derivations.

Regarding the apparent contradictions among previous studies, our analysis can resolve them to some degree. We can attribute the overall bad CDSM results of Kisselew et al. (2015) to an unfortunate choice of hard within-POS derivations. At the same time, we replicate their particularly disappointing results for the lexical function, which contrasts with Lazaridou et al.'s reported performance for that model. To test whether these differences are due to Lazaridou et al.'s prefiltering, we re-evaluated all CDSMs on a "high-quality" subset of our data by throwing away the quartile with the lowest base word frequency (corresponding to a threshold of 420). The results for all models improve by 1–2%, but the lexical function model remains at 15% below the baseline. Obvious remaining differences are the language and the type of the distributional model used. However, these factors were outside the scope of the current study, so we leave them for future work.

We also built an ensemble model over the different CDSMs but did not substantially outperform the best single CDSM. We draw two conclusions from this failure: (a), despite the array of available CDSMs, it makes sense to continue developing new CDSMs to increase complementarity; and (b), the limiting factor in difficult prediction is the idiosyncratic behaviour of base words that our current distributional features capture only imperfectly.

To encourage further research, we make available our dataset with derivationally related word pairs and CDSM performance predictors.[8]

Acknowledgements

We thank the anonymous reviewers for their helpful comments. The first and third authors are supported by Deutsche Forschungsgemeinschaft (SFB 732, project B9). The second author is supported by the ERC Starting Grant COMPOSES (283554). The fourth author has been supported by the Croatian Science Foundation under the project UIP-2014-09-7312.

[8] Available at: `http://www.ims.uni-stuttgart.de/data/derivsem`

References

Harald R. Baayen, Richard Piepenbrock, and Leon Gulikers. 1996. *The CELEX lexical database. Release 2. LDC96L14*. Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA.

Harald Baayen. 2008. *Analyzing Linguistic Data*. Cambridge University Press.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*, pages 1183–1193, Cambridge, MA, USA.

Marco Baroni, Raffaela Bernardi, and Roberto Zamparelli. 2014. Frege in Space: A Program for Compositional Distributional Semantics. *LiLT (Linguistic Issues in Language Technology)*, 9:5–110.

Bernd Bohnet. 2010. Very High Accuracy and Fast Dependency Parsing is not a Contradiction. In *Proceedings of COLING*, pages 89–97, Beijing, China.

Eve V. Clark and Herbert H. Clark. 1979. When Nouns Surface as Verbs. *Language*, 55:767–811.

Herbert H Clark. 1973. The Language-as-Fixed-Effect Fallacy: A Critique of Language Statistics in Psychological Research. *Journal of verbal learning and verbal behavior*, 12(4):335–359.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical Foundations for a Compositional Distributional Model of Meaning. *Linguistic Analysis*, 36:345–386.

Thomas G. Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013. DISSECT - DIStributional SEmantics Composition Toolkit. In *Proceedings of ACL*, pages 31–36, Sofia, Bulgaria.

Wolfgang Dressler. 2005. Word-Formation in Natural Morphology. In Pavol Štekauer and Rochelle Lieber, editors, *Handbook of Word-Formation*, pages 267–284. Springer.

Katrin Erk and Sebastian Padó. 2008. A Structured Vector Space Model for Word Meaning in Context. In *Proceedings of EMNLP*, pages 897–906, Honolulu, HI, USA.

Gertrud Faaß and Kerstin Eckart. 2013. SdeWaC – A Corpus of Parsable Sentences from the Web. In *Language Processing and Knowledge in the Web*, Lecture Notes in Computer Science. Springer.

Rebecca Green, Bonnie J Dorr, and Philip Resnik. 2004. Inducing Frame Semantic Verb Classes from WordNet and LDOCE. In *Proceedings of ACL*, pages 375–382, Barcelona, Spain.

Jane Grimshaw. 1990. *Argument Structure*. MIT Press, Cambridge.

Ulrike Grömping. 2012. Estimators of Relative Importance in Linear Regression Based on Variance Decomposition. *The American Statistician*, 61(2):139–147.

Nizar Habash and Bonnie Dorr. 2003. A Categorial Variation Database for English. In *Proceedings of NAACL*, pages 17–23, Edmonton, Canada.

Jennifer Hay. 2003. *Causes and Consequences of Word Structure*. Routledge.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual Models for Compositional Distributed Semantics. In *Proceedings of ACL*, pages 58–68, Baltimore, Maryland, USA.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought Vectors. In *Proceedings of NIPS*, pages 3294–3302, Montreal, Canada.

Max Kisselew, Sebastian Padó, Alexis Palmer, and Jan Šnajder. 2015. Obtaining a Better Understanding of Distributional Models of German Derivational Morphology. In *Proceedings of IWCS*, pages 58–63, London, UK.

Brenda Laca. 2001. Derivation. In Martin Haspelmath, Ekkehard König, Wulf Oesterreicher, and Wolfgang Raible, editors, *Language Typology and Language Universals: An International Handbook*, volume 1, pages 1214–1227. Walter de Gruyter, Berlin.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly Derived Representations of Morphologically Complex Words in Distributional Semantics. In *Proceedings of ACL*, pages 1517–1526, Sofia, Bulgaria.

Andrea Lechler and Antje Roßdeutscher. 2009. German Particle Verbs with "auf". Reconstructing their Composition in a DRT-based Framework. *Linguistische Berichte*, 220:439–478.

Adrienne Lehrer. 2009. Prefixes in English Word Formation. *Folia Linguistica*, 21(1–2):133–138.

Richard H. Lindeman, Peter F. Merenda, and Ruth Z. Gold. 1980. *Introduction to Bivariate and Multivariate Analysis*. Scott Foresman, Glenview, IL, USA.

Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of CoNLL*, pages 104–113, Sofia, Bulgaria.

Marco Marelli, Simona Amenta, and Davide Crepaldi. 2015. Semantic transparency in free stems: The effect of orthography-semantics consistency on word recognition. *The Quarterly Journal of Experimental Psychology*, 68(8).

Tomas Mikolov, Wen-Tau Yih, and Geoffrey Zweig. 2013. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of HLT-NAACL*, pages 746–751, Atlanta, GA, USA.

Jeff Mitchell and Mirella Lapata. 2010. Composition in Distributional Models of Semantics. *Cognitive Science*, 34(8):1388–1429.

Sebastian Padó, Jan Šnajder, and Britta Zeller. 2013. Derivational Smoothing for Syntactic Distributional Semantics. In *Proceedings of ACL*, pages 731–735, Sofia, Bulgaria.

Ingo Plag. 2003. *Word-Formation in English*. Cambridge University Press, Cambridge.

Frans Plank. 1981. *Morphologische (Ir-)Regularitäten. Aspekte der Wortstrukturtheorie.* Narr, Tübingen.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of NeMLaP*, Manchester, UK.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of EMNLP*, pages 1631–1642, Melbourne, Australia.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble Models for Dependency Parsing: Cheap and Good? In *Proceedings of NAACL*, pages 649–652, Los Angeles, California.

Idan Szpektor and Ido Dagan. 2008. Learning Entailment Rules for Unary Templates. In *Proceedings of COLING*, pages 849–856, Manchester, UK.

Peter D Turney and Patrick Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

Eva Maria Vecchi, Marco Baroni, and Roberto Zamparelli. 2011. (Linear) Maps of the Impossible: Capturing Semantic Anomalies in Distributional Space. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*, pages 1–9, Portland, Oregon, USA.

Britta Zeller, Jan Šnajder, and Sebastian Padó. 2013. DErivBase: Inducing and Evaluating a Derivational Morphology Resource for German. In *Proceedings of ACL*, pages 1201–1211, Sofia, Bulgaria.

Survey on the Use of Typological Information in Natural Language Processing

Helen O'Horan[1]*, **Yevgeni Berzak**[2]*, **Ivan Vulić**[1],
Roi Reichart[3], **Anna Korhonen**[1]
[1] Language Technology Lab, DTAL, University of Cambridge
[2] CSAIL MIT
[3] Faculty of Industrial Engineering and Management, Technion, IIT
helen.ohoran@gmail.com berzak@mit.edu iv250@cam.ac.uk
roiri@ie.technion.ac.il alk23@cam.ac.uk

Abstract

In recent years linguistic typology, which classifies the world's languages according to their functional and structural properties, has been widely used to support multilingual NLP. While the growing importance of typological information in supporting multilingual tasks has been recognised, no systematic survey of existing typological resources and their use in NLP has been published. This paper provides such a survey as well as discussion which we hope will both inform and inspire future work in the area.

1 Introduction

One of the biggest research challenges in NLP is the huge global and linguistic disparity in the availability of NLP technology. Still, after decades of research, high quality NLP is only available for a small number of the thousands of languages in the world. Theoretically, we have two solutions to this problem: i) development of universal, language-independent models which are equally applicable to all natural language, regardless of language-specific variation; ii) comprehensive systematisation of all possible variation in different languages.

The field of linguistic typology offers valuable resources for nearing both of these theoretical ideals: it studies and classifies world's languages according to their structural and functional features, with the aim of explaining both the common properties and the structural diversity of languages. Many of the current popular solutions to multilingual NLP: transfer of information from resource-rich to resource-poor languages (Padó and Lapata, 2005; Khapra et al., 2011; Das and Petrov, 2011; Täckström et al., 2012, inter alia), joint multilingual learning (Snyder, 2010; Cohen et al., 2011; Navigli and Ponzetto, 2012, inter alia), and development of universal models (de Marneffe et al., 2014; Nivre et al., 2016, inter alia), either assume or explicitly make use of information related to linguistic typology.

While previous work has recognised the role of linguistic typology (Bender, 2011), no systematic survey of typological information resources and their use in NLP to date has been published. Given the growing need for multilingual NLP and the increased use of typological information in recent work, such a survey would be highly valuable in guiding further development. This paper provides such a survey for *structural* typology, the areas of typological theory that consider morphosyntactic and phonological features[1], which has been the main focus of typology research in both linguistics and NLP.

We begin by introducing the field of linguistic typology and the main databases currently available (§ 2). We then discuss the role and potential of typological information in guiding multilingual NLP (§ 3). In § 4 we survey existing NLP work in terms of how typological information has been developed (4.1) and integrated in multilingual application tasks (4.2). § 5 discusses future research avenues, and § 6 concludes.

*These authors contributed equally to this work.

[1]Whilst outside of the scope of this paper, other areas of linguistic typology (e.g., lexico-semantic classifications) are also of significance for the NLP community and should be addressed in future work.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1297–1308, Osaka, Japan, December 11-17 2016.

Name	Type	Coverage	Notes
World Atlas of Language Structures (WALS)	Phonology Morphosyntax Lexicosemantics	2,676 languages; 192 features; 17% of features have values	Defines language features and provides values for a large set of languages; originally intended for study of areal distribution of features
Syntactic Structures of the World's Languages (SSWL)	Morphosyntax	262 languages; 148 features; 45% of features have values	Similar to WALS, but differs in being fully open to public editing (Wikipedia-style), and by the addition of numerous example sentences for each feature
Atlas of Pidgin and Creole Language Structures (APiCS)	Phonology Morphosyntax Lexicosemantics	76 languages; 130 features; 18,526 examples	Designed to allow comparison with WALS
PHOIBLE Online	Phonology	1,672 languages; 2,160 segments	Collates and standardises several phonological segmentation databases, in addition to new data
Lyon-Albuquerque Phonological Systems Database (LAPSyD)	Phonology	422 languages	Documents a broader range of features than PHOIBLE, including syllable structures and tone systems; provides bibliographic information and links to recorded samples
URIEL Typological Compendium	Phonology Morphosyntax Lexicosemantics	8,070 languages and dialects; 284 features; approximately 439,000 feature values	Collates features from WALS, SSWL, PHOIBLE, and 'geodata'(e.g. language names, ISO codes, etc.) from sources such as Glottolog and Ethnologue; includes cross-lingual distance measures based on typological features; provides estimates for empty feature values

Table 1: An overview of major publicly accessible databases of typological information.

2 Overview of Linguistic Typology

Languages may share universal features on a deep, abstract level, but the structures found in real-world, surface-level natural language vary significantly. This variation is conventionally characterised into 'languages' (e.g. French, Hindi, Korean)[2], and linguistic typology describes how these languages resemble or differ from one another. The field comprises three pursuits: the definition of language features and their capacity for variance, the measurement and analysis of feature variance across empirical data, and the explanation of patterns observed in this data analysis. Bickel (2007) terms these three pursuits qualitative, quantitative and theoretical typology, respectively.

Typological classifications of languages have strict empirical foundations. These classifications do often support theories of causation, such as historical, areal or phylogenetic relations, but importantly, these hypotheses come second to quantitative data (Bickel, 2007). Indeed, patterns of variance may even run contrary to established theories of relations between languages based on geographical or historical proximity. For instance, Turkish and Korean are typically considered to be highly divergent in lexical features, yet their shared syntactic features make the two languages structurally quite similar. Such indications of similarity are of value for NLP which primarily seeks to model (rather than explain) cross-linguistic variation.

Typologists define and measure features according to the task at hand. Early studies, focused on word order, simply classified languages as SVO (Subject, Verb, Object), VSO, SOV, and so forth (Greenberg, 1963). There are now more various and fine-grained studies based on a wide range of features, including phonological, semantic, lexical and morphosyntactic properties (see (Bickel, 2007; Daniel, 2011) for an overview and further references). While a lot of valuable information is contained in these linguistic studies, this information is often not readily usable by NLP due to factors such as information overlap and differing definitions across studies. However, there is also a current trend towards systematically collecting typological information from individual studies in publicly-accessible databases, which are suitable for direct application in NLP (e.g., for defining features and their values).

[2]Note that there is a lacking consensus on how to define a 'language' (as opposed to a dialect, for instance) and the divisions themselves are often arbitrary and/or political. Nonetheless, the divisions are relevant insofar as they are observed in multilingual NLP.

Table 1 presents a selection of current major databases, including the Syntactic Structures of the World's Languages (SSWL) (Collins and Kayne, 2009), the World Atlas of Language Structures (WALS) (Dryer and Haspelmath, 2013), the Phonetics Information Base and Lexicon (PHOIBLE) (Moran et al., 2014), the URIEL Typological Compendium (Littel et al., 2016), the Atlas of Pidgin and Creole Language Structures (APiCS) (Michaelis et al., 2013), and the Lyon-Albuquerque Phonological Systems Database (LAPSyD) (Maddieson et al., 2013). The table provides some basic information about these databases, including type, coverage, and additional notes. From these databases, WALS is currently by far the most commonly-used typological resource in NLP due to its broad coverage of features and languages.

We next discuss the potential of typological information to guide multilingual NLP and the means by which this can be done.

3 Multilingual NLP and the Role of Typologies

The recent explosion of language diversity in electronic texts has made it possible for NLP to move increasingly towards multilingualism. The biggest challenge in this pursuit has been resource scarcity. In order to achieve high quality performance, NLP algorithms have relied heavily on manually crafted resources such as large linguistically-annotated datasets (treebanks, parallel corpora, etc.) and rich lexical databases (terminologies, dictionaries, etc.). While such resources are available for key NLP tasks (POS tagging, parsing, etc.) in well-researched languages (e.g. English, German, and Chinese), for the majority of other languages they are lacking altogether. Since resource creation is expensive and cannot be realistically carried out for all tasks in all languages, much recent research in multilingual NLP has investigated ways of overcoming the resource problem.

One avenue of research that aims to solve this problem has been unsupervised learning, which exploits unlabelled data that is now available in multiple languages. Over the past two decades increasingly sophisticated unsupervised methods have been developed and applied to a variety of tasks and in some cases also to multiple languages (Cohen and Smith, 2009; Reichart and Rappoport, 2010; Snyder, 2010; Spitkovsky et al., 2011; Goldwasser et al., 2011; Baker et al., 2014, inter alia). However, while purely unsupervised approaches are appealing in side-stepping the resource problem, their relatively low performance has limited their practical usefulness (Täckström et al., 2013). More success has been gained with solutions that use some form of supervision or guidance to enable NLP for less-resourced languages (Naseem et al., 2010; Zhang et al., 2012; Täckström et al., 2013, inter alia). In what follows, we consider three such solutions: language transfer, joint multilingual learning, and the development of universal models. We discuss the guidance employed in each, paying particular attention to typological guidance.

Language Transfer This very common approach exploits the fact that rich linguistic resources do exist for some languages. The idea is to use them for less-resourced languages via data (i.e. parallel corpora) and/or model transfer. This approach has been explored widely in NLP (Hwa et al., 2005; McDonald et al., 2011; Petrov et al., 2012; Zhang and Barzilay, 2015). It has been particularly popular in recent research on dependency parsing, where a variety of methods have been explored. For example, most work for resource-poor languages has combined delexicalised parsing with cross-lingual transfer (e.g. (Zeman and Resnik, 2008; McDonald et al., 2011; Søgaard, 2011; Rosa and Zabokrtsky, 2015)). Here, a delexicalised parser is first trained on a resource-rich source language, with both languages POS-tagged using the same tagset, and then applied directly to a resource-poor target language.

While such a transfer approach outperforms unsupervised learning, it does not achieve optimal performance. One potential reason for this is that the tagset used by a POS tagger may not fit a target language which exhibits significantly different morphological features to a source language for which the tagset was initially developed (Petrov et al., 2012). Although parallel data can be used to give additional guidance which improves transfer (McDonald et al., 2011), such data are only available for some language pairs and cannot be used in truly resource-poor situations.

An alternative direction that has recently emerged uses typological information as a form of non-parallel guidance in transfer. This direction capitalises on the fact that languages do exhibit systematic cross-lingual connections at various levels of linguistic description (e.g. similarities in language structure), despite their great diversity. Captured in typological classifications at the level of generalisation useful

for NLP, such information can be used to support multilingual NLP in a variety of ways (Bender, 2011). For example, it can be used to define the similarity between two languages with respect to the linguistic information one hopes to transfer; it can also help to define the optimal degree, level and method of transfer. For example, direct transfer of POS tagging is more likely to succeed when languages are sufficiently similar in terms of morphology in particular (Hana et al., 2004; Wisniewski et al., 2014).

Typological information has been used to guide language transfer mostly in the areas of part-of-speech tagging and parsing, e.g. (Cohen and Smith, 2009; McDonald et al., 2011; Berg-Kirkpatrick and Klein, 2010; Naseem et al., 2012; Täckström et al., 2013). Section 4 surveys such works in more detail.

Multilingual Joint Learning Another approach involves learning information for multiple languages simultaneously, with the idea that the languages will be able to support each other (Snyder, 2010; Navigli and Ponzetto, 2012). This can help in the challenging but common scenario where none of the languages involved has adequate resources. This applies even with English, where annotations needed for training basic tools are primarily available only for newspaper texts and a handful of other domains. In some areas of NLP, e.g. word sense disambiguation (Navigli and Ponzetto, 2012), multilingual learning has outperformed independent learning even for resource-rich languages, with larger gains achieved by increasing the number of languages.

Success has also been achieved on morphosyntactic tasks. For example, Snyder (2010) observes that cross-lingual variations in linguistic structure correspond to systematic variations in ambiguity, so that what one language leaves implicit, another one will not. For instance, a given word may be tagged as either a verb or a noun, yet its equivalent in other languages may not present such ambiguity. Together with his colleagues, Snyder exploited this variation to improve morphological segmentation, POS tagging, and syntactic parsing for multiple languages. Naseem et al. (2012) introduced a selective sharing approach to improve multilingual dependency parsing where the model first chooses syntactic dependents from all the training languages and then selects their language-specific ordering to tie model parameters across related languages. Because the ordering decisions are influenced by languages with similar properties, this cross-lingual sharing is modelled using typological features. In such works, typological information has been used to facilitate the matching of structural features across languages, as well as in the selection of languages between which linguistic information should be shared.

Development of Universal Models A long-standing goal that has gained renewed interest recently is the development of language-independent (i.e. *universal*) models for NLP (Bender, 2011; Petrov et al., 2012). Much of the recent interest has been driven by the Universal Dependencies (UD) initiative. It aims to develop cross-linguistically consistent treebank annotation for many languages for the purposes of facilitating multilingual parser development and cross-lingual learning (Nivre et al., 2016). The annotation scheme is largely based on universal Stanford dependencies (de Marneffe et al., 2014) and universal POS tags (Petrov et al., 2012). UD treebanks have been developed for 40 languages to date. Whilst still biased towards contemporary Indo-European languages, the collection developed by this initiative is gradually expanding to include additional language families.

The development of a truly universal resource will require taking into account typological variation for optimal coverage. For example, while the current UD scheme allows for language-specific tailoring, in the future, language type-specific tailoring may offer a useful alternative, aligned with the idea of universal modeling (Bender, 2011).

4 Development and Uses of Typological Information in NLP

Given the outlined landscape of multilingual NLP and its relation to structural typology, we now survey existing approaches for obtaining (4.1) and utilizing (4.2) typological information to support various NLP tasks.

4.1 Development of Typological Information for NLP

Typological information has been obtained using two main approaches: i) extraction from manually constructed linguistic resources, such as the databases reviewed in §2; and ii) automatic learning. The two

methods have been used independently and in combination, and both are based on the assumption (be it explicit or implicit) that typological relations may be fruitfully used in NLP.

Manual Extraction from Linguistic Resources Manually crafted linguistic resources – in particular the WALS database – have been the most commonly used sources of typological information in NLP. To date, syntactic parsing (Naseem et al., 2012; Täckström et al., 2013; Zhang and Barzilay, 2015; Ammar et al., 2016) and POS tagging (Zhang et al., 2012; Zhang et al., 2016) were the predominant areas for integration of structural information from such databases. In the context of these tasks, the most frequently used features related to word ordering according to coarse syntactic categories. Additional areas with emerging research which leverages externally-extracted typological features are phonological modeling (Tsvetkov et al., 2016; Deri and Knight, 2016) and language learning (Berzak et al., 2015).

While information obtained from typological databases has been successfully integrated in several NLP tasks, a number of challenges remain. Perhaps the most crucial challenge is the partial nature of the documentation available in manually-constructed resources. For example, WALS currently covers about 17% of its possible feature values (Dryer and Haspelmath, 2013) (see Table 1 for feature coverage of other typological databases). The integration of information from different databases is challenging due to differences in feature taxonomies as well as information overlap across repositories. Furthermore, available typological classifications contain different feature types, including nominal, ordinal and interval variables, and features that mix several types of values. This property hinders systematic and efficient encoding of such features in NLP models – a problem which thus far has only received a partial solution in the form of feature binarisation (Georgi et al., 2010). Further, typological databases are constructed manually using limited resources, and do not contain information on the distribution of feature values within a given language. This results in incomplete feature characterisations, as well as inaccurate generalisations. For example, WALS encodes only the dominant noun-adjective ordering for French, although in some cases this language also permits the adjective-noun ordering.

Other aspects of typological databases may require feature pruning and preprocessing prior to use. For example, some features in WALS, such as feature 81B "Languages with two Dominant Orders of Subject, Object, and Verb" are applicable only to a subset of the world's languages. Currently, no explicit specification for feature applicability is present in WALS or other typological resources. Furthermore, distinct features may encode overlapping information, as in the case of WALS features 81A "Order of Subject Verb and Object" and 83A "Order of Verb and Object", where the latter can be deduced from the former. Although many of these issues have been noted in previous research (Östling, 2015), there are currently no standard procedures for preprocessing typological databases for NLP use.

Despite the caveats presented above, typological resources do offer an abundance of valuable structural information which can be integrated in many NLP tasks. This information is currently substantially underutilised. Out of 192 available features in WALS, only a handful of word order features are typically used to enhance multilingual NLP. Meanwhile, the complementary information on additional languages and feature types offered by other repositories has, to our knowledge, rarely been exploited in NLP. This readily-available information could be used more extensively in tasks such as POS tagging and syntactic parsing, which have already gained from typological knowledge, and it could also be used to support additional areas of NLP.

Automatic Learning of Typological Information The partial coverage of existing typological resources, stemming from the difficulty of obtaining such information manually, have sparked a line of work on automatic acquisition of typological information. Here too, WALS has been the most common reference for defining the features to be learned.

Several approaches were introduced for automatic induction of typological information through multilingual word alignments in parallel texts. Mayer and Cysouw (2012) use alignments to induce language similarities, and use this approach to support learning of fine-grained features, such as the typology of person interrogatives (e.g., English "who"). In Östling (2015) multilingual word alignments are used to project POS tags and syntactic trees for translations of the New Testament, and subsequently learn typological information relating to word order. The predicted typological features, when evaluated against

WALS, achieve high accuracy. This method not only extends WALS word order documentation to hundreds of new languages, but also quantifies the frequency of different word orders across languages – information that is not available in manually crafted typological repositories.

Typological information can also be extracted from Interlinear Glossed Text (IGT). Such resources contain morphological segmentation, glosses and English translations of example sentences collected by field linguists. Lewis and Xia (2008) and Bender et al. (2013) demonstrate that IGT can be used to extract typological information relating to word order, case systems and determiners for a variety of languages.

Another line of work seeks to increase the coverage of typological information using existing information in typological databases. Daumé III and Campbell (2007) and Bakker (2008) use existing WALS features to learn typological implications of the kind pioneered by Greenberg (1963). Such rules can then be used to predict unknown feature values for new languages. Georgi et al. (2010) use documented WALS features to cluster languages, and subsequently predict new feature values using nearest-neighbour projection. A classifier-based approach for predicting new feature values from documented WALS information is presented in (Takamura et al., 2016). Coke et al. (2016) predict word order typological features by combining documented typological and genealogical features with the multilingual alignment approach discussed above.

An alternative approach for learning typological information uses English as a Second Language (ESL) texts (Berzak et al., 2014). This work demonstrates that morphosyntactic typological similarities between languages are largely preserved in second language structural usage. It leverages this observation to approximate typological similarities between languages directly from ESL usage patterns and further utilise these similarities for nearest neighbor prediction of typological features. The method evaluates competitively compared to baselines in the spirit of (Georgi et al., 2010) which rely on existing typological documentation of the target language for determining its nearest neighbors.

In addition, a number of studies learned typological information tailored to the particular task and data at hand (i.e. *task-based development*). For example, Song and Xia (2014) process Ancient Chinese using Modern Chinese parsing resources. They manually identify and address statistical patterns in variation between monolingual corpora in each language, and ultimately optimise the model performance by selectively using only the Modern Chinese features which correspond to Ancient Chinese features.

Although automatically-learned typological classifications have not been used frequently to date, they hold great promise for extending the use of typological information in NLP. Furthermore, such work offers an additional axis of interaction between linguistic typology and NLP, namely using computational modeling in general and NLP in particular to assist linguistic documentation and analysis of typological information. We discuss the future prospects of these research directions in § 6.

4.2 Uses of Typological Information in NLP

Multilingual Syntactic Parsing As mentioned in § 4.1, the main area of NLP in which information from structural typology has been exploited thus far is multilingual dependency parsing. In this task, a priori information about the predominant orderings of syntactic categories across languages are used to guide models when parsing a resource-poor language and using training data from other languages. This information is available in typological resources (e.g., WALS) which, among a variety of other syntactic features, list the dominant word orderings for many languages (see Table 1).

A seminal work that integrates typological word order information in multilingual dependency parsing (Naseem et al., 2012) presents the idea of "selective sharing" between source and target languages. In brief, while the identity of possible dependents for a given syntactic category is (hypothesised to be) language-universal, their ordering is language-specific. The work then presents a generative multilingual parsing model in which dependent ordering parameters are conditioned on word order typology, obtained from WALS. Specifically, the paper utilises the following word order features (henceforth WALS Basic word Order, WBO): 81A (Subject Verb and Object), 85A (Adposition and Noun), 86A (Genitive and Noun), 87A (Adjective and Noun), 88A (Demonstrative and Noun) and 89A (Numeral and Noun). This information enables the model to take into account dependent orderings only when the source language has a similar word order typology to the target language. In a similar vain, Täckström et al. (2013) present

an instance of the typologically guided selective sharing idea within a discriminative parsing framework. They group the model features into features that encode arc directionality and word order, and those that do not. The former group is then coupled with the same WBO features used by Naseem et al. (2012) via feature templates that match the WALS properties with their corresponding POS tags. Additional features that group languages according to combinations of WALS features as well as coarse language groups (Indo-European versus Altaic), result in further improvements in parsing performance.

Zhang and Barzilay (2015) extended the selective sharing approach for discriminative parsing to tensor-based models using the same WBO features as in (Naseem et al., 2012) and (Täckström et al., 2013). While traditional tensor-based parsers represent and assign non-zero weights to all possible combinations of atomic features, this work presents a hierarchical architecture that enables discarding chosen feature combinations. This allows the model to integrate prior typological knowledge, while ignoring uninformative combinations of typological and dependency features. At the same time, it capitalises on the automatisation of feature construction inherent to tensor models to generate combinations of informative typology-based features, further enhancing the added value of typological priors.

Another successful integration of externally-defined typological information in parsing is the work of Ammar et al. (2016). They present a multilingual parser trained on a concatenation of syntactic treebanks of multiple languages. To reduce the adverse impact of contradicting syntactic information in treebanks of typologically distinct languages, while still maintaining the benefits of additional training data for cross-linguistically consistent syntactic patterns, the parser encodes a language-specific bias for each given input language. This bias is based on the identity of the language and its WBO features as used in (Naseem et al., 2012; Täckström et al., 2013; Zhang and Barzilay, 2015). Differently from prior work, their parsing model also encodes all other features in the WALS profile of the relevant language. Overall, this strategy leads to improved parsing performance compared to monolingually trained baseline parsers.

While the papers surveyed above use prior information about word order typology extracted from WALS, word order information for guiding multilingual parsing can also be extracted in a bottom-up, data-driven fashion, without explicit reference to typological taxonomies. For example, in Søgaard (2011), training sentences in a source language are selected based on the perplexity of their coarse POS tag sequence under a target language POS language model. This approach essentially chooses sentences that exhibit similar word orderings in both source and target languages, thus realizing a bottom-up variant of the typology-based selective sharing methods discussed above.

There are also several methods which have made use of less explicit typological information. For instance, Berg-Kirkpatrick and Klein (2010) selectively combine languages in their method for cross-lingual dependency grammar induction using a phylogeny tree, which has been constructed from external (unspecified) knowledge of language families. Zeman and Resnik (2008) demonstrate improved performance of cross-lingually transferred dependency parsers within sets of typologically similar languages (e.g. Swedish-Danish, Hindi-Urdu); they do not explain how languages may be determined as "closely-related", though presumably this decision was based on the intuition of the researchers or on widely-acknowledged generalisations.

POS Tagging, Phonological Modeling and Language Learning Besides dependency parsing, several other areas have started integrating typological information in various forms. A number of such works revolve around the task of POS tagging. For example, in Zhang et al. (2012), the previously discussed WBO features were used to inform mappings from language-specific to a universal POS tagset. In (Zhang et al., 2016), WBO feature values are used to evaluate the quality of a multilingual POS tagger.

Another application area which benefited from integration of typological knowledge are phonological models of text. In (Tsvetkov et al., 2016) a multilingual neural phoneme-based language model is trained on several languages using a shared phonological inventory. The model is conditioned on the identity of the language at hand, as well as its phonological features obtained from a concatenation of phonological features from WALS, PHOIBLE and Ethnologue, extracted from URIEL. The resulting model subsumes and outperforms monolingually trained models for phone sequence prediction. Deri and Knight (2016) use URIEL to obtain phone and language similarity metrics, which are used for adjusting Grapheme to Phoneme (G2P) models from resource rich to resource poor languages.

Berzak et al. (2015) use typological classifications to study language learning. Formalizing the theory of "Contrastive Analysis" which aims to analyse learning difficulties in a foreign language by comparing native and foreign language structures, they build a regression model that predicts language-specific grammatical error distributions by comparing typological features in the native and foreign languages.

5 Typological Information and NLP: What's Next?

§ 4.2 surveyed the current uses of typological information in NLP. Here we discuss several future research avenues that might benefit from tighter integration of linguistic typologies and multilingual NLP.

Encoding Typological Information in Traditional Machine Learning-based NLP One of the major open challenges for typologically-driven NLP is the construction of principled mechanisms for the integration of typological knowledge in machine learning-based algorithms. Here, we briefly discuss a few traditional machine learning frameworks which support encoding of expert information, and as such hold promise for integrating typological information in NLP.

Encoding typological knowledge into machine learning requires mechanisms that can bias *learning (parameter estimation)* and *inference (prediction)* of the model towards predefined knowledge. Algorithms such as the structured perceptron (Collins, 2002) and structured SVM (Taskar et al., 2004) iterate between an inference step and a parameter update step with respect to gold training labels. The inference step is a natural place for encoding external knowledge through constraints. It biases the prediction of the model to agree with external knowledge, which, in turn, affects both the training process and the final model prediction. As typological information often reflects tendencies rather than strict rules, *soft constraints* are helpful. Ultimately, an efficient mechanism for encoding soft constraints into the inference step is needed. Indeed, several modeling approaches have been proposed that do exactly this: constraint-driven learning (CODL) (Chang et al., 2007), posterior regularisation (PR) (Ganchev et al., 2010), generalized expectation (GE) (Mann and McCallum, 2008), and dual decomposition (Globerson and Jaakkola, 2007), among others. Such approaches have been applied successfully to various NLP tasks where external knowledge is available. Examples include POS tagging and parsing (Rush et al., 2010; Rush et al., 2012), information extraction (Riedel and McCallum, 2011; Reichart and Barzilay, 2012), and discourse analysis (Guo et al., 2013), among others. In addition to further extensions to the modeling approaches surveyed in §4.2, these type of frameworks could expedite principled integration of typological information in NLP.

Typologies and Multilingual Representation Learning While the traditional models surveyed above assume a predefined feature representation and focus on generating the best prediction of the output labels, a large body of recent NLP research has focused on learning dense real-valued vector representations —- i.e., word embeddings (WEs). WEs serve as pivotal features in a range of downstream NLP tasks such as parsing, named entity recognition, and POS tagging (Turian et al., 2010; Collobert et al., 2011; Chen and Manning, 2014). The extensions of WE models in bilingual and multilingual settings (Klementiev et al., 2012; Hermann and Blunsom, 2014; Coulmance et al., 2015; Vulić and Moens, 2016, inter alia) abstract over language-specific features and attempt to represent words from several languages in a language-agnostic manner such that similar words (regardless of the actual language) obtain similar representations. Such multilingual WEs facilitate cross-lingual learning, information retrieval and knowledge transfer. The extent to which multilingual WEs capture word meaning across languages has been recently evaluated in (Leviant and Reichart, 2015) with the conclusion that multilingual training usually improves the alignment between the induced WEs and the meaning of the participating words in each of the involved languages.

Naturally, as these models become more established and better understood, the challenge of external knowledge encoding becomes more prominent. Recent work has examined the ability to map from word embeddings to interpretable typological representations (Qian et al., 2016). Furthermore, a number of works (Faruqui et al., 2015; Rothe and Schütze, 2015; Osborne et al., 2016; Mrkšić et al., 2016) proposed means through which external knowledge from structured knowledge bases and specialised linguistic resources can be encoded in these models. The success of these works suggests that more extensive integration of external linguistic knowledge in general, and typological knowledge in particular, is likely to play a key role in the future development of WE representations.

Can NLP Support Typology Construction? As discussed in §4, typological resources are commonly constructed manually by linguists. Despite the progress made in recent years in the digitisation and collection of typological knowledge in centralised repositories, their coverage remains limited. Following the work surveyed in §4.1 on automatic learning of typological information, we believe that NLP could play a much larger role in the study of linguistic typology and the expansion of such resources. Future work in these directions will not only assist in the global efforts for language documentation, but also substentially extend the usability of such resources for NLP purposes.

6 Commentary; conclusion

This paper has provided a survey of linguistic typologies and the many recent works in multilingual NLP that have benefited from such resources. We have shown how combined knowledge of linguistic universals and typological variation has been used to improve NLP by enabling the use of cross-linguistic data in the development and application of resources. Promising examples of both explicit and implicit typological awareness in NLP have been presented. We have concluded with a discussion on how typological information could be used to inform improved experimental and conceptual practice in NLP. We hope that this survey will be useful in both informing and inspiring future work on linguistic typologies and multilingual NLP.

Acknowledgments

This work is supported by ERC Consolidator Grant LEXICAL (no 648909) and by the Center for Brains, Minds and Machines (CBMM) funded by the NSF STC award CCF-1231216. The authors are grateful to the anonymous reviewers for their helpful comments and suggestions.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Many languages, one parser. *TACL*.

Simon Baker, Roi Reichart, and Anna Korhonen. 2014. An unsupervised model for instance level subcategorization acquisition. In *EMNLP*, pages 278–289.

Dik Bakker. 2008. INFER: Inferring implications from the WALS database. *Sprachtypologie und Universalienforschung*, 3(61):186–198.

Emily M. Bender, Michael Wayne Goodman, Joshua Crowgey, and Fei Xia. 2013. Towards creating precision grammars from interlinear glossed text: Inferring large-scale typological properties. In *LaTeCH 2013*.

Emily M. Bender. 2011. On achieving and evaluating language-independence in NLP. *Linguistic Issues in Language Technology*, 3(6):1–26.

Taylor Berg-Kirkpatrick and Dan Klein. 2010. Phylogenetic grammar induction. In *ACL*, pages 1288–1297.

Yevgeni Berzak, Roi Reichart, and Boris Katz. 2014. Reconstructing native language typology from foreign language usage. In *CoNLL*, pages 21–29.

Yevgeni Berzak, Roi Reichart, and Boris Katz. 2015. Contrastive analysis with predictive power: Typology driven estimation of grammatical error distributions in ESL. In *CoNLL*, pages 94–102.

Balthasar Bickel. 2007. Typology in the 21st century: Major current developments. *Linguistic Typology*, 11(1):239–251.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *ACL*, pages 280–287.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750.

Shay Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *NAACL-HLT*, pages 74–82.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *EMNLP*, pages 50–61.

Reed Coke, Ben King, and Dragomir R. Radev. 2016. Classifying syntactic regularities for hundreds of languages. *CoRR*, abs/1603.08016.

Chris Collins and Richard Kayne. 2009. Syntactic structures of the world's languages. `http://sswl.railsplayground.net/`.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. 2015. Trans-gram, fast cross-lingual word embeddings. In *EMNLP*, pages 1109–1113.

Michael Daniel. 2011. Linguistic typology and the study of language. In *The Oxford Handbook of Linguistic Typology*, pages 43–68.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *ACL*, pages 600–609.

Hal Daumé III and Lyle Campbell. 2007. A Bayesian model for discovering typological implications. In *ACL*, pages 65–72.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *LREC*, pages 4585–4592.

Aliya Deri and Kevin Knight. 2016. Grapheme-to-phoneme models for (almost) any language. In *ACL*, pages 399–408.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *NAACL-HLT*, pages 1606–1615.

Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

Ryan Georgi, Fei Xia, and William Lewis. 2010. Comparing language similarity across genetic and typologically-based groupings. In *COLING*, pages 385–393.

Amir Globerson and Tommi S Jaakkola. 2007. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *NIPS*, pages 553–560.

Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. In *ACL*, pages 1486–1495.

Joseph H. Greenberg. 1963. Some universals of grammar with particular reference to the order of meaningful elements. *Universals of Language*, 2:73–113.

Yufan Guo, Roi Reichart, and Anna Korhonen. 2013. Improved information structure analysis of scientific documents through discourse and lexical constraints. In *NAACL-HLT*, pages 928–937.

Jiri Hana, Anna Feldman, and Chris Brew. 2004. A resource-light approach to Russian morphology: Tagging Russian using Czech resources. In *EMNLP*, pages 222–229.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *ACL*, pages 58–68.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara I. Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(3):311–325.

Mitesh M. Khapra, Salil Joshi, Arindam Chatterjee, and Pushpak Bhattacharyya. 2011. Together we can: Bilingual bootstrapping for WSD. In *ACL*, pages 561–569.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *COLING*, pages 1459–1474.

Ira Leviant and Roi Reichart. 2015. Separated by an un-common language: Towards judgment language informed vector space modeling. *arXiv preprint arXiv:1508.00106*.

William D Lewis and Fei Xia. 2008. Automatically identifying computationally relevant typological features. In *IJCNLP*, pages 685–690.

Patrick Littel, David R. Mortensen, and Lori Levin. 2016. URIEL Typological database. Pittsburgh: CMU.

Ian Maddieson, Sébastien Flavier, Egidio Marsico, Christophe Coupé, and François Pellegrino. 2013. LAPSyd: Lyon-Albuquerque phonological systems database. In *INTERSPEECH*, pages 3022–3026.

Gideon S. Mann and Andrew McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *ACL*, pages 870–878.

Thomas Mayer and Michael Cysouw. 2012. Language comparison through sparse multilingual word alignment. In *EACL 2012 Joint Workshop of LINGVIS & UNCLH*, pages 54–62.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *EMNLP*, pages 62–72.

Susanne Maria Michaelis, Philippe Maurer, Martin Haspelmath, and Magnus Huber, editors. 2013. *Atlas of Pidgin and Creole Language Structures Online*.

Steven Moran, Daniel McCloy, and Richard Wright, editors. 2014. *PHOIBLE Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *NAACL-HLT*, pages 142–148.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proc. of EMNLP 2010*.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *ACL*, pages 629–637.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Joining forces pays off: Multilingual joint word sense disambiguation. In *EMNLP-CoNLL*, pages 1399–1410.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *LREC*, pages 1659–1666.

D. Osborne, S. Narayan, and S. B. Cohen. 2016. Encoding prior knowledge with eigenword embeddings. *Transactions of the ACL (to appear)*.

Robert Östling. 2015. Word order typology through multilingual word alignment. In *ACL*, pages 205–211.

Sebastian Padó and Mirella Lapata. 2005. Cross-linguistic projection of role-semantic information. In *EMNLP*, pages 859–866.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *LREC*, pages 2089–2096.

Peng Qian, Xipeng Qiu, and Xuanjing Huang. 2016. Investigating language universal and specific properties in word embeddings. In *ACL*, pages 1478–1488.

Roi Reichart and Regina Barzilay. 2012. Multi event extraction guided by global constraints. In *NAACL*, pages 70–79.

Roi Reichart and Ari Rappoport. 2010. Improved fully unsupervised parsing with zoomed learning. In *EMNLP*, pages 684–693.

Sebastian Riedel and Andrew McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *EMNLP*, pages 1–12.

Rudolf Rosa and Zdenek Zabokrtsky. 2015. KLcpos3 - a language similarity measure for delexicalized parser transfer. In *ACL*, pages 243–249.

Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In *ACL*, pages 1793–1803.

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *EMNLP*, pages 1–11.

Alexander M. Rush, Roi Reichart, Michael Collins, and Amir Globerson. 2012. Improved parsing and POS tagging using inter-sentence consistency constraints. In *EMNLP-CoNLL*, pages 1434–1444.

Ben Snyder. 2010. *Unsupervised Multilingual Learning*. PhD thesis. MIT.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *ACL*, pages 682–686.

Yan Song and Fei Xia. 2014. Modern Chinese helps archaic Chinese processing: Finding and exploiting the shared properties. In *LREC*, pages 3129–3136.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011. Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *EMNLP*, pages 1269–1280.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *NAACL-HLT*, pages 477–487.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *NAACL-HLT*, pages 1061–1071.

Hiroya Takamura, Ryo Nagata, and Yoshifumi Kawasaki. 2016. Discriminative analysis of linguistic features for typological study. In *LREC*, pages 69–76.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin Markov networks. In *NIPS*, pages 25–32.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W. Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. In *NAACL*, pages 1357–1366.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL*, pages 384–394.

Ivan Vulić and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research*, 55:953–994.

Guillaume Wisniewski, Nicolas Pécheux, Souhir Gahbiche-Braham, and François Yvon. 2014. Cross-lingual part-of-speech tagging through ambiguous learning. In *EMNLP*, pages 1779–1785.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *IJCNLP 2008 Workshop on NLP for Less Privileged Languages*, pages 35–42.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *EMNLP*, pages 1857–1867.

Yuan Zhang, Roi Reichart, Regina Barzilay, and Amir Globerson. 2012. Learning to map into a universal POS tagset. In *EMNLP*, pages 1368–1378.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016. Ten pairs to tag – multilingual POS tagging via coarse mapping between embeddings. In *NAACL*, pages 1307–1317.

From phonemes to images: levels of representation in a recurrent neural model of visually-grounded language learning

Lieke Gelderloos
Tilburg University
liekegelderloos@gmail.com

Grzegorz Chrupała
Tilburg University
g.chrupala@uvt.nl

Abstract

We present a model of visually-grounded language learning based on stacked gated recurrent neural networks which learns to predict visual features given an image description in the form of a sequence of phonemes. The learning task resembles that faced by human language learners who need to discover both structure and meaning from noisy and ambiguous data across modalities. We show that our model indeed learns to predict features of the visual context given phonetically transcribed image descriptions, and show that it represents linguistic information in a hierarchy of levels: lower layers in the stack are comparatively more sensitive to form, whereas higher layers are more sensitive to meaning.

1 Introduction

Children acquire their native language with little and weak supervision, exploiting noisy correlations between speech, visual, and other sensory signals, as well as via feedback from interaction with their peers and parents. Understanding this process is an important scientific challenge in its own right, but it also has potential to generate insights useful in engineering efforts to design conversational agents or robots. Computationally modeling the ability to learn linguistic form–meaning pairings has been the focus of much research, under scenarios simplified in a variety of ways, for example:

- distributional learning from pure word-word co-occurrences with no perceptual grounding (Landauer et al., 1998; Kiros et al., 2015);
- cross-situational learning with word sequences and sets of symbols representing sensory input (Siskind, 1996; Fazly et al., 2010);
- cross-situational learning using sensory audio and visual input, but with extremely limited sets of words and objects (Roy and Pentland, 2002; Iwahashi, 2003).

Some recent models have used more naturalistic, larger-scale inputs, for example in cross-modal distributional semantics (Lazaridou et al., 2015) or in implementations of the acquisition process trained on images paired with their descriptions (Chrupała et al., 2015). While in these works the representation of the *visual scene* consists of pixel-level perceptual data, the *linguistic* input consists of sentences segmented into discrete word symbols. In this paper we take a step towards addressing this major limitation, by using the phonetic transcription of input utterances. While this type of input is symbolic rather than perceptual, it goes a long way toward making the setting more naturalistic, and the acquisition problem more challenging: the learner may need to discover structure corresponding to morphemes, words and phrases in an unsegmented string of phonemes, and the length of the dependencies that need to be detected grows substantially when compared to word-level models.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1309–1319, Osaka, Japan, December 11-17 2016.

Our contributions We design and implement a simple architecture based on stacked recurrent neural networks with Gated Recurrent Units (Chung et al., 2014): our model processes the utterance phoneme by phoneme while building a distributed low-dimensional semantic representation through a series of recurrent layers. The model learns by projecting its sentence representation to image space and comparing it to the features of the corresponding visual scene.

We train this model on a phonetically transcribed version of MS-COCO (Lin et al., 2014) and show that it is able to successfully learn to understand aspects of sentence meaning from the visual signal, and exploits temporal structure in the input. In a number of experiments we show that different levels in the stack of recurrent layers represent different aspects of linguistic structure. Low levels focus on local, short-time-scale spans of the input sequence, and are comparatively more sensitive to form. The top level encodes global aspects of the input sequence and is sensitive to visually salient elements of its meaning.

2 Related work

A major part of learning language consists in learning its structure, but in order to be able to communicate it is also of crucial importance to learn the relation of words to entities in the world. Grounded lexical acquisition is often modeled as cross-situational learning, a process of rule-based (Siskind, 1996) or statistical (Fazly et al., 2010; Frank et al., 2007) inference of word-to-referent mappings. Cross-situational models typically work on word-level language input and symbolic representations of the context. However, infants have to learn from continuous perceptual input. Lazaridou et al. (2016) partially remedy this shortcoming and propose a model of learning word meanings from text paired with continuous image representations; the limitation of their work is the toy evaluation dataset.

Recent experimental and computational studies have found that co-occurring visual information may help to learn word forms (Thiessen, 2010; Cunillera et al., 2010; Glicksohn and Cohen, 2013; Yurovsky et al., 2012). This suggests that acquisition of word form and meaning are interactive, rather than separate.

The Cross-channel Early Lexical Learning (CELL) model of Roy and Pentland (2002) and the more recent work of Räsänen and Rasilo (2015) take into account the continuous nature of the speech signal, and incorporate visual information as well. The CELL model learns to discover words in continuous speech through co-occurence with their visual referent, but the visual input only consists of the shape of single objects, effectively bypassing referential uncertainty. Räsänen and Rasilo (2015) propose a probabilistic joint model of word segmentation and meaning acquisition from raw speech and a set of possible referents that appear in the context. In both Roy and Pentland (2002) and Räsänen and Rasilo (2015) the visual context is considerably less noisy and ambiguous than that available to children.

There is an extensive line of research on image captioning (see Bernardi et al. (2016) for a recent overview). Typically, captioning models learn to recognize high-level image features and associate them with words. Inspired by both image captioning research and cross-situational human language acquisition, two recent automatic speech recognition models learn to recognize word forms from visual data. In Synnaeve et al. (2014), language input consists of single spoken words and visual data consists of image fragments, which the model learns to associate. Harwath and Glass (2015) employ two convolutional neural networks, a visual object recognition model and a word recognition model, and an embedding alignment model that learns to map recognized words and objects into the same high-dimensional space. Although the object recognition works on the raw visual input, the speech signal is segmented into words before presenting it to the word recognition model. As both Harwath and Glass (2015) and Synnaeve et al. (2014) work with word-sized chunks of speech, they bypass the segmentation problem that human language learners face.

Character-level input representations have recently gained attention in NLP. Ling et al. (2015) and Plank et al. (2016) use bidirectional LSTMs to compose characters into word embeddings, while Chung et al. (2016) propose machine translation model with character level output. These approaches exploit character-level information but crucially they assume that word boundaries are available in the input.

Character-level neural NLP *without* explicit word boundaries in the input is studied in cases where fixed vocabularies are inherently problematic, e.g. with combined natural and programming language

input (Chrupała, 2013) or when specifically dealing with misspelled words in automatic writing feedback (Xie et al., 2016).

Character-level language models are analyzed in Hermans and Schrauwen (2013) and Karpathy et al. (2015). Both studies show that character-level recurrent neural networks are sensitive to long-range dependencies: for example by keeping track of opening and closing parentheses over stretches of text. Hermans and Schrauwen (2013) describe the hierarchical organization that emerges during training, with higher layers processing information over longer timescales. In our work we show related effects in a model of visually-grounded language learning from unsegmented phonemic strings.

We use phonetic transcription of full sentences as a first step towards large-scale multimodal language learning from speech co-occurring with visual scenes. The use of phonetic transcription rather than raw speech signal simplifies learning and allows us to perform experiments on the encoding of linguistic knowledge as reported in section 4 without additional annotation. Our goal is to model a multi-modal language learning process that includes the segmentation problem faced by human language learners. In contrast to character-level NLP and language modeling approaches, our input data therefore does not contain word boundaries or strong cues such as whitespace and punctuation.

In contrast to Roy and Pentland (2002) and Räsänen and Rasilo (2015), the visual input to our model consists of high-level visual features, which means it contains ambiguity and noise. In contrast to Synnaeve et al. (2014) and Harwath and Glass (2015), we consider full utterances rather than separate words.

To our knowledge, there is no work yet on multimodal phoneme or character-level language modeling with visual input. Although the language input in this study is low-level-symbolic rather than perceptual, the learning problem is similar to that of a human language learner: discover language structure as well as meaning, based on ambiguous and noisy data from another modality.

Chrupała et al. (2015) simulate visually grounded human language learning in face of noise and ambiguity in the visual domain. Their model predicts visual context given a sequence of words. While the visual input consists of a continuous representation, the language input consists of a sequence of words. The aim of this study is to take their approach one step further towards multimodal language learning from raw perceptual input. Kádár et al. (2016) develop techniques for understanding and interpretation of the representations of linguistic form and meaning in recurrent neural networks, and apply these to word-level models. In our work we share the goal of revealing the nature of emerging representations, but we do not assume words as their basic unit. Also, we are especially concerned with the emergence of a hierarchy of levels of representations in stacked recurrent networks.

3 Models

Consider a learner who sees a person pointing at a scene and uttering the unfamiliar phrase *Look, the monkeys're playing*. We may suppose that the learner will update her language understanding model such that the subsequent utterance of this phrase will evoke in her mind something close to the impression of this visual scene. Our model is a particular instantiation of this simple idea.

3.1 Phon GRU

The architecture of our main model of interest, PHON GRU is schematically depicted in Figure 1 and consists of a phoneme encoding layer, followed by a stack of K Gated Recurrent Neural nets, followed by a densely connected layer which maps the last hidden state of the top recurrent layer to a vector of visual features.

Gated Recurrent Units (GRU) were introduced in Cho et al. (2014) and Chung et al. (2014) as an attempt to alleviate the problem of vanishing gradient in standard simple recurrent nets as known since the work of Elman (1990). GRUs have a linear shortcut through timesteps which bypasses the nonlinearity and thus promotes gradient flow. Specifically, a GRU computes the hidden state at current time step, $\mathbf{h}_t$, as the linear combination of previous activation $\mathbf{h_{t-1}}$, and a new *candidate* activation $\tilde{\mathbf{h}}_t$:

$$\mathrm{gru}(\mathbf{x}_t, \mathbf{h}_{t-1}) = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \quad (1)$$

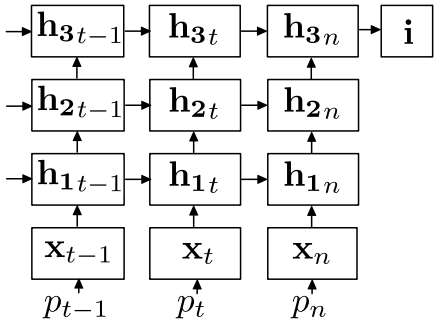


Figure 1: A three-timestep slice of the stacked recurrent architecture with three hidden layers.

ɐjʌŋwʊmənɹaɪdɪŋəhɔːshəʊldɪŋəflag

A young woman riding a horse holding a flag

Figure 2: (Top) Example of a postprocessed phonetic transcription from eSpeak used as input to the PHON GRU model. (Bottom) Corresponding image.

where $\odot$ is elementwise multiplication, and the update gate activation $\mathbf{z_t}$ determines the amount of new information mixed in the current state:

$$\mathbf{z}_t = \sigma_s(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1}) \tag{2}$$

The candidate activation is computed as:

$$\tilde{\mathbf{h}}_t = \sigma(\mathbf{W}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1})) \tag{3}$$

The reset gate $\mathbf{r_t}$ determines how much of the current input $\mathbf{x_t}$ is mixed in the previous state $\mathbf{h}_{t-1}$ to form the candidate activation:

$$\mathbf{r}_t = \sigma_s(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1}) \tag{4}$$

By applying the gru function repeatedly a GRU layer maps a sequence of inputs to a sequence of states:

$$\mathrm{GRU}(\mathbf{X}, \mathbf{h}_0) = \mathrm{gru}(\mathbf{x}_n, \ldots, \mathrm{gru}(\mathbf{x}_2, \mathrm{gru}(\mathbf{x}_1, \mathbf{h}_0))) \tag{5}$$

where $\mathbf{X}$ stands for the matrix composed of input column vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Two or more GRU layers can be composed into a stack:

$$\mathrm{GRU}_2(\mathrm{GRU}_1(\mathbf{X}, \mathbf{h_{1}}_0), \mathbf{h_{2}}_0). \tag{6}$$

In our version of the Stacked GRU architecture we use *residualized* layers:

$$\mathrm{GRU}_{\mathrm{res}}(\mathbf{X}, \mathbf{h}_0) = \mathrm{GRU}(\mathbf{X}, \mathbf{h}_0) + \mathbf{X} \tag{7}$$

Residual convolutional networks were introduced by He et al. (2015), while Oord et al. (2016) showed their applicability to recurrent architectures. We adopt residualized layers here as we observed they speed up learning in stacks of several GRU layers.

Our gated recurrent units use steep sigmoids for gate activations:

$$\sigma_s(z) = \frac{1}{1 + \exp(-3.75z)}$$

and rectified linear units clipped between 0 and 5 for the unit activations:

$$\sigma(z) = \mathrm{clip}(0.5(\mathrm{z} + \mathrm{abs}(\mathrm{z})), 0, 5)$$

There are two more components of our PHON GRU model: the phoneme encoding layer, and mapping from the final state of the top GRU layer to the image feature vector. The phoneme encoding layer is a simply a lookup table $\mathbf{E}$ whose columns correspond to one-hot-encoded phoneme vectors. The input

phoneme p_t of utterance p at each step t indexes into the encoding matrix and produces the input column vector:

$$\mathbf{x}_t = \mathbf{E}[:, p_t]. \tag{8}$$

Finally, we map the final state of the top GRU layer $\mathbf{h}_{\mathbf{K}n}$ to the vector of image features using a fully connected layer:

$$\hat{\mathbf{i}} = \mathbf{I}\mathbf{h}_{\mathbf{K}n} \tag{9}$$

Our main interest lies in recurrent phoneme-level modeling. However, in order to put the performance of the phoneme-level PHON GRU into perspective, we compare it to two word-level models. Importantly, the word models should **not** be seen as baselines, as they have access to word boundary and word identity information not available to PHON GRU.

3.2 Word GRU

The architecture of this model is the same as PHON GRU with the difference that we use words instead of phonemes as input symbols, use learnable word embeddings instead of fixed one-hot phoneme encodings, and reduce the number of layers in the GRU stack. See Section 4 for details.

3.3 Word Sum

The second model we use for comparison is a word-based non-sequential model, consisting of a word embedding matrix, a vector sum operator, and a mapping to the image feature vector:

$$\hat{\mathbf{i}} = \mathbf{I}\sum_{t=1}^{n}\mathbf{E}[:, w_t] \tag{10}$$

where w_t is the word at position t in the input utterance. This model simply learns word embeddings which are then summed into a single vector and projected to the target image vector. Thus this model does not have access to word sequence information, and is a distributed analog of a bag-of-words model.

4 Experiments

For all experiments, the models were trained on the training set of MS-COCO. MS-COCO contains over 163,000 images accompanied by at least five captions by human annotators. With an average of 7.7 labeled object instances per image, images typically contain more objects than the captions mention, making reference to the scene ambiguous. Textual input for the PHON GRU models was transcribed automatically using the grapheme-to-phoneme functionality with the default English voice of the eSpeak speech synthesis toolkit.[1] Stress and pause markers were removed, as well as word boundaries (after storing their position for use in experiments), leaving only phoneme symbols. See Figure 2 for an example transcription.

Visual input for all models was obtained by forwarding images through the 16-layer convolutional neural network described in Simonyan and Zisserman (2014) pre-trained on Imagenet (Russakovsky et al., 2014), and recording the activation vectors of the pre-softmax layer. The z-score transformation was applied to these features to ease optimization.

Most of the details of the three model types and training hyperparameters were adopted from related work, and adapted via a limited amount of exploratory experimentation. Exhaustive exploration of the search space was not feasible due to the large number of adjustable settings in these models and their long running time. Given the importance of depth for our purposes, we did systematically explore the number of layers in the PHON GRU and WORD GRU models. A single layer is optimal for WORD GRU. For PHON GRU, see Section 4.1 below. Other important settings were as follows:

- All models: Implemented in Theano (Bastien et al., 2012), optimized with Adam (Kingma and Ba, 2014), initial learning rate of 0.0002, minibatch size of 64, gradient norm clipped to 5.0.

[1] Available at `http://espeak.sourceforge.net`

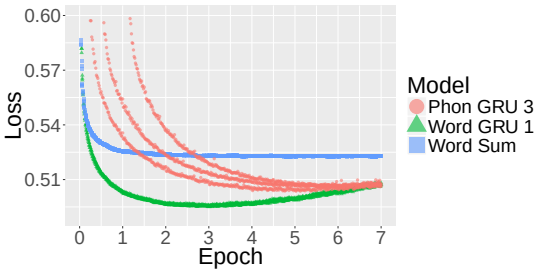


Figure 3: Value of the loss function on validation data during training. Three random initialization of each model are shown.

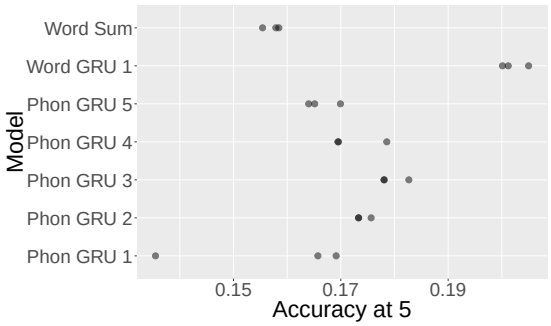


Figure 4: Validation accuracy at 5 on the image retrieval task.

- WORD SUM: 1024-dimensional word embeddings, words with frequencies below 10 replaced by UNK token.
- WORD GRU: 1024-dimensional word embeddings, a single 1024 dimensional hidden layer, words with frequencies below 10 replaced by UNK token.
- PHON GRU: 1024-dimensional hidden layers.

4.1 Prediction of visual features

The models are trained and evaluated on the prediction of visual feature vectors from captions. While our goal is not to develop an image retrieval method, we use this task as it reflects the ability to extract visually salient semantic information from language. For the experiments on the prediction of visual features all models were trained on the training set of MS-COCO. As validation and test data we used a random sample of 5000 images each from the MS-COCO validation set.

Figure 3 shows the value of the validation average cosine distance between the predicted visual vector and the target vector for three random initializations of each of the model types.

The Phonetic GRU model is more sensitive to the initialization: one can clearly distinguish three separate trajectories. The word-level models are much less affected by random initialization. In terms of the overall performance, the PHON GRU model falls between the WORD SUM model and the WORD GRU model.

We also evaluated the models on how well they perform when used to search images: for each validation sentence the model was used to predict the visual vector. The image vectors in the validation data were then ranked by cosine similarity to the predicted vector, and the proportion of times the correct image was among the top 5 was reported. By *correct* image we mean the one which the sentence was used to describe (even though often many other images are also good matches to the sentence).

In Figure 4 we report the validation accuracies on this task for the two word-level models, as well as for the Phon GRU model with different number of hidden layers. We trained each model version with three random initializations for each model setting, and evaluate after each epoch. We report the score of the best epoch for each initialization. The overall ranking of the models matches the direct evaluation of the loss function above: the phoneme-level models are in between the two word-level models. PHON GRU with three hidden layers is the best of the phoneme-level models.

In Table 1 we show the accuracies of the best version of each of the models types on the test images; these are also the model versions used in all subsequent experiments. The scores for the WORD GRU are comparable to what Chrupała et al. (2015) report for their multitask IMAGINET model, whose visual pathway has the same structure, and who use the same data. More recently, Vendrov et al. (2016) report

substantially higher scores for image search with a word-level GRU model, with the following main differences from our setting: better image features, larger training set, and a loss function optimized for the ranking task.[2]

4.2 Word boundary prediction

To explore the sensitivity of the PHON GRU model to linguistic structure at the sub-word level, we investigated the encoding of information about word-boundaries in the hidden layers. Logistic regression models were trained on activation patterns of the hidden layers at all timesteps, with the objective of identifying phonemes that preceded a word boundary. For comparison, we also trained logistic regression models on *n*-gram data to perform the same tasks, with positional phoneme *n*-grams in the range 1-*n*. The location of the word boundaries was taken from the eSpeak transcriptions, which mostly matches the location of word boundaries according to conventional English spelling. However, eSpeak models some coarticulation effects which sometimes leads to word boundaries disappearing from the transcription. For example, *bank of a river* is transcribed as [baŋk əvə ɪɪvə].

All models were implemented using the `LogisticRegression` implementation from Scikit-learn (Pedregosa et al., 2011) with L2-regularization. The random samples of 5,000 images each that served as validation and test sets in the visual feature prediction task were used as training and test sets. For the models based on the activation patterns of the hidden layers, the z-score transformation was applied to the activation values to ease optimization. The optimal value of regularization parameter *C* was determined using `GridSearchCV` with 5-fold cross validation on the training set, after which the model with the optimal settings was trained on the full training sample.

Table 2 reports the scores on the test set. The proportion of phonemes preceding a word boundary is 0.29, meaning that predicting *no word boundary* by default would be correct in 0.71 of cases. At the highest hidden layer, enough information about the word form is available for correct prediction in 0.82 of cases – substantially above the majority baseline. The lower levels allow for more accurate prediction of word boundaries: 0.86 at the middle hidden layer, and 0.88 at the bottom level. Prediction scores of the logistic regression model based on the activation patterns of the lowest hidden layer are comparable to those of a bigram logistic regression model.

These results indicate that information on sub-word structure is only partially encoded by PHON GRU, and is mostly absent by the time the signal from the input propagates to the top layer. The bottom layer does learn to encode a fair amount of word boundary information, but the prediction score substantially below 100% indicates that it is rather selective.

Model	Acc @ 5	Acc @ 10
WORD SUM	0.158	0.243
WORD GRU	0.205	0.306
PHON GRU	0.180	0.276

Table 1: Image retrieval accuracy at 5 and at 10 on test data for the versions of WORD SUM, WORD GRU and PHON GRU chosen by validation.

Model		**Acc**	**Prec**	**Rec**
Majority		0.71		
Phon GRU	Layer 1	0.88	0.82	0.78
	Layer 2	0.86	0.79	0.71
	Layer 3	0.82	0.74	0.60
n-gram	$n = 1$	0.80	0.79	0.41
	$n = 2$	0.87	0.79	0.78
	$n = 3$	0.93	0.86	0.90
	$n = 4$	0.95	0.90	0.93

Table 2: Prediction scores of logistic regression models based on activation vectors of PHON GRU and on positional *n*-grams

4.3 Word similarity

To understand the encoding of semantic information in PHON GRU, we analyzed the cosine similarity of activation vectors for word pairs from the MEN Test Collection (Bruni et al., 2014). The MEN

[2]We have preliminary results indicating that most of the analyses in the rest of Section 4 show the same general pattern for phoneme models trained following the setting of Vendrov et al. (2016).

dataset contains 3,000 pairs of English words with semantic similarity judgements on a 50-point scale, which were obtained through crowd-sourcing.[3] For each word pair in the MEN dataset, the words were transcribed phonetically using eSpeak and then fed to PHON GRU individually. For comparison, the words were also fed to WORD GRU and WORD SUM. Word pair similarity was quantified as the cosine similarity between the activation patterns of the hidden layers at the end-of-sentence symbol. In contrast to WORD GRU and WORD SUM, PHON GRU has access to the sub-word structure. To explore the role of phonemic form in word similarity, a measure of phonemic difference was included: the Levenshtein distance between the phonetic transcriptions of the two words, normalized by the length of the longer transcription.

Table 3 shows Spearman's rank correlation coefficient between human similarity ratings from the MEN dataset and cosine similarity at the last timestep for all hidden layers. In all layers, the cosine similarities between the activation vectors for two words are significantly correlated with human similarity judgements. The strength of the correlation differs considerably between the layers, ranging from 0.09 in the first layer to 0.28 in the highest hidden layer. The second column in Table 3 shows the correlations when only taking into account the 1283 word pairs of which both words appear at least 100 times in the training set of MS-COCO. Correlations for both WORD GRU and WORD SUM are considerably higher than for PHON GRU. This is expected given that these are word level models with explicit word-embeddings, while PHON GRU builds word representations by forwarding phoneme-level input through several layers of processing.

	All words	Frequent words
PHON GRU Layer 1	0.09	0.12
Layer 2	0.21	0.33
Layer 3	0.28	0.45
WORD GRU	0.48	0.60
WORD SUM	0.42	0.56

Table 3: Spearman's correlation coefficient between word-word cosine similarity and human similarity judgements. All correlations significant at $p < 1e{-}4$. Frequent words appear at least 100 times in the training data.

Layer	ρ
1	-0.30
2	-0.24
3	-0.15

Table 4: Spearman's rank correlation coefficient between PHON GRU cosine similarity and phoneme-level edit distance. All correlations significant at $p < 1e{-}15$.

Table 4 shows Spearman's rank correlation coefficient between the edit distance and the cosine similarity of activation vectors at the hidden layers of PHON GRU. As expected, edit distance and cosine similarity of the activation vectors are negatively correlated: words which are more similar in form are also more similar according to the model.[4]

The negative correlation between edit distances and cosine similarities is strongest at the lowest hidden layer and weakest, though still present and stronger than for human judgements, at the third hidden layer.

The correlations of cosine similarities with edit distance on the one hand, and human similarity rating on the other hand, indicate that the different hidden layers reflect increasing levels of representation: whereas at the lowest level mostly encodes information about form, the highest layer mostly encodes semantic information.

4.4 Position of shared substrings

Here we quantify the time-scale at which information is retained in the different layers of PHON GRU. We looked at the location of phoneme strings shared by sentences and their nearest neighbors in the 5,000-image validation sample. We determined each sentence's nearest neighbor for each hidden layer in PHON GRU. The nearest neighbour is the sentence for which the activation vector at the end of sentence symbol has the smallest cosine distance to the activation vector of the original sentence. The

[3]The MEN dataset is available at `http://clic.cimec.unitn.it/~elia.bruni/MEN`

[4]Note that in the MEN dataset, meaning and word form are also (weakly) correlated: human similarity judgements and edit distance are correlated at -0.08 ($p < 1e{-}5$).

position of matching substrings is the average position in the original sentence of symbols in substrings that are shared by the neighbor sentences, counted from the end of the sentence. A high mean average substring position thus means that the shared substring(s) appear early in the sentence. This gives an indirect measure of the timescale at which the different layers operate. Table 5 shows an example.

As can be seen in Table 6, the average position of shared substrings in neighbor sentences is closest to the end for the first hidden layer and moves towards the beginning of the sentence for the second and third hidden layer. This indicates a difference between the layers with regards to the timescale they represent. Whereas in the lowest layer only information from the latest timesteps is present, the higher layers retain the input signal over longer timescales.

Layer 1
A metallic bench **on a path in** the **park**
A man riding a bicycle **on a path** in a **park**
Layer 3
A metallic **bench** on a path in the **park**
A stone park **bench** sitting in an empty green **park**

Table 5: An illustrative sentence with its nearest neighbour at layer 1 and layer 3. For readability, sentences are displayed in conventional spelling, and only highlight matching substrings of length $\geq$ 3. In reality we used phonetic transcriptions to compute shared substring positions, and substrings of all lengths.

Layer	Mean position
1	12.1
2	14.9
3	16.8

Table 6: Average position of phonemes in shared substrings between nearest neighbour sentences according to PHON GRU representations at the different layers. Positions are indexed from end of string.

5 Future work

Although our analyses show a clear pattern of short-timescale information in the lower layers and larger dependencies in the higher layers, the third layer still encodes information about the phonetic form: its activation patterns were predictive of word boundaries, and similarities between word pairs at this level were more strongly correlated with edit distance than human similarity judgements are. It would be interesting to investigate exactly what information that is, and to what extent it is analogous to language representation in the mind of human speakers. In humans both word phonological form and word meaning can act as primes, which is somewhat reminiscent of the behavior of our model.

Finally, we would like to take the next step towards grounded learning of language from raw perceptual input, and apply models similar to the one described here to acoustic speech signal coupled with visual input. We expect this to be a challenging but essential endeavor.

References

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *arXiv preprint arXiv:1601.03896*.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research (JAIR)*, 49:1–47.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*.

Grzegorz Chrupała, Akos Kádár, and Afra Alishahi. 2015. Learning language through pictures. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Grzegorz Chrupała. 2013. Text segmentation with character-level text embeddings. In *Proceedings of the ICML workshop on Deep Learning for Audio, Speech and Language Processing*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Deep Learning and Representation Learning Workshop*.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*.

Toni Cunillera, Matti Laine, Estela Càmara, and Antoni Rodríguez-Fornells. 2010. Bridging the gap between speech segmentation and word-to-world mappings: Evidence from an audiovisual statistical learning task. *Journal of Memory and Language*, 63(3):295 – 305.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Afsaneh Fazly, Afra Alishahi, and Suzanne Stevenson. 2010. A probabilistic computational model of cross-situational word learning. *Cognitive Science: A Multidisciplinary Journal*, 34(6):1017–1063.

Michael C. Frank, Noah D. Goodman, and Joshua B. Tenenbaum. 2007. A Bayesian framework for cross-situational word-learning. In *Advances in Neural Information Processing Systems*, volume 20.

Arit Glicksohn and Asher Cohen. 2013. The role of cross-modal associations in statistical learning. *Psychonomic Bulletin & Review*, 20(6):1161–1169.

David Harwath and James Glass. 2015. Deep multimodal semantic embeddings for speech and images. *arXiv:1511.03690*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv:1512.03385*.

Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 190–198.

Naoto Iwahashi. 2003. Language acquisition through a human–robot interface by combining speech, visual, and behavioral information. *Information Sciences*, 156(1):109–121.

Ákos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2016. Representation of linguistic form and function in recurrent neural networks. *CoRR*, abs/1602.08952.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3276–3284.

Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skip-gram model. In *Proceedings of NAACL HLT 2015 (2015 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies)*.

Angeliki Lazaridou, Grzegorz Chrupała, Raquel Fernández, and Marco Baroni. 2016. Multimodal semantic learning from child-directed input. In *The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014*, pages 740–755. Springer.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of EMNLP*.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of ACL 2016*. Association for Computational Linguistics (ACL).

Okko Räsänen and Heikki Rasilo. 2015. A joint model of word segmentation and meaning acquisition through cross-situational learning. *Psychological review*, 122(4):792.

Deb K Roy and Alex P Pentland. 2002. Learning words from sights and sounds: a computational model. *Cognitive Science*, 26(1):113 – 146.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2014. ImageNet Large Scale Visual Recognition Challenge.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Jeffrey M. Siskind. 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 61(1-2):39–91.

Gabriel Synnaeve, Maarten Versteegh, and Emmanuel Dupoux. 2014. Learning words from images and speech. In *NIPS Workshop on Learning Semantics, Montreal, Canada*.

Erik D Thiessen. 2010. Effects of visual information on adults' and infants' auditory statistical learning. *Cognitive Science*, 34(6):1093–1106.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *ICLR*.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

Daniel Yurovsky, Chen Yu, and Linda B Smith. 2012. Statistical speech segmentation and word learning in parallel: scaffolding from child-directed speech. *Frontiers in Psychology*, 3(374).

Linguistic features for Hindi light verb construction identification

Ashwini Vaidya
IIT Delhi
ird11278@ee.iitd.ac.in

Sumeet Agarwal
IIT Delhi
sumeet@iitd.ac.in

Martha Palmer
University of Colorado, Boulder
martha.palmer@colorado.edu

Abstract

Light verb constructions (LVC) in Hindi are highly productive. If we can distinguish a case such as *nirnay lenaa* 'decision take; decide' from an ordinary verb-argument combination *kaagaz lenaa* 'paper take; take (a) paper', it has been shown to aid NLP applications such as parsing (Begum et al., 2011) and machine translation (Pal et al., 2011). In this paper, we propose an LVC identification system using language specific features for Hindi which shows an improvement over previous work (Begum et al., 2011). To build our system, we carry out a linguistic analysis of Hindi LVCs using Hindi Treebank annotations and propose two new features that are aimed at capturing the diversity of Hindi LVCs in the corpus. We find that our model performs robustly across a diverse range of LVCs and our results underscore the importance of semantic features, which is in keeping with the findings for English. Our error analysis also demonstrates that our classifier can be used to further refine LVC annotations in the Hindi Treebank and make them more consistent across the board.

1 Introduction

Light verb constructions (LVC) are found across languages e.g. Japanese, Korean, Persian as well as English. An LVC consists of a predicating element (usually a noun) and a verb, which is also known as a *light* verb. For instance, *take a walk* or *give a sigh* are LVCs consisting of light verbs *take* and *give* and their corresponding predicating nouns *walk* and *sigh*. The nouns in an LVC contribute to the event semantics and the light verb supplies additional meaning e.g. agentivity, completeness, or permission. In Hindi, LVCs are productive and are also sometimes termed as 'support verb' or 'conjunct verb' constructions. Examples 1 and 2 contrast the use of a simple predicate *de* 'give' with its light verb usage.

(1) Simple predicate

raam=ne mohan=ko kit̪ab d̪-ii
Ram.M.Sg=Erg Mohan.M.Sg=Dat book.F.Sg give-Perf.F.Sg

'Ram gave Mohan a book'

(2) Noun-Verb complex predicate

raam=ne us baat̪=par **zor** **d̪i-yaa**
Ram.M.Sg=Erg that topic=loc pressure.M.Sg give-Perf.M.Sg

'Ram put an emphasis on that topic'

LVCs form a large part of the lexicon in Hindi. In the Hindi treebank (Palmer et al., 2009) (400,000 words), there are nearly 47,163 predicates, of which 37% have been annotated as LVCs. LVCs consist of predicate types that are far more numerous than simple verbs. Hindi has approximately 700 simple verbs, but potentially many more unique LVCs. This makes them a highly productive phenomenon in Hindi.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1320–1329, Osaka, Japan, December 11-17 2016.

Butt (2010) notes that light verbs in Hindi LVCs act as a verbalizers in order to create new predicates and incorporate borrowed items into the language e.g. *email kar* 'email do; email'. Therefore, LVCs are sometimes described as "a preferred way of augmenting the creative potential of the language" (Kachru, 2006, pp 93).

The identification of LVCs in Hindi (as well as other South Asian languages) is an important NLP task, which has been shown to improve parsing accuracy (Begum et al., 2011) as well as machine translation performance (Pal et al., 2011). The detection of multi-words such as LVCs has been widely studied and association measures, linguistic knowledge and parallel corpora have been used.

As LVCs are a type of multiword, a commonly used method is 'N-gram classification' (Green et al., 2013). This strategy extracts n-grams from the corpus, filters them and assigns some values based on a bigram measures such as log-likelihood or mutual information. A classifier is then used to make a LVC/non-LVC decision. However, previous work has shown that LVCs benefit from the use of linguistic features for identification. Vincze et al. (2011) used bigram association measures for English noun-noun compounds and LVCs. They found that LVC detection improves when linguistic features are used in addition to n-gram information. Tu and Roth (2011) showed that linguistic and statistical features perform at par for English LVC detection.

For Hindi, we expect that linguistic features will be useful for automatic detection. At the same time, the productivity and range of LVC constructions in Hindi result in some specific challenges. In the next section, we carry out a linguistic analysis of LVCs based on the annotations in the Hindi Treebank. We use these insights to propose two new features to identify LVCs. Following this, we describe our experimental setup and then discuss the results.

2 Linguistic challenges for Hindi

The linguistic notion of an LVC differs across languages. While annotating an English corpus with LVC information, Tu and Roth (2011) make use of a 'replacing' principle for their annotators, where if a candidate light verb like *take* in *take a walk* can be replaced by *walk* without (too much) of a change in meaning, then a combination like *take a walk* is considered an LVC.

In Hindi, such a 'replacing' principle is not available as the nouns that participate in LVCs are not necessarily deverbal in nature i.e. the majority do not have a direct verbal counterpart. In fact, LVCs are a preferred method of introducing new predicates into the language via borrowed nouns. Bhattacharyya et al. (2007) have described a number of diagnostic criteria for Hindi LVCs, but these are not completely robust–and can only be applied to LVCs that are transitive. In fact, most linguistic diagnostics mentioned in Mohanan (1994) and Bhattacharyya et al. (2007) are appropriate for transitive LVCs that occur with light verb *kar*. Such cases are the most frequently occurring LVCs in Hindi, but do not represent all LVCs.

Consequently, the application of diagnostic tests for LVCs for a large corpus can be challenging. The Hindi Treebank (Palmer et al., 2009) is a relatively large resource that is annotated with LVC information using the `pof` label. We use this data to examine the behaviour of Hindi LVCs, focusing on each component: the light verb and the predicating nominal.

2.1 Light verbs

Jespersen (1965) coined the term 'light' verb to refer to verbs that do not behave like standard verbal predicates as they have a depleted semantic contribution to the event described by the LVC. These light verbs tend to be similar cross-linguistically e.g. *take*, *make* and *give* can be found in English, Persian and Hindi. At the same time, these verbs are distributed differently across languages.

In the Hindi Treebank, the distribution of light verbs reflects some interesting facts about LVC formation in Hindi. Figure 1 shows the 20 most frequently occurring light verbs in the Training and Development sections of the Hindi Treebank (approx. 21,000 sentences). These light verbs include the following: *kar* 'do', *ho* 'be/happen', *de* 'give', *hE* 'be', *raha* 'stay', *aa* 'come', *karaa/karvaa* 'cause to do', *lagaa* 'touch/feel', *jataa* 'convey', *le* 'take', *banaa* 'make', *rakh* 'keep', *chal* 'go', *uthaa* 'rise', *daala* 'put', *laDa* 'fight', *lag* 'seem', *ban* 'become', *maar* 'hit'. Each of these light verbs also appear as 'full' verbs

i.e. they can also appear without a nominal predicate, as a non-LVC.

The bar plot in Figure 1 shows that the frequency of *kar* 'do' is the greatest, followed by *ho*, 'be' and *de* 'give'. The light verb *kar* 'do' has many more positive cases of LVCs as compared to non-LVCs.For other light verbs such as *de*, the distribution is more even and with other light verbs, there are far more non-light usages of these verbs as compared to light.

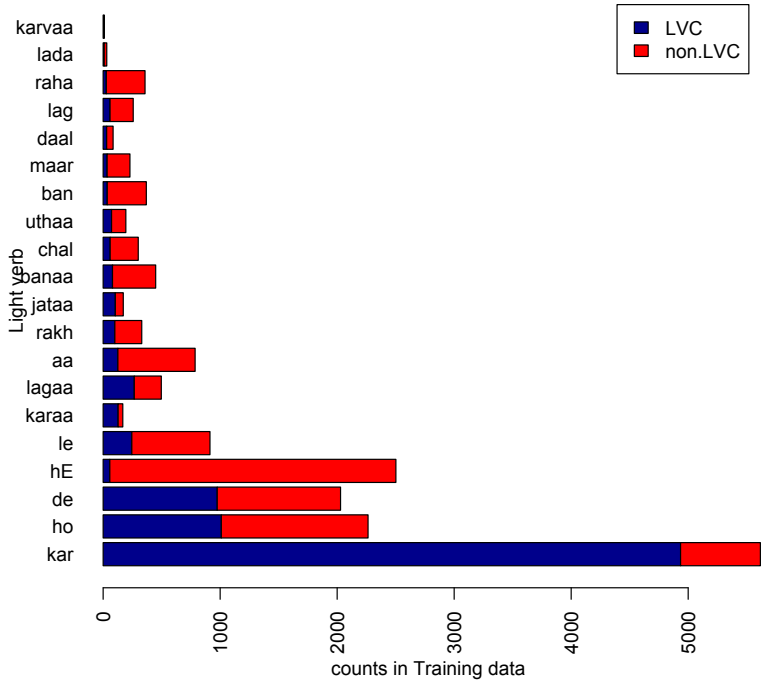


Figure 1: Light verb distribution in the Training and Development section of the Treebank

If we were to divide the training data by light verb and use the majority class to predict the light or non-light case, we would still get reasonably good results. This is because the light *kar* cases far outnumber the others, and one can expect the majority class baseline to be as high as 0.8. Conversely, a light verb like *aa* 'come' has more non-light usages than light, hence the majority class prediction would also be quite high.

Begum et al. (2011) describe a classifier for Hindi LVCs but do not mention the distribution of LVCs in the data. Therefore, it is difficult to know whether the results are applicable to all LVCs or just the light verb 'kar'. In contrast, Butt et al. (2012) focus on light verbs *kar* 'do' and *ho* 'be' alone. In this paper, we make use of the Hindi Treebank LVC annotations to evaluate our 'combined' system, but provide evaluation across individual light verbs in the corpus. This also implies that we must incorporate features that are specific not only to 'kar', but across all light verbs in the data.

2.2 Nominal predicates

The Hindi Treebank consists of more than 3000 unique nominals that can occur as part of an LVC. At the same time, some of these nouns can combine with more than one light verb to form an LVC e.g. *ishaara kar* 'signal do; make a sign' and *ishaara de* 'signal give; give a sign', with some subtle differences in meaning. It is possible that one of these combinations is more frequent than the other- or they may be equiprobable. There are also certain nouns where only one light verb is possible e.g. *maut ho* 'death be; die'.

We carried out a corpus study, examining 1853 unique nouns from the Training and Development sections of the Hindi Treebank and extracted the number of light verbs that occurred with them. Although,

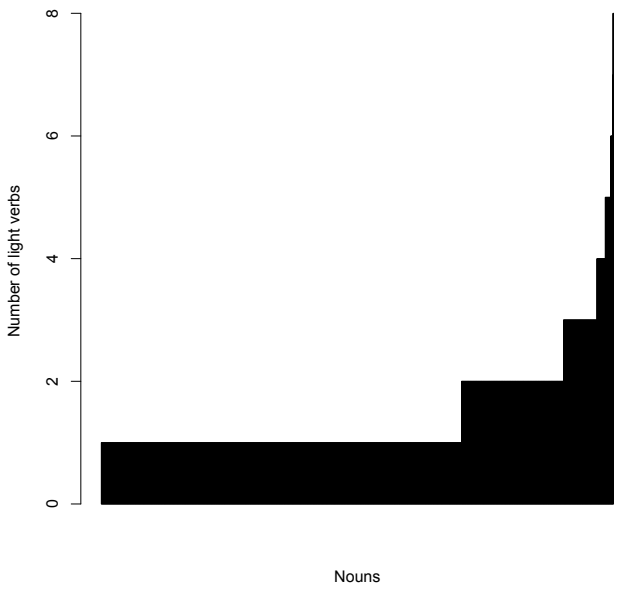


Figure 2: Number of light verbs that occur with a unique noun. (t=1853)

Figure 2 shows a 'long tail', where a large number of nouns occur with just one light verb, about 1/4th of the data consists of nouns that alternate with more than one light verb.

These alternations show that a collocational measure that only looks at the bigram occurrences may not be able to capture a noun-light verb alternation that is relatively infrequent. Therefore, linguistic information would be required to identify a predicating nominal that appears in a number of contexts. In the following section, we propose new features that could help capture this information.

3 Linguistic features used for LVCs

English LVC identification focuses on extracting linguistic features for LVCs (Tan et al., 2006; Grefenstette and Teufel, 1995; Stevenson et al., 2004). For example, the morpho-syntactic similarity between nominal predicates and their verbal counterparts (e.g. *walk* and *take a walk*) is often exploited. Other cues include the presence of indefinite determiners (such as *a*) before the nominal predicate.

Tu and Roth (2011) look at both statistical and linguistic contexts to detect English complex predicates. Among their local linguistic features, they utilize bigram information about the nominal head and light verb, the nouns themselves and the Levin verb class members of deverbal nouns. In a more recent study, Chen et al. (2015) have described an improvement over Tu and Roth (2011)'s performance by using lexical features from WordNet, as well as word sense information. Using the Tu and Roth (2011) testset, they report a 0.89 F-score for English LVCs.

Author/Feature	**Tan et al'06** (Eng)	**Tu and Roth'11** (Eng)	**Begum et al'11** (Hin)
Deverbal noun	Y	Y	
Noun semantics	Y	Y	Y
Light verb list	Y	Y	Y
Presence post-posn			Y
Presence determiner	Y	Y	
Collocational measure		Y	Y

Table 1: Commonly used linguistic features for English and Hindi LVC detection.

The work for Hindi LVC detection makes use of a similar set of linguistic features. Begum et al. (2011) look for the presence of postpositions and demonstratives, which preferentially do not occur with a noun that is a part of an LVC. Like Tu and Roth (2011), they use the verb-object bigram and the noun class information from Hindi WordNet (Narayan et al., 2002). They have achieved an accuracy of around 0.85 for identification of Hindi complex predicates. More recently, Singh et al. (2015) have compared word embeddings and WordNet-based measures to detect Hindi noun compounds and LVCs. While word embeddings are effective for compounds, they perform poorly for LVCs, suggesting the importance of more precise linguistic features.

3.1 Linguistic features for Hindi

Our linguistic analysis of LVCs indicates that the properties of both noun and light verb are crucial as features for identifying LVCs. In the previous section, we described some of the commonly used features for LVC identification. Table 1 shows some of these: the presence of post-positions after the predicating noun, collocational features and lexical features.

In this section, we introduce two new features that are based on our study of Hindi LVCs. The first is based on the idea that there are semantic constraints on the combination of a particular noun and light verb. Sulger and Vaidya (2014) examined the combinatorial properties of noun and light verb based on relative frequency of occurrence. They found that a light verb such as *de* 'give' is likelier to combine with nouns that have a 'transfer' property, whereas nouns that occur with *kar* 'do' will occur with nouns that describe actions with animate agents. Light verb *ho* 'be' often appears with stative nouns or those that indicate mental states.

In order to capture these properties, we used a feature that associated a light verb with the ontological property of the noun that is likely to occur with it. For example *kar* was associated with *Physical_Action_Abstract_Inanimate* and *de* 'give' with *Communication_Action_Abstract_Inanimate*. These ontological properties were extracted from Hindi WordNet (Bhattacharyya, 2010). If a noun occurred with the ontological property that was associated with a particular light verb, it was marked positively for this feature.

The second feature was based on the idea that predicating nominals usually introduce arguments of their own. These usually occur with the postpositions *par* 'on', *se* 'with', *ko* 'to' or *kii* 'of'. These indicate that a nominal has introduced an argument of its own–and is likely to be a predicating nominal rather than an ordinary argument of the verb. Examples 3-5 illustrate the cases where a nominal introduced an argument with *par*, *se* or *kii*.

(3) pulis=ne **logon=par** hamlaa ki-yaa
police=Erg people=loc attack.M.Sg do-Perf.M.Sg
'(The) Police attacked the people'

(4) samir=ne **mohan=se** nafrat k-ii
samir.M.Sg=Erg Mohan.M.Sg=instr hatred.F do-perf.F.Sg
'Samir hated Mohan'

(5) samir=ne **ghadii=kii** chorii k-ii
Samir.M.Sg-Erg watch.F.Sg-gen theft.F do-Perf.F
'Samir stole the watch'

This feature was introduced to overcome some of the shortcomings of the 'presence of post-position' feature on the noun (Table 1). The post-position only looks at the presence or absence of post-positions on the predicating noun, whereas the proposed feature looks at the postpositions on the nominal's *arguments*. The former feature is restricted to agentive nominals, whereas this feature is applicable to all predicating nominals that license arguments.

4 Experimental setup

We make use of the Hindi Treebank data to train our LVC identification system. The Hindi Treebank annotation guidelines describe the use of the label `pof` to identify Hindi LVCs. They make use of

	Train	**Development**	**Test** (Treebank)	**Test** (ICON)
News	14282	3500	1708	2757
LVC	6739	1665	790	1056
non-LVC	7543	1835	918	1701

Table 2: Training, Development and Test instances, with the number of light and non-light verbs

annotators' linguistic intuition to identify **pof** cases with a "full understanding that it may lead to some inconsistency in the data" (Bharati et al., 2012, p41). This is probably because of the lack of reliable linguistic diagnostics mentioned earlier in section 2. As a result, we can think of the Hindi Treebank annotation of LVCs as reflecting a fairly generous conceptualization of LVCs.

In order to reduce any errors and inconsistencies, in this work we take into consideration only the top 20 most frequently occurring light verbs in the corpus[1]. These account for 90% of the LVCs in the Treebank. The remaining light verbs occur only less than 10 times in the corpus and we assume that low frequency might indicate an annotation error. Although this may leave out some valid cases of LVCs, we make the assumption that LVCs represented by these 20 light verbs give us a fairly good representation of the LVC construction. Begum et al. (2011) also make use of the 20 most frequently occurring light verbs in their model. However, they do not provide a list of these light verbs in their paper. Although we may not be able to make a very exact comparison with their model, we would imagine that the differences will be minor, with respect to the low frequency light verbs.

In order to select candidates, we identified positive and negative instances of LVCs in the Hindi parse trees. In the Hindi dependency parse tree, the predicative noun is a dependent of the light verb and in the majority of the cases, both noun and light verb occur next to each other in the sentence. The noun and light verb can be scrambled away from each other, but we found this to be fairly rare in the Treebank LVC examples. Therefore, we chose candidates based on proximity; e.g. if a phrase annotated as an NP occurred next to a verb phrase containing a light verb, this was taken to be a candidate for LVC identification. Apart from NP phrases, we also accepted phrases annotated as 'BLK', which indicated that the noun was borrowed from English. Such nouns often occur as part of LVCs, as complex predication is used to introduce new words into the language.

Our training data made use of the splits provided by the Hindi Treebank, to which we added a small sub-part of the Treebank consisting of conversational data, taken from fiction. The rest of the Hindi Treebank is news text. We made use of the training and test splits given by the Treebank, but kept a small portion of the training set as a development set. Table 2 shows the division between the training, development and test sets and the distribution of positive and negative classes. Across the board, we find that the number of non-LVCs is higher than the LVC instances.

The two test sets are drawn from different genres. The Treebank test set is from the testing split provided by the Hindi Treebank and is news text. The second test set consists of sentences taken from literary criticism. This data is not from the Treebank, but taken from the ICON 2009 Shared task for Hindi parsing (Husain, 2009) and we will refer to this test set as 'ICON'. We included this test set to compare the performance of our model with Begum et al. (2011).

4.1 Features

The features used for identification of LVCs can be grouped into roughly four categories viz. lexical, morphosyntactic, collocational and semantic. We used features that are similar to those included in Begum et al. (2011) as well as Tu and Roth (2011), and additionally introduced two new features (section 3.1). Table 3 shows the set of features used for identifying LVC cases in the Treebank. The other features have been used in previous work to identify English or Hindi LVCs. For the collocational features, the values were obtained from a large corpus (Hindi Wikipedia) and then converted to binary features. For log-likelihood, this was done using a table of critical values to decide whether the ratio was significant.

[1]These verbs are listed in section 2.1

Accordingly, it got the binary feature 0 or 1. In the case of PMI, we checked whether its value was greater than or less than 0 for a given noun and verb candidate. If it was greater, then the noun-verb pair was likely to be a better collocation.

Type	No	Feature
Lexical	1	Verb lemma (Baseline)
	2	Noun lemma
Morpho-syntactic	3	Postposition after noun
	4	Arguments of eventive noun (eventive nouns have an 'extra' argument)
Collocational	5	Log-likelihood value
	6	Pointwise Mutual Information value
Semantic	7	Ontological category of noun
	8	Acceptability of noun with a given light verb

Table 3: Features used for LVC detection

4.2 Model

We experimented with two types of models: a linear model (Logistic Regression) and SVM with an RBF kernel. The eight feature types described earlier generated 3278 features, over which we performed feature selection to choose 1638 (roughly half) of the features based on their individual f-scores. The motivation to carry out feature selection was because of the large number of lexical features, some of which may not have been significantly useful for the classifier. We made use of the *fselect.py* tool for feature selection (Chen and Lin, 2006).

We used the Scikit-learn package (Pedregosa et al., 2011) to train our model. Scikit-learn uses the LIBSVM implementation of support vector machines (Chang and Lin, 2011) and the LIBLINEAR implementation for logistic regression (Fan et al., 2008). We made use of 10-fold cross validation to find the best value of C and gamma for the RBF kernel (C=1, gamma=0.05).

5 Evaluation

We trained our two models using the features described in section 4.1 and evaluated them against the two test sets that we described earlier. We used the verb lemma as our baseline feature. Table 4 shows the performance of our system in comparison to the verb lemma baseline and the system described in Begum et al. (2011).

	Logistic Regression			**SVM with RBF**		
ICON	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
LVC	87.77	76.13	81.54	88.42	76.7	82.15
Non-LVC	86.31	93.41	89.72	86.63	93.76	90.06
Accuracy	86.79			**87.23**		
Begum et. al. (2011)	85.28					
Verb lemma Baseline	75.87			75.66		
News	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
LVC	86.36	91.39	88.80	85.8	90.25	87.97
Non-LVC	92.20	87.58	89.83	91.22	87.14	89.13
Accuracy	89.34			88.58		
Verb lemma Baseline	80.97			80.91		

Table 4: Precision, recall and F1 scores for the ICON and News test sets

Both our models perform better than Begum et al. (2011) on the same test set. Additionally, we also evaluated our system on the news test set from the Hindi Treebank. The performance on the news dataset is better, most probably because of the smaller number of unseen nouns in news (180) as compared to ICON (612).

The diversity of the LVCs in Hindi implies that we would like to check the performance of our system across all light verbs. As the light verb *kar* is the most frequently occurring light verb, we expect that it will give us the best results. We carried out two types of experiments for light verbs: first we

ran individual classifiers across light verbs using the same feature set and compared its micro-averaged performance with that of the combined model. We found that this result was almost exactly similar to the combined model. As a second experiment, we also looked at the performance for individual light verbs within the combined model itself.

Table 5 describes the performance for individual light verbs *kar*, *ho*, *de* and *le*. The other light verbs are fairly infrequent, hence they are grouped into *LF-TR* for transitive light verbs and *LF-INTR* for intransitive light verbs. We find that LVCs with *kar* are identified with high accuracy because of their high frequency. The performance is slightly less accurate for other light verbs, notably *ho* 'be'. However, they still perform above the baseline, indicating that the features are robust enough to identify a wide range of LVCs. The *LF-INT* cases have a poor recall because the number of negative examples far outnumber the positive. The baseline accuracy for *LF-INT* also reflects this imbalance. We see a similar performance for individual light verbs in the news test set.

Individual LVs	LVs in test data	Precision	Recall	F1	Baseline	Accuracy
kar 'do'	650	96.54	95.07	95.80	81.23	93.23
ho 'be'	454	72.26	83.49	77.47	54.62	77.97
de 'give'	216	85.36	70.00	76.92	53.7	80.5
le 'take'	110	91.66	52.38	66.66	61.81	80
LF-TR	263	85.71	42.85	57.14	73.0	86.31
LF-INT	1064	83.33	16.12	27.02	88.34	89.84

Table 5: Precision, Recall and F1 for individual light verbs in the ICON test set, using Logistic Regression. The baseline accuracy uses the verb lemma as the feature.

5.1 Discussion

In order to understand the most informative features for our model, we examined the top 25 best performing features for the SVM model. We found that the best features for the positive class included the light verb lemma *kar* and a high-frequency noun lemma *shuru* 'begin'. Both log-likelihood and PMI were highly predictive of the positive class as well as the new feature using nominal argument postpositions. Semantic features such as 'Artifact,Object,Inanimate,Noun' were predictive of the negative class. This shows us that the linguistically motivated features are indeed effective for identification. From the analysis, it also appeared that semantic features overall can be more discriminative than lexical features for Hindi. This result is congruent with the results from English in Chen et al. (2015), who also use WordNet and sense annotated data as features.

We also made use of the probability scores for each class to understand the confidence of the classifier in assigning an instance to a positive or negative class. We found that in general, both classifiers were more confident in predicting the negative class label as their probabilities formed a distribution that was grouped closer to 1. The scores given to the positive class on the other hand were more distributed, with several instances that were less than 0.5. When we examined the LVCs with lower confidence scores, we saw that this was a mixed bag. For example, there were some LVCs like *bhojan kar* 'meal do; eat', which appeared to be non-LVCs. Others such as *photo le* 'photo take; take a photo' were perhaps cases of noun incorporation as suggested in Davison (2005). Still others were cases like *bojh daal* 'weight put; to be a burden (on someone)', which were more idiomatic in nature. This result shows that perhaps some of these cases are simply less frequent, but also that LVC annotation in the Hindi Treebank itself could be re-considered, or made more fine-grained based on the confidence scores of these models.

Our experiments show that LVCs in Hindi consist of diverse types that can be identified automatically using linguistic features. Unlike English, where the deverbal noun can be used as an important lexical indicator of 'lightness', in Hindi it becomes necessary to make use of other morpho-syntactic cues such as postpositions. However, it appears that the role of semantic features in general seems to be important for light verb identification in Hindi as well as English.

The models we have described in this paper show an improvement over previous work, but at the same time they can also be used to further refine the LVC annotation in the Hindi Treebank. This would give us more clarity with respect to the linguistic behaviour of these cases in Hindi and serve as a guideline

for LVC annotation in the future.

Acknowledgements

We gratefully acknowledge the support of the DST-CSRI (Department of Science and Technology (Govt. of India), Cognitive Science Research Initiative) fellowship funding titled 'Processing of complex event structures in Hindi: an investigation into relative compositionality' for the first author, as well as funding from CLEAR at the University of Colorado. This work was also supported by NSF grants CNS-0751202 and CNS-0709167. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Rafiya Begum, Karan Jindal, Ashish Jain, Samar Husain, and Dipti Misra Sharma. 2011. Identification of Conjunct Verbs in Hindi and their effect on Parsing Accuracy. In *In Proceedings of the 12th CICLing, Tokyo, Japan*.

Akshara Bharati, Dipti Misra Sharma, Samar Husain, Lakshmi Bai, Rafiya Begam, and Rajeev Sangal. 2012. AnnCorra : TreeBanks for Indian Languages. Technical Report Version-2.5, IIIT Hyderabad.

Pushpak Bhattacharyya, Debasri Chakrabarti, and Vaijayanthi Sarma. 2007. Complex Predicates in Indian languages and Wordnets. *Language Resources and Evaluation*, 40(3-4):331–355.

Pushpak Bhattacharyya. 2010. IndoWordNet. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10)*, pages 3785–3792.

Miriam Butt, Tina Bögel, Annette Hautli, Sebastian Sulger, and Tafseer Ahmed. 2012. Identifying Urdu Complex Predication via Bigram Extraction. In *Proceedings of COLING 2012: Technical papers*, pages 409–424.

Miriam Butt. 2010. The Light Verb Jungle: Still Hacking Away. In M. Amberber, M. Harvey, and B. Baker, editors, *Complex Predicates in Cross-Linguistic Perspective*, pages 48–78. Cambridge University Press.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27.

Yi-Wei Chen and Chih-Jen Lin, 2006. *Combining SVMs with Various Feature Selection Strategies*, pages 315–324. Springer Berlin Heidelberg, Berlin, Heidelberg.

Wei-Te Chen, Claire Bonial, and Martha Palmer. 2015. English Light Verb Construction Identification Using Lexical Knowledge. In *Proceedings of the AAAI-15, Austin, TX, USA*.

Alice Davison. 2005. Phrasal predicates: How N combines with V in Hindi/Urdu. In Tanmoy Bhattacharya, editor, *Yearbook of South Asian Languages and Linguistics*, pages 83–116. Mouton de Gruyter.

R.-E. Fan, K.-W. Chang, C.-J. Hseih, X.-R. Wang, and C.-J. Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Spence Green, Marie-Catherine de Marneffe, and Christopher D Manning. 2013. Parsing Models for Identifying Multiword Expressions. *Computational Linguistics*.

Gregory Grefenstette and Simone Teufel. 1995. Corpus-based method for automatic identification of support verbs for nominalization. In *Proceedings of the 7th Meeting of the European Chapter of the Association for Computational Linguistics (EACL'95)*.

Samar Husain. 2009. Dependency Parsers for Indian languages. In *Proceedings of the ICON 2009 Tools Contest: Indian Language Dependency Parsing*.

Otto Jespersen. 1965. *A Modern English Grammar on Historical Principles, Part VI, Morphology*. George Allen and Unwin Ltd.

Yamuna Kachru. 2006. *Hindi*. John Benjamins.

Tara Mohanan. 1994. *Argument Structure in Hindi*. CSLI Publications, Stanford.

Dipak Narayan, Debasri Chakrabarti, Prabhakar Pande, and Pushpak Bhattacharyya. 2002. Experiences in Building the Indo WordNet- A WordNet for Hindi. In *First International Conference on Global WordNet, Mysore, India*.

Santanu Pal, Tanmoy Chakraborty, and Sivaji Bandopadhyay. 2011. Handling Multi-word expressions in Phrase-based Statisical Machine Translation. In *Proceedings of the Workshop on Multiword Expressions (MWE 2011), 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011)*.

Martha Palmer, Rajesh Bhatt, Bhuvana Narasimhan, Owen Rambow, Dipti Misra Sharma, and Fei Xia. 2009. Hindi Syntax: Annotating Dependency, Lexical Predicate-Argument Structure, and Phrase Structure. In *Proceedings of ICON-2009: 7th International Conference on Natural Language Processing*, Hyderabad.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Dhirendra Singh, Sudha Bhingardive, Kevin Patel, and Pushpak Bhattacharyya. 2015. Detection of Multiword Expressions for Hindi Language using Word Embeddings and WordNet-based Features. In *Proceedings of ICON-2015*.

Suzanne Stevenson, Afsaneh Fazly, and Ryan North. 2004. Statistical measures of the semi-productivity of light verb constructions. In *Proceedings of the 2nd ACL Workshop on Multiword Expressions: Integrating Processing*.

Sebastian Sulger and Ashwini Vaidya. 2014. Towards Identifying Hindi/Urdu Noun Templates in Support of a Large-Scale lfg Grammar. In *Proceedings of the Fifth Workshop on South and Southeast Asian Natural Language Processing at COLING 2014*.

Yee Fan Tan, Min-Yen Kan, and Hang Cui. 2006. Extending corpus based identification of light verb constructions using a supervised learning framework. In *Proceedings of the EACL 2006 Workshop on Multi-word-expressions in a multilingual context*.

Yuancheng Tu and Dan Roth. 2011. Learning English Light Verb Constructions: Contextual or Statistical. In *Proceedings of the Workshop on Multiword Expressions (MWE 2011), 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011)*.

Veronika Vincze, István Nagy T, and Gabór Berend. 2011. Detecting noun compounds and light verb constructions: a contrastive study. In *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World (MWE 2011)*.

Cross-lingual Transfer of Correlations between Parts of Speech and Gaze Features

Maria Barrett
Centre for Language Technology
University of Copenhagen
Njalsgade 136
2300 Copenhagen S, Denmark
barrett@hum.ku.dk

Frank Keller
School of Informatics
University of Edinburgh
10 Crichton Street
Edinburgh EH8 9AB, UK
keller@inf.ed.ac.uk

Anders Søgaard
Dpt. of Computer Science
University of Copenhagen
Sigurdsgade 41
2200 Copenhagen N, Denmark
soegaard@di.ku.dk

Abstract

Several recent studies have shown that eye movements during reading provide information about grammatical and syntactic processing, which can assist the induction of NLP models. All these studies have been limited to English, however. This study shows that gaze and part of speech (PoS) correlations largely transfer across English and French. This means that we can replicate previous studies on gaze-based PoS tagging for French, but also that we can use English gaze data to assist the induction of French NLP models.

1 Introduction

The eye movements during normal, skilled reading are known to reflect the processing load associated with reading. Recently, eye movement data has been integrated into natural language processing models for weakly supervised part-of-speech (PoS) induction (Barrett et al., 2016), sentence compression (Klerke et al., 2016), supervised PoS tagging (Barrett and Søgaard, 2015a), and supervised parsing (Barrett and Søgaard, 2015b).

Barrett et al. (2016) used eye movements from the English portion of a large eye tracking corpus, the Dundee corpus (Kennedy et al., 2003), for weakly supervised PoS induction for English, obtaining significant improvements over a baseline without gaze features. They used a second-order hidden Markov Model, which was type-constrained by Wiktionary dictionaries for their experiments. These results suggest an approach to weakly supervised PoS induction using only a dictionary and eye movement data. Such an approach would be applicable for low-resource languages, for which it is difficult to find professional annotators.

The present study further explores to which extent native readers' processing of PoS generalizes across related languages. We use a similar model as Barrett et al. (2016), but perform cross-lingual experiments with both the French and the English portion of the Dundee Corpus.

Contribution This is to the best of our knowledge the first study to explore how the eye movements of native readers that inform PoS models generalize from one language to another. We also introduce a new resource for studying the relation between grammatical class and eye movements in French: we provide PoS annotation for most of the French Dundee Corpus by aligning it with the morphosyntactic annotation of the French Treebank (Abeillé et al., 2003).

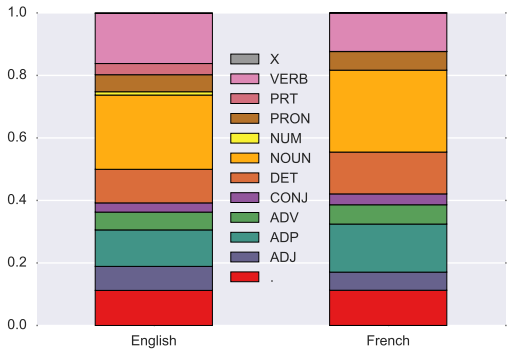
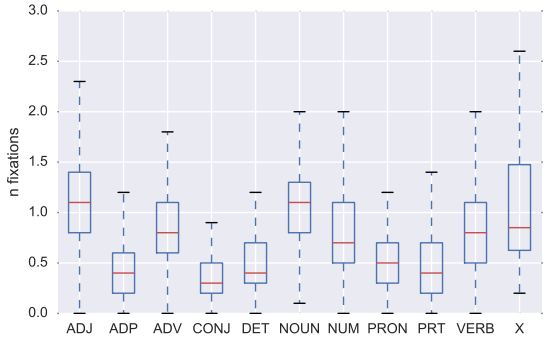
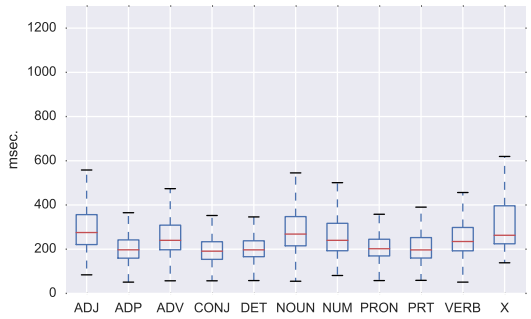


Figure 1: Distribution of PoS in the English and French training sets.

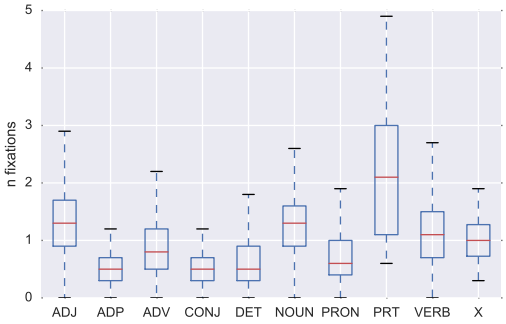
Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1330–1339, Osaka, Japan, December 11-17 2016.



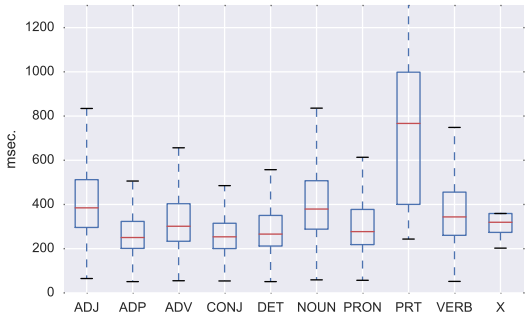
(a) English number of fixations



(b) English First pass duration



(c) French number of fixations



(d) French first pass duration

Figure 2: Two reading measures across PoS class computed on the English and French training sets.

2 Data preparation

The data used for this experiment is the English and French portions of the Dundee Corpus (Kennedy et al., 2003). The Dundee Corpus is the largest available eye movement corpus by token count. For English and French, 10 native speakers of each language read 20 newspaper articles from either *The Independent* (English) or *Le Monde* (French). The corpus comprises around 50,000 tokens per language.

For both the English and the French part of the Dundee Corpus, the original tokenization follows the visual units of the text, and contractions and punctuation are attached to the word whose visual unit they belong to. For instance, *s'entendre* or *rappelle-t-il* are one token in the French Dundee Corpus but two and five, respectively, in the French Treebank. In the English Dundee Corpus, *don't!* is one token, but three in the Dundee Treebank. As a result, eye movement measures are only available for the entire visual unit. We address this issue by duplicating the eye movement measures for all treebank tokens that comprise a Dundee token (i.e., a visual unit). This is the same approach Barrett et al. (2016) used. As a result, the number of tokens increases in the PoS-tagged version of the Dundee Corpus; also, some tokens are associated with eye movement measures that reflect the processing of several tokens.

For English, the treebank tokenization leads to 13.8% increase of tokens to 58,599 tokens. For French, the treebank tokenization leads to an 17.7% increase on token count to 56,683 tokens. For the English training set, 76% of all Dundee Corpus tokens are mapped to one treebank token. The same goes for 62% of the Dundee Corpus tokens for French.

2.1 English

The Dundee Treebank (Barrett et al., 2015) is a recent manual, syntactic annotation layer for the English portion of the Dundee Corpus following the Universal Dependency formalism. For evaluation, we use the PoS labels from this resource. We mapped the Penn Treebank tagset used in the Dundee Treebank

automatically to the Universal PoS tag set (Petrov et al., 2011).

The split into training, development, and test set for the English Dundee corpus is identical to the splits used by Barrett et al. (2016), with 80% of the tokens for training and 10% of the tokens for development and testing, respectively, without splitting up sentences. This split results in 46,879 tokens in 1,896 sentences for training, 5,868 tokens in 230 sentences for development, and a test set of 5,832 tokens in 241 sentences.

2.2 French

The text for the French part of the Dundee Corpus is originally from the French Treebank version 1.4 (Abeillé et al., 2003) and we re-aligned the two corpora for this experiment. We first manually identified the relevant subset of the French Treebank (which is discontinuous). A small part (2,518 tokens equivalent of 5.31% of the French Dundee tokens) of the Dundee Corpus could not be found by manual search in the French Treebank and was therefore omitted from the experiment. Only entire sentences were removed. The morphosyntactic annotation of the French Treebank was semi-manually aligned with the Dundee Corpus by a set of heuristic rules and by manually fixing all exceptions. Due to tokenization inconsistencies in both the French Treebank and the Dundee Corpus, manual intervention was required.

For French there are some treebank tokens with no token string, only PoS, lemma etc. For example, *du* should be split into *de* and *le*, but in some instances the token string for *le* is missing. These missing tokens were omitted from this experiment.

The French Dundee Corpus does not come with a training-development-test split. We use a similar approach as for English, with the first 80% of the tokens for training, the next 10% of the tokens for development and the last 10% for testing. No sentences were split into separate sets. That results in 43,383 tokens in 1,585 sentences for training, 5,407 tokens in 240 sentences for development, and 5,444 tokens in 178 sentences for testing.

The tagset of the French Treebank was automatically mapped to the Universal PoS tag set (Petrov et al., 2011). We make the aligned, morphosyntactic annotation for the French Dundee Corpus available at `https://bitbucket.org/lowlands/release`.

2.3 Reading differences between English and French

This section discusses the results of existing studies comparing reading in French and English. The two main studies used the two Dundee corpora for their analysis.

Pynte and Kennedy (2006) compared the eye movements of the French and English Dundee corpus to explore local effects (e.g., word frequency, word length, local context) and global effects (e.g., predictability, reading strategy, inspection strategy) on five eye movement metrics.

They first of all noted that French was read slower than English with more and longer fixations. This effect is significant and is even more pronounced for long words and there are also significantly more re-fixations for French compared to English. Kennedy and Pynte (2005) argue that re-fixations reflect the most crucial difference between French and English. Besides being an obvious difference in the processing of the target word, more re-fixations also enhance preview of the next word. Pynte and Kennedy (2006) report that participants of the English and French experiments were matched (though not on which factors) and that the procedure, including calibration technique, equipment, control software, instructions, and data-reduction software, were identical across language, though the French data was collected in Aix-en-Provence, France and the English data in Dundee, UK. Therefore they ascribed this difference to the text itself. Even though they found that French words (5.2 characters) are on average longer than English ones (4.7 characters), there are more two-letter words in French (19.7%) than in English (17.2%). Therefore Kennedy and Pynte (2005) suggest that the reading difference is due the distribution of information across the letters of a given words, which is different across these two languages. For example, in French, terminal accents, case markers, and gender and tense marking convey crucial morphological information. This is in line with their finding that eye movements in the English part of the Dundee Corpus were more sensitive to the length of the next word, whereas French showed equivalent effects of the informativeness of the word-initial trigram.

	− suffix feats			+ suffix feats		
TR-TE	**No gaze**	**Token**	**Type**	**No gaze**	**Token**	**Type**
Development set accuracy						
EN-EN	77.44	80.01	**83.38**	80.21	81.46	**83.86**
FR-EN	**73.16**	72.92	72.92			
FR-FR	82.45	83.08	**86.55**	83.39	84.11	**87.52**
EN-FR	79.38	80.86	**80.97**			
Test set accuracy						
EN-EN	76.49	78.49*	**82.14***	80.37	80.60	**83.25***
FR-EN	71.38	71.39	**71.58**			
FR-FR	81.30	82.27*	**85.03***	83.16	83.30*	**86.22***
EN-FR	78.34	79.83*	**79.92***			

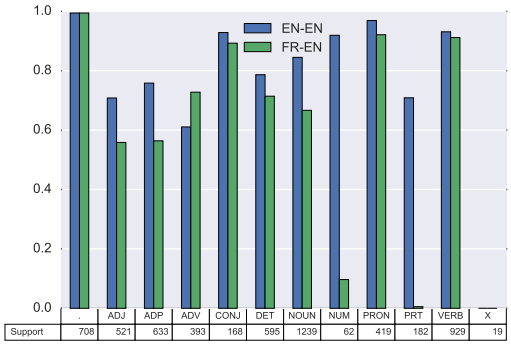
Table 1: Accuracy on development and test set for type-and token-level experiments. Best condition per experimental set-up per language combination in bold. *) For test set results: $p < 0.001$ according to mid-p McNemar test when compared to baseline.

Overall, Pynte and Kennedy (2005; 2006) conclude that the English and French inspection strategies are remarkably similar, which is the same conclusion Sparrow et al. (2003) made when testing the English EZ reader model on another eye movement corpus of 134 words of French. Kennedy and Pynte (2005) provide an analysis of the statistical differences between English and French, but besides re-fixations being more frequent in French, they seem to conclude that the reading is in many respects similar, which is also supported by their choice of mainly analyzing French and English jointly.

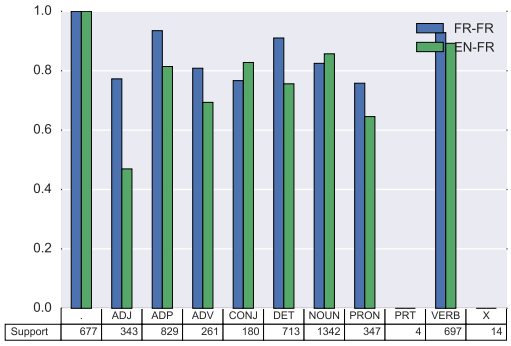
The treebank annotation includes sentence boundaries, which makes it possible to compare the length and the complexity of the sentences for both languages. We find that the average sentence length of the English training set is 24.7 tokens (SD 13.1). For French it is 28.7 tokens (SD 17.8). Sentence length was not considered by Pynte and Kennedy (2005; 2006). A consequence of longer sentences is that reading difficulty increases. The Coleman-Liau index (Coleman and Liau, 1975) is 10.38 for the English training set and 12.98 for the French.[1] This could stem from different writing styles in *Le Monde* and *The Independent* or a biased sampling of articles.

The conclusion can go no further than to say that French and English readers *can* display a more or less similar inspection strategy when reading text under matched conditions. Some effects, e.g., the fact

[1] calculated using `http://www.online-utility.org/`

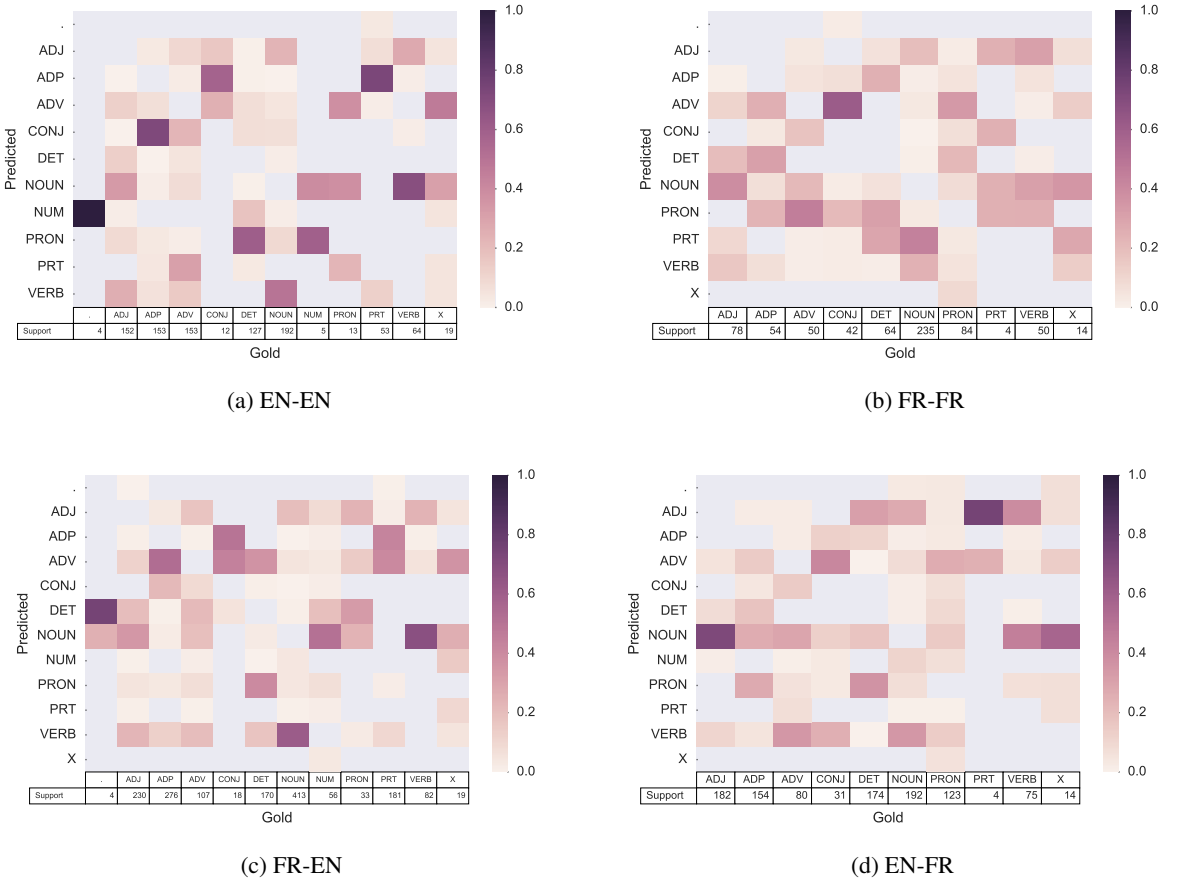


(a) EN-EN and FR-EN



(b) FR-FR and EN-FR

Figure 3: Accuracy on development set for all PoS classes.



(a) EN-EN

(b) FR-FR

(c) FR-EN

(d) EN-FR

Figure 4: Erroneous predictions per gold PoS for all combinations of training and testing language on development set.

that word-initial trigrams are more important for fixation durations in French than in English, could be due to cross-lingual differences in the spelling of the two languages, leading to re-fixations in order to increase preview. But slower reading could also be partly due to the presence of more difficult texts in the French corpus. See Section 7 for a further discussion on grammatical processing differences across languages.

2.3.1 Comparing reading of PoS for English and French

The statistics presented in the following section were computed on the French and English training sets and extends the comparison of Section 2.3 with respect to PoS. We show that the PoS classes are overall read similarly across the two languages with few exceptions due to systematic biases.

Figure 1 shows the distribution of PoS classes in the English and French data. The biggest differences are that there are no NUM tags in French. This is due to the annotation scheme and our automatic mapping, in which no tags map to NUM. There are also very few particles in the French data compared to English.

Figure 2 shows boxplots for two different reading metrics: number of fixations and first pass duration, across PoS class for English and French. The first pass duration is the sum of fixation durations for a token in the first pass through the text. This measure is said to encompass early syntactic and lexical processing. The number of fixations encompasses re-fixations and regressions to a token and reflects later syntactic and semantic processing.

Note that punctuation is almost always glued to a word and any eye movements on a punctuation will mainly—if not solely—reflect the processing of the other token. Therefore punctuation is excluded from Figure 2.

When comparing Figure 2d and 2b, we can confirm Pynte and Kennedy's (2006) finding that fixations

are generally longer in the French portion than in the English portion of Dundee. Average gaze duration in the training set is 236 ms for English and 303 ms for French.

It can be seen from Figure 2 that the measures differ across PoS for most classes in an intuitive way. For instance, PoS classes of short, frequent, closed-class words such as CONJ, ADP, PRON and DET get fewer and shorter fixations than, e.g., NOUN, VERB, ADJ, and ADV. This seems to be consistent across the two languages, and is in line with a similar analysis for English (Barrett and Søgaard, 2015a) for a smaller data set of naturally occurring text from five different domains.

The PRT category seems to be an exception. In French, PRT seems to require extensive early and late processing. Remember from Figure 1 that there are more PRTs for English (3.6%) and fewer for French (0.05%). The sets of PRT words for the two languages reveal a systematic bias in the annotation scheme or automatic mapping. For the French training set, the set of PRTs is {*vice-*, *pseudo-*, *post-*, *contre-*, *anti-*, *non-*, *quasi-*, *soviéto-*, *supra-*, *néo-*, *inter-*}. For English it is {*off*, *down*, *To*, *about*, *on*, *in*, *over*, *around*, *back*, *up*, *out*, *to*, *away*, *'*, *'s*}. French particles are therefore always at least two-token visual units that seem to be quite infrequent as well as long, whereas English particles are short and frequent.

3 Features

For our weakly supervised PoS tagging experiments, we use 22 gaze features that measure both early processing and late processing. They are equivalent to the 22 gaze features used by Barrett et al. (2016). Early processing measures are said to reflect different aspects of early syntactic and semantic processing and include first pass duration and first fixation duration. Late processing measures reflect, e.g., late syntactic and semantic integration (Rayner, 1998). Examples are number and duration of regressions going to a word, as well as the total reading time for a word.

Non-gaze features are usually included in eye movement models, because they explain a lot of the variance in fixation durations. Word frequency and word length together have been found to explain 69% of the variance in the mean gaze duration (Carpenter and Just, 1983). Like Barrett et al. (2016), we use word length, log word frequencies from a big corpus and log word frequencies from the Dundee training set for the target word, and the previous and next words. From the Dundee training set, we also extract the forward and backward transitional probability, i.e., the conditional probabilities for a word given the next or previous word. Our non-gaze features are almost equivalent to Barrett et al. (2016). The only difference is that they also used forward and backward transitional probabilities from a big corpus.

The big corpus log frequencies were obtained from the British National Corpus[2] for English, extracted with KenLM (Heafield, 2011) and Lexique[3] for French. The Dundee log frequencies were calculated on the respective training sets using CMU Language Modeling Toolkit[4] with Witten-Bell smooting.

In total we have 29 features. All features are first averaged over all 10 readers of the corpus, then scaled to a value between 0 and 1 by minmax scaling. The best model of the feature ablation study of Barrett et al. (2016) used all features, which suggests that grammatical processing of a broad set of PoS categories is reflected across many features and need non-gaze features as well.

4 Experiment

We replicate the experimental setup of Barrett et al. (2016), which used the best model from Li et al. (2012), a second-order hidden Markov model with maximum entropy emissions (SHMM-ME) constrained by Wiktionary tags such that emissions are confined to the allowed PoS tags of the Wiktionary given that the token exists in the Wiktionary. Li et al. (2012) report considerable improvements from the Wiktionary contraint when comparing to unsupervised methods.

The second-order model includes transition probabilities from the antecedent state like a first order model (Berg-Kirkpatrick et al., 2010) as well as from the second-order antecedent state.

We use the original implementation of Li et al. and we also include a subset of their word-level features, viz., four features detecting hyphens, numerals, punctuation and capitalization. We leave out the three

[2] http://www.natcorp.ox.ac.uk

[3] http://www.lexique.org

[4] http://www.speech.cs.cmu.edu/SLM/toolkit.html

suffix features from Li et al.'s basic feature model, as these features do not transfer across languages. These features were also included by Barrett et al. (2016).

We use the English Wiktionary dumps made available by Li et al.[5] The French Wiktionary dump is from Wisniewski et al. (2014) and does not include any punctuation. We therefore augment it with all punctuation entries from the English Wiktionary. Furthermore, tokens for the tag ADP are completely missing from the French Wiktionary, and the tokens for the class DET were sparse. We therefore add all examples of DET and ADP from the French training set to the French Wiktionary.

For the cross-lingual experiments, we use the union of the French and the English Wiktionary dictionaries.

Barrett et al. (2016) used Li et al.'s model for weakly supervising PoS induction with gaze features for English, and performed model tuning and feature ablation. We use their best hyper-parameter setting, i.e., five EM iterations, as well as the best feature combination: all features. Following Barrett et al. (2016), we try token-level and type-level features. For the token-level experiments, each token is represented by its feature vector. For the type-level experiments, each token is represented by an average of the feature vectors for all occurrences of the lower-cased word type of the training set.

5 Results

The tagging accuracy for all combinations of training and testing language on the development set and the test set can be seen in Table 1.

For all conditions, type-level features work better than token-level, though the type-level improvement over the baseline is not significant for FR-EN.

The English monolingual condition plus suffix is almost equivalent to the best model in Barrett et al. (2016). The only difference is the two missing non-gaze features described in Section 3. On the test set, they report a baseline accuracy of 79.77, a token-level accuracy of 81.00, and a type-level accuracy of 82.44, which is in line with our results. We observe that the suffix features seem to help in the monolingual conditions. For monolingual conditions, we confirm that type-level gaze-features and token-level ones outperform the baseline. These differences are significant, except for the EN-EN token-level plus suffix condition.

FR-FR PoS tagging seems to be a slightly an easier task than EN-EN PoS tagging, achieving overall higher accuracies.

The cross-lingual conditions generally achieve lower performance than the monolingual. When training on English and testing on French, both token-level and type-level conditions are significantly better than baseline.

Metric	Cosine sim
n refixations	0.6318
First pass duration	0.8480
Re-read probability	0.8489
n fixations	0.9097
Total fixation duration	0.9217
n regressions to	0.9354
n long regressions from	0.9375
Total duration of regressions from	0.9377
Total duration of regression to	0.9385
n regressions from	0.9404
n long regressions to	0.9644
Fixation probability	0.9795
w-1 fixation duration	0.9839
w+1 fixation duration	0.9934
w-1 fixation probability	0.9947
w+2 fixation duration	0.9961
w+1 fixation probability	0.9967
w-2 fixation probability	0.9975
w+2 fixation probability	0.9986
First fixation duration	0.9992
Mean fixation duration	0.9992
w-2 fixation duration	0.9992

Table 2: Cosine similarity between PoS averaged French and English train set gaze vectors across gaze features. Sorted by similarity.

6 Error Analysis

There are—as expected—more errors when using cross-lingual gaze data. This section will explore these errors by comparing the predictions of the cross-lingual experiments with the predictions of the mono-

[5] https://code.google.com/archive/p/wikily-supervised-pos-tagger/

lingual experiments. All analysis is on the development set output of the type-level models. We compare them to the output of the type-level monolingual models.

Figure 3 shows accuracy scores per PoS class comparing experiments with same test set. The accuracy of punctuations is due to the basic feature model and the Wiktionary constraints—not the eye movement measures. PRT and NUM are real challenges for FR-EN compared to EN-EN. This can be assumed to be due to the different use of the PRT tag and the missing NUM class in the French dataset described in Section 2.3.1. ADJ also seems like a cross-lingual challenge, though harder when trained on English and tested on French than the other way around.

Figure 4 shows the erroneous predictions per gold PoS tag, allowing us to compare error types across experiments. When comparing Figure 4a and Figure 4c, both evaluated on English, most classes seem to have almost the same set of misclassified labels though for some labels in different magnitude or ratio depending on whether they are trained on English or French. The main differences are: when trained on French, ADP and ADJ are generally more often misclassified, ADP is not mainly misclassified as CONJ, but more often as ADV, DET is also misclassified as VERB and ADV, PRT is misclassified as ADV and not mainly as ADP.

When comparing Figure 4b and Figure 4d, both evaluating on French, we also find that for many of the PoS classes, the misclassifications are of the same type, though different in magnitude or ratio. The main differences we observe when training on English are: ADJ is mainly misclassified as NOUN instead of ADP, ADV, DET, NOUN, and PRT; ADV is misclassified as VERB; DET is never misclassified as PRT, but more often as NOUN and ADJ; and NOUN is rarely misclassified as PRT. The last error probably has to do with the long gaze durations for PRT in the French data (resembling gaze durations of NOUNs) opposed to the short gaze durations of English PRT.

Table 2 shows the cosine similarity between the English and French PoS-averaged gaze vectors from the train set for all gaze features. This gives information about which gaze feature averages differ between French and English PoS. Pynte and Kennedy (2006) found that French had more re-fixations than English, which is reflected in the table. Measures correlating with re-fixations like re-read probability, number of fixtions, and total fixation duration are naturally also different across languages. First pass duration is not directly correlated with number of re-fixations, and must be considered an distinct pattern.

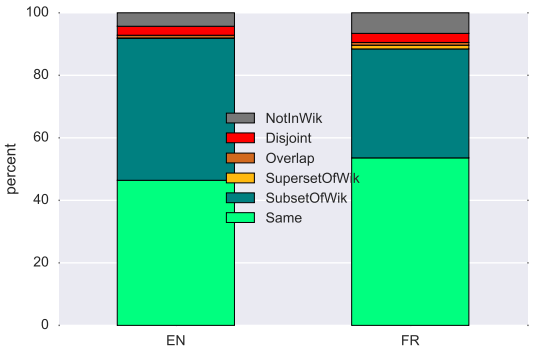


Figure 5: Development set word type lookup in Wiktionary for English and French: the percentage of word types assigned a set of tags that is either: identical to, a subset of, a superset of, overlapping with, disjoint with, or not in the Wiktionary.

6.1 Wiktionary agreement

Figure 5 shows the word types for the English and French development set according to their representation in the respective monolingual Wiktionary. This figure is inspired by Li et al. (2012). For English, more PoS types agree with the Wiktionary (Same or SubsetOfWik) than for French. We also computed token-level accuracies, where a tag licensed by Wiktionary counts as correct. For the French development set, this maximum dictionary accuracy is 0.95, whereas for English it is 0.92.

7 Discussion

We presented four experiments with PoS induction using gaze data in a monolingual and cross-lingual setup with a second-order hidden Markov model. Our experiments confirm the main conclusion from Barrett et al. (2016), viz., that type-level gaze vectors improve PoS induction. We replicated their result

for English and report the same finding for French as well as for French when trained on English gaze vectors.

It is difficult to determine how much the relatedness of the French and English languages is responsible for the ability of the model to generalize cross-lingually. The psycholinguistic literature does not reveal how different PoS categories are processed across languages; most experimental work in the literature studies single phenomena in one language. For instance, in reaction time studies of lexical decision tasks it has been found that the processing of English plural and singular nouns is influenced by surface frequency only[6] (Sereno and Jongman, 1997), whereas for Dutch (Baayen et al., 1997) and French (New et al., 2004), the lexical processing of singular and plural nouns is influenced by the base frequency[7]. The English data thus support a full-storage cognitive model, whereas the French and the Dutch data support the Parallel Dual-Route model where a word is processed as segments in parallel with whole word processing. These results suggest that nouns are processed differently in the brain for native speakers of different languages. This means that our results may not generalize to other combinations of languages and in the specific case of nouns it suggests that Dutch and French nouns are processed more similarly than French and English.

8 Conclusion

This is, to the best of our knowledge, the first study to explore whether gaze features generalize from one language to another for a broad set of syntactic categories. We used a type-constrained second-order HMM for monolingual and cross-lingual PoS induction on the English and French portions of the Dundee eye tracking corpus. We experimented with both token-level and type-level features and confirmed that type-level gaze features improve monolingual PoS induction for both English and French. We also showed that type-level gaze features significantly improve PoS induction for French, even when the model is trained on English gaze vectors.

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for French. In *Treebanks*, pages 165–187.

Harald R Baayen, Ton Dijkstra, and Robert Schreuder. 1997. Singulars and plurals in Dutch: Evidence for a parallel dual-route model. *Journal of Memory and Language*, 37(1):94–117.

Maria Barrett and Anders Søgaard. 2015a. Reading behavior predicts syntactic categories. *CoNLL 2015*, pages 345–349.

Maria Barrett and Anders Søgaard. 2015b. Using reading behavior to predict grammatical functions. In *Workshop on Cognitive Aspects of Computational Language Learning (CogACLL)*, pages 1–5.

Maria Barrett, Željko Agić, and Anders Søgaard. 2015. The Dundee treebank. In *The 14th International Workshop on Treebanks and Linguistic Theories (TLT 14)*, pages 242–248.

Maria Barrett, Joachim Bingel, Frank Keller, and Anders Søgaard. 2016. Weakly supervised part-of-speech tagging using eye-tracking data. In *ACL*, pages 579–584.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Cote, John DeNero, , and Dan Klein. 2010. Painless unsupervised learning with features. In *NAACL*, pages 582–590.

Patricia A Carpenter and Marcel Adam Just. 1983. What your eyes do while your mind is reading. *Eye movements in reading: Perceptual and language processes*, pages 275–307.

Meri Coleman and Ta Lin Liau. 1975. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283–284.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197.

[6] the token frequency of a word form

[7] the sum of the frequencies of all inflections of a word

Alan Kennedy and Joël Pynte. 2005. Parafoveal-on-foveal effects in normal reading. *Vision research*, 45(2):153–168.

Alan Kennedy, Robin Hill, and Joël Pynte. 2003. The Dundee Corpus. *Poster presented at the 12th European Conference on Eye Movement*.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *NAACL*, pages 1528–1533.

Shen Li, João Graça, and Ben Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *EMNLP*, pages 1389–1398.

Boris New, Marc Brysbaert, Juan Segui, Ludovic Ferrand, and Kathleen Rastle. 2004. The processing of singular and plural nouns in French and English. *Journal of Memory and Language*, 51(4):568–585.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. CoRR abs/1104.2086.

Joel Pynte and Alan Kennedy. 2006. An influence over eye movements in reading exerted from beyond the level of the word: Evidence from reading English and French. *Vision Research*, 46(22):3786–3801.

Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological bulletin*, 124(3):372–422.

Joan A Sereno and Allard Jongman. 1997. Processing of English inflectional morphology. *Memory & Cognition*, 25(4):425–437.

Laurent Sparrow, Sébastien Miellet, and Yann Coello. 2003. The effects of frequency and predictability on eye fixations in reading: An evaluation of the EZ reader model. *Behavioral and Brain Sciences*, 26(04):503–505.

Guillaume Wisniewski, Nicolas Pécheux, Souhir Gahbiche-Braham, and François Yvon. 2014. Cross-lingual part-of-speech tagging through ambiguous learning. In *EMNLP*, volume 14, pages 1779–1785.

Sentence Similarity Learning by Lexical Decomposition and Composition

Zhiguo Wang and **Haitao Mi** and **Abraham Ittycheriah**
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
{zhigwang, hmi, abei}@us.ibm.com

Abstract

Most conventional sentence similarity methods only focus on similar parts of two input sentences, and simply ignore the dissimilar parts, which usually give us some clues and semantic meanings about the sentences. In this work, we propose a model to take into account both the similarities and dissimilarities by decomposing and composing lexical semantics over sentences. The model represents each word as a vector, and calculates a semantic matching vector for each word based on all words in the other sentence. Then, each word vector is decomposed into a similar component and a dissimilar component based on the semantic matching vector. After this, a two-channel CNN model is employed to capture features by composing the similar and dissimilar components. Finally, a similarity score is estimated over the composed feature vectors. Experimental results show that our model gets the state-of-the-art performance on the answer sentence selection task, and achieves a comparable result on the paraphrase identification task.

1 Introduction

Sentence similarity is a fundamental metric to measure the degree of likelihood between a pair of sentences. It plays an important role for a variety of tasks in both NLP and IR communities. For example, in paraphrase identification task, sentence similarity is used to determine whether two sentences are paraphrases or not (Yin and Schütze, 2015; He et al., 2015). For question answering and information retrieval tasks, sentence similarities between query-answer pairs are used for assessing the relevance and ranking all the candidate answers (Severyn and Moschitti, 2015; Wang and Ittycheriah, 2015).

However, sentence similarity learning has following challenges:

1. There is a lexical gap between semantically equivalent sentences. Take the E_1 and E_2 in Table 1 for example, they have the similar meaning but with different lexicons.

2. Semantic similarity should be measured at different levels of granularity (word-level, phrase-level and syntax-level). E.g., "not related" in E_2 is an indivisible phrase when matching with "irrelevant" in E_1 (shown in square brackets).

3. The dissimilarity (shown in angle brackets) between two sentences is also a significant clue (Qiu et al., 2006). For example, by judging the dissimilar parts, we can easily identify that E_3 and E_5 share the similar meaning "The study is about salmon", because "sockeye" belongs to the salmon family, and "flounder" does not. Whereas the meaning of E_4 is quite different from E_3, which emphasizes "The study is about red (a special kind of) salmon", because both "sockeye" and "coho" are in the salmon family. How we can extract and utilize those information becomes another challenge.

In order to handle the above challenges, researchers have been working on sentence similarity algorithms for a long time. To bridge the lexical gap (challenge 1), some word similarity metrics were proposed to match different but semantically related words. Examples include knowledge-based metrics (Resnik, 1995) and corpus-based metrics (Jiang and Conrath, 1997; Yin and Schütze, 2015; He et al., 2015). To measure sentence similarity from various granularities (challenge 2), researchers have explored features extracted from n-grams, continuous phrases, discontinuous phrases, and parse trees (Yin and Schütze, 2015; He et al., 2015; Heilman and Smith, 2010). The third challenge did not get much

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1340–1349, Osaka, Japan, December 11-17 2016.

E_1	The research is [irrelevant] to sockeye.
E_2	The study is [not related] to salmon.
E_3	The research is relevant to salmon.
E_4	The study is relevant to sockeye, ⟨instead of coho⟩.
E_5	The study is relevant to sockeye, ⟨rather than flounder⟩.

Table 1: Examples for sentence similarity learning, where sockeye means "red salmon", and coho means "silver salmon". "coho" and "sockeye" are in the salmon family, while "flounder" is not.

attention in the past, the only related work of Qiu et al. (2006) explored the dissimilarity between sentences in a pair for paraphrase identification task, but they require human annotations in order to train a classifier, and their performance is still below the state of the art.

In this paper, we propose a novel model to tackle all these challenges jointly by decomposing and composing lexical semantics over sentences. Given a sentence pair, the model represents each word as a low-dimensional vector (challenge 1), and calculates a semantic matching vector for each word based on all words in the other sentence (challenge 2). Then based on the semantic matching vector, each word vector is decomposed into two components: a similar component and a dissimilar component (challenge 3). We use similar components of all the words to represent the similar parts of the sentence pair, and dissimilar components of every word to model the dissimilar parts explicitly. After this, a two-channel CNN operation is performed to compose the similar and dissimilar components into a feature vector (challenge 2 and 3). Finally, the composed feature vector is utilized to predict the sentence similarity. Experimental results on two tasks show that our model gets the state-of-the-art performance on the answer sentence selection task, and achieves a comparable result on the paraphrase identification task.

In following parts, we start with a brief overview of our model (Section 2), followed by the details of our end-to-end implementation (Section 3). Then we evaluate our model on answer sentence selection and paraphrase identifications tasks (Section 4).

2 Model Overview

In this section, we propose a sentence similarity learning model to tackle all three challenges (mentioned in Section 1). To deal with the first challenge, we represent each word as a distributed vector, so that we can calculate similarities for formally different but semantically related words. To tackle the second challenge, we assume that each word can be semantically matched by several words in the other sentence, and we calculate a semantic matching vector for each word vector based on all the word vectors in the other side. To cope with the third challenge, we assume that each semantic unit (word) can be partially matched, and can be decomposed into a similar component and a dissimilar component based on its semantic matching vector.

Figure 1 shows an overview of our sentence similarity model. Given a pair of sentences S and T, our task is to calculate a similarity score $sim(S, T)$ in following steps:

Word Representation. Word embedding of Mikolov et al. (2013) is an effective way to handle the lexical gap challenge in the sentence similarity task, as it represents each word with a distributed vector, and words appearing in similar contexts tend to have similar meanings (Mikolov et al., 2013). With those pre-trained embeddings, we transform S and T into sentence matrixes $S = [s_1, ..., s_i, ..., s_m]$ and $T = [t_1, ..., t_j, ..., t_n]$, where s_i and t_j are d-dimension vectors of the corresponding words, and m and n are sentence length of S and T respectively.

Semantic Matching. In order to judge the similarity between two sentences, we need to check whether each semantic unit in one sentence is covered by the other sentence, or vice versa. For example, in Table 1, to check whether E_2 is a paraphrase of E_1, we need to know the single word "irrelevant" in E_1 is matched or covered by the phrase "not related" in E_2. In our model, we treat each word as a primitive semantic unit, and calculate a semantic matching vector $\hat{s}_i$ for each word s_i by composing part or full word vectors in the other sentence T. In this way, we can match a word s_i to a word or phrase in T.

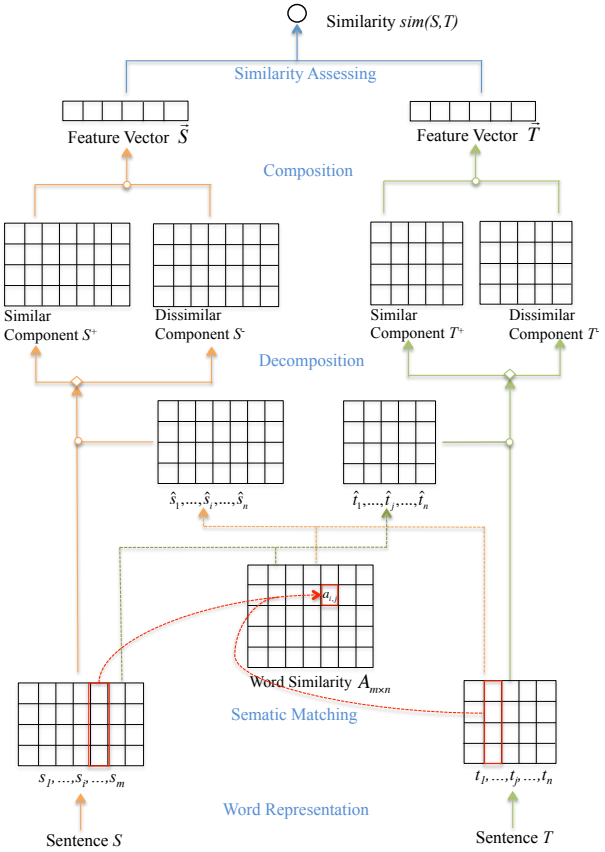


Figure 1: Model overview.

Similarly, for the reverse direction, we also calculate all semantic matching vectors $\hat{t_j}$ in T.

$$\begin{aligned} \hat{s_i} &= f_{match}(s_i, T) \qquad \forall s_i \in S \\ \hat{t_j} &= f_{match}(t_j, S) \qquad \forall t_j \in T \end{aligned} \tag{1}$$

We explore different f_{match} functions later in Section 3.

Decomposition. After the semantic matching phase, we have the semantic matching vectors of $\hat{s_i}$ and $\hat{t_j}$. We interpret $\hat{s_i}$ (or $\hat{t_j}$) as a semantic coverage of word s_i (or t_j) by the sentence T (or S). However, it is not necessary that all the semantics of s_i (or t_j) are fully covered by $\hat{s_i}$ (or $\hat{t_j}$). Take the E_1 and E_2 in Table 1 for example, the word "sockeye" in E_1 is only partially matched by the word "salmon" (the similar part) in E_2, as the full meaning of "sockeye" is "red salmon" (the semantic meaning of "red" is the dissimilar part). Motivated by this phenomenon, our model further decomposes word s_i (or t_j), based on its semantic matching vector $\hat{s_i}$ (or $\hat{t_j}$), into two components: similar component s_i^+ (or t_j^+) and dissimilar component s_i^- (or t_j^-). Formally, we define the decomposition function as:

$$\begin{aligned} [s_i^+; s_i^-] &= f_{decomp}(s_i, \hat{s_i}) \qquad \forall s_i \in S \\ [t_j^+; t_j^-] &= f_{decomp}(t_j, \hat{t_j}) \qquad \forall t_j \in T \end{aligned} \tag{2}$$

Composition. Given a similar component matrix $S^+ = [s_1^+, ..., s_m^+]$ (or $T^+ = [t_1^+, ..., t_n^+]$) and a dissimilar component matrix $S^- = [s_1^-, ..., s_m^-]$ (or $T^- = [t_1^-, ..., t_n^-]$), our goal in this step is how to utilize those information. Besides the suggestion from Qiu et al. (2006) that the significance of the dissimilar parts alone between two sentences has a great effect of their similarity, we also think that the dissimilar and similar components have strong connections. For example, in Table 1, if we only look at the dissimilar or similar part alone, it is hard to judge which one between E_4 and E_5 is more similar to E_3. We can easily identify that E_5 is more similar to E_3, when we consider both the similar and dissimilar

parts. Therefore, our model composes the similar component matrix and dissimilar component matrix into a feature vector $\vec{S}$ (or $\vec{T}$) with the composition function:

$$\begin{aligned} \vec{S} &= f_{comp}(S^+, S^-) \\ \vec{T} &= f_{comp}(T^+, T^-) \end{aligned} \tag{3}$$

Similarity assessing. In the final stage, we concatenate the two feature vectors ($\vec{S}$ and $\vec{T}$) and predict the final similarity score:

$$sim(S, T) = f_{sim}(\vec{S}, \vec{T}) \tag{4}$$

3 An End-to-End Implementation

Section 2 gives us a glance of our model. In this section, we describe details of each phase.

3.1 Semantic Matching Functions

This subsection describes our specifications for the semantic matching function f_{match} in Eq. (1). The goal of f_{match} is to generate a semantic matching vector $\hat{s}_i$ for s_i by composing the vectors from T.

For a sentence pair S and T, we first calculate a similarity matrix $A_{m\times n}$, where each element $a_{i,j} \in A_{m\times n}$ computes the cosine similarity between words s_i and t_j as

$$a_{i,j} = \frac{s_i^T t_j}{\|s_i\| \cdot \|t_j\|} \qquad \forall s_i \in S, \forall t_j \in T. \tag{5}$$

Then, we define three different semantic matching functions over $A_{m\times n}$:

$$f_{match}(s_i, T) = \begin{cases} \frac{\sum_{j=0}^{n} a_{i,j} t_j}{\sum_{j=0}^{n} a_{i,j}} & \textit{global} \\ \frac{\sum_{j=k-w}^{k+w} a_{i,j} t_j}{\sum_{j=k-w}^{k+w} a_{i,j}} & \textit{local-w} \\ t_k & \textit{max} \end{cases} \tag{6}$$

where $k = \text{argmax}_j\, a_{i,j}$. The idea of the *global* function is to consider all word vectors t_j in T. A semantic matching vector $\hat{s}_i$ is a weighted sum vector of all words t_j in T, where each weight is the normalized word similarity $a_{i,j}$. The *max* function moves to the other extreme. It generates the semantic matching vector by selecting the most similar word vector t_k from T. The *local-w* function takes a compromise between *global* and *max*, where w indicates the size of the window to consider centered at k (the most similar word position). So the semantic matching vector is a weighted average vector from t_{k-w} to t_{k+w}.

3.2 Decomposition Functions

This subsection describes the implementations for the decomposition function f_{decomp} in Eq. (2). The intention of f_{decomp} is to decompose a word vector s_j based on its semantic matching vector $\hat{s}_j$ into a similar component s_i^+ and a dissimilar component s_i^-, where s_i^+ indicates the semantics of s_i covered by $\hat{s}_i$ and s_i^- indicates the uncovered part. We implement three types of decomposition function: *rigid*, *linear* and *orthogonal*.

The *rigid* decomposition only adapts to the *max* version of f_{match}. First, it detects whether there is an exactly matched word in the other sentence, or s_i equal to $\hat{s}_i$. If yes, the vector s_i is dispatched to the similar component s_i^+, and the dissimilar component is assigned with a zero vector $\mathbf{0}$. Otherwise, the vector s_i is assigned to the dissimilar component s_i^-. Eq. (7) gives the formal definition:

$$\begin{aligned} &[s_i^+ = s_i;\ s_i^- = \mathbf{0}] \qquad if\ s_i = \hat{s}_i \\ &[s_i^+ = \mathbf{0};\ s_i^- = s_i] \qquad otherwise \end{aligned} \tag{7}$$

The motivation for the *linear* decomposition is that the more similar between s_i and $\hat{s}_i$, the higher proportion of s_i should be assigned to the similar component. First, we calculate the cosine similarity

α between s_i and $\hat{s}_i$. Then, we decompose s_i linearly based on α. Eq. (8) gives the corresponding definition:

$$\begin{aligned} \alpha &= \frac{s_i^T \hat{s}_i}{\|s_i\| \cdot \|\hat{s}_i\|} \\ s_i^+ &= \alpha s_i \\ s_i^- &= (1-\alpha) s_i \end{aligned} \tag{8}$$

The *orthogonal* decomposition is to decompose a vector in the geometric space. Based on the semantic matching vector $\hat{s}_i$, our model decomposes s_i into a parallel component and a perpendicular component. Then, the parallel component is viewed as the similar component s_i^+, and perpendicular component is taken as the dissimilar component s_i^-. Eq. (9) gives the concrete definitions.

$$\begin{aligned} s_i^+ &= \frac{s_i \cdot \hat{s}_i}{\hat{s}_i \cdot \hat{s}_i} \hat{s}_i \qquad parallel \\ s_i^- &= s_i - s_i^+ \qquad perpendicular \end{aligned} \tag{9}$$

3.3 Composition Functions

The aim of composition function f_{comp} in Eq. (3) is to extract features from both the similar component matrix and the dissimilar component matrix. We also want to acquire similarities and dissimilarities of various granularity during the composition phase. Inspired from Kim (2014), we utilize a two-channel convolutional neural networks (CNN) and design filters based on various order of n-grams, e.g., unigram, bigram and trigram.

The CNN model involves two sequential operations: *convolution* and *max-pooling*. For the *convolution* operation, we define a list of filters $\{w_o\}$. The shape of each filter is $d \times h$, where d is the dimension of word vectors and h is the window size. Each filter is applied to two patches (a window size h of vectors) from both similar and dissimilar channels, and generates a feature. Eq. (10) expresses this process.

$$c_{o,i} = f(w_o * S^+_{[i:i+h]} + w_o * S^-_{[i:i+h]} + b_o) \tag{10}$$

where the operation $A * B$ sums up all elements in B with the corresponding weights in A, $S^+_{[i:i+h]}$ and $S^-_{[i:i+h]}$ indicate the patches from S^+ and S^-, b_o is a bias term and f is a non-linear function (we use *tanh* in this work). We apply this filter to all possible patches, and produce a series of features $\vec{c_o} = [c_{o,1}, c_{o,2}, ..., c_{o,O}]$. The number of features in $\vec{c_o}$ depends on the shape of the filter w_o and the length of the input sentence. To deal with variable feature size, we perform a *max-pooling* operation over $\vec{c_o}$ by selecting the maximum value $c_o = max\ \vec{c_o}$. Therefore, after these two operations, each filter generates only one feature. We define several filters by varying the window size and the initial values. Eventually, a vector of features is captured by composing the two component matrixes, and the feature dimension is equal to the number of filters.

3.4 Similarity Assessment Function

The similarity assessment function f_{sim} in Eq. (4) predicts a similarity score by taking two feature vectors as input. We employ a linear function to sum up all the features and apply a *sigmoid* function to constrain the similarity within the range [0, 1].

3.5 Training

We train our sentence similariy model by maximizing the likelihood on a training set. Each training instance in the training set is represented as a triple (S_i, T_i, L_i), where S_i and T_i are a pair of sentences, and $L_i \in \{0, 1\}$ indicates the similarity between them. We assign $L_i = 1$ if T_i is a paraphrase of S_i for the paraphrase identification task, or T_i is a correct answer for S_i for the answer sentence selection task. Otherwise, we assign $L_i = 0$. We implement the mathematical expressions with Theano (Bastien et al., 2012) and use Adam (Kingma and Ba, 2014) for optimization.

4 Experiment

4.1 Experimental Setting

We evaluate our model on two tasks: answer sentence selection and paraphrase identification. The answer sentence selection task is to rank a list of candidate answers based on their similarities to a question sentence, and the performance is measured by mean average precision (MAP) and mean reciprocal rank (MRR). We experiment on two datasets: *QASent* and *WikiQA*. The statistics of the two datasets can be found in Yang et al. (2015), where *QASent* (Wang et al., 2007) was created from the TREC QA track, and *WikiQA* (Yang et al., 2015) is constructed from real queries of Bing and Wikipedia. The paraphrase identification task is to detect whether two sentences are paraphrases based on the similarity between them. The metrics include the accuracy and the positive class F_1 score. We experiment on the Microsoft Research Paraphrase corpus (MSRP) (Dolan et al., 2004), which includes 2753 true and 1323 false instances in the training set, and 1147 true and 578 false instances in the test set. We build a development set by randomly selecting 100 true and 100 false instances from the training set. In all experiments, we set the size of word vector dimension as d =300, and pre-train the vectors with the *word2vec* toolkit (Mikolov et al., 2013) on the English Gigaword (LDC2011T07).

4.2 Model Properties

There are several alternative options in our model, e.g., the semantic matching functions, the decomposition operations, and the filter types. The choice of these options may affect the final performance. In this subsection, we present some experiments to demonstrate the properties of our model, and find a good configuration that we use to evaluate our final model. All the experiments in this subsection were performed on the *QASent* dataset and evaluated on the development set.

First, we evaluated the effectiveness of various semantic matching functions. We switched the semantic matching functions among {*max*, *global*, *local-l*}, where $l \in \{1, 2, 3, 4\}$, and fixed the other options as: the *linear* decomposition, the filter types including {unigram, bigram, trigram}, and 500 filters for each type. Figure 2 (a) presents the results. We found that the *max* function worked better than the *global* function on both MAP and MRR. By increasing the window size, the *local-l* function acquired progressive improvements when the window size is smaller than 4. But after we enlarged the window size to 4, the performance dropped. The *local-3* function worked better than the *max* function in term of the MAP, and also got a comparable MRR. Therefore, we use the *local-3* function in the following experiments.

Second, we studied the effect of various decomposition operations. We varied the decomposition operation among {*rigid*, *linear*, *orthogonal*}, and kept the other options unchanged. Figure 2 (b) shows the performance. We found that the *rigid* operation got the worst result. This is reasonable, because the *rigid* operation decomposes word vectors by exactly matching words. The *orthogonal* operation got a similar MAP as the *linear* operation, and it worked better in term of MRR. Therefore, we choose the *orthogonal* operation in the following experiments.

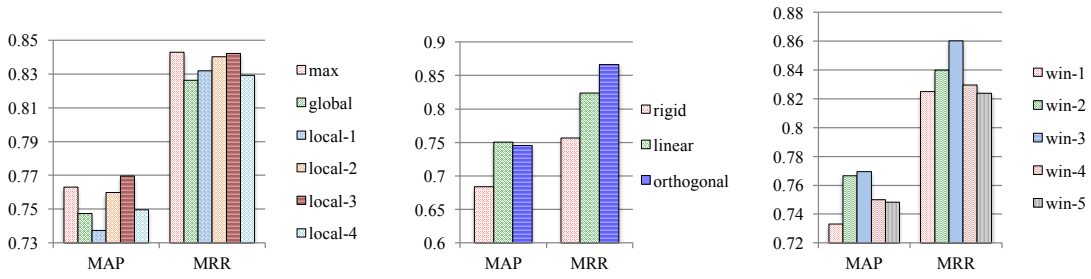
Third, we tested the influence of various filter types. We constructed 5 groups of filters: *win-1* contains only the unigram filters, *win-2* contains both unigram and bigram filters, *win-3* contains all the filters in *win-2* plus trigram filters, *win-4* extends filters in *win-3* with 4-gram filters, and *win-5* adds 5-gram filters into *win-4*. We generate 500 filters for each filter type (with different initial values). Experimental results are shown in Figure 2 (c). At the beginning, adding higher-order ngram filters was helpful for the performance. The performance reached to the peak, when we used the *win-3* filters. After that, adding more complex filters decreased the performance. Therefore, the trigram is the best granularity for our model. In the following experiments, we utilize filter types in *win-3*.

4.3 Comparing with State-of-the-art Models

In this subsection, we evaluated our model on the test sets of *QASent*, *WikiQA* and *MSRP*.

***QASent* dataset.** Table 2 presents the performances of the state-of-the-art systems and our model, where the performances were evaluated with the standard trec_eval-8.1 script [1]. Given a pair of sentences,

[1] http://trec.nist.gov/trec_eval/



(a) Semantic matching. (b) Decomposition. (c) Filter types in composition.

Figure 2: Influence of different configuration.

Models	MAP	MRR
Severyn and Moschitti (2015) (CNN only)	0.6709	0.7280
Severyn and Moschitti (2015) (CNN + sparse features)	0.7459	0.8078
Wang and Ittycheriah (2015) (Word embedding alignment)	0.7460	0.8200
dos Santos et al. (2016) (Attention-based CNN)	0.7530	**0.8511**
This work	**0.7714**	0.8447

Table 2: Results on the QASent dataset.

Models	MAP	MRR
Yang et al. (2015) (2-gram CNN)	0.6520	0.6652
dos Santos et al. (2016) (Attention-based CNN)	0.6886	0.6957
Miao et al. (2015) (Attention-based LSTM)	0.6886	0.7069
Yin et al. (2015) (Attention-based CNN)	0.6921	0.7108
This work	**0.7058**	**0.7226**

Table 3: Results on the WikiQA dataset.

Severyn and Moschitti (2015) employed a CNN model to compose each sentence into a vector separately, and joined the two sentence vectors to compute the sentence similarity. Because only the sentence-level granularity was used, the performance is much lower (the second row of Table 2). After adding some word overlap features between the two sentences, the performance was improved significantly (the third row of Table 2). Therefore, the lower-level granularity is an indispensable factor for a good performance. Wang and Ittycheriah (2015) conducted word alignment for a sentence pair based on word vectors, and measured the sentence similarity based on a couple of word alignment features. They got a slightly better performance (the fourth row of Table 2), which indicates that the vector representation for words is helpful to bridging the lexical gap problem. dos Santos et al. (2016) introduced the attention mechanism into the CNN model, and learnt sentence representation by considering the influence of the other sentence. They got better performance than all the other previous work. Our model makes use of all these useful factors and also considers the dissimilarities of a sentence pair. We can see that our model (the last row of Table 2) got the best MAP among all previous work, and a comparable MRR than dos Santos et al. (2016).

***WikiQA* dataset.** Table 3 presents the results of our model and several state-of-the-art models. Yang et al. (2015) constructed the dataset and reimplemented several baseline models. The best performance (shown at the second row of Table 3) was acquired by a bigram CNN model combining with the word overlap features. Miao et al. (2015) models the sentence similarity by enriching LSTMs with a latent stochastic attention mechanism. The corresponding performance is given at the fourth row of Table 3. Yin et al. (2015) introduced the attention mechanism into the CNN model, and captured the best performance (the fifth row of Table 3). The semantic matching phase in our model is similar to the attention mechanism. But different from the previous models, our model utilizes both the similarity and dissimilarity simultaneously. The last row of Table 3 shows that our model is more effective than the other models.

***MSRP* dataset.** Table 4 summarized the results from our model and several state-of-the-art models. Yin and Schütze (2015) employed a CNN model to learn sentence representations on multiple level of

Models	Acc	F1
Yin and Schütze (2015) (without pretraining)	72.5	81.4
Yin and Schütze (2015) (with pretraining)	78.4	84.6
He et al. (2015) (without POS embeddings)	77.8	N/A
He et al. (2015) (without Para. embeddings)	77.3	N/A
He et al. (2015) (POS and Para. embeddings)	78.6	84.7
Yin et al. (2015) (with sparse features)	78.9	84.8
Ji and Eisenstein (2013)	**80.4**	**86.0**
This work	78.4	84.7

Table 4: Experimental results for paraphrase identification on MSRP corpus.

granularity and modeled interaction features at each level for a pair of sentences. They obtained their best performance by pretraining the model on a language modeling task (the 3rd row of Table 4). However, their model heavily depends on the pretraining strategy. Without pretraining, they got a much worse performance (the second row of Table 4). He et al. (2015) proposed a similar model to Yin and Schütze (2015). Similarly, they also used a CNN model to extract features at multiple levels of granularity. Differently, they utilized some extra annotated resources, e.g., embeddings from part-of-speech (POS) tags and PARAGRAM vectors trained from the Paraphrase Database (Ganitkevitch et al., 2013). Their model outperformed Yin and Schütze (2015) without the need of pretraining (the sixth row of Table 4). However, the performance was reduced after removing the extra resources (the fourth and fifth rows of Table 4). Yin et al. (2015) applied their attention-based CNN model on this dataset. By adding a couple of sparse features and using a layerwise training strategy, they got a pretty good performance. Comparing to these neural network based models, our model obtained a comparable performance (the last row of Table 4) without using any sparse features, extra annotated resources and specific training strategies. However, the best performance so far on this dataset is obtained by Ji and Eisenstein (2013). In their model, they just utilized several hand-crafted features in a Support Vector Machine (SVM) model. Therefore, the deep learning methods still have a long way to go for this task.

5 Related Work

The semantic matching functions in subsection 3.1 are inspired from the attention-based neural machine translation (Bahdanau et al., 2014; Luong et al., 2015). However, most of the previous work using the attention mechanism in only LSTM models. Whereas our model introduces the attention mechanism into the CNN model. A similar work is the attention-based CNN model proposed by Yin et al. (2015). They first build an attention matrix for a sentence pair, and then directly take the attention matrix as a new channel of the CNN model. Differently, our model uses the attention matrix (or similarity matrix) to decompose the original sentence matrix into a similar component matrix and a dissimilar component matrix, and then feeds these two matrixes into a two-channel CNN model. The model can then focus much on the interactions between similar and dissimilar parts of a sentence pair.

6 Conclusion

In this work, we proposed a model to assess sentence similarity by decomposing and composing lexical semantics. To bridge the lexical gap problem, our model represents each word with its context vector. To extract features from both the similarity and dissimilarity of a sentence pair, we designed several methods to decompose the word vector into a similar component and a dissimilar component. To extract features at multiple levels of granularity, we employed a two-channel CNN model and equipped it with multiple types of ngram filters. Experimental results show that our model is quite effective on both the

answer sentence selection task and the paraphrase identification task .

Acknowledgments

We thank the anonymous reviewers for useful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *HLT-NAACL*, pages 758–764.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586.

Michael Heilman and Noah A Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019. Association for Computational Linguistics.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *EMNLP*, pages 891–896.

Jay J Jiang and David W Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representation (ICLR)*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2015. Neural variational inference for text processing. *arXiv preprint arXiv:1511.06038*.

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

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 18–26. Association for Computational Linguistics.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM.

Zhiguo Wang and Abraham Ittycheriah. 2015. Faq-based question answering via word alignment. *arXiv preprint arXiv:1507.02628*.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*, volume 7, pages 22–32.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs.

Chinese Hypernym-Hyponym Extraction from User Generated Categories

Chengyu Wang, Xiaofeng He*
School of Computer Science and Software Engineering
East China Normal University, Shanghai, China
chywang2013@gmail.com, xfhe@sei.ecnu.edu.cn

Abstract

Hypernym-hyponym ("*is-a*") relations are key components in taxonomies, object hierarchies and knowledge graphs. While there is abundant research on *is-a* relation extraction in English, it still remains a challenge to identify such relations from Chinese knowledge sources accurately due to the flexibility of language expression. In this paper, we introduce a weakly supervised framework to extract Chinese *is-a* relations from user generated categories. It employs piecewise linear projection models trained on a Chinese taxonomy and an iterative learning algorithm to update models incrementally. A pattern-based relation selection method is proposed to prevent "semantic drift" in the learning process using bi-criteria optimization. Experimental results illustrate that the proposed approach outperforms state-of-the-art methods.

1 Introduction

A hypernym-hyponym ("*is-a*") relation is a word/phrase pair (x, y) such that x is a hyponym of y. These relations are extensively employed in machine reading (Etzioni et al., 2011), query understanding (Hua et al., 2015) and other NLP tasks. The extraction of *is-a* relations is necessary to construct taxonomies for Web-scale knowledge graphs (Suchanek et al., 2007; Wu et al., 2012; Wang et al., 2015).

In previous work, *is-a* relations were obtained by either using expert-compiled thesauri such as WordNet (Miller, 1995), or harvested automatically from the Web. Since knowledge in thesauri is usually limited in quantity and variety, it is more prevalent to harvest *is-a* relations from online encyclopedias (Ponzetto and Strube, 2007), Web corpora (Wu et al., 2012), etc. Currently, a majority of existing methods focus on syntactic, lexical and/or semantic analysis on English corpora, but most of these approaches are language dependent. It is not easy to apply methods for one language to knowledge sources in another language directly. For example, in Chinese, the word formation, grammar, semantics and tenses are flexible and more irregular. Thus patttern-based methods can only cover few linguistic circumstances. As pointed out by Li et al. (2013), the performance of syntactic analysis and named entity recognition on Chinese corpora still needs to be improved to support robust relation extraction. Furthermore, it is still difficult to use machine translation-based methods to extract such relations because there are great differences in word orders between English and Chinese (Cai et al., 2014).

More recently, word embedding (or distributed word representation) has been empirically proved effective in modeling some of the semantic relations between words by offsets of word vectors (Mikolov et al., 2013a; Mikolov et al., 2013b). The learning of word embeddings only requires shallow processing of a large text corpus. As Fu et al. (2014) suggest, the representation of *is-a* relations is more complicated than vector offsets. By studying the relations of word embeddings between hyponyms and their respective hypernyms, *is-a* relations can be identified by learning semantic prediction models.

In this paper, we consider the problem of harvesting Chinese *is-a* relations from user generated categories, which frequently appear in online encyclopedias and vertical websites. These category names

* Corresponding author.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1350–1361, Osaka, Japan, December 11-17 2016.

are classes, concepts or topics manually added by human contributors. For instance, in *Baidu Baike*[1], the page "奥巴马(Barack Obama)" has the following categories: "政治人物(Political Figure)", "外国(Foreign Country)", "元首(Leader)" and "人物(Person)". Given an entity and its category set, we aim to predict whether each category name is the hypernym of the entity. We observe that vector offsets of *is-a* relations are quite different in varied data sources and domains (discussed in Section 3). This implies that using a single model is difficult to preserve all the linguistic regularities of *is-a* relations. Furthermore, models learned from one knowledge source are not necessarily effective to extract *is-a* relations from another source, while it is a common practice to construct large-scale taxonomies from multiple Web sources (Fu et al., 2013; Dong et al., 2014; Wang et al., 2014).

To address this problem, we propose a weakly supervised framework to extract *is-a* relations automatically. In the initial stage, we build piecewise linear projection models trained on samples from an existing Chinese taxonomy (Li et al., 2015). In this stage, a K-means based incremental clustering technique is employed to group *is-a* relations with similar semantics together. In each cluster, a separate model maps entities to their respective hypernyms in the embedding space. After that, clustering results are updated incrementally with projection models retrained in an iterative manner. In each iteration, we extract previously unseen *is-a* relations from a collection of unlabeled <entity, category> pairs. To avoid "semantic drift" (Carlson et al., 2010b), a bi-criteria optimization method is proposed such that only those extracted *is-a* relations that are validated by three types of Chinese patterns in a corpus can be labeled as "positive" and added to the training set. In this way, projection models for the target knowledge source are trained without any labeling efforts.

The rest of this paper is organized as follows. Section 2 summarizes the related work. Details of our approach for addressing the *is-a* relation extraction problem are described in Section 3. Experimental results are presented in Section 4. We conclude our paper and discuss the future work in Section 5.

2 Related Work

The *is-a* relation extraction problem has been addressed by identifying hyponyms and their hypernyms from various data sources. Here, we present a summarization on methods on *is-a* relation extraction.

Pattern matching based methods employ syntactic/lexical patterns to extract *is-a* relations. The early work introduced by Hearst (1992) utilizes manually designed patterns to obtain *is-a* relations from text corpora. For instance, based on "NP_1 such as NP_2", it can be inferred that NP_2 is the hypernym of NP_1, where NP_1 and NP_2 are noun phases. These patterns are effective for English and are used to build the largest taxonomy Probase (Wu et al., 2012). However, it is hard to handcraft all valid *is-a* patterns. Ortega-Mendoza et al. (2007) use "seed instances" (i.e., *is-a* word pairs) to discover lexical patterns from the Web using search engines and harvest new instances automatically. Snow et al. (2004) detect syntactic *is-a* patterns by analyzing the parse trees and train a hypernym classifier based on syntactic features. Similar approaches have been adopted in a variety of research (Caraballo, 1999; Etzioni et al., 2004; Sang, 2007; Ritter et al., 2009; Pantel and Pennacchiotti, 2006; Kozareva and Hovy, 2010). As Fu et al. (2014) suggest, many *is-a* relations are expressed in highly flexible manners in Chinese and these approaches have limited extraction accuracy.

Thesauri and encyclopedias can serve as knowledge sources to construct object hierarchies. Suchanek et al. (2007) link concepts in Wikipedia to WordNet synsets (Miller, 1995) by considering the textual patterns of Wikipedia categories. Ponzetto and Strube (2007) design lexical, syntactic and connectivity features to predict whether there is an *is-a* relation between a Wikipedia entity and its category. For Chinese language, Li et al. (2015) introduce a set of language-specific features to predict *is-a* relations using a SVM classifier and construct a large-scale Chinese taxonomy from Wikipedia. Fu et al. (2013) utilize multiple data sources such as encyclopedias and search engine results to design a ranking function in order to extract the most possible hypernym given an entity. Cross-lingual links in Wikipedia are leveraged in (Wang et al., 2014) to derive a bilingual taxonomy by a dynamic boosting model. These methods are more precise than free text extraction but have limited scope constrained by sources.

[1]Baidu Baike (http://baike.baidu.com/) is one of the largest encyclopedias in China, with over 13M entries up till July, 2016.

Text inference approaches make use of distributional similarity measures, which go beyond pattern matching and instead compare the contexts of word pairs in a corpus to infer their relations indirectly. Kotlerman et al. (2010) consider the asymmetric property of *is-a* relations and design directional similarity measures to make lexical inference. Other directional measures are proposed in (Bhagat et al., 2007; Szpektor et al., 2007; Clarke, 2012; Lenci and Benotto, 2012). These methods assume that a hyponym can only appear in some of the contexts of its hypernym and a hypernym can appear in all contexts of its hyponyms. One potential limitation is that the contexts in Chinese are usually flexible and sparse.

To tackle the data sparsity problem, word embedding based approaches have been proposed to solve a series of NLP tasks, such as sentiment classification (Zhou et al., 2015), machine translation (Zhang et al., 2014) and question answering (Yang et al., 2014). In these approaches, words are mapped to a low dimensional space by training neural network based language models, such as *CBOW* and *Skip-gram* models (Mikolov et al., 2013a). The dense word representations are more likely to deal with the context sparsity issue in Chinese stemmed from the flexible expressions. The state-of-the-art method in (Fu et al., 2014) is most related to ours, which takes a Chinese thesaurus as a-priori knowledge and train piecewise linear projection models based on word embeddings. In this paper, we further improve the performance of the word embedding based method by iterative learning of projection models and *is-a* relation selection based on Chinese textual patterns.

3 Weakly Supervised Is-a Relation Extraction

In this section, we describe the formal definition of our problem. The motivation of our method is discussed and the detailed steps introduced, namely, initial model training and iterative learning process.

3.1 Problem Statement

A taxonomy is a *direct acyclic graph* $G = (E, R)$ where nodes E represent entities/classes and edges R denote *is-a* relations. Following the work in Fu et al. (2014), *is-a* relations are regarded as *asymmetric* and *transitive* relations. Therefore, all correct *is-a* relations derived from G are in the transitive closure of R, denoted as R^* where $R^* = \bigcup_{i=0}^{\infty} R^{(i)}$ and $R^{(i+1)} = R \circ R^{(i)}$ with initial condition $R^{(0)} = R$ and $\circ$ being the composition operator of relations.

To extract *is-a* relations from user generated categories, we obtain the collection of entities E^* from the knowledge source (such as *Baidu Baike*). The set of user generated categories for each $e \in E^*$ is denoted as $Cat(e)$. Thus we need to design a learning algorithm F based on R^* to predict whether there is an *is-a* relation between e and c where $e \in E^*$ and $c \in Cat(e)$. In this way, we can utilize an existing taxonomy to harvest new *is-a* knowledge automatically.

3.2 Motivation of Our Method

The state-of-the-art method for Chinese *is-a* relation extraction is the word embedding based approach in (Fu et al., 2014). In their work, the projection parameters of a piecewise linear projection model learned from a Chinese thesaurus are used to identify *is-a* relations in encyclopedias. In this paper, we take a deeper look at the word vectors of hyponyms and hypernyms. As a preliminary experiment, we randomly sample *is-a* relations from a Wikipedia-based Chinese taxonomy (Li et al., 2015) and a Chinese thesaurus *CilinE*[2]. We compute the offsets of embedding vectors (i.e., $\vec{v}(x) - \vec{v}(y)$) where x is the hyponym of y. We have three observations, with examples shown in Table 1[3].

- **Observation 1.** For a fixed x, if y_1 and y_2 are hypernyms of different levels, it is likely that $\vec{v}(x) - \vec{v}(y_1) \not\approx \vec{v}(x) - \vec{v}(y_2)$. For example, "Country" is a high-level hypernym of "Japan" while "Asian Country" covers a narrow spectrum of entities.

- **Observation 2.** If $(x_1, instanceOf, y_1)$ and $(x_2, subClassOf, y_2)$ hold, it is likely that $\vec{v}(x_1) - \vec{v}(y_1) \not\approx \vec{v}(x_2) - \vec{v}(y_2)$. Although both *instanceOf* and *subClassOf* are *is-a* relations in a broad

[2]http://www.ltp-cloud.com/download/

[3]We use the l_2 norm of vector offsets to quantify the difference.

	Example with English Translation	Vector Offsets
True Positive	$\vec{v}$(日本) − $\vec{v}$(国家) ≈ $\vec{v}$(澳大利亚) − $\vec{v}$(国家) $\vec{v}$(Japan) − $\vec{v}$(Country) ≈ $\vec{v}$(Australia) − $\vec{v}$(Country)	$1.03 \approx 0.99$
Observation 1	$\vec{v}$(日本) − $\vec{v}$(国家) $\not\approx$ $\vec{v}$(日本) − $\vec{v}$(亚洲国家) $\vec{v}$(Japan) − $\vec{v}$(Country) $\not\approx$ $\vec{v}$(Japan) − $\vec{v}$(Asian Country)	$1.03 \not\approx 0.71$
Observation 2	$\vec{v}$(日本) − $\vec{v}$(国家) $\not\approx$ $\vec{v}$(主权国) − $\vec{v}$(国家) $\vec{v}$(Japan) − $\vec{v}$(Country) $\not\approx$ $\vec{v}$(Sovereign State) − $\vec{v}$(Country)	$1.03 \not\approx 1.32$
Observation 3	$\vec{v}$(日本) − $\vec{v}$(国家) $\not\approx$ $\vec{v}$(西瓜) − $\vec{v}$(水果) $\vec{v}$(Japan) − $\vec{v}$(Country) $\not\approx$ $\vec{v}$(Watermelon) − $\vec{v}$(Fruit)	$1.03 \not\approx 0.39$

Table 1: Examples of three observations.

sense, the *is-a* relations between (i) entities and classes, and (ii) classes and classes are different in semantics.

- **Observation 3.** For *is-a* pairs in two different domains (x_1, y_1) and (x_2, y_2), it is likely that $\vec{v}(x_1) - \vec{v}(y_1) \not\approx \vec{v}(x_2) - \vec{v}(y_2)$. This implies that *is-a* relations can be divided into more fine-grained relations based on their topics, such as politics, grocery, etc. A similar finding is also presented in (Fu et al., 2014).

These situations bring the challenges in modeling *is-a* relations correctly. Furthermore, *is-a* relations across different knowledge sources vary in characteristics. For example, *is-a* relations in a taxonomy are mostly *subClassOf* relations between concepts, while a large number of *is-a* relations derived from online encyclopedias are *instanceOf* relations, especially in the emerging domains, such as the Internet, new technologies, etc. The differences of *is-a* representations between knowledge sources suggest that a simple model trained on the taxonomy is not effective for *is-a* extraction from encyclopedias. The observations prompt us to take a two-stage process to deal with this problem. In the initial stage, we train piecewise linear projection models based on the taxonomy, aiming to learn prior representations of *is-a* relations in the embedding space. Next, we iteratively extract new *is-a* relations from user generated categories using models in the previous round and adjust our models accordingly. The characteristics of *is-a* relations of the target source are learned in a step-by-step manner.

3.3 Initial Model Training

We first train a *Skip-gram* model over a Chinese text corpus with over 1 billion words to obtain word embeddings. We randomly sample *is-a* relations from R^* as training data, denoted as $R^{'} \subset R^*$. In previous work, Mikolov et al. (2013b) and Fu et al. (2014) employ vector offsets and projection matrices to map words to their hypernyms, respectively. In this paper, we further combine the two relation representations together in the embedding space. For a pair (x_i, y_i), we assume a projection matrix $\mathbf{M}$ and an offset vector $\vec{b}$ map x_i to y_i in the form: $\mathbf{M} \cdot \vec{v}(x_i) + \vec{b} = \vec{v}(y_i)$.

To capture the multiple implicit language regularities in the training data, we follow the piecewise model training technique in (Fu et al., 2014). We first partition $R^{'}$ into K groups by K-means, denoted as $R^{'} = \bigcup_{k=1}^{K} C_k$ where C_k is the collection of *is-a* pairs in the kth cluster. Each pair $(x_i, y_i) \in R^{'}$ is represented as the vector offset $\vec{v}(x_i) - \vec{v}(y_i)$ for clustering. In each cluster, we assume *is-a* relations share the same projection matrix and vector offset. Therefore, we aim to learn K projection matrices and offset vectors as representations of *is-a* relations. For each cluster C_k $(k = 1, 2, \cdots, K)$, we aim to minimize the following objective function:

$$J(\mathbf{M}_k, \vec{b}_k; C_k) = \frac{1}{|C_k|} \sum_{(x_i, y_i) \in C_k} \|\mathbf{M}_k \cdot \vec{v}(x_i) + \vec{b}_k - \vec{v}(y_i)\|^2$$

where $\mathbf{M}_k$ and $\vec{b}_k$ are the projection matrix and the offset vector for C_k, learned via Stochastic Gradient Descent.

3.4 Iterative Learning Process

In the iterative learning process, we train *is-a* relation projection models on a series of dynamically enlarged training set $R^{(t)}$ ($t = 1, 2, \cdots, T$). The main idea is to update clustering results and prediction models iteratively in order to achieve a better generalization ability on the target knowledge source.

Initialization. We have two datasets: (i) the positive dataset $R^{(1)} = R^{'}$ and (ii) the unlabeled dataset $U = \{(x_i, y_i)\}$, which is created from user generated categories. Usually, we have $|U| \gg |R^{(1)}|$. Denote $C_k^{(t)}$ as the collection of *is-a* pairs, $\vec{c}_k^{(t)}$ as the cluster centroid, and $\mathbf{M}_k^{(t)}$ and $\vec{b}_k^{(t)}$ as model parameters in the kth cluster of the tth iteration. We set $C_k^{(1)} = C_k$, $\vec{c}_k^{(1)} = \frac{1}{|C_k|} \sum_{(x_i,y_i) \in C_k} \vec{v}(x_i) - \vec{v}(y_i)$, $\mathbf{M}_k^{(1)} = \mathbf{M}_k$ and $\vec{b}_k^{(1)} = \vec{b}_k$ as the initial values.

Iterative Process. For each iteration $t = 1, \cdots, T$, the models are updated as follows:

- **Step 1.** Randomly sample $\delta \cdot |U|$ instances from U and denote it as $U^{(t)}$ where δ is a sampling factor. For each $(x_i, y_i) \in U^{(t)}$, compute the cluster ID as $p_i = \arg\min_{k=1,\cdots,K} \|\vec{v}(x_i) - \vec{v}(y_i) - \vec{c}_k^{(t)}\|$. We first compute the difference $d^{(t)}(x_i, y_i)$ as $d^{(t)}(x_i, y_i) = \|\mathbf{M}_{p_i} \cdot \vec{v}(x_i) + \vec{b}_{p_i} - \vec{v}(y_i)\|$. The prediction result is $f_M^{(t)}(x_i, y_i) = I(d^{(t)}(x_i, y_i) < \epsilon)$ where $I(\cdot)$ is an indicator function and ϵ is a pre-defined threshold. We use $U_-^{(t)}$ to represent word pairs in $U^{(t)}$ predicted as "positive" in this step.

- **Step 2.** For each $(x_i, y_i) \in U_-^{(t)}$, predict the label (*is-a* or *not-is-a*) by pattern-based relation selection method (introduced in Section 3.5), denoted as $f_P^{(t)}(x_i, y_i)$. Define $U_+^{(t)} = \{(x_i, y_i) \in U_-^{(t)} | f_P^{(t)}(x_i, y_i) = 1\}$. Update the two datasets as follows: (i) $U = U \setminus U_+^{(t)}$ and (ii) $R^{(t+1)} = R^{(t)} \cup U_+^{(t)}$.

- **Step 3.** Denote the collection of *is-a* pairs in $U_+^{(t)}$ that belongs to the kth cluster as $U_k^{(t)}$. Update the cluster centroid $\vec{c}_k^{(t)}$ as follows:

$$\vec{c}_k^{(t+1)} = \vec{c}_k^{(t)} + \lambda \cdot \frac{1}{|U_k^{(t)}|} \sum_{(x_i,y_i) \in U_k^{(t)}} (\vec{v}(x_i) - \vec{v}(y_i) - \vec{c}_k^{(t)})$$

  where λ is a learning rate in $(0, 1)$ that controls the speed of cluster centroid "drift" over time. Re-assign the membership of clusters $C_k^{(t+1)}$ for each $(x_i, y_i) \in R^{(t+1)}$ based on new centroids.

- **Step 4.** For each cluster $C_k^{(t+1)}$, update model parameters by minimizing the objective function:

$$J(\mathbf{M}_k^{(t+1)}, \vec{b}_k^{(t+1)}; C_k^{(t+1)}) = \frac{1}{|C_k^{(t+1)}|} \sum_{(x_i,y_i) \in C_k^{(t+1)}} \|\mathbf{M}_k^{(t+1)} \cdot \vec{v}(x_i) + \vec{b}_k^{(t+1)} - \vec{v}(y_i)\|^2$$

  with the initial parameter values $\mathbf{M}_k^{(t+1)} = \mathbf{M}_k^{(t)}$ and $\vec{b}_k^{(t+1)} = \vec{b}_k^{(t)}$.

Model Prediction. After the training phase, given a pair (x_i, y_i) in the test set, our method predicts that x_i is the hyponym of y_i if at least one of the following conditions holds:

1. (x_i, y_i) is in the transitive closure of $R^{(T+1)}$ (based on *transitivity* property of *is-a* relations).

2. $f_M^{(T+1)}(x_i, y_i) = 1$ (based on final model prediction).

Discussion. The key techniques of the algorithm lie in two aspects: (i) combination of *semantic* and *syntactic-lexico is-a* extraction and (ii) incremental learning. The positive relation selection method in Step 2 can also be regarded as a variant of coupled learning (Carlson et al., 2010a). We ensure only when the results of semantic projection and pattern-based approach are consistent, these relations are added to our training set. Also, at each iteration, the model parameters are updated incrementally. By solving the recurrent formula, the update rule of centroids in Step 3 is equivalent to:

$$\vec{c}_k^{(T+1)} = (1-\lambda)^T \cdot \vec{c}_k^{(1)} + \lambda \cdot \sum_{t=1}^{T} \left(\frac{(1-\lambda)^{T-t}}{|U_k^{(t)}|} \cdot \sum_{(x_i,y_i) \in U_k^{(t)}} (\vec{v}(x_i) - \vec{v}(y_i) - \vec{c}_k^{(t)}) \right)$$

We can see that $\vec{c}_k^{(T+1)}$ is a weighted average of vector offsets of *is-a* relations added into the cluster, where the weight increases exponentially over time. With cluster assignments and prediction models updated, our models gradually fit the semantics of new *is-a* relations extracted from the unlabeled dataset.

3.5 Pattern-Based Relation Selection

We now introduce the pattern-based approach used in Step 2 of the iterative learning process. Although Chinese patterns for relation extraction can not guarantee high precision and coverage, we employ them as a "validation" source for model-based extraction results. The goal of this method is to select only a small portion of relations as $U_+^{(t)}$ from $U_-^{(t)}$ with high confidence to add to the training set $R^{(t)}$.

Previously, Fu et al. (2013) design several Chinese Hearst-style patterns manually for *is-a* extraction. In this paper, we collect a broader spectrum of patterns related to *is-a* relations, and categorize them into three types: "Is-A", "Such-As" and "Co-Hyponym". The examples are shown in Table 2[4]. We have the following two observations:

- **Observation 4.** If x_i and y match an "Is-A" or "Such-As" pattern, there is a large probability that x_i is the hyponym of y. Let $n_1(x_i, y)$ be the number of matches for x_i and y in a text corpus.
- **Observation 5.** If x_i and x_j match a "Such-As" or "Co-Hyponym" pattern, there is a large probability that no *is-a* relation exists between x_i and x_j. Let $n_2(x_i, x_j)$ be the number of matches for x_i and x_j, and $n_2(x_i)$ be the number of matches for x_i and x^* where x^* is an arbitrary hyponym other than x_i.

Category	Examples	Corresponding English Translation
Is-A	x_i是一个y	x_i is a kind of y
	x_i是y之一	x_i is one of y
Such-As	y，例如x_i、x_j	y, such as x_i and x_j
	y，包括x_i、x_j	y, including x_i and x_j
Co-Hyponym	x_i、x_j等	x_i, x_j and others
	x_i和x_j	x_i and x_j

Table 2: Examples of Chinese hypernym/hyponym patterns.

In this algorithm, we utilize the prediction of projection models and Chinese hypernym/hyponym patterns jointly to decide which relations in $U_-^{(t)}$ should be added into $U_+^{(t)}$. For each $(x_i, y_i) \in U_-^{(t)}$, denote $PS^{(t)}(x_i, y_i)$ and $NS^{(t)}(x_i, y_i)$ as the positive and negative scores that indicate the level of confidence. We define the positive score based on model prediction and Observation 4:

$$PS^{(t)}(x_i, y_i) = \alpha \cdot \left(1 - \frac{d^{(t)}(x_i, y_i)}{\max_{(x,y) \in U_-^{(t)}} d^{(t)}(x, y)}\right) + (1-\alpha) \cdot \frac{n_1(x_i, y_i) + \gamma}{\max_{(x,y) \in U_-^{(t)}} n_1(x, y) + \gamma}$$

[4]In practice, there can be over two candidate hyponyms in "Such-As" and "Co-Hyponym" patterns. For simplicity, we only list two here, denoted as x_i and x_j.

where $\alpha \in (0,1)$ is a tuning weight to balance the two factors and γ is a smoothing parameter. For simplicity, we empirically set $\alpha = 0.5$ and $\gamma = 1$ in this paper. We define the negative score based on Observation 5 as follows:

$$NS^{(t)}(x_i, y_i) = \log \frac{n_2(x_i, y_i) + \gamma}{(n_2(x_i) + \gamma) \cdot (n_2(y_i) + \gamma)}$$

A high negative score between x_i and y_i means the strong evidence of the frequent co-occurrence of x_i and y_i in "Such-As" or "Co-Hyponym" patterns, where x_i and y_i are likely to be co-hyponyms. This indicates that there is a low probability of the existence of an *is-a* relation between them.

A bi-criteria optimization problem can be formed where positive and negative scores should be maximized and minimized simultaneously, which is hard to optimize. We further covert it into a positive score maximization problem with negative score constraints:

$$\begin{aligned} \max \quad & \sum_{(x_i,y_i)\in U_+^{(t)}} PS^{(t)}(x_i, y_i) \\ \text{s. t.} \quad & \sum_{(x_i,y_i)\in U_+^{(t)}} NS^{(t)}(x_i, y_i) < \theta, U_+^{(t)} \subset U_-^{(t)}, |U_+^{(t)}| = m \end{aligned}$$

where m is the size of $U_+^{(t)}$ and θ is used to constrain negative score limits. This problem is a special case of the *budgeted maximum coverage problem* (Khuller et al., 1999), which is NP-hard. Based on the proof in (Khuller et al., 1999), the objective function is *monotone* and *submodular*. Therefore, we design a greedy relation selection algorithm to solve this problem with the accuracy of $1 - \frac{1}{e}$, shown in Algorithm 1. Finally, for each $(x_i, y_i) \in U_+^{(t)}$, we make the prediction as: $f_P^{(t)}(x_i, y_i) = I((x_i, y_i) \in U_+^{(t)})$.

Algorithm 1 Greedy Relation Selection Algorithm

```
Initialize U_+^(t) = ∅;
while |U_+^(t)| < m do
    Select candidate is-a pair with largest PS: (x_i, y_i) = arg max_{(x_i,y_i)∈U_+^(t)} PS^(t)(x_i, y_i);
    Remove the pair from U_-^(t): U_-^(t) = U_-^(t) \ {(x_i, y_i)};
    if NS^(t)(x_i, y_i) + Σ_{(x,y)∈U_+^(t)} NS^(t)(x, y) < θ then
        Add the pair to U_+^(t): U_+^(t) = U_+^(t) ∪ {(x_i, y_i)};
    end if
end while
return Collection of is-a relations U_+^(t);
```

4 Experiments

In this section, we conduct comprehensive experiments to evaluate our method on publicly available datasets. We also compare it with state-of-the-art approaches to make the convincing conclusion.

4.1 Experimental Data

In the experiments, we use four datasets consisting of word pairs, and a large Chinese text corpus. The statistics of our datasets are summarized in Table 3.

Dataset	Positive	Negative	Unknown
Wiki Taxonomy	7,312	-	-
Unlabeled Set	-	-	78,080
Validation Set	349	1,071	-
Test Set	1,042	3,223	-

Table 3: Datasets summarization.

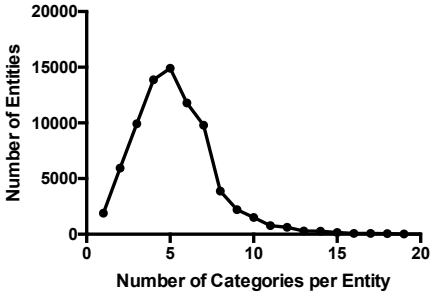


Figure 1: Unlabeled data statistics.

To learn word embeddings, we crawl Web pages from *Baidu Baike* and extract the contents to form a Chinese text corpus, consisting of 1.088B words. We use the open source toolkit *Ansj*[5] for word segmentation. Finally, we train a *Skip-gram* model to obtain 100-dimensional embedding vectors of 5.8M words. We calculate statistics for the pattern-based method in Section 3.5 using the same corpus.

The taxonomy data is obtained from the authors (Li et al., 2015), which consists of 1.3M *is-a* relations derived from Chinese Wikipedia. We use 7,312 *is-a* relations sampled from the taxonomy to train the initial projection models. To construct the unlabeled set, we randomly sample 0.1M entities from our *Baidu Baike* data, filter out entities without user generated categories and extract 78K <entity, category> pairs. The distribution of the number of categories per entity is illustrated in Figure 1.

To our knowledge, the only publicly available dataset for evaluating Chinese *is-a* relation extraction is published in (Fu et al., 2014), containing 1,391 *is-a* relations and 4,294 unrelated entity pairs. We use it to evaluate our method by splitting the dataset into 1/4 for validation and 3/4 for testing randomly.

4.2 Evaluation of Our Method

To tune the parameters of our method, we first run the K-means algorithm several times and train projection models. When we set the cluster number $K = 10$, our initial model achieves the best performance with a 73.9% F-measure. We also vary the value of parameter ϵ from 0.5 to 2 and find that the highest F-measure is achieved when $\epsilon = 1.05$.

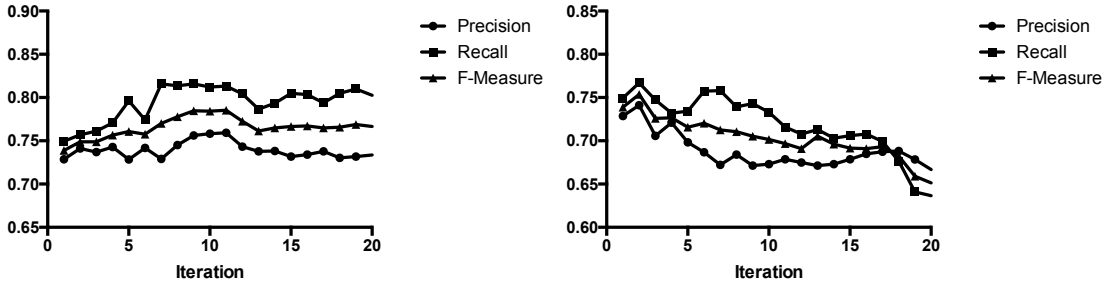
We report the performance of our method in 20 iterations to illustrate the effectiveness of the iterative process. We tune the parameters on the validation set and finally set $\delta = 0.2$, $\lambda = 0.5$ and add 500 new *is-a* relations into the training set in each iteration. In Figure 2(a), these new *is-a* relations are selected based on Algorithm 1. The F-measure increases from 74.9% to 78.5% in the first 10 iterations, which shows that newly extracted *is-a* relations can be of help to boost the performance of our models. The F-measure slightly drops and finally keeps stable after 15 iterations with F-measure around 76.7%. The possible cause of the drop is that a few false positive pairs are still inevitably selected by Algorithm 1 and added to the training set. After manual checking of these pairs, the average accuracy is 98.8%. Some of the erroneous cases include <脂肪(Fat), 健康(Health)>, <萧亚轩(Elva Hsiao), 时尚(Fashion)>, <信息(Information), 科学(Science)>, etc. They express *topic-of* relations rather than is-a relations. The performance becomes stable because the newly selected *is-a* relations tend to be similar to ones already in the training set after a sufficient number of iterations. In Figure 2(b), we directly sample 500 word pairs that are predicted as "positive" into our training set. Despite the slight improvement in the first iteration, the performance drops significantly because a large number of false positive instances are added to the training set for projection learning.

4.3 Comparison with Previous Methods

We evaluate our method and previous methods on the test set. The results are shown in Table 4.

We first re-implement three corpus-based *is-a* relation extraction methods on the *Baidu Baike* corpus. The pattern matching method for English *is-a* relations is originally proposed in (Hearst, 1992). For a Chinese corpus, we implement this method by employing Chinese Hearst-style patterns translated by Fu et al. (2013). The result shows that hand-craft patterns have low coverage for Chinese relation extraction

[5]http://nlpchina.github.io/ansj_seg/



(a) With the pattern-based relation selection method

(b) With no relation selection method

Figure 2: Performance of our method in each iteration.

because the language expressions are flexible. The automatic pattern detection approach in (Snow et al., 2004) improves the recall from 19.8% to 28.1%. However, the precision drops 28.9% because the syntactic parser for Chinese is still not sufficiently accurate, causing errors in feature extraction. The distributional similarity measure introduced in (Lenci and Benotto, 2012) has a 58.1% F-measure and is not effective for our task because the contexts of entities appeared in the free text are noisy. We also directly take the Chinese Wikipedia-based taxonomy (Li et al., 2015) to match *is-a* relations in the testing set. The result has a 98.5% precision but low recall due to the limited coverage of *is-a* relations in Chinese Wikipedia. The word embedding based approach in (Fu et al., 2014) achieves the highest F-measure 73.3% compared to all the previous methods. It shows the projection of word embeddings can model the semantics of Chinese *is-a* relations.

We now discuss our weakly supervised relation extraction method (WSRE) and its variants. In Table 4, *WSRE (Initial)* refers to the *is-a* extraction models trained in the initial stage. Although it is similar to (Fu et al., 2014), F-measure is improved by 2% because we consider both vector offsets and matrix projection in *is-a* representation learning, which is more precise. *WSRE (Random)*, *WSRE (Positive)* and *WSRE* employ the iterative learning process for *is-a* extraction. In *WSRE (Random)*, new *is-a* relations added to the training set are selected randomly from word pairs predicted as "positive" by our model. *WSRE (Positive)* considers only maximizing positive scores in relation selection, ignoring the effects of negative scores. *WSRE* is the full implementation of our method. Based on the results, the performance of *WSRE (Random)* decreases because of false positives in the training set. The F-measure of the latter two methods is increased by 2.3% and 3.3%, respectively, compared to *WSRE (Initial)*, which indicates that the proposed approach can improve prediction performance and generalization ability. *WSRE* outperforms *WSRE (Positive)* by 1% in F-measure, which shows the negative score constraint reduces the error rate in the relation selection process. Overall, our approach outperforms the state-of-the-art method (Fu et al., 2014) by 5.3% in F-measure. We further combine our method with the taxonomy (*WSRE+Taxonomy*) and achieve an 81.6% F-measure, which also has a better performance than Fu's method combined with the extension of a manually-built hierarchy, as shown in (Fu et al., 2014).

4.4 Error Analysis

We analyze errors occurred in our algorithm. The majority of the errors (approximately 72%) stems from the difficulty in distinguishing *related-to* and is-a relations. Some word pairs in the test set have very close semantic relations but are not strictly *is-a* relations. Such cases include <中药(Traditional Chinese medicine), 药草(Herb)>, <元帅(Marshal), 军事家(Strategist)>, etc. For example, most major components in traditional Chinese medicine are herbs, however, "药草(Herb)" is not a hypernym of "中药(Traditional Chinese medicine)" from a medical point of view. These cases are difficult to handle without additional knowledge. The errors in the iterative learning process (discussed in Section 4.2) also contribute to inaccurate prediction of this type.

The rest of the errors are caused by the inaccurate representation learning for fine-grained hypernyms. Take an example of the hyponym "兰科(Orchids)" in the test set, our algorithm recognizes that "植物(Plant)" is a correct hypernym, but it fails for "单子叶植物纲(Monocotyledon)". The possible causes

Method	Precision (%)	Recall (%)	F-Measure (%)
Previous Methods			
Hearst (Hearst, 1992)	96.2	19.8	32.8
Snow (Snow et al., 2004)	67.3	28.1	39.6
Taxonomy (Li et al., 2015)	**98.5**	25.4	40.4
DSM (Lenci and Benotto, 2012)	48.5	58.1	52.9
Embedding (Fu et al., 2014)	71.7	74.9	73.3
Our Method and Its Variants			
WSRE (Initial)	74.1	76.7	75.3
WSRE (Random)	69.0	75.7	72.2
WSRE (Positive)	75.4	80.1	77.6
WSRE	75.8	81.4	78.6
WSRE+Taxonomy	78.8	**84.7**	**81.6**

Table 4: Performance comparison between different methods.

is that "单子叶植物纲(Monocotyledon)" rarely appears in the corpus and is not well represented in the embedding space. We will improve learning of word and relation embeddings in the future.

5 Conclusion and Future Work

In this paper, we propose to extract Chinese *is-a* relations from user generated categories. Specifically, the task can be divided into two steps: initial model training and iterative learning. In the initial stage, word embedding based piecewise linear projection models are trained on a Chinese taxonomy to map entities to hypernyms. Next, an iterative learning process combined with a pattern-based relation selection algorithm is introduced to update models without human supervision. Experimental results show that this approach outperforms state-of-the-art methods. However, our experiments illustrate that free-text Chinese relation extraction still suffers from low coverage. We aim to address this issue by learning generalized pattern representations under the guidance of existing relations in the future.

Acknowledgements

This work is supported by the National Key Research and Development Program of China under Grant No. 2016YFB1000904.

References

Rahul Bhagat, Patrick Pantel, and Eduard H. Hovy. 2007. LEDIR: an unsupervised algorithm for learning directionality of inference rules. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 161–170.

Jingsheng Cai, Masao Utiyama, Eiichiro Sumita, and Yujie Zhang. 2014. Dependency-based pre-ordering for chinese-english machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 155–160.

Sharon A. Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *27th Annual Meeting of the Association for Computational Linguistics*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010a. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010b. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third International Conference on Web Search and Web Data Mining*, pages 101–110.

Daoud Clarke. 2012. A context-theoretic framework for compositionality in distributional semantics. *Computational Linguistics*, 38(1):41–71.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–610.

Oren Etzioni, Michael J. Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th international conference on World Wide Web*, pages 100–110.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 3–10.

Ruiji Fu, Bing Qin, and Ting Liu. 2013. Exploiting multiple sources for open-domain hypernym discovery. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1199–1209.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *14th International Conference on Computational Linguistics*, pages 539–545.

Wen Hua, Zhongyuan Wang, Haixun Wang, Kai Zheng, and Xiaofang Zhou. 2015. Short text understanding through lexical-semantic analysis. In *31st IEEE International Conference on Data Engineering*, pages 495–506.

Samir Khuller, Anna Moss, and Joseph Naor. 1999. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359–389.

Zornitsa Kozareva and Eduard H. Hovy. 2010. Learning arguments and supertypes of semantic relations using recursive patterns. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1482–1491.

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 543–546.

Hai-Guang Li, Xindong Wu, Zhao Li, and Gong-Qing Wu. 2013. A relation extraction method of chinese named entities based on location and semantic features. *Appl. Intell.*, 38(1):1–15.

Jinyang Li, Chengyu Wang, Xiaofeng He, Rong Zhang, and Ming Gao. 2015. User generated content oriented chinese taxonomy construction. In *Web Technologies and Applications - 17th Asia-PacificWeb Conference*, pages 623–634.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 746–751.

George A. Miller. 1995. Wordnet: a lexical database for english. *Communications of the Acm*, 38(11):39–41.

Rosa M. Ortega-Mendoza, Luis Villaseñor Pineda, and Manuel Montes-y-Gómez. 2007. Using lexical patterns for extracting hyponyms from the web. In *Proceedings of Mexican International Conference on Advances in Artificial Intelligence*, pages 904–911.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

Simone Paolo Ponzetto and Michael Strube. 2007. Deriving a large-scale taxonomy from wikipedia. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 1440–1445.

Alan Ritter, Stephen Soderland, and Oren Etzioni. 2009. What is this, anyway: Automatic hypernym discovery. In *Learning by Reading and Learning to Read, the 2009 AAAI Spring Symposium*, pages 88–93.

Erik F. Tjong Kim Sang. 2007. Extracting hypernym pairs from the web. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems 17, NIPS 2004*, pages 1297–1304.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, pages 697–706.

Idan Szpektor, Eyal Shnarch, and Ido Dagan. 2007. Instance-based evaluation of entailment rule acquisition. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, page 456–463.

Zhigang Wang, Juanzi Li, Shuangjie Li, Mingyang Li, Jie Tang, Kuo Zhang, and Kun Zhang. 2014. Cross-lingual knowledge validation based taxonomy derivation from heterogeneous online wikis. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 180–186.

Chengyu Wang, Ming Gao, Xiaofeng He, and Rong Zhang. 2015. Challenges in chinese knowledge graph construction. In *31st IEEE International Conference on Data Engineering Workshops*, pages 59–61.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Qili Zhu. 2012. Probase: a probabilistic taxonomy for text understanding. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 481–492.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 645–650.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014. Bilingually-constrained phrase embeddings for machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 111–121.

Huiwei Zhou, Long Chen, Fulin Shi, and Degen Huang. 2015. Learning bilingual sentiment word embeddings for cross-language sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 430–440.

Dynamic Generative model for Diachronic Sense Emergence Detection

Martin Emms
Dept of Computer Science
Trinity College, Dublin
Ireland
martin.emms@scss.tcd.ie

Arun Jayapal
Dept of Computer Science
Trinity College, Dublin
Ireland
jayapala@scss.tcd.ie

Abstract

As time passes words can acquire meanings they did not previously have, such as the 'twitter post' usage of 'tweet'. We address how this can be detected from time-stamped raw text. We propose a generative model with senses dependent on times and context words dependent on senses but otherwise eternal, and a Gibbs sampler for estimation. We obtain promising parameter estimates for positive (resp. negative) cases of known sense emergence (resp non-emergence) and adapt the 'pseudo-word' technique (Schütze, 1998) to give a novel further evaluation via 'pseudo-neologisms'. The question of ground-truth is also addressed and a technique proposed to locate an emergence date for evaluation purposes.

1 Introduction

It is widely noted that a single word can have several senses. The diachronic aspect of this is that the set of senses possessed by a word changes over time. (1a) and (1b) below illustrate this:

(1) (a) *she was a* **gay** *little soul, enjoying everything and always trilling with laughter* (1905)
(b) *applying heightened scrutiny to discrimination against* **gay** *men and lesbians* (1990)
(a′) *sie war ein* **Homosexuell** *kleine Seele, alles zu genießen und immer rollen vor Lachen*

(1a), from 1905, illustrates a 'being happy' sense of *gay*, while (1b), dating from 1990, illustrates a 'homosexual' sense, a sense which the word did not possess in 1905, and came to possess at some time since. The advent of a new sense for an existing form is sometimes called a *semantic* neologism (Tournier, 1985), in contrast to the simpler *formal* neologism, where simply a new form arrives (eg. *selfie*). The concern of this paper is to propose an unsupervised algorithm for detecting semantic neologisms, an algorithm which can be given time-stamped but plain-text examples of a particular word and detect whether (and when) the word gained a sense.

Information about such lexical change could be useful to NLP tasks. For example, if a SMT system is trained on data from particular times and is to be applied to texts from different times, either later or earlier, advance warning of sense changes could be of use. To illustrate, (1a′) gives the English→German translation via Google translate[1] of (1a), mistranslating the 1905 usage of *gay* as *Homosexuell*, probably due to the newer sense predominating in training data.

We will propose a diachronic sense model where a target's sense is conditioned on time and the context words are conditioned just on the target's sense, and not the time. We use the Google n-gram data set (Michel et al., 2011) which provides time-stamped data but *no* sense information and develop a Gibbs sampling algorithm (Gelfand and Smith, 1990) to estimate the parameters in an unsupervised fashion. We will show that the algorithm is able to provide an accurate date of sense emergence (true positives), and also to detect the absence of sense emergence when appropriate (true negatives). We adapt also the 'pseudo-word' technique first proposed by Schütze (1998) to give a further means of algorithm assessment. We also make a number of points concerning difficulties and possibilities evaluating such a sense-emergence system.

[1] Executed on Apr 13, 2016

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1362–1373, Osaka, Japan, December 11-17 2016.

2 A Diachronic Sense Model

Assume we have a corpus of D *time-stamped* occurrences of some particular target expression σ, with each time-stamp shared by many items. Let $\boldsymbol{w}$ be the sequence of words in a window around σ, and Y be its time-stamp, ranging from 1 to N. Assume σ exhibits K different senses and let S be the sense of a particular item. S will be treated as a hidden variable: the actual data provides values just for Y and $\boldsymbol{w}$. We will now propose a particular model of the joint probability $p(Y, S, \boldsymbol{w})$.

It may be factorised as $P(Y) \times P(S|Y) \times p(\boldsymbol{w}|S, Y)$ without loss of generality. We now make two independence assumptions: (a) that conditioned on S the words $\boldsymbol{w}$ are independent of Y, so $p(\boldsymbol{w}|S, Y) = P(\boldsymbol{w}|S)$ and (b) that conditioned on S the words are independent of each other, so $P(\boldsymbol{w}|S) = \prod_i p(w_i|S)$. With these assumptions the equation for a single data item is

$$P(Y, S, \boldsymbol{w}; \boldsymbol{\pi}_{1:N}, \boldsymbol{\theta}_{1:K}, \boldsymbol{\tau}_{1:N}) = P(Y; \boldsymbol{\tau}_{1:N}) \times P(S|Y; \boldsymbol{\pi}_{1:N}) \times \prod_i p(w_i|S; \boldsymbol{\theta}_{1:K}) \quad (2)$$

where we have also introduced explicit notations for model parameters: (i) for every time t, $\boldsymbol{\pi}_t$ is a length K vector of sense probabilities (ii) for every sense k, $\boldsymbol{\theta}_k$ is a length V vector of context word probabilities for target sense k — V is the size of the vocabulary encountered in all the data and (iii) for every t, τ_t is a 'time' probability reflecting simply the abundance of target σ at t.

The fact that for every time t there is a parameter $\boldsymbol{\pi}_t$ directly models that a sense's likelihood varies temporally; for an *established* sense this variation might be due to changes in the world and in people's concerns, but for an *emergent* sense, part of the variation represents genuine *language* change and for some k, for some early range of times, $\pi_t[k]$ should ideally be *zero*. Assumption (a) reflects an expectation that the vocabulary co-occuring with a particular *sense* is comparatively time independent. This particular time-independence is perhaps plausible but certainly is not absolute and its viability as an assumption can only be confirmed or disconfirmed by our later experiments.

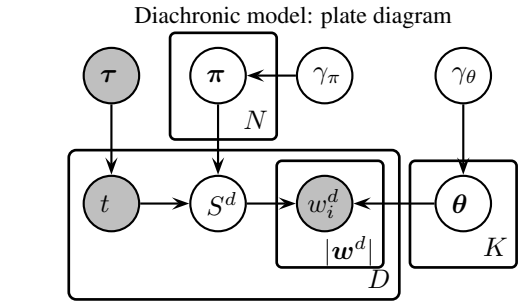
To develop the model further to the level of a corpus and incorporate parameter priors, let $\boldsymbol{t}^{1:D}, \boldsymbol{s}^{1:D}, \boldsymbol{w}^{1:D}$ be the values of Y, S and $\boldsymbol{W}$ on D items, and let the $\boldsymbol{\pi}_t$ sense probability vectors have a K-dimensional Dirichlet prior with parameter γ_π and the $\boldsymbol{\theta}_k$ word probability vectors have a V-dimensional Dirichlet prior with parameter γ_θ. We consider the joint probability $P(\boldsymbol{t}^{1:D}, \boldsymbol{s}^{1:D}, \boldsymbol{w}^{1:D}, \boldsymbol{\pi}_{1:N}, \boldsymbol{\theta}_{1:K}; \boldsymbol{\gamma}_\pi, \boldsymbol{\gamma}_\theta, \boldsymbol{\tau}_{1:N})$ and its formula under our assumptions is equation (3) in Figure 1, above which the model is depicted as a plate diagram.

From a generative perspective, the $\boldsymbol{\pi}_{1:N}$ and $\boldsymbol{\theta}_{1:K}$ are generated from N and K samplings and then used for *all* D data items. In actual data, the time-stamps and words are known, so for any fixed setting of $\boldsymbol{\gamma}_\pi, \boldsymbol{\gamma}_\theta$ there is a defined *posterior* distribution on $\boldsymbol{s}^{1:D}, \boldsymbol{\pi}_{1:N}, \boldsymbol{\theta}_{1:K}$. A Gibbs sampling algorithm (Gelfand and Smith, 1990) can be derived, generating a large set of samples of this $(D + N + K)$-tuple, representative of this posterior, from which *mean* values of the model parameters $\boldsymbol{\pi}_{1:N}$ and $\boldsymbol{\theta}_{1:K}$ can be derived. To arrive at the Gibbs sampler, sampling distributions for s^d, $\boldsymbol{\pi}_t$ and $\boldsymbol{\theta}_k$ are needed, in each case conditioned on all *other* parts of the sample tuple. The formulae for these sampling distributions are shown as (4 – 6) in Figure 1 : in these formulae $\mathbb{S}_t[k]$ is the number of data items with time-stamp t and sampled sense k and $\mathbb{V}_k[v]$ is the number of times word v occurs in data items with sampled sense k. The derivation of these formulae is relatively straightforward given well-known conjugacy properties of Dirichlet priors – Appendix A gives an outline derivation for $\boldsymbol{\pi}_t$. The sampling algorithm is given in pseudo-code in Figure 1.

2.1 Ground truth for semantic neologisms?

Given a large-scale, time-stamped and *sense*-labeled corpus for a target expression σ, it would be easy to determine a true emergence date — call it C_0 — at which a new sense for σ first departed from zero frequency (and continued to climb from zero). It has been noted (Lau et al., 2012; Cook et al., 2013) that such reference corpora do not exist, thus posing the question of what can serve as ground truth instead.

One option, sometimes adopted though far from ideal, is simple speaker intuition, which is subjective, of low temporal resolution and at best applicable to recent innovations. It is natural to consider dictionaries for something better. Form/meaning pairings are added to dictionaries at some particular time, so inspecting a series of dictionary editions, though labour intensive, can give a *first inclusion* date – call



$$P(\boldsymbol{t}^{1:D}, \boldsymbol{s}^{1:D}, \boldsymbol{w}^{1:D}, \boldsymbol{\pi}_{1:N}, \boldsymbol{\theta}_{1:K}; \boldsymbol{\gamma}_\pi, \boldsymbol{\gamma}_\theta, \boldsymbol{\tau}_{1:N}) = \prod_t \text{Dir}(\boldsymbol{\pi}_t; \boldsymbol{\gamma}_\pi) \times \prod_k \text{Dir}(\boldsymbol{\theta}_k; \boldsymbol{\gamma}_\theta) \times \prod_d [\, P(t^d; \boldsymbol{\tau}_{1:N}) \times P(s^d | t^d; \boldsymbol{\pi}_{1:N}) \times \prod_i P(w_i^d | s^d; \boldsymbol{\theta}_{1:K}) \,] \quad (3)$$

$$P(s^d) = \frac{\pi_{t,k} \prod_{i=1}^{|\boldsymbol{w}^d|} \theta_{k,w_i^d}}{\sum_{k'} \pi_{t,k'} \prod_{i=1}^{|\boldsymbol{w}^d|} \theta_{k',w_i^d}} \quad (4)$$

$$P(\boldsymbol{\pi}_t) = \text{Dir}(\boldsymbol{\pi}_t; \boldsymbol{\gamma}_\pi + \mathbb{S}_t) \quad (5)$$

$$P(\boldsymbol{\theta}_k) = \text{Dir}(\boldsymbol{\theta}_k; \boldsymbol{\gamma}_\theta + \mathbb{V}_k) \quad (6)$$

as in text, assume data as
context words $\boldsymbol{w}^{1:D}$, time-stamps $\boldsymbol{t}^{1:D}$
assume K is a supplied no of senses
create $S[D]$; `// data sense labels`
create $\mathbb{S}[T][K]$; `// see text`
create $\mathbb{V}[K][V]$; `// see text`
for *itr:=1 to no-iterations* **do**
  set $\mathbb{S}[t][k] = \mathbb{V}[k][v] = 0$ for all t,k,v
  for *d:=1 to D* **do**
    $k \sim P(s^d)$ see (4);
    $S[d] = k$;
    $\mathbb{S}[t^d][k]$ += 1; `// incr count`
    for *i:=1 to* $len(\boldsymbol{w}^d)$ **do**
      $\mathbb{V}[k][w_i^d]$ += 1; `// incr count`
    end
  end
  for *t:=1 to N* **do**
    $\boldsymbol{\pi}_t \sim P(\boldsymbol{\pi}_t)$ see (5);
  end
  for *k:=1 to K* **do**
    $\boldsymbol{\theta}_k \sim P(\boldsymbol{\theta}_k)$ see (6);
  end
end

Algorithm 1: Gibbs sampling estimation

Figure 1: From top left in anti-clockwise shows: plate digram for diachronic model, Gibbs sampler updates, and pseudo-code for Gibbs sampler

this D_0^i. The time resolution of this is low and the subtle criteria involved in inclusion decisions make it a non-ideal approximation of C_0 (Sheidlower, 1995; Simpson, 2000; Barnhart, 2007). Some researchers (Lau et al., 2012) use this, though we will not. More accessibly, an historically oriented dictionary (eg. the Oxford English Dictionary (OED)) strives to include the earliest known use of a word in a particular sense — the so-called *earliest citation*. If we call this D_0^c, it seems to makes sense to use D_0^c as a *lower* bound on C_0, and we will do this. D_0^c represents a first use, which might be followed by a long interlude before the usage is really taken up: the experiments in Section 3.2 will highlight examples of this.

We propose to use a different technique to establish C_0 more precisely. If there are words which it is intuitive to expect in the vicinity of a target word σ in the novel sense, and not in other senses, then by consulting a time-stamped corpus one should see the probability of finding these words in σ's context start to climb at a particular time. For example, *mouse* has come to have a 'computer pointing device' sense, and in this usage it is intuitive to expect words like *click*, *button*, *pointer* and *drag* in it's context. For any word w and target σ let $P_t(w|\sigma)$ be w's probability of occurring in σ's context in data from time t, and let $track_\sigma(w)$ to be the sequence these values. If when $track_{mouse}(w)$ is plotted for the above words, they all show a sharp increase at the *same* time point, this is good evidence that this is C_0 – the right-hand plot in Figure 3(a) is an example of this. This combines co-occurrence intuitions with corpus data, and does *not* rely on somewhat unreliable speaker intuitions of recency. To forestall any possible confusion, this procedure of inspecting the probabilities of words thought especially associated with a particular sense is *not* being advanced as a proposed unsupervised algorithm to locate sense emergences. It is advanced as a way to establish a ground-truth concerning emergence by which to evaluate our proposed unsupervised algorithm.

3 Data and Experiments

In the experiments reported below we use the Google N-gram dataset (Michel et al., 2011). This is a data-set based on Google's digitized publication holdings and it provides per-year counts of n-grams, for

Target	Years	Lines	New sense	OED	Tracks	GS-Date	$< 10\%$
mouse	1950-2008	910k	computer pointing device	1965	1983	1982	yes
gay	1900-2008	1253k	homosexual person	1922	1966	1969	yes
strike	1800-2008	5052k	industrial action	1810	1880	1866	yes
bit	1920-2008	7393k	basic unit of information	1948	1965	1954	no
paste	1950-2008	318k	duplicate text in computer edit	1975	1982	1981	yes
compile	1950-2008	689k	transform to machine code	1952	1966	1971	yes
surf	1950-2008	182k	exploring internet	1992	1994	1993	yes
boot	1920-2008	1285k	computer start up	1980	1980	1984	yes
rock	1920-2008	4136k	genre of music	1956	1965	1965	yes
stoned	1930-2008	79k	under drug influence	1952	1970	1979	no

Target	Years	Lines
ostensible	1800-2008	130k
present	1850-2008	56333k
cinema	1950-2008	305k
promotion	1930-2008	1681k
theatre	1950-2008	1125k
play	1950-2008	13726k
plant	1900-2008	8175k
spirit	1930-2008	11573k

Table 1: Google 5 gram dataset - the left table provides the information for targets that are neologisms while the right one has the targets for non-neologisms – see text for explanation of columns

$1 \leq n \leq 5$: so for any given n-gram, $\boldsymbol{x}$, and any year, t, it gives the total frequency[2] of occurrences of $\boldsymbol{x}$ across all books dated to year t. For a given target word σ we use the subset of all the data consisting of the *5*-grams that contain σ; we use the 5-grams as they provide most context around the target σ. For the target *mouse* the following is an example of a line of data from the corpus

```
Enter or click the mouse 1990 9 7
```

The first of the final three numbers is a year. The penultimate number is the number of occurrences of the 5-gram in all publications from that year – this is the significant count data for the algorithm. The final number is the number of publications from the year that contain the 5-gram, which is not significant for the algorithm.

For the experiments reported in Section 3.2 two sets of targets were chosen. The first set {*mouse, gay, strike, bit, paste, compile, surf, boot, rock, stoned*} are words which, relative to particular time periods, are known to exhibit sense emergence. The second set {*ostensible, present, cinema, promotion, theatre, play, plant, spirit*} are words which, relative to particular time periods, are thought *not* to exhibit sense emergence. Following Lau et al. (2012) the idea is that these should provide both positive and negative tests for the algorithm. Table 1 lists the targets. For each target, the 'Years' and 'Lines' columns give the range of years used and the total number of 5-gram lines of data for that year-range. For the positive targets the 'New sense' column gives an indication of the emergent sense and the next two columns give two kinds of reference dating information – see Section 2.1 – the OED first citation date and the 'tracks'-based date that is apparent from 'tracks' plots for words that are intuitively associated with the emergent sense (the right-hand plot in Figure 3(a) is an example). The 'GS date' column gives the emergence date inferred when the inference algorithm was run and will be discussed further in Section 3.2.

Before describing the experiments it is necessary to emphasize the Google n-gram data-set is best thought of as a frequency table giving per-year counts associated with 5-gram *types*. It is not really a corpus of text tokens. For brevity Algorithm 1 was formulated assuming that each data item represented a single target *token*. Any original publication token of a target σ could have contributed to several different 5-gram *type* counts (up to 5) but the data-set makes it impossible to know to what extent this is so. We therefore effectively treat each 5-gram data entry d listed with frequency of n_d as if it derives from n_d tokens of σ which contributed to no other 5-gram counts. This leads to changing the count increment operations in Algorithm 1 to add n_d rather than 1, that is, $\mathbb{S}[t][k]{+}{=}n_d$ and $\mathbb{V}[k][w]{+}{=}n_d$.

For all of the experiments sampling is done according to Algorithm 1, for 10000 iterations, with the first 1000 discarded as 'burn-in' samples and then means are determined for the model parameters $\boldsymbol{\pi}_{1:N}$ (sense-given-year) and $\boldsymbol{\theta}_{1:K}$ (word-given-sense) from the sampled values. The parameters $\boldsymbol{\gamma}_\pi$ and $\boldsymbol{\gamma}_\theta$ of the Dirichlet priors are set to have 1 in all positions to make them non-informative priors so uniform over all possible $\boldsymbol{\pi}_t$ and $\boldsymbol{\theta}_k$. The sampler is initialised with values $\boldsymbol{\pi}_{1:N}$ and $\boldsymbol{\theta}_{1:K}$ in the following way. Let P_{corp} be the observable corpus word probabilities in $\boldsymbol{w}^{1:D}$. Each $\boldsymbol{\theta}_k$ is set to $(1-\alpha)P_{corp} + \alpha P_{ran}$, where P_{ran} is a random word distribution and α is a mixing proportion, here set to 10^{-1}. The π_t are set to some shared set of sense probabilities. Thus initially the word distributions for each sense k are almost identical, and the sense distributions are the same at all times, so far from the neologism situation.

[2]They exclude 5-grams with total count < 40.

The procedure was implemented in C++. To obtain the code or data see `www.scss.tcd.ie/Martin.Emms/SenseDynamics`.

3.1 Experiments with 'pseudo'-neologisms

The '*pseudo-word*' technique was introduced in Schütze (1998) as a possible means to test unsupervised word-sense discrimination. It can be given a diachronic twist to furnish what might be called '*pseudo-neologisms*' in the following way. Relative to some period of time select two words, σ_1 and σ_2, both unambiguous, with σ_1 in use throughout the time period, but with σ_2 first emerging at some point, t_e, in the period. If the 5-grams for σ_1 and σ_2 are then all treated as examples of the fake word 'σ_1-σ_2' this functions as an artifical *semantic* neologism, manifesting σ_2's sense only from t_e onwards. Furthermore, if we say $f_t(\sigma_i)$ gives the true empirical probability of target σ_i in pooled σ_1,σ_2 data for time t, then ideally the outcome of inference when run for $K = 2$ should be that for each k, the trajectory of the $\pi_t[k]$ values is very similar to that for one of the $f_t(\sigma_i)$. We tested this, for the time-period 1850–2008, with 'ostensible' for σ_1 (present throughout), and 'supermarket', 'genocide' and 'byte' as possibilities for σ_2 (which emerged as new words over this time frame) and indeed obtained the desired correspondence between inferred $\pi_t[k]$ and empirical $f_t(\sigma_i)$ trajectories – Figure 2 shows the outcomes for the first two. For the first case the succession of $\pi_t[1]$ values matches closely the succession of f_t('supermarket') values, and in the second case the $\pi_t[0]$ values match the f_t('genocide') values. To get an insight into the inferred $\boldsymbol{\theta}_k$ values, we defined $gist(S)$ to be the top 20 words when ranked according to the ratio of $P(w|S)$ to $P_{corp}(w)$. For the apparently neologistic sense S, Figure 2 also shows $gist(S)$ and it can be seen that these sets of words seem very consistent with relevant parts of the pseudo-neologisms.

Thus on these pseudo-neologisms, the proposed model and algorithm has been successful, identifying an emerging 'sense' in an unsupervised fashion. Moving on from this first test of the algorithm, the next section considers outcomes on authentic words.

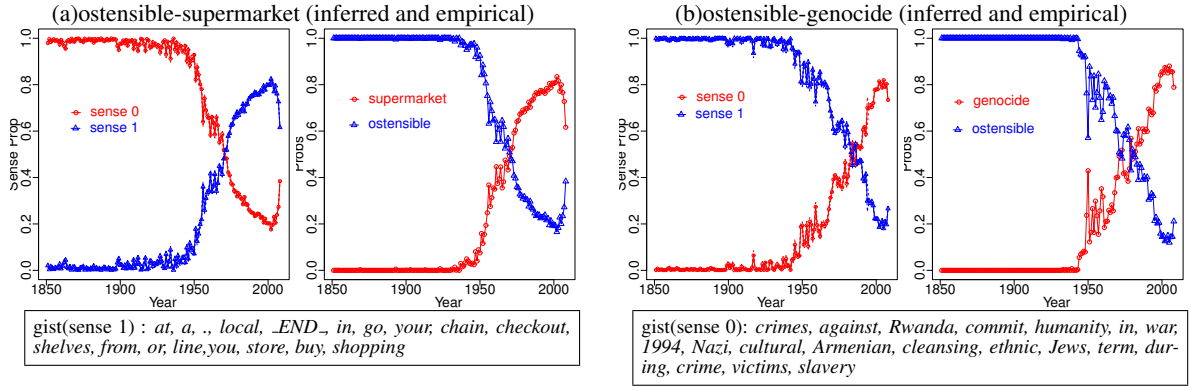


Figure 2: For (a-b), the left-hand plots show the inferred $\pi_t[k]$ sense parameters for a pseudo-neologism σ_1-σ_2, and right-hand plot shows the known σ_1 and σ_2 proportions. Below the plots are 'gist(S)' words associated to the apparent neologism sense – see text.

3.2 Experiments with genuine neologisms

Table 1 listed both targets expected to show sense emergence and targets expected to not show sense emergence. For several of the sense emergence targets, Figure 3(a-d) depicts various aspects of the outcomes. In each case the leftmost plot for a target σ shows for each k the succession of inferred $\pi_t[k]$ values – the sense-given-year values – plotted as a solid line[3]; the rightmost plot in each case is a 'tracks' plot (see Section 2.1), showing for some collection of words considered to be associated with the novel sense the succession of their probabilities of occuring in n-grams for the target σ, $P_t(w|\sigma)$. These are the basis for the 'tracks' column in Table 1.

mouse Figure 3(a): The algorithm was run looking for 3 sense variants on data between 1950 and 2008. The blue line for the $\pi_t[1]$ sequence in the left-hand plot shows a neologistic pattern, starting near 0 and

[3]also shown is the HPD interval around the mean as dotted lines

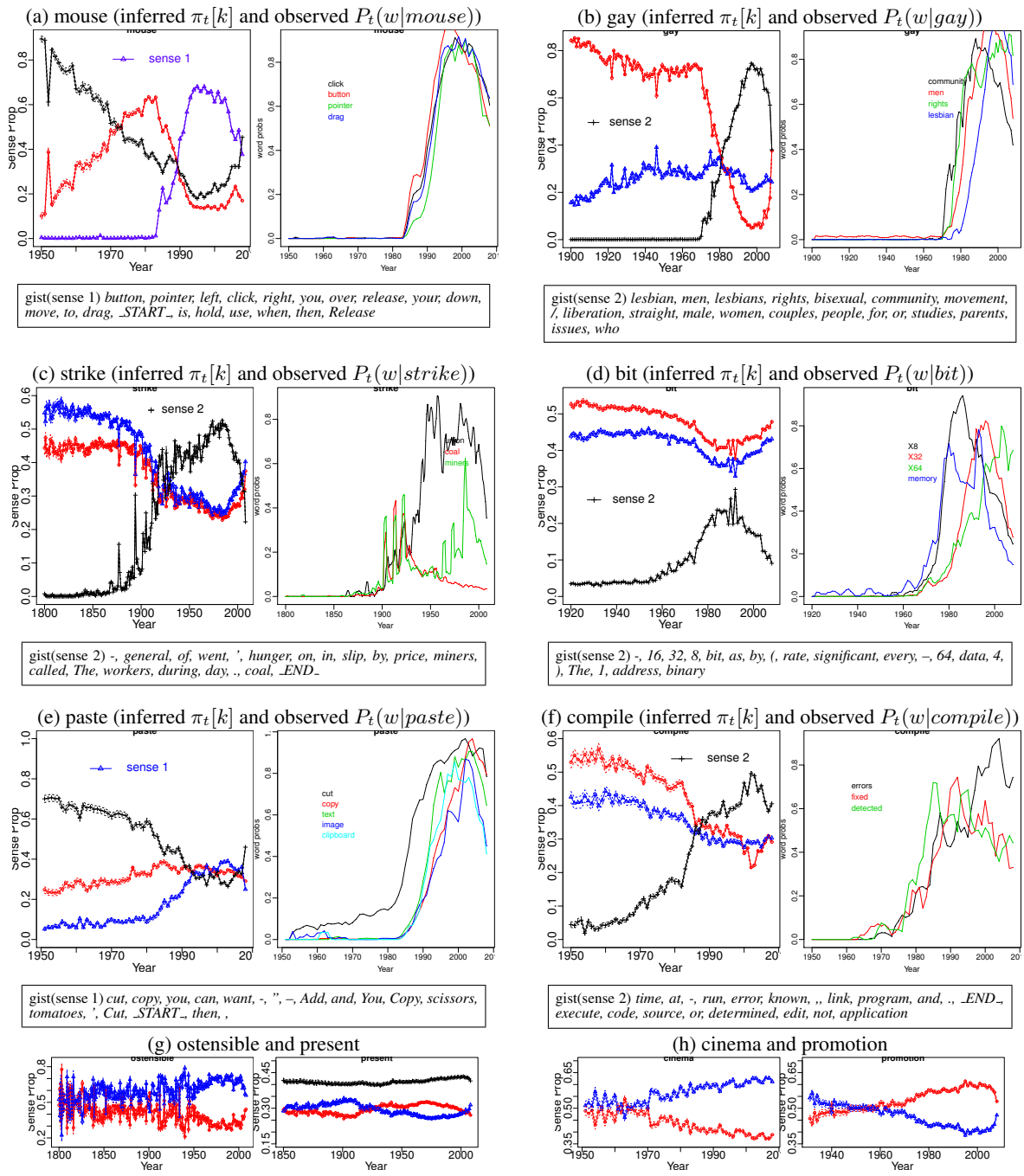


Figure 3: For (a-f), the left-hand plot shows the inferred $\pi_t[k]$ sense parameters, with the sense number S of the potential neologism labeled; the right-hand plot show probability 'tracks' for some words intuitively associated with the neologism (see text for further details). The box below the plots has top 20 $gist(S)$ words for the neologism sense S. (g-h) show the inferred $\pi_t[k]$ sense parameters for negative targets

departing from 0 around 1983. The 'tracks' plot also shows that several words intuitively associated with the neologistic sense, also drastically increase their probability conditioned on *mouse* around the same time. The 'GS-date' column of Table 1 gives the time t, if any, in a $\pi_t[k]$ sequence where it appears to depart from, and continues to climb from, zero. The '$< 10\%$' column records whether this agrees with the tracks-based date to within 10% of the time-span considered – which it does in this case. Notably in this case, the GS-based emergence date, though close to the tracks-based date, is more than 20 years *later* than the OED first citation date. The OED first citation comes from a research paper in 1965, but the mouse computer peripheral only became popular considerably later and it is not unexpected that the

date at which this use of the term *mouse* departed and continued to climb from zero in the n-grams books based data is substantially later. We take this to illustrate why simply taking the OED first citation date, D_0^c, as a gold standard for the true corpus emergence date, C_0, would be a mistake[4]. The box below the plots in Figure 3(a) tries to give some insight into the estimated $\boldsymbol{\theta}_1$ parameter concerning word-given-sense probabilities by showing the words belonging to $gist(1)$ (see definition in section 3.1). They seem mostly consistent with the 'pointing device' sense.

gay Figure 3(b): In this case the procedure was run on data from 1900 to 2008, for 3 senses. In the left-hand plot the black line, for the $\pi_t[2]$ sequence, shows sense emergence, appearing to depart from near zero first around 1969. The 'tracks' plots to the right seem to increase around around 1966. The OED first citation date of 1922 predates both considerably. The 'gist' words for $S = 2$ also seem mostly consistent the 'homosexual' sense.

Similar to *mouse* and *gay*, the detailed outcomes for *strike, bit, paste, compile* are shown in figures (c - f) with the procedures run on data for 3 senses. For space reasons, these details are not shown for *boot, surf, strike, rock* and *stoned* but Table 1 summarizes all outcomes: in each case the inferred date was later than the OED first citation date, and in all cases close to the tracks-based date, just missing the 10% margin in two cases.

Turning to the words which were *not* expected to exhibit an emergent sense, Figure 3(g-h) shows the plots of the inferred $\pi_t[k]$ sequences for the targets *ostensible, present, cinema* and *promotion*. None show a clear neologistic pattern, in line with expectations. Though the details are not shown in Figure 3 the same kind of outcome was found for the other negative targets listed in Table 1.

The value of K varied somewhat between the experiments. That the number of senses possessed by the different targets varies is somewhat to be expected and in some cases where a neologistic trend was less clear with n senses, it became clearer with $n + 1$. The automatic setting of this parameter remains an area for future work.

4 Comparisons to related work and conclusions

We have looked at the detection that a word has acquired the possibility to express a meaning which it could not hitherto (eg. *mouse* as pointing device). One can also look at senses themselves as changing over time, perhaps widening or narrowing, and there has been prior work addressing this issue (Sagi et al., 2009). We would like to treat this as a separate issue, though drawing a conclusive line between the two is tricky.

Concerning sense emergence specifically, it has to be stressed there is no strict quantitative state-of-the-art, because it is not the case that prior works share the same targets, use the same data, or address in the same way the tricky ground-truth issue (see Section 2.1). Bearing this in mind we have tried to organise the discussion below around major design options and papers that exemplify these.

There have been some proposals concerning sense emergence detection *without* modelling senses at all (Cook and Hirst, 2011; Kim et al., 2014). Though able to detect a difference between corpora from different eras, these systems tend to lack a capacity to pick out instances exemplifying a putative novel sense, which is arguably a desirable feature.

Moving on to systems which do involve some kind of modelling of senses, a noteworthy characteristic of many is that they often apply a WSI algorithm which is time-*unaware*. One design option is to *pool* all training data for the WSI phase, then assign likeliest senses to examples, and then to finally check for a correlation with time, such as a sense only being assigned after a particular time. Another design option is to *separate* the data into eras, perform independent WSI on each subset and then seek to consider how the sense representations from each era may (or may not) be *linked* to each other.

The *pooling* design option is exemplified in (Lau et al., 2012; Cook et al., 2013; Cook et al., 2014). Their time-unaware WSI system is based on LDA (Blei et al., 2003), and treats the I words of a context as generated from I topics, and then identifies a target's *sense* with the most frequent amongst the I topics of the context words. It is furthermore an HDP variant of LDA (Teh et al., 2004) in which the

[4] other than as a lower bound

number of topics is self-determined by the training process. The equating of senses with topics could be questioned (Wang et al., 2015) and also the self-determined sense number in their illustrative examples seems strikingly high (10), with many unintuitive components included. Rather than a year-by-year time-line, their data is time-stamped to just two time *eras*, $\mathcal{E}_1$ and $\mathcal{E}_2$ (eg. in one of their papers $\mathcal{E}_1$ is the late 20^{th} century (BNC) and $\mathcal{E}_2$ is 2007 (UkWaC)), and so they attempt a much lower resolution of emergence dating than we do. Their approach to ground-truth on sense emergence was different also, being that using D_0^i (dictionary *inclusion date*, mentioned in section 2.1), and so has some of the drawbacks that were noted there. As we did, they had both positive and negative targets. Without a time-line their evaluation cannot be a comparison of true and inferred emergence date and instead they count success as a tendency to place positives above negatives when ranked by a 'novelty' score: the max over k of the ratio of $\mathcal{E}_2$ to $\mathcal{E}_1$ frequency of assigned sense k. They obtain thus a ranking on their targets: { **domain**(116.2), **worm**(68.4), **mirror**(38.4), *guess*(16.5), **export**(13.8), *founder*(11.0), *cinema*(9.7), **poster**(7.9), *racism*(2.4), *symptom*(2.1) } (with positive targets in bold and negative in italics). As a possible generalisation of this score to a time-line, consider a 'novelty' score computed in the following way: from the sequence of $\pi_t[k]$ values, find 'min' and 'max' values and divide the temporally later value by the temporally earlier one, letting the novelty score be the max over k of this ratio. On our targets this gives a ranking: { **stoned**(10^5), **strike**(5442.7), **gay**(2791.1), **mouse**(1485.9), **surf**(156.7), **compile**(26.6), **bit**(10), **rock**(7.4), **boot**(7), *ostensible*(3.5), *plant*(1.89), *play*(1.8), *promotion*(1.4), *cinema*(1.3), *theatre*(1.1), *spirit*(1) }, separating the positive from the negative targets. Due to the data and target differences it would not make sense to compare these rankings. Earlier work by Rohrdantz et al. (2011) also instantiates the *pooling* option to exploit a time-unaware system. Their system was again LDA-based, their ground-truth approach was also D_0^i-based and their data was news articles between 1987 and 2007.

The *separate-then-link* strategy for deploying time-unaware WSI to nonetheless attempt to detect sense dynamics is exemplified in (Mitra et al., 2014; Mitra et al., 2015). The time-unaware WSI system in this case is a so-called 'Distributional Thesaurus' clustering approach (Rychlý and Kilgarriff, 2007; Biemann and Riedl, 2013), which starting from a word(type) co-occurrence graph where edges reflect co-occurrence, induces sets of words to represent a sense. Their data set consists of 'syntactic dependency n-grams' as produced by Goldberg and Orwant (2013) from the same digitised books as those from which the Google n-gram data is derived. They divide the entire time-line into eras $\mathcal{E}_1 \ldots \mathcal{E}_8$ of ever shortening duration but containing equal amounts of data (eg. $\mathcal{E}_2$ = 1909–53, $\mathcal{E}_7$ = 2002-05). For a given target, for each era they run their clustering to induce sense-representing word sets, and then they propose ways to link the clusters for $\mathcal{E}_i$, $\{s_1^i, \ldots s_m^i\}$ to the clusters of later era $\mathcal{E}_j$, $\{s_1^j, \ldots s_n^j\}$. Roughly speaking a cluster in $\mathcal{E}_j$ is judged a 'birth' (ie. sense emergence) if sufficiently few of its member words belong to the any of the clusters for the earlier era $\mathcal{E}_i$. In the paper they discuss outcomes concerning apparent 'births' when comparing the 1909-1953 and 2002-2005 eras. They do not test with respect to known positive and negative examples. Instead they apply the procedure to *all* words, obtain a very large set of candidate 'births', apply a relatively complex multi-stage filtering process to this and then on a randomly selected 48 cases from the filtered 'births' they find 60% are correct. Their approach to ground-truth concerning sense emergence (cf. Section 2.1) is somewhat varied but essentially was author intuition in (Mitra et al., 2014) and dictionary first citation D_0^c in (Mitra et al., 2015), though as we have noted this should only serve as lower bound[5].

Unlike these proposals, the experiments in this paper concern a model which is *not* time-unaware: the model has variables and parameters referring to time. Earlier versions of this idea were discussed in (Emms, 2013; Emms and Jayapal, 2014; Emms and Jayapal, 2015) though differing from the work presented here in number of respects (such as the estimation approach (EM), data used (text snippets via Google custom date search) and the targets considered (multiword expressions)). This aspect of including time explicitly in a probabilistic model seems to have been considered much less often than the above-mentioned essentially time-unaware approaches. The work of Wijaya and Yeniterzi (2011) is one example. They do not propose a sense-emergence detection algorithm per-se but do make some

[5]Without getting too lost in case-by-case details, it is worth noting that some seem incorrectly judged true 'births' relative to the eras considered, such as an assailant sense of *thug*, a calculus sense of *derivative*

analyses on the Google n-gram data to seek indicators of sense change. They sought to apply the *Topics Over Time* variant of *LDA* (Wang and McCallum, 2006), to do which they somewhat curiously collapse a year's worth of n-grams for a target into a *single* 'document' for that year. They found for example that for *gay*, training for 2 topics, there is a switch from a strong preference for one topic to preference for the other around 1970.

The recent work of Frermann and Lapata (2016) is a further example of a time-aware probabilistic model, in fact one having much in common with the model we have been discussing. They, as we have done, consider a generative model in which for a given time t a sense k is chosen, according to some discrete distribution[6] $\boldsymbol{\pi}_t$, and then, again as we have assumed, the context words in $\boldsymbol{w}$ are generated independently of each other. Whereas we have assumed that word choices are conditionally independent of the time t given the sense k, and so have for each sense k a parameter $\boldsymbol{\theta}_k$ of word probabilities, they do *not* assume this independence, and so for each *time* t and sense k have a parameter $\boldsymbol{\theta}_{tk}$ of word probabilities. The key further feature of their model is their use of *intrinsic Gaussian Markov Random Fields* (iGRMFs) to have priors which control how the distributions $\boldsymbol{\pi}_t$ and $\boldsymbol{\theta}_{tk}$ change over time: basically there is a precision hyper-parameter κ such that a *high* κ favours *small* changes in successive values. For the succession of $\boldsymbol{\theta}_{tk}$ values, they set κ to a high value, so that although $\boldsymbol{\theta}_{tk}$ does not have to be constant over time, only small variation is anticipated by the prior. The succession of π_t values is allowed greater variation. This use of iGMRF-based priors requires in its turn a more sophisticated Gibbs sampling approach to parameter estimation than that which we have used – which they achieve following ideas of Mimno et al. (2008). From the perspective of their model, our model is more or less what would be arrived at by (i) letting κ for $\boldsymbol{\theta}_{tk}$ tend to ∞, preventing any change of word-given-sense probabilites in successive times and (ii) letting κ for $\boldsymbol{\pi}_t$ tend to 0, allowing arbitrary change of sense-given-time probabilities. They evaluated their model in a variety of ways, the most comparable of which was to consider particular target words in the *Corpus of Historical American English* (Davies, 2010) with number of senses set to 10 and a time-resolution of 10-year time spans. As with the other papers already discussed, their use of different targets and a different data-set means again there is not the possibility at the moment of a quantitative comparison. In our work whilst we do not have a prior to encourage smooth change of the $\boldsymbol{\pi}_t$ values, nonetheless relatively smooth change *is* obtained, and sense emergence was successively detected in a number of cases, suggesting that for the n-gram data at least, the more complex system of Frermann and Lapata (2016) is not required. It may be of interest in future work to investigate to what extent this is dependent on the data-set used: the data-set they used contains ~100 times fewer occurrences for a given target per time-period compared to the n-gram data-set we have used and it could be that with less data the priors they propose become more necessary.

In conclusion we have proposed a simple generative model, with a $P(S|Y)$ term for time-dependent sense likelihood, and a $p(\boldsymbol{W}|S)$ term expressing that the context words are independent of time given the target's sense. The fact that intuitive outcomes were obtained on our pseudo-neologisms, and on some authentic cases of sense emergence and non-emergence is indicative at least that the model's assumptions are tolerable. It remains for future work involving further targets to test the limits of these assumptions. Amongst several possibiities for further investigation it would be of interest to reformulate the model to refer not just to plain words but rather to syntactic annotations, as well as to consider data sources representing other and more recent text types, such as social media posts.

Appendix A. Derivation of sampling formula for $\boldsymbol{\pi}_t$

For the sampling formula for $\boldsymbol{\pi}_t$ we need the conditional probability $P(\boldsymbol{\pi}_t|\boldsymbol{\pi}_{-(t)}, \boldsymbol{s}^{1:D}, \boldsymbol{w}^{1:D}, \boldsymbol{t}^{1:D}, \boldsymbol{\tau}_{1:N}, \boldsymbol{\theta}_{1:K}; \boldsymbol{\gamma}_\pi, \boldsymbol{\gamma}_\theta)$, where indexing by $-(t)$ is meant to indicate consideration of all indices *except* t. This conditional probability is given by

$$\frac{P(\boldsymbol{\pi}_t, \boldsymbol{\pi}_{-(t)}, \boldsymbol{s}^{1:D}, \boldsymbol{w}^{1:D}, \boldsymbol{t}^{1:D}, \boldsymbol{\tau}_{1:N}, \boldsymbol{\theta}_{1:K}; \boldsymbol{\gamma}_\pi, \boldsymbol{\gamma}_\theta)}{\int_{\boldsymbol{\pi}_t} P(\boldsymbol{\pi}_t, \boldsymbol{\pi}_{-(t)}, \boldsymbol{s}^{1:D}, \boldsymbol{w}^{1:D}, \boldsymbol{t}^{1:D}, \boldsymbol{\tau}_{1:N}, \boldsymbol{\theta}_{1:K}; \boldsymbol{\gamma}_\pi, \boldsymbol{\gamma}_\theta)}$$

Recalling the model formula (3) given in Figure 1 the numerator in this fraction is

[6] Adapting their notation to make things as comparable as possible: they have Φ^t rather than $\boldsymbol{\pi}_t$ and $\Psi^{t,k}$ rather than $\boldsymbol{\theta}_{tk}$.

$$\prod_d \left[\tau_{t^d} \times \pi_{t^d,s^d} \times \prod_{i=1}^{|\boldsymbol{w}^d|} \theta_{s^d,w_i^d} \right] \times \text{Dir}(\boldsymbol{\pi}_t;\boldsymbol{\gamma}_\pi) \times \prod_{-(t)} \text{Dir}(\boldsymbol{\pi}_{-(t)};\boldsymbol{\gamma}_\pi) \times \prod_{1:K} \text{Dir}(\boldsymbol{\theta}_k;\boldsymbol{\gamma}_\theta)$$

and the denominator differs only by the the integral over $\boldsymbol{\pi}_t$. $\boldsymbol{\pi}_t$ is involved in the $\text{Dir}(\boldsymbol{\pi}_t;\boldsymbol{\gamma}_\pi)$ term and in those parts of the product for d where you have $t_d = t$. Because of this most terms in the denominator can be taken outside the scope of the integral and then cancel with corresponding terms in the numerator. Because of this, the fraction can be written

$$\frac{\prod_{d:t_d=t} [\boldsymbol{\pi}_{t,s_d}] \times \text{Dir}(\boldsymbol{\pi}_t;\boldsymbol{\gamma}_\pi)}{\int_{\boldsymbol{\pi}_t} \left[\prod_{d:t_d=t} [\boldsymbol{\pi}_{t,s_d}] \times \text{Dir}(\boldsymbol{\pi}_t;\boldsymbol{\gamma}_\pi) \right]}$$

In the data items $\{d : t_d = t\}$ a variety of sense values have been sampled and the numerator can instead be expressed using $\mathbb{S}_{t,k}$, which counts the sampled sense values (see Section 2). Re-expressing the numerator in this way and using the definition of the Dirichlet (Heinrich, 2005), we get

$$\prod_k \boldsymbol{\pi}_{t,k}^{\mathbb{S}_{t,k}} \times \frac{1}{\beta(\boldsymbol{\gamma}_\pi)} \prod_k \boldsymbol{\pi}_{t,k}^{\boldsymbol{\gamma}_\pi[k]-1} = \frac{1}{\beta(\boldsymbol{\gamma}_\pi)} \prod_k \boldsymbol{\pi}_{t,k}^{\mathbb{S}_{t,k}+\boldsymbol{\gamma}_\pi[k]-1}$$

Hence the fraction can be written

$$\frac{\prod_k \boldsymbol{\pi}_{t,k}^{\mathbb{S}_{t,k}+\boldsymbol{\gamma}_\pi[k]-1}}{\int_{\boldsymbol{\pi}_t} [\prod_k \boldsymbol{\pi}_{t,k}^{\mathbb{S}_{t,k}+\boldsymbol{\gamma}_\pi[k]-1}]} = \frac{1}{\beta(\mathbb{S}_t + \boldsymbol{\gamma}_\pi)} \prod_k \boldsymbol{\pi}_{t,k}^{\mathbb{S}_{t,k}+\boldsymbol{\gamma}_\pi[k]-1}$$

where the last step uses the fact that in any Dirichlet $\text{Dir}(\boldsymbol{x}_{1:K};\boldsymbol{\alpha}_{1:K}) = \frac{1}{\beta(\boldsymbol{\alpha}_{1:K})} \prod_{k=1}^K x_k^{\alpha_k-1}$, the 'normalizing' constant $\beta(\boldsymbol{\alpha}_{1:K})$ is the integral of the main product term. Hence we finally obtain

$$P(\boldsymbol{\pi}_t|\boldsymbol{\pi}_{-(t)}, \boldsymbol{s}^{1:D}, \boldsymbol{w}^{1:D}, \boldsymbol{t}^{1:D}, \boldsymbol{\theta}_{1:K}; \boldsymbol{\gamma}_\pi, \boldsymbol{\gamma}_\theta) = \text{Dir}(\boldsymbol{\pi}_t; \boldsymbol{\gamma}_\pi + \mathbb{S}_t)$$

which is the sampling formula given earlier as (6). The derivation of the sampling formula for $\boldsymbol{\theta}_k$ is similar, and that for the discrete s^d is straightforward.

Acknowledgments

This research is supported by Science Foundation Ireland through the CNGL Programme (Grant 12/CE/I2267) in the ADAPT Centre (www.adaptcentre.ie) at Trinity College Dublin.

References

David Barnhart. 2007. A calculus for new words. *Dictionary*, 28:132–138.

Chris Biemann and Martin Riedl. 2013. Text: Now in 2d! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1(1):55–95, Apr.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. In *Journal of Machine Learning Research*, volume 3, pages 993–1022,, March.

Paul Cook and Graeme Hirst. 2011. Automatic identification of words with novel but infrequent senses. In *Proceedings of the 25th Pacific Asia Conference on Language Information and Computation (PACLIC 25)*, pages 265–274, Singapore, December.

Paul Cook, Jey Han Lau, Michael Rundell, Diana McCarthy, and Timothy Baldwin. 2013. A lexicographic appraisal of an automatic approach for detecting new word senses. In *Proceedings of eLex 2013*.

Paul Cook, Jey Han Lau, Diana McCarthy, and Timothy Baldwin. 2014. Novel word-sense identification. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, page 1624–1635. ACL, August.

Mark Davies. 2010. The corpus of historical american english: 400 million words, 1810-2009. available online at http://corpus.byu.edu/coha.

Martin Emms and Arun Jayapal. 2014. Detecting change and emergence for multiword expressions. In *Proceedings of the 10th Workshop on Multiword Expressions (MWE)*, pages 89–93, Gothenburg, Sweden. Association for Computational Linguistics.

Martin Emms and Arun Jayapal. 2015. An unsupervised em method to infer time variation in sense probabilities. In *ICON 2015: 12th International Conference on Natural Language Processing*, pages 266–271, Trivandrum, India, December.

Martin Emms. 2013. Dynamic EM in neologism evolution. In Hujun Yin, Ke Tang, Yang Gao, Frank Klawonn, Minho Lee, Thomas Weise, Bin Li, and Xin Yao, editors, *Proceedings of IDEAL 2013*, volume 8206 of *Lecture Notes in Computer Science*, pages 286–293. Springer.

Lea Frermann and Mirella Lapata. 2016. A Bayesian model of diachronic meaning change. *Transactions of the Association for Computational Linguistics*, 4:31–45.

Alan E. Gelfand and Adrian F. M. Smith. 1990. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410):398–409, jun.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of english books. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 241–247, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Gregor Heinrich. 2005. Parameter estimation for text analysis. Technical report, Fraunhofer Computer Graphics Institute.

Yoon Kim, Yi-I. Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. *CoRR*, abs/1405.3515.

Jey Han Lau, Paul Cook, Diana McCarthy, David Newman, and Timothy Baldwin. 2012. Word sense induction for novel sense detection. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012)*, pages 591–601, Avignon, France, April.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.

David Mimno, Hanna Wallach, and Andrew McCallum. 2008. Gibbs sampling for logistic normal topic models with graph-based priors. In *NIPS Workshop on Analyzing Graphs*.

Sunny Mitra, Ritwik Mitra, Martin Riedl, Chris Biemann, Animesh Mukherjee, and Pawan Goyal. 2014. That's sick dude!: Automatic identification of word sense change across different timescales. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, page 1020–1029. Association for Computational Linguistics, June.

Sunny Mitra, Ritwik Mitra, Suman Kalyan Maity, Martin Riedl, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2015. An automatic approach to identify word sense changes in text media across timescales. *Natural Language Engineering*, 21(5):773–798.

Christian Rohrdantz, Annette Hautli, Thomas Mayer, Miriam Butt, Daniel A. Keim, and Frans Plank. 2011. Towards tracking semantic change by visual analytics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 305–310. ACL, June.

Pavel Rychlý and Adam Kilgarriff. 2007. An efficient algorithm for building a distributional thesaurus (and other sketch engine developments). In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 41–44, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eyal Sagi, Stefan Kaufmann, and Brady Clark. 2009. Semantic density analysis: Comparing word meaning across time and phonetic space. In *Proceedings of the EACL 2009 Workshop on GEMS: GEometical Models of Natural Language Semantics*, page 104–111. Association for Computational Linguistics, March.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

Jesse T. Sheidlower. 1995. Principles for the inclusion of new words in college dictionaries. *Dictionaries*, 16:32–43.

John Simpson. 2000. Preface to the third edition of the oed. `public.oed.com/the-oed-today/preface-to-the-third-edition-of-the-oed`.

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2004. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101.

Jean Tournier. 1985. *Introduction descriptive à la lexicogénétique de l'anglais contemporain*. Champion-Slatkine.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: A non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 424–433, New York, NY, USA. ACM.

Jing Wang, Mohit Bansal, Kevin Gimpel, Brian D. Ziebart, and Clement T. Yu. 2015. A sense-topic model for word sense induction with unsupervised data enrichment. In Hwee Tou Ng, editor, *Transactions of the Association for Computational Linguistics*, page 59–71. Association for Computational Linguistics, January.

Derry Tanti Wijaya and Reyyan Yeniterzi. 2011. Understanding semantic change of words over centuries. In *Proceedings of the 2011 International Workshop on DETecting and Exploiting Cultural diversiTy on the Social Web*, DETECT '11, pages 35–40, New York, NY, USA. ACM.

Semi-supervised Word Sense Disambiguation with Neural Models

Dayu Yuan **Julian Richardson** **Ryan Doherty** **Colin Evans** **Eric Altendorf**
Google, Mountain View CA, USA
{dayuyuan,jdcr,portalfire,colinhevans,ealtendorf}@google.com

Abstract

Determining the intended sense of words in text – word sense disambiguation (WSD) – is a long-standing problem in natural language processing. Recently, researchers have shown promising results using word vectors extracted from a neural network language model as features in WSD algorithms. However, a simple average or concatenation of word vectors for each word in a text loses the sequential and syntactic information of the text. In this paper, we study WSD with a sequence learning neural net, LSTM, to better capture the sequential and syntactic patterns of the text. To alleviate the lack of training data in all-words WSD, we employ the same LSTM in a semi-supervised label propagation classifier. We demonstrate state-of-the-art results, especially on verbs.

1 Introduction

Word sense disambiguation (WSD) is a long-standing problem in natural language processing (NLP) with broad applications. Supervised, unsupervised, and knowledge-based approaches have been studied for WSD (Navigli, 2009). However, for *all-words* WSD, where all words in a corpus need to be annotated with word senses, it has proven extremely challenging to beat the strong baseline, which always assigns the most frequent sense of a word without considering the context (Pradhan et al., 2007a; Navigli, 2009; Navigli et al., 2013; Moro and Navigli, 2015). Given the good performance of published supervised WSD systems when provided with significant training data on specific words (Zhong and Ng, 2010), it appears lack of sufficient labeled training data for large vocabularies is the central problem.

One way to leverage unlabeled data is to train a neural network language model (NNLM) on the data. Word embeddings extracted from such a NNLM (often Word2Vec (Mikolov et al., 2013)) can be incorporated as features into a WSD algorithm. Iacobacci et al. (2016) show that this can substantially improve WSD performance and indeed that competitive performance can be attained using word embeddings alone.

In this paper, we describe two novel WSD algorithms. The first is based on a Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997). Since this model is able to take into account word order when classifying, it performs significantly better than an algorithm based on a continuous bag of words model (Word2vec) (Mikolov et al., 2013; Iacobacci et al., 2016), especially on verbs.

We then present a semi-supervised algorithm which uses label propagation (Talukdar and Crammer, 2009; Ravi and Diao, 2016) to label unlabeled sentences based on their similarity to labeled ones. This allows us to better estimate the distribution of word senses, obtaining more accurate decision boundaries and higher classification accuracy.

The best performance was achieved by using an LSTM language model with label propagation. Our algorithm achieves state-of-art performance on many SemEval all-words tasks. It also outperforms the most-frequent-sense and Word2Vec baselines by 10% (see Section 5.2 for details).

Organization: We review related work in Section 2. We introduce our supervised WSD algorithm in Section 3, and the semi-supervised WSD algorithm in Section 4. Experimental results are discussed in Section 5. We provide further discussion and future work in Section 6.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1374–1385, Osaka, Japan, December 11-17 2016.

2 Related Work

The development of large lexical resources, such as WordNet (Fellbaum, 1998) and BabelNet (Navigli and Ponzetto, 2012), has enabled knowledge-based algorithms which show promising results on all-words prediction tasks (Ponzetto and Navigli, 2010; Navigli et al., 2013; Moro and Navigli, 2015). WSD algorithms based on supervised learning are generally believed to perform better than knowledge-based WSD algorithms, but they need large training sets to perform well (Pradhan et al., 2007a; Navigli et al., 2007; Navigli, 2009; Zhong and Ng, 2010). Acquiring large training sets is costly. In this paper, we show that a supervised WSD algorithm can perform well with ~ 20 training examples per sense.

In the past few years, much progress has been made on using neural networks to learn word embeddings (Mikolov et al., 2013; Levy and Goldberg, 2014), to construct language models (Mikolov et al., 2011), perform sentiment analysis (Socher et al., 2013), machine translation (Sutskever et al., 2014) and many other NLP applications.

A number of different ways have been studied for using word embeddings in WSD. There are some common elements:

- Context embeddings. Given a window of text $w_{n-k}, ..., w_n, ..., w_{n+k}$ surrounding a focus word w_n (whose label is either known in the case of example sentences or to be determined in the case of classification), an embedding for the context is computed as a concatenation or weighted sum of the embeddings of the words $w_i, i \neq n$. Context embeddings of various kinds are used in both (Chen et al., 2014) and (Iacobacci et al., 2016).
- Sense embeddings. Embeddings are computed for each word sense in the word sense inventory (e.g. WordNet). In (Rothe and Schütze, 2015), equations are derived relating embeddings for word senses with embeddings for undisambiguated words. The equations are solved to compute the sense embeddings. In (Chen et al., 2014), sense embeddings are computed first as weighted sums of the embeddings of words in the WordNet gloss for each sense. These are used in an initial bootstrapping WSD phase, and then refined by a neural network which is trained on this bootstrap data.
- Embeddings as SVM features. Context embeddings (Iacobacci et al., 2016; Taghipour and Ng, 2015b), or features computed by combining context embeddings with sense embeddings (Rothe and Schütze, 2015), can be used as additional features in a supervised WSD system e.g. the SVM-based *IMS* (Zhong and Ng, 2010). Indeed Iacobacci et al. (2016) found that using embeddings as the only features in *IMS* gave competitive WSD performance.
- Nearest neighbor classifier. Another way to perform classification is to find the word sense whose sense embedding is closest, as measured by cosine similarity, to the embedding of the classification context. This is used, for example, in the bootstrapping phase of (Chen et al., 2014).
- Retraining embeddings. A feedforward neural network can be used to jointly perform WSD and adjust embeddings (Chen et al., 2014; Taghipour and Ng, 2015b).

In our work, we start with a baseline classifier which uses 1000-dimensional embeddings trained on a 100 billion word news corpus using Word2Vec (Mikolov et al., 2013). The vocabulary consists of the most frequent $1,000,000$ words, without lemmatization or case normalization. Sense embeddings are computed by averaging the context embeddings of sentences which have been labeled with that sense. To classify a word in a context, we assign the word sense whose embedding has maximal cosine similarity with the embedding of the context. This classifier has similar performance to the best classifier in (Iacobacci et al., 2016) when SemCor is used as a source of labeled sentences. The Word2Vec embeddings are trained using a bag of words model, i.e. without considering word order in the training context, and word order is also not considered in the classification context. In Section 3 we show that using a more expressive language model which takes account of word order yields significant improvements.

Semi-supervised learning has previously been applied successfully to word sense disambiguation. In (Yarowsky, 1995) bootstrapping was used to learn a high precision WSD classifier. A low recall classifier was learned from a small set of labeled examples, and the labeled set then extended with those sentences from an unlabeled corpus which the classifier could label with high confidence. The classifier was then retrained, and this iterative training process continued to convergence. Additional heuristics helped to maintain the stability of the bootstrapping process. The method was evaluated on a small data set.

In (Niu et al., 2005), a label propagation algorithm was proposed for word sense disambiguation and compared to bootstrapping and a SVM supervised classifier. Label propagation can achieve better performance because it assigns labels to optimize a *global* objective, whereas bootstrapping propagates labels based on *local* similarity of examples.

In Section 4 we describe our use of label propagation to improve on nearest neighbor for classification.

3 Supervised WSD with LSTM

Neural networks with long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997) make good language models which take into account word order (Sundermeyer et al., 2012). We train a LSTM language model to predict the held-out word in a sentence. As shown in Figure 1, we first replace the held-out word with a special symbol \$, and then, after consuming the remaining words in the sentence, project the h dimensional hidden layer to a p dimensional context layer, and finally predict the held out word with softmax. By default, the LSTM model has 2048 hidden units, 512 dimensional context layer and 512 dimensional word embeddings. We also studied other settings, see Section 5.2.2 for details. We train the LSTM on a news corpus of about 100 billion tokens, with a vocabulary of $1,000,000$ words. Words in the vocabulary are neither lemmatized nor case normalized.

Our LSTM model is different from that of Kgebck and Salomonsson (Kågebäck and Salomonsson, 2016). We train a LSTM language model, which predicts a held-out word given the surrounding context, with a large amount of unlabeled text as training data. The huge training dataset allows us to train a high-capacity model (2048 hidden units, 512 dimensional embeddings), which achieves high precision without overfitting. In our experiments, this directional LSTM model was faster and easier to train than a bidirectional LSTM, especially given our huge training dataset. Kgebck and Salomonsson's LSTM directly predicts the word senses and it is trained with a limited number of word sense-labeled examples. Although regularization and dropout are used to avoid overfitting the training data, the bidirectional LSTM is small with only $74 + 74$ neurons and 100 dimensional word embeddings (Kågebäck and Salomonsson, 2016). Because our LSTM is generally applicable to any word, it achieves high performance on *all-words WSD* tasks (see Section 5 for details), which is the focus of this paper. Kgebck and Salomonsson's LSTM is only evaluated on *lexical sample WSD* tasks of SemEval 2 and 3 (Kågebäck and Salomonsson, 2016).

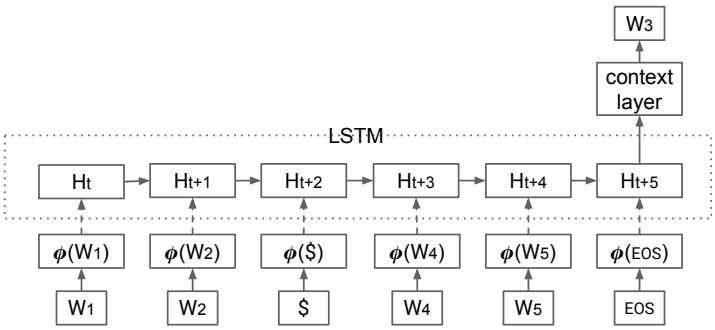


Figure 1: LSTM: Replace the focus word w_3 with a special symbol \$ and predict w_3 at the end of the sentence.

The behavior of the LSTM can be intuited by its predictions. Table 1 shows the top 10 words predicted by an LSTM language model for the word 'stock' in sentences containing various senses of 'stock'.

In our initial experiments, we computed similarity between two contexts by the overlap between their bags of predicted words. For example (Table 1) the top predictions for the query overlap most with the LSTM predictions for 'sense#1' —we predict that 'sense#1' is the correct sense. This bag of predictions, while easily interpretable, is just a discrete approximation to the internal state of the LSTM when predicting the held out word. We therefore directly use the LSTM's context layer from which the bag of predictions was computed as a representation of the context (see Figure 1). Given context vectors extracted from the LSTM, our supervised WSD algorithms classify a word in a context by finding the sense vector which has maximum cosine similarity to the context vector (Figure 2a). We find the sense vectors

id	sentence	top 10 predictions from LSTM	sense
1	Employee compensation is offered in the form of cash and/or *stock*.	cash, stock, equity, shares, loans, bonus, benefits, awards, equivalents, deposits	sense#1
2	The *stock* would be redeemed in five years, subject to terms of the company's debt.	bonds, debt, notes, shares, stock, balance, securities, rest, Notes, debentures	
3	These stores sell excess *stock* or factory overruns .	inventory, goods, parts, sales, inventories, capacity, products, oil, items, fuel	sense#2
4	Our soups are cooked with vegan *stock* and seasonal vegetables.	foods, food, vegetables, meats, recipes, cheese, meat, chicken, pasta, milk	sense#3
query	In addition, they will receive *stock* in the reorganized company, which will be named Ranger Industries Inc.	shares, positions, equity, jobs, awards, representation, stock, investments, roles, funds	?

Table 1: Top predictions of 'stock' in 5 sentences of different word senses

by averaging context vectors of all training sentences of the same sense. We observed in a few cases that the context vector is far from the held-out word's embedding, especially when the input sentence is not informative. For example, the LSTM language model will predict "night" for the input sentence "I fell asleep at [work]." when we hold out "work". Currently, we treat the above cases as outliers. We would like explore alternative solutions, e.g., forcing the model to predict words that are close to one sense vector of the held-out word, in further work. As can be seen in SemEval all-words tasks and Tables 6, this LSTM model has significantly better performance than the Word2Vec models.

4 Semi-supervised WSD

(a) nearest neighbor (b) semi-supervised WSD with label propagation

Figure 2: WSD classifiers. Filled nodes represent labeled sentences. Unfilled nodes represent unlabeled sentences.

The non-parametric nearest neighbor algorithm described in Section 3 has the following drawbacks:

- It assumes a spherical shape for each sense cluster, being unable to accurately model the decision boundaries given the limited number of examples.
- It has no training data for, and does not model, the sense prior, omitting an extremely powerful potential signal.

To overcome these drawbacks we present a semi-supervised method which augments the labeled example sentences with a large number of unlabeled sentences from the web. Sense labels are then propagated from the labeled to the unlabeled sentences. Adding a large number of unlabeled sentences allows the decision boundary between different senses to be better approximated.

A *label-propagation graph* consists of (a) vertices with a number of labeled seed nodes and (b) undirected weighted edges. Label propagation (LP) (Talukdar and Crammer, 2009) iteratively computes a distribution of labels on the graph's vertices to minimize a weighted combination of:

- The discrepancy between seed labels and their computed labels distributions.
- The disagreement between the label distributions of connected vertices.
- A regularization term which penalizes distributions which differ from the prior (by default, a uniform distribution).

We construct a graph for each lemma with labeled vertices for the labeled example sentences, and unlabeled vertices for sentences containing the lemma, drawn from some additional corpus. Vertices for sufficiently similar sentences (based on criteria discussed below) are connected by an edge whose weight is the cosine similarity between the respective context vectors, using the LSTM language model. To

classify an occurrence of the lemma, we create an additional vertex for the new sentence and run LP to propagate the sense labels from the seed vertices to the unlabeled vertices.

Figure 2 (b) illustrates the graph configuration. Spatial proximity represents similarity of the sentences attached to each vertex and the shape of each node represents the word sense. Filled nodes represent seed nodes with known word senses. Unfilled nodes represent sentences with no word sense label, and the ? represents the word we want to classify.

With too many edges, sense labels propagate too far, giving low precision. With too few, sense labels do not propagate sufficiently, giving low recall. We found that the graph has about the right density for common senses when we ranked vertex pairs by similarity and connected the pairs above the 95 percentile. This may still leave rare senses sparsely connected, so we additionally added edges to ensure that every vertex is connected to at least 10 other vertices. Our experiments (Table 9) show that this setting achieves good performance on WSD, and the performance is stable when the percentile ranges between 85 to 98. Since it requires running LP for every classification, the algorithm is slow compared to the nearest neighbor algorithm.

5 Experiments

We evaluated the LSTM algorithm with and without label propagation on standard SemEval all-words tasks using WordNet as the inventory. Our proposed algorithms achieve state-of-art performance on many SemEval all-words WSD tasks. In order to assess the effects of training corpus size and language model capacity we also evaluate our algorithms using the New Oxford American Dictionary (NOAD) inventory with SemCor (Miller et al., 1993) or MASC [1].

5.1 SemEval Tasks

In this section, we study the performance of our classifiers on Senseval2 (Edmonds and Cotton, 2001), Senseval3 (Snyder and Palmer, 2004), SemEval-2007 (Pradhan et al., 2007b), SemEval-2013 Task 12 (Navigli et al., 2013) and SemEval-2015 task 13 (Moro and Navigli, 2015) [2]. We focus the study on all-words WSD tasks. For a fair comparison with related works, the classifiers are evaluated on all words (both polysemous and monosemous).

Following related works, we use SemCor or OMSTI (Taghipour and Ng, 2015a) for training. In our LP classifiers, unlabeled data for each lemma consists either of 1000 sentences which contain the lemma, randomly sampled from the web, or all OMSTI sentences (without labels) which contain the lemma.

	Senseval2		Senseval3		SemEval7		SemEval7-Coarse		SemEval13
model	**all**	**n.**	**all**	**n.**	**all**	**n.**	**all**	**n.**	**n.**
IMS + Word2Vec (T:SemCor)	0.634	0.742	0.653	0.701	0.578	0.686			
IMS + Word2Vec (T:OMSTI)	0.683	0.777	0.682	0.741	0.591	0.715			
Taghipour and Ng (2015b)			0.682						
Chen et al. (2014)							0.826	**0.853**	
Weissenborn et al. (2015)				0.688		0.660		**0.855**	**0.728**
Word2Vec (T:SemCor)	0.678	0.737	0.621	0.714	0.585	0.673	0.795	0.814	0.661
LSTM (T:SemCor)	0.736	0.786	0.692	0.723	0.642	**0.723**	0.828	0.834	0.670
LSTM (T:OMSTI)	0.724	0.777	0.643	0.680	0.607	0.673	0.811	0.820	0.673
LSTMLP (T:SemCor, U:OMSTI)	0.739	0.797	0.711	0.748	**0.637**	0.704	**0.843**	0.834	0.679
LSTMLP (T:SemCor, U:1K)	0.738	0.796	**0.718**	**0.763**	0.635	0.717	0.836	0.831	0.695
LSTMLP (T:OMSTI, U:1K)	**0.744**	**0.799**	0.710	0.753	0.633	0.717	0.833	0.825	0.681

Table 2: F1 scores on SemEval all-words tasks. T:SemCor stands for models trained with SemCor. U:OMSTI stands for using OMSTI as unlabeled sentences in semi-supervised WSD. IMS + Word2Vec scores are from (Iacobacci et al., 2016)

Table 2 shows the Sem-Eval results. Our proposed algorithms achieve the highest all-words F1 scores except for Sem-Eval 2013. Weissenborn et al.(2015) only disambiguates nouns, and it outperforms our algorithms on Sem-Eval 2013 by 4%, but is ranked behind our algorithms on Senseval-3 and SemEval-7

[1] http://www.anc.org/MASC/About.html/

[2] We mapped all senses to WordNet3.0 by using maps in https://wordnet.princeton.edu/wordnet/download/current-version/ and http://web.eecs.umich.edu/ mihalcea/downloads.html

tasks with an F1 score more than 6% lower than our algorithms. Unified WSD (Chen et al., 2014) has the highest F1 score on Nouns (Sem-Eval-7 Coarse), but our algorithms outperform Unified WSD on other part-of-speech tags.

Settings For a fair comparison of Word2Vec and LSTM, we do not use pre-trained word-embeddings as in (Iacobacci et al., 2016), but instead train the Word2Vec and LSTM models on a 100 billion word news corpus [3] with a vocabulary of the most frequent 1,000,000 words. Our self-trained word embeddings have similar performance to the pre-trained embeddings, as shown in Table 2. The Word2Vec word vectors are of dimension 1024. The LSTM model has 2048 hidden units, and inputs are 512-dimensional word vectors. We train the LSTM model by minimizing sampled softmax loss (Jean et al., 2014) with Adagrad (Duchi et al., 2011). The learning rate is 0.1. We experimented with other learning rates, and observed no significant performance difference after the training converges. We also downsample frequent terms in the same way as (Mikolov et al., 2013).

Word2Vec vectors Vs. LSTM To better compare LSTM with word vectors we also build a nearest neighbor classifier using Word2Vec word embeddings and SemCor example sentences, Word2Vec (T:SemCor). It performs similar to IMS + Word2Vec (T:SemCor), a SVM-based classifier studied in (Iacobacci et al., 2016). Table 2 shows that the LSTM classifier outperforms the Word2Vec classifier across the board.

SemCor Vs. OMSTI Contrary to the results observed in (Iacobacci et al., 2016), the LSTM classifier trained with OMSTI performs worse than that trained with SemCor. It seems that the larger size of the OMSTI training data set is more than offset by noise present in its automatically generated labels. While the SVM classifier studied in (Iacobacci et al., 2016) may be able to learn a model which copes with this noise, our naive nearest neighbor classifiers do not have a learned model and deal less well with noisy labels.

Label propagation We use the implementation of DIST_EXPANDER (Ravi and Diao, 2016). We test the label propagation algorithm with SemCor or OMSTI as labeled data sets and OMSTI or 1000 random sentences from the web per lemma as unlabeled data. The algorithm performs similarly on the different data sets.

Table 3 shows the results of Sem-Eval 2015. The LSTM LP classifier with an LSTM language model achieves the highest scores on nouns and adverbs as well as overall F1. The LSTM classifier has the highest F1 on verbs.

algorithm	**all**	**n.**	**v.**	**adj.**	**adv.**
LIMSI	0.647				0.795
DFKI		0.703	0.577		
UNIBA				0.790	
BFS Baseline	0.663	0.667	0.551	**0.821**	0.825
Word2Vec (T:SemCor)	0.667	0.661	0.555	0.789	0.810
LSTM (T:SemCor)	0.721	0.713	**0.642**	0.813	0.845
LSTMLP (T:SemCor, U:1K)	**0.726**	**0.728**	0.622	0.813	**0.857**

Table 3: F1 Scores of SemEval-2015 English Dataset. The BFS baseline uses BabelNet first sense.

5.2 NOAD Eval

Many dictionary lemmas and senses have no examples in SemCor or OSTMI, giving rise to losses in all-words WSD when these corpora are used as training data. The above SemEval scores do not distinguish errors caused by missing training data for certain labels or inaccurate classifier. To better study the proposed algorithms, we train the classifiers with the New Oxford American Dictionary (NOAD) (Stevenson and Lindberg, 2010), in which there are example sentences for each word sense.

[3] The training corpus could not be released, but we have plans to open source the well-trained models

5.2.1 Word Sense Inventory

The NOAD focuses on American English and is based on the Oxford Dictionary of English (ODE) (Stevenson, 2010). It distinguishes between coarse (*core*) and fine-grained (*sub*) word senses in the same manner as ODE. Previous investigations (Navigli, 2006; Navigli et al., 2007) using the ODE have shown that coarse-grained word senses induced by the ODE inventory address problems with WordNet's fine-grained inventory, and that the inventory is useful for word sense disambiguation.

For our experiments, we use NOAD's core senses, and we also use lexicographer-curated example sentences from the Semantic English Language Database (SELD)[4], provided by Oxford University Press. We manually annotated all words of the English language SemCor corpus and MASC corpora with NOAD word senses in order to evaluate performance [5]. Table 4 shows the total number of polysemes (more than one core sense) and average number of senses per polyseme in NOAD/SELD (hereafter, NOAD), SemCor and MASC. Both SemCor and MASC individually cover around 45% of NOAD polysemes and 62% of senses of those polysemes.

		noun	verb	adj.	adv.
Number of polysemous lemmas in dictionary / corpus	NOAD	8097	2124	2126	266
	SemCor	2833	1362	911	193
	MASC	2738	1250	829	181
Sense count per polyseme	NOAD	2.46	2.58	2.30	2.47
	SemCor	1.44	1.69	1.49	1.84
	MASC	1.48	1.66	1.48	2.01

Table 4: NOAD polysemous lemma in NOAD, SemCor and MASC

Table 5 gives the number of labeled sentences of these datasets. Note that although NOAD has more labeled sentences than SemCor or MASC, the average numbers of sentences per sense of these datasets are similar. This is because NOAD has labeled sentences for each word sense, whereas SemCor (MASC) only covers a subset of lemmas and senses (Table 4). The last column of Table 5 shows that each annotated word in SemCor and MASC has an average of more than 4 NOAD corse senses. Hence, a random guess will have a precision around $1/4$.

	example count (in 1000's)					example count	number of candidate
dataset	all	n.	v.	adj.	adv	per sense	senses per example
NOAD	580	312	150	95	13	18.43	3.1
SemCor	115	38	57	11.6	8.6	14.27	4.1
MASC	133	50	57	12.7	13.6	17.38	4.2

Table 5: Number of examples in each dataset and the average sense count per example.

In the default setting, we use NOAD example sentences as labeled training data and evaluate on SemCor and MASC. We evaluate all polysemous words in the evaluation corpus.

5.2.2 LSTM classifier

We compare our algorithms with two baseline algorithms:

- Most frequent sense: Compute the sense frequency (from a labeled corpus) and label word w with w's most frequent sense.
- Word2Vec: a nearest-neighbor classifier with Word2Vec word embedding, which has similar performance to cutting-edge algorithms studied in (Iacobacci et al., 2016) on SemEval tasks.

Table 6 compares the F1 scores of the LSTM and baseline algorithms. LSTM outperforms Word2Vec by more than 10% over all words, where most of the gains are from verbs and adverbs. The results suggest that syntactic information, which is well modeled by LSTM but ignored by Word2Vec, is important to distinguishing word senses of verbs and adverbs.

[4] http://oxfordgls.com/our-content/english-language-content/

[5] https://research.google.com/research-outreach.html#/research-outreach/research-datasets

eval data		SemCor					MASC				
model	**train data**	**all**	**n.**	**v.**	**adj.**	**adv.**	**all**	**n.**	**v.**	**adj.**	**adv.**
Frequent Sense	SemCor						0.753	0.799	0.713	0.758	0.741
Frequent Sense	MASC	0.752	0.751	0.749	0.737	0.789					
Word2Vec	NOAD	0.709	0.783	0.657	0.736	0.693	0.671	0.790	0.562	0.724	0.638
Word2Vec	SemCor						0.692	0.806	0.592	0.754	0.635
Word2Vec	NOAD,SemCor						0.678	0.808	0.565	0.753	0.604
Word2Vec	MASC	0.698	0.785	0.619	0.766	0.744					
Word2Vec	NOAD,MASC	0.695	0.801	0.605	0.767	0.719					
LSTM	NOAD	0.786	0.796	0.782	0.781	0.784	0.786	0.805	0.772	0.776	0.786
LSTM	SemCor						0.799	0.843	0.767	0.808	0.767
LSTM	NOAD,SemCor						**0.812**	**0.846**	**0.786**	**0.816**	**0.798**
LSTM	MASC	0.810	0.825	0.799	0.809	**0.825**					
LSTM	NOAD,MASC	**0.821**	**0.834**	0.814	**0.818**	0.821					

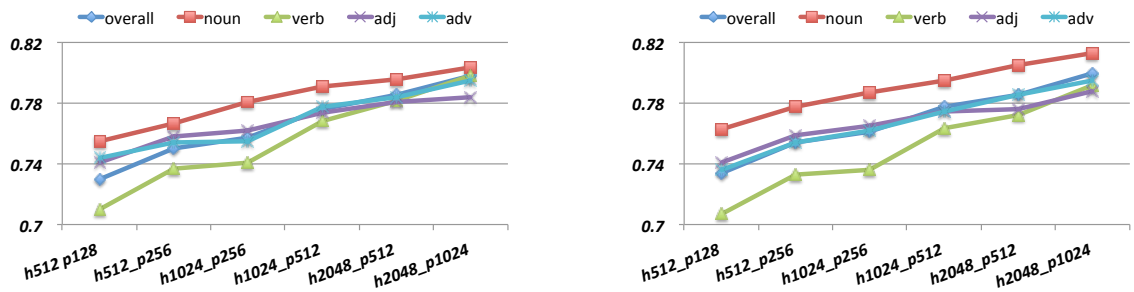
Table 6: F1 scores of LSTM algorithm in comparison with baselines

eval data		SemCor					MASC				
model	**train data**	**all**	**n.**	**v.**	**adj.**	**adv.**	**all**	**n.**	**v.**	**adj.**	**adv.**
LSTM	NOAD	0.769	0.791	0.759	0.751	0.672	0.780	0.791	0.768	0.780	0.726
LSTM	SemCor						0.656	0.663	0.668	0.643	0.581
LSTM	NOAD,SemCor						**0.796**	0.805	**0.790**	**0.794**	**0.742**
LSTM	MASC	0.631	0.653	0.606	0.617	0.600					
LSTM	NOAD,MASC	**0.782**	**0.803**	**0.774**	**0.761**	**0.688**					

Table 7: Macro F1 scores of LSTM classifier

Change of training data By default, the WSD classifier uses the NOAD example sentences as training data. We build a larger training dataset by adding labeled sentences from SemCor and MASC, and study the change of F1 scores in Table 6. Across all part of speech tags and datasets, F1 scores increase after adding more training data. We further test our algorithm by using SemCor (or MASC) as training data (without NOAD examples). The SemCor (or MASC) trained classifier is on a par with the NOAD trained classifier on F1 score. However, the macro F1 score of the former is much lower than the latter, as shown in Table 7, because of the limited coverage of rare senses and words in SemCor and MASC.

Change of language model capacity In this experiment, we change the LSTM model capacity by varying the number of hidden units h and the dimensions of the input embeddings p and measuring F1. Figure 3 shows strong positive correlation between F1 and the capacity of the language model. However, larger models are slower to train and use more memory. To balance the accuracy and resource usage, we use the second best LSTM model ($h = 2048$ and $p = 512$) by default.



(a) eval on SemCor

(b) eval on MASC

Figure 3: F1 scores of LSTM models with different capacity: h is the number of hidden units; p is the embedding dimension.

5.2.3 Semi-supervised WSD

We evaluate our semi-supervised WSD classifier in this subsection. We construct the graph as described in Section 4 and run LP to propagate sense labels from the seed vertices to the unlabeled vertices. We evaluate the performance of the algorithm by comparing the predicted labels and the gold labels on eval nodes. As can be observed from Table 8, LP did not yield clear benefits when using the Word2Vec language model. We did see significant improvements, 6.3% increase on SemCor and 7.3% increase on MASC, using LP with the LSTM language model. We hypothesize that this is because LP is sensitive to the quality of the graph distance metric.

eval data		SemCor					MASC				
model	**train data**	**all**	**n.**	**v.**	**adj.**	**adv.**	**all**	**n.**	**v.**	**adj.**	**adv.**
Word2Vec LP	NOAD	0.642	0.733	0.554	0.705	0.725	0.643	0.752	0.524	0.726	0.664
LSTM LP	NOAD	0.822	0.859	0.800	0.817	0.816	0.831	0.865	0.806	0.825	0.821
LSTM LP	NOAD,SemCor						**0.872**	**0.897**	**0.852**	**0.865**	**0.868**
LSTM LP	NOAD,MASC	**0.873**	**0.883**	**0.870**	**0.858**	**0.874**					

Table 8: F1 scores of label propagation

Change of seed data: As can be seen in Table 8, LP substantially improves classifier F1 when the training datasets are SemCor+NOAD or MASC+NOAD. As discussed in Section 4, the improvement may come from explicitly modeling the sense prior. We did not see much performance lift by increasing the number of unlabeled sentences per lemma.

Change of graph density: By default, we construct the LP graph by connecting two nodes if their affinity is above 95% percentile. We also force each node to connect to at least 10 neighbors to prevent isolated nodes. Table 9 shows the performance of the LP algorithm by changing the percentile threshold. The F1 scores are relatively stable when the percentile ranges between 85 to 98, but decrease when the percentile drops to 80. Also, it takes longer to run the LP algorithm on a denser graph. We pick the 95 percentile in our default setting to achieve both high F1 scores and short running time.

	SemCor						MASC					
pos-tag	**98**	**95**	**90**	**85**	**80**	**70**	**98**	**95**	**90**	**85**	**80**	**70**
overall	**0.823**	0.822	**0.823**	0.818	0.813	0.800	0.827	0.831	**0.835**	0.830	0.824	0.806
n.	0.848	**0.859**	0.852	0.846	0.840	0.828	0.863	0.865	**0.868**	0.865	0.861	0.847
v.	**0.810**	0.800	0.803	0.797	0.792	0.778	0.800	**0.806**	**0.806**	0.799	0.794	0.769

Table 9: F1 scores of the LSTM LP trained on NOAD with varying graph density.

6 Conclusions and Future Work

In this paper, we have presented two WSD algorithms which combine (1) LSTM neural network language models trained on a large unlabeled text corpus, with (2) labeled data in the form of example sentences, and, optionally, (3) unlabeled data in the form of additional sentences. Using an LSTM language model gave better performance than one based on Word2Vec embeddings. The best performance was achieved by our semi-supervised WSD algorithm which builds a graph containing labeled example sentences augmented with a large number of unlabeled sentences from the web, and classifies by propagating sense labels through this graph.

Several unanswered questions suggest lines of future work. Since our general approach is amenable to incorporating any language model, further developments in NNLMs may permit increased performance. We would also like to better understand the *limitations* of language modeling for this task: we expect there are situations – e.g., in idiomatic phrases – where per-word predictions carry little information.

We believe our model should generalize to languages other than English, but have not yet explored this. Character-level LSTMs (Kim et al., 2015) may provide robustness to morphology and diacritics and may prove useful even in English for spelling errors and out of vocabulary words.

We would like to see whether our results can be improved by incorporating global (document) context and multiple embeddings for polysemous words (Huang et al., 2012).

Finally, many applications of WSD systems for nominal resolution require integration with resolution systems for named entities, since surface forms often overlap (Moro et al., 2014; Navigli and Ponzetto, 2012). This will require inventory alignment work and model reformulation, since we currently use no document-level, topical, or knowledge-base coherence features.

We thank our colleagues and the anonymous reviewers for their insightful comments on this paper.

References

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *EMNLP*, pages 1025–1035. Citeseer.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July.

Philip Edmonds and Scott Cotton. 2001. SENSEVAL-2: Overview. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5, Toulouse, France, July. Association for Computational Linguistics.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

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

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *ACL*. Citeseer.

S. Jean, K. Cho, R. Memisevic, and Y. Bengio. 2014. On Using Very Large Target Vocabulary for Neural Machine Translation. *ArXiv e-prints*, December.

M. Kågebäck and H. Salomonsson. 2016. Word Sense Disambiguation using a Bidirectional LSTM. *ArXiv e-prints*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Honza Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

George A Miller, Claudia Leacock, Randee Tengi, and Ross T Bunker. 1993. A semantic concordance. In *Proceedings of the workshop on Human Language Technology*, pages 303–308. Association for Computational Linguistics.

Andrea Moro and Roberto Navigli. 2015. Semeval-2015 task 13: multilingual all-words sense disambiguation and entity linking. *Proc. of SemEval*, pages 288–297.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Roberto Navigli, Kenneth C Litkowski, and Orin Hargraves. 2007. Semeval-2007 task 07: Coarse-grained English all-words task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 30–35. Association for Computational Linguistics.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. Semeval-2013 task 12: Multilingual word sense disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, volume 2, pages 222–231.

Roberto Navigli. 2006. Meaningful clustering of senses helps boost word sense disambiguation performance. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 105–112, Stroudsburg, PA, USA. Association for Computational Linguistics.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

Zheng-Yu Niu, Dong-Hong Ji, and Chew Lim Tan. 2005. Word sense disambiguation using label propagation based semi-supervised learning. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 395–402. Association for Computational Linguistics.

Simone Paolo Ponzetto and Roberto Navigli. 2010. Knowledge-rich word sense disambiguation rivaling supervised systems. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1522–1531, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sameer S Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007a. Semeval-2007 task 17: English lexical sample, SRL and all words. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 87–92. Association for Computational Linguistics.

Sameer S Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007b. Semeval-2007 task 17: English lexical sample, srl and all words. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 87–92. Association for Computational Linguistics.

Sujith Ravi and Qiming Diao. 2016. Large scale distributed semi-supervised learning using streaming approximation. In *AISTATS*.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1793–1803, Beijing, China, July. Association for Computational Linguistics.

Benjamin Snyder and Martha Palmer. 2004. The english all-words task. In *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Angus Stevenson and Christine A. Lindberg. 2010. New Oxford American Dictionary.

Angus Stevenson. 2010. Oxford Dictionary of English.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *INTERSPEECH*, pages 194–197.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Kaveh Taghipour and Hwee Tou Ng. 2015a. One million sense-tagged instances for word sense disambiguation and induction. *CoNLL 2015*, page 338.

Kaveh Taghipour and Hwee Tou Ng. 2015b. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 314–323, Denver, Colorado, May–June. Association for Computational Linguistics.

Partha Pratim Talukdar and Koby Crammer. 2009. New regularized algorithms for transductive learning. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09, pages 442–457, Berlin, Heidelberg. Springer-Verlag.

Dirk Weissenborn, Leonhard Hennig, Feiyu Xu, and Hans Uszkoreit. 2015. Multi-objective optimization for the joint disambiguation of nouns and named entities. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 596–605.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL*, ACLDemos '10.

Fast Gated Neural Domain Adaptation: Language Model as a Case Study

Jian Zhang, Xiaofeng Wu, Andy Way, Qun Liu
ADAPT Centre
School of Computing
Dublin City University, Ireland
jian.zhang,xiaofeng.wu,andy.way,qun.liu@adaptcentre.ie

Abstract

Neural network training has been shown to be advantageous in many natural language processing applications, such as language modelling or machine translation. In this paper, we describe in detail a novel domain adaptation mechanism in neural network training. Instead of learning and adapting the neural network on millions of training sentences – which can be very time-consuming or even infeasible in some cases – we design a domain adaptation gating mechanism which can be used in recurrent neural networks and quickly learn the out-of-domain knowledge directly from the word vector representations with little speed overhead. In our experiments, we use the recurrent neural network language model (LM) as a case study. We show that the neural LM perplexity can be reduced by 7.395 and 12.011 using the proposed domain adaptation mechanism on the Penn Treebank and News data, respectively. Furthermore, we show that using the domain-adapted neural LM to re-rank the statistical machine translation *n*-best list on the French-to-English language pair can significantly improve translation quality.

1 Introduction

One challenge which rises above others in natural language processing (NLP) is where application performance decreases when there are *dissimilarities* between the training and the testing environments. In research on domain adaptation, training data with the same style and topic (van der Wees et al., 2015) as the test data is often defined as in-domain (ID) data, with everything else called general-domain (GD) data. The ID data is often scarce and expensive to obtain, whereas the GD is plentiful and easy to access.

One approach to address the situation of scarce ID training data is through data selection. For example, using the perplexity difference between ID and GD can select data that is close to ID and away from GD (Moore and Lewis, 2010). The selected data can then be concatenated with ID data for training. However, the drawback of using data selection is a threshold needs to be set, which often requires many models to be trained and evaluated. The situation will worsen if neural networks are used since the computational cost is immense in neural network training, where models often require days to converge even with the help of GPU-accelerated computing. Thus, simply adapting previous domain adaptation techniques into the neural network framework may not be efficient or effective. Ideally, we want to have an adaptation approach which has the ability to learn knowledge from huge corpora at speed. The question that arises here is how to make use of large amounts of GD data but avoiding long training times.

In neural network training, words are represented as distributed representations, so-called "word vectors", which can be pre-trained or trained with a specific task in mind. Although a pre-trained word vector model is also learned with a neural network, the training can be very fast. Recent optimized work shows learning word vectors can process more than 100 billion tokens in one day on a single machine (Mikolov et al., 2013c). Another advantage of a pre-trained word vector model is its flexibility, as it can be used later for different task-specific models. Furthermore, the pre-trained and the task-specific word vector models have no functional difference. Accordingly, we think it is very natural to use them

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1386–1397, Osaka, Japan, December 11-17 2016.

together. In this paper, we propose to perform domain adaptation from the large pre-trained word vector models instead of the raw text, i.e. adapting from large pre-trained word vector into the task-specific one. In this approach, we can make use of huge GD corpora with little speed overhead. We can also adapt the richer GD word representations into ID training.

2 Background

2.1 Word Vectors

Word vectors are important building blocks in neural networks. While much work has been proposed (Hinton et al., 1986; Mikolov et al., 2013b), we focus on the most popular approach of Mikolov et al. (2013b), which uses a neural network to produce distributed word representations. Unlike other training algorithms, labeled data is not required. It uses context words as features to make predictions. Word vectors can capture linguistic regularities and similarities (Mikolov et al., 2013d) in the training corpus. For example, using vector operations "king" - "man" + "woman" can result in a vector which is close to the word "queen".[1]

2.2 Neural Language Model

In a nutshell, the Recurrent Neural Network (RNN) language model (LM) uses the previous words to estimate the probability of the next word. The simplest RNN LM consists of a look-up layer, a hidden layer with recurrent connections and an output layer. The input words are firstly taken by the look-up layer and converted into word vector representations. Then, the hidden layers project the word vectors into a context vector with the states of input histories maintained. The output layer is a *Softmax* function. It decodes the context vector and distributes probabilities over all words in the vocabulary. The word with the highest probability is then chosen as the predicted output.

For notational convenience, the look-up layer, the hidden layer and the output layer of RNN LM can be represented as in Equations (1), (2) and (3), respectively:

$$x_t = \textit{look-up}(s) \tag{1}$$

$$h_t = \textit{RNN}(x_t, h_{t-1}) \tag{2}$$

$$y(t) = Softmax(f(h_t)) \tag{3}$$

where x_t is the word vector representation of s, s is the input at time t, and *look-up* is the look-up layer. The hidden layer *RNN* is then applied on x_t and the previous hidden state h_{t-1} to obtain the current hidden state h_t. f is a function that can map the hidden state into a vocabulary size vector. $y(t)$ is the prediction, which is the distribution over all words in the vocabulary.

Early work on neural LMs used simpler networks, such as the feed-forward neural network (Bengio et al., 2003). Later work (Mikolov et al., 2010) used simple RNNs for the input sentences, which showed large improvements over neural LMs. However, the simple RNN suffers from the vanishing gradient problem (Bengio et al., 1994). The Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) or the more recently introduced gated recurrent unit (GRU) (Chung et al., 2014) use gates to control the information flow from previous words. Thus, LSTM and GRU are better at capturing long-term dependencies than simple RNNs, and are often chosen in practice.

2.3 Gated Recurrent Unit

We use the GRU as a case study in this paper. The GRU consists of an update gate and a reset gate, as in Equation (4):

$$\begin{aligned} u_t &= \sigma(W_u x_t + U_u h_{t-1} + b_u) \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \\ \tilde{h}_t &= tanh(W x_t + U(r_t \odot h_{t-1}) + b) \\ h_t &= (1 - u_t) \odot h_{t-1} + u_t \odot \tilde{h}_t \end{aligned} \tag{4}$$

[1]The example is taken from Mikolov et al. (2013d)

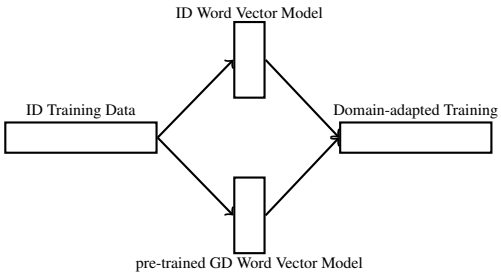


Figure 1: Adaptation Flow

where u_t is the update gate; r_t is the reset gate; $\tilde{h}_t$ is the candidate activation; $\odot$ is the element-wise multiplication operation, and h_t is the linear interpolated output between the previous hidden state h_{t-1} and the candidate activation. Intuitively, the update gate determines the interpolation weights between the previous hidden state h_{t-1} and the candidate activation, and the reset gate determines the information flow from previous hidden states. If the reset gate is set to be 1 and the update gate is set to be 0, the GRU is equivalent to the simple RNN. W_u, U_u, W_r, U_r, W and U are the weight parameters, and b_u, b_r and b are the bias values of the corresponding gates. σ is the sigmoid function and $tanh$ is the hyperbolic tangent function.

3 Adaptation Mechanisms

In this section, we describe several adaptation mechanisms proposed in this paper. In order to distinguish the input layers or hidden layers used in the network, we use Equations (1) and (2) to represent the ID training path. We use Equations (5) and (6) to notate the training path of GD:

$$x_t^* = \textit{look-up}^*_{pre-trained}(s) \tag{5}$$

$$h_t^* = \textit{RNN}^*(x_t^*, h_{t-1}^*) \tag{6}$$

where $\textit{look-up}^*_{pre-trained}$ is a pre-trained word vector model and static.[2] x_t^* is the word vector representation of input s, h_t^* is the hidden state and h_{t-1}^* is the previous hidden state of word s. For example, given input s from the ID training data, we can obtain two word vector representations (x_t and x_t^*). In addition, the two representations can then be fed into the corresponding hidden layer (*RNN* and *RNN**[3]). It is also worth mentioning that the proposed LMs are trained from scratch on ID training data, but adapting knowledge from the GD word vector model. Thus, the hidden state in Equation (6) is not strictly the "GD hidden state"; it uses the word embeddings from the GD pre-trained word vector model, but the inputs are still ID data.[4] Figure 1 shows the adaptation flow proposed in this paper, where the domain-adapted training is presented in Section 3.1, 3.2 and 3.3.

3.1 Adaptation on Word Vectors

The look-up table in the neural network contains word vector representations for all words in the vocabulary. We first propose to integrate the word vectors of ID and GD. The word vector from the ID look-up table contains the input word meanings with adaptation, whereas the GD look-up table contains richer word representations trained on a very large data set. We propose the following two approaches to adapt from the GD look-up table:

1. Word Vector Concatenation (WVC): we concatenate the word vectors obtained from ID and GD lookup-tables, as in Equation (7):

$$x_t^{WVC} = [x_t^*, x_t] \tag{7}$$

[2]By static, we mean the pre-trained word vector model is not updated during training.

[3]Depending on the adaptation methods that we will describe in this section, *RNN** is not always used. For example, when the adaptation is performed on word vectors (Section 3.1), *RNN** is not used.

[4]Note that Equation (1) and Equation (5) have the same input s.

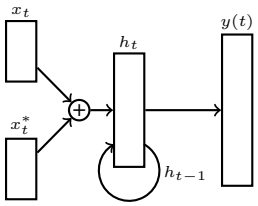


Figure 2: RNN LM with adaptation on word vectors, where $\oplus$ indicates the adaptation mechanisms described in Section 3.1.

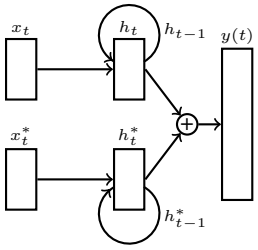


Figure 3: RNN LM with adaptation on context representations, where $\oplus$ indicates the adaptation mechanisms described in Section 3.2.

2. Word Vector Sum (WVS): we sum the two word vectors from ID and GD lookup-tables. In this approach, the two word vectors need to always have the same dimensionality, as in Equation (8):

$$x_t^{WVS} = x_t^* + x_t \tag{8}$$

When adapting, we replace the x_t in Equation (2) with x_t^{WVC} or x_t^{WVS}. Figure 2 is a graphical illustration of adaptation on word vectors.

3.2 Adaptation on Context Representations

Another approach is to delay the domain adaptation step until the context information is available. The RNN encapsulates the word vectors of the current words and previous context, and then produces a representation of the current context. Intuitively, if we maintain separate RNNs, where one uses the ID word representation and another one uses the GD word representation, there will be two pieces of context information available to us, namely contexts with the meaning of ID and GD. In this case, the following domain adaptation approaches can be taken:

1. Context Vector Concatenation (CVC): we can concatenate the two context vectors, as in Equation (9):

$$h_t^{CVC} = [h_t^*, h_t] \tag{9}$$

2. Weighted Context Vector Concatenation (WCVC): we can extend the CVC approach by applying a concatenation weight on h_t^* in Equation (6). Thus, the network can have simple control over the amount of the information flowing from GD, as in Equation (10):

$$h_t^{WCVC} = [Wh_t^*, h_t] \tag{10}$$

3. Context Vector Sum (CVS): we can also add the ID context vector and the GD context vector. We then have the compacted information from two domains to represent the context, as in Equation (11):

$$h_t^{CVS} = h_t^* + h_t \tag{11}$$

4. Weighted Context Vector Sum (WCVS): another approach is to apply a weight vector on the GD context vector. Thus the information from GD can be controlled before compacting, as in Equation (12):

$$h_t^{WCVS} = Wh_t^* + h_t \tag{12}$$

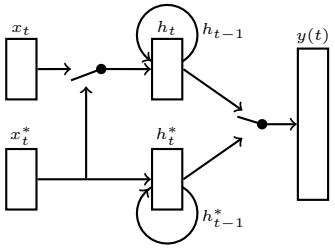


Figure 4: RNN LM with gated adaptation, where ● indicates the gates mechanisms described in Section 3.3.

Therefore, we replace the h_t in Equation (3) with h_t^{CVC}, h_t^{WCVC}, h_t^{CVS} or h_t^{WCVS} when adapting. Figure 3 is a graphical illustration of adaptation on context representations.

3.3 Gated Adaptation

However, directly applying adaptation on the context may not be efficient. The information adapted from GD is forced to be used by ID, i.e. the concatenation or sum operations. Furthermore, the adaptation operations are segmented. Ideally, we want to have an adaptation mechanism that can sequentially adapt the information from GD for the sequence inputs. Thus, we propose various gated domain adaptation mechanisms.

1. Word Vector Gating (WVG): in the WVG approach, we first design a gate to control the information flow from GD word vectors, as in Equation (13):

$$u_t^{WVG} = \sigma(W_u x_t + U_u h_{t-1} + W_u^* x_t^* + U_u^* h_{t-1}^* + b_u) \tag{13}$$

   where u_t^{WVG} is the adapted update gate on word vectors. It is computed using the known knowledge, i.e. x_t and x_t^* are the word vectors of the current word from ID and GD, respectively, and h_{t-1} and h_{t-1}^* are the previous context vectors of ID and GD, respectively. We then use a linear sum to combine the word vector representations (ID and GD), as in Equation (14):

$$x_t^{WVG} = u_t^{WVG} \odot x_t + (1 - u_t^{WVG}) \odot x_t^* \tag{14}$$

   where x_t^{WVG} is the domain-adapted word vector. Such an adaptation approach ensures that when the gate u_t^{WVG} tends to 1, we only use the ID word vector x_t, and when u_t^{WVG} tends to 0, the information from GD is fully cascaded to the current word vector.

   To use the domain-adapted word vector, we can simply replace the original x_t in Equation (2) with x_t^{WVG}.

2. Context Vector Gating (CVG): a similar gating mechanism can also be applied to the context vector, as in Equation (15):

$$r_t^{CVG} = \sigma(W_r x_t + U_r h_{t-1} + W_r^* x_t^* + U_r^* h_{t-1}^* + b_r) \tag{15}$$

   where r_t^{CVG} is the adapted update gate on the context vectors. We can also use the linear sum operation to combine the context vectors of ID and GD, as in Equation (16):

$$h_{t-1}^{CVG} = r_t^{CVG} \odot h_{t-1} + (1 - r_t^{CVG}) \odot h_{t-1}^* \tag{16}$$

   We then use the domain-adapted context vector h_{t-1}^{CVG} to replace the original context vector h_{t-1} in Equation (2).

3. Domain-Adapted GRU (DAGRU): the WVG and CVG mechanisms can also be used together. In this way, we can adapt both the word and context information of GD to the ID word and context vectors.

Figure 4 illustrates the gated adaptation mechanisms.

	Sentences	Tokens
Training	42,068	887,521
Validation	3,370	70,390
Test	3761	78,669

Table 1: Statistics of the Penn Treebank corpus.

3.4 Properties

It is also worth mentioning the following properties in our proposed adaptation mechanisms:

Property 1: One of the main differences between our proposed adaptation mechanisms and previous adaptation mechanisms is that we adapt word vector representations from a pre-trained word vector model trained using a large GD data set. However, previous work focused on adapting raw word(s) from GD, i.e. the data selection approach or the model combination approach. Our adaptation method is a more natural fit for neural network training. Furthermore, we think there is no contradiction between our approach and the previous data selection approach (Moore and Lewis, 2010), as data selection can always be performed before LM training. Note that our approach performs domain adaptation during LM training.

Property 2: Our approach also differs in the notation of GD data. Previous work defines a large corpus as the GD data, whereas our GD data here is the pre-trained GD word vector model. Thus, our adapted RNN LM model is still (only) trained using ID sentences. In practice, this is an efficient adaptation mechanism since there will be no extra training time brought by the additional training corpus.

Property 3: The pre-trained GD word vector model is static, which means it is not updated during training. This is because the pre-trained word vector model is obtained from a very large GD data set. The word vectors in such a model are not domain-specific. By keeping it static, we interpret it as a "knowledge database", and the knowledge should be consistent. Another practical reason for not updating the pre-trained GD word vector model is that fewer parameters need to be optimized in the network.

4 Experiments

4.1 Adaptation on Penn Treebank and News Corpus

The typical setting of domain adaptation is small amount of ID training data and large amount of DG training data. Accordingly, we choose to use the availability of widely known Penn Treebank (Marcus et al., 1993) portion of the Wall Street Journal corpus in our LM adaptation experiment.[5] The words outside the 10K vocabulary frequency list are mapped to the special *unk* token; sections 0-20 are used for training, and sections 21-22 are used for validation. We report the perplexity on data from sections 23-24. More detailed data statistics are summarized in Table 1. We use the pre-trained word vector Google *word2vec*[6] (Mikolov et al., 2013a) as the GD "look-up table". It is trained on about 100 billion words, and consists of 3 million words and phrases. The word vectors are 300-dimensional in the *word2vec* model.

In the experiments, our LM is trained with a single GRU hidden layer containing 600 hidden units. We uniformly initialize the weight parameters between [-0.1,0.1]. We set the maximum training iterations to 25. We set the initial learning rate to be 1, and then apply the learning rate with a decay factor of 0.5 after 13 iterations. The model is optimized using stochastic gradient descent with batch size of 20. We set the back-propagation through time to 40 time steps. The word embedding size in the loop-up layer is set to 600 for the baseline models. For a fair comparison, we use word embedding size 300 in the gated adaptation experiments since we are also using the pre-trained word vectors of 300.

Table 2 lists the perplexity results for the LM experiments. The baseline (KN5) model is a 5-gram LM trained using modified Kneser-Ney smoothing. It results in 148.007 and 141.186 perplexities on the validation and test data set, respectively. The baseline (*word2vec*) model is a neural LM, which uses the

[5]We download the data from `http://www.fit.vutbr.cz/~imikolov/rnnlm/simple-examples.tgz`

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

	Validation Set	Test Set
Baselines		
Baseline (KN5)	148.007	141.186
Baseline (*word2vec*)	121.871	117.730
Baseline (Standard)	92.983	89.295
Adaptation on Word Representations		
WVC	95.149	91.414
WVS	88.398	85.231
Adaptation on Context Representations		
CVC	90.337	86.168
WCVC	88.551	85.067
CVS	88.244	84.721
WCVS	90.293	86.679
Gated Adaptation		
WVG	90.937	87.853
CVG	90.301	86.832
DAGRU	**86.247**	**81.900**

Table 2: LM perplexity on Penn Treebank corpus.

	Sentences	Tokens
Training	181,108	4,691k
Validation	3,000	64k
Test	3,003	71k

Table 3: Statistics of the News corpus.

pre-trained *word2vec* model as the embedding layer. It can achieve 121.871 and 117.730 perplexities on the validation and test data set, respectively. The baseline (Standard) model is a neural LM, which is trained only on the ID data, it can achieve 92.983 and 89.295 perplexities on the validation and test data set, respectively. For the baseline (*word2vec*) model, we observe a sudden explosion in the evaluation perplexity. We experimentally set the learning rate with a decay factor of 0.5 after 4 iterations.

In the adaptation on word representations experiments, we found that summing up the word vectors in WVS can outperform the concatenation approach of WVC. Adding up the context representations in CVS is also more useful than concatenating the context representations in CVC. Thus, we can draw the conclusion that information from GD should be compressed (summed) into ID rather than using scattered (concatenated) representations. However, weighted vectors can be harmful to the sum approaches in WCVS. We think this is because the adapted representation is a newly computed vector after the sum operation, where the weight vectors are hidden behind the sum operation. Thus the model can be hard to optimize. In contrast, when applying weight vectors in the concatenation cases in WCVC, the adapted representation is still separable into domains. Thus the weights in the adapted model are easy to optimize. However, observing the experimental results, only a small positive impact can be observed when applying weighted vectors to the concatenation approaches. For example, approximately 1 perplexity point difference can be found between CVC and WCVC models. This indicates that the approach of using weight vectors for domain adaptation in neural network training is too simple.

We now move our focus to the News corpus. We test the DAGRU adaptation approach on the target side of the French-to-English News Commentary v10 corpus from the WMT2015 translation task. We use corpus newstest 2013 for evaluation and newstest 2014 for testing the trained LMs. More detailed data statistics are summarized in Table 3. Table 4 presents the LM perplexity differences between the baseline LM and the adapted LM. On the News training corpus, the adapted LM can also produce a better result than the baseline LM with a perplexity reduction of 12.

4.2 Statistical Machine Translation Re-ranking

In this experiment, we apply the DAGRU-adapted LMs trained in Table 4 on the statistical machine translation (SMT) *n*-best re-ranking task. We use the French-to-English News Commentary v10 and Europarl v7 corpus from the WMT2015 translation task to train our baseline SMT system. The newstest 2013 and 2014 data sets are used for tuning and testing for the SMT, respectively. The SMT baseline

System	Validation Set	Test Set
Baseline LM	94.341	109.420
DAGRU LM	85.551	**97.409**

Table 4: LM perplexity on News dataset.

System	Tuning	Test
Baseline	26.18	27.14
Baseline + baseline LM	26.71	27.65
Baseline + DAGRU LM	27.17	**27.96**

Table 5: BLEU scores of baseline SMT and re-ranked SMT.

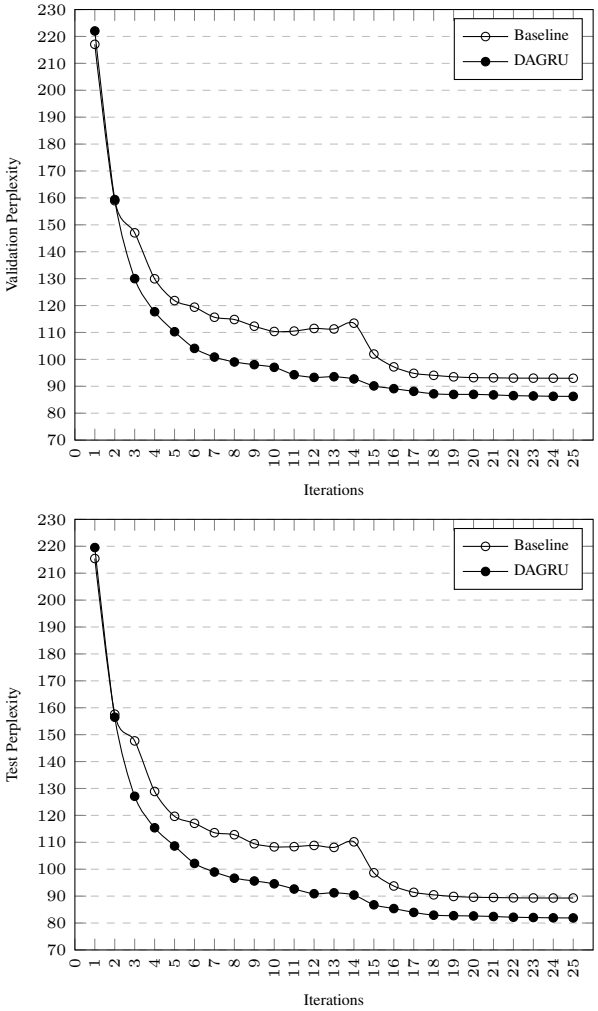


Figure 5: Language model coverage plot for baseline LM and DAGRU LM on the Penn Treebank data.

is trained using Moses (Koehn et al., 2007), with a lexicalized reordering model (Galley and Manning, 2008) and a 5-gram KenLM (Heafield, 2011) LM trained on the target side of the SMT training data.

The re-ranked SMT systems can both increase the baseline BLEU score significantly as seen in Table 5. Using the baseline LM for re-ranking, we can observe a 0.51 absolute improvement (1.8% relative). Using the DAGRU LM for re-ranking, we an obtain a 0.82 absolute (3% relative) improvement. Comparing the two re-ranked systems, we observe that using DAGRU LM for re-ranking can provide an additional increase of 0.31 (1.1% relatively) in BLEU score. The improvement is statistically significant (Koehn, 2004). The significance testing uses bootstrapping method at the level p = 0.05 level with 1,000 iterations.

	Validation Set	Test Set
Word Embedding = 50		
Baseline	104.465	101.285
SENNA (Collobert et al., 2011)	95.453	92.026
GloVe (glove_6b) (Pennington et al., 2014)	**95.292**	**91.251**
Word Embedding = 100		
Baseline	97.034	93.645
GloVe (glove_6b) (Pennington et al., 2014)	**89.521**	**85.330**
Word Embedding = 200		
Baseline	94.201	90.993
GloVe (glove_6b) (Pennington et al., 2014)	**86.638**	**82.762**
Word Embedding = 300		
Baseline	92.983	89.295
GloVe (glove_6b) (Pennington et al., 2014)	86.438	82.593
GloVe (glove_42b) (Pennington et al., 2014)	87.096	82.297
GloVe (glove_840b) (Pennington et al., 2014)	86.853	82.165
Google (word2vec)	**86.247**	**81.900**

Table 6: The DAGRU adaptation on different word vector models

4.3 Observations

There are many reasons to explain the efficiency of the DAGRU approach. First, from the perspective of adapted information, our method is not only adapting knowledge from GD, but also with very little speed overhead in the neural network training framework. It makes use of the internal data representation in neural network training. We adapt GD information directly from distributed word representations, which is a more natural way of learning in the neural network.

A further possible explanation is that the DAGRU approach is a better fit to the sequence learning tasks. We proposed several adaptation methods in Section 3, where adapting word vector representations ignores the previously adapted histories. Words are adapted individually from GD at each step. The approach of adapting context vector representations has the same issue. Although the adapted information is the context vectors which are generated by previous histories, the adaptation is still segmented. Taking the CVC approach as an example, the adapted context vector h_t^{CVC} is only used for predicting outputs in the output layer at step t, but not in the adaptation operation at step $t+1$. In contrast, the DAGRU approach can achieve sequential adaptation. The gates (u_t^{WVG} and r_t^{CVG}) are computed by the previous adapted context vector h_{t-1} and the previous GD context vector $h_{t_1}^*$. Furthermore, the computation of the current adapted representation also involves h_{t-1} and $h_{t_1}^*$. Thus, the adaptation histories are managed by the model and sequentially traverse along the input string.

Furthermore, we also compare the learning curves between the baseline LM and the DAGRU LM on validation and test data of the Penn Treebank. As Figure 5 shows, the predictions become more certain and accurate after iterations for training both LMs. Already after training iteration 2, the DAGRU LM starts to outperform the baseline LM in terms of perplexity at every iteration. The plots flatten after 18 iterations, and the learning begins to converge for both the baseline LM and DAGRU LM.

To demonstrate the scalability of the DAGRU adaptation approach, we also train LMs adapting from other freely available word vector models. SENNA (Semantic/syntactic Extraction using a Neural Network Architecture) is the word vector model received after a LM training (Collobert et al., 2011). The training data is obtained from Wikipedia. GloVe (Pennington et al., 2014) – Global Vectors for Word Representation – provides several versions of word vector models. The glove_6b model is trained on Wikipedia data and the English Gigaword Fifth Edition corpus;[7] the glove_42b model is trained on the Common Crawl data; and the glove_840b model is trained on the the Common Crawl and additional web data.

Table 6 presents the experimental results of DAGRU adaptation using different word vector models as GD data. The baseline models are trained on the Penn Treebank only. The word embedding numbers in Table 6 indicate the word vector size of the adapting word vector model, e.g. the SENNA model has a word vector size of 50 under Word Embedding = 50 setting. For the baseline systems with embedding

[7] https://catalog.ldc.upenn.edu/LDC2011T07

size 50 and 100, we observe a sudden explosion in the evaluation perplexities with the decay factor setting described in Section 4.1. We experimentally set the learning rate to a decay factor of 0.5 after 8 iterations. In Table 6, the DAGRU approach can produce better perplexity results in all settings.

5 Related Work

Domain adaptation for *n*-gram LMs is a well-studied research field. In general, there are approaches to select data which are similar to ID from GD (Moore and Lewis, 2010; Axelrod et al., 2011; Duh et al., 2013; Toral, 2013). There are also model mixture approaches (Bellegarda, 2004; Hsu and Glass, 2006; Allauzen and Riley, 2011), which try to find a weight to combine the ID LM and GD LM.

In neural LM work, one approach to perform domain adaptation is to use an additional adaptation layer to combine the GD neural LM into the ID neural LM (Park et al., 2010; Ter-Sarkisov et al., 2014). However, an LM trained on all GD data is required. Curriculum learning (Bengio et al., 2009), which rearranges the training data in a particular order to improve generalization, is also applied on domain adaptation on a neural LM (Shi et al., 2013). In the work of Mikolov and Zweig (2012), word predictions are conditioned on the word topic representations. Thus, building multiple topic-specific language models is avoided.

6 Conclusions and Future Work

In this paper, we present and compare several domain adaptation approaches using neural LM training. Compared to previous domain adaptation methods, we propose the idea of learning GD knowledge directly from GD word vectors instead of from raw sentences. Using the proposed domain adaptation gating mechanism, we demonstrate LM perplexity improvements on the Penn Treebank and News domain data sets. We compare the adaptation performance on several publicly available word vector models. We also apply the adapted LM in the SMT re-ranking task. The experimental results suggest that the proposed domain adaptation gating approach can efficiently produce a better LM and significantly improve SMT translation performance.

Our domain adaptation gating mechanism is not only limited to LM training. We are interested in further exploring its performance on other NLP tasks, e.g. neural machine translation. In future work, we are also interested to find out the efficiency of applying it to other gated RNNs, such as LSTM.

Acknowledgments

We would like to thank the three anonymous reviewers for their valuable and constructive comments and the Irish Center for High-End Computing (`www.ichec.ie`) for providing computational infrastructures. The ADAPT Centre for Digital Content Technology (`www.adaptcentre.ie`) at Dublin City University is funded under the Science Foundation Ireland Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

References

Cyril Allauzen and Michael Riley. 2011. Bayesian Language Model Interpolation for Mobile Speech Input. In *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association*, pages 1429–1432, Florence, Italy.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain Adaptation via Pseudo In-domain Data Selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 355–362, Edinburgh, UK.

Jerome R. Bellegarda. 2004. Statistical Language Model Adaptation: Review and Perspectives. *Speech Communication*, 42(1):93–108.

Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. 1994. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Trans. Neural Networks*, 5(2):157–166.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 41–48, Montreal, Quebec, Canada.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *Computing Research Repository (CoRR)*, abs/1412.3555.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation Data Selection using Neural Language Models: Experiments in Machine Translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Volume 2: Short Papers*, pages 678–683, Sofia, Bulgaria.

Michel Galley and Christopher D. Manning. 2008. A Simple and Effective Hierarchical Phrase Reordering Model. In *2008 Conference on Empirical Methods in Natural Language Processing, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 848–856, Honolulu, Hawaii, USA.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland.

G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. 1986. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1. In *Distributed Representations*, pages 77–109. MIT Press, Cambridge, MA, USA.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. LONG SHORT-TERM MEMORY. *Neural Computation*, 9(8):1735–1780.

Bo-June Paul Hsu and James R. Glass. 2006. Style & Topic Language Model Adaptation Using HMM-LDA. In *EMNLP 2007, Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, 22-23 July 2006*, pages 373–381, Sydney, Australia.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*, pages 388–395, Barcelona, Spain.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Tomas Mikolov and Geoffrey Zweig. 2012. Context Dependent Recurrent Neural Network Language Model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 234–239, Miami, Florida ,USA.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent Neural Network Based Language Model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*, pages 1045–1048, Makuhari, Chiba, Japan.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *Computing Research Repository (CoRR)*, abs/1301.3781.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013b. Exploiting Similarities among Languages for Machine Translation. *Computing Research Repository (CoRR)*, abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013c. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*, pages 3111–3119, Lake Tahoe, Nevada, United States.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013d. Linguistic Regularities in Continuous Space Word Representations. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 746–751, Atlanta, Georgia, USA.

Robert C. Moore and William D. Lewis. 2010. Intelligent selection of language model training data. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 220–224, Uppsala, Sweden.

Junho Park, Xunying Liu, Mark J. F. Gales, and Philip C. Woodland. 2010. Improved Neural Network Based Language Modelling and Adaptation. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, September 26-30, 2010*, pages 1041–1044, Makuhari, Chiba, Japan.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543, Doha, Qatar.

Yangyang Shi, Martha Larson, and Catholijn M. Jonker. 2013. K-component Recurrent Neural Network Language Models using Curriculum Learning. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, December 8-12, 2013*, pages 1–6, Czech Republic.

Aram Ter-Sarkisov, Holger Schwenk, Fethi Bougares, and Loïc Barrault. 2014. Incremental Adaptation Strategies for Neural Network Language Models. *Computing Research Repository (CoRR)*, abs/1412.6650.

Antonio Toral. 2013. Hybrid Selection of Language Model Training Data Using Linguistic Information and Perplexity. In *Proceedings of the Second Workshop on Hybrid Approaches to Translation*, pages 8–12, Sofia, Bulgaria.

Marlies van der Wees, Arianna Bisazza, Wouter Weerkamp, and Christof Monz. 2015. What's in a Domain? Analyzing Genre and Topic Differences in Statistical Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 560–566, Beijing, China.

Machine Translation Evaluation for Arabic using Morphologically-Enriched Embeddings

Francisco Guzmán, Houda Bouamor*, Ramy Baly† and Nizar Habash‡
Qatar Computing Research Institute
*Carnegie Mellon University in Qatar
†American University of Beirut
‡New York University Abu Dhabi
fguzman@qf.org.qa, hbouamor@cmu.edu,
rgb15@mail.aub.edu, nizar.habash@nyu.edu

Abstract

Evaluation of machine translation (MT) into morphologically rich languages (MRL) has not been well studied despite posing many challenges. In this paper, we explore the use of embeddings obtained from different levels of lexical and morpho-syntactic linguistic analysis and show that they improve MT evaluation into an MRL. Specifically we report on Arabic, a language with complex and rich morphology. Our results show that using a neural-network model with different input representations produces results that clearly outperform the state-of-the-art for MT evaluation into Arabic, by almost over 75% increase in correlation with human judgments on pairwise MT evaluation quality task. More importantly, we demonstrate the usefulness of morpho-syntactic representations to model sentence similarity for MT evaluation and address complex linguistic phenomena of Arabic.

1 Introduction

Statistical machine translation (SMT) into morphologically rich languages (MRL) faces many challenges: from handling a complex and rich vocabulary, to designing adequate MT metrics that take morphology into account. While the first problem has widely explored (e.g. by using morphological analysis tools to reduce sparsity), the evaluation part has only been partly addressed. This is problematic since traditional MT metrics struggle to distinguish between (*i*) incorrect lexical choices; (*ii*) valid alternative lexical or syntactic variations; and (*iii*) differences in morphological inflection that are the result of incorrect case assignment or morphological agreement. While metrics like METEOR (Denkowski and Lavie, 2011) have made it possible to distinguish between (*i*) and (*ii*) by using paraphrases, (*iii*) is still an open problem. As a result, progress in SMT for MRL is hindered by the lack of adequate evaluation metrics. Since SMT metrics are used not only for evaluation but also for tuning system parameters, it is crucial that the MT metrics correctly handle morphology.

Most recently, deep learning models have been used more heavily in different parts of the natural language processing (NLP) community, including MT and MT evaluation. One of the main advantages of such models is the use of distributed word representations (embeddings). It has been shown that word embeddings are able to capture to certain semantic and syntactic aspects of words (Mikolov et al., 2013). Further refinements allow the inclusion of morphological information into distributed representations (Cotterell and Schütze, 2015). Word embeddings have been shown to help with modeling textual similarity well in the context of MT evaluation for MT into English (Guzmán et al., 2015), and community Question Answering (Guzmán et al., 2016). Nonetheless little exploration has been done on the use of embeddings for MT into MRL.

In this paper, we investigate how embeddings obtained from different levels of lexical and morpho-syntactic linguistic analysis can improve MT evaluation into a MRL. Specifically we report on Arabic, a language with complex and rich morphology paired with a high degree of ambiguity (Habash, 2010). Our results show that using a pairwise neural-network over different representations produces results

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1398–1408, Osaka, Japan, December 11-17 2016.

that clearly outperform the state-of-the-art for MT evaluation into Arabic, by almost over 75% increase in correlation with human judgments on pairwise MT evaluation quality task. More importantly, we demonstrate that the use of embeddings based on morpho-syntactic representations in conjunction with the non-linear modeling capabilities of a neural-network help to capture the preferences of human judges.

Next, we present related work in Section 2. We describe our approach in detail in Section 3; and we evaluate in Section 4. We present a discussion of our findings in Section 5.

2 Related Work

Despite its well-known shortcomings (Callison-Burch et al., 2006), BLEU continues to be the de-facto MT evaluation metric. Several studies have attempted to improve upon it by taking into account different aspects of linguistic structures including: (*i*) synonym dictionaries or paraphrase tables (Denkowski and Lavie, 2011; Snover et al., 2010); (*ii*) syntactic information (Liu and Gildea, 2005; Giménez and Màrquez, 2007; Liu et al., 2010; Chen and Kuhn, 2011); (*iii*) morphology (Tantug et al., 2008); (*iv*) semantics (Dahlmeier et al., 2011; Lo et al., 2012) and (*v*) discourse (Guzmán et al., 2014b; Joty et al., 2014). Generally, these metrics have been focused on translation into English. However, there has been little attention into their direct applicability to languages with rich morphology.

Our work focuses on automatic evaluation of translation into morphologically rich languages, Arabic more specifically. In that sense, our work is related to AL-BLEU (Bouamor et al., 2014) which is an adaptation of BLEU that gives partial credits for stem and morphological matchings of hypothesis and reference words. Here, in addition to using lexical information captured by n-gram metrics, we show that using morpho-syntactic representations can significantly improve the correlation with human judgments. Furthermore, we use a neural-network, which uses non-linearities to improve modeling.

Over the past few years, neural network models have dramatically improved the state-of-the-art of different NLP applications (Goldberg, 2015). For instance, in SMT we have observed an increased use of neural nets for language modeling (Bengio et al., 2003; Mikolov et al., 2010), for improving answer ranking in community Question Answering (Guzmán et al., 2016), for improving the translation modeling (Devlin et al., 2014; Bahdanau et al., 2014; Cho et al., 2014) and for machine translation evaluation (Guzmán et al., 2015; Gupta et al., 2015). Our work is related to Guzmán et al. (2015), in several levels of lexical, syntactic and semantic are combined in a compact fashion using a pairwise neural framework. There are several differences between that work and ours: (*i*) we do not use syntactic embedding representations, (*ii*) we include additional pairwise features, namely the pairwise cosine similarity between embeddings; and (*ii*) we focus on an MRL language. While use of syntactic representations has proven a useful component to evaluate English, it relies heavily on an syntactic neural parser (Socher et al., 2013), which increases the complexity of the evaluation setup, and is not readily available for every language. Here, we instead use morpho-syntactic representations which capture both syntactic and morphological aspects of language. In our experiments, these simple representations are powerful enough to provide state-of-the-art performance.

In this work, we use neural network models to improve MT evaluation into Arabic using representations that capture morphology. Morphological structure has been shown to improve the quality of word clusters (Clark, 2003), word vector representations (Cotterell and Schütze, 2015) and neural language models (Botha and Blunsom, 2014). The novelty of our work resides in the way we integrate lexical and morpho-syntactic distributed representations into a neural-network. We demonstrate that combining several sources of complementary information is useful to capture sentence similarity in a translation evaluation scenario. And arguably, capture complex phenomena like morphological agreement.

3 Approach

We use a pairwise approach to translation evaluation (Guzmán et al., 2014a) using neural networks. We use neural networks for two reasons. First, to take advantage of their ability to model complex non-linear relationships efficiently. Second, to have a framework that allows for easy incorporation of distributed representations captured by lexical and morpho-syntactic embeddings. In this section, we describe the neural-network model and the distributed representations we use as features in this work.

3.1 Learning framework

Neural Network Model Our full neural network model for pairwise evaluation is depicted in Figure 1. It is a direct adaptation of the feed-forward NN proposed for English MTE (Guzmán et al., 2015). Technically, we have a binary classification task with input $x = (r, t_1, t_2)$, which outputs 1 if t_1 is a *better* translation than t_2 in the context of the reference r, or 0 otherwise. The network computes a sigmoid function $f(r, t_1, t_2) = \text{sig}(\mathbf{w_v^T}\phi(r, t_1, t_2) + b_v)$, where $\phi(x)$ transforms the input x through the hidden layer, $\mathbf{w_v}$ are the weights from the hidden layer to the output layer, and b_v is a bias term.

To decide which hypothesis is *better* given the tuple (r, t_1, t_2) as input, we map the hypotheses and the reference to a fixed-length vector $[\mathbf{x}_r, \mathbf{x}_{t_1}, \mathbf{x}_{t_2}]$, using embeddings based on different lexical and morpho-syntactic representations.

We model three types of interactions (between t_1, t_2 and r) using different groups of nodes in the hidden layer h_{12}, h_{1r}, h_{2r}. The input to each of these groups is the concatenation of the vector representations of the two interacting components i.e., $\mathbf{x_{1r}} = [\mathbf{x}_{t_1}, \mathbf{x}_r]$, $\mathbf{x_{2r}} = [\mathbf{x}_{t_2}, \mathbf{x}_r]$, $\mathbf{x_{12}} = [\mathbf{x}_{t_1}, \mathbf{x}_{t_2}]$.

In summary, the transformation $\phi(t_1, t_2, r) = [\mathbf{h_{12}}, \mathbf{h_{1r}}, \mathbf{h_{2r}}]$ can be written as :

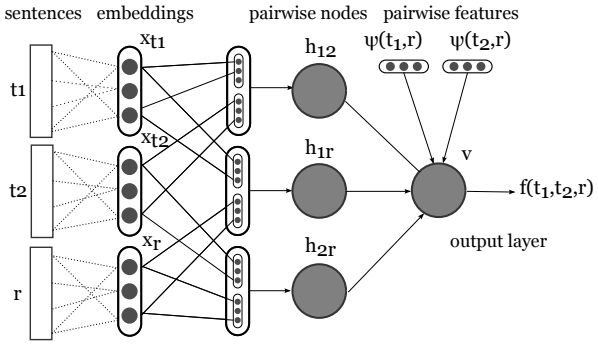


Figure 1: Overall architecture of the neural network.

$$\mathbf{h_{ij}} = \tanh(\mathbf{W_{ij}}\mathbf{x_{ij}} + \mathbf{b_{ij}})$$

where $ij \in \{12, 1r, 2r\}$, $\tanh(.)$ is a non-linear component-wise activation function, $\mathbf{W} \in \mathbb{R}^{H \times N}$ are the associated weights between the input layer and the hidden layer, and $\mathbf{b}$ are the bias terms.

The model further allows to incorporate external sources of information in the form of *skip arcs* ψ that go directly from the input to the output layer. These arcs represent pairwise *similarity* between each translation and the reference (we denote them as $\psi_{1r} = \psi(t_1, r)$ and $\psi_{2r} = \psi(t_2, r)$). We use these feature vectors to encode two basic elements: (*i*) the pairwise cosine similarity between the embeddings from each translation and the reference; and (*ii*) N-gram based MT evaluation measures (e.g., AL-BLEU, METEOR, and NIST). We provide more detail about pairwise features in the next section.

Pairwise Network Training The negative log-likelihood of the training data for the model parameters $\theta = (\mathbf{W_{12}}, \mathbf{W_{1r}}, \mathbf{W_{2r}}, \mathbf{w_v}, \mathbf{b_{12}}, \mathbf{b_{1r}}, \mathbf{b_{2r}}, b_v)$ can be written as follows:

$$J_\theta = -\sum_n y_n \log \hat{y}_{n\theta} + (1 - y_n) \log (1 - \hat{y}_{n\theta}) + \lambda \sum \theta^2 \quad (1)$$

where $\hat{y}_{n\theta} = f_n(t_1, t_2, r)$ is the activation at the output layer for the n-th data instance, and λ is the L_2 regularization penalty. The network is trained with stochastic gradient descent (SGD), mini-batches and adagrad updates (Duchi et al., 2011), using Theano (Bergstra et al., 2010).

Evaluating a single translation Most of the MT evaluation metrics are not designed to do pairwise, but absolute evaluation. In other words, we are interested in generating a score for a single translation t_1 given a reference r. To achieve this with our pairwise network, we compute the *goodness* margin, which tells us how good translation t_1 is better than any other translation $t_\varnothing$ given the reference r. In this case, $t_\varnothing$ is the average representation for all translations observed during training.

To compute the margin, we subtract the scores for the direct and reverse network predictions, and generate the final score for the sentence: $score(t, r) = 1 + (f(r, t, t_\varnothing) - f(r, t_\varnothing, t))/2$.

Note that we use both direct and reverse predictions as our network is not exactly symmetric. We also shift the score to range between $[0, 1]$.

3.2 Features

In this work, we compare different sets of features representing different levels lexical and morpho-syntactic information. As a baseline, we also used several MT metrics that are based on n-gram matches.

Lexical units A distinguishing characteristic of Arabic morphology is the presence of concatenative morphemes, where words are formed via concatenations of stems, affixes and clitics. To allow our system to model how morphemes interact at a finer level, we split the morphemes. We used MADAMIRA (Pasha et al., 2014), the state-of-the-art morphological analyzer and disambiguator, to perform morphological tokenization following ATB scheme (Habash and Sadat, 2006). We extracted two forms of lexical features: NORM and TOKEN, which are tokens with and without Alef/Yaa normalization, respectively. We also extract the LEMMA feature; a morphological abstraction that represents words related by inflectional morphology.

Morpho-Syntactic units We extracted part-of-speech (POS) tags according to different POS tagsets including: (*i*) CATIBPOS(Habash et al., 2009), (*ii*) KULICKPOS[1] (Kulick et al., 2006), (*iii*) BUCKWALTERPOS (Buckwalter, 2004) and (*iv*) STANFORDPOS tagsets. These tagsets differ in their richness and complexity they capture. CATIBPOS is the simplest with only 6 base tags[2], BUCKWALTERPOS is the richest with 485 base tags, and KULICKPOS and STANFORDPOS come in-between with 43 and 32 base tags, respectively. These tags were extracted using MADAMIRA, except for the Stanford tags, for which we used Stanford CoreNLP (Manning et al., 2014). Table 1 illustrates an example sentence with its lexical and morpho-syntactic features.[3]

Sentence:	عاد المصريون الذين اختطفوا إلى بلدهم – *ςAd AlmSrywn Alðyn AxtTfwA Ălý bldhm*						
TOKEN	+هم *+hm*	بلد *bld*	إلى *Ălý*	اختطفوا *AxtTfwA*	الذين *Alðyn*	المصريون *AlmSrywn*	عاد *ςAd*
NORM	+هم *+hm*	بلد *bld*	الي *Ăly*	اختطفوا *AxtTfwA*	الذين *Alðyn*	المصريون *AlmSrywn*	عاد *ςAd*
LEMMA	+هُم *+hum*	بَلَد *balad*	إِلَى *Ăilaý*	اِخْتَطَف *Aix.taTaf*	الَّذِي *Al~aðiy*	مِصرِيّ *miS.riy~*	عَاد *ςaAd*
CATIBPOS	+NOM	NOM	PRT	VBD-PASS	NOM	NOM	VRB
KULICKPOS	+PRP$	NN	IN	VBN	WP	DT+NNS	VBD
STANFORDPOS	PRP$	NN	IN	VBN	WP	DTNNS	VBD
BUCKWALTERPOS	+POSS _PRON _3MP	NOUN +CASE _DEF_GEN	PREP	PV_PASS +PVSUFF _SUBJ:3MP	REL_PRON	DET+NOUN +NSUFF _MASC_PL_NOM	PV+ PVSUFF _SUBJ:3MS
GLOSS	their	country	to	were abducted	which	the Egyptians	returned
English	The Egyptians who were abducted returned to their country.						

Table 1: Illustration of the lexical and morpho-syntactic feature representations extracted for an example sentence. The sentence is presented from right to left following the directionality of writing Arabic.

Distributed representations To obtain embeddings based on the lexical and morpho-syntactic units described above, we annotated the fifth edition of the Gigaword corpus (LDC2011T11) with each of the representations. Then, we trained word embeddings using word2vec (Mikolov et al., 2013). To

[1](Pasha et al., 2014) refers to Kulick tagset as the Penn ATB tagset, while it is Buckwalter tagset that is used in Penn ATB.

[2]The size of each tagset is expected to increase since Arabic morphemes can function as clitics, and their corresponding POS tags are then assigned a clitic marker (+). The final sizes of the extracted POS tags are shown in Table 4.

[3]Arabic transliteration is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007) (in alphabetical order): أ ب ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ي *Â b t θ j H x d ð r z s š S D T Ď ς γ f q k l m n h w y*, and the additional symbols: ء ’, أ Â, إ Ă, آ Ā, ؤ ŵ, ئ ŷ, ة ħ, ى ý. Diacritics are represented as: ـَ a, ـُ u, ـِ i, ـْ ., ـً ã, ـٌ ũ, ـٍ ĩ, and ـّ ~.

obtain sentence-level representations for each of the translations and the references, we used additive composition (Mitchell and Lapata, 2010) with dropping unknown words.

N-gram MT metrics We used the different n-gram based metrics to serve as a benchmark, and as additional features that capture lexical similarity. We used: BLEU+1 (Nakov et al., 2012),NIST (Doddington, 2002); METEOR (Denkowski and Lavie, 2011), 1-TER (Snover et al., 2006), and AL-BLEU (Bouamor et al., 2014), to compute scores at the sentence-level. For consistency with previous work, we report scores over words, and not over morphemes.

4 Experimental Setup

In this section, we describe the experimental settings we used through our study. First, we introduce our evaluation criteria, then we elaborate on the dataset and various settings we used for our experiments.

4.1 Performance evaluation

Automatic evaluation metrics are evaluated based on their correlation with human-performed evaluations (Soricut and Brill, 2004). In this work, we use Kendall's τ, a coefficient that measures the agreement between rankings produced by human judgments and rankings produced by an automatic metric, at the sentence-level. We use the WMT'12 (workshop of machine translation) definition of Kendall's τ that ignores ties, and is calculated as follows: $[\tau = (\#\text{ concordant pairs} - \#\text{ discordant pairs})\,/\text{total pairs}]$, where the *# concordant pairs* is the number of times the human judgment and the automatic metric agree in the ranking of any two translations that belong to the same source sentence. The *# discordant pairs* is the opposite. The value of τ ranges from -1 (all pairs are discordant) to $+1$ (all pairs are concordant).

4.2 Data

To evaluate the performance of the neural-network, we evaluated its Kendall's τ given different input representations. We used a medium-scale corpus of human judgments for Arabic MT outputs covering different topics in the news domain (Bouamor et al., 2014). The corpus is composed of 1,383 sentences selected from two datasets: (*i*) the standard English-Arabic NIST 2005 corpus, commonly used for MT evaluations and composed of political news stories; and (*ii*) a small dataset of translated Wikipedia articles. This corpus contains the source and target text along with the automatic translations produced by five English-to-Arabic MT systems: three research-oriented phrase-based systems with various morphological and syntactic features and two commercial, off-the-shelf systems. The corpus contains annotations that assess the quality of the five systems, by ranking their translation candidates from best to worst for each source sentence in the corpus. The annotation was conducted by mirroring the WMT evaluation campaigns (Callison-Burch et al., 2011), but with few key differences: (*i*) the full corpus was annotated with no random sampling, and (*ii*) the task was performed by two independent native speakers of Arabic. The total number of annotated pairs in this corpus is 33,192 (each sentence has two annotations, each yielding 12 rankings). The agreement between the annotators in terms of Kendall's τ is 49.20 (which roughly translates to agreement in 75% of all rankings). We used random partitions of 783 sentences for training (TRAIN), 300 sentences for tuning (DEV) and 300 sentences for testing (TEST).

4.3 Network Settings

We train our model on TRAIN with hidden layers of size 10 for 30 epochs with mini batches of size 30, L_2 regularization of 0.0001, and a learning rate of 0.01. We normalize the input feature values to the $[-1; 1]$ interval using minmax, and we initialize the network weights by sampling from a uniform distribution as in (Bengio and Glorot, 2010).

We evaluate the model on DEV after each epoch, and keep the one that achieves the highest accuracy. We selected the above parameter values on the DEV dataset using the full model, and we use them for all experiments described in Section 5, where we evaluate on the official TEST dataset.

Note that, we train the pairwise neural-network using all pairwise rankings in the TRAIN set. At test time, we compute the scores of the translations in TEST using the absolute scoring previously described.

5 Results

In this section we present the main results from our experiments. Since the dataset is new, we first present the Kendall's τ scores obtained from five n-gram based metrics that are popular in the community. Then, we present the results of using embeddings obtained from different representations as input to train the neural-network. Finally, we present combination of representations, that shed light on the capabilities of the neural-network to learn how to exploit different levels of lexical and morpho-syntactic information.

A. MT Metrics		**Kendall's τ**	**B. Embeddings**		**Kendall's τ**
			Lexical		
1	NIST	17.94	7	TOKEN	**24.35**
2	METEOR	17.90	8	NORM	23.22
3	AL-BLEU	17.02	9	LEMMA	21.17
4	BLEU	16.11	*Morpho-syntactic*		
5	1-TER	4.97	10	BUCKWALTERPOS	**25.49**
			11	KULICKPOS	16.25
6	5METRICS	**18.12**	12	STANFORDPOS	10.90
			13	CATIBPOS	5.41

Table 2: Kendall's τ on the TEST set for traditional n-gram based metrics as well as metrics built using different lexical and morpho-syntactic embeddings as input to the neural-network.

5.1 MT Metrics

In Table 2.A, we present four of the most popular MT metrics: BLEU, NIST, METEOR and 1-TER, along with AL-BLEU; a precision-based metric that is designed to handle Arabic morphology. These metrics were calculated over tokenized text. From the results, we observe that NIST and METEOR obtain very similar performances for this task, with 17.94 and 17.90, respectively.[4] This suggests that both the paraphrasing that METEOR uses, and the precision weighting by n-gram *importance* that NIST does, yield results that are more in line with human judgments than other metrics. Additionally, the role of morphology is important, as AL-BLEU presents an improvement over BLEU (17.02 vs 16.11).

Next, we combine the five metrics in a logistic regression model. We observe that the 5METRICS combination yields only minor improvements over the best single metrics, suggesting limited complementarity. We use this 5METRICS combination as a baseline to compare to next experiments.

5.2 Embeddings

In Table 2.B, we present the results of using the neural-network with embeddings for different representations. Using the embeddings for any lexical representations (TOKEN, NORM, LEMMA) produces significant improvements (+3%) over any of the MT metrics and their combination, yielding state-of-the-art results. The relative increase over 5METRICS ranges from 17% (+3.05% absolute with LEMMA) to 34% (+6.23% absolute with TOKEN).

Using the embeddings obtained for morpho-syntactic representations results in a wide-range of results. Surprisingly, the BUCKWALTERPOS representation obtains very competitive scores, even surpassing the lexical representations and yielding the highest score yet with a 41% relative increase over 5METRICS (+7.37% absolute). However, none of the other morpho-syntactic representations improve on 5METRICS. There is a clear correlation between the size of the POS tag-set and the performance of the metric. We explore this relationship more in depth in Section. 5.4.

5.3 Combination of Representations

Given the complimentary information embedded in the different representations, it is natural to combine them to obtain a stronger metric. To combine different embedding representations, we simply concatenate the different embedding representations before feeding them to the network. Below, we present

[4]Note that the Kendall's τ results for METEOR are in the range of the results for translation from English into other two morphologically rich languages (German: 18.0 and Czech 16.0) reported in WMT 2012 (Callison-Burch et al., 2012)

	Combinations	Kendall's τ result	prev. best	delta
C. Embeddings and N-gram based metrics				
Lexical				
14	5METRICS+TOKEN	23.62	24.35	(-0.73)
15	5METRICS+NORM	**24.17**	23.22	(+0.95)
16	5METRICS+LEMMA	23.51	21.17	(+2.34)
Morpho-syntactic				
17	5METRICS+BUCKWALTERPOS	**29.81**	25.49	(+4.32)
18	5METRICS+KULICKPOS	23.58	18.12	(+5.46)
19	5METRICS+STANFORDPOS	21.79	18.12	(+3.67)
20	5METRICS+CATIBPOS	18.93	18.12	(+0.81)
D. Embedding mixtures				
Lexical + Lexical				
21	TOKEN +NORM	25.12	24.35	(+0.77)
22	NORM + LEMMA	**25.42**	23.22	(+2.20)
23	TOKEN+LEMMA	24.90	24.35	(+0.55)
24	TOKEN+NORM+LEMMA	25.34	25.42	(-0.08)
Lexical + Morpho-syntactic				
25	BUCKWALTERPOS+TOKEN	* **31.87**	25.49	(+6.38)
26	BUCKWALTERPOS+NORM	30.69	25.49	(+5.19)
27	BUCKWALTERPOS+LEMMA	30.69	25.49	(+5.19)
E. Embedding mixtures + Ngram-based metrics				
Lexical + Lexical				
28	5METRICS+TOKEN+NORM	25.56	25.12	(+0.44)
29	5METRICS+NORM+LEMMA	25.42	25.42	(+0.00)
30	5METRICS+TOKEN+LEMMA	25.45	24.90	(+0.55)
31	5METRICS+TOKEN+NORM+LEMMA	**28.35**	25.42	(+2.93)
Lexical + Morpho-syntactic				
32	5METRICS+BUCKWALTERPOS+TOKEN	29.78	31.87	(-2.09)
33	5METRICS+BUCKWALTERPOS+NORM	**30.73**	30.69	(+0.04)
34	5METRICS+BUCKWALTERPOS+TOKEN+LEMMA+NORM	30.44	31.87	(-1.43)

Table 3: Kendall's τ on the TEST set for metrics built using combination of lexical and morpho-syntactic embeddings in addition to the 5METRICS. For comparison, we compare the result of the combination to the best result of any of the components in the combination. The best result overall is marked with *.

three sets of results involving different types of combinations in Table 3. In addition to the Kendall's τ of a particular combination x, we indicate the best previous result (i.e., result of a sub-combination that was presented already, *underlined*), and its delta from combination x.

Embeddings and MT Metrics In Table 3.C, we present the results of adding the 5METRICS to the different embeddings in Table 2.B as *skip-arc* features. For lexical embeddings, we observe slight improvements, except for the TOKEN representation, which has a small decrease. The combination of each of the morpho-syntactic embeddings with 5METRICS improves over 5METRICS. For the best performer so far, BUCKWALTERPOS, we get even more improvements, reaching an 11.69 absolute increase over 5METRICS (65% relative). This showcases that the neural-network is able to successfully make use of the complementarity between n-gram-based MT metrics and other sources of morpho-syntactic information.

Embedding Mixtures Table 3.D presents the results of combining lexical and morpho-syntactic embeddings. Every pairwise combination of lexical embeddings improves over either embedding combined. However, putting all lexical embeddings together is not as good as NORM+LEMMA. Combining the best morpho-syntactic performer (BUCKWALTERPOS) with each lexical embedding produces large improvements. This highlights the ability of the neural-network to exploit the interactions between different representations to provide a more robust metric. With BUCKWALTERPOS+TOKEN, we reach our best results, improving over the 5METRICS baseline by 13.75% (or 75.9% relative improvement).

Embedding Mixtures and MT Metrics Finally, we present in Table 3.E the results of combining 5METRICS with different combinations of embeddings. The addition of 5METRICS to pairs of lexical embeddings does not hurt and only slightly improves the scores. However, putting all of the lexical embeddings together with 5METRICS makes a nice increase, although still not competitive with our best result so far. Finally, combining 5METRICS+BUCKWALTERPOS with different lexical embeddings seems to make little improvement if any.

5.4 Discussion

One of the interesting results from Table 2.B is the wide range of scores for the different embeddings. Surprisingly, the BUCKWALTERPOS representation does remarkably well. Here we investigate some possibilities for this behavior.

Feature expressiveness In Table 4, we consider different aspects of the different embeddings: their vocabulary size, their token out-of-vocabulary rate, the standard deviation of the cosine similarity scores they assign to each sentence in TEST, as well as their respective Kendall's τ scores. The larger the variance, the more expressive and informative the representation is, as it gives more diverse values for cosine similarity between translation and references.

	Vocab Size	Token OOV Rate (%)	StDev Cos Sim	Kendall's τ
TOKEN	98,923	1.32	$9.57{\cdot}10^{-2}$	24.35
NORM	97,870	1.29	$9.51{\cdot}10^{-2}$	23.22
LEMMA	51,493	1.25	$9.12{\cdot}10^{-2}$	21.17
BUCKWALTERPOS	714	0.00	$9.87{\cdot}10^{-2}$	25.49
KULICKPOS	53	0.00	$4.97{\cdot}10^{-2}$	16.25
STANFORDPOS	32	0.00	$4.16{\cdot}10^{-2}$	10.90
CATIBPOS	11	0.00	$3.84{\cdot}10^{-2}$	5.41

Table 4: Statistics for the different embedding representations and their impact on TEST: vocabulary size of the embedding representations, token OOV rate of the embedding vocabulary on the TEST set, the standard deviation (StDev) of cosine similarity values for all translation–reference pairs in the TEST set and the Kendall's τon the TEST.

Overall, we observe that the variance (or the standard deviation) of the values in the cosine similarity and the Kendall's τ scores correlate very well at $\rho = 0.94$. One way to interpret this is that the more expressive the representation is, the better it performs at MT evaluation. The size of the vocabulary generally adds expressiveness to a feature, and correlates within the lexical subset and the morpho-syntactic subset of the representations (but not across or overall). However, lexical representations lose expressiveness because OOVs.

Agreement The BUCKWALTERPOS representation is particularly rich and captures the morphological complexity of Arabic. Thus, it can be used to represent patterns of grammatical agreement across words, e.g., verb-subject and noun-adjective. While the lexical embedding may capture that the two verbs يكتب *yktb* 'he writes' and تكتب *tktb* 'she writes' may occur in similar contexts, the BUCKWALTERPOS representation models for agreement with the context they appear in. Since Arabic uses agreement heavily across verbs and noun phrases, we expect that the simple additive combination used with word embeddings is able to capture impressions of gender or number, just enough to allow the model to distinguish between sequences that are closer to the reference from those that are not. For example, the sentence يريد الطفل الصغير أن يكتب *yryd AlTfl AlSγyr Ân yktb* 'the little boy wants to write' encodes the masculine singular gender in four of its words. Simpler POS tags do not mark this information and thus are unable to distinguish between sentences that use the correct and incorrect gender information. In our experience, human judges pay careful attention to agreement errors as they reduce the fluency of the text even as the basic "accuracy" of it is kept.

Morpho-semantics The BUCKWALTERPOS representation contains some semantic features related to specific POS tags. To give a concrete example, in our data set, a future verb was translated correctly

using the particle سوف *swf* 'will' (BUCKWALTERPOS: FUT_PART) by system A, and incorrectly as the particle لن *ln* 'will not' (BUCKWALTERPOS: NEG_PART) by system B. However, the reference had no future particle. While both translations were penalized equally (in terms of cosine similarity) in the TOKEN representation, the sentence with the negation POS NEG_PART was penalized more in the BUCKWALTERPOS representation. There are a number of negative particles in Arabic and all get the same BUCKWALTERPOS tag. This, we believe allows the neural-network to abstract and model them correctly.

Task specific Another reason that using morpho-syntactic helps MT evaluation even in isolation is that by definition, the setup of the MT evaluation forces translation and references to be somewhat close. Here, capturing agreement and POS semantics seem to correlate well with human judgments. We don't expect that using the morpho-syntactic to capture other type of sentence similarity (e.g mining sentence pairs in comparable corpora) will work as well.

6 Conclusions and Future Work

In this paper, we explored the use of different lexical and morpho-syntactic representations to model similarity in the context of MT evaluation for a morphologically rich language such as Arabic. Our results show that using the neural-network with the input embeddings obtained from different representations makes impressive gains individually and, especially, in combination. This confirms the neural-network ability to model complex interactions between the feature sets. The fact that the best performers in each of these subsets when combined (BUCKWALTERPOS+TOKEN) show very high gains; suggests that they are modeling very different aspects of the text. The lexical embeddings we posit model semantic similarity between test and reference; while the morpho-syntactic embeddings model syntactic similarity, and complex phenomena like morphological agreement.

Furthermore, when paired to morpho-syntactic representations, the distributed lexical information seems to be a good alternative to n-gram metrics to obtain state-of-the-art results. In the future, we would like to use rich morpho-syntactic representations for evaluation into other MRL languages to validate our observations for Arabic. For instance, this framework can be easily ported to European languages, such as German, Russian or Czech, where: (*i*) there the existence of morphological analyzers makes it feasible to develop morpho-syntactic embeddings, and (*ii*) there are datasets available (from WMT Evaluation Campaigns) to develop and evaluate such metrics. Finally, we want to test the use of morpho-syntactic representation in other tasks such as source language modeling for Neural Machine Translation.

Acknowledgment

We would like to thank the annotators, Fatma Tlili and Aisha Mohamed, for providing the machine translation quality human judgments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio and Xavier Glorot. 2010. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of AISTATS 2010*, volume 9, pages 249–256.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3(Feb):1137–1155.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference*, SciPy '10, Austin, Texas.

Jan A Botha and Phil Blunsom. 2014. Compositional Morphology for Word Representations and Language Modeling. In *ICML*, pages 1899–1907.

Houda Bouamor, Hanan Alshikhabobakr, Behrang Mohit, and Kemal Oflazer. 2014. A Human Judgement Corpus and a Metric for Arabic MT Evaluation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 207–213, Doha, Qatar.

Tim Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Catalog No.: LDC2004l02. Technical report, ISBN 1-58563-324-0.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the Role of BLEU in Machine Translation Research. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar Zaidan. 2011. Findings of the 2011 Workshop on Statistical Machine Translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Edinburgh, Scotland.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada.

Boxing Chen and Roland Kuhn. 2011. AMBER: A Modified BLEU, Enhanced Ranking Metric. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 71–77, Edinburgh, Scotland.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN encoder-decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078*.

Alexander Clark. 2003. Combining Distributional and Morphological Information for Part of Speech Induction. In *Proceedings of the Tenth Conference on European chapter of the Association for Computational Linguistics*, pages 59–66.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological Word-Embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1287–1292, Denver, Colorado.

Daniel Dahlmeier, Chang Liu, and Hwee Tou Ng. 2011. Tesla at WMT 2011: Translation Evaluation and Tunable Metric. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 78–84.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proceedings of the EMNLP 2011 Workshop on Statistical Machine Translation*, Edinburgh, UK.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1370–1380, Baltimore, Maryland.

George Doddington. 2002. Automatic Evaluation of Machine Translation Quality using n-gram Co-occurrence Statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Jesús Giménez and Lluís Màrquez. 2007. Linguistic Features for Automatic Evaluation of Heterogenous MT Systems. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 256–264.

Yoav Goldberg. 2015. A Primer on Neural Network Models for Natural Language Processing. *CoRR*, abs/1510.00726.

Rohit Gupta, Constantin Orasan, and Josef van Genabith. 2015. ReVal: A Simple and Effective Machine Translation Evaluation Metric Based on Recurrent Neural Networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1066–1072, Lisbon, Portugal.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, Alessandro Moschitti, Preslav Nakov, and Massimo Nicosia. 2014a. Learning to differentiate better from worse translations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 214–220, Doha, Qatar, October. Association for Computational Linguistics.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2014b. Using discourse structure improves machine translation evaluation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 687–698, Baltimore, Maryland.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2015. Pairwise neural machine translation evaluation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 805–814, Beijing, China.

Francisco Guzmán, Lluís Màrquez, and Preslav Nakov. 2016. Machine translation evaluation meets community question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 460–466, Berlin, Germany.

Nizar Habash and Fatiha Sadat. 2006. Arabic Preprocessing Schemes for Statistical Machine Translation. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 49–52. Association for Computational Linguistics.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Habash, Reem Faraj, and Ryan Roth. 2009. Syntactic Annotation in the Columbia Arabic Treebank. In *Proceedings of MEDAR International Conference on Arabic Language Resources and Tools, Cairo, Egypt*.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Shafiq Joty, Francisco Guzmán, Lluís Màrquez, and Preslav Nakov. 2014. DiscoTK: Using Discourse Structure for Machine Translation Evaluation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, WMT '14, pages 402–408, Baltimore, Maryland, USA.

Seth Kulick, Ryan Gabbard, and Mitchell Marcus. 2006. Parsing the Arabic Treebank: Analysis and Improvements. In *Proceedings of the Treebanks and Linguistic Theories Conference*, pages 31–42. Citeseer.

Ding Liu and Daniel Gildea. 2005. Syntactic Features for Evaluation of Machine Translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 25–32.

Chang Liu, Daniel Dahlmeier, and Hwee Tou Ng. 2010. TESLA: Translation Evaluation of Sentences with Linear-Programming-Based Analysis. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics (MATR)*, pages 354–359.

Chi-kiu Lo, Anand Karthik Tumuluru, and Dekai Wu. 2012. Fully Automatic Semantic MT Evaluation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 243–252.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent Neural Network Based Language Model. In *Interspeech*, volume 2, page 3.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2010. Composition in Distributional Models of Semantics. *Cognitive Science*, 34(8):1388–1439.

Preslav Nakov, Francisco Guzman, and Stephan Vogel. 2012. Optimizing for Sentence-Level BLEU+1 Yields Short Translations. In *Proceedings of COLING 2012*, pages 1979–1994, Mumbai, India.

Arfath Pasha, Mohamed Al-Badrashiny, Mona T Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *LREC*, volume 14, pages 1094–1101.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas*, AMTA '06, Cambridge, Massachusetts, USA.

Matthew Snover, Nitin Madnani, Bonnie J. Dorr, and Richard Schwartz. 2010. TER-Plus: paraphrase, semantic, and alignment enhancements to Translation Edit Rate. *Machine Translation*, 23(2-3).

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 455–465, Sofia, Bulgaria.

Radu Soricut and Eric Brill. 2004. A Unified Framework For Automatic Evaluation Using 4-Gram Co-occurrence Statistics. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 613–620, Barcelona, Spain.

Cüneyd Tantug, Kemal Oflazer, and Ilknur Durgar El-Kahlout. 2008. BLEU+: a Tool for Fine-Grained BLEU Computation. In *Proceedings of the 6th edition of the Language Resources and Evaluation Conference*, Marrakech, Morocco.

Ensemble Learning for Multi-Source Neural Machine Translation

Ekaterina Garmash and **Christof Monz**
Informatics Institute, University of Amsterdam
Science Park 904, 1098 XH Amsterdam, The Netherlands
{e.garmash,c.monz}@uva.nl

Abstract

In this paper we describe and evaluate methods to perform ensemble prediction in neural machine translation (NMT). We compare two methods of ensemble set induction: sampling parameter initializations for an NMT system, which is a relatively established method in NMT (Sutskever et al., 2014), and NMT systems translating from different source languages into the same target language, i.e., multi-source ensembles, a method recently introduced by Firat et al. (2016). We are motivated by the observation that for different language pairs systems make different types of mistakes. We propose several methods with different degrees of parameterization to combine individual predictions of NMT systems so that they mutually compensate for each other's mistakes and improve overall performance. We find that the biggest improvements can be obtained from a context-dependent weighting scheme for multi-source ensembles. This result offers stronger support for the linguistic motivation of using multi-source ensembles than previous approaches. Evaluation is carried out for German and French into English translation. The best multi-source ensemble method achieves an improvement of up to 2.2 BLEU points over the strongest single-source ensemble baseline, and a 2 BLEU improvement over a multi-source ensemble baseline.

1 Introduction

It has been shown for various machine learning applications that combining multiple systems can substantially improve performance (Rokach, 2010). System combination has also been successfully applied to statistical machine translation system (SMT) (Och and Ney, 2001; Matusov et al., 2006; Schwartz, 2008; Schroeder et al., 2009). However, system combination methods in the phrase-based (PBSMT) (Koehn et al., 2003) and hierarchical (HSMT) frameworks (Chiang, 2007) tend to be rather complex, requiring potentially non-trivial mappings between the partial hypotheses across the search spaces of the individual systems. For this reason SMT system combination is often limited to combining hypotheses from the n-best list. Alternatively, SMT systems can also be combined by processing different inputs as is the case for multilingual system combination. Unfortunately, input sentences in different languages may have very different structure, requiring elaborate methods to align sentences, which means that multilingual system combination is in practice restricted to languages with similar structures.

On the other hand, the recently emerged neural machine translation (NMT) framework offers a straightforward way to combine multiple systems. Most of the current NMT architectures (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015) formalize the target sentence generation as a word sequence prediction task. At each step during sequence prediction, a translation system outputs a *full* probability distribution over the target vocabulary. Therefore, the task of NMT system combination can be cast as an ensemble prediction task and a variety of existing general prediction combination methods can be applied. While this paper focuses on word-based models, the ensemble methods discussed in this paper can be applied to character-based sequential NMT models (Ling et al., 2015) in a very similar fashion.

Ensemble prediction is frequently used in NMT. A commonly reported method is uniform weighting of the output layers, i.e., distributions over the target vocabulary, produced by different trained instances of the same NMT architecture for the same language pair (Bojar et al., 2014). We use this method as

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1409–1418, Osaka, Japan, December 11-17 2016.

a baseline where the variations of the same system are produced by different parameter initializations. Alternatively, it is also possible to take parameter snapshots from different training epochs (Sennrich and Haddow, 2016). Recently, a new type of ensemble has been introduced in NMT: a *multi-source ensemble* (Firat et al., 2016), which is a set of NMT systems that translate from different source languages into the same target language. The general idea behind ensembles is that different predictors are likely to produce slightly different errors for different input instances, and if their predictions are combined, the overall error is reduced. Therefore, it is essential to introduce a minimum of diversity into an ensemble. Different random initializations force the same training algorithm to converge to different local optima. Different source sentences may differ quite significantly in their structure and thus present a different training task to an NMT learning algorithm. Multi-source ensembles offer a *linguistic* source of variation for translation systems, which may range from the way particular words are translated to the way the whole sentence is structured. This is in line with the common observation that translation systems trained on language pairs with different source languages differ in their performance (Bojar et al., 2014).

Besides the linguistic interest, multi-source translation can be applied in practical real-life scenarios. Examples include multi-lingual websites, where some content has already been made available in a couple of languages (by human translators) but needs to be further translated into other languages. Another example are parliamentary proceedings, typically available in many languages. Furthermore, Firat et al. (2016) study neural multi-source translation in the context of zero-resource translation, and experimentally show that multi-source pivot-based translation improves translation quality compared to simple pivot-based translation.

In this paper, we compare ensemble combination methods and evaluate how much performance gain they provide in the multi-source translation setting. All previous approaches employing NMT ensembles do so by applying a simple linear, uniform weighting of the output probability distributions. However, it seems intuitive to assign different weights to different systems, especially for the case of multi-source translation. Firat et al. (2016) evaluate multi-source ensemble method on French-Spanish into English. All three languages, and especially French and Spanish are very similar, so for this scenario their contribution may be close to equal. In order to explore the diversity offered by linguistic variation, we chose to evaluate on English, German, and French. German is structurally substantially different from both English and French, while English and French are quite similar and are typically easily mutually translatable (Bojar et al., 2014). Here, we translate from French into English and from German into English and we expect the former translation direction to perform substantially better than the latter. As we mentioned above, the performance of translation systems can vary with respect to different aspects of translation quality, such as correct reordering or lexical translation choice. In order to combine the strengths of individual systems, we train different combination functions on a held-out data set. We consider two types of combination: global (fixed weight for every prediction instance) and context-dependent, where weights are estimated for every prediction step. The latter is more fine-grained and is in principle able to capture more linguistic nuance, but may be difficult to train due to data sparsity.

This paper proceeds as follows: In Section 2 we describe the NMT architecture and optimization parameters that we use to train our translation systems. We further describe the training data and report individual translation performances of the trained systems. In Section 3 we discuss in detail the ensemble methods that we wish to employ. We present a grid search experiment to explore the performance potential of the two types of ensemble sets (single-source with different random initializations and multi-source). The experiment confirms our intuition that linguistic diversity can be beneficial for building NMT ensembles. In Section 4 we describe two models to train an ensemble combination function: a global weighting function and a context-dependent gating network. Section 5 contains results of translation experiments with different types of ensembles and their discussion. Section 6 provides some conclusions and outlook on future work.

2 Attention-based sequence-to-sequence NMT

We use an encoder-decoder neural translation model as described in Luong et al. (2015) to train individual predictors in an ensemble. The source encoder is a four-layer unidirectional LSTM. The final hidden

	Set	N. of lines	N. of word tokens	N. of word types
(a)	train DE-EN	123,955	DE: 2,3M; EN: 2,5M	DE: 89K; EN: 41K
	train FR-EN	127,755	FR: 2,9M; EN: 2,6M	FR: 58K; EN: 41,9K
	valid DE-EN	2,052	DE: 40.3K; EN: 41.5K	DE: 6.3K; EN: 4.7K
	valid FR-EN	887	FR: 21.5K; EN: 20K	FR: 3.7K; EN: 3.1K
(b)	combin.train DE-EN-FR	19,000	DE: 372K; FR: 447K; EN: 396K	DE: 29K; FR: 22.8K; EN: 17K
	combin.valid DE-EN-FR	1,000	DE: 19K; FR: 23K; EN: 20.5K	DE: 4.3K; FR: 4K; EN: 3.5K
(c)	test DE-EN-FR	3,000	DE: 62.9K; FR: 78K; EN: 69.5K	DE: 9.3K; FR: 8.3K; EN: 6.5K

Table 1: Data statistics for (a) NMT training, (b) ensemble combination function training, and (c) testing.

states of the encoder are used to initialize the decoder, which is also a four-layer unidirectional LSTM. On top of that, at each time step i the target hidden state from the top layer h_i^t and the set of all source hidden states $\{h_1^s, ..., h_n^s\}$ are used to compute a context vector c_i, where n is the length of the source sentence. We use the global attention mechanism with the *dot score* function from Luong et al. (2015):

$$c_i = \sum_{j=1}^{n} \text{softmax}(\textit{score}(h_i^t, h_j^s))h_j^s \tag{1}$$

$$\textit{score}(h^t, h^s) = (h^t)^\top h^s \tag{2}$$

Finally, the last (non-recurrent) hidden state $\widetilde{h_i}$ is computed to produce the output layer y_i, which in turn is used to compute a probability distribution over the target vocabulary by applying the softmax function:

$$\widetilde{h_i} = \tanh(W_c[c_i; h_i^t]) \tag{3}$$

$$y_i = \text{softmax}(W_y\widetilde{h_i}) \tag{4}$$

In the following two subsections we describe how we train the translation systems, which are later used as part of an ensemble during beam decoding, and how they perform individually.

2.1 Training details

2.1.1 Data

We consider a multi-source scenario based on French, German, and English. We chose these languages to introduce diversity into ensembles: German is structurally substantially different from both English and French, while English and French are structurally similar and are typically easily mutually translatable (Bojar et al., 2014). We expect a French-English system to perform better than German-English. But also, given the linguistic intuition about the structural differences between these translation pairs, we hope that the two systems compensate for each other's weaknesses when used in an ensemble.

To ensure that the only distinguishing factor between different language pairs is the source language, we chose training data which is to a large extent parallel across all three languages, i.e., it is a trilingual parallel text with small bilingual parts. To this end, we train all of our systems on the TED talks data set (Cettolo et al., 2012). All available data is split into a training and a validation set to train individual NMT systems, a training and a validation set to train a combination function for ensembles, and a test set for the final evaluation. The training data for learning the ensemble combination function and the test set should necessarily be fully parallel (tri-parallel). Therefore, we extracted our test set from the available trilingual data and did not use the test sets provided by (Cettolo et al., 2012) since they are not parallel across all three languages. Of course, the test set does not overlap with the training data. Table 1 provides some statistics of the training data.

2.1.2 Network and optimization details

We set the size of all embeddings and hidden layers to $1,000$. We use LSTM units for the recurrent hidden states and apply dropout with a probability of 0.2 (Luong et al., 2015). We make sure that the output (target vocabulary) layer is exactly the same for all NMT systems in an ensemble. To this end we precomputed the intersection of the target vocabularies for the respective language pairs in an ensemble.

system	BLEU	MET EOR
de→ en best	20.58	49.16
de→ en mean	20.31 ± 0.34	48.88 ± 0.33
fr→ en best	27.80	56.91
fr→ en mean	27.03 ± 0.87	56.05 ± 0.82

Table 2: Translation results for individual NMT systems. Decoding beam size is equal to 20. For each language pair we trained four NMT systems with different weight parameter initializations. For each pair we provide the best score and the mean score with standard deviation.

We rank the words in the intersection by their summed frequencies in order to select the top n words for the output later and map the remaining words to $\langle unk \rangle$. For the source sides, we applied the same procedure except for computing the intersection. For French and German we set the vocabulary size to $35,000$ and for English to $24,000$.

For network training we use a Neural MT system Tardis implemented in Torch.[1] All parameters are uniformly initialized, except for the embeddings which were initialized by sampling from a Gaussian with unit variance. Each translation system is trained for 20 epochs. We use SGD with mini-batches of size 20 with a learning rate of 1 and a decay rate of 0.8 after the fifth epoch. During training we limit the lengths of predicted sequences to 50 tokens.

2.2 Translation experiments with individual systems

For each language pair we train four systems by sampling different initial parameter values. Table 2 summarizes the performances of the individual systems. In all of the translation experiments the beam decoding size was set to 20. We evaluate performance with BLEU (Papineni et al., 2002) and METEOR (Lavie and Denkowski, 2009). The first thing to notice is that distributions for both metrics are consistent.

Since we focus on ensemble translation, we are interested in how diverse the translation systems are in their performance. First, we can see that the two language pairs have different degrees of internal variation. The performance variance of German-English is smaller than that of French-English. Second, we see differences between the two source languages when translating into the same target language. These differences are much higher than those within one language pair. This raises the question how helpful a source language with a significantly lower performance will be in a multi-source ensemble.

3 Ensemble prediction in NMT

Generally speaking, in order to specify an ensemble method, one needs to specify how a set of predictors is induced and how they are subsequently combined to make joint predictions (Rokach, 2010). For the construction of the set of predictors, it is essential that they make diverse predictions, to decrease the prediction error. Our goal in this paper is two-fold: First, understand how different ensemble induction methods influence the quality of translation. Second, find the optimal way to combine individual predictors into an ensemble. In Section 2.2 we discussed diversity across different NMT systems. In the remainder of this section we describe the induction and combination methods we evaluate in this paper. We also present experimental analysis of achievable translation improvements.

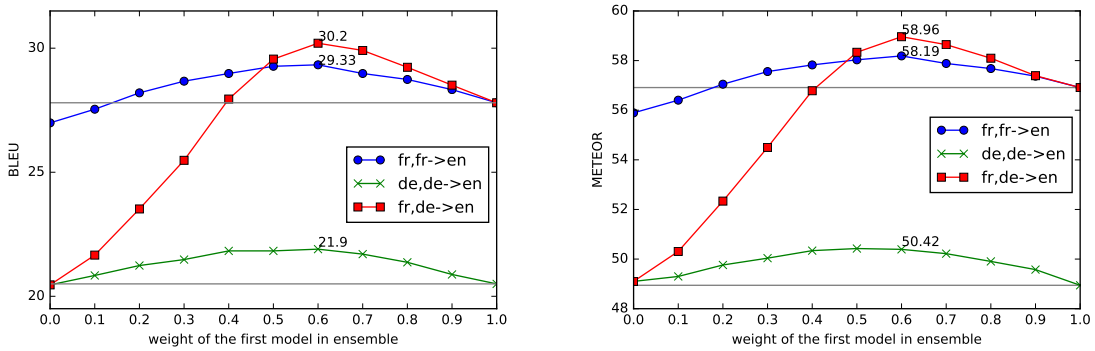
In this paper we consider two ways to induce an ensemble of translators:

1) different random initializations of NMT parameter values;

2) using semantically equivalent source sentences in different languages to translate into the same target language (translation systems with different source languages but the same target language).

The second method can be seen as different hidden state initializations of a (trained) NMT decoder. Different languages encode the same information in structurally different ways, which may influence the way the decoder is able to infer the translation from that.

[1] https://github.com/ketranm/tardis



(a) BLEU scores for Fr, De into En ensembles.

(b) METEOR scores for Fr, De into En ensembles.

Figure 1: Results of a 2-ensemble parameter sweep for the two types of ensemble induction. The x-axis represents the value of the first combination weight w_1. Number-marked points are the maximal observed scores for a given ensemble. The horizontal gray lines represent the scores of individual NMT systems used within the ensembles.

	stronger system in ensemble		uniform		best combination	
ensemble	BLEU	METEOR	BLEU	METEOR	BLEU	METEOR
fr_1,fr_2→en	**27.8**	**55.8**	29.2	58.0	29.3	58.1
de_1,de_2→en	20.5	49.10	21.8	50.4	21.9	50.4
fr,de→en	**27.8**	**55.8**	**29.5**	**58.3**	**30.2**	**58.9**

Table 3: Summary of the grid search of the scalar combination weights

In NMT the decision of which word to predict is based on the output layer and therefore we have to combine the output layers (Equation 4) of individual translators to obtain an ensemble prediction (Equation 5). The word thus predicted by an ensemble is then fed as input at the next prediction step in a sequence-to-sequence model.

$$y^{\mathcal{E}} = comb(y^1, ..., y^m) \tag{5}$$

We would like the method to be applicable to a situation where a trilingual parallel corpus, i.e., the corpus needed to train a multi-source combination function, is a scarce resource, which is a realistic assumption. Therefore we are interested in combination functions with a small number of trainable parameters. In our approach, we concentrate on *scalar* prediction combination: $comb(y^1, ...; , y^m) = w_1 y^1 + ... + w_m y^m$, where $\sum_i w_i = 1$ are scalar weights. In addition to the methods described in this paper we also investigated geometric mean combination ($\sqrt[m]{w_1 y^1 \cdot ... \cdot w_m y^m}$), but the resulting ensemble system under-performed the stronger individual system of the ensemble, therefore in the rest of the experiments we proceeded with the arithmetic mean function only.

Both single-source ensembles with different initializations (Sutskever et al., 2014) and multi-source ensembles have been used before (Firat et al., 2016). However all of the previous approaches use simple, uniform weighting. We refer to this method as *uniform combination*, as it does not assume anything about the contributions of the individual predictors. We perform grid search over the global combination weights[2] $\langle w_1, w_2 \rangle$ for a two-element ensemble (for both ensemble induction methods) over our test set with a step size of 0.1; see Section 2.1.1 for data and system setup.

The results of the grid search experiments are presented in Figure 1 and summarized in Table 3. We observe an increase in performance for both metrics for all ensembles. Moreover, we see that the metric scores are higher in the region of 0.5, which justifies uniform ensemble combination. At the same time, none of the graphs are completely symmetric: the highest scores are achieved with a weight value of 0.6 or 0.7 assigned to the stronger system in an ensemble. This result is intuitive, and it suggests

[2]I.e., a weight value is fixed for every instance in the test set.

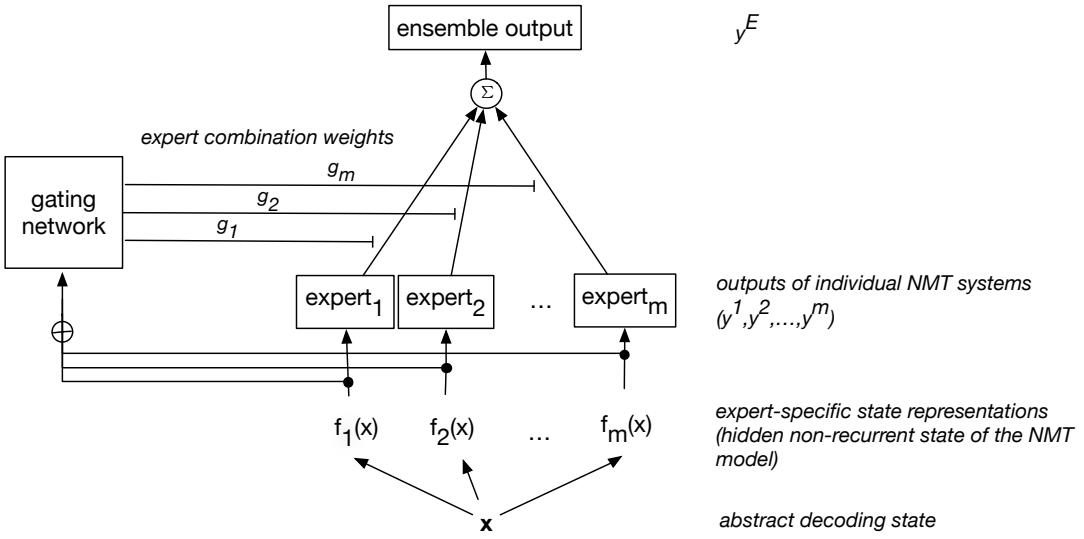


Figure 2: Mixture of NMT experts used to make context dependent translation prediction.

to investigate a combination method that could distinguish between the relative contributions of the individual members of an ensemble. We will distinguish between two kinds of combination functions: global and context-dependent. The former combines NMT predictions in the same way for every input at every decoding time step. The latter can combine predictions differently depending on the current context during decoding. We describe the corresponding learning methods in Section 4.

The second major finding of our parameter sweep is that the multi-source ensemble gives a higher upper bound performance than single-source ensembles, even though one of the ensemble members is substantially weaker in its individual performance. This finding reinforces the original linguistic motivation for multi-source ensembles with which we can obtain improvements of up to 0.87 BLEU and 0.77 METEOR over the highest single-source ensemble result.

In the following sections we describe how we make use of the uncovered potential of the two types of ensembles. We are especially interested in making full use of the complementary strengths of systems with different source languages.

4 Combination function learning

Having established that contributions of individual systems towards a correct prediction in single-source and multi-source ensembles are not equal, we develop an approach that is capable of training a function that can combine them optimally. For the case of multi-source ensembles the combination training set is a trilingual set consisting of $19,000$ lines (see Section 2.1.1 for details). We deliberately chose a small data set to establish how applicable the method is in a low-resource scenario. We use the same training set to train single-source ensembles. In this section we present two kinds of combination models, as well as their training details.

First, when a scalar combination vector is fixed for every prediction step, we refer to it as *global combination*. The optimized function is a vector $\langle w_1, ..., w_m \rangle$, where m is the size of the ensemble set. We train it with AdaGrad (Duchi et al., 2011) for 10 epochs with a learning rate of 0.001.

Second, we explore a more fine-grained combination method, where the contributions of individual predictors are assessed based on the decoding state. We adapt a mixture of experts model (Jacobs et al., 1991) to learn the *context-dependent combination*. The original mixture of experts model works as follows: We have a set of experts (predictors) and an input x, which is fed to each of the predictors. x is also fed to a gating network which outputs weights for each of the experts. The resulting prediction is a weighted sum of experts: $\mu = \sum_i g_i \mu_i$, where μ_i is the output of the i-th expert and g_i is its gating weight. Here, we realize context dependence by making use of a parameterized gating network.

Adapting the mixture of experts model (Jacobs et al., 1991) to the NMT scenario presents itself with a few challenges. NMT models are sequential and therefore the output at time step i depends on the current input word and the previous hidden state, which encodes the translation history for a *given* expert. This leads to two problems: the input representation is specific to an expert and it is also quite complex as it is a combination of hidden state and previously predicted word. We address the first problem by simply concatenating vectors which are inputs to each of the translators at time step i. There are a number of ways to address the second problem. But essentially, we would like to think of input x as some abstract decoding state corresponding to the context of the ensemble translation process at time step i.

In the first set of experiments, we opt for using the already available representations for the decoding state x, rather than formulating an explicitly, linguistically-motivated definition. Given the complex modular structure of an NMT model, there are a number of hidden states, such as the hidden recurrent states, the context vector, or the non-recurrent hidden state $\widetilde{h}$, which can be chosen to represent the decoder state which is the input to the gating network. In our approach, we use the last hidden state $\widetilde{h}$. We choose $\widetilde{h}$ because it already captures a large amount of information such as the previously predicted word, previous hidden state, and attention distribution over the source words.[3] In addition, the output layer is more directly connected to $\widetilde{h}$ than any of the states from lower layers. This is an important consideration given that the amount of training data is severely limited.

The architecture of our context-dependent combination function is presented graphically in Figure 2. The ensemble output $y^{\mathcal{E}}$ is computed as in Equation 6, where g_j is the gating weight, x represents the abstract decoding state at step i and $f^j(x)$ is its expert-specific representation (for expert j), namely $\widetilde{h}^j$:

$$y^{\mathcal{E}}(x) = \sum_j g_j \mu_j(x) \tag{6}$$

$$\mu_j(x) = \text{softmax}(W_y f^j(x)) \tag{7}$$
$$= \text{softmax}(W_y \widetilde{h}^j) \tag{8}$$
$$= y^j \tag{9}$$

g_j is the j-th output unit of the gating network computed as in Equation 10. The gating network is a feed-forward neural network with one hidden layer of size 250 and tanh non-linear activation function. The output layer is of size m, where m is the number of experts. Values of the output layer are normalized by applying softmax. The mixture model allows to back propagate errors both to the gating network and the experts themselves. However, considering the small size of the training data and the complexity of the experts, in terms of number of parameters, full back propagation is likely to result in over-fitting. Therefore, we only update the weights of the gating network, where the weights of the NMT predictors have been pre-trained separately. We train our mixture model for 10 epochs with AdaGrad with a learning rate of 0.001.

$$g = \mathit{softmax}(W_{\mathit{gate}} \mathit{tanh}(W_{\mathit{hid}}[f^1(x); ...; f^m(x)] + b_{\mathit{hid}}) + b_{\mathit{gate}}) \tag{10}$$

5 Translation experiments with trained ensembles

In the previous sections we have shown that NMT ensembles, and in particular multi-source ensembles, can improve translation quality. We proposed two methods to learn an ensemble combination function from data, which is more capable of exploiting the potential of ensembles than simple uniform weighting. In this section we test these methods in translation experiments. We compare the two ensemble induction methods (same-source systems with different parameter initializations and multi-source set) and apply all combination methods. All results are presented in Table 4.

[3] In our preliminary experiments, we also experimented with using other layers of the NMT model as the decoding state x: the top-most recurrent layer of decoder, the context vector, and the embedding of the previously predicted target word as the decoding state. However, using the non-recurrent hidden state $\widetilde{h}$ achieved the best results overall.

Ensemble set	Combination type					
	uniform		global		context-dependent	
	BLEU	METEOR	BLEU	METEOR	BLEU	METEOR
$de_1^2 \rightarrow$ en	21.8	50.4	21.8	50.4	21.8	50.3
$de_1^4 \rightarrow$ en	21.8	50.4	21.8	50.4	-	-
$\{de_1^2, de_3^4\} \rightarrow$ en	-	-	21.8	50.4	**22.8**	**51.0**
$fr_1^2 \rightarrow$ en	29.2	58.0	29.2	58.1	29.3	58.1
$fr_1^4 \rightarrow$ en	29.2	58.0	29.2	58.1	-	-
$\{fr_1^2, fr_3^4\} \rightarrow$ en	-	-	29.2	58.1	**30.2**	**59.0**
de,fr$\rightarrow$ en	**29.5**	**58.3**	**29.9**	**58.7**	30.3	59.2
de_1,de_2,fr_1,$fr_2 \rightarrow$ en	29.4	58.3	29.3	58.2	-	-
$\{\{de_1,fr_1\},\{de_2,fr_2\}\}\rightarrow$ en	-	-	29.2	57.9	**31.5**	**60.3**

Table 4: Translation experiments for the French, German into English scenario. *Ensemble set* designates ensemble induction method. *Combination type* refers to the method used to combine predictions during decoding. We use curly brackets to denote hierarchical ensemble combination.

For each ensemble set type, we evaluate ensembles of size 2 and 4. The notation below should be understood as follows: $de_k^{k+l} \rightarrow$ en stands for a single-source ensemble of l German-English systems. Analogously for French-English. de,fr $\rightarrow$ en is a multi-source ensemble of size 2, and we use subscript indices if there are more than 2 systems in an ensemble covering the same source language. We only apply context-dependent combination for ensembles of size 2 to avoid overfitting for bigger ensembles. Note that this does not prevent the application of our context-dependent combination method to bigger ensembles, as we can combine systems hierarchically. In a hierarchical ensemble the set of predictors is divided into disjoint subsets and each of the subsets is combined separately. The resulting combination systems can then be treated as predictors in a new ensemble, and thus can be further combined for a joint prediction. In our case the maximal number of predictors is 4, therefore our hierarchical ensembles have 2 levels. Hierarchical ensembles allow one to make prediction combinations multiple times which can further boost the potential of an ensemble. Since in this paper we only consider a low-resourced scenario with a small amount of training trilingual data, we do not train a hierarchical combination function. Instead, we do global or context dependent combination for ensemble sets at the bottom level (level of individual NMT systems) and then weight their outputs uniformly (level of combined systems). We use curly brackets denote hierarchical combination.

We see in Table 4 that multi-source ensembles generally perform better that single-source ensembles. The largest improvements for multi-source ensembles are due to our context-dependent combination method. This suggests that the trained gating network is able to capture linguistic context. We note that a multi-source ensemble with contextual combination outperforms the empirical upper bound estimated in Section 3, although it should be noted that this is an upper bound for a global combination method. On the other hand, for single-source ensembles, context-dependent combination (by itself) does not provide additional improvements as compared to global weighting. This suggests that the variation found in single-source ensembles is not as systematic as in multi-source ensembles. As part of future work, we are planning to perform a more linguistically oriented analysis to identify contexts triggering a high degree variation in ensembles. The results of such analysis will also provide the basis for a more linguistically oriented definition of the decoding state x as defined in Section 4.

We also note that simply increasing the size of an ensemble does not necessarily improve translation performance. Previous approaches using NMT ensembles often report performance increases for ensembles consisting of a larger number of systems, typically 8 or 12. One could therefore speculate that ensembles of 4 systems are not enough to significantly increase diversity as compared to an ensemble of size 2. Of course, our result are also influenced by several other factors such as the choice of languages, training data, etc. However, we should point out that hierarchically combining a set of 4 systems does improve translation quality. At this point, hierarchical combination still requires further investigation,

but for the time being, it can be seen as a simple 'recipe' to boost translation quality further.

6 Conclusions

In this paper we compared existing ensemble set induction methods for NMT and proposed a number of system combination methods: global (across instances) weighting of predictors' outputs and context-dependent weighting. Our main goal was to validate the linguistic hypothesis that translation systems from different source language into the same target language have complementary strengths and weaknesses in terms of translation performance and introduce an approach that can exploit the respective strenghts and weaknesses to achieve better translation quality. In our experiments with German-English and French-English we found that multi-source ensembles yield the best performance, compared to the individual translation systems, as well as compared to single-source ensembles of NMTs produced by different random initializations. This is an interesting finding because individually the two systems differ substantially in their translation quality. We also found that ensemble combination based on a gating network that decides how to combine systems at every prediction step achieves better performance as compared to a global (constant) combination function or uniform weighting in the majority of cases.

Overall this is a compelling result and it leaves us with a number of questions for future work. First, can we characterize linguistically what types of contexts are mores suited to be translated by a German-English system, and which are more suited to be translated by a French-English system? Gaining insights in that direction can help us answer another question: is there a better way to represent the current context which is the input to the gating network? In this paper we used a concatenation of each system's last hidden state $\widetilde{h}$, but a potentially more effective and linguistically more intuitive representation may be found. Finally, it would be interesting to see to what extent our approach can benefit from three or more mutually different source languages.

Acknowledgements

This research was funded in part by the Netherlands Organization for Scientific Research (NWO) under project number 639.022.213 and a Google Faculty Research Award. We also thank NVIDIA for their hardware support. We thank Ke Tran for providing the neural machine translation baseline system and the anonymous reviewers for their helpful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58. Association for Computational Linguistics Baltimore, MD, USA.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy, May.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T Yarman Vural, and Kyunghyun Cho. 2016. Zero-resource translation with multi-lingual neural machine translation. *to appear in Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

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

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54. Association for Computational Linguistics.

Alon Lavie and Michael J Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3):105–115.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *Association for Computational Lingustics*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September. Association for Computational Linguistics.

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing consensus translation for multiple machine translation systems using enhanced hypothesis alignment. In *European Chapter of the Association for Computational Linguistics*.

Franz Josef Och and Hermann Ney. 2001. Statistical multi-source translation. In *Proceedings of MT Summit*, volume 8, pages 253–258.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Lior Rokach. 2010. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39.

Josh Schroeder, Trevor Cohn, and Philipp Koehn. 2009. Word lattices for multi-source translation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 719–727. Association for Computational Linguistics.

Lane Schwartz. 2008. Multi-source translation methods. In *Proceedings of AMTA 2008*, October.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. *Proceedings of the First Conference on Machine Translation, Volume 1: Research Papers. Berlin, Germany, pp. 83-91.*

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

Phrase-based Machine Translation using Multiple Preordering Candidates

Yusuke Oda† **Taku Kudo**‡ **Tetsuji Nakagawa**‡ **Taro Watanabe**‡
†Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma-shi, Nara 630-0192, Japan
‡Google Japan Inc.
6-11-1 Roppongi, Minato-ku, Tokyo 106-6108, Japan
oda.yusuke.on9@is.naist.jp {taku, tnaka, tarow}@google.com

Abstract

In this paper, we propose a new decoding method for phrase-based statistical machine translation which directly uses multiple preordering candidates as a graph structure. Compared with previous phrase-based decoding methods, our method is based on a simple left-to-right dynamic programming in which no decoding-time reordering is performed. As a result, its runtime is very fast and implementing the algorithm becomes easy. Our system does not depend on specific preordering methods as long as they output multiple preordering candidates, and it is trivial to employ existing preordering methods into our system. In our experiments for translating diverse 11 languages into English, the proposed method outperforms conventional phrase-based decoder in terms of translation qualities under comparable or faster decoding time.

1 Introduction

One of the main problem of phrase-based statistical machine translation (PBMT) (Koehn et al., 2003; Och and Ney, 2004) is handling the difference of word orders between source and target languages. Decoding-time reordering models (Koehn et al., 2005; Zens and Ney, 2006; Galley and Manning, 2008) measure positional relationship between each phrase at the decoding time. However, reordering models have a common problem in that it is difficult to take global information in the source sentence into account, and as a result the decoder may generate grammatically incorrect word orders. In addition, using reordering models demands a complicated decoding algorithm, in which the decoder has to consider concatenations of source phrases with arbitrary orders.

On the other hand, preordering methods (Xia and McCord, 2004; Isozaki et al., 2010; Neubig et al., 2012; Nakagawa, 2015) change word orders of source sentence to be close to the target sentence before starting the decoding process. These methods can use global information in the source sentence and may generate grammatically correct reordering results. However, previous PBMT methods with preordering usually take only one-best preordered sentence and it is difficult to avoid the noise of the input caused by the errors from preordering methods.

One of the trivial way to avoid preordering errors is to obtain N-best preordering candidates, translate each candidate one-by-one and select the most probable result (Li et al., 2007; Zhu, 2014). This method has an obvious problem on computation time because the decoding process is executed N times. Another way to resolve preordering errors is combining a preordering method and decoding-time reordering models. However, it is not trivial to integrating these methods in a single system while comprehending their interactions.

In this study, we propose a new phrase-based decoding method which employs multiple preordering candidates for a source sentence. Our method first encodes multiple preordering candidates as a compact graph structure (we call it *preordering lattice*), and generates translations by a single-pass traversal on the preordering lattice which can take into account all preordering candidates.

Several previous work proposed decoding methods using graph structures with respect to preordering (Niehues and Kolss, 2009; Herrmann et al., 2013a; Herrmann et al., 2013b); however, these methods are tightly integrated with a specific graph structure defined on top of the methods themselves. Another previous work focused on multi-source translation based on a confusion network of multiple source

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1419–1428, Osaka, Japan, December 11-17 2016.

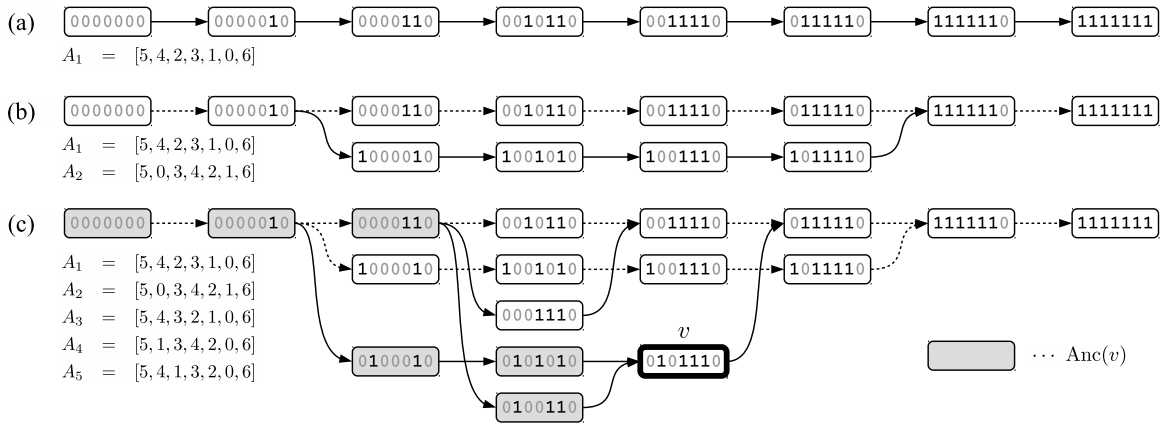


Figure 1: Generating preordering lattice from multiple preordering candidates.

sentences (Schroeder et al., 2009; Jiang et al., 2011); however, the derived confusion network is too constrained to represent variation of preordering, and it cannot take the advantage of alternative reordering in the multiple source sentences.

Compared with above previous works, our method is more generic in that the preordering lattice is constructed based only on the word permutations of the source sentence which are generated from arbitrary and independent preordering methods, and the preordering lattice guarantees that all preordering candidates are compactly encoded in the graph structure. In addition, we show that our preordering lattice approach ourperforms conventional decoding-time reordering methods even with a simple left-to-right dynamic programming algorithm. Our experiments show that the proposed method can achieve comparable or higher translation qualities against a conventional phrase-based method under diverse 11 language pairs: Ar/Zh/Fr/De/It/Ja/Ko/Pt/Ru/Es/Tr into English.

2 Graph Representation of Multiple Preordering Candidates

We denote the source sentence S as an array of words $S = [s_1, s_2, \cdots, s_I]$, and assume that a preordering method takes the source sentence S as an input and returns multiple preordering candidates $\mathcal{A} = \{A_1, A_2, \cdots, A_N\}$, $A_n = [a_1^n, a_2^n, \cdots, a_I^n]$ together with their confidence scores $\mathcal{C} = \{C_1, C_2, \cdots, C_N\} \in \mathbb{R}^N$, where I is the number of words in the source sentence, and N is the number of preordering candidates, and each confidence scores satisfies that $C_j > C_k$ if $j < k$. Each $a_i^n \in \{0, 1, \cdots, I-1\}$ denotes an original position of the source word in S. For example, if we take a Japanese sentence $S =$ [“今日 (today)”, “は”, “いい (good)”, “天気 (weather)”, “です (be)”, “ね”, “。”] (“It is nice weather today.”) and a preordering candidate $A = [5, 4, 2, 3, 1, 0, 6]$, then we obtain a preordered sentence $S' =$ [“ね”, “です”, “いい”, “天気”, “は”, “今日”, “。”].

We introduce an alternative view of the preordering candidate A_n which is represented as a chain graph structure illustrated in Figure 1(a), and we call this graph *preordering lattice*. Each node in the preordering lattice has a *coverage* generated by the preordering candidate, and each edge represents the one-word transition between two coverages. Each coverage represents a set of *processed* words at each timing of decoding, and we encoded each coverage as a bit vector representation in the same way as those used in a conventional phrase-based decoding algorithm. For example, a coverage vector $[0000110]$ indicated that 7 source words should be translated while decoding, and 5th and 6th words are already translated before reaching this coverage. Preordering lattice can be uniquely obtained from a preordering candidate by starting from an empty coverage ($[000\cdots]$) and adding 1 into a_n-th element one-by-one. The process can be regarded as a left-to-right decoding process with single word phrase pairs in a phrase-based decoding.

When multiple preordering candidates are available, we merge them into a single graph structure. Specifically, we integrate nodes in two preordering lattices if their nodes have same coverage vectors each other as shown in Figure 1(b), in which another preordering candidate $[5, 0, 3, 4, 2, 1, 6]$ is merged

Algorithm 1 Integrating preordering lattices

1: $\mathcal{A} \leftarrow$ Array of preordering candidates
2: $G \leftarrow$ Empty lattice
3: **for** $A \in \mathcal{A}$ **do**
4: $\quad G' \leftarrow \mathrm{Lattice}(A)$
5: $\quad$ **for** $v' \in V(G')$ **do**
6: $\quad\quad$ **if** $\neg(\exists v, v \in V(G) \wedge L(v) = L(v'))$ **then**
7: $\quad\quad\quad V(G) \leftarrow V(G) \cup \{v'\}$
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: $\quad$ **for** $(v'_1, v'_2) \in E(G')$ **do**
11: $\quad\quad$ **if** $\neg(\exists v_1, v_2, (v_1, v_2) \in E(G) \wedge L(v_1) = L(v'_1) \wedge L(v_2) = L(v'_2))$ **then**
12: $\quad\quad\quad v''_1 \leftarrow v$ s.t. $v \in V(G) \wedge L(v) = L(v'_1)$
13: $\quad\quad\quad v''_2 \leftarrow v$ s.t. $v \in V(G) \wedge L(v) = L(v'_2)$
14: $\quad\quad\quad E(G) \leftarrow E(G) \cup \{(v''_1, v''_2)\}$
15: $\quad\quad$ **end if**
16: $\quad$ **end for**
17: **end for**
18: **return** G

together. In this case, there are 4 nodes with same coverage vectors [0000000], [0000010], [1111110], [1111111] in two preordering lattices, which are integrated as shared nodes. Figure 1(c) shows an example of merging 5 preordering candidates in $\mathcal{A}$ as a single preordering lattice. It is guaranteed that this preordering lattice is uniquely determined given a set of preordering candidates $\mathcal{A}$, and the final graph structure does not depend on the integrating order. Thus, we can merge new lattice paths by enumerating preordering candidates one-by-one. Algorithm 1 shows the method to generate integrated preordering lattice G from $\mathcal{A}$, where $V(G), E(G), E_i(v), E_o(v), L(v)$ represent the sets of nodes/edges in G, input/output edges of given node v, head (exit side) and tail (entrance side) nodes of the edge e, and the label (= coverage vector) of given node v, respectively. $\mathrm{Lattice}(A)$ represents a unique chain graph determined by a preordering candidate A as described above.

Preordeing lattices generated by Algorithm 1 guarantee that all integrated preordering candidates are represented as a path over the lattice, and also guarantee that all words in the source sentence appear only once in any paths over the lattice. In addition, the preordering lattice often includes extra paths which represents other preordering candidates not in the original candidate set $\mathcal{A}$. Thus, the decoding algorithm described in the next section actually explores more preordering candidates when compared with a decoding algorithm which relies only on $\mathcal{A}$.

The preordering lattice is similar to the word lattice structure (Dyer et al., 2008), but all edges in the preordering lattice represent specific words in one source sentence. Daiber et al. (2016) also described a similar structure to our lattice using finite-state transducer (FST), and applied determinization and minimization to compress the lattice. On the other hand, we introduced more simple algorithm described in Algorithm 1 to achieve similar compression.

3 Decoding Algorithm over the Preordering Lattice

We also intoduce a simple decoding algorithm to generate translations directly using the preordering lattice. Our algorithm runs by traversing the lattice in a left-to-right manner. Algorithm 2 presents our decoding algorithm over the preordering lattice without score calculation, where $\mathrm{Begin}(G)$ and $\mathrm{End}(G)$ represents leftmost and rightmost nodes in a given preordering lattice. The decoder determines partial translations at each node in the preordering lattice according to a topological order ($\mathrm{TopologicalSort}(\cdots)$ in line 4), and finally returns the one-best hypothesis at $\mathrm{End}(G)$ ($\mathrm{BestResult}(\cdots)$ in line 15).

Now we focus on the partial translation hypotheses generated at the node v in Figure 1(c). When

Algorithm 2 Decoding over the Preordering Lattice

```
1: G ← Preordering lattice
2: v_L ← Begin(G)
3: H[v_L] ← {""}
4: for v ← TopologicalSort(V(G)\{v_L}) do
5:     H'[v] ← { }
6:     for ∀v'. v' ∈ Anc(v) ∧ IsCandidatePath(v', v) do
7:         for h' ← H[v'] do
8:             for φ ← PhrasePairs(v', v) do
9:                 H'[v] ← H'[v] ∪ {Join(h', φ)}
10:            end for
11:        end for
12:    end for
13:    H[v] ← Prune(H'[v]; K)
14: end for
15: return BestResult(H[End(G)])
```

computing a partial translation hypothesis at v, the decoder first computes ancestor nodes $\mathrm{Anc}(v)$, i.e., a set of nodes from which v is reachable, as shown in the gray nodes in Figure 1(c). Partial hypotheses $H[v'], v' \in \mathrm{Anc}(v)$ is already determined, and the decoder then enumerates new hypotheses $H'[v]$ by concatenating one hypothesis ($\mathrm{Join}(\cdots)$ in line 9) in $H[v']$ and a phrase pair ($\mathrm{PhrasePairs}(\cdots)$ in line 8) connecting v' and v. Note that $H[\cdot]$ and $H'[\cdot]$ may encode additional state information for features requiring non-local contexts, e.g., history of an n-gram language model. Finally, only K top scored hypotheses are preserved in $H[v]$ by applying a beam search strategy ($\mathrm{Prune}(\cdots)$ in line 13).

$\mathrm{IsCandidatePath}(v', v)$ checks whether at least one path exists between v and v' in the original preordering candidates in order to avoid enumerating spurious many phrase pairs.

Each hypothesis $h \in H[v]$ has a score calculated from its feature functions, which are used by $\mathrm{Prune}(\cdots)$ and $\mathrm{BestResult}(\cdots)$ to choose better hypothesis. We used the weighted linear combination for the scoring policy:

$$\mathrm{Score}(h) := \boldsymbol{w}^\top \boldsymbol{f}(h), \tag{1}$$

where $\boldsymbol{w}$ is a weight vector and $\boldsymbol{f}(h)$ is a set of feature functions for each hypothesis h, e.g., features associated with phrase pairs or extra features, such as n-gram language models.

We add scores for each existing path between v' and v in the preordering lattice according to the confidence of preordering candidates, which are used as an additional feature during decoding. After numbers of preliminary experiments, we adopted the product of maximum preordering confidence score and the ratio of phrase length based on our preliminary studies:

$$f(p) := \frac{|p|}{I} \cdot \max_n \gamma(n, p), \tag{2}$$

$$\gamma(n, p) := \begin{cases} C_n, & \text{if } p \subset \mathrm{Lattice}(A_n) \\ -\infty, & \text{otherwise,} \end{cases} \tag{3}$$

where p represents an arbitrary path over the preordering lattice. $-\infty$ means the decoder never choose the path p, and this formulation corresponds to $\mathrm{IsCandidatePath}$ condition in Algorithm 2.

Compared with the conventional decoding method (Zens and Ney, 2008), the proposed method can eliminate some complex score calculations, e.g., rest cost estimation and decoding-time reorderings, because each path in the reordering lattice holds complete information of the word order. As a result, the proposed method makes the decoding algorithm simpler than the conventional method.

4 Experiments

4.1 Experimental settings

We evaluated our proposed method under the settings of translating into English. We chose 11 language pairs consisting of 6 European languages (Fr/De/It/Pt/Ru/Es) and 5 Asian languages (Ar/Zh/Ja/Ko/Tr), which have different linguistic characteristics when compared with English.

For the training data, we used a parallel corpus by mining from the Web using an in-house crawler. The corpus contains 9.5M sentences and 160M words on average, at least 8.0M sentences and 140M words for each language pair. For the development/test data, we separately sampled and manually translated 3,000/5,000 sentences from other data sources on the Web for each language pair. All hyperparameters for each method are optimized using the development data and final evaluation is performed using the test data. During word alignment, IBM Model 1 (Brown et al., 1993) and HMM alignment (Vogel et al., 1996) were performed using one-best preordered source sentences and corresponding target sentences. The phrase table was built according to the alignment results, and shared with all decoding methods. For the English language model, a 4-gram model with stupid backoff smoothing (Brants et al., 2007) was built and commonly used for all settings. Each configuration of the word alignment and the language model was decided according to the preliminary experiments on the baseline system.

For the baseline system, we employed a standard PBMT system, similar to that of (Och and Ney, 2004) with a lexical reordering model (Zens and Ney, 2006) enhanced by a state-of-the-art preordering method based on bracketing transduction grammar (Nakagawa, 2015). We used similar decoding strategy and other basic feature functions to Moses (Koehn et al., 2007) except some neural lexical features such as NNJM (Devlin et al., 2014). Only one-best preordering candidate is used for the baseline system. We chose the best distortion limit of the baseline system for each language pair by the BLEU (Papineni et al., 2002) score on the development data.

We also compared the reranking method (Li et al., 2007), which translates all preordering candidates using conventional PBMT (our baseline system) and chose one with the best score. To do that, we used simple linear interpolation between decoder's score D and preordering confidence C with a hyperparameter λ as follows:

$$\mathrm{Score}(C, D) := \lambda \cdot C + (1 - \lambda) \cdot D. \tag{4}$$

We varied the number of preordering candidates (1, 2, 4, 8, 16, 32, 64-bests) for the proposed method and the reranking method, and chose the one with the best BLEU on the development data. For the reranking method, we trained two variants by differentiating distortion limits, a system sharing the same limit with the PBMT baseline and those with 0, in order to examine the effects of preordering and decoding-time reordering.

For all methods, we used lattice-based Minimum Error-rate Training (MERT) (Macherey et al., 2008) to optimize weights of features.

Evaluation is carried out by BLEU using all test data, and subjective evaluation with 7-grade (0 to 6) Likert scale about translation acceptance using 400 randomly selected samples from the test data in each language pair.

4.2 Results and Discussion

Figure 2 shows the number of nodes in actual preordering lattices generated from each source sentence in Japanese-English test data under 64-best preordering candidates. Upper group in this graph shows the number of unmerged nodes in which nodes are not shared when combining multiple preordering candidates in Algorithm 1, and lower group shows the number of merged nodes. The numbers directly reflect actual computation for decoding. There are averagely 10 times fewer nodes in merged preordering lattices, so our lattice construction of Algorithm 1 efficiently suppress the complexity of decoding when compared with directly using each preordering candidate independently.

Table 1 shows BLEU scores of the PBMT baseline, proposed method, and reranking method, respectively. The table also shows distortion limits (DL) for the PBMT baseline, and numbers of preordering candidates (N) for the proposed method and the reranking method. First, there are roughly the same

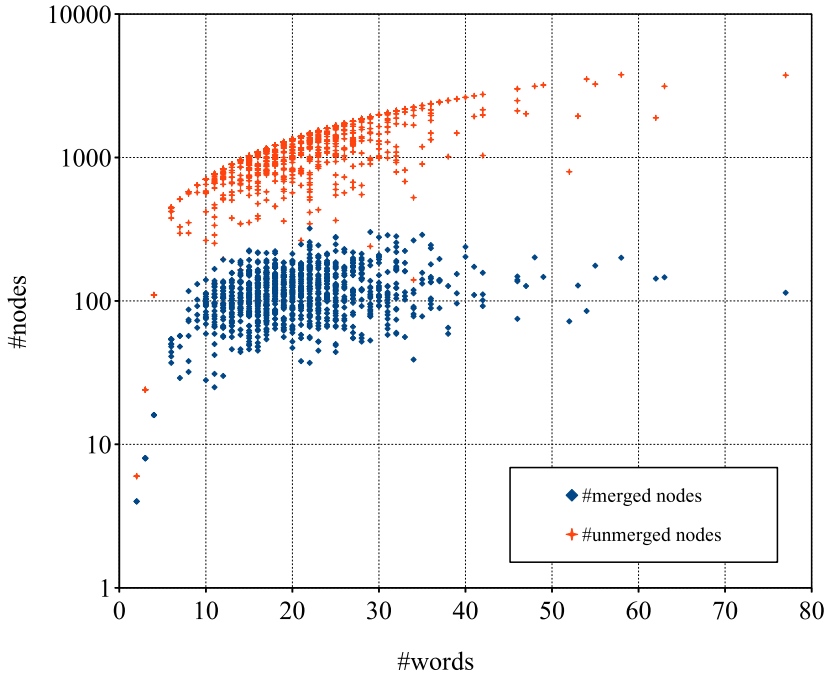


Figure 2: Number of nodes in preordering lattices in Japanese-English translation.

tendencies between the proposed method and the reranking method with similar BLEU improvement, and their systems averagely improves BLEU scores against the PBMT baseline in most language pairs. These results clearly indicate that multiple preordering candidates can largely improve the translation accuracy. In addition, we also see that there are high variance of N mainly in the reranking method. This tendency might come from the accuracy of the preordering method, i.e., if the preordering could perform well, then we require only few preordering candidates to generate accurate translations, and large N introduces less information. Actually, we observed that there is BLEU satulation in some language pairs when using large N, which means low-rank preordering candidates are rarely used to the final translation, according to the language pair.

In comparison between the proposed method and two reranking systems (DL>0 or $=0$), the proposed method without distortion (DL$=0$) often achieves higher BLEU score than DL>0. We conjecture that the tendencies may come from the use of better preordering among multiple candidates instead of a distortion-wise decoding-time reordering. These results clearly show that the decoding-time reordering is not necessary if better reordering is encoded in a preordering lattice.

We also see that there are comparatively higher BLEU improvements when translating from Ja/Ko/Tr than other languages. We speculate that these tendencies come from the grammatical characteristics of source languages. For example, Japanese is one of languages with high flexibility of word order, and the ambiguity may make it difficult to estimate correct preordering. In this case, the use of multiple preordering candidates is a straight-forward way to avoid this problem.

Table 2 shows the results of subjective evaluation for the proposed method against the PBMT baseline. We evaluated statistical significance of each system via t-test of difference between two averages, and this table shows their two-tailed p-values. Note that $\emptyset$ represents some small values ($p < 0.001$). We also included the change rate of these systems, which represents the amount of different translations by both PBMT baseline and the proposed method.

In this table, the proposed method achieves better translations with statistical significance ($p < 0.05$). We can also see that, in 5 Asian to English settings, the proposed method also achieved high translation accuracies under the subjective evaluation, although the proposed method generates divergent translations compared with the PBMT baseline. These languages have more large divergences from English,

Table 1: BLEU scores of each method/language.

Language	PBMT		Proposed			Reranking (DL>0)			Reranking (DL$=0$)		
	BLEU	DL	BLEU	Δ	N	BLEU	Δ	N	BLEU	Δ	N
Ar-En	36.81	6	36.99	+0.18	64	37.28	+0.47	16	36.94	+0.13	32
Zh-En	30.12	4	31.14	+1.02	64	30.90	+0.78	64	31.36	+1.24	64
Fr-En	33.10	5	34.03	+0.93	64	33.98	+0.88	32	34.13	+1.03	64
De-En	30.36	6	31.05	+0.69	32	31.53	+1.17	16	31.35	+0.99	32
It-En	37.65	6	38.22	+0.57	64	38.70	+1.05	16	38.40	+0.75	8
Ja-En	15.51	5	16.68	+1.17	32	17.01	+1.50	64	16.58	+1.07	32
Ko-En	20.32	5	22.34	+2.02	64	22.75	+2.43	64	22.86	+2.54	64
Pt-En	40.66	6	41.43	+0.77	64	41.48	+0.82	64	41.38	+0.72	64
Ru-En	25.45	6	25.79	+0.34	64	26.12	+0.67	16	25.55	+0.10	32
Es-En	35.50	5	36.11	+0.61	64	34.96	−0.54	8	36.42	+0.92	64
Tr-En	26.71	6	28.87	+2.16	64	29.27	+2.56	32	29.05	+2.34	64

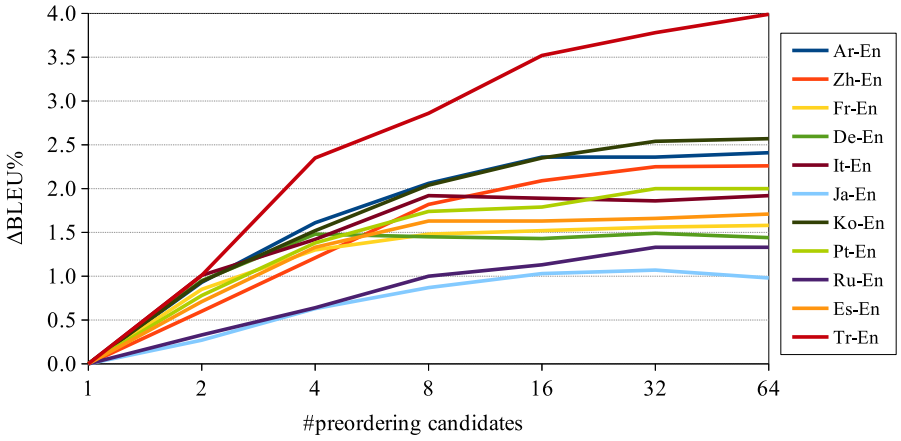
Table 2: Results of subjective evaluation.

Language	Score (PBMT)	Score (Proposed)	Δ	p-value	Change rate%
Ar-En	3.998	4.128	+0.130	0.028	66.14
Zh-En	3.135	3.278	+0.143	0.024	77.94
Fr-En	4.278	4.563	+0.285	$\emptyset$	36.81
De-En	3.902	4.259	+0.358	$\emptyset$	39.80
It-En	4.286	4.429	+0.143	0.046	35.54
Ja-En	2.943	3.238	+0.295	$\emptyset$	80.33
Ko-En	2.900	3.139	+0.239	$\emptyset$	70.44
Pt-En	4.392	4.642	+0.250	0.004	32.64
Ru-En	4.003	4.160	+0.158	0.029	41.61
Es-En	4.237	4.262	+0.025	0.712	48.78
Tr-En	3.150	3.553	+0.403	$\emptyset$	79.18

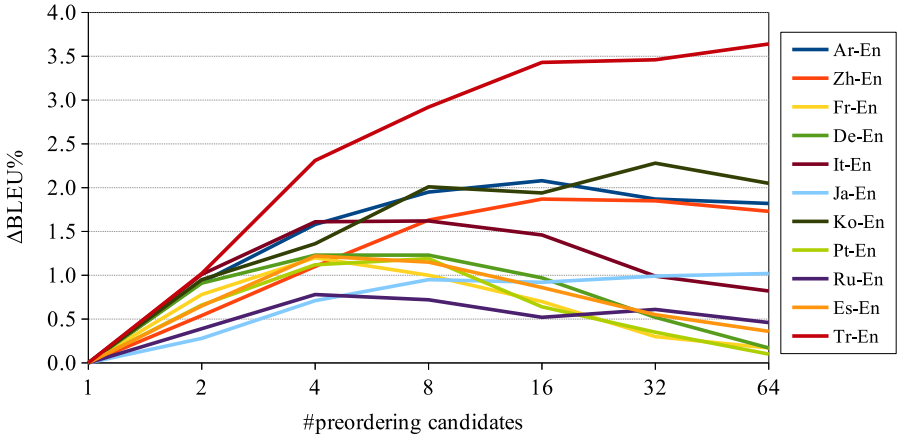
and estimating correct preorderings is more difficult than European languages. By these results, we can also say that the proposed method performs more effectively than PBMT baseline under BLEU and subjective evaluation.

Figure 3 shows BLEU changes of the proposed method by increasing the number of preordering candidates N. The baselines of these graphs are the BLEU score using only one-best preordering candidate, and these scores are roughly similar to a conventional PBMT system with DL$=0$. Figure 3(a) shows cases that use preordering confidence scores described in Section 2 as an additional feature, and Figure 3(b) shows cases of ignoring those scores. In Figure 3(a), the proposed method improves translation accuracy by increasing the number of preordering candidates in nearly all language pairs. Figure 3(a) also shows the BLEU satulation when using large N described in the previous paragraph. And in Figure 3(b), there are non-negligible BLEU reduction by using many preordering candidates. This tendency is expected, because ignoring confidence scores of preordering candidates implies treating all preordering candidates with the same importance, and the decoder have finally chosen the hypothesis with accidentally high scores by other features. Thus, introducing preordering confidence into decoding features is effective to prevent these kind of errors and guarantee translation accuracies.

Figure 4 shows mean decoding times of the PBMT baseline and the proposed method in Japanese-English setting with various decoding parameters. In the PBMT baseline, we changed both distortion limit (0 to 6) and beam width for each coverage of source sentence (1, 2, 4, 8,16 ,32 ,64 ,128). In the proposed method, we varied the number of preordering candidates (1, 2, 4, 8, 16, 32, 64) and beam width as same as PBMT baseline. Basically, increasing distortion limit or the number of preordering candi-



(a) With path score



(b) Without path score

Figure 3: BLEU changes according to the number of applied preordering candidates.

dates require much computation amount. In this figure, the proposed method using many preordering candidates can achieve high translation accuracy, as well as the proposed method runs as same range of computation time as PBMT.

Table 3 shows some examples in Japanese-English setting. In the first and second examples, the proposed method achieves better translation by exploiting multiple candidates. However, the last example demonstrates a weakness in our method mainly caused by the low confidence in preordering decision, e.g., parallel phrases. In this case, the language model of "image information" is stronger than that of "information and images" and this difference of scores exceeds the preordering confidence. As a result, the decoder fails to choose correct preordering. Avoiding these kinds of problems should be one of our future work.

5 Conclusion

In this paper, we proposed a new phrase-based decoding method using multiple preordering candidates. Our method outperforms previous PBMT systems without using any decoding-time reordering.

In this study, we used only one preordering method. Our method can be easily extended to any preordering methods along as they can emit N-best preordering candidates with optional confidence scores. In addition, the proposed method further may be able to combine multiple preordering candidates from different preordering methods by introducing multiple path scores for each preordering methods. In future work, we will plan to evaluate the effect of using or combining other preordering methods for the proposed method.

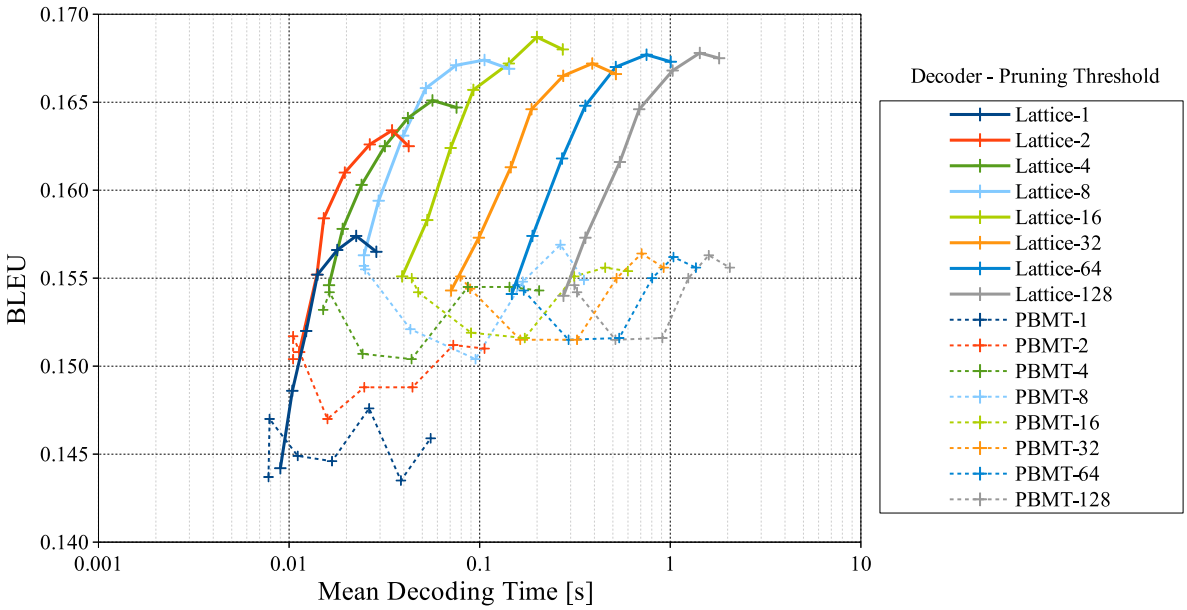


Figure 4: Relationship between decoding time and BLEU in Japanese-English translation.

Table 3: Translate examples of PBMT baseline and proposed method.

Type	Source Sentence/Translate	Score
Source	では, この問題をどうやって解決するつもりですか.	
PBMT	So, are you going to solve **how** this problem.	1
Proposed	So, **how** do you intend to solve this problem.	6
Source	私の車は, 私を含む全員がシートベルトを着用するまで駆動しません.	
PBMT	My car, **everyone including the I does not drive up** to wear a seat belt.	1
Proposed	My car **does not drive until everyone, including me** to wear a seat belt.	5
Source	技術革新により, 情報と画像をカードの表面に印刷できます.	
PBMT	By technological innovation, you can print the **information and images** on the card surface.	6
Proposed	By technological innovation, you can print the **image information** on the surface of the card.	4

Acknowledgement

Part of this work was supported by Grant-in-Aid for JSPS Fellows Grant Number 15J10649.

References

Thorsten Brants, Ashok C Popat, Peng Xu, Franz J Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proc. EMNLP-CoNLL*, pages 858–867.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Joachim Daiber, Miloš Stanojevic, Wilker Aziz, and Khalil Sima ' an. 2016. Examining the relationship between preordering and word order freedom in machine translation. In *Proc. WMT*, pages 118–130.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proc. ACL*, pages 1370–1380.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proc. ACL-HLT*, pages 1012–1020.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proc. EMNLP*, pages 848–856.

Teresa Herrmann, Jan Niehues, and Alex Waibel. 2013a. Combining word reordering methods on different linguistic abstraction levels for statistical machine translation. In *Proc. SSST*, pages 39–47.

Teresa Herrmann, Jochen Weiner, Jan Niehues, and Alex Waibel. 2013b. Analyzing the potential of source sentence reordering in statistical machine translation. In *Proc. IWSLT*.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2010. Head finalization: A simple reordering rule for SOV languages. In *Proc. WMT-MetricsMATR*, pages 244–251.

Jie Jiang, Jinhua Du, and Andy Way. 2011. Incorporating source-language paraphrases into phrase-based smt with confusion networks. In *Proc. SSST*, pages 31–40.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. NAACL-HLT*, pages 48–54.

Philipp Koehn, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, David Talbot, and Michael White. 2005. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *Proc. IWSLT*, pages 68–75.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL*, pages 177–180.

Chi-Ho Li, Minghui Li, Dongdong Zhang, Mu Li, Ming Zhou, and Yi Guan. 2007. A probabilistic approach to syntax-based reordering for statistical machine translation. In *Proc. ACL*, pages 720–727.

Wolfgang Macherey, Franz Josef Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proc. EMNLP*, pages 725–734.

Tetsuji Nakagawa. 2015. Efficient top-down btg parsing for machine translation preordering. In *Proc. ACL-IJCNLP*, pages 208–218.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. 2012. Inducing a discriminative parser to optimize machine translation reordering. In *Proc. EMNLP-CoNLL*, pages 843–853.

Jan Niehues and Muntsin Kolss. 2009. A POS-based model for long-range reorderings in SMT. In *Proc. WMT*, pages 206–214.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proc. ACL*, pages 311–318.

Josh Schroeder, Trevor Cohn, and Philipp Koehn. 2009. Word lattices for multi-source translation. In *Proc. EACL*, pages 719–727.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proc. COLING*, pages 836–841.

Fei Xia and Michael McCord. 2004. Improving a statistical mt system with automatically learned rewrite patterns. In *Proc. COLING*, pages 508–514.

Richard Zens and Hermann Ney. 2006. Discriminative reordering models for statistical machine translation. In *Proc. WMT*, pages 55–63.

Richard Zens and Hermann Ney. 2008. Improvements in dynamic programming beam search for phrase-based statistical machine translation. In *Proc. IWSLT*, pages 198–205.

Zhongyuan Zhu. 2014. Weblio pre-reordering statistical machine translation system. In *Proc. WAT*, pages 33–38.

Hand in Glove: Deep Feature Fusion Network Architectures for Answer Quality Prediction in Community Question Answering

Sai Praneeth Suggu[⋆] **Kushwanth N. Goutham**[⋆]
Manoj K. Chinnakotla[†] **Manish Shrivastava**[⋆]

[⋆]IIIT Hyderabad, India
{suggusai.praneeth,kushwanth.naga}@research.iiit.ac.in
m.shrivastava@iiit.ac.in

[†]Microsoft, India
manojc@microsoft.com

Abstract

Community Question Answering (cQA) forums have become a popular medium for soliciting answers to specific user questions from experts and experienced users in a given topic. However, for a given question, users sometimes have to sift through a large number of low-quality or irrelevant answers to find out the answer which satisfies their information need. To alleviate this, the problem of Answer Quality Prediction (AQP) aims to predict the quality of an answer posted in response to a forum question. Current AQP systems either learn models using - a) various hand-crafted features (HCF) or b) Deep Learning (DL) techniques which automatically learn the feature representations.

In this paper, we propose a novel approach for AQP known as - *"Deep Feature Fusion Network (DFFN)"* which combines the advantages of both hand-crafted features and deep learning based systems. Given a question-answer pair along with its metadata, a DFFN architecture independently - a) learns features using the Deep Neural Network (DNN) and b) computes hand-crafted features leveraging various external resources and then *combines them* using a fully connected neural network trained to predict the quality of the given answer. DFFN is an end-end differentiable model and trained as a single system. We propose two different DFFN architectures which vary mainly in the way they model the input question/answer pair - a) DFFN-CNN which uses a Convolutional Neural Network (CNN) and b) DFFN-BLNA which uses a Bi-directional LSTM with Neural Attention (BLNA). Both these proposed variants of DFFN (DFFN-CNN and DFFN-BLNA) achieve *state-of-the-art performance* on the standard SemEval-2015 and SemEval-2016 benchmark datasets and outperforms baseline approaches which individually employ either HCF or DL based techniques alone.

1 Introduction

Community Question Answering (cQA) forums (such as *Yahoo! Answers*, *Stack Overflow*, *etc.*) have become a popular medium for many internet users to get precise answers or opinions to their questions from experts or other experienced users in the topic. Such forums are usually open, allowing any user to respond to a given question. As a result, for a question posed by the user, the quality of response often varies a lot - ranging from highly precise and detailed answers given by authentic users to highly imprecise or non-comprehensible one-word or single line answers posted by spammy and other non-serious users. This severely hampers the effectiveness of cQA forums since users will have to sift through a large number of irrelevant posts to find the answer which satisfies their information need. To alleviate this problem, cQA forums often include feedback mechanisms such as votes, ratings *etc.* for evaluating the quality of answers and users. Such user feedback could also be used as signals for ranking multiple answers for given a question. However, popularity based signals (votes, ratings) are often prone to spam

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1429–1440, Osaka, Japan, December 11-17 2016.

due to users who may artificially inflate their ratings, votes with the help of other users whom they know. To overcome the above problems, recent approaches (Tran et al., 2015; Hou et al., 2015; Nicosia et al., 2015; Yi et al., 2015; Wang et al., 2009; Zhou et al., 2015; Severyn and Moschitti, 2015; Yu et al., 2014; Filice et al., 2016; Barrón-Cedeño et al., 2016) have focused on automatically ranking answers for a given question based on their quality.

The problem of answer quality prediction is defined as follows: Given a question Q and its set of community answers $C = \{A_1, A_2, \ldots, A_n\}$, rate the answers corresponding to their quality. The cQA tasks of SemEval-2015 (Task A) (Nakov et al., 2015) and SemEval-2016 (Task A) (Nakov et al., 2016) provide a universal benchmark dataset for evaluating research on this problem. In SemEval-2015, the answers are to be rated as {*good,potentially useful or bad*} and in SemEval-2016, the answers are to be rated as either {*good or bad*}.

Recent approaches for answer quality prediction can be categorized into - a) Hand-crafted Feature (HCF) based approaches (Tran et al., 2015; Hou et al., 2015; Nicosia et al., 2015; Yi et al., 2015; Wang et al., 2009; Filice et al., 2016; Barrón-Cedeño et al., 2016) or b) Deep Learning (DL) based approaches (Zhou et al., 2015; Severyn and Moschitti, 2015; Tymoshenko et al., 2016; Yu et al., 2014). HCF based approaches mainly rely on capturing various semantic and syntactic similarities between the question and answer and behavior of users using manual feature engineering. For computing these similarities, recent approaches have also leveraged external knowledge resources such as WordNet and other text corpora. DL based approaches, on the other hand, automatically learn the feature representations while learning the target quality scoring function. As a result, they are language-agnostic and don't require feature engineering or any external resources except for a large training corpus.

In this paper, we propose "Deep Feature Fusion Network (DFFN)" - a novel approach which combines HCF and DL based approaches. The DFFN architecture is designed as a Deep Neural Network (DNN) which takes the question, answer and their metadata as inputs and predicts the quality of answer as output. In the above architecture, HCF is also introduced separately as inputs into the overall DNN. We propose two different architectures of DFFN which mainly differ in the way they model the input question, answer pair - a) Convolutional Neural Network Model (DFFN-CNN) which employs a Convolutional Neural Network (CNN) to model the input question/answer pair and b) Bi-directional Long Short-Term Memory (LSTM) Network Model with Neural Attention (DFFN-BLNA) which uses a Bi-Directional LSTM Model with Neural Attention (BLNA) to model the question/answer pair. DFFN effectively leverages the advantage of both HCF and DL based approaches *i.e.* ability to - a) encode similarities between question-answer pair using external knowledge resources such as Wikipedia, Anchor Text information from Google Cross-Lingual Dictionary (GCD) and Clickthrough data and b) automatically learning features relevant to the target function. During training phase, given a question, answer pair along with its metadata and HCF, DFFN automatically learns deep features which are relevant for the target task using CNN or BLNA. Later, DFFN combines these deep features with HCF, which are computed using various external resources, for predicting the quality rating of the answer. The two proposed architectures of DFFN achieve *state-of-the-art performance* on the standard SemEval-2015 and SemEval-2016 benchmark datasets and also perform better than baseline approaches which individually employ either HCF or DL based techniques. In this context, the following are our main contributions:

- We propose a novel approach to combine resource-based hand-crafted and automatically learnt DL features for the answer quality prediction task.
- We also propose two different architectures of combining HCF and DL based features using CNNs and BLNA.
- Using the above novel architectures, we achieve state-of-the-art performance on SemEval 2015 and SemEval 2016 cQA answer quality prediction tasks.

The rest of the paper is organized as follows: Section 2 discusses the related work in this area. Section 3 presents our contribution DFFN in detail. Section 4 discusses our experimental set-up. Section 5 presents our results and finally Section 6 concludes the paper.

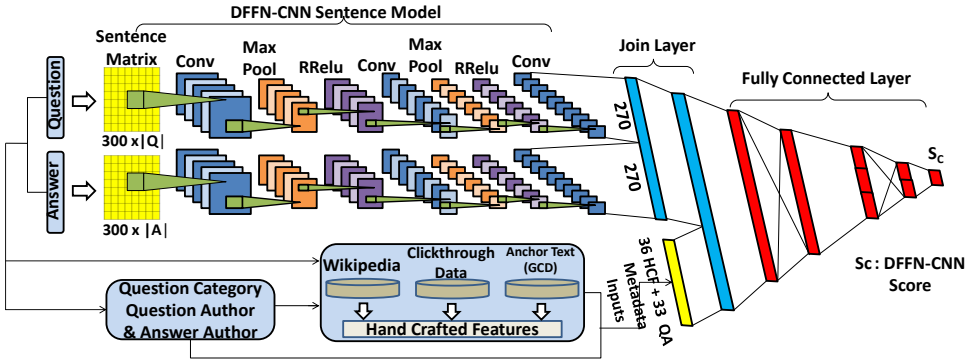


Figure 1: System Architecture of Deep Feature Fusion Network - Convolutional Neural Network with Neural Attention (DFFN-CNN)

2 Related Work

AQP in cQA forums has been researched a lot in the IR community. (Jeon et al., 2006) employ non-textual features such as clicks, print counts, copy counts *etc.* to predict the quality of an answer in a cQA forum. (Liu et al., 2008) investigate a slightly related problem i.e. predicting whether an asker would be satisfied with the answers provided so far to the given question. (Burel et al., 2012) have used a combination of content, user and thread related features for predicting answer quality. (Dalip et al., 2013) propose a learning to rank approach for AQP using eight different groups of features. (Yao et al., 2013; Wang and Manning, 2010) used CRF models with extracted features for AQP. (Li et al., 2015) studied the various factors such as shorter length, authors reputation which lead to a high answer quality rating as rated by peers.

More recently, (Tran et al., 2015) made use of topic models, word vectors and other hand crafted rules to train a SVM classifier for AQP. (Hou et al., 2015) made use of statistics like avg. word length of a sentence (question or answer), sentence length with other hand-crafted features to train an ensemble of classifiers for AQP. (Wang et al., 2009) use Bayesian logistic regression and link prediction models for AQP. (Filice et al., 2016) used kernel based features for AQP.

(Wang and Nyberg, 2015) apply a combination of stacked Bi-Directional LSTMs and keyword matching. (Nicosia et al., 2015) have used lexical similarity between word n-grams, tree kernels, word-embeddings and other hand crafted features for AQP. (Severyn and Moschitti, 2015) used a CNN to automatically learn features for matching short text pairs. (Zhou et al., 2015) used a 2-dimensional CNN to represent a question-answer pair and ranked the representations using a RNN.

Our current work resembles the work of (Wu et al., 2016), in the computer vision community, who employ the idea of combining hand-crafted features and deep features for person re-identification task. However, in our case, the idea of using hand-crafted features is motivated by the availability of large similarity resources such as Wikipedia text, Anchor text of Google Cross-Lingual Dictionary and Click-through data which could be leveraged to infer richer syntactic and semantic similarities between textual elements.

3 Deep Feature Fusion Network (DFFN)

The central idea in a DFFN is to design a Deep Neural Network (DNN) which takes the question (Q), answer (A) and its metadata (MD) as inputs and predicts the quality of an answer as output. DFFN also computes various HCF between Q, A and MD using external resources such as Wikipedia text and Anchor text of Google Cross-Lingual Dictionary (GCD) and Click-through data. These HCFs are also included into the overall DNN as inputs. We propose two different architectures of DFFN depending on the way in which the Q, A pairs are modeled in a NN - a) DFFN-CNN which employs a Convolutional Neural Network (CNN) to model the input question-answer pair and b) DFFN-BLNA which uses a Bi-Directional LSTM Model with Neural Attention (BLNA) to model the question-answer pair. Figures 1,2

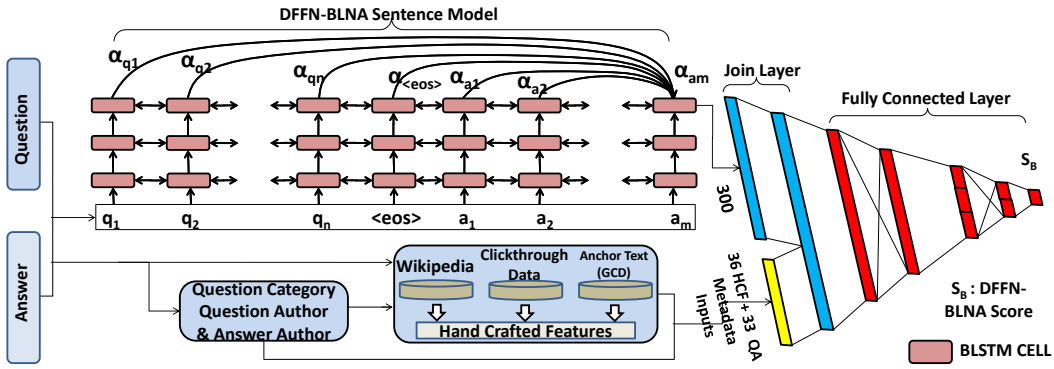


Figure 2: System Architecture of Deep Feature Fusion Network - Bi-directional Long-Short Term Memory Network with Neural Attention (DFFN-BLNA)

depict the architectures of the two proposed variants. Both these variants are end-end differentiable and hence the training is performed end-end.

DFFN-CNN comprises of two parallel CNN based sentence models for the question and answer while DFFN-BLNA has a sequential Bi-directional Long Short-Term Memory Network model with Neural Attention for the question and answer together. Let CNN-FR and BLNA-FR be the deep feature representations generated by using CNN and BLNA respectively. CNN-FR and BLNA-FR are individually joined with HC-FR and metadata and are given as input to a Fully Connected Neural Network (FCNN) which predicts the score representing answer quality. We will now discuss DFFN in detail.

3.1 Sentence Model

DFFN has a sentence model which projects a sentence (question/answer) into the semantic space and learns a good intermediate representation of the given question/answer. The different architectures DFFN-CNN and DFFN-BLNA vary in the way they perform sentence modeling. Here is a brief description of their sentence models:

3.1.1 DFFN - Convolutional Neural Network (DFFN-CNN)

In this architecture, the sentence model is a deep Convolutional Neural Network (CNN). CNN extract features independent of the position in the sentence to create (sub-)sentence representations. CNN consists of sentence matrix and multiple convolutional, pooling and non-linearity layers as in Figure 1 .

Sentence Matrix: The input to the sentence matrix is a vector of words from the sentence (question/answer) $s = [w_1, w_2, ...w_{|s|}]$. We build the sentence matrix by mapping each word w_i in the sentence to its corresponding word embedding in d dimensions. Word embeddings represent similar words by similar vectors and, thus, identify synonyms and other important context words.

We use GLoVE (Pennington et al., 2014) based embeddings of 300 dimensions to map the words in the question and answer. We limit the size of the sentence upto certain threshold. We ignore the words in the sentence after a certain threshold if the length of the sentence is greater than the threshold and pad zeros upto the threshold if the length of the sentence is less than the threshold. The sentence matrix is given as input to the convolutional layer.

Convolution: Convolution is an operation where the feature map (input sentence matrix) and the convolution filter mix together to form a transformed feature map. The convolutional layer extracts patterns i.e, discriminative word sequences that are common in the input train sentences. The convolutional layer is applied on the sentence matrix by convolving a filter with weights $F \in R^{h \times w}$ where h is the filter height and w is the filter width. A filter consisting of a layer of weights is applied to a small patch of sentence matrix to get a single unit as output. The filter is slided across the sentence matrix and the outputs of each patch are combined to get the resultant transformed feature map as the output.

Max Pooling: Max pooling extracts globally most relevant features through local convolution. Max pooling performs a type of non-linear down-sampling. It combines the information and reduce the size

of feature map. It partitions the output of the convolutional layer into small non-overlapping slices and independently operates on every slice by taking the maximum value in each slice as the value in the output of reduced size. We apply max pooling layer on the top of output given by the convolutional layer to extract crucial local features and form a reduced size feature map representation.

Non-Linearity: We use Randomized Leaky Rectified Linear unit (RReLU) (Xu et al., 2015) to learn non-linear decision boundaries. It is a randomized version of leaky ReLU (Xu et al., 2015). RReLU is applied to every element of the output of the max pool layer, thus the resultant feature map will be of the same dimension as the input feature map.The sentence matrix is convolved through multiple convolution, pooling and non-linearity layers to get the feature representations of the question/answer. Using this variation of sentence model, we get the individual feature representations (270 dimensions) of the question and answer. These are concatenated to produce a combined feature representation (540 dimensions).

3.1.2 DFFN - Bi-directional LSTMs with Neural Attention (DFFN-BLNA)

Although, CNN extracts similar patterns on all the patches of the sentence matrix but they do not capture sequential relationships that exists between question-answer pair. LSTMs are memory models which overcome this limitation by feeding the hidden layers from the previous step as an additional input into the next step. In DFFN-BLNA architecture as shown in Figure 2, in stead of a CNN, we use a Bi-directional Long-Short Term Memory (BLSTM) Network with Neural Attention (Bahdanau et al., 2014), for modeling the sentences of question and answer. A question-answer sequence is given as input to BLNA where the sequence is passed through a Bi-directional LSTM Network and the outputs at each step are attended with Neural Attention mechanism. Here, we describe the architecture in more detail.

Long-Short Term Memory (LSTMs) (Hochreiter and Schmidhuber, 1997) are variants of Recurrent Neural Network (RNN) (Werbos, 1990; Rumelhart et al., 1988) architectures which - a) overcome the vanishing and exploding gradients problem of conventional RNNs and b) have the ability to capture long-term dependencies between symbols of a sequence using their gating mechanism which controls information flow. LSTMs only utilize previous context without making use of future context. To overcome this issue, Bidirectional LSTMs (BLSTMs) learn the sequential patterns from both forward and backward directions and then combine information from both directions. The drawback of LSTMs or BLSTMs is that we represent a very long sentence as a single vector which is the output of the last time step. However, using BLSTM with Neural Attention (NA) mechanism, we represent the sequence of vectors as a combined weighted representation vector by selectively attending to the past outputs.

We map the words of the question and answer to their corresponding vector representations using GLoVE embeddings. <eos> is a special symbol used to separate question and answer. A question-answer sequence is generated by concatenating question sentence, <eos> and answer sentence. A BLSTM has two LSTMs that read the QA sequence in both forward and backward directions. At each time step, the output vector is generated by combining the output vectors of two LSTMs, thereby allowing it to consider the contextual information across the entire question-answer sequence i.e. both the question and answer sentence. The neural attention mechanism represents sequence of vectors as a combined weighted representation vector by selectively attending to the past outputs. Thus at each time step, DFFN-BLNA considers the whole context of question and answer and adaptively attends to the subset of past outputs of the BLSTM Network which contributes in better modeling the similarity between question and answer.

Let $Q = [q_1, q_2, ..., q_n]$ be a question of length n and $A = [a_1, a_2, ..., a_m]$ be an answer of length m, then total number of steps in BLSTM will be n+m+1 (additional step is for <eos> after the question). Let $Y = [y_1, y_2, ...y_n, y_{<eos>}, y_{n+2}, y_{n+3}.....y_{n+m+1}]$ be output of BLSTM Network without attention. The combined weighted vector representation c generated using attention mechanism is computed as

$$c = \sum_{i=1}^{n} \alpha_i y_i + (\alpha_{n+1} y_{n+1}) + \sum_{i=n+2}^{m} \alpha_i y_i \tag{1}$$

and $a(y_i, y_k)$ is a latent alignment model which outputs higher score if y_i is useful in capturing the

similarity between question answer pair. where

$$\alpha_i = \frac{\exp(a(y_i, y_{n+m+1}))}{\sum_{j=1}^{n+m+1} \exp(a(y_j, y_{n+m+1}))} \tag{2}$$

Using this variation of sentence model, we generate a combined feature representation of question and answer (300 dimensions). In each of the above architectures, the feature representation derived from the sentence model is combined with the hand crafted features and metadata and is given as input to the Fully Connected Neural Network. We describe these in detail in the following subsections.

3.2 Hand Crafted Features (HCF)

The question and answer text usually consists of several Named Entities (NEs) and concepts along with their various variants. For example, the cricketer *Sachin Tendulkar* could be referred to as *Sachin*, *Tendulkar*, *The Little Master etc.* Such variants are hard to capture using CNN or BLNA based features alone. Hence, we make use of resources such as Wikipedia text, Anchor text of Google Cross-Lingual Dictionary (GCD), Named Entity Recognizers (NER) and Clickthrough data to come up with hand-crafted features which can capture such rich similarities. We also observe that user behavior and specific patterns on metadata and question-answer text are useful. We use these features to compute individual similarity scores between question and answer and combine these scores as Hand Crafted Features to give them as input to the Fully Connected Neural Network. We describe the details of the features below:

3.2.1 Wikipedia Based Features

In this section, we describe the similarity features which are computed by using Wikipedia as a resource.

TagMe Similarity: We extract TagMe concepts from the question and answer by mapping the question and answer to their corresponding Wikipedia page titles using TagMe (Ferragina and Scaiella, 2010). TagMe identifies meaningful substrings in an unstructured text and links them to their relevant wikipedia pages. We compute the similarity between two TagMe concepts using WikiMiner (Milne and Witten, 2013). WikiMiner computes similarity between two wikipedia pages based on the number of common inlinks and outlinks between them. Similarity between question and answer, represented by TagMe concepts, using WikiMiner is computed as the mean average of the similarity between pairs of TagMe concepts (one each from the question and the answer) as in Equation 3

$$qa_{sim} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} sim(c_i, c_j)}{nm} \tag{3}$$

where qa_{sim} is the similarity between question and answer based on TagMe Similarity n, m are the number of TagMe concepts in the question and answer respectively, c_i, c_j are the i^{th} and j^{th} TagMe concepts in the question and answer respectively, $sim(c_i, c_j)$ is the similarity between c_i and c_j calculated using WikiMiner.

Named Entities Similarity: We extract Named Entities from the question and answer, using Stanford CoreNLP NER Tagger (Toutanova et al., 2003) and compute the similarity between two Named Entities using a Google Cross-Lingual Dictionary(GCD) based similarity feature. The GCD based similarity between two Named Entities is computed as the ratio of number of wikipedia documents in which these two named entities co-occur in the top k retrieved documents when queried on GCD. Similarity between question and answer represented by Named Entities is calculated as in Equation 3 where we use Named Entities instead of TagMe concepts and GCD based similarity feature instead of WikiMiner to calculate the similarity between two Named Entities.

3.2.2 AnchorText based Features:

Google Cross-Lingual Dictionary (GCD) (Spitkovsky and Chang, 2012) is a string to concept mapping on the vast link structure of the web, created using anchor text from various pages across the web. A concept is an individual Wikipedia document. The text strings are the anchor texts that refer to these concepts. Thus, each anchor text string represents a concept.

We extract common and proper nouns from the question and answer using Stanford CoreNLP POS Tagger (Toutanova et al., 2003) and query them individually on GCD anchor texts to get top ten unique concepts related to question and answer. We calculate the similarity between two GCD concepts using WikiMiner. The similarity between question and answer represented by GCD concepts is calculated as in Equation 3 where we use GCD concepts instead of TagMe concepts.

3.2.3 Clickthrough Features

Sent2Vec Similarity: Sent2Vec maps a pair of short texts to a pair of feature vectors in a continuous, low-dimensional space. Sent2Vec performs the mapping using the Deep Structured Semantic Model (DSSM) built using Clickthrough data (Huang et al., 2013), or the DSSM with convolutional-pooling structure (CDSSM) (Gao et al., 2014; Shen et al., 2014).

We map the question and answer to vectors using both DSSM and CDSSM. We compute the Sent2Vec DSSM similarity between the question and answer as the cosine similarity between the vectors of question and answer obtained by using Sent2Vec performing the mapping of vectors using DSSM. Similarly by using CDSSM instead of DSSM we also compute the Sent2Vec CDSSM similarity between the question and answer.

Paragraph2vec Similarity: Paragraph2Vec(Le and Mikolov, 2014) allows to model vectors for text of any arbitrary length. It learns continuous distributed vector representations for pieces of texts. We train the para2vec model on the training data of the particular tasks only (SemEval'15 and SemEval'16) by treating each question-answer pair as a single document. We train only on the good question-answer pairs from the training data. A good question-answer pair is a pair in which answer is rated as a "good" answer for that question. We map the question and answer to vectors using para2vec and compute the similarity between the question and answer as the cosine similarity between their para2vec vectors.

3.2.4 Metadata Based Features

Author Reputation Score: We observed that the reputation of an answer author, within a forum plays a key role in determining the quality of answer. We capture this through a author reputation feature. We have two reputation features namely Good Reputation and Bad Reputation. Good reputation of an author is computed as the ratio of the number of good answers given by that author to the maximum number of good answers given by any individual author in the entire forum. Similarly, by using the number of bad answers instead of good answers, we compute a score for the bad reputation of an author. In addition, we also compute Good and Bad reputation scores of an author across each question category.

Is Answer Seeker?: We have a boolean feature to represent whether the answer (comment) is written by the person who has asked the question.

Authors' Response Pattern: We compute features based on whether the question author has commented before or after the present answer and if that comment by the question author is a question. Usually, the question author posts comments/questions below an answer if one is not satisfied with the current answer. Similarly we compute features based on whether the answer author has commented before or after the present answer and if that comment by the answer author is a question. Usually, the answer author posts further questions or comments below ones answer to seek additional information regarding the question or explain his answer more briefly. These features capture the behavior.

Miscellaneous: Besides, we extract and add features related to - a) statistics of each question category (number of good, potential and bad answers in that category) b) position of the answer. c) presence of URL, e-mail in the answer d) presence of question marks, exclamation marks in the answer e) boolean features for the presence of various emoticons such as happy (eg: ":)", ":D"), sad (eg: ":(" , ":'(") in the answer. We obtain the similarity scores and together call them as Hand-crafted features (36 dimensions). We join them as a vector and give them as input to Fully Connected Neural Network along with Metadata as described below.

3.3 Metadata Information

We observe that category of the question plays an important role in answer quality prediction as it may be easy to write good answers for some categories and difficult for some. We encode question category,

SemEval 2015			SemEval 2016			
Model	**F1**	**Acc.**	**Model**	**MAP**	**F1**	**Acc.**
DFFN-BLNA	**62.01***	**75.20***	DFFN-BLNA	**83.91***	**66.70***	**77.65***
DFFN-CNN	60.86	74.54	DFFN-CNN	81.77	65.75	76.42
JAIST	57.29	72.67	Kelp	79.19	64.36	75.11
HITSZ-ICRC	56.44	69.43	ConvKN	78.71	63.55	74.95
DFFN-BLNA *w/o* HCF	56.85	70.45	DFFN-BLNA *w/o* HCF	75.12	61.57	73.12
DFFN-CNN *w/o* HCF	56.06	69.79	DFFN-CNN *w/o* HCF	74.38	60.90	71.96
DFFN *w/o* CNN and BLNA	52.83	66.90	DFFN *w/o* CNN and BLNA	71.56	56,46	69.20
ICRC-HIT	53.82	73.18				

Table 1: Overall Results of DFFN on SemEval 15 and 16 datasets. Results marked with a * were found to be statistically significant with respect to the nearest baseline systems i.e top performing systems of SemEval-15 and 16 at 95% confidence level ($\alpha = 0.05$) when tested using paired two-tailed t-test.

question author and answer author using a logarithmic function and give them as input to the Fully Connected Neural Network.

3.4 Fully Connected Neural Network (FCNN)

The vector representations from the sentence model (540 dimensions from DFFN-CNN or 300 dimensions from DFFN-BLNA), the feature representations from HCF (36 dimensions) and direct inputs from Metadata (33 dimensions) are combined to get a single feature vector of 609 dimensions (DFFN-CNN) or 369 dimensions (DFFN-BLNA). This vector is given as input to FCNN consisting of fully connected layers. These layers model various interactions between the features present in the vector and finally output a score predicting the answer quality.

3.5 Training

The parameters of the network are learnt with an objective to maximize the accuracy of prediction given the target categories. For example, in SemEval-2015, the target categories were $\{good, potentially\ useful, bad\}$ and $\{good, bad\}$ in SemEval-2016 . For training, we used the training data provided in the SemEval 2015 (Nakov et al., 2015) and 2016 (Nakov et al., 2016) tasks which consists of question, answer, metadata along with their ideal quality rating. We tuned the DFFN parameters on the corresponding development sets of SemEval 2015 and 2016. We used adagrad (Duchi et al., 2011) and stochastic gradient descent (SGD) for optimization in DFFN-CNN and DFFN-BLNA respectively.

Given an input (p, t) where p is the predicted answer quality score by DFFN-CNN and t is the true label depicting answer quality, we used SmoothL1 loss criterions which is computed as:

$$loss_{smoothl1}(p,t) = \frac{1}{n} \times \begin{cases} 0.5 \times (p-t)^2, & \text{if } |p-t| < 1 \\ |p-t| - 0.5, & \text{if } |p-t| \geq 1 \end{cases}$$

t is 1 for good question-answer pair (answer labeled as *good* for that question) and -1 for bad question answer pair. (answer labeled as *bad* for that question). The model is trained by minimizing the loss function in a batch of size n. For DFFN-BLNA we used Mean Squared Error (MSE) as loss criterion.

4 Experimental Setup

We use the SemEval 2016 (Nakov et al., 2016) and SemEval 2015 (Nakov et al., 2015) datasets for our experiments as they exactly match our problem description. We use standard evaluation metrics - Mean Average Precision (MAP), F1 score and Accuracy. We compare our approach with the top two best performing systems from SemEval 2015 - JAIST (Tran et al., 2015) and HITSZ-ICRC (Hou et al., 2015) both of which use HCF based models. We also compare with ICRC-HIT (Zhou et al., 2015) as it uses a purely DL based model. Similarly, for SemEval 2016, we compare with their corresponding top two

Question and Answer	TL	DC	JA	HI	IH	Comment
Q: Hi friends, I have State Bank of India Debit card. I try to with draw the money to the atm. It's not accepted. Anybody know which bank atm will accept SBI debit card for withdraw the money ? **A:** dear all banks here accept all international (visa/master/diner club/american express) ATM's cards unless you activate international withdrawal from your mother bank in your mother country.	G	G	B	G	B	***TagMe*** links *visa, master, diner club, american express* to their wiki pages and finds out that they all belong to debit/atm/credit card class.
Q: Can anyone plz help me this problem? I need to send a mobile phone to (Jaipur) India. I contacted DHL but they are charging very high. Is there any other company like DHL? Plz specify... **A:** You can send by post office for cheap price (compare to Courier service)	G	G	B	P	P	***GCD*** similarity feature captures that *post office, DHL, courier* are linked to similar pages when they occur as anchor texts.
Q: What softwares are you using for downloading movies? I'm using limewire and utorrent. How about you? **A:** im using azureus client..limewire sucks (lol)	G	G	B	G	B	***TagMe*** links *azureus, limewire, utorrent* to their wiki pages and finds out that they all belong to movie torrent software class.
Q: I saw a little girl running by the streets , and she had a cat attached to her .....is that normal in this country? **A:** I saw a little girl running by the streets , and she had a parent attached to her .....is that normal in this country?	B	P	P	B	B	***Author Reputation*** gives neutral sim. score; author had written very few answers and had almost equal number of Good and Bad answers. ***Question-Authors' Response Pattern*** gives neutral;author has commented before and after this answer. ***Wiki based*** features gave high scores as question and answer are exactly same except for one word.

Table 2: Qualitative Analysis of DFFN-CNN Results with respect to other baseline approaches. **Note: G: Good, B: Bad, P: Potential; DC: DFFN-CNN, JA: JAIST, HI:HITSZ-ICRC IH:ICRC-HIT**

best performing systems - Kelp (Filice et al., 2016) and ConvKN (Barrón-Cedeño et al., 2016) both of which use kernel-based features.

5 Results and Discussion

Table 1 shows the overall results of DFFN-CNN and DFFN-BLNA on SemEval 2015 and SemEval 2016 datasets. Both the architectures perform better than the top systems across all the metrics. The improvement is higher in SemEval 2015 although the task is harder due to lesser training data and more granularity in target labels. We also observe that DFFN-CNN and DFFN-BLNA perform better than CNN (without HCF) or BLNA (without HCF). Also, the model outperforms hand-crafted feature based model (DFFN with HCF but without CNN or BLNA). Overall, the best performing model was found to be DFFN-BLNA as it more closely models and encodes the semantic dependencies in the QA pair. In Table 2, we present the qualitative analysis of DFFN-CNN architecture results with the best performing baselines on SemEval 2015 dataset. In Table 3, we present the qualitative analysis of DFFN-BLNA architecture results with the best performing baselines on SemEval 2016 dataset. Finally, in Table 4, we compare the performance of DFFN-CNN and DFFN-BLNA using some results from the above datasets.

6 Conclusion

We present a novel approach *"Deep Feature Fusion Networks (DFFN)"*, an end-to-end differentiable approach which combines HCF features into CNN and BLNA models for improving answer quality prediction. DFFN enriches the feature representations learnt through CNN and BLNA by introducing more similarity features computed using external resources such as Wikipedia text, Anchor text of Google Cross-Lingual Dictionary (GCD) and Clickthrough Data. As a result, we show that DFFN achieves *state-of-the-art performance* on the standard SemEval-2015 and SemEval-2016 benchmark datasets and shows better performance than baseline approaches which individually employ either HCF or DL based techniques.

7 Acknowledgement

We acknowledge the support of Crowdfire in the form of an International Travel Grant, to attend this conference.

Question and Answer	TL	DB	KE	CK	Comment
Q: I am very interested to know if there are any expatriate tennis clubs in Doha that anyone can join? I am at a decent standard and would like to play once / twice per week so joining a club would be ideal. If anyone would like a game then please drop me an e-mail and we can arrange something. **A:** We normally play tennis at Khalifa Tennis at least twice a week (Tuesday and Friday); but I would prefer to play for at least 3 times a week or even more. So; if you are interested; I could introduce you to some players so that we could play together.	G	G	B	B	DFFN-BLNA matches with keywords *"tennis","interested,"*. Also due to right intent/response matching.
Q: Has anyone in QL bought any laptop from Jarir bookstore on loan through Qatar Finance company? One more thing;what do you guys think about HP laptops? As i've never bought anything on loan through a bank or a finance company in Qatar **A:** Jarir has an arrangements with QFC that let the customers to purchase laptops and pay back in installments. but their formalities r a bit complex; a lot of documentations required; etc..	G	G	B	G	DFFN-BLNA identifies *Finance Company, Loan* are related to *installments, formalities* . NE *Jarir* matched in both Q&A
Q: My visa is issued and the agency told me it will be going to authenticate it in the embassy here in the philippines how long is the process??? after my visa is stamp what will be the next process?? **A:** If you are going through an agency in the Philippines like what happened to me; it will take at least 1 month of waiting. But in Doha; based on the processing of our PRO in the office; it only takes a week or less. Even for visa renewal; in one week i can have the original I.D.	G	G	G	B	*visa, agency, month, week* co-occur in wiki pages and anchor text of web pages. GCD identified them as similar.
Q: Qtel's settings for mobile internet? I cant seem to access the internet through my Iphone. I've called 111 and they gave me the settings which would be: General - Networks - Cellular Data Network - APN: gprs.qtel Still no signs of getting the internet going. Anyone else have this problem with Iphone? I'm on Shary value pack btw if that info helps. **A:** QTEL is difficult but Vodafone has it's settings on their website. It helped me on my iPhone w/ Vodafone sim.	B	G	B	G	DFFN-BLNA predicts incorrectly due to multiple NE *(Qtel, Iphone)* perfect matches.

Table 3: Qualitative Analysis of DFFN-BLNA Results with respect to other baseline approaches. **Note: G: Good, B: Bad; DB: DFFN-BLNA, KE: Kelp, CK: ConvKN**

Question and Answer	TL	DB	DC	Comment
Q: Hi; my wife was on a visit visa; today; her residency visa was issued; so i went to immigration and paid 500 so there is no need to leave the country and enter again on the residency visa. she has done her medical before for the visit visa extension; do we need to do the medical again for the residency visa? thanks **A:** Hi can u pls. help me ? I just want to know what is the requirements for the family visit visa here in Qatar i want to apply family visit visa for my wife and to my daughter. and also is it true that i can extend the visa up to 6 months? Is there any salary bracket requirements for this visa? I hope u can help me thanks	B	B	G	DFFN-CNN merely compares extent of similarity since keywords like *visa* match while DFFN-BLNA identifies question intent expressed in the sentence.
Q: What is best mall in Doha to buy good furniture? Where are best furniture stores and showrooms. **A:** There are several; my Favorite is Pan Emirates @ Salwa Road.	G	G	B	DFFN-BLNA identified intent of phrase *"There are several"* while DFFN-CNN does not find more explicit matches.
Q: I would like to get some opinion about online job recruitment sites like bayt; gulftalent etc.. Do they really consider the CV's send ? Has anybody got jobs via these online sites ? **A:** They look for key words to match against a given criteria. They have no means of assessing an individual or his/her real skills.	G	B	G	DFFN-CNN finds related keywords in the context such as *"job", "skills", "assessing", "criteria"*.

Table 4: Qualitative Comparison of DFFN-CNN and DFFN-BLNA Results. **Note: G: Good, B: Bad; DB: DFFN-BLNA, DC: DFFN-CNN**

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Shafiq Joty, Alessandro Moschitti, Fahad Al-Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. ConvKN at SemEval-2016 Task 3: Answer and Question Selection for Question Answering on Arabic and English Fora. In *SemEval-2016*. ACL.

Grégoire Burel, Yulan He, and Harith Alani. 2012. Automatic Identification of Best Answers in Online Enquiry Communities. In *9th Extended Semantic Web Conference, ESWC 2012*.

Daniel Hasan Dalip, Marcos André Gonçalves, Marco Cristo, and Pavel Calado. 2013. Exploiting User Feedback to Learn to Rank Answers in QA Forums: A Case Study with Stack Overflow. In *SIGIR '13*. ACM.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.*

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). CIKM '10. ACM.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KeLP at SemEval-2016 Task 3: Learning Semantic Relations between Questions and Answers. In *SemEval-2016*. ACL.

Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, Li Deng, and Yelong Shen. 2014. Modeling Interestingness with Deep Neural Networks. EMNLP.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November.

Yongshuai Hou, Cong Tan, Xiaolong Wang, Yaoyun Zhang, Jun Xu, and Qingcai Chen. 2015. HITSZ-ICRC: Exploiting Classification Approach for Answer Selection in Community Question Answering. In *SemEval 2015*. ACL.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning Deep Structured Semantic Models for Web Search using Clickthrough Data.

Jiwoon Jeon, W. Bruce Croft, Joon Ho Lee, and Soyeon Park. 2006. A Framework to Predict the Quality of Answers with Non-textual Features. In *SIGIR '06*. ACM.

Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. *CoRR*, abs/1405.4053.

Lei Li, Daqing He, Wei Jeng, Spencer Goodwin, and Chengzhi Zhang. 2015. Answer Quality Characteristics and Prediction on an Academic QA Site: A Case Study on ResearchGate. WWW '15 Companion. ACM.

Yandong Liu, Jiang Bian, and Eugene Agichtein. 2008. Predicting Information Seeker Satisfaction in Community Question Answering. In *SIGIR '08*. ACM.

David Milne and Ian H. Witten. 2013. An Open-source Toolkit for Mining Wikipedia. *Artif. Intell.*

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2015. Semeval-2015 Task 3: Answer Selection in Community Question Answering. In *SemEval '15*. ACL.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. SemEval-2016 Task 3: Community Question Answering. SemEval '16. ACL.

Massimo Nicosia, Simone Filice, Alberto Barrón-Cedeño, Iman Saleh, Hamdy Mubarak, Wei Gao, Preslav Nakov, Giovanni Da San Martino, Alessandro Moschitti, Kareem Darwish, Lluís Màrquez, Shafiq Joty, and Walid Magdy. 2015. QCRI: Answer Selection for Community Question Answering - Experiments for Arabic and English. In *SemEval 2015*. ACL.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. SIGIR '15. ACM.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. 2014. A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. CIKM.

Valentin I. Spitkovsky and Angel X. Chang. 2012. A Cross-Lingual Dictionary for English Wikipedia Concepts. In *LREC'12*. ELRA.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. NAACL '03. ACL.

Quan Hung Tran, Vu Tran, Tu Vu, Minh Nguyen, and Son Bao Pham. 2015. JAIST: Combining multiple features for Answer Selection in Community Question Answering. In *SemEval 2015*. ACL.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *Proceedings of NAACL-HLT*, pages 1268–1278.

Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic Tree-edit Models with Structured Latent Variables for Textual Entailment and Question Answering. COLING '10. ACL.

Di Wang and Eric Nyberg. 2015. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. In *ACL 2015*.

Xin-Jing Wang, Xudong Tu, Dan Feng, and Lei Zhang. 2009. Ranking Community Answers by Modeling Question-answer Relationships via Analogical Reasoning. In *SIGIR '09*. ACM.

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

Shangxuan Wu, Ying-Cong Chen, Xiang Li, An-Cong Wu, Jin-Jie You, and Wei-Shi Zheng, editors. 2016. *An Enhanced Deep Feature Representation for Person Re-identification*.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical Evaluation of Rectified Activations in Convolutional Network. *CoRR*, abs/1505.00853.

Xuchen Yao, Benjamin Van Durme, Chris Callison-burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *(NAACL)*.

Liang Yi, JianXiang Wang, and Man Lan. 2015. ECNU: Using Multiple Sources of CQA-based Information for Answers Selection and YES/NO Response Inference. In *SemEval 2015*. ACL.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. *CoRR*, abs/1412.1632.

Xiaoqiang Zhou, Baotian Hu, Jiaxin Lin, Yang xiang, and Xiaolong Wang. 2015. ICRC-HIT: A Deep Learning based Comment Sequence Labeling System for Answer Selection Challenge. In *SemEval 2015*. ACL.

Learning Event Expressions via Bilingual Structure Projection

Fangyuan Li[1], **Ruihong Huang**[2], **Deyi Xiong**[1]*, **Min Zhang**[1]
Soochow University, Suzhou, China[1]
Texas A&M University, College Station, USA[2]
fyli@stu.suda.edu.cn, huangrh@cse.tamu.edu
{dyxiong, minzhang}@suda.edu.cn

Abstract

Identifying events of a specific type is a challenging task as events in texts are described in numerous and diverse ways. Aiming to resolve high complexities of event descriptions, previous work (Huang and Riloff, 2013) proposes multi-faceted event recognition and a bootstrapping method to automatically acquire both event facet phrases and event expressions from unannotated texts. However, to ensure high quality of learned phrases, this method is constrained to only learn phrases that match certain syntactic structures. In this paper, we propose a bilingual structure projection algorithm that explores linguistic divergences between two languages (Chinese and English) and mines new phrases with new syntactic structures, which have been ignored in the previous work. Experiments show that our approach can successfully find novel event phrases and structures, e.g., phrases headed by nouns. Furthermore, the newly mined phrases are capable of recognizing additional event descriptions and increasing the recall of event recognition.

1 Introduction

Event recognition aims to identify documents that describe a specific type of event. Accurate event recognition is challenging due to ambiguities of event keywords. In the previous work, Huang and Riloff (2013) (hereafter H&R) proposed multi-faceted event recognition method that uses event expressions as well as event defining characteristics (aka "event facets", such as "agents" and "purpose") to achieve high accuracy in identifying civil unrest events. They also presented a bootstrapping solution that can learn event expressions and event facet phrases from unannotated texts. However, to achieve high quality phrases, strict syntactic constraints have been enforced and their bootstrapping algorithm can only learn two particular types of V-O (Verb-Object) Structure for both event expressions and facet phrases. Obviously, diverse forms of other verb phrases and non-verb phrases exist to describe events and are ignored by the proposed algorithm. For instance, a verb phrase where two verbs are connected with a particular dependency relation "xcomp"[1], (e.g., "came out to demonstrate") is one of these structures. Civil unrest events can also be invoked by some noun structure phrases, such as just a noun word phrase (e.g., "sit-ins") or phrases starting with a noun (e.g., "disobedience of order"), even a passive form phrase structure like "rallies held (in)".

In order to address this issue, we propose a simple yet effective bilingual structure projection method that explores syntactic divergences (Georgi et al., 2012) between two languages and mines new syntactic structures for event expressions and event facet phrases effectively using parallel corpora. This is inspired by many recent cross-lingual research that utilize the second language to provide a different view (Balcan and Blum, 2005; Burkett et al., 2010; Ganchev et al., 2012) and complementary cues (Che et al., 2013; Wang et al., 2013) in improving Natural language Processing (NLP) tasks for the target language, analogous to co-training (Chen and Ji, 2009; Wan, 2009; Hajmohammadi et al., 2015) but between two different languages. In order to learn new event phrases and their syntactic structures, we map phrases[2] back and

*Corresponding author

[1]"xcomp" is a dependency relation between a verb or an adjective and its open clausal complement in a dependency tree. In sentence "Workers came out to demonstrate", the relation between verb "came" and verb "demonstrate" is "xcomp".

[2]We start with initial phrases learned by H&R, thanks to the authors for sharing the learned phrases and evaluation data.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1441–1450, Osaka, Japan, December 11-17 2016.

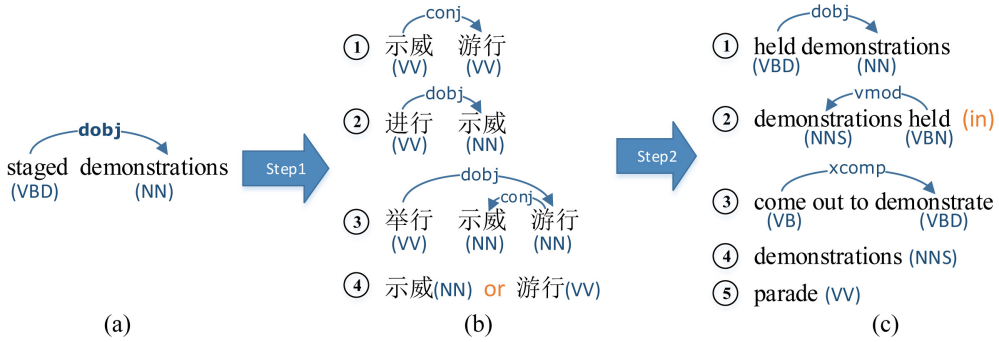


Figure 1: The bilingual structure mapping procedure

forth multiple times between two languages using parallel corpora to make full use of these divergence information. We choose Chinese as the pivot language to learn English event phrases and event facet phrases. On the one hand, both Chinese and English share a common sentence structure SVO (Subject-Verb-Object) and other similar sentence composition. On the other hand, these two languages have many significant differences, e.g., an uninflected language (Chinese) vs. an inflected language (English). The commonalities enable bilingual projection while the language divergences stimulate occurrences of new phrases and new phrase structures. The input to our bilingual structure projection system are two English verb phrase lists, event phrases and purpose facet phrases which are learned by H&R's multi-faceted event recognition method. After each mapping step, new syntactic structures and new phrases are learned.

Figure 1 illustrates one iteration of bilingual structure mapping with examples. Given the English phrase "staged demonstrations" with the structure of "VBD<dobj>NN", English sentences containing this phrase in the parallel corpora are identified. Then various Chinese phrases (Figure 1(b)) are generated when mapping the English phrase to its Chinese correspondence based on word alignments of the parallel sentences. Interestingly, a multi-word verb phrase in English can be expressed with only one noun (e.g., "示威"/demonstration) or a verb (e.g., "游行"/parade) in Chinese. Furthermore, we have observed that Chinese tends to use a conjunction of two verbs that have roughly the same meaning when referring to one single event, e.g., "示威 游行" in Figure 1(b). More examples are given in Figure 1. When we map these already diversified Chinese phrases back to English, new phrases with richer syntactic structures (Figure 1(c)) are generated, including some interesting noun structure phrases and one single-word phrase. To fully exploit language divergences, the bilingual structure projection run back and forth between the two languages and continues for several iterations.

Experiment results show that our approach can successfully find hundreds of new English phrasal structures, e.g., structures headed by nouns, and learn thousands of new event expression and event facet phrases. Furthermore, using the same evaluation data and evaluation method as in H&R, the newly mined phrases are capable of recognizing additional event descriptions, and significantly increasing the recall of event recognition by 8.2 points and the overall F_1-score by 3.5 points.

This paper is structured as follows: Section 2 describes the H&R's multi-faceted event recognition approach. Section 3 details our bilingual structure projection method and some heuristic rules used in our method. Section 4 describes our experimental design and evaluation results. Then section 5 discusses a variety of new phrases and structures generated by our approach. Section 6 introduces the related work of bilingual methods for various NLP tasks. Last, section 7 summarizes the bilingual structure method and expounds our future work.

2 Background

Accurately identifying documents that describe a specific type of event is a challenging task because events can be mentioned in various complex contexts. Using event keywords alone are barely reliable. For example, while the words "strike", "rally" and "riot" are commonly used to describe civil unrest events, they frequently refer to other events that are dramatically different from civil unrest events including "bowling strike", "rally car" and "imagination riot". In the previous work, Huang and Riloff

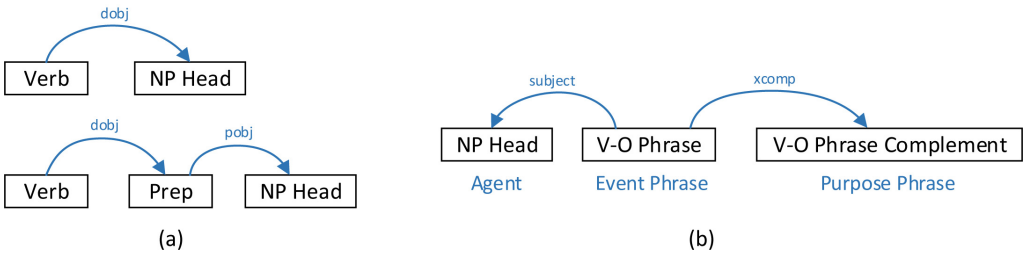


Figure 2: Syntactic constraints in H&R's method. (a) shows two pre-defined phrase structures, (b) is the sentence pattern in H&R's bootstrapping system

(2013) proposed the multi-faceted event recognition approach that uses both event expressions and event defining characteristics (called event facets) to accurately identify event occurrences. As described in the paper, agents and purpose are two types of event facets that are essential to distinguish many types of events. For instance, both natural disasters and military incidents can mention injuries and deaths as consequences, however, their agents are distinct. The agents of natural disasters have to be natural force while the agents of military incidents have to include military personnel. Similarly, purposes describe motivations of events and are extremely helpful to distinguish various types of events.

H&R also proposed a bootstrapping framework to learn event expressions and event facet dictionaries from the unannotated texts automatically requiring only minimal supervision with a few event keywords and a few seed phrases for each event facet. They observed that event facet phrases and event expressions tend to co-occur in event introductory sentences. Therefore, the bootstrapping system first learns event expressions from the sentences that contain both types of event facets and then learns more event facet phrases from the sentences that contain an event expression and a different type of event facet, in an iterative manner. Their multi-faceted event recognition with bootstrapped event dictionaries achieved high precision (88%) with a reasonable recall (71%) on identifying civil unrest events. However, to ensure high quality of learned phrases, strict syntactic constraints were enforced at both the phrasal and sentential level. Specifically, they only considered event expressions and purpose phrases as verb phrases in two types (Figure 2(a)), the first phrasal type is a verb followed by the head noun of its direct object and the second phrasal type is a verb with an attached prepositional phrase, while reasonably both event expressions and purpose phrases can exist in many other syntactic structures. Furthermore, within a sentence, specific dependency relations are required between both facet phrases and the main event expression (Figure 2(b)), the agent term has to be the syntactic subject of the event expression and the purpose phrase has to be a clausal complement of the event expression. Obviously, these harsh syntactic constraints pose limitations to the types of event phrases and event facet phrases that can be learned using this framework. Our research is committed to mine new phrases and phrase structures that go beyond these constraints leveraging divergences across two languages.

3 Learning Event Expressions via Bilingual Structure Projection

Our algorithm can iterate multiple times to learn new phrases in new syntactic structures automatically. In our experiments, we expand two types of phrases: event phrases (EP) and purpose phrases (PP) learned by H&R's method.

3.1 One Iteration of Bilingual Structure Projection

In our bilingual projection, we use structured phrases for mapping. Structured phrases[3] comprise both lexical and structural information. One iteration of the projection consists of two stages: mapping English event phrases to Chinese and projecting Chinese equivalents back to English. Next, we use an example as shown in Figure 2 to illustrate the projection process from English to Chinese.

[3]A structured phrase is defined as "start_node <relation1> in_node1 <relation2> in_node2 <realation3> ... <realtionN> end_node", where each node is a word, and the relation between two nodes is their dependency relation. Structured phrases capture both lexical and structural information for event expressions. Each structured phrase is essentially a path between two nodes in a dependency parse tree.

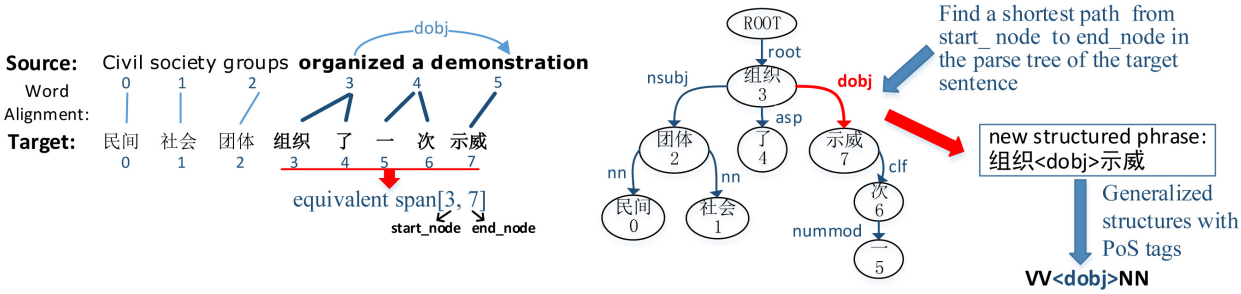


Figure 3: The illustration of projection from English to Chinese

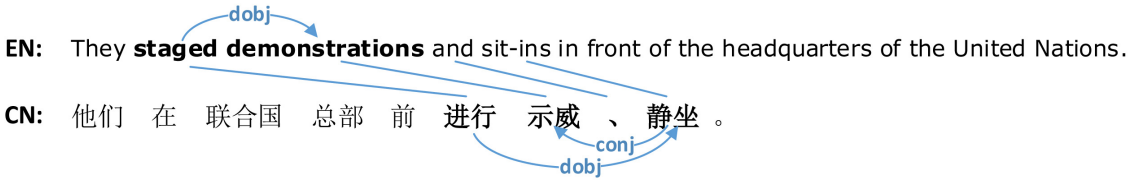


Figure 4: One verb with two coordinate objects

First, we identify an English event phrase on the source side of our parallel corpora, e.g., "organized a demonstration" in Figure 3, and extract the corresponding structured phrase "organized <dobj> demonstration" according to the dependency tree. Second, we detect phrase span of the translation equivalent for the extracted structured phrase by computing the aligned span on the target side via word alignments. In Figure 3, the equivalent span on the Chinese side is [3, 7]. In the third step, we take the leftmost and rightmost word of the translation equivalent span as the start_node and end_node. Then we further generate the structured phrase on the target side by finding the shortest path from the start_node to the end_node in the dependency tree, e.g., "组织 <dobj> 示威". For the in-depth analysis in Section 5, we use part-of-speech (PoS) tags to replace words in the found structured phrase to obtain a generalized structure, e.g., VV<dobj>NN in Figure 3. Then, mapping the new generated Chinese structured phrases ("组织 <dobj> 示威" in Figure 3 for example) back to English in the second stage of the bilingual projection following the similar procedure as described above. After the two-stages mapping, we obtain diversified English event phrases.

3.2 Phrase Decomposition

From our training data, we have learned many phrases with a conjunction. We find that most of them follow two structure patterns. The first is that one verb has two coordinate objects that express two related events. For example, the Chinese equivalent of English phrase "staged demonstrations" in Figure 4 is "进行 示威". However, there is not a direct dependency relation between "进行" and "示威" in the dependency tree. Instead, they are connected by a word "静坐" (sit-ins). Apparently, " staged demonstrations" (进行 示威) and "staged sit-ins" (进行 静坐) are two related events. The other pattern is an interesting phenomenon we have observed in our structure projection experiments. Chinese language tends to use a conjunction of two words that have roughly the same meaning when referring to an event while in English only one of the two coordinates is used to refer to the same event. For example, "捍卫 <dobj> 人权 <conj> 民主" (defend human rights and democracy) is a common expression in Chinese. However, in English, "defend human rights" is used to express the same meaning. To exploit this conjunction structure and linguistic divergence, we split such phrases into two separate phrases and keep both original phrases and decomposed phrases in the bilingual projection. For example, "进行 示威、静坐" is separated into two phrases "进行 示威" and "进行 静坐".

3.3 Phrase Filtering

In order to alleviate error propagation from word alignments and dependency trees, we apply phrase filtering. Particularly, we adopt three strategies to filter out inappropriate phrases.

Filtering by phrase frequencies: We keep phrases that occur at least t times and discard phrases occurring less than t times to minimize the impact of word alignment and dependency parse errors. Parameter t is tuned on the development set as we harvest new phrases.

Structural filtering: We use syntactic structure information to rule out incomplete phrases. For instance, Chinese phrase "进行 了" (carry on sth.) with a phrase structure "VV<asp>AS" is not a complete phrase since it does not have an object. Similarly, we filter out phrases ending with "AS", "P", "DEC", "LC", "PU", "CD", "MSP" [4].

Filtering by phrase specificity: We keep phrases that are closely related to our topic. Some phrases occur many times during the learning procedure for two reasons. The first reason is that they are closely related to our topic. The second is because they are high-frequency phrases in our corpora. We have observed that some highly frequent and general phrases in our corpora often occur in the learning process, mainly due to word alignment errors or dependency parsing errors. Aiming to learn phrases that are specific event expressions, we define a metric called phrase specificity to avoid bringing in corpora-wide frequent phrases. The metric for phrase *p* is defined as follows:

$$phrase\_specificity(p) = \frac{N_l}{N_c} * 100 \quad (1)$$

where N_l denotes the number of occurrences of phrase *p* in our projection procedure, N_c denotes the number of occurrences in our entire corpora. This metric measures how close a new phrase is related to the topic of events that we want to detect. If N_ls of two phrases are close to each other, but N_c of one phrase is bigger than that of the other, we deem the phrase with bigger N_c more likely to be a high-frequency phrase. In our experiments, we only consider phrases that have this metric over a certain threshold. We use a development set (section 4.1) to determine the threshold.

3.4 Iterative Projection

We further extend the projection process described in Section 3.1 with phrase decomposition (Section 3.2) and phrase filtering (Section 3.3) to an automatic iterative system. This allows us to use newly learned phrases to learn more new phrases. The most straightforward idea is executing the projection procedure in Section 3.1 many times. However in practice, the growth rate of the number of newly learned phrases is far beyond our imagination. During the iterative projection between the two languages, thousands of incomplete or incorrect phrases are generated. In order to control the growth rate of new phrases and to avoid generating bad phrases, we only keep new phrases that are found at least twice by the iterative system. We deem phrases learned repeatedly more reliable than those occasionally learned. For example, we can learn five different phrases: "举行 <dobj> 示威"(4)、"示威"(2)、"举行 <dobj> 游行"(2)、"举行 <dobj> 活动"(1)、"示威者"(1) when phrase "held<dobj>demonstrations" is mapped to Chinese. The numbers in brackets show the times of phrase learned. According to the strategy, the last two phrases are removed. This is different from filtering by phrase frequencies strategy in Section 3.3. The phrase frequency in section 3.3 means the total times of a new phrase learned by all original phrases. But in the iterative projection, we talk about the frequency of different new phrases learned by one particular original phrase.

4 Experiments

After each structure projection iteration, we appended the newly learned phrases to their corresponding phrase list (EP or PP) and ran the same event recognition evaluation procedure as in H&R's but with the appended longer phrase lists.

4.1 Data

Our experiment bilingual data consists of 3.57M bilingual sentences from LDC corpora LDC2004E12, LDC2004T08, LDC2005T10, LDC2003E14, LDC2002E18, LDC2005T06, LDC2003E07,

[4] AS: aspect markers, P: prepositions, DEC: Chinese "的" for relative clauses, LC: localizers, PU: punctuations, CD: cardinal numbers, MSP: some particles.

Method	Phrases	Recall	Precision	F_1
H&R's Iter #4	EP:623 PP:569	71	88	79
Iteration 1	EP:1096 PP:2219	76.2	86.5	81.1
Iteration 2	EP:4273 PP:4597	79.2	86.0	82.5
Iteration 3	EP:8041 PP:9169	79.2	86.0	82.5
Iteration 4	EP:9868 PP:11705	79.2	86.0	82.5

Table 1: Results of the projection method using H&R's phrase lists as seed phrases for expansion and projection

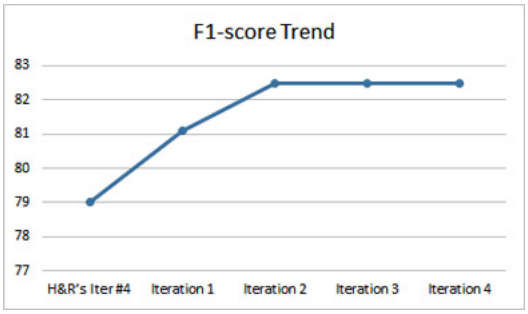


Figure 5: F1-score curve against the number of iterations

LDC2004T07. We ran Giza++ (Och, 2003) and Stanford dependency parser (De Marneffe et al., 2006; Chang et al., 2009) on the parallel sentence pairs to obtain word alignments and dependency trees. In addition, we used the same evaluation method and data as H&R's. The evaluation data contains 400 news articles that were randomly sampled from the English Gigaword Fifth Edition corpora (Parker et al., 2011). Each article contains one of six commonly used civil unrest keywords or their morphological variations. The development set contains 100 documents and the rest 300 documents are used as the test set.

4.2 Event Recognition with Expanded Phrases

We examine the effectiveness of our bilingual structure projection algorithm on the task of event recognition. We choose H&R's best result as our baseline. H&R's multi-faceted event recognition approach achieves the best result after four iterations of bootstrapping.

Our first experiment was designed to expand the EP and PP lists learned by H&R's method at the 4th iteration with our bilingual structure projection system. Our system ran for multiple iterations. According to the development data, the best F_1-score was achieved after the first two iterations. Table 1 shows the event recognition performance of our bilingual structure projection method. The original multi-faceted event recognition approach at the 4th iteration has achieved a high accuracy (88%) with a relatively low recall (71%). After the first iteration of projection, we obtained an improvement of 5.2 points on recall and 2.1 points on F_1-score over the baseline. With the newly learned phrases in the first iteration projection, the event recognition recall can be further improved by another 3 points after the second iteration. Overall, with a little loss in precision, the recall has increased by 8.2 points and the F_1-score 3.5 points. We further observed that results cannot be elevated further after the 2nd iteration even with more phrases, as shown in Figure 5. We conjecture that the reasons are twofold. First, the limited original phrases may not supply more useful phrases after two iterations, which results in a saturated useful phrase list. Second, we do get more useful phrases. However the test data is not large enough so that all newly learned phrases can be found in the test data. Therefore, we cannot see further changes in performance. From the results, we can see the bilingual structure projection algorithm can mine thousands of new phrases. With the newly learned phrases, we can successfully identify additional civil unrest events in the test data.

Due to the noise in H&R's phrase lists (the precision is 88%, indicating 12% noisy phrases) and the features of bootstrap system, phrases learned in previous iterations often have a high precision, but the quality of phrases normally decrease in the succeeding iterations. We further conducted experiments with the phrase lists (EP and PP) learned from the first to the third iteration by H&R's method, which have a high quality. The results are shown in Table 2. In these three iterations, our bilingual structure projection algorithm can improve the recall with almost no loss in precision. This illustrates that our method can recall more phrases and patterns still with a high precision. Note that our method has already outperformed H&R's best result at the third iteration (73.3% in recall and 79.6% in F1-score) while H&R's method achieved this performance after the 4th iteration. Therefore, with bilingual structure projection, the number of iterations of the original bootstrapping learning process can be decreased.

phrase iteration	Method	EP numbers	PP numbers	Recall	Precision	F_1
Iter #1	H&R's Method	145	124	50	88	63
	Bilingual Projection	279	888	53.5	90.0	67.1(+4.1)
Iter #2	H&R's Method	410	356	63	89	74
	Bilingual Projection	790	1387	68.3	88.5	77.1(+3.1)
Iter #3	H&R's Method	504	402	68	88	77
	Bilingual Projection	968	1501	73.3	87.1	79.6(+2.6)

Table 2: Results at the first three iterations

phrase iteration	Method	Recall	Precision	F_1
TermLex	H&R's Method	66	85	74
	Bilingual Projection	61	81	70
PairLex	H&R's Method	10	91	18
	Bilingual Projection	12	100	21
TermSets	H&R's Method	59	83	69
	Bilingual Projection	57	73	64
PairSets	H&R's Method	68	84	75
	Bilingual Projection	79	78	79
ALLSets	H&R's Method	70	84	76
	Bilingual Projection	78	78	78
Average of Five	H&R's Method	54.6	85.4	62.4
	Bilingual Projection	57.4	82.0	62.4

Table 3: Results of SVM with the bilingual projection method

4.3 SVM Classifiers with Bilingual Structure Projection

H&R also experimented with a suite of supervised classifiers by engineering features based on their learned event dictionaries. In their presented results, supervised classifiers yielded worse event recognition performance than the multi-faceted approach that simply relies on exact match with learned event dictionaries. One guess for this inferior comparison is that their learned phrases are still not diverse and rich enough and their induced feature vectors are too sparse. We have learned many more phrase for both event expressions and the purpose facet through our bilingual structure projection method. We rebuilt the same set of supervised classifiers with the same features. But the features are induced based on the augmented EP lists and PP lists using our bilingual structure projection algorithm. Agent phrase lists (AP) keep the same as H&R's. We ran experiments on five SVM classifiers as shown in Table 3 and performed ten-fold cross validation on the test set, the same as H&R's. All features are binary. We use a vector of 0 and 1 to represent a document. TermLex encodes a binary feature for every phrase in all three phrase lists. PairLex encodes a binary feature for each pair combination from two different lists and requires them to occur in one same sentence. TermSets encodes three binary features for each list, a feature gets 1 when at least one phrase occurs in the document from the corresponding list. PairSets encodes three binary features and each feature represents a combination of two different lists (EP+PP, PP+AG, EP+AG). If any pair occurs in the same sentence, the value gets 1 otherwise 0. Last, the ALLSets encodes 7 binary features, the previous six features plus another binary feature of a sentence containing at least an entry combination from all three lists. Table 3 shows the comparison of our projection method and H&R's method. Although our expanded phrases do not work well on TermLex and TermSets, they still can improve other three classifiers in different degrees. The last row in Table 3 shows the average performance of two methods. Generally, compared to multi-faceted based phrases, our expanded phrases increase the recall, but lower the precision, overall F is the same. Our experiments reconfirm that multi-faceted event dictionary match based event recognition approach, while simple, is more effective than trained supervised classifiers that use dictionary matches as features.

5 Analysis: New Phrases and Structures

We further analyze syntactic structures of the newly learned phrases by bilingual structure projection. Due to linguistic divergences between English and Chinese, various novel new structures are observed in learned Chinese phrases and English phrases.

New Chinese Structures and Examples
NN: 静坐 (stage sit-ins), 怠工 (stop work), 罢工 (went on strike)
VV: 纵火 (set fire), 泄愤 (vent their anger)
VV<rcomp>VV: 纵火 焚烧 (set fire), 进行 绝食 (go on hunger strike)
VV<dobj>NN<conj>NN: 举行 游行 示威 (stage demonstrations), 加入 抗议 罢工 (join the strike and protest)
VV<dobj>NN<relcl>VV: 放火 焚烧 车辆 (set fire to vehicles), 表达 反对 呼声 (express opposition)

Table 4: Examples of new Chinese structures learned

New English Structures and Examples
NN: self-immolation, demonstrations, sit-ins
NN<prep>NN: overuse of force, boycott of elections, disobedience of order
NN<vmod>VBN: rallies held (in), objections expressed (by), rocks thrown (at), disturbances caused (by)
VV: demonstrated, parade
VB<xcomp>VB: cease (to) function, came (out to) demonstrate, pledged (to) support, urge (them to) resign
VB<dobj>NN<conj>NN: held rallies and demonstrations, staged sit-ins and hunger strikes
VB<dobj>NN<prep_of>NN: prevent acts of discrimination, condemned acts of terrorism

Table 5: Examples of new English structures learned

Table 4 shows examples of several new Chinese phrase structures. Interestingly, a multi-word verb phrase in English can be expressed with only one noun or verb word in Chinese, e.g., "went on strike" vs. "罢工" (a noun in Chinese), "vent their anger" vs. "泄愤" (a verb in Chinese). Even more interestingly, we have observed that Chinese tends to put together two coordinate words with roughly the same meaning when referring to one single event, e.g., "staged demonstrations" aligned to "举行 游行 示威" ("游行" and "示威" both mean demonstrations). The reason for putting two words with similar meanings together is to emphasize on the occurrence of the event. More examples are given in Table 4.

Table 5 shows a few examples of new English phrase structures. Dramatically different from the two pre-defined types of verb phrases as specified in H&R's research, many new phrases are headed by nouns, including individual nouns "sit-ins", nouns with a prepositional attachment "boycott of elections" and nouns modified by a passive voiced verb phrase "rallies held in". In addition, we have seen some new verb structures in English phrases that consist of a single verb or a verb with complex objects as shown in Table 5.

6 Related Work

Recent years have witnessed increasing interests in leveraging bilingual corpora or resources to improve performance of monolingual NLP tasks. Generally, The introduction of bilingual corpora or resources serves two purposes. The first purpose is to alleviate the problem that we have few labeled instances in some resource-impoverished languages by a resource-rich language (Hwa et al., 2005; Ganchev et al., 2009; Das and Petrov, 2011; He et al., 2015). The second purpose is to leverage divergences found in different languages to obtain complementary cues (Li et al., 2012; Wang et al., 2013; Che et al., 2013) or extra information (Snyder et al., 2009; Burkett et al., 2010) from another language. Our projection method follows the latter.

In the first purpose, Das and Petrov (2011) explored existing abundant English labeled resources as features to assist building tools for eight European languages. Different to projecting labels as feature, Wang and Manning (2014) proposed a method that projected model expectations as feature for training. He et al. (2015) transferred the sentiment information of a resource-rich language to replenish the lost information of the target language.

In the second purpose, Chen and Ji (2009) proposed a bootstrap framework of co-training among two languages, which uses Chinese event extraction as a case study and bilingual texts as a new source of information. Burkett et al. (2010) attached a bilingual model as a second view (Balcan and Blum, 2005; Ganchev et al., 2012) onto original monolingual models, and used rich features from unannotated bitext to train parameters in bilingual models, which can help to reproduce training data of monolingual model. Che et al. (2013) exploited the complementary cues between two languages as bilingual constraints to help detect errors in a mono-lingual tagger task, which can improve the annotation quality of named entities. Zhu et al. (2013) translated English sentences into Chinese sentences (with the same topic) in ACE 2005 evaluation data with google machine translation system as a second text representation feature

so as to alleviate the data sparseness problem effectively.

Our method is also related to paraphrase learning (Bannard and Callison-Burch, 2005; Callison-Burch, 2008; Zhao et al., 2008; Snover et al., 2009; Ganitkevitch et al., 2013). However, there are two significant differences. First, paraphrase learning translates phrases strictly via word alignments while we use word alignments to find phrase spans on the target language. Second, our purpose is to obtain structured phrases (with syntactic constraints) rather than plain phrases as structured phrases can help us find new phrase structures as shown in Section 3.

7 Conclusion and Future Work

We have presented a bilingual structure projection algorithm that explores structural divergences between languages and can effectively dig up new phrase with various new structures by mapping phrases back and forth across two languages. We combine syntactic information with machine translation technology, not only can reduce the effect of word alignment errors, but also diversify the original two pre-defined event phrase structures. Our experiments show that the newly learned event phrases are capable of recognizing additional event descriptions and considerably increasing the recall of event recognition with minimal loss on precision. Bilingual structural divergences between human languages are common, the proposed bilingual structure projection algorithm is general and can be applied to any pair of languages, and easily extended to the scenario with multiple languages. In addition to event recognition, the proposed structure projection algorithm across languages is potentially useful to many other NLP tasks that utilize extraction patterns by automatically generating novel and diverse phrasal patterns. In our future work, we will attempt to explore the possibility and effect of expanding phrases among other language pairs and other NLP tasks using our bilingual structure projection method.

8 Acknowledgments

The authors were supported by National Natural Science Foundation of China (Grant Nos. 61403269, 61432013 and 61525205) and Natural Science Foundation of Jiangsu Province (Grant No. BK20140355). This research was also partially supported by Ruihong Huang's startup funds in Texas A&M University. We also thank the anonymous reviewers for their insightful comments.

References

Maria-Florina Balcan and Avrim Blum. 2005. A pac-style model for learning from labeled and unlabeled data. In *Learning Theory*, pages 111–126. Springer.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604. Association for Computational Linguistics.

David Burkett, Slav Petrov, John Blitzer, and Dan Klein. 2010. Learning better monolingual models with unannotated bilingual text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 46–54. Association for Computational Linguistics.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 196–205. Association for Computational Linguistics.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D Manning. 2009. Discriminative reordering with chinese grammatical relations features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*, pages 51–59. Association for Computational Linguistics.

Wanxiang Che, Mengqiu Wang, Christopher D Manning, and Ting Liu. 2013. Named entity recognition with bilingual constraints. In *HLT-NAACL*, pages 52–62.

Zheng Chen and Heng Ji. 2009. Can one language bootstrap the other: a case study on event extraction. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, pages 66–74. Association for Computational Linguistics.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 600–609. Association for Computational Linguistics.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 369–377. Association for Computational Linguistics.

Kuzman Ganchev, Joao Graca, John Blitzer, and Ben Taskar. 2012. Multi-view learning over structured and non-identical outputs. *arXiv preprint arXiv:1206.3256*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *HLT-NAACL*, pages 758–764.

Ryan Georgi, Fei Xia, and William D Lewis. 2012. Measuring the divergence of dependency structures cross-linguistically to improve syntactic projection algorithms. In *LREC*, pages 771–778.

Mohammad Sadegh Hajmohammadi, Roliana Ibrahim, Ali Selamat, and Hamido Fujita. 2015. Combination of active learning and self-training for cross-lingual sentiment classification with density analysis of unlabelled samples. *Information sciences*, 317:67–77.

Xiaonan He, Hui Zhang, Wenhan Chao, and De Qing Wang. 2015. Semi-supervised learning on cross-lingual sentiment analysis with space transfer. In *2015 IEEE First International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 371–377.

Ruihong Huang and Ellen Riloff. 2013. Multi-faceted event recognition with bootstrapped dictionaries. In *HLT-NAACL*, pages 41–51.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(03):311–325.

Qi Li, Haibo Li, Heng Ji, Wen Wang, Jing Zheng, and Fei Huang. 2012. Joint bilingual name tagging for parallel corpora. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1727–1731. ACM.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition, june. *Linguistic Data Consortium, LDC2011T07*.

Matthew G Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2009. Ter-plus: paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23(2-3):117–127.

Benjamin Snyder, Tahira Naseem, and Regina Barzilay. 2009. Unsupervised multilingual grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 73–81. Association for Computational Linguistics.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 235–243. Association for Computational Linguistics.

Mengqiu Wang and Christopher D Manning. 2014. Cross-lingual pseudo-projected expectation regularization for weakly supervised learning. *Transactions of the Association for Computational Linguistics*, 2:55–66.

Mengqiu Wang, Wanxiang Che, and Christopher D Manning. 2013. Effective bilingual constraints for semi-supervised learning of named entity recognizers. In *AAAI*. Citeseer.

Shiqi Zhao, Haifeng Wang, Ting Liu, and Sheng Li. 2008. Pivot approach for extracting paraphrase patterns from bilingual corpora. In *ACL*, volume 8, pages 780–788.

Zhu Zhu, Shoushan Li, Guodong Zhou, and Rui Xia. 2013. Bilingual event extraction: a case study on trigger type determina-tion.

Global Inference to Chinese Temporal Relation Extraction

Peifeng Li, Qiaoming Zhu, Guodong Zhou, Hongling Wang
School of Computer Science & Technology
Soochow University, Suzhou, 215006, China
{pfli, qmzhu, gdzhou, hlwang}@suda.edu.cn

Abstract

Previous studies on temporal relation extraction focus on mining sentence-level information or enforcing coherence on different temporal relation types among various event mentions in the same sentence or neighboring sentences, largely ignoring those discourse-level temporal relations in nonadjacent sentences. In this paper, we propose a discourse-level global inference model to mine those temporal relations between event mentions in document-level, especially in nonadjacent sentences. Moreover, we provide various kinds of discourse-level constraints, which derived from event semantics, to further improve our global inference model. Evaluation on a Chinese corpus justifies the effectiveness of our discourse-level global inference model over two strong baselines.

1 Introduction

Temporal relation extraction is to determine the temporal relationship (e.g., *Before* and *After*) holding among events. It has been drawing more and more attention due to the crucial importance of temporal information to various natural language processing (NLP) applications, such as language generation, information extraction, summarization, and question answering. The difficulty with this task is that temporal information about event mentions is sometimes not stated explicitly and one can only infer from their context. Currently, temporal relation extraction still remains a challenge in corpus construction and inference mechanism.

On one hand, although the TimeBank corpus (Pustejovsky et al., 2003), the commonly used corpus in previous studies, has largely promoted the development of temporal relation extraction, it only annotates a small subset of easily-identified event mention pairs. Moreover, it largely ignores almost all temporal relations between event mentions in nonadjacent sentences. These lead to fragmented relations and limit its applications to other NLP tasks, such as information extraction, and summarization. Finally, while constructing a fully-annotated corpus is expensive and time-consuming, many NLP tasks are normally interested in specific types of events. For example, a summarization or information extraction system on terrorism attacks may only concern with a few event types (e.g., *Attack*, *Die*, and *Injure*). Therefore, annotating an event-driven fully-annotated temporal relation corpus becomes a crucial issue to the success of real-life applications.

On the other hand, previous studies on temporal relation extraction focus on mining sentence-level information or enforcing coherence on different temporal relation types among various mentions in the same sentence or neighboring sentences, largely ignoring those discourse-level temporal relations in nonadjacent sentences. Specifically, only a few studies apply global inference models to exploit temporal relations in discourse level. Therefore, how to acquire discourse-level temporal information from those long-distance event mention pairs in nonadjacent sentences becomes another crucial issue to temporal relation extraction, especially for Chinese, as a discourse-driven language with a broad range of ellipsis and flexible sentence structures.

In this paper, we first annotate an event-driven fully-annotated Chinese temporal relation corpus, on the top of the ACE (Automatic Content Extraction) 2005 Chinese corpus. Then, we propose a discourse-

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1451–1460, Osaka, Japan, December 11-17 2016.

level global inference model to mine those temporal relations between event mentions in document-level (especially in nonadjacent sentences) with various kinds of discourse-level constraints, which derived from event semantics, to further improve the global inference model. Evaluation indicates the appropriateness of our event-driven fully-annotated Chinese corpus and justifies the effectiveness of our discourse-level global inference model over two strong baselines.

2 Related Work

In this section, we give a brief overview of temporal relation extraction from two aspects: corpus construction and inference mechanism.

2.1 Corpus Construction

Most of existing corpora for temporal relation extraction focus on English. As the commonly used corpus in temporal relation extraction, the TimeBank corpus (Pustejovsky et al., 2003) has been adopted in a series of TempEval competitions (Verhagen et al., 2007; Verhagen et al., 2010; Uz-Zaman et al., 2013), facilitating the development and evaluation of temporal relation extraction systems. The problems with the TimeBank corpus are that it only annotates a small subset of easily-identified event mention pairs and that it largely ignores those temporal relations between event mentions in nonadjacent sentences. These lead to fragmented relations and much limit its applications.

To overcome above problems, Do et al. (2012) produced an event-driven corpus on the ACE 2005 English corpus. However, "the annotator was not required to annotate all pairs of event mentions, but as many as possible", as stated in their paper. This makes the annotation inconsistent and difficult to follow. Recently, Cassidy et al. (2014) enriched the TimeBank-Dense corpus, on the top of TimeBank. Specifically, they approximated the completeness by labeling locally complete graphs over neighboring sentences.

In comparison, there are few corpora for Chinese temporal relation extraction. Li et al. (2004) annotated a Chinese corpus including 700 sentences. The TempEval-2 competition (Verhagen et al., 2010) provided 780 instances of Chinese temporal event relations. Obviously, both corpora are rather small and largely impede the research in Chinese temporal relation extraction. For example, no team participated in the TempEval-2 competition on Chinese temporal relation extraction.

2.2 Inference Mechanism

Due to the corpus limitation, previous studies on temporal relation extraction focus on inferring temporal relations between event mentions in the same sentence or neighboring sentences from English text, dominated by feature-based approaches. Mani et al. (2006) applied the temporal transitivity rule to greatly expand the corpus. Lapata and Lascarides (2006) introduced various kinds of syntactic and clause-ordering features to classify the temporal relationship. Chambers et al. (2007) used previously learned event attributes to classify the temporal relationship. Laokulrat et al. (2013), the best performing one in the TempEval-3 competition, applied various predicate-argument structure features from a deep syntactic parser to enhance their classifier. Mirza and Tonelli (2014) illustrated that simple features resulted in a better performance than sophisticated features. Chambers et al. (2014) proposed a sieve-based architecture to joint those different tasks of temporal relation extraction.

In comparison, few studies concern temporal relation extraction from Chinese text. Chen et al. (2008) used verbal attributes to identify temporal relations of verbs. Li et al. (2004) presented a classifier-based collaborative bootstrapping approach to analyze temporal relations in a small Chinese corpus.

While above studies focus on local information, a few studies sort to global inference, with focus on exploiting global information via various kinds of temporal logic reflexivity and transitivity constraints, using frameworks like Integer Linear Programming and Markov Logic Networks (Bramsen et al., 2006; Chambers and Jurafsky, 2008; Yoshikawa et al., 2009). However, their gains are rather small, largely due to the common disconnectedness in the sparsely annotated corpora (Chambers et al., 2014). To overcome this problem, Denis and Muller (2011) decomposed temporal entities into sub-graphs and enforced the coherence only within these substructures, while Do et al. (2012) proposed a joint event-event and event-time classification model to enforce various coreference constraints.

Different from previous studies, we build an event-driven fully-annotated Chinese corpus and propose

a discourse-level global inference model to extract temporal relations between event mentions in document level, especially in nonadjacent sentences. To our knowledge, this is the first attempt in discourse-level global inference for temporal relation extraction from an event-driven fully-annotated corpus.

3 Data Construction and Baseline

In this section, we present the construction of our Chinese temporal relation corpus and the learning-based baseline.

3.1 Data Construction

To address various problems in existing corpora, as described above, we build an event-driven fully-annotated Chinese temporal relation corpus, on the top of the ACE 2005 Chinese corpus with 8 predefined event types and 33 predefined event subtypes (e.g., *Die*, *Attack*, and *Transport*). That is, all other event mentions of non-predefined event types are ignored in our corpus.

Different from previous corpora, each document in our corpus is annotated with the temporal relations between the mentions of all the events relevant to concerned events in the document, with the constraint of event-relevant completeness. Besides, we focus on four temporal relations, i.e. *Before*, *After*, *Overlap* and *Unknown* (without relationship or with vague relationship). This is a simplification of the TimeBank corpus, which defines 14 temporal relations. Since differentiating 14 temporal relation types is too hard, even for a well-educated person, much work has been done to simplify the temporal relation types, e.g. 6 types (Mani et al., 2006; Chambers et al., 2007; Cassidy et al., 2014) and 4 types (Do et al., 2012; UzZaman et al., 2013 (Chinese subtask)). Our work is a typical practice of such tendency.

Specifically, 163 documents from the ACE 2005 Chinese corpus are selected as our experimental data, which contains 1166 event mentions. These documents are from three different data sources (i.e., Broadcast News, Newswire and WebLog), very different in various aspects, such as quality, length and style. Two postgraduates in computer science are involved in corpus annotation and the Kappa value between the two annotators is 0.70, similar to TimeBank's 0.71.

Table 1 shows 4 temporal relations and their occurrence frequencies in our event-driven fully-annotated corpus. The total number of event pairs in our corpus is three times larger than that of TimeBank. Since the ACE 2005 corpus is licensed, we cannot upload our annotated data. If anyone obtain the license of the ACE 2005 corpus, our corpus is free available for research purpose on request.

Type	Before	After	Overlap	Unknown	Total
#Number	7402	7402	4834	1494	21132

Table 1. The 4 temporal relations and their occurrence frequencies

We have implemented a tool to help the annotators to tag event relations easily and enforce the coherence in document level. Due to the reflexive property of event-event relationship, the annotators only need annotate half of the relations shown in Table 1. Besides, 7.1% of annotated event relations are *Unknown*, and this figure is much lower than that in TimeBank. The reason is that the relations between two ACE events of the predefined event types are relatively easy to be identified and this also verifies the relatively high Kappa value between the two annotators. In our corpus, the maximal size of the relations in a document is 625 (25 event mentions), while the minimal size is 2 (only 2 event mentions). If we ignore those *Unknown* relations, 32% of documents are not graph, but forest.

3.2 Baseline

Similar to the state-of-the-art system in temporal relation extraction, we employ a learning-based system as one of our baselines. As an event-event (E-E) classifier, this baseline predicts one of the four temporal relations, i.e. *Before(B)*, *After(A)*, *Overlap(O)*, and *Unknown(U)*, between two event mentions e_i and e_j as follows:

$$C_{E-E}(e_i, e_j) \rightarrow \{\underline{B}, \underline{A}, \underline{O}, \underline{U}\} \quad (1)$$

Besides those features adopted in English temporal relation extraction (e.g., D'Souza and Ng, 2014; Mirza and Tonelli, 2014), we also apply various kinds of Chinese-specific features to further boost the

performance of this baseline. Specifically, for each event mention pair <e_1, e_2> in a document, with trigger mentions t_1 and t_2 respectively, its feature set can be divided into 5 categories:

1) **Lexical features (14)**: the tokens of t_1 and t_2 (2); their POS tags (2); their preceding and succeeding words (4); the POS tags of their preceding and succeeding words (4); the hedge or negative word before t_1 or t_2 (2);
2) **Syntactic features (4)**: the dependency path between t_1 and t_2 (1); the governors of t_1 and t_2 (2); the constituent path between t_1 and t_2 (1);
3) **Event features (18)**: the tense, polarity, genericity, modality and event type of e_1 and e_2 (10); the agents (2), the patients (2), the times (2), and the places of e_1 and e_2 (2);
4) **Pairwise features (9)**: the conjunction between e_1 and e_2 (1); whether e_1 and e_2 are in the same sentence (1); whether e_1 and e_2 have the same tense (1), the same polarity (1), the same genericity (1), the same modality (1), the same event type (1), and the same *Time* argument (1); whether e_1 is before e_2 in the document (1);
5) **Semantic features (7)**: whether t_1 and t_2 are synonym (1); whether the agent of e_1 is the patient of e_2 (1); whether the agent of e_2 is the patient of e_1 (1); whether e_1 and e_2 have the same time (1), place (1), agent (1) or patient (1).

All the sentences in the corpus are divided into words using the word segmentation tool ICTCLAS. Besides, we use *Berkley Parser* and *Stanford Parser* to create the constituent and dependency parse trees respectively. The event features (e.g., trigger, event tense, event type, event arguments) are derived from the annotated data in the ACE 2005 Chinese corpus. After creating the training instances, we train four *one-vs-rest* classifiers using the Maximum Entropy tool *MaxEnt*.

4 Global Inference on Event Semantics

While existing approaches, as the baseline described above, focus on limited event mention pairs in the same sentence or neighboring sentences, our global inference model attempts to address those in non-adjacent sentences. In this section, we first present the discourse-level global inference model to temporal relation extraction and then introduce various kinds of discourse-level constraints to achieve global optimization on the temporal relations of event mention pairs in both nonadjacent and adjacent sentences.

4.1 Global Inference Model

To mine the interaction among events in a document, we optimize the predicted temporal graph, formed by prediction from $C_{E\text{-}E}$, with various kinds of discourse-level constraints derived from event semantics.

Let $E=\{e_1,e_2,\ldots,e_n\}$ denote the set of event mentions in a document, $\varepsilon=\{(e_i,e_j)\in E\times E|e_i,e_j\in E, i\neq j\}$ the set of event mention pairs, and $R=\{\underline{B},\underline{A},\underline{O},\underline{U}\}$ the set of temporal relations. Besides, let $P_{<i,j,r>}$ denote the prediction probability of (e_i,e_j) with relation r ($r\in R$), given by the event-event classifier $C_{E\text{-}E}$, and $x_{<i,j,r>}$ the binary indicator on the existence of relation r for (e_i,e_j). Following Roth and Yih (2004) and Li et al. (2013) in information extraction, we define the following log costs:

$$c_{<i,j,r>} = -\log(P_{<i,j,r>}) \tag{2}$$

$$\bar{c}_{<i,j,r>} = -\log(1-P_{<i,j,r>}) \tag{3}$$

Specifically, ILP (Integer Logical Programming), a global inference is employed to achieve global optimization with the following objective function to maximize over a document as follows:

$$\arg\min_{x} \sum_{(e_i,e_j)\in\varepsilon} \sum_{r\in R} (c_{<i,j,r>} \times x_{<i,j,r>} + (1-x_{<i,j,r>}) \times \bar{c}_{<i,j,r>}) \tag{4}$$

s.t.

$$x_{<i,j,r>} \in \{0,1\} \tag{5}$$

$$\sum_{r\in R} x_{<i,j,r>} = 1 \tag{6}$$

while binary constraint (5) ensures that $x_{<i,j,r>}$ is binary value and equality constraint (6) ensures that exactly only one temporal relation can be assigned to each event mention pair.

In addition, the reflexivity and transitivity constraints, as deployed in previous inference models

(Bramsen et al., 2006; Chambers and Jurafsky, 2008; Do et al., 2012), are also applied to our model as follows:

$$x_{<i,j,r>} - x_{<j,i,\bar{r}>} = 0 \qquad \forall r, \bar{r} \in R \tag{7}$$

$$x_{<i,j,r>} + x_{<j,k,r>} - x_{<i,k,r>} \leq 1 \qquad \forall r \in \{\underline{B}, \underline{A}, O\} \tag{8}$$

Here, **reflexivity constraint** (7) enforces the reflexive property of the event-event relationship, where relation $\bar{r}$ denotes inverse relation r with possible (r, $\bar{r}$) pairs { ($\underline{A}$,$\underline{B}$) , ($\underline{B}$,$\underline{A}$) , ($\underline{U}$,$\underline{U}$) }, and **transitivity constraint** (8) states that if both event mention pairs (e_i,e_j) and (e_j,e_k) have the same temporal relation r, temporal relation r must hold between e_i and e_k, with event mention e_j as a bridge to link e_i and e_k.

4.2 Discourse-level Constraints

Different from the TimeBank corpus which only annotates the temporal relations in the same sentence or neighboring sentences, our corpus is event-driven fully-annotated. That is, besides the temporal relations in the same sentence or neighboring sentences, our corpus also contains those in nonadjacent sentences, which occupy 56.3%. This poses the great necessity to address those temporal relations in nonadjacent sentences. Besides, although all the event types in the ACE corpus have a *Time* role, the statistics on our corpus shows that only 35.9% of event mentions have explicit *Time* arguments. This poses the great challenge to address those temporal relations in nonadjacent sentences due to the frequent lack of explicit *Time* arguments.

Motivated by the intuition that the intrinsic semantics of event mentions is helpful to reveal their temporal relations due to the semantic nature in the event definition, we propose various kinds of discourse-level constraints on time arguments, event relevance, event tense, discourse connective, and coreference to mine the temporal relations in both nonadjacent and adjacent sentences.

Argument *Time* constraint

Generally, an event can be expressed as "5W1H" (*Who*, *What*, *Whom*, *Where*, *When* and *How*). *When*, one of "5W", indicates the time an event happens. Naturally, this argument is the solid evidence to identify the temporal relation between two event mentions. For example, if the *Time* argument of one event mention e_1 is "今日" (today) and that of the other mention e_2 is "昨日" (yesterday), it is obvious that the relation between e_1 and e_2 is *After*.

For time arguments, we can obviously have the following constraint, stating that if *Time* argument at_i of event mention e_i is before *Time* argument at_j of event mention e_j, the temporal relation between e_i and e_j is $\underline{B}$, and if at_i is equal to at_j, or they have overlap part, the temporal relation between e_i and e_j is $\underline{O}$.

$$x_{<i,j,r>} = 1 \qquad \forall r \in \{\underline{B}, \underline{O}\} \wedge r = rel(at_i, at_j) \tag{9}$$

where function $rel(at_i,at_j)$ returns one of the four temporal relations between at_i and at_j. Due to the reflexivity constraint, it is unnecessary to enforce the constraint on *after* relation.

Since the ACE corpus uses *Timex2* to annotate all temporal expressions, the *Time* arguments need to be normalized. In this study, we first divide all time tags into two categories: time point and time duration. Then, we implement a simple rule-based tool based on the DCT (Document Create Time) to normalize all time points as "*year:month:day: hour:minute*" and all time durations as (*begintime, endtime*) where *begintime* and *endtime* are normalized as the style of time point. As a result, 92.7% of *Time* arguments are normalized correctly.

Event relevance constraint

In a discourse, most of event mentions are normally structured around a specific topic, which acts as a bone to link all the relevant event mentions together into a narration, via various kinds of event relations. Those semantics-based event relations, i.e. event relevance, are thus helpful to infer the temporal relations among event mentions. For example, if there is a causal relation between an *Attack* and a *Die* event mention, it is obvious to infer they have the *Before* temporal relation. That is, a *Die* event is always the result of an *Attack* event.

Specifically, we learn event relevance from the training set by counting the occurrence frequency $f_{<i,j,r>}$ for each event type pair (evt_i, evt_j) (e.g., (*Attack, Die)*) with relation r in the training set. To eliminate

the accidental factor in statistics, we modify the occurrence frequency of an event type pair to 0 when it only appears once in the training set.

Accordingly, we have the following constraint on event relevance, stating that if an event type pair (evt_i, evt_j) only has one occurrence frequency $f_{<i,j,r>}$(larger than 0), the temporal relations of all event mention pairs in the test set with event type pairs (evt_i, evt_j), are assigned with the temporal relation r according to constraint (10), and that if an event type pair (evt_i, evt_j) has two or three occurrence frequencies (i.e., larger than 0), the relations of all event mention pairs with event type pair (evt_i, evt_j), are enforced according to constraint (11).

$$x_{<i,j,r>} = 1 \qquad \forall f_{<i,j,r>} > 0 \wedge \sum_{r' \in SR_1} f_{<i,j,r'>} = 0 \tag{10}$$

$$\sum_{r \in SR_2} x_{<i,j,r>} = 1 \qquad \forall\, f_{<i,j,r>} > 0 \tag{11}$$

where SR_1 refers to the set of all the temporal relations except r and SR_2 refers to the set of all temporal relations whose occurrence frequencies are larger than 0.

Tense constraint

Event tense is also a helpful evidence to infer temporal relations. In the ACE 2005 corpus, each event mention has an annotated tense attribute, whose values are *Past(P)*, *Present(R)* and *Future(F)* and have been used in the baseline. For example, it is normal to infer the *Before* relation between two event mentions whose tense are *Past* and *Future* respectively. Accordingly, we can have the following constraint, stating that if the tense te_i of e_i is *Past* and that of e_j is *Present* or *Future*, the temporal relation of (e_i, e_j) is $\underline{B}$; 2), and that if te_i is *Present* and te_j is *Future*, the temporal relation of (e_i, e_j) is $\underline{B}$.

$$x_{<i,j,\underline{B}>} = 1 \qquad \forall te_i = P \wedge te_j \in \{R, F\} \vee te_i = R \wedge te_j = F \tag{12}$$

Connective constraint

In a discourse, the connective between two adjacent sentences or clauses can largely reveal their discourse relations. For example, the connective "because" illustrates the *Cause* relation. Likewise, the connective between two adjacent event mentions also explicitly unveils their temporal relation. For example, if the preceding event mention is the cause of the succeeding event mention, their temporal relation is *Before*. Besides, we find out that some verbs which represent the meaning of causality can indicate the temporal relation of an event mention pair. Take the following sentence as an example:

*E1: 这起炸弹攻击(EV1: Attack)事件**造成**了 2 个人死亡(EV2: Die)。(This bomb terror (EV1) **caused** two persons to death (EV2).)* *-From CBS20001120.1000.0823*

In sentence E1, the verb 造成 (cause) indicates that the temporal relation between the event mentions *EV1* and *EV2* is *Before*. Hence, we enumerate a set of Chinese verbs (e.g., 导致, 造成, 引起) whose meaning are "cause" and add them into our connective set *CS*. Besides, we find out that only causal and temporal connectives are helpful to infer temporal relations. Therefore, respective connectives are selected from Appendix B of the PDTB 2.0 annotation manual, which provides a list of classified explicit connectives. Finally, we divided all words in CS into two subsets CS_1 and CS_2, according to the statistics from the training set, where all words in CS_1 indicate that the preceding event mention occurs earlier than the succeeding one and all words in CS_2 indicate that the preceding event mention occurs later than the succeeding one.

Accordingly, we have the following constraint on discourse connective, stating that if there is a connective *con* (*con*$\in$*CS*) between two adjacent event mentions e_i and e_j in the same sentence or neighboring sentences, the temporal relation between e_i and e_j depends on whether *con* belongs to CS_1 or CS_2 as follows.

$$\begin{aligned} x_{<i,j,\underline{B}>} &= 1 \qquad \forall con \in CS_1 \\ x_{<i,j,\underline{A}>} &= 1 \qquad \forall con \in CS_2 \end{aligned} \tag{13}$$

Coreference constraint

An event may have more than one mention in a document and these mentions refer to the same event, called coreference events. Take the following two sentences as examples:

E2: 埃塞俄比亚与厄立特里亚 2 3 日在这里举行谈判(EV3: Meet)。(The talk (EV3) between Ethiopia and Eritrea will be held on 23rd.)

E3: 这次谈判(EV4: Meet)的目的是... (The goal of this talk (EV4) is ...) -From XIN20001024.2000.0141

It is obvious that two coreference event mentions EV3 and EV4 must have the same occurrence time and their relation is *Overlap*. Following Do et al. (2012), we also apply this constraint to our model and enforce the following constraint on event coreference, stating that if mentions e_i and e_j are coreferential event mentions, their temporal relation is *Overlap*.

$$x_{<i,j,O>} = 1 \qquad \forall\, cr(e_i, e_j) = true \tag{14}$$

where function *cr()* return true when e_i and e_j are coreferential. Besides, we use the tool described in Teng et. al. (2015) to construct those coreference event chains.

5 Experimentation

In this section, we first evaluate our model for Chinese temporal relation extraction and then report the experimental results on our event-driven fully-annotated Chinese temporal relation corpus.

5.1 Experimental Settings

All the experiments are done on the event-driven fully-annotated Chinese temporal relation corpus, annotated on the top of the ACE 2005 Chinese corpus, as described in Subsection 3.1. We conduct all evaluations with 5-fold cross-validation at document level and each fold contains about 4000 event mention pairs. Following previous studies on temporal relation extraction, we employ *Accuracy* as evaluation metric, which measures the percentage of correctly classified test instances. This metric is the same as micro F-score since each temporal relation of each event mention pair must belong to one of the four relations.

In this study, we use *lp_solve* as the ILP solver which implements the Branch-and-Bound algorithm. It takes less than 3 seconds on average to decode a document on a PC with 3.4Ghz Intel i7 CPU and 16GB memory.

5.2 Experimental Results

Performance comparison

To evaluate the performance of our discourse-level global inference model (DGIM), we compare it with two strong baselines. The first is a classifier-based system, mentioned in Subsection 3.2, originated from the top performing English systems (D'Souza and Ng, 2014; Mirza and Tonelli, 2014). The second (GIM) is a global inference model with the reflexivity and transitivity constraints following Bramsen et al. (2006), Chambers and Jurafsky (2007), and Do et al. (2012). It is worth to note that all annotated data in DGIM are also used in the first classifier-based system. Table 2 compares the performance of two baselines and our inference model DGIM.

Model	Accuracy (%)(Gold events)	Accuracy (%)(Auto events)
Baseline 1 ($C_{E\text{-}E}$)	62.17	36.21
Baseline 2 (GIM)	64.12	37.85
DGIM	68.36	40.92

Table 2. Performance comparison of different models

Table 2 shows that when all event mentions are known, i.e. with gold event mentions, DGIM significantly outperform the two baselines by 6.19% and 4.71% in accuracy respectively. All improvements from two baselines to DGIM are statistically significant ($p<0.000001$, McNemar's test, 2-tailed). Besides, GIM outperforms the classifier-based model by 2.05% in accuracy, indicating the limitation of

the conventional reflexivity and transitivity constraints in previous studies. In comparison, DGIM outperforms GIM by 4.24% in accuracy, indicating the effectiveness of our various discourse-level global constraints.

Table 2 also shows the performance comparison consistency when all the event mentions are automatically extracted, as described in (Li et al., 2013) with the F1-score of 68.2% and 53.7% in event trigger extraction and event argument extraction respectively.

Contributions of discourse-level constraints

Table 3 illustrates the contributions of different discourse-level constraints to our DGIM model with gold event mentions.

System	Accuracy (%)
Baseline 1	62.17
+Reflexivity (7)	+0.29
+Before/After Transitivity (8)	+0.54
+Argument Time (9)	+2.23
+Event relevance (10,11)	+1.26
+Tense (12)	+0.48
+Connective (13)	+0.74
+Coreference (Learned) (14)	+0.69
+Coreference (Gold) (14)	+0.86

Table 3. Contributions of different discourse-level constraints to temporal relation extraction

1) The conventional reflexivity and before/after transitivity constraints slightly improve the accuracy. This is not as effective as that on TimeBank, due to that those wrong probabilities produced by the event-event classifier will be incorrectly propagated to more temporal relations of event mention pairs since each document in our corpus is fully-annotated. Although the improvements of above two constraints are limited, they can interact with others to either improve the performance or reduce the time complexity. For example, the reflexivity constraint can simplify our discourse-level constraints, since if we have applied a constraint to an event pair, it is unnecessary to apply the opposite constraint to their inverses.
2) The argument *Time* constraint gains most with 2.23% in accuracy, while the tense constraint gains least among all constraints. Different from TimeBank, an event always has a *Time* role in the ACE 2005 corpus. If both event mentions have the *Time* arguments, we have a high confidence to determine their temporal relation. The error of the argument *Time* constraint mainly comes from those event mentions with a vague time (e.g., 最近 (recently), 日前 (a few days ago)).
3) Intuitively, tense can clearly identify the temporal relation of two event mentions if they have different tenses. However, our preliminary experiment shows that this constraint harms the accuracy if we apply it to the whole document. The reason is that the tenses annotated in the ACE 2005 corpus are relative ones based on the statement of a sentence itself. For example, although two *Transport* event mentions "他要**来**美国" (He will **come** to U.S.) and "他**来**到了美国" (He **arrive**d U.S.) have the *Future* and *Past* tenses respectively, they are coreferential events with different statement times. In this study, we only enforce this constraint on the sentence level.
4) The event relevance constraint gains an improvement of 1.26% in accuracy. This verifies that relevant events always occur in a regular order. In our experiments, we extract total 65 event type pairs to construct this constraint. For example, an *Arrest-Jail* event often occurs after an *Attack* event. Although this constraint contributes third, this is far from our expectation. Our error analysis shows that this constraint introduces lots of wrong predictions due to the lack of deep semantics, which is worth exploring in our future work.
5) Although the improvement of the connective constraint is not significant enough with a gain of 0.74% in accuracy, it achieves a high precision in predicting almost all event mention pairs enforced by this constraint, through discourse connective like "因为" (because), "后" (after) and "造成" (cause).
6) The conference constraint gains an improvement of 0.69% and 0.86% in accuracy with automatically learned (with F1-score of 61.7% using the tool described in Teng et. al. (2015)) and gold conference respectively. These figures are much smaller than those in Do et al. (2012) (2.33% for

learned 9.91% for gold). This is largely due to that although 40% of event mentions in our corpus are coreferential, the accuracy of the temporal relation among two coreferential event mentions, produced by the baseline classifier-based model, is already very high (86.2%). In comparison, the baseline in Do et al. (2012) only achieves ~40% in accuracy. Besides, Do et al. (2012) employed a much smaller corpus of only 20 documents, in comparison with 163 documents in our study.

Performance of different temporal relations

Table 4 shows the accuracy on four temporal relations with gold event mentions. From Table 4, the performance of temporal relations *Before*, *After* and *Overlap* is higher than that of *Unknown*, much due to the low recall of the *Unknown* relation, caused by its low percentage (7.1%). Compared to the two baselines, our DGIM improves the F1-scores for all temporal relations, with the highest improvement on the *Before* and *After* relations and the lowest improvement on the *Unknown* relation, much due to the fact that almost all discourse-level constraints focus on relations *Before*, *After* and *Overlap*.

Relation	Baseline 1	Baseline 2	DGIM
Before	63.43	65.30	72.21
After	63.44	65.30	72.21
Overlap	64.50	66.40	69.17
Unknown	45.60	47.20	47.44

Table 4 Accuracies (%) of four temporal relations

Analysis on adjacent or nonadjacent sentences

Table 5 shows the percentages and performance of event mention pairs in same sentence, adjacent sentences and nonadjacent sentences with gold event mentions. We can find out that 56.3% of event mention pairs are in nonadjacent sentences. Those event mention pairs in the same sentence achieve the highest accuracy while those in nonadjacent sentences gains least among all three types. Table 5 also proves that our DGIM outperforms two baselines in all three sentence levels significantly.

Distance	Rate(%)	Baseline 1	Baseline 2	DGIM
Same	18.6	68.13	69.13	72.25
Adjacent	25.1	64.54	65.47	69.09
Nonadjacent	56.3	59.15	61.87	66.75

Table 5. Accuracies (%) of event mention pairs in same sentence (Same), adjacent sentences (Adjacent) and nonadjacent sentences (Nonadjacent)

6 Conclusion

This paper first annotates an event-driven fully-annotated Chinese temporal relation corpus and then presents a novel discourse-level global inference model, enforced by various kinds of discourse-level constraints derived from event semantics, to recognize temporal relations of Chinese events in document-level, especially in nonadjacent sentences. Evaluation on an event-driven fully-annotated Chinese temporal relation corpus justifies the effectiveness of our discourse-level global inference model over two strong baselines.

Although our model focuses on Chinese, it can be naturally applied to other languages (e.g., English). Our future work will focus on how to introduce more linguistics-driven knowledge to boost our model and construct a joint modelling of temporal event relation extraction and event extraction on both Chinese and English.

Acknowledgements

The authors would like to thank three anonymous reviewers for their comments on this paper. This research was supported by the National Natural Science Foundation of China under Grant No. 61472265, No. 61402314 and No. 61331011, and partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

Reference

Philip Bramsen, Pawan Deshpande, Yoong Keok Lee, and Regina Barzilay. 2006. Inducing temporal graphs. *In Proceedings of EMNLP 2006*, pages 189-198, Sydney, Australia.

Taylor Cassidy, Bill McDowell, Nathanael Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. *In Proceedings of ACL 2014*, pages 501-506, Baltimore, MD, USA.

Nathanael Chambers, ShanWang and Dan Jurafsky. 2007. Classifying temporal relations between events. *In Proceedings of ACL 2007*, pages 173-176, Prague, Czech Republic.

Nathanael Chambers and Dan Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. *In Proceedings of EMNLP 2008*, pages 698-706, Singapore.

Nathanael Chambers, Taylor Cassidy, Bill McDowell and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273-284.

Yuchang Cheng, Masayuki Asahara and Yuji Matsumoto. 2008. Use of event types for temporal relation identification in Chinese text. *In Proceedings of the 6th SIGHAN Workshop on Chinese Language Processing*, pages 31-38, Hyderabad, India.

Pascal Denis and Philippe Muller. 2011. Predicting globally-coherent temporal structures from texts via endpoint inference and graph decomposition. *In Proceeding of IJCAI 2011*, pages 1788-1793, Barcelona, Spain.

Jennifer D'Souza and Vincent Ng. 2013. Classifying temporal relations with rich linguistic knowledge. *In Proceedings of NAACL-HLT 2013*, pages 918-927, Atlanta, Georgia.

Quang Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. *In Proceedings of EMNLP 2012*, pages 677-687, Jeju Island, Korea.

Natsuda Laokulrat, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Uttime: Temporal relation classification using deep syntactic features. *In Proceedings of the 7th International Workshop on Semantic Evaluation*, pages 88-92, Atlanta, Georgia, USA.

Mirella Lapata and Alex Lascarides. 2006. Learning sentenceinternal temporal relations. *Journal of AI Research*, 27:85-117.

Wenjie Li, Kam-Fai Wong, Guihong Cao and Chunfa Yuan. 2004. Applying machine learning to Chinese temporal relation resolution. *In Proceedings of ACL 2004*, pages 582-588, Barcelona, Spain.

Peifeng Li, Qiaoming Zhu, and Guodong Zhou. 2013. Joint modeling of argument identification and role determination in Chinese event extraction with discourse-level information. *In Proceedings of IJCAI 2013*, pages 2120-2126, Beijing, China.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. *In Proceedings of ACL 2006*, pages 753-760, Sydney, Australia.

Paramita Mirza and Sara Tonelli. 2014. Classifying temporal relations with simple features. I*n Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 308-317, Gothenburg, Sweden.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, et al. 2003. The TimeBank corpus. *Corpus linguistics*, 647-656.

Dan Roth and wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. *In Proceedings of CoNLL 2004*, pages 1-8, Boston, MA, USA.

Jiayue Teng, Peifeng Li, Qiaoming Zhu. 2015. Chinese event co-reference resolution based on trigger semantics and combined features, *In Proceedings of Chinese Lexical Semantic Workshop (CLSW 2015)*, pages 494-503, Beijing, China.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. *In Proceedings of the 7th International Workshop on Semantic Evaluation*, pages 1-9, Atlanta, Georgia, USA.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. *In Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 75-80, Stroudsburg, PA, USA.

Marc Verhagen, Roser Saur´ı, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 task 13: TempEval-2. *In Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 57-62, Stroudsburg, PA, USA.

Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with Markov Logic. *In Proceedings of the Joint Conference of ACL-AFNLP*, pages 405-413, Singapore.

Improved Relation Classification by Deep Recurrent Neural Networks with Data Augmentation

Yan Xu,[1,*,‡] **Ran Jia,**[1,*] **Lili Mou,**[1] **Ge Li,**[1,†] **Yunchuan Chen,**[2] **Yangyang Lu,**[1] **Zhi Jin**[1,†]
[1]Key Laboratory of High Confidence Software Technologies (Peking University),
Ministry of Education, China; Institute of Software, Peking University
{xuyan14,lige,luyy11,zhijin}@sei.pku.edu.cn
{jiaran1994,doublepower.mou}gmail.com
[2]University of Chinese Academy of Sciences chenyunchuan11@mails.ucas.ac.cn

Abstract

Nowadays, neural networks play an important role in the task of relation classification. By designing different neural architectures, researchers have improved the performance to a large extent in comparison with traditional methods. However, existing neural networks for relation classification are usually of shallow architectures (e.g., one-layer convolutional neural networks or recurrent networks). They may fail to explore the potential representation space in different abstraction levels. In this paper, we propose deep recurrent neural networks (DRNNs) for relation classification to tackle this challenge. Further, we propose a data augmentation method by leveraging the directionality of relations. We evaluated our DRNNs on the SemEval-2010 Task 8, and achieve an F_1-score of 86.1%, outperforming previous state-of-the-art recorded results.[1]

1 Introduction

Classifying relations between two entities in a given context is an important task in natural language processing (NLP). Take the following sentence as an example: "Jewelry and other smaller [valuables]$_{e_1}$ were locked in a [safe]$_{e_2}$ or a closet with a deadbolt." The marked entities *valuables* and *safe* are of relation Content-Container(e_1, e_2). Relation classification plays a key role in various NLP applications, and has become a hot research topic in recent years.

Nowadays, neural network-based approaches have made significant improvement in relation classification, compared with traditional methods based on either human-designed features (Kambhatla, 2004; Hendrickx et al., 2009) or kernels (Bunescu and Mooney, 2005; Plank and Moschitti, 2013). For example, Zeng et al. (2014) and Xu et al. (2015a) utilize convolutional neural networks (CNNs) for relation classification. Xu et al. (2015b) apply long short term memory (LSTM)-based recurrent neural networks (RNNs) along the shortest dependency path. Nguyen and Grishman (2015) build ensembles of gated recurrent unit (GRU)-based RNNs and CNNs.

We have noticed that these neural models are typically designed in shallow architectures, e.g., one layer of CNN or RNN, whereas evidence in the deep learning community suggests that deep architectures are more capable of information integration and abstraction (Graves et al., 2013; Hermans and Schrauwen, 2013; Irsoy and Cardie, 2014). A natural question is then whether such deep architectures are beneficial to the relation classification task.

In this paper, we propose the deep recurrent neural networks (DRNNs) to classify relations. The deep RNNs can explore the representation space in different levels of abstraction and granularity. By visualizing how RNN units are related to the ultimate classification, we demonstrate that different layers indeed learn different representations: low-level layers enable sufficient information mix, while high-level layers are more capable of precisely locating the information relevant to the target relation between

[*]Equal contribution. [†]Corresponding authors. [‡]Yan Xu is currently a research scientist at Inveno Co., Ltd.
[1]Code released on https://sites.google.com/site/drnnre/

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1461–1470, Osaka, Japan, December 11-17 2016.

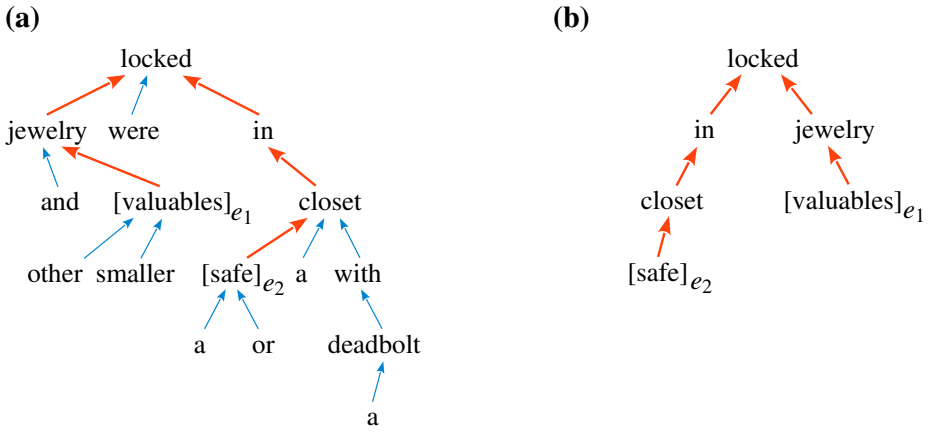


Figure 1: (a) The dependency parse tree corresponding to the sentence "Jewelry and other smaller [valuables]$_{e_1}$ were locked in a [safe]$_{e_2}$ or a closet with a deadbolt." Red arrows indicate the shortest dependency path between e_1 and e_2. (b) The augmented data sample.

two entities. Following our previous work (Xu et al., 2015b), we leverage the shortest dependency path (SDP, Figure 1) as the backbone of our RNNs.

We further observe that the relationship between two entities are directed. Two sub-paths, separated by entities' common ancestor, can be mapped to `subject-predicate` and `object-predicate` components of a relation. By changing the order of these two sub-paths, we obtain a new data sample with the inversed relationship (Figure 1b). Such data augmentation technique can provide additional data samples without using external data resources.

We evaluated our proposed method on the SemEval-2010 relation classification task. Even if we do not apply data augmentation, the DRNNs model has achieved a high performance of 84.2% F_1-score with a depth of 3, but the performance decreases when the depth is too large. This is because the deep RNN is a large model, which necessitates more data samples for training. Applying data augmentation can alleviate the problem of data sparseness and sustain a deeper RNN to improve the performance to 86.1%. The results show that both our deep networks and the data augmentation strategy have contributed to the relation classification task, and that they are coupled well together for further performance improvement.

The rest of this paper is organized as follows. Section 2 reviews related work; Section 3 describes our DRNNs model in detail. Section 4 presents in-depth experimental results. Finally, we have conclusion in Section 5.

2 Related Work

Traditional methods for relation classification mainly fall into two groups: feature-based or kernel-based. The former approaches extract different types of features and feed them into a classifier, e.g., a maximum entropy model (Kambhatla, 2004). Various features, including lexical, syntactic, as well as semantic ones, are shown to be useful to relation classification (Hendrickx et al., 2009). By contrast, kernel-based methods do not have explicit feature representations, but require predefined similarity measure of two data samples. Bunescu and Mooney (2005) design a kernel along the shortest dependency path (SDP) between two entities by observing that the relation strongly relies on SDPs. Plank and Moschitti (2013) combine structural information and semantic information in a tree kernel.

Neural networks have now become a prevailing technique in this task. Socher et al. (2011) design a recursive neural network along the constituency parse tree. Hashimoto et al. (2013), also on the basis of recursive networks, emphasize more on important phrases; Ebrahimi and Dou (2015) restrict recursive networks to SDP. In our previous study (Xu et al., 2015b), we introduce SDP-based recurrent neural network to classify relations.

Zeng et al. (2014), on the other hand, apply CNNs to relation classification. Along this line, dos Santos et al. (2015) replace the common softmax loss function with a ranking loss in their CNN model. Xu et

al. (2015a) design a negative sampling method for SDP-based CNNs.

Besides, representative hybrid models of CNNs and recursive/recurrent networks include Liu et al. (2015) and Nguyen and Grishman (2015).

3 The Proposed Methodology

In this section, we describe our methodology in detail. Subsection 3.1 provides an overall picture of our DRNNs model. Subsections 3.2 and 3.3 describe deep recurrent neural networks. The proposed data augmentation technique is introduced in Subsection 3.4. Finally, we present our training objective in Subsection 3.5.

3.1 Overview

Figure 2 depicts the overall architecture of the DRNNs model. Given a sentence and its dependency parse tree,[1] we follow our previous work (Xu et al., 2015b) and build DRNNs on the shortest dependency path (SDP), which serves as a backbone. In particular, an RNN picks up information along each sub-path, separated by the common ancestor of marked entities. Also, we take advantage of four information channels, namely, word embeddings, POS embeddings, grammatical relation embeddings, and WordNet embeddings.

Different from Xu et al. (2015b), we design deep RNNs with up to four hidden layers so as to capture information in different levels of abstraction. For each RNN layer, max pooling gathers information from different recurrent nodes. Notice that the four channels (with eight sub-paths) are processed in a similar way. Then all pooling layers are concatenated and fed into a hidden layer for information integration. Finally, we have a softmax output layer for classification.

3.2 Recurrent Neural Networks on Shortest Dependency Path

In this subsection, we introduce a single layer of RNN based on SDP, serving as a building block of our deep architecture.

Compared with a raw word sequence or a whole parse tree, the shortest dependency path (SDP) between two entities has two main advantages. First, it reduces irrelevant information; second, grammatical relations between words focus on the action and agents in a sentence and are naturally suitable for relation classification. Existing studies have demonstrated the effectiveness of SDP (Ebrahimi and Dou, 2015; Liu et al., 2015; Xu et al., 2015b; Xu et al., 2015a); details are not repeated here.

Focused on the SDP, an RNN keeps a hidden state vector $\boldsymbol{h}$, changing with the input word at each step accordingly. Concretely, the hidden state $\boldsymbol{h}_t$, for the t-th word in the sub-path, depends on its previous state $\boldsymbol{h}_{t-1}$ and the current word's embedding $\boldsymbol{x}_t$. For the simplicity and without loss of generality, we use vanilla recurrent networks with perceptron-like interaction, that is, the input is linearly transformed by a weight matrix and non-linearly squashed by an activation function, i.e.,

$$\boldsymbol{h}_t = f(W_{\text{in}}\boldsymbol{x}_t + W_{\text{rec}}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h) \tag{1}$$

where W_{in} and W_{rec} are weight matrices for the input and recurrent connections, respectively. $\boldsymbol{b}_h$ is a bias term, and f is a non-linear activation function (ReLU in our experiment).

3.3 Deep Recurrent Neural Networks

Although an RNN, as described above, is suitable for picking information along a sequence (a subpath in our task) by its iterative nature, the machine learning community suggests that deep architectures may be more capable of information integration, and can capture different levels of abstraction.

A single-layer RNN can be viewed that it is deep along *time steps*. When unfolded, however, the RNN has only one hidden layer to capture the current input, as well as to retain the information in its previous step. In this sense, single-layer RNNs are actually shallow in information processing (Hermans and Schrauwen, 2013; Irsoy and Cardie, 2014).

[1] Parsed by the Stanford parser (de Marneffe et al., 2006).

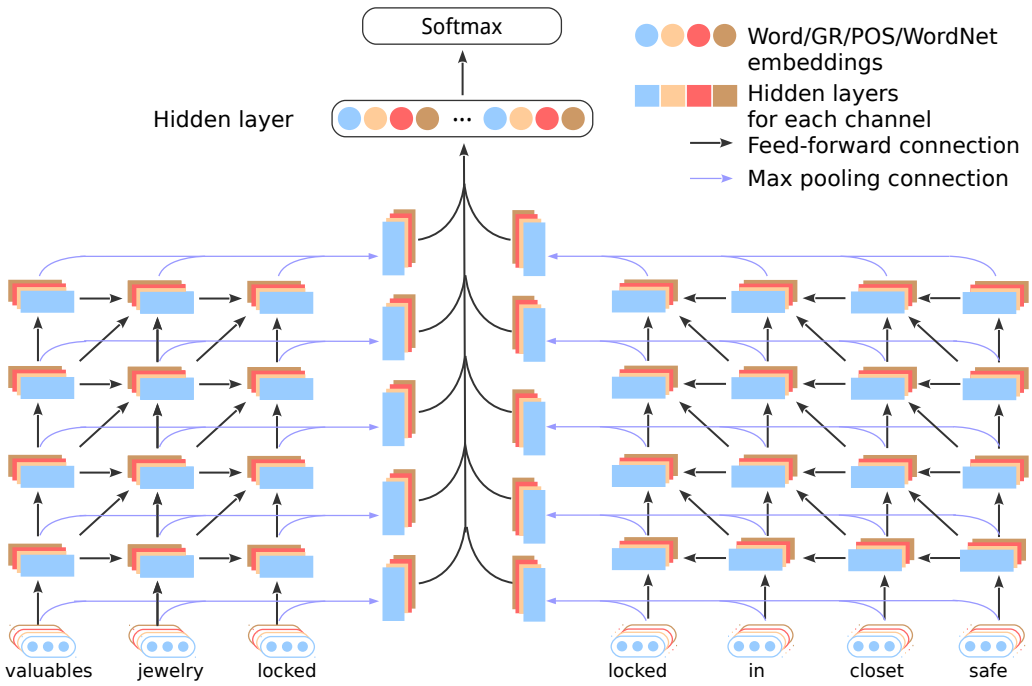


Figure 2: The overall architecture of DRNNs. Two recurrent neural networks pick up information along the shortest dependency path, separated by its common ancestor. We use four information channels, namely words, part-of-speech tags, grammatical relations (GR), and WordNet hypernyms.

In the relation classification task, words along SDPs provide information from different perspectives. On the one hand, the marked entities themselves are informative. On the other hand, the entities' common ancestor (typically verbs) tells how the two entities are related to each other. Such heterogeneous information might necessitate more complex machinery than a single RNN layer.

Following such intuition, we investigate deep RNNs by stacking multiple hidden layers on the top of one another, that is, every layer treats its previous layer as input, and computes its activation similar to Equation 1. Formally, we have

$$\boldsymbol{h}_t^{(i)} = f(W_{\text{in}}^{(i-1)}\boldsymbol{h}_t^{(i-1)} + W_{\text{rec}}^{(i)}\boldsymbol{h}_{t-1}^{(i)} + W_{\text{cross}}^{(i-1)}\boldsymbol{h}_{t-1}^{(i-1)} + \boldsymbol{b}^{(i)}) \tag{2}$$

where the subscripts refer to time steps, and superscripts indicate the layer number. To enhance information propagation, we add a "cross" connection for hidden layers ($i \geq 2$) from the lower layer in the previous time step, given by $W_{\text{cross}}^{(i-1)}\boldsymbol{h}_{t-1}^{(i-1)}$ in Equation 2. (See also $\nearrow$ and $\nwarrow$ arrows in Figure 2).

3.4 Data Augmentation

Neural networks, especially deep ones, are likely to be prone to overfitting. The SemEval-2010 relation classification dataset, we use, comprises only several thousand samples, which may not fully sustain the training of deep RNNs.

To mitigate this problem, we propose a data augmentation technique for relation classification by making use of the directionality of relationships.

The two sub-paths

$[\text{valuables}]_{e_1} \rightarrow \text{jewelry} \rightarrow \text{locked}$

$\text{locked} \leftarrow \text{in} \leftarrow \text{closet} \leftarrow [\text{safe}]_{e_2}$

in Figure 1, for example, can be mapped to the `subject-predicate` and `object- predicate` components in the relation `Content-Container`(e_1, e_2). If we change the order of these two sub-paths, we obtain

$[\text{safe}]_{e_1} \rightarrow \text{closet} \rightarrow \text{in} \rightarrow \text{locked}$
$\text{locked} \leftarrow \text{jewelry} \leftarrow [\text{valuables}]_{e_2}$

Then the relationship becomes `Container-Content`(e_1, e_2), which is exactly the inverse of `Content-Container`(e_1, e_2). In this way, we can augment the dataset without using additional resources.

3.5 Training Objective

For each recurrent layer and embedding layer (over each sub-path for each channel), we apply a max pooling layer to gather information. In total, we have 40 pools, which are concatenated and fed to a hidden layer for information integration.

Finally, a softmax layer outputs the estimated probability that two sub-paths (s^{left} and s^{right}) are of relation r. For a single data sample i, we apply the standard cross-entropy loss, denoted as $J(s_i^{\text{left}}, s_i^{\text{right}}, r_i)$. With the data augmentation technique, our overall training objective is

$$J = \sum_{i=1}^{m} J(s_i^{\text{left}}, s_i^{\text{right}}, r_i) + J(s_i^{\text{right}}, s_i^{\text{left}}, r_i^{-1}) + \lambda \sum_{i=1}^{\omega} \|W_i\|_F$$

where r^{-1} refers to the inverse of relation r. m is the number of data samples in the original training set. ω is the number of weight matrices in DRNNs. λ is a regularization coefficient, and $\|\cdot\|_F$ denotes Frobenius norm of a matrix.

For decoding (predicting the relation of an unseen sample), the data augmentation technique provides new opportunities, because we can use the probability of $r(e_1, e_2)$, $r^{-1}(e_2, e_1)$, or both. Section 4.3 provides detailed discussion.

4 Experiments

In this section, we present our experiments in detail. Subsection 4.1 introduces the dataset; Subsection 4.2 describes hyperparameter settings. We discuss the details of data augmentation in Subsection 4.3 and the rationale for using RNNs in Subsection 4.4. Subsection 4.5 compares our DRNNs model with other methods in the literature. In Subsection 4.6, we have quantitative and qualitative analysis of how the depth affects our model.

4.1 Dataset

We evaluated our DRNNs model on the SemEval-2010 Task 8 dataset, which is an established benchmark for relation classification (Hendrickx et al., 2009). The dataset contains 8000 sentences for training, and 2717 for testing. We split 800 samples out of the training set for validation.

There are 9 directed relations and an undirected default relation `Other`; thus, we have 19 different labels in total. However, the `Other` class is not taken into consideration when we compute the official measures.

4.2 Hyperparameter Settings

This subsection presents hyperparameters of our proposed model. We basically followed the settings in our previous work (Xu et al., 2015b). Word embeddings were 200-dimensional, pretrained ourselves using `word2vec` (Mikolov et al., 2013) on the Wikipedia corpus; embeddings in other channels were 50-dimensional initialized randomly. The hidden layers in each channel had the same number of units as their embeddings (either 200 or 50); the penultimate hidden layer was 100-dimensional. An ℓ_2 penalty of 10^{-5} was also applied as in Xu et al. (2015b), but we chose the dropout rate by validation with a granularity of 5% for our model variants (with different depths).

We also chose the depth of DRNNs by validation from the set $\{1, 2, \cdots, 6\}$. The 3-layer and 4-layer DRNNs yield the highest performance with and without data augmentation, respectively. Section 4.6 provides both quantitative and qualitative analysis regarding the effect of depth.

We applied mini-batched stochastic gradient descent for optimization, where gradients were computed by standard back-propagation.

Variant of Data augmentation	F_1
No Augmentation	84.16
Augment all relations	83.43
Augment `Other` only	83.01
Augment directed relations only	86.10

Table 1: Comparing variants of data augmentation.

	Depth	
	1	2
CNN	84.01	83.78
RNN	84.43	85.04

Table 2: Comparing CNNs and RNNs (also using F_1-score as the measurement).

4.3 Data Augmentation Details

As mentioned in Section 4.1, the SemEval-2010 Task 8 dataset contains an undirected class `Other` in addition to 9 directed relations (18 classes). For data augmentation, it is natural that the inversed `Other` relation is also in the `Other` class itself. However, if we augment all the relations, we observe a performance degradation of 0.7% (Table 1). We deem the `Other` class contains mainly noise, and is inimical to our model. Then we conducted another experiment where we only augmented the `Other` class. The result verifies our conjecture as we obtained an even larger degradation of 1.1% in this setting.

The pilot experiments suggest that we should take into consideration unfavorable noise when performing data augmentation. In this experiment, if we reverse the directed relations only and leave the `Other` class intact, the performance is improved by a large margin of 1.9%. This shows that our proposed data augmentation technique does help to mitigate the problem of data sparseness, if we carefully rule out the impact of noise.

During validation and testing, we shall decode the target label of an unseen data sample (with two entities e_1 and e_2). Through data augmentation, we are equipped with the probability of $r^{-1}(e_2, e_1)$ in addition to $r(e_1, e_2)$. In our experiment, we tried several settings and chose to use $r^{-1}(e_2, e_1)$ only, because it yields the highest the validation result. We think this is probably because the `Other` class brings more noise to r than r^{-1}, as the `Other` class is not augmented (and hence asymmetric).

We would like to point out that our data augmentation method is a general technique for relation classification, which is not *ad hoc* to a specific dataset; that the methodology for dealing with noise is also potentially applicable to other datasets.

4.4 RNNs vs. CNNs

As both RNNs and CNNs are prevailing neural models for NLP, we are curious whether deep architectures are also beneficial to CNNs. We tried a CNN with a sliding window of size 3 based on SDPs, similar to Xu et al. (2015a); other settings were as our DRNNs.

The results are shown in Table 2. We observe that a single layer of CNN is also effective, yielding an F_1-score slightly worse than our RNN. But the deep architecture hurts the performance of CNNs in this task. One plausible explanation is that, when convolution is performed, the beginning and end of a sentence are typically padded with a special symbol or simply zero. However, the shortest dependency path between two entities is usually not very long ($\sim$4 on average). Hence, sentence boundaries may play a large role in convolution, which makes CNNs vulnerable.

On the contrary, RNNs can deal with sentence boundaries smoothly, and the performance continues to increase with up to 4 hidden layers. (Details are deferred to Subsection 4.6.)

4.5 Overall Performance

Table 3 compares our DRNNs model with previous state-of-the-art methods.[2] The first entry in the table presents the highest performance achieved by traditional feature-based methods. Hendrickx et al. (2009) feed a variety of handcrafted features to the SVM classifier and achieve an F_1-score of 82.2%.

Recent performance improvements on this dataset are mostly achieved with the help of neural networks. In an early study, Socher et al. (2012) build a recursive network on constituency trees, but

[2]This paper was preprinted on arXiv on 14 Jan 2016.

Model	Features	F_1
SVM (Hendrickx et al., 2009)	POS, WordNet, prefixes and other morphological features, depdency parse, Levin classes, PropBank, FanmeNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2
RNN (Socher et al., 2011)	Word embeddings	74.8
	+ POS, NER, WordNet	77.6
MVRNN (Socher et al., 2012)	Word embeddings	79.1
	+ POS, NER, WordNet	82.4
CNN (Zeng et al., 2014)	Word embeddings	69.7
	+ position embeddings, WordNet	82.7
Chain CNN (Ebrahimi and Dou, 2015)	Word embeddings, POS, NER, WordNet	82.7
CR-CNN (dos Santos et al., 2015)	Word embeddings	82.8
	+ position embeddings	84.1
FCM (Yu et al., 2014)	Word embeddings	80.6
	+ dependency parsing, NER	83.0
SDP-LSTM (Xu et al., 2015b)	Word embeddings	82.4
	Word + POS + GR + WordNet embeddings	83.7
DepNN (Liu et al., 2015)	Word embeddings + WordNet	83.0
	Word embeddings + NER	83.6
depLCNN (Xu et al., 2015a)	Word + WordNet + words around nominals	83.7
	+ negative sampling from NYT dataset	85.6
Ensemble Methods (Nguyen and Grishman, 2015)	Word+POS+NER+WordNet embeddings, CNNs, RNNs + Stacking	83.4
	Word+POS+NER+WordNet embeddings, CNNs, RNNs + Voting	84.1
DRNNs	Word+POS+GR+WordNet embeddings w/o data augmentation	84.2
	+ data augmentation	**86.1**

Table 3: Comparison of previous relation classification systems.

achieve a performance worse than Hendrickx et al. (2009). They extend their recursive network with matrix-vector interaction and elevate the F_1-score to 82.4%. Ebrahimi and Dou (2015) restrict the recursive network to SDP, which is slightly better than a sentence-wide network. In our previous study (Xu et al., 2015b), we introduce recurrent neural networks based on SDP and improve the F_1-score to 83.7%.

In the school of convolution, Zeng et al. (2014) construct a CNN on the word sequence; they also integrate word position embeddings, which benefit the CNN architecture. dos Santos et al. (2015) propose a similar CNN model, named CR-CNN, by replacing the common softmax cost function with a ranking-based cost function. By diminishing the impact of the `Other` class, they achieve an F_1-score of 84.1%. Xu et al. (2015a) design an SDP-based CNN with negative sampling, improving the performance to 85.6%.

Hybrid models of CNNs and RNNs do not appear to be very useful, achieving up to an F_1-score of 84.1% (Liu et al., 2015; Nguyen and Grishman, 2015).

Yu et al. (2014) propose a Feature-rich Compositional Embedding Model (FCM), which combines unlexicalized linguistic contexts and word embeddings. They do not use neural networks (at least in the usual sense) and achieve an F_1-score of 83.0%.

Our DRNNs model, along with data augmentation, achieves an F_1-score of 86.1%. Even if we do not apply data augmentation, the DRNNs model yields 84.2% F_1-score, which is also the highest score achieved without special treatment to the noisy `Other` class. The above results show the effectiveness of DRNNs, especially trained with a large (augmented) dataset.

4.6 Analysis of DRNNs' Depth

In this subsection, we analyze the effect of depth in our DRNNs model. We have tested the depth from the set $\{1, 2, \cdots, 6\}$, and plot the results in Figure 3. Initially, the performance increases if the depth is larger in both settings with and without augmentation. However, if we do not augment data, the performance peaks when the depth is 3. Provided with augmented training samples, the F_1-score continues to increase with up to 4 layers, and ends up with an F_1-score of 86.1%.

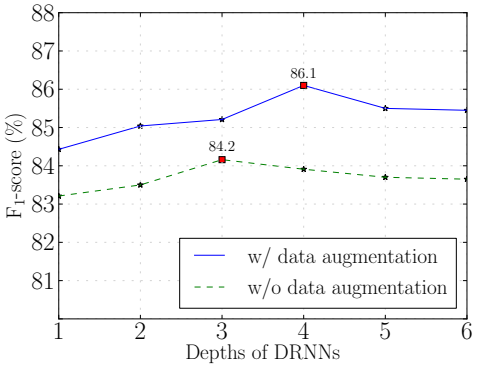


Figure 3: Analysis of the depth.[3]

We next investigate how RNN units in different layers are related to the ultimate task of interest. This is accomplished by tracing back information from pooling layers. Noticing that the pooling layer takes maximum value in each dimension, we can compute how much a hidden layer's units are gathered by pooling for further processing. In this way, we are able to demonstrate the information flow in RNN hidden units. We plot three examples in Figure 4. Here, rectangles refer to RNN hidden layers, unfolded along time. (Rounded rectangles are word embeddings.) The intensity of color reflects the ratio of the pooling proportion.

- Sample 1: "Until 1864 $[\text{vessels}]_{e_1}$ in the service of certain UK public offices defaced the Red Ensign with the $[\text{badge}]_{e_2}$ of their office" with label `Instrument-Agency`(e_2, e_1). Its two sub-paths of SDP are

  $[\text{vessels}]_{e_1} \rightarrow \text{until} \rightarrow \text{defaced}$
  $\text{defaced} \leftarrow \text{with} \leftarrow [\text{badge}]_{e_2}$

  From Figure 4a, we see that entities like *vessels* and *badge* are darker than the verb phrase *defaced with* on the embedding layer. When information is propagating horizontally and vertically, these entities are getting lighter, while the verb phrase becomes darker gradually. Intuitively, we think that, considering the relation `Instrument-Agency`(e_2, e_1), it is less informative with only two entities *vessels* and *badge*. When adding the semantic of verb phrase *defaced with*, we are more aware of the target relation.

- Sample 2: "Most of the $[\text{verses}]_{e_1}$ of the plantation songs had some reference to $[\text{freedom}]_{e_2}$" with label `Message-Topic`(e_1, e_2). Its two sub-paths of SDP are

  $[\text{verses}]_{e_1} \rightarrow \text{of} \rightarrow \text{most} \rightarrow \text{had}$
  $\text{had} \leftarrow \text{reference} \leftarrow \text{to} \leftarrow [\text{freedom}]_{e_2}$

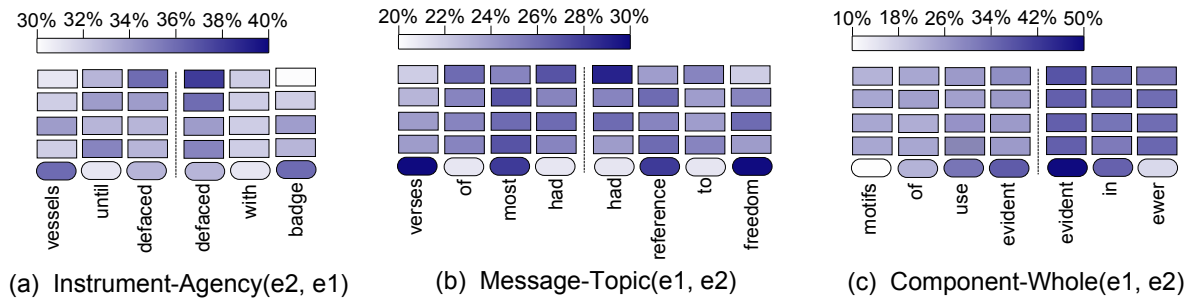
  Similar to Sample 1, we see from Figure 4b that the color of the "pivot" verb *had* is getting darker vertically, and becomes the darkest in the fourth RNN layer, indicating the highest pooling portion. This is probably because *had* links two ends of the relation, `Message` and `Topic`.

- Sample 3: "A more spare, less robust use of classical $[\text{motifs}]_{e_1}$ is evident in a $[\text{ewer}]_{e_2}$ of 1784-85" with label `Component-Whole`(e_1, e_2). Its two sub-paths of SDP are

  $[\text{motifs}]_{e_1} \rightarrow \text{of} \rightarrow \text{use} \rightarrow \text{evident}$
  $\text{evident} \leftarrow \text{in} \leftarrow [\text{ewer}]_{e_2}$

  Different from Figures 4a and 4b, higher layers pay more attention to entities rather than their ancestor. In this example, *motifs* and *ewer* appear to be more relevant to the relation `Component-Whole` than their common ancestor *evident*. The pooling proportion of entities (*motifs*, *ewer*) is increasing, while other words' proportion is decreasing.

[3] Using vanilla RNN with a depth of 1, we obtained a slightly better accuracy in this paper than Xu et al. (2015b).



(a) Instrument-Agency(e2, e1) (b) Message-Topic(e1, e2) (c) Component-Whole(e1, e2)

Figure 4: Visualization of information propagation along multiple RNN layers.

We summarize our findings as follows. (1) Pooled information usually peaks at one or a few words in the embedding layer. This makes sense because there is no information flow in this layer. (2) Information scatters over a wider range in hidden layers, showing that the recurrent propagation does mix information. (3) For a higher-level layer, the network pays more attention to those words that are more relevant to the relation, but whether entities or their common ancestor is more relevant is not consistent among different data samples.

5 Conclusion

In this paper, we proposed deep recurrent neural networks, named DRNNs, to improve the performance of relation classification. The DRNNs model, consisting of several RNN layers, explores the representation space of different abstraction levels. By visualizing DRNNs' units, we demonstrated that high-level layers are more capable of integrating information relevant to target relations. In addition, we have designed a data augmentation strategy by leveraging the directionality of relations. When evaluated on the SemEval dataset, our DRNNs model results in substantial performance boost. The performance generally improves when the depth increases; with a depth of 4, our model reaches the highest F_1-measure of 86.1%.

Acknowledgments

We thank all reviewers for their constructive comments. This research is supported by the National Basic Research Program of China (the 973 Program) under Grant No. 2015CB352201, the National Natural Science Foundation of China under Grant Nos. 61232015, 91318301, 61421091, and 61502014.

References

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the International Conference on Language Resources and Evaluation*, volume 6, pages 449–454.

Cıcero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of 53rd Annual Meeting of the Association for Computational Linguistics*, pages 626–634.

Javid Ebrahimi and Dejing Dou. 2015. Chain based rnn for relation classification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1244–1249.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99.

Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 190–198.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 720–728.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 178–181.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng WANG. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 285–290.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Thien Huu Nguyen and Ralph Grishman. 2015. Combining neural networks and log-linear models to improve relation extraction. *arXiv preprint arXiv:1511.05926*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1498–1507.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 536–540.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794.

Mo Yu, Matthew Gormley, and Mark Dredze. 2014. Factor-based compositional embedding models. In *NIPS Workshop on Learning Semantics*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Relation Extraction with Multi-instance Multi-label Convolutional Neural Networks

Xiaotian Jiang[†‡], Quan Wang[†‡*], Peng Li[†‡], Bin Wang[†‡]
[†]Institute of Information Engineering, Chinese Academy of Sciences
No.89A Minzhuang Road, Beijing 100093, China
[‡]University of Chinese Academy of Sciences
No.19A Yuquan Road, Beijing 100049, China
{jiangxiaotian,wangquan,lipeng,wangbin}@iie.ac.cn

Abstract

Distant supervision is an efficient approach that automatically generates labeled data for relation extraction (RE). Traditional distantly supervised RE systems rely heavily on handcrafted features, and hence suffer from error propagation. Recently, a neural network architecture has been proposed to automatically extract features for relation classification. However, this approach follows the traditional expressed-at-least-once assumption, and fails to make full use of information across different sentences. Moreover, it ignores the fact that there can be multiple relations holding between the same entity pair. In this paper, we propose a multi-instance multi-label convolutional neural network for distantly supervised RE. It first relaxes the expressed-at-least-once assumption, and employs cross-sentence max-pooling so as to enable information sharing across different sentences. Then it handles overlapping relations by multi-label learning with a neural network classifier. Experimental results show that our approach performs significantly and consistently better than state-of-the-art methods.

1 Introduction

Relation extraction (RE), defined as the task of extracting binary relations from plain text, has long been a crucial task in natural language processing. Supervised methods are widely used for this task due to their relatively high performance (Zhou et al., 2005; Surdeanu and Ciaramita, 2007). Such methods, however, usually require intensive human annotation and can be time-consuming. To address this issue, distant supervision is proposed to generate labeled data automatically, by aligning facts in a knowledge base (KB) with sentences mentioning these facts (Mintz et al., 2009; Riedel et al., 2010; Riedel et al., 2013).

Traditional (distantly) supervised RE methods use as input numerous lexical and syntactic features, e.g., POS tags, dependency paths, and named entity tags (Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012). These features are extracted from sentences using various NLP algorithms, thus inevitably have errors. The induced errors become more serious for long sentences (McDonald and Nivre, 2007), which is unfortunately very common in real-world relation extraction corpus (Zeng et al., 2015). Building distant supervision methods on faulty features inevitably leads to error propagation, the main culprit responsible for performance degradation. Recent studies have shown promising results on using deep neural networks for automatic feature extraction (Zeng et al., 2014; Liu et al., 2015; Xu et al., 2015). Particularly, Zeng et al. (2015) proposed a piecewise convolutional neural network (PCNN) architecture, which can build an extractor based on distant supervision. PCNN automatically extracts features with convolutional neural networks, and introduces piecewise max-pooling to better fit the RE scenario. Although PCNN achieves substantial improvements in distantly supervised relation extraction, it still has the following deficiencies.

First, PCNN uses the expressed-at-least-once assumption (Riedel et al., 2010) for labeled data generation, which states that "if two entities participate in a relation, at least one sentence that mentions

*Corresponding author: Quan Wang (wangquan@iie.ac.cn).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1471–1480, Osaka, Japan, December 11-17 2016.

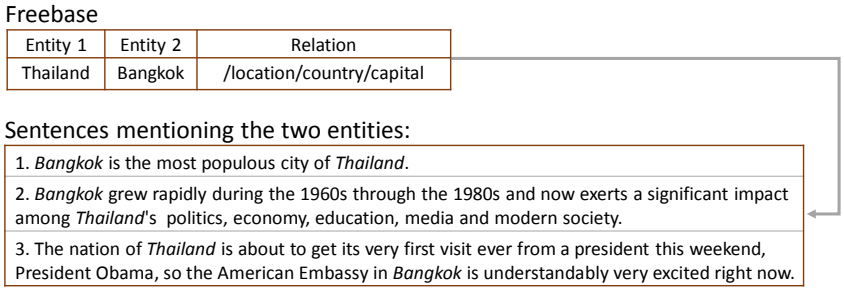


Figure 1: The new assumption states that a relation holding between two entities can be either expressed explicitly or inferred implicitly from all sentences that mention these two entities.

these two entities will express that relation". According to this assumption, PCNN selects only the most likely sentence for each entity pair in training and prediction. We argue, however, that the expressed-at-least-once assumption might be too strong, and selecting a single sentence will definitely lose rich information contained in other sentences. Actually, given two entities participating in a KB relation, it might be difficult to find from the training text the exact single sentence that expresses the relation. Aggregating information available in multiple sentences would probably make the alignment an easier task. Take Figure 1 for example. Given the KB fact (`Thailand`, `/location/country/capital`, `Bangkok`), none of the three sentences mentioning `Thailand` and `Bangkok` expresses the relation of `/location/country/capital`. But if we consider these sentences collectively, we will get more evidence supporting the fact, profiting from the relevant information available in different sentences.

Second, PCNN treats distantly supervised RE as a single-label learning problem and selects for each entity pair a single relation label, ignoring the fact that there might be multiple relations holding between the same entity pair. In fact, as pointed out by Hoffmann et al. (2011), about 18.3% of the distant supervision facts in Freebase that match sentences in the New York Times 2007 corpus have overlapping relations.

In this paper, we propose a multi-instance multi-label convolutional neural network (MIMLCNN) architecture to address the two problems described above. For the first problem, we relax the expressed-at-least-once assumption, and instead assume that "*a relation holding between two entities can be either expressed explicitly or inferred implicitly from all sentences that mention these two entities*" (see Figure 1 for a simple illustration). Therefore, after automatically extracting features within each sentence using a convolutional architecture, we employ cross-sentence max-pooling to select features across different sentences, and then aggregate the most significant features into a vector representation for each entity pair. Since the resultant representation consists of features from different sentences, we successfully make full use of all available information contained in these sentences. For the second problem, we handle overlapping relations by designing various multi-label loss functions in the neural network classifier. The overall architecture is sketched in Figure 2.

The main contributions of this paper can be summarized as follows: (1) We relax the expressed-at-least-once assumption, and propose a more realistic one that naturally enables information sharing from multiple sentences for relation extraction. (2) We propose a multi-instance multi-label convolutional neural network architecture, which handles the multi-label nature of relation extraction. (3) We evaluate our approach on a real-world dataset, and show significant and consistent improvements over state-of-the-art methods.

2 Related Work

Relation extraction is one of the most important tasks in NLP, and has been applied in many practical scenarios (Kordjamshidi et al., 2011; Madaan et al., 2016). Supervised methods has relatively high performance and better practicability, but require massive human annotation, which is both expensive and time consuming. Distant supervision solves this problem by using heuristic assumptions to align triples in a knowledge base with sentences in real-world text corpus, and has been employed in building

large-scale knowledge bases like Knowledge Vault (Dong et al., 2014). A well-known approach in distant supervision is Mintz et al. (2009), which aligns Freebase with Wikipedia articles and extracts relations with logistic regression. Follow-up studies use the feature set developed in this approach, but with deeper understanding on the nature of distant supervision. For example, Riedel et al. (2010) relaxes the assumption used in Mintz et al. (2009) and formulates distant supervision as a multi-instance learning issue; Hoffmann et al. (2011) and Surdeanu et al. (2012) consider overlapping relations between an entity pair. Further effects are also made to model missing data (Ritter et al., 2013), reduce noise (Roth et al., 2013), inject logical background knowledge (Rocktäschel et al., 2015), etc.

In recent years, deep neural network has proven its ability to learn task-specific representation automatically, so that avoiding error propagation suffered by traditional feature-based models. In particular, many neural network approaches have been proposed and shown better performance in relation classification (Zeng et al., 2014; Liu et al., 2015; Xu et al., 2015) and relation extraction (Nguyen and Grishman, 2015). However, these two tasks differ from ours in that relations are extracted at sentence-level, while annotation data is readily available. In distant supervision paradigm, Zeng et al. (2015) is a known neural network model that uses expressed-at-least-once assumption for multi-instance single-label learning. Nevertheless, it selects only one sentence as the representation of an entity pair in training phrase, which wastes the information in the neglected sentences. Besides, it also fails to consider other relations that might hold between this entity pair. The proposed method, on the other hand, leverages evidences collected from all the aligned sentences, and models overlapping relations with multi-label learning.

In traditional supervised learning, an example is usually represented by one instance and one class label. However, there are real-world issues that an example contains multiple instances and has a set of labels. This multi-instance multi-label (MIML) learning scenario was formulated in Zhou et al. (2012), and get widely employed in various tasks (Zha et al., 2008; Zhou and Zhang, 2006; Li et al., 2012). Distant supervised relation extraction is by nature a MIML learning issue, where example is entity pair, instance is sentence aligned with the pair, and label denotes relations. Among previous distant supervision methods, (Surdeanu et al., 2012) formally proposed a multi-instance multi-label framework in a Bayesian framework. In contrast, our method is constructed under a neural network architecture, with the merit of no dependency on lexical and syntactic features.

3 Our Approach

The proposed model takes as input an entity pair (e_1, e_2) as well as all the sentences aligned to this pair, and outputs a set of KB relations that hold between the two entities. As illustrated in Figure 2, our approach consists of three key steps: (1) sentence-level feature extraction, (2) cross-sentence max-pooling, and (3) multi-label relation modeling, detailed as follows.

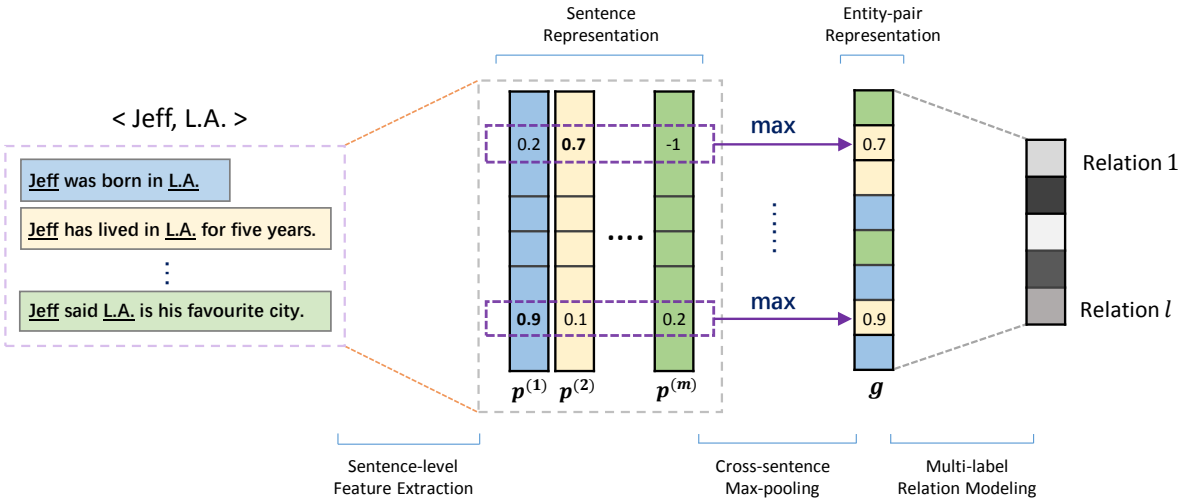


Figure 2: Overall architecture of MIMLCNN.

3.1 Sentence-level Feature Extraction

Sentence-level feature extraction aims to produce a vector feature for each of the aligned sentences. We first pad sentence length to h with zero and transform it to a matrix representation, where each row represents a word token. Convolution, piecewise max-pooling operations are then applied on the matrix to get the vector representation, as illustrated in Figure 3.

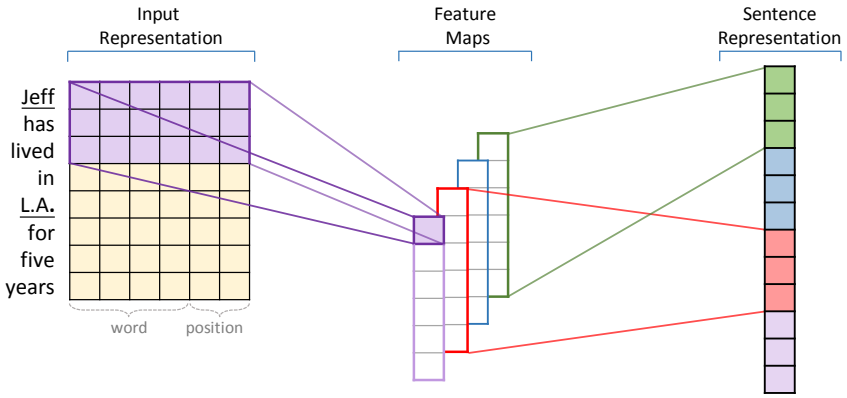


Figure 3: Sentence-level feature extraction using a convolutional architecture.

3.1.1 Input Representation

Two kinds of information are used to construct the input representation for each sentence:

- Raw tokens: We first split the sentence into a sequence of word tokens, then map each token to a d_w dimensional vector called word embedding. The embedding vectors are learned by model training.
- Position features: we use position features (Zeng et al., 2014) to point out the relative positions of a token to e_1 and e_2 in the sentence. Each token has two relative positions, and they are mapped to two different d_p dimensional vectors, separately.

We concatenate the result of these two parts and get matrix $\mathbf{X} \in \mathbb{R}^{h \times d_s}$ as input representation, where $d_s = d_w + 2 * d_p$.

3.1.2 Convolution

The convolution operation aims to extract features from input matrix $\mathbf{X}$, and can be formulated as:

$$c_i = f(\sum_{j=1}^{w_c} \sum_{k=1}^{d_s} \mathbf{W}_{j,k} \mathbf{X}_{j+i-1,k} + b) \tag{1}$$

Here $\mathbf{W} = \mathbb{R}^{w_c \times d_s}$ is a convolutional matrix, where w_c is the width of convolution window; $b \in \mathbb{R}$ is a bias; $f(\cdot)$ is a non-linear function such as Tanh, ReLU. A feature map $\mathbf{c} = [c_1, c_2, \cdots, c_{(h-w_c+1)}]$ is produced by sliding convolution window down the sentence and applying this function at each valid position. To extract n features from the sentence, we repeat the above process with different $\mathbf{W}, b$ for n times. The resultant feature maps are then stacked to construct matrix $\mathbf{C} \in \mathbb{R}^{n \times (h-w_c+1)}$.

3.1.3 Piecewise Max-pooling

To capture the most important feature, max-over-time pooling is often used to select the maximum activation value in each feature map. Piecewise max-pooling (Zeng et al., 2015) improves this idea by first dividing each feature map $\mathbf{C}_i$ into three components $\{\mathbf{c}_{i1}, \mathbf{c}_{i2}, \mathbf{c}_{i3}\}$ based on the positions of the two entities, and then applying max-over-time pooling on each component. This process is formulated as:

$$p_{ij} = max(\mathbf{c}_{ij}) \qquad 1 \le i \le n, 1 \le j \le 3 \tag{2}$$

When piecewise max-pooling is finished, the results of each feature map are concatenated to form vector $\mathbf{p} \in \mathbb{R}^{3n}$, as the feature representation for this sentence.

3.2 Cross-sentence Max-pooling

In the last subsection, we obtain a feature vector $\mathbf{p}$ for each single sentence, but how to take fully usage of the information across sentences is still worth attention. In this paper, we solve this problem by relaxing expressed-at-least-once assumption as:

Assumption: *A relation holding between two entities can be either expressed explicitly or inferred implicitly from all sentences that mention these two entities.*

That is, we relax the expressed-at-least-once assumption by not only allowing making predictions from evidences in each single sentence, but also allowing making predictions by inferring from evidences in all sentences collectively. By nature of this assumption, we skip sentence-level relation extraction and directly make prediction at entity-pair-level, which is more concerned for downstream application and beneficial for evidence aggregation, as described in Riedel et al. (2010).

We propose cross-sentence max-pooling to take the advantage of this assumption. Suppose there are m sentences aligned with the entity pair, and $p_i^{(j)}$ denotes the i^{th} component of the vector representation of the j^{th} sentence, cross-sentence max-pooling aggregates all sentence representations into an entity-pair-level representation $\mathbf{g} = [g_1, g_2, ..., g_{3n}]$, where:

$$g_i = \max(p_i^{(1)}, p_i^{(2)}, \cdots, p_i^{(m)}) \tag{3}$$

This operation brings the following benefits: First, it aggregates features from each sentence, thus supporting entity-pair-level relation extraction directly. Second, it can collect evidence from different sentences, which enables classifiers to make prediction with evidences from different sentences. Besides, compared with Zeng et al. (2015) who only selects one sentence for training at one time, we take advantage of information from all available sentences in each training iteration.

Other approaches, such as mean-pooling, can also be applied in this phrase, but we use cross-sentence max-pooling for the following reason: We consider that multiple occurrences of a feature do not supply much extra information in entity-pair-level relation extraction. That is, a discriminative signal that appears only once can also be sufficient for extracting a relation. This thinking is embodied in the cross-sentence max-pooling operation, where the maximum activation level of each feature is collected across sentences. In contrast, mean pooling averages activation signals by the number of sentences, so that predictive features may be diluted in the representation of entity-pairs that have multiple mentions. This claim is supported by the experimental results.

3.3 Multi-label Relation Modeling

In distant supervision, there are often multiple relations holding between an entity pair. Existing neural network method adopts multi-instance learning, but with single label. In this paper, we model distant supervision under neural network architecture as a multi-label learning problem.

We first calculate the confidence scores for each label by:

$$\mathbf{o} = \mathbf{W_1 g} + \mathbf{b_1} \tag{4}$$

where matrix $\mathbf{W}_1 \in \mathbb{R}^{3n \times l}$ is the collection of weight vectors for each label; $\mathbf{b}_1 \in \mathbb{R}^l$ is a bias. Afterwards, we apply sigmoid function on each element of the score vector $\mathbf{o}$ to calculate the probability of each relation:

$$p(i|M, \theta) = \frac{1}{1 + e^{-o_i}}, \quad i = \{1, 2, \cdots, l\} \tag{5}$$

where M denotes the set of the aligned sentences, and l is the number of relation labels.

A binary label vector $\mathbf{y}$ is set to indicate the set of true relations holding between the entity pair, where 1 means an relation in the set, and 0 otherwise. This way, NA (meaning there is no relation between the entity pair) is naturally represented as an all-zero vector, the complement of the combinations of positive relations.

It is worth noting that relations are often not independent. For example, if triple (A, *capital*, B) holds, another triple (A, *contains*, B) will hold as well. In our model, dependencies between relations are handled by using a shared entity-pair-level representation for all relation labels.

Following this setting, we design two loss functions for multi-label modeling:

$$Loss_{sigmoid} = -\sum_{i=1}^{l} y_i log(p_i) + (1 - y_i) log(1 - p_i) \tag{6}$$

$$Loss_{squared} = \sum_{i=1}^{l} (y_i - p_i)^2 \tag{7}$$

where $y_i \in \{0, 1\}$ is the true value on label i. In the rest sections of this paper, these two loss functions are denoted by sigmoid loss and squared loss, respectively.

The proposed method is trained in an end-to-end fashion. Loss functions are optimized with Adadelta (Zeiler, 2012), which is an robust variant of Stochastic Gradient Decent (SGD) method and features adaptive learning rate over time. Dropout (Srivastava et al., 2014) is also employed on formula (4) for regularization. Specifically, at training time, each element in $\mathbf{g}$ is randomly dropped out by multiplying a Bernoulli random variable with probability p of being 0. At test time, the learned matrix $\mathbf{W}_1$ is scaled by p (i.e. $\hat{\mathbf{W}}_1 = p\mathbf{W}_1$) before scoring. Given an entity pair, the proposed model selects relations whose probability exceeds 0.5 as predicted labels. If there is no such relation, NA is assigned to this entity pair.

4 Experiments

4.1 Dataset

We evaluate our approach on the basis of NYT10, a dataset developed by (Riedel et al., 2010) and then widely used in distantly supervised relation extraction (Hoffmann et al., 2011; Surdeanu et al., 2012; Zeng et al., 2015). NYT10 was generated by aligning Freebase relations with the New York Times (NYT) corpus, with sentences from the years of 2005 and 2006 used for training and sentences from 2007 used for testing.

We follow (Zeng et al., 2015) and use a filtered version of NYT10 released by them[1]. The filtered version prunes the original NYT10 data slightly by removing (1) duplicated sentences for each entity pair, (2) sentences which have more than 40 tokens between a pair of entities, and (3) sentences with entity names that are substrings of other entity names in Freebase. As a result, some relations with low frequency are removed. Statistics of this dataset is shown in Table 1.

	# EPs	# positive EPs	# negative EPs	# sentences	# relations
Training	65,726	4,266	61,460	112,941	26
Testing	93,574	1,732	91,842	152,416	26

Table 1: Statistics of the filtered NYT10 dataset, where EP denotes entity pair.

4.2 Evaluation Metrics

In the following experiments, we use held-out evaluation. At testing time, predicted triples are judged by comparing them with ground truth triples in the testset. We evaluate the performance of each model with Precision-Recall curve, a common used metric for the ranked retrieval results, and P@N metric.

4.3 Baseline Methods

We select three popular feature-based traditional methods as well as the CNN-based method as baselines. We briefly introduce these baselines as follows:

- PCNN: employed a convolutional neural network based method for relation extraction. In contrast to traditional methods, this method allows for automatic feature extraction from raw text, hence avoiding error propagations. Besides, it also uses piecewise max-pooling for sentence-level relation

[1] http://www.nlpr.ia.ac.cn/cip/~liukang/liukangPageFile/code/ds_pcnns-master.zip

extraction. In the following experiments, we use the PCNN code[1] published on the authors' website, along with the dataset.

- Mintz++: (Mintz et al., 2009) proposed distant supervision paradigm that aligns knowledge base entity pairs with text corpus in relation extraction, thus voiding human annotation. The method uses as input lexical and syntactic features, and multi-class logistic regression for classification. [2]
- Multir: (Hoffmann et al., 2011) pointed out that many entity pairs have more than one relation. Their method models overlapping relations by combining sentence-level relation extraction results into entity-pair-level results, with a deterministic decision.
- MIMLRE: (Surdeanu et al., 2012) proposed a novel multi-instance multi-label approach for distant supervision using a graph model. For each entity pair, this method jointly models its multiple instances and multiple labels. Besides, it also models the correlation between labels.

4.4 Implementation Details

As a common practice in neural network models, word embeddings are initialized with pre-training. We run skip-gram model (Mikolov et al., 2013) on training dataset, and use the obtained word vector to initialize the word embedding part of model input. Position features are randomly initialized with uniform distribution between [-1, 1]. For convenience of comparing with baseline methods, our model uses the same parameter settings as (Zeng et al., 2015). Specifically, At model input layer, we use a mini-batch of 50 entity pairs, set the dimension of word embedding $d_w = 50$ and the dimension of position feature $d_p = 5$. At convolutional layers, windows size w_c is set to 3, and the number of feature maps to $n = 230$. Dropout rate p is set to 0.5. Two Adadelta parameters , $\rho = 0.95$ and $\epsilon = e^{-6}$, are set with default values. For baseline models, we use the codes released by (Surdeanu et al., 2012) [3] and Zeng et al. (2015). Since PCNN and MIMLCNN are influenced by random factors when running on GPU, we run both models with the above-mentioned settings for ten times and use the averaged results in the following comparisons.

4.5 Comparison with Baseline Methods

To evaluate the performance of the proposed method, we first compare it to baseline methods. In the following experiments, we use MIMLCNN to refer to the proposed model with cross-sentence max-pooling and sigmoid loss. Figure 4 shows the resulting precision-recall curve in the most concerned area.

From the curves, we observe that MIMCNN can consistently and significantly outperform all baseline methods in the entire range of recall. Comparing neural network methods with traditional feature-based methods, we can conclude that PCNN exceeds traditional methods for its alleviation of error propagation, while MIMCNN exceeds PCNN for its usage of cross-sentence max-pooling and multi-label modeling. The result indicates that the proposed method has the best sense of exploiting the characteristic of distant supervision in a neural network framework. It is worth emphasising that the best of baseline methods can keep a reasonable precision level (larger than 0.5) when recall is less than 0.17. In contrast, our model can keep the same precision level with recall at 0.28, amounting to a 64% increase. Also note that beyond the truncated recall level (0.35), the curve of our method can extend to recall at 0.66 without any loss of precision. This brings 103% increase at the maximum recall level in comparison with the best of baseline methods.

Table 2 further presents the results using P@N metric. In accordance with our observation in precision-recall curve, MIMLCNN is still the winner at most of the entire P@N levels. It is interesting that both of the neural network methods are all good at predicting top-ranked results compared with traditional feature-based methods, especially MIMLCNN. As N gets smaller, the superiority becomes more evident. At P@10, the precision of MIMLCNN can even reach to 0.90, while neither of the baseline methods can exceed 0.84. Also, MIMLCNN is the only method whose mean value of P@N exceeds 0.7.

[2]Note that in the following experiment we use the Surdeanu et al. (2012) implemented version, which has been reported significantly better performance than the original one.

[3]http://nlp.stanford.edu/software/mimlre.shtml

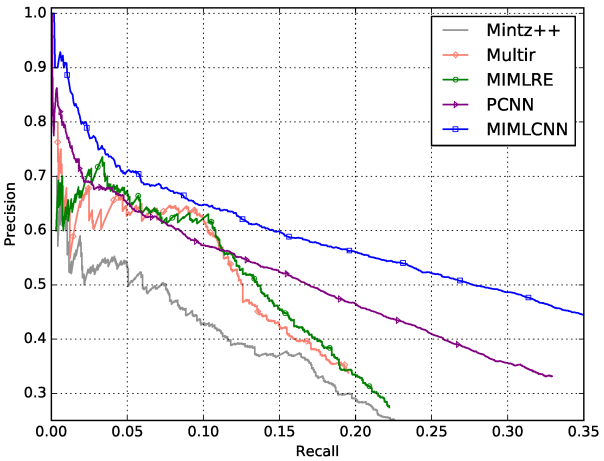


Figure 4: Precision-recall curves of the proposed method and four baselines.

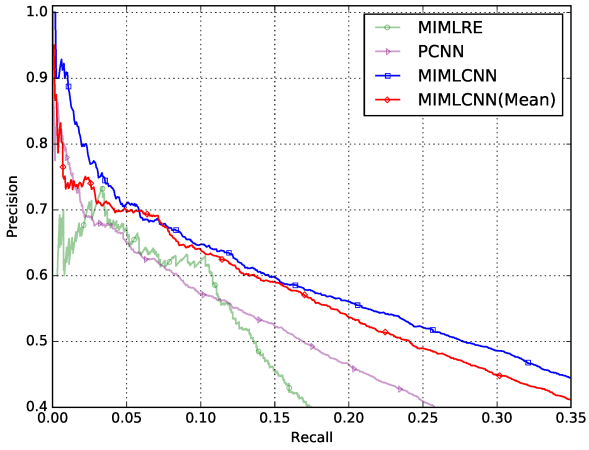
	Mintz++	MultiR	MIMLRE	PCNN	MIMLCNN
P@10	0.70	0.80	0.60	0.84	**0.90**
P@20	0.65	0.65	0.70	0.80	**0.83**
P@30	0.60	0.63	0.63	0.76	**0.80**
P@50	0.54	0.62	0.68	0.72	**0.75**
P@100	0.53	0.62	0.68	0.68	**0.69**
P@200	0.51	0.63	**0.64**	0.62	**0.64**
P@300	0.49	**0.63**	0.62	0.58	0.59
P@500	0.42	0.48	0.51	**0.53**	**0.53**
Mean	0.56	0.63	0.63	0.69	**0.72**

Table 2: P@N results.

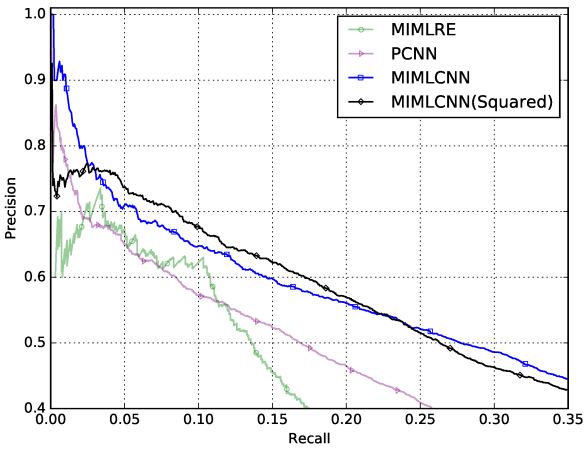
4.6 Effects of Cross-sentence Max-pooling and Multi-label Learning

In this subsection, we empirically prove the effects of cross-sentences max-pooling and multi-label learning, respectively.

In order to prove the effectiveness of cross-sentence max-pooling, we create a baseline method called MIMLCNN(Mean). Comparing with MIMLCNN, this method merely replaces cross-sentence max-pooling with the average of feature representations of all the aligned sentences. Experimental result is presented in Figure 5(b). In almost the entire curves of these two models, MIMLCNN shows better performance. The superiority is especially significant in the front and rearward part of recall levels. This observation supports our claim that cross-sentence max-pooling helps improving performance. It is also interesting that MIMLCNN(Mean) still shows improvements over the baseline methods, though not comparable with MIMLCNN.



(a) Cross-sentence max-pooling.



(b) Multi-label learning.

Figure 5: Effects of cross-sentence max-pooling and multi-label learning. PCNN and MIMLRE are used for reference.

We further compare the effect of using different loss functions in our model, as demonstrated in Figure 5(b). MIMLCNN(Squared) refers to the proposed model with cross-sentence max-pooling and squared loss. From the curves of these two models, we can see that different loss functions have diverse emphases. When we use sigmoid loss (MIMLCNN), most of the improvement resides in recall range [0.1, 0.3], but still remains competitive or slightly better in range [0, 0.1]. Compared with sigmoid loss, using

squared loss brings better performance at the middle area of the curve, but it is also less competitive with respect to the top-ranked results. In contrast with baseline methods, the superiority of MIMLCNN and MIMLCNN(Squared) indicates that multi-label modeling contributes to improving performance in distant supervision.

5 Conclusion

In this paper, we propose a novel neural network method for distant supervision with multi-instance multi-label learning. Given an entity pair, we relax the expressed-at-least-once assumption to take full usage of information from all the aligned sentences with cross-sentence max-pooling, and model multiple relations holding between the entity pair in a neural network architecture. We conduct experiments on a real-world dataset, and prove empirically (1) the proposed method has significantly and consistently better performance than state-of-the-art methods. (2) both cross-sentence max-pooling and multi-label learning take effects. In the future, we would like to further investigate how different loss functions influence performance, and enrich experiments by carrying out human evaluation as well as making detailed analysis on each relation.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments and suggestions. This research is supported by the National Natural Science Foundation of China (grant No. 61402465) and the Strategic Priority Research Program of the Chinese Academy of Sciences (grant No. XDA06030200).

References

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Parisa Kordjamshidi, Paolo Frasconi, Martijn Van Otterlo, Marie-Francine Moens, and Luc De Raedt. 2011. Relational learning for spatial relation extraction from natural language. In *International Conference on Inductive Logic Programming*, pages 204–220. Springer.

Ying-Xin Li, Shuiwang Ji, Sudhir Kumar, Jieping Ye, and Zhi-Hua Zhou. 2012. Drosophila gene expression pattern annotation through multi-instance multi-label learning. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(1):98–112.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, page 285290.

Aman Madaan, Ashish Mittal, Ganesh Ramakrishnan, Sunita Sarawagi, et al. 2016. Numerical relation extraction with minimal supervision. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Ryan T McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 122–131.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of NAACL-HLT*, pages 39–48.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. *NAACL HLT 2013*, pages 74–84.

Alan Ritter, Luke Zettlemoyer, Oren Etzioni, et al. 2013. Modeling missing data in distant supervision for information extraction. *Transactions of the Association for Computational Linguistics*, 1:367–378.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*.

Benjamin Roth, Tassilo Barth, Michael Wiegand, and Dietrich Klakow. 2013. A survey of noise reduction methods for distant supervision. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 73–78. ACM.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Mihai Surdeanu and Massimiliano Ciaramita. 2007. Robust information extraction with perceptrons. In *Proceedings of the NIST 2007 Automatic Content Extraction Workshop (ACE07)*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 536–540.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal*, pages 17–21.

Zheng-Jun Zha, Xian-Sheng Hua, Tao Mei, Jingdong Wang, Guo-Jun Qi, and Zengfu Wang. 2008. Joint multi-label multi-instance learning for image classification. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

Zhi-Hua Zhou and Min-Ling Zhang. 2006. Multi-instance multi-label learning with application to scene classification. In *Advances in neural information processing systems*, pages 1609–1616.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 427–434. Association for Computational Linguistics.

Zhi-Hua Zhou, Min-Ling Zhang, Sheng-Jun Huang, and Yu-Feng Li. 2012. Multi-instance multi-label learning. *Artificial Intelligence*, 176(1):2291–2320.

Named Entity Disambiguation for little known referents: a topic-based approach

Andrea Glaser and Jonas Kuhn
Institute for Natural Language Processing
University of Stuttgart, Germany
andrea.glaser@ims.uni-stuttgart.de

Abstract

We propose an approach to Named Entity Disambiguation that avoids a problem of standard work on the task (likewise affecting fully supervised, weakly supervised, or distantly supervised machine learning techniques): the treatment of name mentions referring to people with no (or very little) coverage in the textual training data is systematically incorrect. We propose to indirectly take into account the property information for the "non-prominent" name bearers, such as nationality and profession (e.g., for a Canadian law professor named *Michael Jackson*, with no Wikipedia article, it is very hard to obtain reliable textual training data). The target property information for the entities is directly available from name authority files, or inferrable, e.g., from listings of sportspeople etc. Our proposed approach employs topic modeling to exploit textual training data based on entities sharing the relevant properties. In experiments with a pilot implementation of the general approach, we show that the approach does indeed work well for name/referent pairs with limited textual coverage in the training data.

1 Introduction

A central subtask for complex information retrieval and natural language understanding problems lies in the determination of what are the real-world entities that the proper names in a text refer to. While for some proper names (e.g., *Henry VIII of England*), the correct referent can be uniquely determined, the great majority of name mentions requires disambiguation. For instance, the name *Michael Jackson* can refer to the famous American singer, a British writer and beer expert, a Canadian actor, and many other people.

The corresponding technical Natural Language Processing (NLP) Task, which is known by various names – Named Entity Disambiguation (e.g., Hoffart et al. (2011)), Entity Linking (e.g., Han et al. (2011)), or Wikification (Mihalcea and Csomai, 2007) – is typically construed as determining for each textual mention of a proper name, which of (typically) several entries in a knowledge base (such as DBpedia or Wikipedia) representing unique referents is the correct one in the given context.

The standard approach to this task is to view it as a supervised classification problem, i.e., training data of textual mentions labeled with the correct disambiguation target are used to induce knowledge about indicative contextual clues for each candidate. A considerable amount of research has gone into the development of effective models (Mann and Yarowsky, 2003; Malin, 2005; Bollegala et al., 2006; Chen and Martin, 2007), and particularly into weakly supervised or distant supervision techniques, i.e., finding ways of exploiting explicit or implicit indications for the correct name references in real-life data (Cohen, 2005; Bunescu and Pasca, 2006; Cucerzan, 2007; Mihalcea and Csomai, 2007; Han et al., 2011; Hoffart et al., 2011). Indeed, for medium-to-high frequency name/referent combinations, it is not hard to harvest the web for suitable training material.

The contribution that we present in this paper is motivated by a type of name/referent pairs that falls outside of the standard training scenario and has so far received little attention from the research com-

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1481–1492, Osaka, Japan, December 11-17 2016.

Entity	Wikipedia
American singer	✓
British writer	✓
Canadian actor	✓
...	
Canadian law professor	×

Table 1: Examples for *Michael Jackson*.

munity[1]: besides the "prominent" bearers of a name, there are almost always others for whom little or no training material can be found by web harvesting, and who will often not even appear in the major reference knowledge bases (DBpedia, Wikipedia etc.). Table 1 shows examples of entities with the proper name *Michael Jackson* and whether or not they have a Wikipedia article. Persons need to be "worthy of notice" to be included in Wikipedia. There is a *Michael Jackson* who is a Canadian law professor, but is not among the 35 or so Michael Jacksons with their own Wikipedia articles. Still, the reference for many of these people *could* be uniquely identified by the name authority files curated by national libraries (e.g., the German Integrated Authority File; "Gemeinsame Normdatei") or other lists that provide unique identification from a specific application perspective, such as staff lists on institutional websites or listings of sportspeople.

Under the established approaches, a trained Named Entity Disambiguation system will incorrectly map those mentions which actually refer to the less known name bearers to the contextually most similar prominent person. In a standard evaluation, these false positives are often negligible since the "non-prominent" mentions are orders of magnitude less frequent. In practice however, any system that is systematically missing out on theoretically identifiable people is problematic (e.g., search engines and Question Answering (QA) systems would return more unwanted or incorrect results, or not find results about the person at all because of an incorrect mapping to a similar, more well-known person).

The purpose of this paper is (i) to propose a general approach for dealing with these cases systematically, and (ii) to present a pilot implementation of this idea using topic modeling. For evaluation, we "simulate" the situation of non-prominent name bearers by leaving documents about them out of the actual training data; this allows us to perform comparative experiments on a manageable collection. As test data we actually use texts containing explicit links to Wikipedia entries, which allows us to circumvent costly manual annotation of the name disambiguation (we call this resource our "silver standard" corpus).

The approach rests on the idea that for non-prominent name bearers, for which no or very little training material containing real mentions is available, we will nevertheless know some characteristic properties – namely, the ones mentioned in a name authority file (typically profession and nationality, as well as age and place of birth, plus possibly institutional affiliation), or similarly properties derivable from other listings. So, while we have no textual training data for the specific person (say, the Canadian law professor *Michael Jackson*), we can use some aggregate of the textual material about different people with the same properties as a proxy (i.e., mentions of all Canadians, of all law scholars, professors etc.).

For our pilot implementation, we conducted several experiments with different parameters (context size around the proper name, number of topics, different corpora) to analyze and evaluate how to the prediction quality is affected. Our results indicate that our approach is indeed very helpful for disambiguating (i) entities for which not much training material is available, and (ii) also for entities with little surrounding context, both of which are useful for many applications.

In Section 2 we discuss related work. In Section 3 we describe our approach and how we extract properties and apply them to new unknown proper names in a document. Section 4 presents the data we use and describes how we create a silver standard corpus for evaluating our system. In Section 5 we describe our experiments and discuss our experimental results and the effects of different parameters. We give our conclusions and future work plans in Section 6.

[1]Sarmento et al. (2009) address the high skewness of distribution of mentions by developing a scalable approach that can cluster a billion mentions.

2 Related Work

Topic models have been used for named entity disambiguation in previous work. Bhattacharya and Getoor (2006) look at data with authors and try to determine the number of individual authors and link them to their corresponding entities. They do not make pairwise decisions of whether two names refer to the same entity, but look at all names in the collection and make a collective decision. For this they use a Latent Dirichlet Allocation (LDA) Model and introduce a hidden variable that captures relationships between entities. With their work they extend work done by Rosen-Zvi et al. (2004) and propose a solution to the duplicate authors problem. Shu et al. (2009) extend the LDA model to include global information from documents to identify authors. All of these approaches look at author data only, they cannot be applied to people in general which we do in our work.

Some work has been done using topic models on knowledge bases. Zhang et al. (2011) train a topic model on a knowledge base, then incorporate the information as a semantic feature by mapping documents with the name mention to the hidden topic space. They use Wikipedia pages to train their model on the first parts of the pages and to test it on the second parts of the pages. Sen (2012) use topic models to learn context information and relations between co-occurring entities. They train their model on only those Wikipedia articles which describe the entities they are dealing with. Yet, to use their model for other entities they would have to use an expanded version of the knowledge base. This would require the knowledge base to contain information about these entities. Han and Sun (2012) combine context compatibility of a referent entity and its context and topic coherence of the entity and the document's main topics. Kataria et al. (2011) use a hierarchical variant of LDA that incorporates information from Wikipedia (words, annotations, and category information) and have a separate topic for each Wikipedia entry in their model. They report results on precision while we also look at the recall and F_1 score, and compare different parameters.

All of these approaches are limited to the entities that occur in the knowledge bases. To apply the approaches to other entities, the knowledge base would have to be extended or data about these entities would have to be collected otherwise. While our approach also uses Wikipedia as a collection to train some of our models, the information we obtain is independent from specific entities and can be applied to any new entity, even if there is no information about them available in Wikipedia.

Li et al. (2013) use information from Wikipedia and an external source (websites referring to entities in Wikipedia obtained by crawling the web). They conduct experiments on two datasets, TAC-KBP 2009 and twitter data about 25 ambiguous and randomly picked entities, however they filter this data to only include entities that occur in Wikipedia, because their approach is also limited to entities in Wikipedia.

Bamman et al. (2013) extend a topic model to learn character types (e.g., {*dark, major, henchman*} or {*shoot, aim, overpower*}) in movies. In subsequent work, Bamman et al. (2014) apply an extended topic model to learn character types in English novels of the 18th and 19th century. They do not identify and link the individual names to their real world entities.

Topic models have also been used in named entity recognition (NER). Guo et al. (2009) use LDA for NER in query. Ritter et al. (2011) extend LDA and take information from Freebase for NER on twitter data. They have a similar problem with ambiguous expressions which they need to solve to determine the correct class (e.g., *China* can belong to several classes such as LOCATION or PERSON). However, they do not identify the actual real world entity of the expression (e.g., there are several cities called *China* in the US and other countries, but they all belong to the class LOCATION).

3 Approach

Our system does not rely on having textual training data for a specific person (e.g., the Canadian law professor *Michael Jackson*). Instead, it uses some aggregate of the textual material about different people that have the same properties (i.e., mentions of all Canadians, of all law scholars, and all professors etc.). This means that our system is independent from existing training data or obtaining training data through other means (e.g., web scraping), and can be applied to any text without the preliminary step of extracting specific information about the entities in the text.

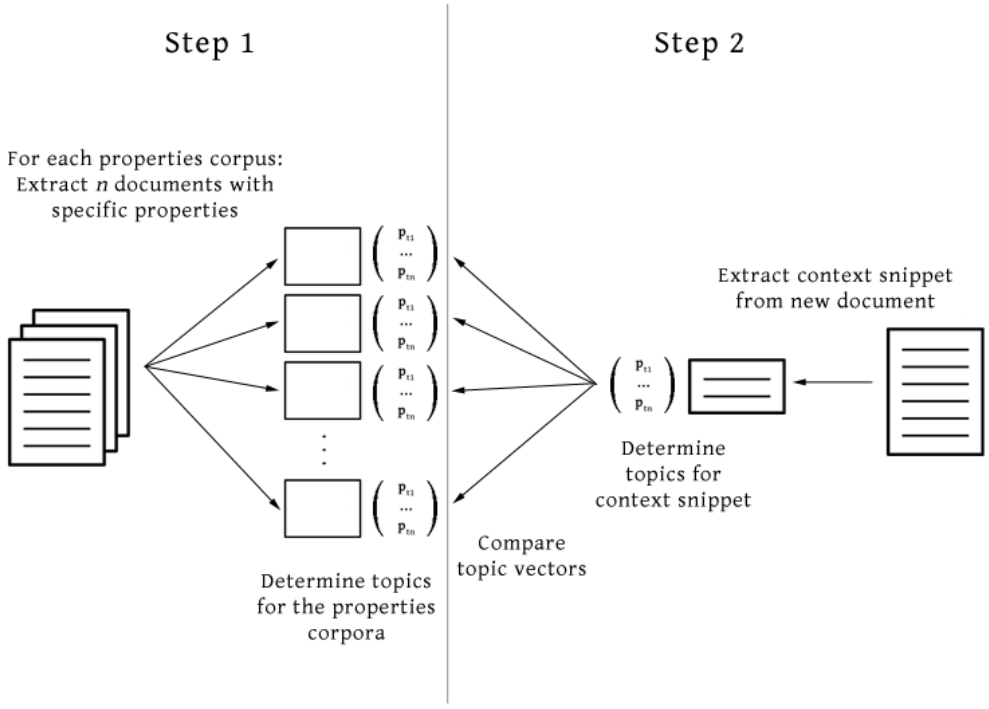


Figure 1: System that learns characteristics of people and uses them to disambiguate unknown people.

Figure 1 shows our system, which is divided into two steps. In the first step, we take a collection of documents and extract documents with specific properties, e.g., documents about singers, authors, and other professions, and documents about Americans, Canadians, and other nationalities. We then concatenate these extracted documents to individual corpora (e.g., a *singer corpora*), which we call "properties corpora". After this, the topics for these properties corpora are determined by using a topic model. In the second step, our system is applied to new unknown proper names. This is done by determining the topics for the proper name based on the chosen context and then comparing these topics with the topics of our properties corpora to find the ones with the most similarity. We describe both steps in more detail in the next two subsections.

3.1 Learning Properties of Persons

In the first step (left side of Figure 1), our system learns characteristic properties that people can have. Properties that can be learned are for example:

- Professions (singer, author, president, tennis player, ...)
- Nationalities (American, Canadian, German, Irish, Japanese, ...)
- Affiliations (university, company, ...)
- Engagements (organization, charity, ...)

In our pilot implementation of the approach we investigate the usefulness of professions and nationalities as properties. We chose these properties because they are the most prominent properties people have and can be obtained without much effort. Our system can be easily expanded with other properties in the future. Helpful properties are ones that are mentioned with the entity, for example, affiliations or engagements.

We first extract documents from a collection (e.g., Wikipedia) with the specific properties. For example, we extract documents about singers, authors, and other professions, and documents about people whose nationality is American, Canadian etc. We then concatenate randomly n of the extracted documents of one category into one corpus. After this we have many small corpora consisting of documents

with certain properties, e.g., a *singer corpus* and an *American corpus*. We call these concatenated documents "properties corpora". These properties corpora are the basis for disambiguating new entities. For example, *Michael Jackson, the American singer* has properties of the corpora *singer* and *American*.

The properties corpora are not very helpful in this form yet because the extracted documents contain a lot of information about many specific people which might not be helpful for disambiguating a new person. To obtain the relevant information from these corpora we use topic models. In natural language processing, topic models can be used to extract and explore topics from a collection of documents. Every topic consists of certain words that characterize this topic. In our work we use Latent Dirichlet Allocation (LDA) (Blei et al., 2003).

Using the topic model consists of two steps. First, we need to train the topic model. For this we use different training collections which we describe in Section 4.1. After the topic model is trained, we apply it to our properties corpora to obtain topic information from them. For example, the singer corpus will have a topic with words related to a singer, e.g., album, music, concert etc. with a high probability given the corpus.

3.2 Using Properties to Disambiguate Persons

In the second step (right side of Figure 1), our system is applied to new unknown proper names. To disambiguate a new person, we need to extract some context around the person (e.g., a sentence or a paragraph). We call the extracted text "context snippet". We use the same trained topic models as in Section 3.1 and apply them to the context snippet to obtain topic information from it.

The topic information we obtain from a document (i.e., a properties corpus or a context snippet) can be represented as a vector of length n, where n is the number of topics used by the topic model. Each entry p_{t_i} in the vector corresponds to the probability of the topic t_i given the document.

We then compare the topic vector of the new person with the topic vectors of each properties corpora to find the corpus that is most similar to the context the person occurs in. For this comparison we use cosine similarity. Let $\mathbf{x}$ be the vector of the new person and $\mathbf{y}_c$ be the vector of the properties corpus c. The cosine similarity between these two vectors is defined as:

$$\cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}_c}{||\mathbf{x}|| \cdot ||\mathbf{y}_c||} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \qquad \text{with } \mathbf{x} = \begin{pmatrix} p_{t_1} \\ \vdots \\ p_{t_n} \end{pmatrix} \text{ and } \mathbf{y}_c = \begin{pmatrix} p_{t_1} \\ \vdots \\ p_{t_n} \end{pmatrix}$$

where p_{t_i} is the probability of topic t_i given the document.

4 Data

We use two collections in this work: (i) the English Wikipedia[2] (February 2014 version) and (ii) the English Gigaword corpus (Graff and Cieri, 2003).

4.1 Corpora for Training Topic Models

We have trained different topic models on different collections and parts of collections to study the effects of different sources.

- **Wikipedia - all ($\mathbf{W}_{all}$):** Wikipedia is an internet encyclopedia which provides a great variety of articles. For this model we used all articles without any restrictions.
- **Wikipedia - living people ($\mathbf{W}_{lp}$):** Wikipedia articles are classified into different categories. We built a model that only uses articles from the category *living people* to see if the properties we can learn from articles about people are more helpful for identifying new people than the properties we can learn from the entire Wikipedia.

[2]wiki dump enwiki-20140203

- **Wikipedia - living people - individual sections (W_{lps}):** Many Wikipedia articles are very long and separated into several sections that are often very different with respect to their topics (e.g., *early life, career*). We want to investigate if we can obtain more helpful topics by using individual sections that are about specific topics as opposed to taking entire articles that consist of many different topics.
- **English Gigaword - nyt (G_{nyt}):** The English Gigaword corpus consists of newswire text data in English. With this model we want to analyze if a newswire corpus provides different topics than an encyclopedia. We only use one part of the English Gigaword corpus for this model, newswire data from the source *The New York Times Newswire Service* (nyt).

		Topics					
Corpus	Docs	100	1000	2500	5000	7500	10000
W_{all}	4.3m	✓	✓	×	×	×	×
W_{lp}	650k	✓	✓	✓	✓	✓	✓
W_{lps}	1.7m	✓	✓	×	×	×	×
G_{nyt}	1.3m	✓	✓	✓	×	×	×

Table 2: Used models.

We experiment with different numbers of topics for each of these collections, ranging from 100 (more coarse-grained topics) to 10000 (more fine-grained topics). Table 2 shows the numbers of topics we used for the different collections. We restrict the larger collections to a maximum of 1000 topics (Wikipedia) and 2500 topics (English Gigaword) due to efficiency reasons. The approximate number of used documents for each collection is listed in the second column of Table 2.

By experimenting with different numbers of topics we want to investigate if more fine-grained topics will give us better results when disambiguating new people, or if a smaller number of topics is enough for the task.

4.2 Properties Corpora

We created properties corpora by extracting Wikipedia articles about people that share these properties. To determine whether a person has certain properties we used metadata in Wikipedia. For our purpose the `short description` field in `Persondata` provides the information we need about nationalities and professions, such as *American singer*. For example, we created a corpus with the property *American* by extracting articles that contain the word *American* in this field, and created a corpus with the property *singer* by extracting articles that contain *singer* in this field.

For each properties corpus we concatenated n random articles with this property. We chose $n = 500$. Some properties are rare in Wikipedia and we extracted less than 500 articles. In these cases we concatenated all articles we could find.

For the experiments in our pilot implementation we created a total of 15 nationalities corpora and 83 professions corpora. Choosing which nationalities and professions to use was done manually, extracting and concatenating documents to the properties corpora was done automatically. To apply the system to entities that are not included in our test system, more properties corpora need to be created. This can be done automatically, for example, by using lists of nationalities and professions.

4.3 Silver Standard Test Corpus

For our experiments we need data about persons that share the same name. Since collecting and manually annotating data is expensive, we automatically created a dataset by exploiting the link structure in Wikipedia.

We chose 14 people with the same names for which we show statistics in Table 3. All names refer to different people found in Wikipedia, with the number of different real world entities for each name ranging from 2 to 38, which can be seen in the second column (Ent).

We then went through all Wikipedia articles and every time we found one of these names linked to another article, we extracted the name, the link, and the context around the name. We experimented

Proper name	Ent	All	Max	Avg
David Mitchell	9	201	111	22.56
David Thomas	9	85	47	9.44
Jack Johnson	8	427	230	53.38
John Edwards	7	418	400	59.71
John Smith	38	466	107	12.11
John Williams	30	852	650	28.37
Michael Collins	7	443	364	63.29

Proper name	Ent	All	Max	Avg
Michael Jackson	12	3968	3899	330.67
Michael Moore	9	566	532	62.89
Paul Williams	13	385	149	29.62
Peter Müller	2	15	10	7.50
Richard Burton	4	606	592	151.50
Roger Taylor	3	79	39	26.33
Tony Martin	13	275	91	21.15

Table 3: Statistics for extracted entities. Numbers are counts.

with three different context sizes: (i) the sentence around the entity, (ii) the paragraph around the entity, (iii) the section around the entity. The third column in Table 3 (All) shows the overall numbers of context snippets we extracted for each name, the fourth column (Max) shows the maximum numbers of snippets for one entity with the name, i.e., the numbers of snippets used for the majority class baseline. For example, we extracted a total of 3968 snippets for the name *Michael Jackson* (All). 3899 of these snippets (Max) belong to the entity *Michael Jackson, the American singer*. The remaining 69 snippets belong to the other 11 entities with the same name. We have some extreme cases like this one in our dataset, with one entity having many more snippets than the others because it is a very famous person. In other cases the number of snippets is more evenly distributed over all entities with the same name. The last column in Table 3 (Avg) lists the average numbers of snippets that were extracted for each entity.

We use the extracted link information as gold labels to disambiguate the person. For example, if we find the link *Michael_Jackson_(English_singer)* in an article, we know that the extracted name and context around it refers to *Michael Jackson, the English singer*. By using this link information we do not have to annotate a dataset manually. Similar datasets have been created before (Nothman et al., 2008; Nothman et al., 2013; Hahm et al., 2014).

5 Experiments and Discussion

For obtaining topic information we use the MALLET toolkit (McCallum, 2002) which contains several machine learning applications, for example, document classification, clustering, and information extraction. For topic modeling it provides implementations of Latent Dirichlet Allocation (LDA), Pachinko Allocation, and Hierarchical LDA. We use the ParallelTopicModel class which is a simple parallel threaded implementation of LDA based on Newman et al. (2009) and Yao et al. (2009). We trained different topic models using the corpora and topic parameters we described in Section 4.1.

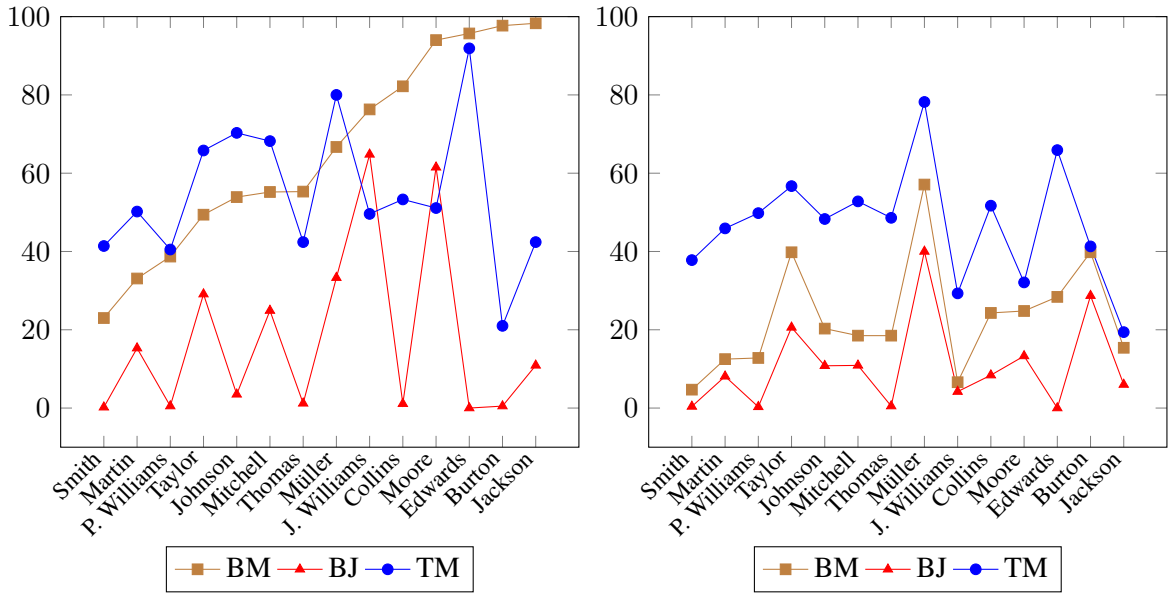
For every entity in our test set we created a corpus consisting of the properties of this entity (e.g., for *Michael Jackson, the American singer* we created a corpus with the properties *American* and *singer*) as described in Section 4.2. We then apply our trained topic models to obtain topic information from these corpora, as well as from the test snippets of our silver standard corpus. The vector representing the topic information from each new entity is then compared to each vector consisting of topic information from the properties corpora as we showed in Section 3.2.

We use two baselines. The **Baseline Majority (BM)** simply predicts the majority class. The **Baseline Jaccard (BJ)** uses the Jaccard index to compare the similarity between the context of the new unknown entity and the corpora we created. The Jaccard index is defined as the size of the intersection divided by the size of the union of two sample sets A and B:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

5.1 Results and Discussion

We conducted a total of 546 experiments with different parameters (context size, number of topics, corpus used to train the topic model). Due to limited space we show results only for one setting of parameters. We chose parameters which had an average performance in the experiments to give an estimate of how the approach works. Using other parameters can improve the results.



(a) micro-averaged F_1

(b) macro-averaged F_1

Figure 2: F_1 score of BM, BJ, and TM with parameters: context size=paragraphs, topics=1000, corpus=W_{all}.

Figure 2 shows the F_1 score for all experiments with the following parameters: context size = paragraphs, topics = 1000, corpus = W_{all}. The names are ordered by the micro-averaged F_1 score of the Baseline Majority (BM). In subsequent graphs we use the same ordering for better comparison. We present both the micro-averaged F_1 score (2a) and the macro-averaged F_1 score (2b), which give quite different results. Micro-averaged F_1 score weights each classification decision equally, i.e., it favors large classes, while macro-averaged F_1 score weights each class equally, i.e., it shows the effectiveness of small classes better (Manning et al., 2008).

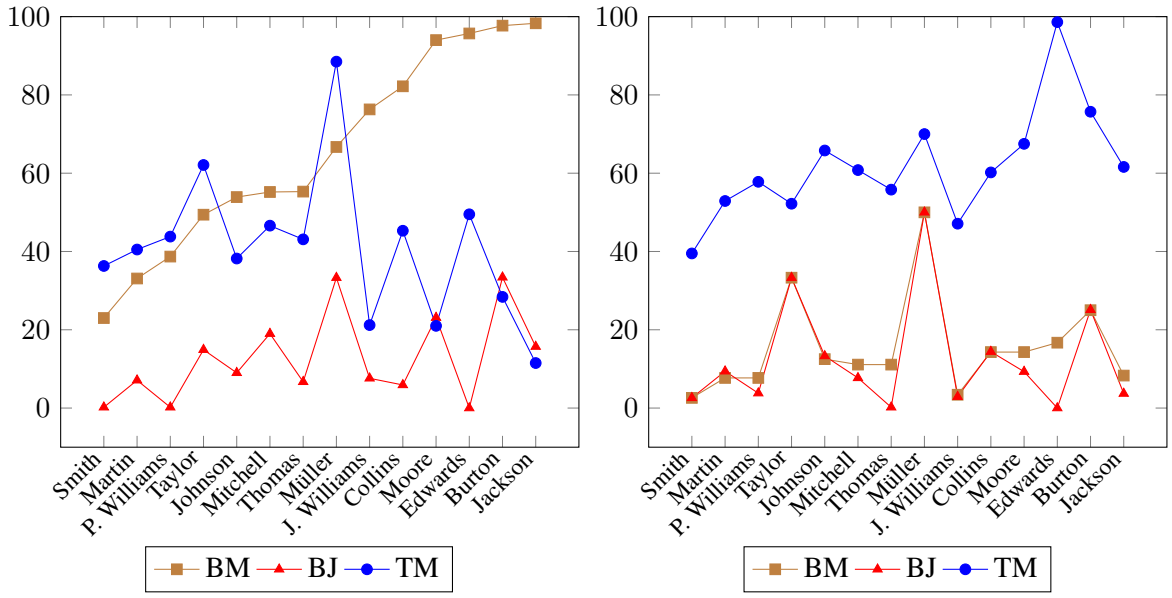
Figure 2a shows that when using micro-averaging, in some cases the majority baseline is better than our topic model approach. This is the case when there is one famous name-bearer who has many more examples than the other persons with the same name in the test set, e.g., *Edwards, Burton, Jackson*, which results in one class that is much larger than the others. For example, our test set contains 3968 snippets for *Michael Jackson*, with 3899 of the snippets belonging to the majority class (*Michael Jackson, the American singer*), which results in a micro-averaged F_1 score of 98.26% for BM.

The macro-averaged results in Figure 2b give a better sense of how well our approach works on smaller classes. It can be seen that our approach performs better than both baselines in all cases. Since we are interested in the not well-known entities, which have smaller classes in our test set, the macro-averaged results show that our approach works well for these entities.

The Baseline Jaccard (BJ) stays even below the Baseline Majority in most cases. It generally does not work well for persons that are rather unknown. One reason is that the contexts extracted for these entities are often smaller and do not provide as much information as is needed for this baseline approach. The main advantage of our TM approach is that it works well for unknown entities which usually have little or no available training material, and for entities which have little surrounding context.

In Figure 3 we give more insight into the macro-averaged results of Figure 2b and show the macro-averaged precision (3a) and macro-averaged recall (3b) with the same parameters as before. Precision-wise, the Baseline Majority does better in cases with large classes. This is expected because if most examples in the test set belong to the majority class, the number of false positives is small, which leads to higher precision.

Figure 3b shows that the recall for our approach outperforms both baselines by far in all cases. In one case (*Edwards*) we even achieve a recall of 98.6%. The reason for this is that the baseline approaches do not work well for the not well-known people with small classes in our test set. BM does not work well



(a) macro-averaged precision

(b) macro-averaged recall

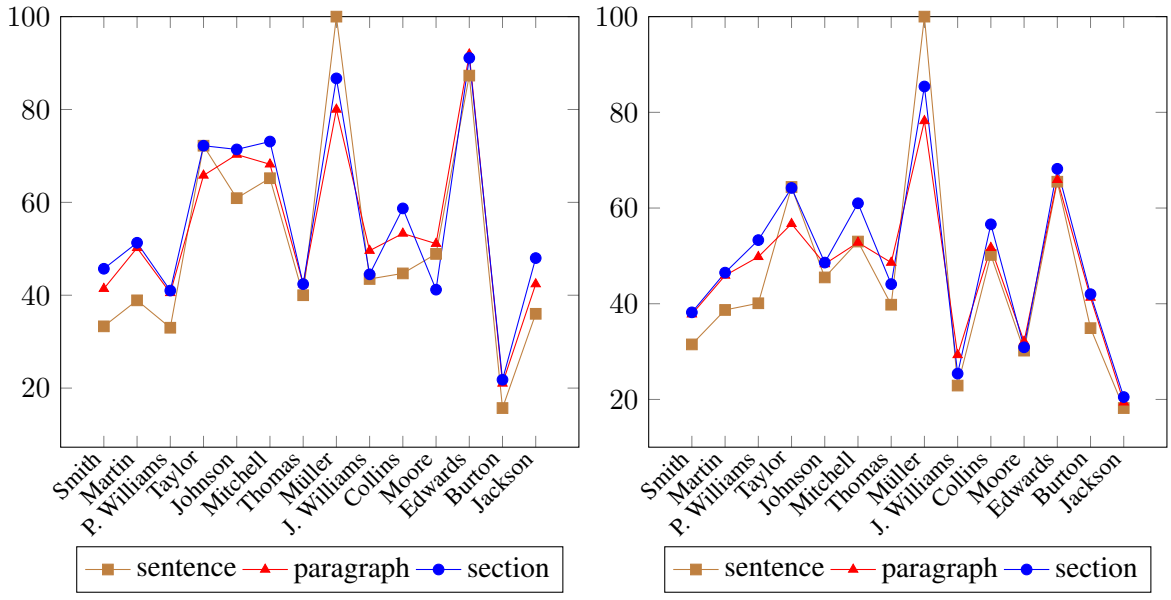
Figure 3: Precision and Recall of BM, BJ, and TM with parameters: context size=paragraphs, topics=1000, corpus=W$_{all}$.

because it incorrectly tags the small classes with the majority class and BJ does not work well because the information in the context snippets is not sufficient for this approach. Measuring the similarity between words, n-grams, or similar approaches suffers from sparsity problems. Our TM approach makes better use of smaller context snippets because it extracts and uses relevant information (i.e., the topic information) about a person more effectively. This leads to higher recall results in most cases. In some cases with different parameters, we even achieve a recall of 100% for some entities (not shown in Figure 3). Obtaining a high recall is important in applications that aim for high recall results (e.g., in search engines or QA systems; the system returns more correct results instead of returning results of similar persons which are more well-known).

We have investigated how the context size, number of topics, and the corpus used for training the topic model influences the results of our TM approach. Figure 4 shows results for the different context sizes (sentence, paragraph, section), when using the same parameters for the topic model as in the previous figures (topics = 1000, corpus = W$_{all}$). In most cases, taking more context gives better results (i.e., paragraphs are better than sentences and sections are better than paragraphs). In some cases, a larger context introduces more noise which leads to worse results than when taking a smaller context (e.g., *Moore* in Figure 4a). Generally, taking a smaller context does not worsen the results a lot and in some cases gives nearly the same results as when taking a larger context. This shows that our TM approach works well if there is only little context available for a person. This is important because often the context for a person is rather limited, for example, when a document is mainly about another person there might be only one relevant sentence as context, or when the document is very short.

When we investigated the usefulness of the number of topics, we found that taking a small number of topics (like 100) works best. Using a higher number results in more fine-grained topics, which makes finding the most similar corpus harder. An advantage of using fewer topics is that it takes less time for the topic model to determine the topics of a new document. In a real application it is important to reduce the time needed as much as possible because a user of the application does not want to wait for several minutes or even hours for results.

There is no clear preference for which corpus is the most useful one for the task and it also depends on the context size. Overall, using the entire Wikipedia seems to be the least helpful and it is better to use a more specific collection which produces better topics for the task. We also found that when using different collections, the results can change a lot. While the results for using different parts of the



(a) micro-averaged F_1

(b) macro-averaged F_1

Figure 4: Different context sizes of TM with parameters: topics=1000, corpus=W_{all}.

Wikipedia collection all followed the same trend, the results for using the Gigaword corpus were very different and often worse than Wikipedia's results.

6 Conclusion and Future Work

We presented work on a new system for Named Entity Disambiguation which does not need specific textual training data to disambiguate unknown persons. Instead, it uses some aggregate of the textual material about different people that share the same properties (e.g., nationalities and professions). The system learns which properties people can have, then uses these properties to disambiguate new proper names in documents.

To learn properties we extracted documents about people sharing the same properties from a collection, then applied topic models on the data to obtain topic information. The topic models were trained on different collections and with different numbers of topics to investigate which parameters are most useful. To disambiguate a new unknown person in a text document, we obtained topic information from the context of the person, then compared this topic information with the information from the extracted material to find out which properties are closest to the person.

In our pilot implementation of the approach, we conducted 546 experiments on 14 ambiguous names using different parameters (context size, number of topics, corpus used for training the topic model). For evaluating our system we created a silver standard corpus that does not need any manual annotation, and compared our approach to two baselines. We showed that our system outperforms the baselines in many cases, especially for (i) entities for which not much training material is available, and (ii) entities with little surrounding context. We also showed that with our approach we can achieve high recall results, which is important for many applications, e.g., search engines and QA systems.

The approach can be expanded with more properties in the future. For example, it could include properties like age, place of birth, and affiliation (university, company etc.) and properties about what people are doing besides their profession (e.g., working in organizations or for charity).

Acknowledgements

The work reported in this paper was supported by a Nuance Foundation Grant.

References

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 352–361, Sofia, Bulgaria, August. Association for Computational Linguistics.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 370–379, Baltimore, Maryland, June. Association for Computational Linguistics.

Indrajit Bhattacharya and Lise Getoor. 2006. A latent dirichlet model for unsupervised entity resolution. In *SIAM Conference on Data Mining (SDM)*, April. Winner of the Best Paper Award.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March.

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2006. Extracting key phrases to disambiguate personal name queries in web search. In *Proceedings of the Workshop on How Can Computational Linguistics Improve Information Retrieval?*, CLIIR '06, pages 17–24, Stroudsburg, PA, USA. Association for Computational Linguistics.

Razvan Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceesings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, pages 9–16, Trento, Italy.

Ying Chen and James Martin. 2007. Towards robust unsupervised personal name disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 190–198, Prague, Czech Republic, June. Association for Computational Linguistics.

Aaron M. Cohen. 2005. Unsupervised gene/protein named entity normalization using automatically extracted dictionaries. In *Proceedings of the ACL-ISMB Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*, ISMB '05, pages 17–24, Stroudsburg, PA, USA. Association for Computational Linguistics.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic, June. Association for Computational Linguistics.

David Graff and Christopher Cieri. 2003. English Gigaword LDC2003T05. Web Download. Philadelphia: Linguistic Data Consortium.

Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named entity recognition in query. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 267–274, New York, NY, USA. ACM.

Younggyun Hahm, Jungyeul Park, Kyungtae Lim, Youngsik Kim, Dosam Hwang, and Key-Sun Choi. 2014. Named entity corpus construction using wikipedia and dbpedia ontology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2565–2569, Reykjavik, Iceland, May.

Xianpei Han and Le Sun. 2012. An entity-topic model for entity linking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 105–115, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: A graph-based method. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 765–774, New York, NY, USA. ACM.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 782–792, Stroudsburg, PA, USA. Association for Computational Linguistics.

Saurabh S. Kataria, Krishnan S. Kumar, Rajeev R. Rastogi, Prithviraj Sen, and Srinivasan H. Sengamedu. 2011. Entity disambiguation with hierarchical topic models. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 1037–1045, New York, NY, USA. ACM.

Yang Li, Chi Wang, Fangqiu Han, Jiawei Han, Dan Roth, and Xifeng Yan. 2013. Mining evidences for named entity disambiguation. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 1070–1078, New York, NY, USA. ACM.

Bradley Malin. 2005. Unsupervised name disambiguation via social network similarity. In *In Proceedings of the SIAM Workshop on Link Analysis, Counterterrorism, and Security*, pages 93–102.

Gideon S. Mann and David Yarowsky. 2003. Unsupervised personal name disambiguation. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 33–40, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 233–242, New York, NY, USA. ACM.

David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2009. Distributed algorithms for topic models. *J. Mach. Learn. Res.*, 10:1801–1828, December.

Joel Nothman, James R. Curran, and Tara Murphy. 2008. Transforming wikipedia into named entity training data. In *In Proceedings of the Australasian Language Technology Association Workshop 2008*, pages 124–132.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence*, 194(0):151–175. Artificial Intelligence, Wikipedia and Semi-Structured Resources.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1524–1534, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pages 487–494, Arlington, Virginia, United States. AUAI Press.

Lus Sarmento, Alexander Kehlenbeck, Eugnio C. Oliveira, and Lyle H. Ungar. 2009. An approach to web-scale named-entity disambiguation. In Petra Perner, editor, *MLDM*, volume 5632 of *Lecture Notes in Computer Science*, pages 689–703. Springer.

Prithviraj Sen. 2012. Collective context-aware topic models for entity disambiguation. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 729–738, New York, NY, USA. ACM.

Liangcai Shu, Bo Long, and Weiyi Meng. 2009. A latent topic model for complete entity resolution. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, ICDE '09, pages 880–891, Washington, DC, USA. IEEE Computer Society.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 937–946, New York, NY, USA. ACM.

Wei Zhang, Yan Chuan Sim, Jian Su, and Chew Lim Tan. 2011. Entity linking with effective acronym expansion, instance selection and topic modeling. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 1909–1914. AAAI Press.

Building RDF Content for Data-to-Text Generation

Laura Perez-Beltrachini
CNRS/LORIA
Nancy (France)
laura.perez@loria.fr

Rania Mohamed Sayed
Université de Lorraine
Nancy (France)
rania.mohamed.sayed@gmail.com

Claire Gardent
CNRS/LORIA
Nancy (France)
claire.gardent@loria.fr

Abstract

In Natural Language Generation (NLG), one important limitation is the lack of common benchmarks on which to train, evaluate and compare data-to-text generators. In this paper, we make one step in that direction and introduce a method for automatically creating an arbitrary large repertoire of data units that could serve as input for generation. Using both automated metrics and a human evaluation, we show that the data units produced by our method are both diverse and coherent.

1 Introduction

In Natural Language Generation, one important limitation is the lack of common benchmarks on which to train, evaluate and compare data-to-text generators. In this paper, we make one step in that direction and introduce a method to automatically create an arbitrary large repertoire of data units which could serve as input for data-to-text generation. We focus on generation from RDFS data where the communicative goal is to describe entities of various categories (e.g., astronauts or monuments).

RDF data consists of (subject property object) triples (e.g., `(Alan_Bean occupation Test_pilot)`) – as illustrated in Figure 1, RDF data can be represented by a graph in which edges are labelled with properties and vertices with subject and object resources. To construct a corpus of RDF data units which could serve as input for NLG, we introduce a content selection method which, given some DBPedia entity, retrieves DBPedia subgraphs that encode relevant and coherent knowledge about that entity.

Our approach differs from previous work on content selection in that it leverages the categorial information provided by large scale knowledge bases about entities of a given ontological type. Based on this ontological knowledge, we learn two types of category-specific bigram models: one model (S-Model) for bigrams occurring in sibling triples (triples with a share subject) and one model (C-Model) for bigrams occurring in chained triples (the object of one triple is the subject of the other). The intuition is that these two models capture different types of coherence, namely, topic-based coherence for the S-Model and discourse-based coherence for the C-Model.

Using these bigram models of RDF properties, we formulate the content selection task as an Integer Linear Programming problem and select for a given entity of category C, subgraphs with maximal probability that is, subgraphs which contain properties that are true of that entity, that are typical of that category and that support the generation of a coherent text.

We evaluate the impact of our n-gram models on content selection (how well do they help support the selection of a coherent and diverse set of data units?) using quantitative metrics, a human evaluation and a qualitative analysis.

2 Related Work

Our approach has similarity with approaches on entity summarisation, content planning from DBpedia data and ILP (Integer Linear Programming) approaches for content planning. There is also a vast literature on using ILP for natural language processing.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1493–1502, Osaka, Japan, December 11-17 2016.

Entity Summarisation (Cheng et al., 2015) presents an approach which focuses on a task very similar to ours, namely the task of selecting, for a given entity e, a subgraph of the knowledge graph whose root is e. The goal is to generate entity summaries that is, sets of facts which adequately summarise a given entity. The method used extends a standard random surfer model navigating the knowledge graph based on metrics indicating (i) the informativeness of a fact and (ii) the relatedness between two facts. In this way, the selected subgraphs are both coherent (solutions which maximise relatedness are preferred) and informative (facts that helps distinguishing the entity to be summarised from others are preferred).

We depart from (Cheng et al., 2015) both in terms of goals and of methods.

In terms of goals, while (Cheng et al., 2015) aim to produce entity summaries, our goal is to produce a large set of content units that are varied both in terms of content and in terms of structure. In particular, one important difference is that we produce trees of varying shapes and depths while the graphs produced by (Cheng et al., 2015) are restricted to trees of depth one i.e., set of DBpedia triples whose subject is the entity to be described. As discussed in Section 5.1, this allows us to produce knowledge trees which, because they vary in shape, will give rise to different linguistic structures and will therefore better support the creation of a linguistically varied benchmark for Natural Language Generation.

Our approach also departs from (Cheng et al., 2015)'s in that the methods used are very different. While we use Integer Linear Programming and language models to select DBpedia subgraphs that are both discourse- and topic-coherent, (Cheng et al., 2015) use a random surfer model, pointwise mutual information and probabilistic estimates to measure relatedness and informativeness. Generally, the two methods are complementary using different resources, algorithms and metrics thereby opening interesting possibilities for combination. It would be interesting for instance, to investigate how modifying our ILP formulation to integrate the relatedness metrics used by (Cheng et al., 2015) would impact results.

Content Planning (Biran and McKeown, 2015) describes a discourse planning approach applied to the generation of comparison stories from DBpedia data. Given two DBpedia entity e_1 and e_2, they first select all DBpedia triples whose subject is either e_1 or e_2. Based on the shape of the triples (shared entities or predicates) and on the property they include, they then enrich this set of DBpedia triples with discourse relations. For instance, if two triples share the same predicate and object, an expansion relation is added between the two triples (e.g., "John has a ball. Mary also has a ball"). Discourse planning then consists in finding a path through the resulting multigraphs of potential relations between DBpedia triples using an bigram model over discourse relations. Good discourse plans are those which maximise the probability of a sequence of discourse relations. In this way, the proposed approach determines both the order of the events and the discourse relation holding between them.

(Lampouras and Androutsopoulos, 2013) present an Integer Linear Programming model of content selection, lexicalisation and aggregation for generating text from OWL ontologies. The objective function used in their ILP model maximises the total importance of selected facts and minimizes the number of distinct elements mentioned in each sentence thereby favouring aggregated sentences i.e., sentences where repeated elements are avoided through e.g., ellipsis or coordination.

(Bouayad-Agha et al., 2011) introduces an ontology driven content selection procedure in which a base domain ontology is used to infer new facts. For instance, given the numerical scores of two teams playing in the same game, a result event will be inferred between the winner and the loser and a causal relation will be inferred between the number of goals of a given team and this result event. Content selection proceeds in three steps. First, a set of hand written rules is used to select a subset of the knowledge base. Second, relevance scores learned from a parallel data/text corpus are used to select the most relevant individual and relation instances. Third, hand-written templates are used to determine the content to be included in the generated text.

Our approach differs from these proposals in that it focuses on content selection from typed RDF data. Using bigram models whose basic units are DBpedia triples, we maximise global coherence by favouring content where DBpedia properties that often co-occur are selected together. In contrast, (Lampouras and Androutsopoulos, 2013) assumes that the relevance scores are given. Moreover, while they focus on selecting content that leads to maximally aggregated content, we focus on selecting content that is discourse coherent. Like us, (Biran and McKeown, 2015) focus on DBpedia data and use bigram

models. However their approach investigate discourse planning not content selection and relatedly, the basic units of their bigram models are discourse relations rather than triples. Our approach also differs from (Barzilay and Lapata, 2005) in that it is unsupervised and does not require an aligned data-text corpus.

Finally, the work presented here is closely related to a simpler proposal we introduced in (Mohammed et al., 2016). It differs from it in that it defines the notions of chain, sibling and mixed models for n-grams of DBpedia properties; relate them to the notion of topic- and discourse-coherence; and provide a comparative evaluation of their impact on content selection.

Integer Linear Programming and NLP. Finally, there has been much work in recent years on using ILP for natural language processing. In particular, (Kuznetsova et al., 2012) proposes an ILP formulation for the generation of natural image descriptions from visual and text data and (Filippova and Strube, 2008) uses ILP to model sentence compression. The ILP formulation of our content selection method is most similar to that proposed for sentence compression in (Filippova and Strube, 2008). One important difference though is both the application (content selection rather than sentence compression) and the way in which relevance is computed. While (Filippova and Strube, 2008) uses weights derived from a treebank to determine the relative importance of an edge, we use bigram models over DBpedia properties to estimate the relative importance of DBpedia triples.

3 Task and Method

Given an entity e of category C and its associated DBpedia *entity graph* G_e, our task is to select a (target) subgraph T_e of G_e such that:

- T_e is *relevant*: the DBpedia properties contained in T_e are commonly (directly or indirectly) associated with entities of type C
- T_e maximises *topic-based coherence*: DBpedia triples that often co-occur in type C are selected together
- T_e supports *discourse coherence*: the set of DBpedia triples contained in T_e capture a sequence of entity-based transitions which supports the generation of discourse coherent texts i.e., texts such that the propositions they contain are related through shared entities.

To implement these constraints, we first build bigram models of properties for DBpedia categories. We then use these models and Integer Linear Programming to retrieve from DBpedia, entity graphs with maximal probability.

3.1 Building Bigram Models for DBpedia Categories

For each DBpedia categories (e.g., Astronaut or University), we learn two bigram models `S` and `C`, each designed to capture different aspects of content coherence.

The `S`(ibling)-model consists of bigrams that are sibling properties in DBpedia. Two properties are sibling of each other if they occur in triples sharing the same subject. Thus, the DBpedia graph shown in Figure 1 contains 5 `S`-bigrams namely, `birthPlace-mission`, `birthDate-mission`, `birthDate-birthPlace`, `country-leader` and `crewMember-operator`[1]. The `S`-model aims to capture local coherence i.e., topic-based associations between DBpedia properties.

In contrast to the `S`-model, the `C`(hain)-model aims to capture *discourse coherence* i.e., associations between DBpedia triples that involve a shared entity other than the entity being described. It consists of DBpedia triples that are related by a shared entity. The DBpedia graph shown in Figure 1 contains 4 `C`-bigrams namely, `mission-crewMember`, `mission-operator`, `birthPlace-country` and `birthPlace-leader`.

[1]Sibling bigrams (`S`-bigrams) are normalised using alphabetical order. Thus, given the two triples `(A mission B)`, `(A nationality C)`, the associated `S`-bigram is `mission-nationality` – not `nationality-mission`.

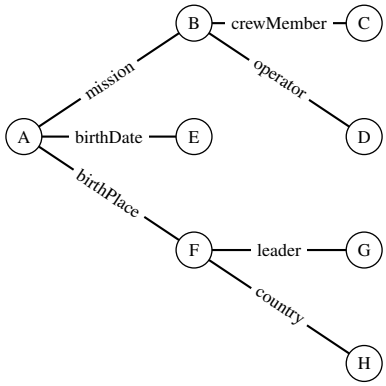


Figure 1: Example DBpedia Graph (To save space subject and object names have been replaced by capital letters).

3.2 Extracting DBpedia Subgraphs

We use the two bigram models just described and an interpolation (M-Model) of these two models to select from an entity graph subtrees whose coherence is either topic-based (S-Model), discourse-based (C-Model) or both (M-Model).

The ILP formulation of the task is as follows.

Representing triples. Given an entity graph G_e for the DBpedia entity e of category C (e.g. Astronaut), for each triple $t = (s, p, o)$ in G_e, we introduce a binary variable $x^p_{s,o}$ such that:

$$x_t = x^p_{s,o} = \begin{cases} 1 & \text{if the triple is preserved} \\ 0 & \text{otherwise} \end{cases}$$

Because we use bigrams to capture local and discourse coherence (properties that often co-occur together), we also have variables y_{t_1,t_2} for bigrams of triples such that:

$$y_{t_1,t_2} = \begin{cases} 1 & \text{if the pair of triples is preserved} \\ 0 & \text{otherwise} \end{cases}$$

For the S-Model, these binary variables capture pairs of triples which share the same subject. That is, for each bigram of triples $t_1 = (s1, p1, o1)$ and $t2 = (s2, p2, o2)$ in G_e such that $s1 = s2$, we introduce a binary variable y_{t_1,t_2}.

Similarly, for the C-Model, we introduce a binary variable y_{t_1,t_2} for each pair of triples such that the object of one is the subject of the other. That is, $(t_1,\ t_2)$ is a C-bigram iff $t_1 = (s1, p1, o1)$, $t2 = (s2, p2, o2)$ and $o1 = s2$.

Maximising Relevance and Coherence. To maximise relevance and coherence, we seek to find a subtree of the input graph G_e which maximises the S-bigram probability (S-Model), the C-bigram probability (C-Model) or an interpolation of both (M-Model).

For the S- and the C-Model, we maximise the following objective function over the set of all bigrams Y from the set of triples X:

$$S(X) = \ \sum_Y y_{t_i,t_j}\ .\ P(t_i, t_j) \tag{1}$$

where y_{t_i,t_j} is the ILP binary variable for (t_i, t_j) and $P(t_i, t_j)$ is the bigram probability for category C. Let B_c be the set of property bigrams occurring in the entity graphs of all DBPedia entities of category C. Let *count(b,C)* be the number of time b occurs in B_c, then the bigram probability $P(b)$ of b for category C is defined as follows:

$$P(b) = \frac{\textit{count(b,C)}}{\sum_{b_i \in B_C} \textit{count(b}_i\textit{,C)}} \tag{2}$$

For the S-Model, only S-bigrams are included in the counts while for the C-Model, only C-bigram counts. For the M-Model, the objective function to be maximised is defined as:

$$S(X) = \quad \gamma * \sum_Y y_{t_i,t_j} \ . \ P(t_i,t_j) + (1-\gamma) \sum_Z z_{t_k,t_l} \ . \ P(t_k,t_l) \tag{3}$$

where y_{t_i,t_j} is restricted to S-bigrams and z_{t_k,t_l} to C-bigrams and γ is a parameter to balance the contribution of local- or discourse- probabilities.

Consistency Constraints. We ensure consistency between the triple and the bigram variables so that if a bigram is selected then so are the corresponding triples (eq. 5). Conversely, eq. 6 requires that if two triples t_i and t_j are selected then so is the corresponding bigram y_{t_i,t_j}[2]

$$\forall i,j \ (y_{i,j} \leq x_i \textit{ and } y_{i,j} \leq x_j) \tag{5}$$

$$y_{i,j} + (1-x_i) + (1-x_j) \geq 1 \tag{6}$$

Ensuring Discourse Coherence (Tree Shape). Solutions are constrained to be trees by requiring that each object has at most one subject (eq. 7) and all triples are connected (eq. 8).

$$\forall o \in X, \sum_{s,p} x^p_{s,o} \quad \leq \quad 1 \tag{7}$$

$$\forall o \in X, \sum_{s,p} x^p_{s,o} \quad - \quad \frac{1}{\mid X \mid} \quad \sum_{u,p} x^p_{o,u} \geq \quad 0 \tag{8}$$

where X is the set of triples that occur in the solution (except the root node). This constraint makes sure that if o has a child then it also has a head.

Restricting the size of the resulting tree. Solutions are constrained to contain α triples:

$$\sum_x x^p_{s,o} \quad = \quad \alpha \tag{9}$$

4 Experimental Setup

We test our approach on 3 DBPedia categories chosen to be diverse in that they represent different levels of animacy namely, *Monument, University* and *Astronaut*. These provide with different sets of DBPedia properties for the evaluation.

Building bigram models of DBpedia properties. To build the bigram models, we extract from DBpedia the graphs associated with all entities of those categories up to depth 5 and separately extract C-bigrams and S-bigrams. Table 1 shows some statistics for these graphs. We build the C-Model and the S-Model using the SRILM toolkit.

Building Entity Graphs. For each of the three categories, we take 5 randomly chosen entities and extract their DBpedia graph up to depth 2[3]. Table 2 shows some statistics for these entities.

[2]Note that these constraints do not require that every selected triple be part of at least one bigram containing that triple. We have only recently added this constraint (eq. 4) to further improve topic coherence.

$$\forall i,j \ s.t. \ i = t \ or \ j = t, \ x_t <= \sum y_{i,j} \tag{4}$$

[3]It would of course be possible to extract deeper graphs using but this would required building higher order n-gram models and data sparsity might degrade results. Here, we leave this point open for further research.

Category	Entities	Triples	Properties
Astronaut	110	1664033	4167
Monument	500	818145	6521
University	500	969541	7441

Table 1: Category Graphs

Entity	A		M		U	
	d1	d2	d1	d2	d1	d2
e1	14	24	13	18	6	20
e2	21	32	20	21	13	21
e3	16	28	7	14	6	10
e4	12	24	6	14	9	16
e5	15	22	4	11	27	34

Table 2: Size in number of triples for each Entity Graph for each category (A = Astronaut, M = Monument, U = University) and depth (d1 = Depth 1 and d2 = Depth 2).

Selecting Data Units. To ensure that our content selection procedure produces varied data with respect to both form and content, we run the ILP program on entities belonging to three DBPedia categories (Astronaut, University, Monument) and using each of the bigram models (S-Model and C-Model) and their combination (M-Model). Using different DBPedia categories ensures that the selected data units vary in terms of RDF resources (entities and properties). Using the different bigram models permit producing data units exhibiting different levels of topic- and discourse-coherence. The intuition is that the S-Model will yield data units where topic-based coherence dominates, the C-Model discourse data units emphasizing transition-based, discourse coherence and M-Model data units which display a balance between topic-based and discourse coherence. We set γ to 0.4 (eq.3), after running several experiments we observed that this weight balanced the solutions favouring C-bigrams which in general have smaller probability values than S-bigrams.

We run the ILP with α (the number of triples occurring in the solution) ranging from 3 to 10 and input entity graphs with depth 1 and 2.

5 Evaluation

Our goal is to generate a large corpus of data units which could be used as a basis to build a data-to-text benchmark for training, testing and comparing data-to-text generators. In the evaluation, we therefore focus on assessing (i) the diversity and (ii) the coherence of the selected data units.

5.1 Diversity

As discussed in the preceding sections, the three ILP models generate solutions with slightly different properties. This can be viewed as a controlled *sampling* procedure. Using the different ILP models, we can sample subgraphs of the same entity graph which have the same size but are markedly distinct.

To better assess the degree to which the solutions generated by our models differ from each other, we compute two metrics designed to capture both the overlap between the solutions produced and the number of distinct shapes found.

Number of Distinct Solution Shapes. The shape of the trees extracted from an entity graph will impact the possible syntactic structure of the corresponding text. For instance, trees such as (1a) where the subject entity is shared by two triples, will naturally induce the use of an adjective modifier (1b). In contrast, trees such as (1d) where the object entity of a triple is the subject of another triple naturally suggests the use of a participial or a relative clause (1d-e).

(1) a. `(Alan_Bean occupation Test_pilot) (Alan_Bean nationality USA)`
b. *Alan Bean was an **American** test pilot*
c. `(Alan_Bean mission Apollo_12) (Apollo_12 operator NASA)`
d. *Alan Bean flew on the Apollo 12 mission **operated by** NASA*
e. *Alan Bean flew on the Apollo 12 mission **which was operated by** NASA*

More generally, to ensure a good linguistic coverage, a benchmark should contain a large number of distinct input shapes. We approximate the shape of an input unit U_e describing the entity e by using a classification which combines (i) the number D_e of triples $t \in U_e$ whose subject is e, (ii) the number

O_e of subject entities $e' \in U_e$ other than e and (iii) the number I_e of triples $t \in U_e$ whose subject is not e. That is, an input shape is defined as a triple (D_e, O_e, I_e) indicating the number D_e of triples directly connected to the entity e being described, the number O_e of subject entities other than this entity and the number I_e of triples indirectly connected to e.

When considering the 10 best solutions produced by the M-Model on the entity graphs of the 15 entities mentioned above, the total number of distinct input shapes is 75 with a minimum, a maximum and an average number of instances per input shape of 1, 24 and 5.31 respectively.

Overlap. To assess the degree to which the solutions produced by our approach differ from each other we compute the average overlap between solutions for the same configuration both within and across models. A configuration is defined by the number of triples appearing (3 to 10) in the solution, the depth of the input graph (1 or 2) and the model used (S-Model, C-Model or M-Model). For each configuration, the average overlap is defined as $\frac{\sum_{i,j} O(s_i,s_j)}{N}$ where s_i, s_j are solutions produced in that configuration, N is the number of distinct pairs produced by that configuration and the overlap, $O(s_i, s_j)$, between two solutions is the ratio between the number of property they share and the number of triples contained in (s_i, s_j)[4].

Table 6 (left) shows the results for the three models given 16 configurations and 3 DBpedia categories. The 16 configurations correspond to solutions of size 3 to 10 on graphs of depth 1 and 2.

With an average overlap within and across models ranging from 0.18 to 0.31, these results indicate a good level of diversity whereby the C-Model and the M-Model are found to be slightly better at providing solutions with small overlap (avg. 0.24 and 0.26 respectively) than the S-Model (avg. 0.31).

Similarly, Table 6 (right) shows that the overlap across models is relatively low (Min: 0.18, Max: 0.24) indicating that solutions produced for the same configuration by different models are usually markedly distinct (no more than a quarter or a small half of the triples are shared between any two solutions).

In sum, by modifying the ILP parameters to select various numbers of triples, we can generate solutions of different sizes whilst the 3 ILP models permit producing solutions with relatively small overlap both within and across models. That is, our content selection method can be used to automatically create a graduated benchmark for natural language generation in which the inputs are of increasing size and exhibit a good level of semantic variability. Using crowdsourcing, these RDF input could be associated with appropriate verbalisations whereby annotators could be gradually trained to verbalise the data by exposing them to input of gradually increasing length.

5.2 Coherence

Because they are retrieved from DBpedia, the data units selected by our approach are semantically coherent overall. In particular, the triples that are directly connected to the entity being described are all relevant. However when selecting a subtree of the input entity graph, the coherence between siblings and between chained triples may decrease. For instance, given the entity graph shown in Figure 1, subgraph (2a) is more topically-coherent that subgraph (2b). Similarly, subgraph (2c) is more discourse-coherent that subgraph (2d).

(2) a. `(A birthDate E) (A birthPlace F)`

b. `(A birthDate E) (A mission B)`

c. `(A mission B) (B crewMember C)`

d. `(A birthPlace F) (F leader G)`

We compare our approach with a baseline where a subtree of DBpedia triples is randomly selected from the entity graph using an automatic metric and a human evaluation.

[4]Since the S-Model is designed to favour sibling or topically related triples but not triples related by a shared entity, we disregard in all evaluation counts the solutions of depth 2 produced by the S-Model. Conversely, we exclude from the evaluation counts the solutions of depth 1 produced by the C-Model.

		Min	Max	Avg	# Solns
d1	BL	0	2	0.44	400
	S-Model	0	1.75	**0.31**	271
d2	BL	0	2	0.73	218
	C-Model	0	1.94	0.59	382
	M-Model	0	1.25	**0.43**	152
	S-Model	0.07	1.29	0.54	123

Table 3: Averaged number of irrelevant property descriptions for solutions of depth 1 (d1) and 2 (d2) on the Astronaut category.

	BL	S-Model	C-Model	M-Model
C (3)	6	18	1	2
M (2)	15	11	20	13
L (1)	10	2	9	15
Avg	1.87	2.52	2.27	2.43

Table 4: Coherence scores for the different models (C = Coherent, M = Medium, L = Less coherent).

Number of Irrelevant Triples. We quantify the number of irrelevant triples contained in solutions produced by the different models by first, manually labelling each property present in the Astronaut graph as relevant or irrelevant and second, counting the number of irrelevant properties occurring in the solutions produced by the baseline and the 3 ILP models. In practice, irrelevant properties are properties that are indirectly related to the entity being described. For instance, the `leader` property shown in Figure 1 is much less relevant when describing an astronaut than the `crewMember` or the `mission` property.

Table 3 shows the results. The baseline consistently shows a higher number of irrelevant properties indicating that our method is efficient in filtering them out. For depth 2, the M-Model shows the best results. The lower score (higher number of irrelevant properties) of the S-Model shows that selecting triples based on sibling bigrams only, fails to eliminate indirectly related triples which are irrelevant to the entity being described. Sibling properties are selected for entities related to the entity being described which are not relevant in context. For instance, in Figure 1, the S-bigram `leader-country` has little relevance when describing the target entity A. For the C-Model, examination of the distribution per solution size shows that the number of irrelevant properties increases with the solution size. This is explained by the fact that as the number of triples in the solution increases, the number of C-bigrams to be selected increases leading to the selection of bigrams (e.g., `birthPlace-leader`) with lower probability.

Human Evaluation. Using the Crowdflower platform, we ran a human evaluation to compare the coherence of the solutions produced by the different models. The annotators were shown two data units of the same size but produced by different content selection models and were asked to rate the coherence of each dataset as coherent (3), medium (2) or less coherent (1).

To assess the impact of the S-Model on topic-based coherence, we compared the S-Model with the baseline. The evaluation was carried out on 23 pairs of data units ranging from size 3 to 10 and describing entities of all three categories. We collected 10 judgements for each pair (230 judgements total). Similarly, we compare the M-Model and the C-Model to assess the extent to which the M-Model is successful in combining discourse- and topic-based coherence. Table 4 summarises the results. For all models, the scores are much higher than for the baseline indicating that the bigrams we learn successfully model coherence. The S-Model has the highest coherence, which is unsurprising as only graphs of depth 1 are considered and properties that are directly related to the entity being described are by definition relevant. The C-Model and M-Model also achieve relatively high scores thereby confirming the good results obtained with the other metrics (number of irrelevant properties)[5].

Qualitative Analysis. Table 5 shows some example output produced by the variants of our model which illustrate the main differences between the baseline and the three ILP models.

The baseline model tends to generate solutions with little cohesion between triples. Facts are enumerated which range over distinct topics. BL solutions also often include properties such as "source" which are generic rather than specific to the type of entity being described.

In contrast, S-Model solutions often contain sets of topically related properties (e.g., birth date and birth place) while C-Model solutions enumerate facts (affiliations, mascot, president, battle) about related

[5]The average confidence score produced by Crowdflower for the ratings is 0.63. Running a Fisher's exact test we obtain that the difference between the BL and the S-Model is statistically significant with p-value < 0.002. In contrast, C-Model and M-Model models are not significantly different, p-value < 0.1674.

	Example Solutions
BL (d1,n5)	Elliot_See | almaMater | University_of_Texas_at_Austin Elliot_See | status | "Deceased" Elliot_See | deathPlace | St._Louis Elliot_See | source | "See's feelings about being selected as an astronaut" Elliot_See | birthDate | "1927-07-23"
S-Model (d1,n5)	Elliot_See | almaMater | University_of_Texas_at_Austin Elliot_See | status | "Deceased" Elliot_See | deathPlace | St._Louis Elliot_See | birthDate | "1927-07-23" Elliot_See | birthPlace | Dallas
C-Model (d2,n6)	Elliot_See | almaMater | University_of_Texas_at_Austin Elliot_See | rank | United_States_Navy_Reserve University_of_Texas_at_Austin | affiliations | University_of_Texas_System University_of_Texas_at_Austin | mascot | Hook_'em_(mascot) University_of_Texas_at_Austin | president | Gregory_L._Fenves United_States_Navy_Reserve | battle | War_on_Terror
M-Model (d2,n6)	Elliot_See | deathDate | "1966-02-28" Elliot_See | deathPlace | St._Louis Elliot_See | rank | United_States_Navy_Reserve Elliot_See | almaMater | University_of_Texas_at_Austin University_of_Texas_at_Austin | affiliations | University_of_Texas_System University_of_Texas_at_Austin | athletics | Big_12_Conference

Table 5: Example content selections for the Astronaut entity Elliot_See.

entities (University of Texas, Austin and United States Navy Reserve). The M-Model lies in between, producing solutions that include both information about related entities and topic-grouped (death date, death place) facts about the entity being described.

	Depth 1	Depth 2	
	S-Model	C-Model	M-Model
n3	0.18	0.16	0.24
n4	0.29	0.21	0.35
n5	0.29	0.23	0.27
n6	0.27	0.23	0.23
n7	0.34	0.25	0.27
n8	0.36	0.26	0.24
n9	0.34	0.27	0.25
n10	0.39	0.30	0.25
Avg.	0.31	0.24	0.26

	Depth 2	Depth1 vs. Depth 2	
	C-Model M-Model	S-Model C-Model	S-Model M-Model
n3	0.21	0.10	0.12
n4	0.25	0.15	0.19
n5	0.25	0.16	0.19
n6	0.23	0.17	0.21
n7	0.25	0.19	0.25
n8	0.26	0.20	0.23
n9	0.26	0.21	0.22
n10	0.25	0.27	0.20
Avg.	0.24	0.18	0.20

Table 6: Quantifying the overlap between solutions (left) and between models (right).

6 Conclusion

We presented a method for selecting content from DBpedia data which leverages the n-gram information provided by large scale knowledge bases about entities of distinct ontological type. Based on the DBpedia graphs associated with entities of a given ontological type, we learn domain- specific n-gram models of DBpedia properties. To capture both discourse and topic-based coherence, we derive these n-grams either from chain or from sequence configurations of triples. As a result, we can extract content units based either on topic similarity, on elaboration-based discourse transition or on both. Using various metrics, we showed that our method supports the selection of content units that are both coherent and diverse.

We are currently working on exploiting this content selection procedure to semi-automatically construct a large data-to-text resource for training and testing RDF verbalisers. To associate the RDF subtrees we produce with the verbalisations required by supervised learning and evaluation, we plan to explore different methods including, the automatic generation of output using existing symbolic generators, the manual and semi-automatic validation of these automatically generated texts and the verbalisation of data units by humans, using crowdsourcing.

Acknowledgements

We thank the French National Research Agency for funding the research presented in this paper in the context of the WebNLG project.

References

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 331–338. Association for Computational Linguistics.

Or Biran and Kathleen McKeown. 2015. Discourse planning with an n-gram model of relations. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 1973–1977. Association for Computational Linguistics.

Nadjet Bouayad-Agha, Gerard Casamayor, and Leo Wanner. 2011. Content selection from an ontology-based knowledge base for the generation of football summaries. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 72–81. Association for Computational Linguistics.

Gong Cheng, Danyun Xu, and Yuzhong Qu. 2015. Summarizing entity descriptions for effective and efficient human-centered entity linking. In *Proceedings of the 24th International Conference on World Wide Web*, pages 184–194. ACM.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 25–32. Association for Computational Linguistics.

Polina Kuznetsova, Vicente Ordonez, Alexander C Berg, Tamara L Berg, and Yejin Choi. 2012. Collective generation of natural image descriptions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 359–368. Association for Computational Linguistics.

Gerasimos Lampouras and Ion Androutsopoulos. 2013. Using integer linear programming in concept-to-text generation to produce more compact texts. In *ACL (2)*, pages 561–566. Citeseer.

Rania Mohammed, Laura Perez-Beltrachini, and Claire Gardent. 2016. Category-driven content selection. In *Proceedings of the 9th International Natural Language Generation Conference (INLG)*, Edinburgh, Scotland. Poster.

Parallel Sentence Compression

Julia Ive[1,2], **François Yvon**[1]
LIMSI, CNRS, Univ Paris-Sud, Université Paris-Saclay, 91 403 Orsay, France,[1]
Cochrane France, INSERM U1153, 75181 Paris, France[2]
{firstname.lastname}@limsi.fr

Abstract

Sentence compression is a way to perform text simplification and is usually handled in a monolingual setting. In this paper, we study ways to extend sentence compression in a bilingual context, where the goal is to obtain parallel compressions of parallel sentences. This can be beneficial for a series of multilingual natural language processing (NLP) tasks. We compare two ways to take bilingual information into account when compressing parallel sentences. Their efficiency is contrasted on a parallel corpus of News articles.

1 Introduction

Text simplification is a well studied application of Natural Language Processing (NLP) techniques. Its main goal is to reduce the complexity of a text without degrading the informational content. This task proves useful for a wide range of applications, be they human-oriented (e.g. text adaptation for language learning purposes, for people with reading disabilities etc. (Siddharthan, 2014; Klaper et al., 2013)) or machine-oriented, serving as a basis to improve the efficiency of other NLP downstream components (e.g., parsing (Jonnalagadda et al., 2009), semantic role labeling (Vickrey and Koller, 2008) etc.). Simplification can be performed at different linguistic levels: lexical (Paetzold and Specia, 2016), syntactic (Siddharthan, 2011), or both (Paetzold and Specia, 2013).

Sentence compression is a way to perform simplification at the level of sentences, by reducing the sentence length without sacrificing important information. Many works only consider purely syntactic simplifications, though lexical changes are also possible, especially in language learning scenarios (Cohn and Lapata, 2008; Napoles et al., 2011a).

By and large, the motivations that have been put forward for monolingual sentence compression can be also used to motivate bilingual sentence compression, understood here as the generation of parallel compressions of parallel sentences. Bilingual Sentence Compression can be used, for instance, to produce simpler versions of a parallel text for learning purposes, or to generate summaries and subtitles in different languages, or even to build simplified parallel corpora for training a Machine Translation (MT) system.

Parallel sentence compression can be approached in many ways: it is first possible to compress independently each side of a bitext using monolingual simplification, an approach which however runs the risk of breaking the parallelism of the resulting corpus. Compression could also be performed in a symmetric manner by generalizing monolingual algorithms to the case of parallel sentences. We focus here on an **asymmetrical scenario**, where the target compression is a translation of a previously simplified source sentence.

In this context, our main contribution consists in studying various bitext compression methodologies that should **ensure the parallelism of the simplified bitext**, which, to our knowledge, is the first attempt of this kind. Two compression methods are developed and compared in Section 2 : (1) a dynamic programming (DP) approach, considering the final compression

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1503–1513, Osaka, Japan, December 11-17 2016.

as a result of a series of local optimal decisions and (2) an integer linear programming (ILP) method, capable to handle global constraints. These methods are used to compress texts in the News domain (see Section 3); we experiment there with data distributed in the context of the WMT'15 translation task (English-French, automatic compression attempted on French).

2 Bilingual and Monolingual Methods for Sentence Compression

In our asymmetrical bilingual context, we formulate the compression problem as follows: given a source sentence $\mathbf{e} = e_1, e_2, \ldots, e_J$, its compressed version $\mathbf{e}' = e'_1, e'_2, \ldots, e'_{J'}$ and its translation $\mathbf{f} = f_1, f_2, \ldots, f_I$, we search for $\mathbf{f}' = f'_1, f'_2, \ldots, f'_{I'}$ that translates $\mathbf{e}'$. $\mathbf{f}'$ should both **preserve the meaning** of $\mathbf{e}'$ and respect the **grammaticality** requirements of the target language. Most approaches to monolingual compression further assume that a (dependency) parse tree of $\mathbf{f}$ is available, taking the form of a set of (dependent, head) pairs. We make the same assumption here, denoting $\tau = \{(f_i, h(f_i)), 1 \leq i \leq I\}$ this dependency tree.

Recent proposals for solving this task use dynamic programming (DP) techniques (McDonald, 2006; Filippova, 2010) or integer linear programming (ILP) (Clarke and Lapata, 2008; Filippova et al., 2015), in both cases actively taking syntactic information into account. Inspired by those approaches, we propose below two methods for bilingual compression, enriched with MT-related bilingual information. We also include a description of our baseline compression system, based on two independent monolingual compressions.

2.1 Compressing with Finite-State Machines and Dynamic Programming (DPbi)

Our first approach to compression (DPBI) uses finite-state techniques. Recall that a weighted finite-state automaton (WFSA) over a set of weights $\mathbb{K}$ is represented by a 7-tuple $A = (\Sigma, Q, B, F, E, \lambda, \rho)$, where Σ is a finite alphabet, Q is a finite set of states, $B \subseteq Q$ contains the initial states and $F \subseteq Q$ the final states; $E \subseteq Q \times \Sigma \times \mathbb{K} \times Q$ is a set of weighted transitions, $\lambda : B \to \mathbb{K}$ and $\rho : F \to \mathbb{K}$ are respectively the initial and final weight functions (Mohri, 1997).

Given $\mathbf{f}$, the search space for compression is built as follows: assuming $\mathbf{f}$ conventionally starts (respectively ends) with `<s>` at index f_0 (resp. `</s>` at index f_{I+1}), we first build the standard automaton $A_{\mathbf{f}}$ for $\mathbf{f}$, the states of which correspond to the prefixes of $\mathbf{f}$.

We then add "skip" transitions $(q_k, f_l, w, q_l,)$, $\forall l > k+1$. In this step, we make sure to preserve the syntactic dependency relationships so as to ensure that the subgraph induced by words in the compression is a subtree of the complete dependency tree. To this end, skip transitions $(q_k, f_l, w, q_l,)$ are created subject to the condition that $\forall m, k < m < l$, f_m is neither an ancestor of f_k nor an ancestor of f_l.

When the dependency trees are projective, these conditions are sufficient to ensure that compressions will be grammatical: if a compression contains a word f_i, it will also contain its head. To see why, assume with no loss of generality that $h(f_i) < f_i$ for some i. If f_i is in the compression, the incoming arc in $A_{\mathbf{f}}$ either starts in node f_k with $k < l$, or $h(f_i)$ precedes f_k (it cannot be skipped). Either $f_k = h(f_i)$, which is what we seek; or f_k is a descendant of $h(f_i)$. We can then repeat the same argument with the arc labeled f_k. It is also routine to check that all possible grammatical compressions can be obtained in this way.

Transitions $(q_k, f_l, w, q_l,)$, with $l > k + 1$, are weighted according to a score w aggregating:

- S_{LM} - a 2-gram language model (LM) score : $S_{LM}(f_l|f_k) = \log P(f_l|f_k)$. The generalization to higher-order n-grams is straightforward. In our implementation, we have used a POS-based LM, as we believe it will provide a better generalization than a word-based LM;
- $S_{IBM}(f_l|\mathbf{e}')$ - the posterior log-probability of f_l in the IBM model 1 of Brown et al. (1993):

$$S_{IBM}(f_l|\mathbf{e}') = \log\big(\frac{1}{(J'+1)}\sum_{j=1}^{J'} t(f_l|e'_j)\big). \quad (1)$$

These scores are summed along a path and yield the total IBM score: $IBM_1(\mathbf{f}'|\mathbf{e}')$.

- $S_{ali}(f_l)$ - this score approximates the contribution of f_l to the posterior log-probability of the $IBM_1(\mathbf{e}'|\mathbf{f})$ at the sentence level:

$$S_{ali}(f_l) = \sum_{i=1}^{J'} \mathbb{1}\{f_l = \operatorname*{argmax}_f t(e'_i|f)\} t(e'_i|f_l), \tag{2}$$

  where $\mathbb{1}\{T\}$ is the indicator function for predicate T. This approximation is required due to the impossibility to decompose the inverse IBM model 1 score over the arcs of $A_{\mathbf{f}}$.

The use of IBM model 1 scores is meant to ensure the preservation of the meaning of the provided source compression $\mathbf{e}'$, hence the parallelism of the resulting sentence pair. Each arc is additionally weighted with a word penalty, which should ensure that short paths are not improperly given preference over longer ones.

The score of a path generating a target string $\mathbf{f}'$ is finally computed as a summation of all arcs in the path:

$$S(\mathbf{f}'|\mathbf{e}') = \alpha \cdot S_{LM}(\mathbf{f}') + \beta \cdot S_{IBM}(\mathbf{f}'|\mathbf{e}') + \gamma \cdot S_{ali}(\mathbf{f}') + \delta \cdot l_{\mathbf{f}'}, \tag{3}$$

where α, β, γ and δ are tunable parameters. The optimal target compression is computed via standard shortest path techniques. Note that this approach can be generalized in many ways, notably including additional costs, subject to *locality constraints*: scoring functions evaluating properties of the compressed sentence should decompose over the arcs of the automaton. This restriction warrants a more generic approach to the problem, relying on Integer Linear Programming.

2.2 Compressing with Integer Linear Programming (ILPbi)

Integer Linear Programming (ILP) (Dantzig and Thapa, 1997) is an optimization approach to solving combinatorial problems that can be expressed as linear programs and in which some or all of the variables are restricted to be non-negative integers. In the following, we heavily rely on the work of Clarke and Lapata (2008), who develop an approach based on ILP for monolingual sentence compression.

The formulation of an ILP problem requires the definition of:

- decision variables, they will be binary in our case;
- an objective function, corresponding to the compression score we wish to maximize;
- constraints, i.e. linguistic or consistency conditions restricting the possible values of decision variables.

The main decision variables in our problem indicate whether a target word f_i initially in $\mathbf{f}$ also occurs in the compressed text:

$$\forall i \in [1 \dots I], x_i = \begin{cases} 1 & \text{if } f_i \text{ is in the compression} \\ 0 & \text{otherwise} \end{cases}$$

Following again Clarke and Lapata (2008), we define additional decision variables for the 2-gram LM scores (again, the generalization to higher-order n-grams is straightforward):

$$\forall i \in [1 \dots I], y_i = \begin{cases} 1 & \text{if } f_i \text{ starts the compression} \\ 0 & \text{otherwise} \end{cases}$$

$$\forall i \in [1 \dots I], p_i = \begin{cases} 1 & \text{if } f_i \text{ ends the compression} \\ 0 & \text{otherwise} \end{cases}$$

$$\forall i \in [1 \dots I-1], \forall k \in [i+1 \dots I], z_{ik} = \begin{cases} 1 & \text{if the pair } (f_i, f_k) \text{ is in the compression} \\ 0 & \text{otherwise} \end{cases}$$

Our ILPBI model is again integrated with IBM model 1 scores, which help ensure the preservation of meaning in $\mathbf{e}'$. For our approximation of the $IBM_1(\mathbf{e}'|\mathbf{f})$ score described in Section 2.1, we introduce a new variable:

$$\forall i \in [1 \dots I], \forall m \in [1 \dots J], a_{im} = \begin{cases} 1 & \text{if } g(t(\mathbf{e}'_m|f_i)) > 0 \\ 0 & \text{if } g(t(\mathbf{e}'_m|f_i)) = 0 \end{cases}, \tag{4}$$

where $g(t(\mathbf{e}'_m|f_i)) = \mathbb{1}_{f=\text{argmax}_f\, t(\mathbf{e}'_m|\mathbf{f})}(t(\mathbf{e}'_m|f_i))$ tests whether f_i is the best alignment for $\mathbf{e}'_m$.

Our objective function models 2-gram LM scores, as well as bilingual $IBM_1(\mathbf{f}|\mathbf{e}')$ and our approximation of $IBM_1(\mathbf{e}'|\mathbf{f})$:

$$\begin{aligned} S(\mathbf{f}'|\mathbf{e}') \; =& \alpha \sum_{i=1}^{I} x_i \cdot \log\big(\frac{1}{(I+1)} \sum_{i=1}^{I} t(f_l|e'_i)\big) + \gamma \sum_{i=1}^{I} y_i \cdot \log P(f_i|\text{<s>}) \\ &+ \gamma \sum_{i=1}^{I-1} \sum_{k=i+1}^{I} z_{ik} \cdot \log P(f_k|f_i) + \beta \sum_{m=1}^{J} \sum_{i=1}^{I} a_{i,m} \cdot \log t(\mathbf{e}'_m \mathbf{f}_i) + \gamma \sum_{i=1}^{I} p_i \cdot \log P(\text{</s>}|f_i) \end{aligned} \tag{5}$$

subject to: $x_i, y_i, z_{ik}, p_i, a_{i,m} \in \{0, 1\}$. The following constraints are also applied for generation of valid n-gram sequences without word repetition:

Constraint 1 Exactly one word can start a compression.

$$\sum_{i=1}^{I} y_i = 1 \tag{6}$$

Constraint 2 If a word is in the compression it must either start it, or must follow another word.

$$\forall k : k \in [1 \dots I], x_k - y_k - \sum_{i=1}^{k-1} z_{ik} = 0 \tag{7}$$

Constraint 3 If a word is in the compression it must either be followed by another word or end the sentence.

$$\forall i : i \in [1 \dots I], x_i - (\sum_{k=i+1}^{I} z_{ik}) - p_i = 0 \tag{8}$$

Constraint 4 Exactly one word can end a compression.

$$\sum_{i=1}^{I} p_i = 1 \tag{9}$$

Constraint 5 A dependent f_i cannot be included in a compression without its head $h(f_i)$ from the dependency tree $\tau_{\mathbf{f}}$, the constraint that ensures the grammaticality of $\mathbf{f}'$. This is a simplified version of constraints (20)-(24) of (Clarke and Lapata, 2008).

$$\forall i, x_{h(i)} - x_i \geq 0 \tag{10}$$

Constraint 6 If a compression contains a left bracket/quote mark it should contain the right bracket/quote mark.

$$x_{left} - x_{right} = 0 \tag{11}$$

Constraint 7 A compression is to be at least b tokens long. This constraint controls the compression length and prevents the model to generate too short compression preferred by LM.

$$\sum_{i=1}^{I} x_i \geq b \tag{12}$$

Constraint 8 If a word is in the compression it can have a best alignment in $\mathbf{e}'$.

$$x_i - a_{im} >= 0 \tag{13}$$

To complete this section, we now present our tools for monolingual compression, that will be used in our baseline system.

2.3 Monolingual Compression (DPmono and ILPmono)

In the monolingual scenario, we are given a target sentence $\mathbf{f} = f_1, f_2, \ldots, f_I$. The goal is to produce a target compression $\mathbf{f'} = f'_1, f'_2, \ldots, f'_{I'}$ by removing any subset of words in $\mathbf{f}$, given the dependency tree τ. Here we also assume to be given the lemmas corresponding to the words in $\mathbf{f}$: $m(f)$ denotes the lemma associated with word f. Here again, we look for $\mathbf{f'}$ that should be grammatical and preserve the main aspects of the meaning of $\mathbf{f}$.

We introduce grammaticality constraints in our compressions in a way similar to our bilingual methods. To help meaning preservation we introduce the following semantic importance score for the sense-bearing words of $\mathbf{f}$, namely nouns, verbs, adjectives and adverbs, inspired again by Clarke and Lapata (2008):

$$S_{sem} = \frac{1}{D_{f_{sb}}} fr_{m(sb)} \log \frac{F}{F_{m(sb)}} \tag{14}$$

This score is a `TD-IDF` measure for the lemma $m(f)$ of each sense-bearing word, weighted proportionally to its depth in the dependency tree $D_{f_{sb}}$, where $fr_{m(sb)}$ and $F_{m(sb)}$ are respectively the frequency of the lemma in the news article and in the corpus, and F is the count of all sense-bearing lemmas in the corpus.

1. DPmono For the DPmono approach, we construct an WFSA in a way similar to the one described in 2.1. We weight each arc in the automaton with S_{LM} and S_{sem}. The score of a path generating a target string $\mathbf{f'}$ is computed as:

$$S(\mathbf{f'}|\mathbf{e'}) = \alpha \cdot S_{LM}(\mathbf{f'}) + \beta \cdot S_{sem}(\mathbf{f'}) + \gamma \cdot l_{\mathbf{f'}}, \tag{15}$$

where α, β and γ are tunable parameters.

2. ILPmono The objective function for ILPmono similarly models $S_{sem}(f_i)$ and 2-gram LM scores:

$$S(\mathbf{f'}|\mathbf{e'}) = \sum_{i=1}^{I} x_i \cdot \theta S_{sem}(f_i) + \eta \sum_{i=1}^{I} y_i \cdot \log P(f_i|\text{<s>}) + \eta \sum_{i=1}^{I-1} \sum_{k=i+1}^{I} z_{ik} \cdot \log P(f_k|f_i) + \eta \sum_{i=1}^{I} p_i \cdot \log P(\text{</s>}|f_i),$$

subject to: $x_i, y_i, z_{ik}, p_i \in \{0, 1\}$, with θ and η as tunable parameters. Constraints 1-7 of ILPbi are applied.

3 Experimental Setup

In our evaluation experiments we were guided by the following questions: (1) Does the proposed bilingual compression methodology ensure **the resulting corpus parallelism** (compressed target text succeeds in preserving the meaning of the input compressed source text)? (2) Which of the proposed bilingual compression methods is more **efficient**?

3.1 Metrics

To answer the above questions, we compared the results of our bilingual methods DPBI and ILPBI to the results of our monolingual methods DPMONO and ILPMONO (baselines) using the compression rate (COMPR) estimation, F-SCORE metric for the relations in grammatical dependency parse trees (Chen and Manning, 2014), as well as the standard MT metric BLEU (Papineni et al., 2002; Clarke and Lapata, 2008; Napoles et al., 2011b). F-SCORE measures how much meaning is preserved in the compression using the grammatical-functional information, BLEU measures the fluency of the produced compressions.

We also computed the confidence score (CS) of the parallelism between the produced compressions and the compressed source. We used a Logistic Regression model trained on the parallel sentences extracted from a corpus of manual alignments. The model exploits such features as the length difference ratio, IBM model 1 scores, the cosine similarity of the distributional representations for source and target etc. Details regarding the model are in (Xu et al., 2015).

3.2 Data and Translation Models

Our experiments used the News data provided by the organizers of the WMT'15 news translation task (English-French) [1]. Details regarding data preparation are in (Marie et al., 2015).

We randomly chose around 60 articles from the News 2014 test set to be used as our development and test sets. The compressed source and reference data were created manually by three annotators. Two of them were native French speakers, all the three annotators were fluent in both English and French. The annotators were given the instruction to delete any quantity of words without disturbing the meaning and the grammaticality of a source English sentence. Target French words "translating" deleted source words were deleted accordingly under the condition that a sentence stays grammatical. This resulted in the small compression rate of COMPR = 79.74 at the source side and of COMPR = 84.45 at the target side [2].

In our experiments, we used the French 3-gram POS LMs trained with Witten-Bell smoothing on a part of the target side of the parallel Giga corpus (≈14% of the available corpus) using the SRILM (Stolcke, 2002) toolkit. The same parallel data and the uncompressed development and test sets were used to compute the IBM1 scores in both directions with the help of MGIZA++ (Gao and Vogel, 2008), and to estimate the lemma frequencies for the semantic importance scores (Schmid, 1995). The dependency parse trees were produced using the Stanford parser (Chen and Manning, 2014).

The statistics of the development and test data, as well as the data used for model estimation are in Table 1.

		Full		Compression	
	lines	tok., en	tok., fr	tok., en	tok., fr
development set	504	11K	13K	9K	11K
test set	489	12K	14K	10K	12K
Giga	3M	71M	86M		

Table 1: Data Statistics

For the WFSA experiments, we mostly used the SRILM toolkit (Stolcke, 2002). The ILP

[1] http://www.statmt.org/wmt15/translation-task.html

[2] The traces of compression operations for the corpus are available at http://perso.limsi.fr/ive.

experiments were performed with the GLPK toolkit.[3] All the tunable parameters were tuned to optimize BLEU. In our implementation of ILPBI, we slightly modified constraint 5 to take into account the negation relations: when the head of such relation are included in the compression, then their immediate dependent must also be selected. This is because the preservation of such relation is crucial for the overall meaning of a sentence. Finally, constraint 7 is parametrized by a preset compression ratio, rather than a preset target length; in our experiments this compression ratio was set to match the compression ratio of DPBI so as to make our results more comparable.

3.3 Evaluation

Results of our experiments comparing bilingual and monolingual methods are in Table 2.

model	COMPR	F-SCORE	BLEU	CS
DPMONO	82.58	80.37	73.37	0.94
DPBI	84.63	81.71	**76.09**	0.96
ILPMONO	85.10	80.18	62.59	0.95
ILPBI	85.06	**82.72**	69.85	**0.97**
Ref.	83.59			0.99

Table 2: Evaluation results

As reflected by the automatic metrics, the bilingual methods produce compressions that better preserve the meaning of the source sentence than the corresponding monolingual methods, hence improving the parallelism of the resulting compressions (average ΔF-SCORE = 1.94, ΔBLEU = 4.99 and ΔCS = 0.02, between the ILP and DP monolingual and bilingual methods) (see Table 3).

Ref.	Irak: **octobre** a été le mois le plus sanglant depuis 2008
	'Iraq: **October** was the bloodiest month since 2008'
DPMONO	Irak: a été le mois le plus sanglant depuis 2008
	'Iraq: was the bloodiest month since 2008'
DPBI	Irak : **octobre** a été le mois le plus sanglant depuis 2008
	'Iraq: **October** was the bloodiest month since 2008'
Ref.	La ville de New York en envisage **un**.
	'The city of New York is considering **one**.'
ILPMONO	La ville de New York en envisage.
	'The city of New York is considering.'
ILPBI	Ville de New York en envisage **un**.
	'City of New York is considering **one**.'

Table 3: Examples of compressions produced by the monolingual and bilingual methods

F-SCORE and CS estimations suggest that ILPBI is a slightly more efficient method than DPBI in terms of preserving parallelism (+1.01 F-SCORE, +0.01 CS). Due to the global constraints, ILPBI tends to include more sense-bearing words in the compression (see Table 4, first example). Our example shows that ILPBI kept the word "cour" 'court'. This noun is more important for understanding the meaning of the sentence than the preposition "en" 'in' following the verb "présenter" 'appear', chosen due to the local decision taken by DPBI.

At the same time the length constraint "obliges" ILPBI to compress every sentence. In our setting with the small compression rate, short sentences often stay uncompressed. In this case, DPBI is able to choose the automaton path of the maximum length (see Table 4, second example). ILPBI though in this case deletes the auxiliary words (articles in our example). This

[3] `http://glpk-java.sourceforge.net`

is reflected in the decrease of the BLEU score for ILPBI as compared to DPBI (-6.24 BLEU). BLEU here is heavily penalized by the decrease in matching n-grams of the order $n > 1$.

ILP methods can also be very convenient for a series of reasons. E.g., in the absence of development corpora ILP compressions can be obtained with minimum or without any tuning. Another advantage is the easy parameterization of the compression rate. This criteria was very important for our task with the small compression rate. Keeping it was crucial for correct evaluation of our methodology. Thus, for the DP methods we observe the compression rate variation of ΔCOMPR = 2.05, for the ILP methods this variation is insignificant (ΔCOMPR = 0.04).

Ref.	Omar Hassan est toujours en détention et se présenter **en cour** vendredi.
	'Omar is still in custody and will appear **in court** on Friday.'
ILPBI	Omar Hassan est toujours en détention et se présenter **cour** vendredi.
	'Omar is still in custody and will appear **court** on Friday.'
DPBI	Omar Hassan est toujours en détention et se présenter **en** vendredi.
	'Omar is still in custody and will appear **in** on Friday.'
Ref.	Après **un** accord de paix signé en 1992, elle est devenue **un** parti d'opposition .
	'After **a** peace agreement signed in 1992, it became **an** opposition party.'
ILPBI	Après accord de paix signé en 1992, elle est devenue parti opposition.
	'After peace agreement signed in 1992, it became opposition party.'
DPBI	Après **un** accord de paix signé en 1992, elle est devenue **un** parti d'opposition .
	'After **a** peace agreement signed in 1992, it became **an** opposition party.'

Table 4: Examples of ILPBI and DPBI compressions

4 Related Work

The deletion-based compression problem has been studied using a series of modeling paradigms. We mention first the work of Knight and Marcu (2002), who use the *noisy channel* model. This approach aims to maximize $P(\mathbf{f}'|\mathbf{f}) \propto P(\mathbf{f}')P(\mathbf{f}|\mathbf{f}')$, where $P(\mathbf{f}')$ is the source model, and $P(\mathbf{f}|\mathbf{f}')$ models the syntactic parse tree probability of the long sentence being an expansion of the compressed one. The noisy channel model is also used by approaches that consider compression as a monolingual translation problem (Napoles et al., 2016).

McDonald (2006) formulates the problem as a binary sequence labeling problem with a rich syntactic feature set, and proposes a solving procedure based on dynamic programming techniques. More recent DP solutions to the sentence compression problem use neural network architectures (Filippova et al., 2015).

The ILP approach to compression was introduced by Clarke and Lapata (2008). The main motivation was the necessity to take global features into account (e.g., the constraint to have at least one verb in the compressed sentences). This approach has been widely reused in research related to text compression with various modifications to syntactic and informativeness scores used by Clarke and Lapata (2008) (see also (Wei et al., 2015; Filippova and Altun, 2013)).

In our bilingual framework we compare the performance of DP and ILP approaches. As far as we know this is the first attempt to create compressed parallel bitext in asymmetrical setting. A closely related work is that of Aziz et al. (2012), who also exploits bilingual information. The authors propose a PBSMT solution for joint translation and compression of subtitles, which dynamically decides where it is necessary to impose a space/time constraint on the translated text.

5 Conclusions

In this paper we consider sentence compression in a bilingual setting. We adopt an asymmetrical view to the task, where we first compress the source, then look for a compressed target translating the reduced source. Based on recent research on these issues, our main contribution is to adapt

existing monolingual compression techniques to produce compressed bitext. As we know this the first attempt of the kind. We use dynamic programming (DP) and integer linear programming methods (ILP) enriched with bilingual features. Both methods improve the preservation of the compressed source meaning, hence the parallelism of the resulting bitext, as opposed to using independently monolingual methods in source and target.

In our setting, ILP was found to perform better than DP; the ILP method is more flexible and requires less resources for tuning; furthermore, it can accommodate more complex (e.g. global) constraints. Our future work includes exploring additional global constraints, experimenting with shorter compressions, as well as using basic phrase-based statistical machine translation (PBSMT) techniques, including applying the noisy channel model and beam-search decoding to find the best possible target compression.

The results of bilingual compression can be used for human-oriented (language learning, subtitles generation etc.) purposes, or in a large spectrum of natural language processing (NLP) tasks. The approach can be extended to paraphrastic compression (as opposed to deletion-based), as well as applied in the symmetric compression scenario, when source and target are compressed simultaneously.

Acknowledgements

The work of the first author is supported by a CIFRE grant from the French ANRT. We would like to thank Yong Xu for helping us with the computation of the parallelism confidence scores, as well as the annotators for their participation in the creation of the compressed corpus.

References

Wilker Aziz, Sheila Castilho Monteiro de Sousa, and Lucia Specia. 2012. Cross-lingual sentence compression for subtitles. In *16th Annual Conference of the European Association for Machine Translation*, EAMT, pages 103–110, Trento, Italy.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression an integer linear programming approach. *J. Artif. Int. Res.*, 31(1):399–429, March.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 137–144, Manchester, UK, August. Coling 2008 Organizing Committee.

George B. Dantzig and Mukund N. Thapa. 1997. *Linear Programming 1: Introduction*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*, pages 1481–1491. (Click on the Abstract link to get access to the described dataset).

Katja Filippova, Enrique Alfonseca, Carlos Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*.

Katja Filippova. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 322–330, Beijing, China, August. Coling 2010 Organizing Committee.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 49–57, Columbus, Ohio, June. Association for Computational Linguistics.

Siddhartha Jonnalagadda, Luis Tari, Jörg Hakenberg, Chitta Baral, and Graciela Gonzalez. 2009. Towards effective sentence simplification for automatic processing of biomedical text. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 177–180, Boulder, Colorado, June. Association for Computational Linguistics.

David Klaper, Sarah Ebling, and Martin Volk. 2013. Building a german/simple german parallel corpus for automatic text simplification. In *Proceedings of the Second Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pages 11–19, Sofia, Bulgaria, August. Association for Computational Linguistics.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artif. Intell.*, 139(1):91–107, July.

Benjamin Marie, Alexandre Allauzen, Franck Burlot, Quoc-Khanh Do, Julia Ive, Elena Knyazeva, Matthieu Labeau, Thomas Lavergne, Kevin Löser, Nicolas Pécheux, and François Yvon. 2015. LIMSI@WMT'15 : Translation task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 145–151, Lisbon, Portugal, September. Association for Computational Linguistics.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 297–304.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23:269–311.

Courtney Napoles, Chris Callison-Burch, Juri Ganitkevitch, and Benjamin Van Durme. 2011a. Paraphrastic sentence compression with a character-based metric: Tightening without deletion. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 84–90, Portland, Oregon, June. Association for Computational Linguistics.

Courtney Napoles, Benjamin Van Durme, and Chris Callison-Burch. 2011b. Evaluating sentence compression: Pitfalls and suggested remedies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, MTTG '11, pages 91–97, Stroudsburg, PA, USA. Association for Computational Linguistics.

Courtney Napoles, Chris Callison-Burch, and Matt Post. 2016. Sentential paraphrasing as black-box machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 62–66, San Diego, California, June. Association for Computational Linguistics.

Gustavo Paetzold and Lucia Specia. 2013. Text simplification as tree transduction. In *Ninth Brazilian Symposium in Information and Human Language Technology*, STIL, pages 116–125, Fortaleza, Brazil.

Gustavo Paetzold and Lucia Specia. 2016. Benchmarking lexical simplification systems. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, May. European Language Resources Association (ELRA).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL*, Philadelphia, US.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to German. In *In Proceedings of the ACL SIGDAT-Workshop*, pages 47–50.

Advaith Siddharthan. 2011. Text simplification using typed dependencies: A comparision of the robustness of different generation strategies. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 2–11, Nancy, France, September. Association for Computational Linguistics.

Advaith Siddharthan. 2014. A survey of research on text simplification. *International Journal of Applied Linguistics*, 165(2):259–298.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, pages 901–904, Denver, Colorado, September.

David Vickrey and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of ACL-08: HLT*, pages 344–352, Columbus, Ohio, June. Association for Computational Linguistics.

Zhongyu Wei, Yang Liu, Chen Li, and Wei Gao. 2015. Using tweets to help sentence compression for news highlights generation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 50–56, Beijing, China, July. Association for Computational Linguistics.

Yong Xu, Aurélien Max, and François Yvon. 2015. Sentence alignment for literary texts. *Linguistic Issues in Language Technology*, 12(6):25 pages.

An Unsupervised Multi-Document Summarization Framework Based on Neural Document Model

Shulei Ma **Zhi-Hong Deng*** **Yunlun Yang**
Key Laboratory of Machine Perception (Ministry of Education)
School of Electronics Engineering and Computer Science
Peking University, Beijing 100871, China
mashulei@pku.edu.cn, zhdeng@cis,pku.edu.cn
incomparable-lun@pku.edu.cn

Abstract

In the age of information exploding, multi-document summarization is attracting particular attention for the ability to help people get the main ideas in a short time. Traditional extractive methods simply treat the document set as a group of sentences while ignoring the global semantics of the documents. Meanwhile, neural document model is effective on representing the semantic content of documents in low-dimensional vectors. In this paper, we propose a document-level reconstruction framework named *DocRebuild*, which reconstructs the documents with summary sentences through a neural document model and selects summary sentences to minimize the reconstruction error. We also apply two strategies, sentence filtering and beamsearch, to improve the performance of our method. Experimental results on the benchmark datasets DUC 2006 and DUC 2007 show that *DocRebuild* is effective and outperforms the state-of-the-art unsupervised algorithms.

1 Introduction

Multi-document summarization aims at capturing the important information of a set of documents related to the same topic and presenting it in a brief, representative, and pertinent summary. Most existing researches focus on extraction-based methods, in which sentences are selected from the original document set.

Typically, two kinds of unsupervised models are used for sentence selection. One is based on sentence ranking, which uses methods such as clustering (Lin and Hovy, 2002; Radev et al., 2004), PageRank (Erkan and Radev, 2004; Mihalcea and Tarau, 2005) and topic modeling (Harabagiu and Lacatusu, 2005; Wang et al., 2008), to rank the sentences. Considering that top-ranked sentences tend to convey much redundant information, additional strategies are usually applied to reduce redundancy when selecting sentences. This kind of methods usually need to weigh between relevance and redundancy, which may be hard to balance.

The other is based on sparse reconstruction (He et al., 2012; Liu et al., 2015; Yao et al., 2015), which selects a sparse subset of the sentences that can linearly reconstruct all the sentences in the original document set. This kind of methods has a good motivation but also weaknesses in their hypothesises. First, reconstructing single sentences may lose the global information of documents; Second, there are more reasonable ways than linear combination in reconstruction.

Commonly, in the above methods the document set is treated as a set of sentences and all the operations are carried out on the sentence set, losing the global information of documents. Meanwhile, neural document model (Le and Mikolov, 2014; Li et al., 2015; Lin et al., 2015) is an emerging technique which has made significant progress in capturing semantic information of documents by projecting the text into the low-dimensional continuous distributed representation. It has been applied to natural language processing tasks such as sentiment classification (Tang et al., 2015).

*Zhi-Hong Deng is the corresponding author.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1514–1523, Osaka, Japan, December 11-17 2016.

In this paper, we address the aforementioned problems of existing methods and propose a document-level reconstruction framework based on neural document model, named *DocRebuild*, for multi-document summarization.

Intuitively, a good summary is supposed to reconstruct the main content of the multi-document set. In our model, we first introduce neural document model to represent the content of each document and use averaging to obtain the main content of the document set. The main content is reconstructed by concatenating the summary sentences into a sequence and feeding the summary sequence into the document model. Hence the multi-document summarization is converted into an optimization problem via taking the reconstruction error as the objective function. Summary sentences are selected to minimize this error. Furthermore, two strategies are addressed in selecting sentence to yield a better performance. First, irrelevant sentences are filtered and only a subset of related sentences is reserved as candidate set. Second, a beamsearch algorithm is applied to get a better solution in sentence selection stage.

Our contributions can be concluded as follows:

- We introduce neural document model into multi-document summarization task. As far as we know, no such works have been presented before.
- We propose a document-level reconstruction framework *DocRebuild*, and further adopt two effective strategies to improve the performance of our method.
- The experimental results on two benchmark DUC data sets show that our method outperforms the state-of-the-art unsupervised approaches.

2 Related Work

Multi-document summarization has received widespread attention in recent years. Most existing multi-document systems use extraction-based methods, in which sentences are directly selected from the original document set.

The majority of these methods use the idea of sentence-ranking, assigning salient scores to sentences of the original document set and choosing the top sentences to form the summary. Typical methods are the centroid-based methods (Lin and Hovy, 2002; Radev et al., 2004), which score sentences basing on features such as cluster centroids, sentence position and TF-IDF. Besides, graph based models (Erkan and Radev, 2004; Mihalcea and Tarau, 2005) first measure the sentence similarity then use ranking algorithm such as PageRank on the similarity graph to estimate the importance of different sentences. Topics in documents are also discovered to be an effective feature for sentence ranking (Hardy et al., 2002; Harabagiu and Lacatusu, 2005; Wang et al., 2008). Maximum Marginal Relevance (MMR) (Goldstein et al., 1999) is widely used for greedily selecting sentences while considering the tradeoff between relevance and redundancy.

However, it is usually hard to get a good balance between relevance and redundancy. Recently, a couple of works have employed the idea of data reconstruction in the summarization task. DSDR (He et al., 2012) reconstructs each sentence in the document set by a non-negative linear combination of summary sentences then minimizes the reconstruction error. MDS-Sparse (Liu et al., 2015) introduces the diversity constraint and proposes a two-level sparse representation model to reconstruct the sentences in the document set. SpOpt (Yao et al., 2015) follows the sparse representation framework while simultaneously doing sentence selection and compression by adjusting reconstruction coefficients and compression coefficients alternately in optimization.

In this work, neural document model is involved in performing summarization task on document level. With the development of deep learning, some attempts have been made to model documents with neural networks. Le and Mikolov (2014) extends the neural network of word embedding (Mikolov et al., 2013) to learn the document embedding. Li et al. (2015) uses a hierarchical long-short term memory auto-encoder to reconstruct the original document. Lin et al. (2015) proposes a hierarchical recurrent neural network language model to consider sentence history information in word prediction. Tang et al. (2015) presents a convolutional-gated recurrent neural network and applies it to sentiment classification task.

However, few such researches have been reported on document level in multi-document summarization task.

3 Proposed Framework

We propose our framework *DocRebuild* in this section. The neural document model is introduced in Section 3.1, the objective function is formulated in Section 3.2, and summary sentences are selected with two effective strategies as shown in Section 3.3. Figure 1 illustrates the framework of *DocRebuild*.

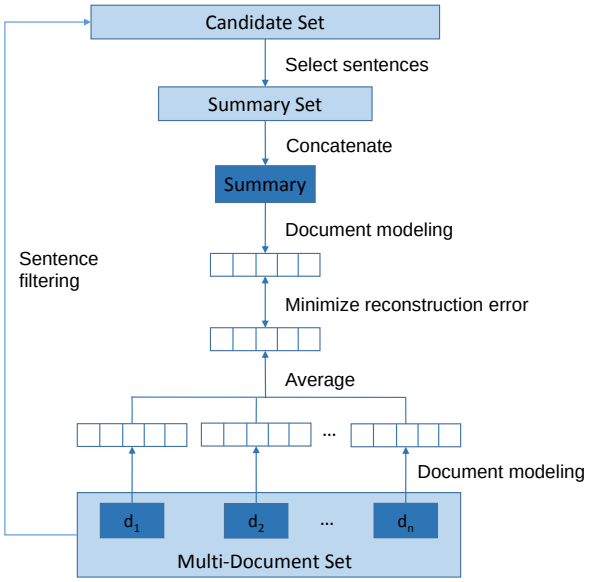


Figure 1: The framework of *DocRebuild*. Light blue boxes represent the sets of documents or sentences, deep blue boxes represent the document or summary sequences. Document modeling process projects document or summary sequences into real-valued vectors as represented by the bars.

3.1 Neural Document Model

Neural document model aims to represent the semantic content of a document with low-dimensional vector representation. As the basis of our framework, it is the key to a good performance. Here we focus on the unsupervised methods and exploit two kinds of unsupervised document models, named Bag-of-Words(BoW) model and Paragraph Vector(PV) model respectively, in our task.

In the BoW model, we simply use the bag-of-words of the document without considering the original order or relationships between neighboring words. Word embedding(Mikolov et al., 2013) has been proven of great significance in most natural language processing tasks in recent years. So we represent each word by its corresponding word embedding and the document is represented as the weighted average of all the words in the document.

Since BoW model is likely to lose the semantic information hidden in the order and composition of words, we introduce a more complex model which takes word order into consideration. PV (Le and Mikolov, 2014) is an unsupervised framework that learns distributed representations for sentences and documents. Compared with other hierarchical document models (Li et al., 2015; Lin et al., 2015) built upon sentences, PV handles texts with various length in a common way, making it possible to measure sentences, short summaries and long documents in the same semantic space.

Figure 2 illustrates the framework of PV model. In this model, every document is mapped to a unique vector, as a column in matrix $D \in \mathbb{R}^{n \times l}$ and every word is mapped to a unique vector, as a column in matrix $W \in \mathbb{R}^{m \times l}$. n, m, l represent document set size, vocabulary size and dimensionality of vector, respectively.

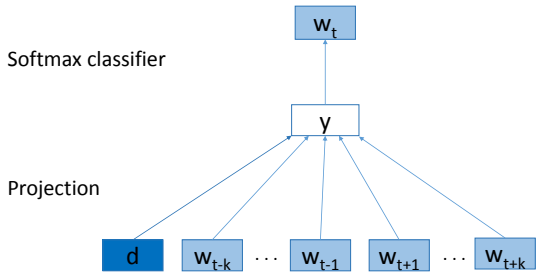


Figure 2: The framework of PV model

It employs a similar idea as the one in Mikolov et al. (2013), to predict the next word given many contexts sampled from the document. More precisely, given a document d consisting of a sequence of words $w_1, w_2, \ldots, w_T$ and the fixed window size k of context, the objective of PV model is to maximize the log-likelihood,

$$\mathcal{L} = \sum_{t=1}^{T} \log \mathbb{P}(w_t | w_{t-k} : w_{t+k}, d) \tag{1}$$

Note that $w_{t-k} : w_{t+k}$ represents the word sequence from w_{t-k} to w_{t+k} except w_t. The probability $\mathbb{P}(w_t | w_{t-k} : w_{t+k}, d)$ is defined using the softmax,

$$\mathbb{P}(w_t | w_{t-k} : w_{t+k}, d) = \frac{e^{y_{w_t}}}{\sum_{i=1}^{m} e^{y_{w_i}}} \tag{2}$$

$\mathbf{y} \in \mathbb{R}^m$ represents un-normalized log probability for all the possible words in vocabulary, computed as,

$$\mathbf{y} = Uh(w_{t-k} : w_{t+k}, d) + b \tag{3}$$

where $U \in \mathbb{R}^{m \times l}, b \in \mathbb{R}^m$ are the parameters of softmax classifier and h stands for the averaging of document vector and word vectors extracted from D and W.

During training, parameters D, W, U, b are randomly initialized and then updated to maximize equation (1) using stochastic gradient descent via backpropagation. Once the model is trained, it can further be employed in predicting representations of the documents not included in the training set.

At inference stage, a new document is fed into the PV model to do the same prediction task as the training documents. But this time only the document vector d is randomly initialized while other parameters W, U, b are fixed. Then we update the document vector d to maximize equation (1) by gradient descent as well. After convergence, d is taken as the corresponding document vector.

3.2 Objective Function

We denote the **multi-document set** as $D = \{d_1, d_2, \ldots, d_n\}$. All the documents in D are processed into a group of sentences, defined as the **candidate set** and denoted as $C = \{s_1, s_2, \ldots, s_m\}$. The sentences selected from S form the **summary set**, denoted as $S = \{s_1^*, s_2^*, \ldots, s_l^*\}$, where $S \subset C$ and $|S| \ll |C|$. Note that all the elements in the above sets are sequences of words. Let θ denote the required summary length, our task is to select the optimal subset of candidate set C that composes summary shorter than θ.

We consider the multi-document summarization task as a data reconstruction problem. We assume that a good multi-document summary is supposed to reconstruct the main content of the document set. Therefore we focus on two issues: (1) how to represent the main content of the document set, and (2) how to use the summary set to reconstruct the main content. In this work, both issues are resolved by document modeling.

As an example, we randomly choose four document sets and their corresponding human-written summaries in DUC2006 dataset, compute their vector representation with PV model and project the vectors into the two-dimensional space. As shown in figure 3, each color corresponds to a document set and four

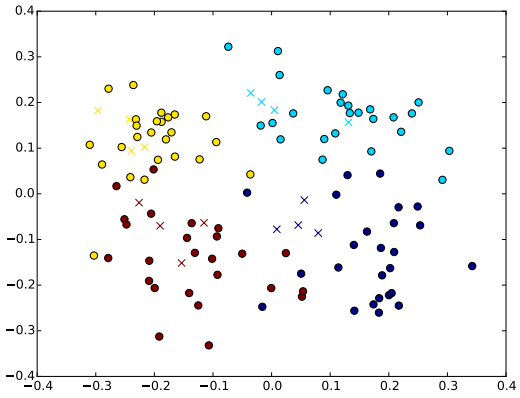


Figure 3: Visualization of document vectors and summary vectors. Documents are represented by circles and summaries are represented by crosses.

summaries, and we find that the centre of summary vectors are close to the centre of document vectors in the same color, implying the effectiveness of averaging.

Hence in our model, each document in the document set is mapped into a vector through document model, then the document vectors are averaged to represent the main content. As for the summary set, all the sentences in S are sequentially concatenated into a sequence S^* as the corresponding summary. Then the summary sequence S^* can be seen as a short document and fed into the document model to reconstruct the main content. Naturally, reconstruction error is applied as objective function and measured by distance between the summary vector and the main content vector. Summary set S is adjusted to minimize the reconstruction error.

Provided that the document modeling process is represented by DM, it takes a document or summay x as input and obtains the semantical vector representation of x, denoted as $DM(x)$. Our reconstruction model is formalized as follows:

$$\begin{aligned} &\min_{S \subset C} \quad \|DM(S^*) - \frac{1}{n}\sum_{i=1}^{n} DM(d_i)\|_2^2 \\ &s.t. \quad len(S^*) \leq \theta \end{aligned} \tag{4}$$

Where S^* denotes the corresponding summary sequence of summary set S and $len(S^*)$ denotes the length of the summary sequence.

Our formulation is similar to the intuition behind He et al. (2012), but differs from it mainly in two aspects: first, we directly reconstruct the original document set on document level; second, we introduce neural document model to represent and reconstruct documents with the summary sentences.

In addition, multi documents are usually considered to bring the redundancy problem in previous works. Contrarily, multiple documents benefit our method by helping represent documents reliably and capture the main content unbiasedly.

3.3 Sentence Selection

Our task is essentially to find the optimal subset of sentences that minimize equation (4) with length constraints, which can be seen as a generalization of knapsack problem and is NP-hard as explained in Lin and Bilmes (2011). The simple approximate approach is to select sentences sequentially from the candidate set with a greedy algorithm. Here we introduce two strategies in sentence selection stage to guarantee both efficiency and effectiveness.

Sentence filtering This strategy aims to narrow the search space by filtering the irrelevant noisy sentences and reserving the promising sentences as candidate. It also benefits the document modeling process by removing noisy sentences with rare words or in bad format. Unsurprisingly, other existing

summarization systems are suitable for this task. In the experiments, we utilize the baseline methods to rank the sentences first and reserve a subset of top-ranked sentences as candidate. Our method then selects summary sentences from the filtered candidate set.

Algorithm 1 BeamSearch

Require: Candidate set C, multi-document set D, document model DM, beam size k, summary length threshold θ

Ensure: A list L_k including top-k summary sets

$L_k, L_{old}, L_{new} \leftarrow \varnothing$
$S \leftarrow \varnothing$ and append S to L_{old}
while L_{old} is not empty **do**
  for each sentence s in C **do**
    for each summary set S in L_{old} **do**
      if $s \notin S$ **then**
        $S_{new} \leftarrow S \cup s$
        if $len(S^*_{new}) < \theta$ **then**
          $\delta \leftarrow \|DM(S^*_{new}) - \frac{1}{n}\sum_{i=1}^{n} DM(d_i)\|_2^2$
          if S_{new} can't further extend **then**
            Update L_k to reserve the top-k final summary sets with loss δ
          else
            Update L_{new} to reserve the top-k promising summary sets with loss δ
          end if
        end if
      end if
    end for
  end for
  $L_{old} \leftarrow L_{new}$
  $L_{new} \leftarrow \varnothing$
end while
return L_k

BeamSearch Algorithm Beamsearch algorithm can be seen as the extension to greedy algorithm, which traverses the entire candidate set C while limiting itself to k potential sentences at each selection step. This is similar to the algorithm used for sentence decoding in neural machine translation tasks (Bahdanau et al., 2014; Sutskever et al., 2014) except that the search space in neural machine translation is the vocabulary of words.

As shown in Algorithm 1, a list L_k is maintained to store top-k final summaries and two lists L_{old}, L_{new} are used to store the top-k promising summaries at each iteration. The algorithm iterates on the candidate set C over and over to select sentences until the required summary length is satisfied. At each iteration, all the sentences in the candidate set are added to the current promising summaries in L_{old} to calculate the reconstruction error. The new summary sets within the length restriction are reserved. The reserved summary sets that have no room for adding new sentences are used to update L_k, while the rest are used to update L_{new}. At last, the top summary set in L_k is chosen and the sentences in it are concatenated sequentially as result.

4 Experiments

In this section, we present the experimental results of our model compared with other baseline approaches and analyze the influence of sentence selection strategies.

4.1 Experimental Setup

Data Set The document understanding conference (DUC)[1] was the main forum providing benchmarks for researchers working on document summarization. We employ DUC 2006 and DUC 2007, the benchmark datasets in multi-document summarization task, for evaluation. DUC 2006 and DUC 2007 contain 50 and 45 document sets respectively. Each document set has 25 news articles for summarization and 4 human-written summaries as ground truth. The length of a result summary is limited to 250 words.

Evaluation Metric We use ROUGE toolkit (Lin, 2004) as our evaluation metric, which is adopted by DUC for automatic summarization evaluation. ROUGE measures summary quality by counting overlapping units such as the n-gram, word sequences and word pairs between the candidate summary (produced by algorithms) and the reference summary (produced by humans). Here we report the average F-measures of ROUGE-1, ROUGE-2 and ROUGE-SU4[2], which are based on uni-gram match, bi-gram match, and unigram plus skip-bigram match with maximum skip distance of 4 between the candidate summary and the reference summary, respectively.

Document Model Training DUC datasets are of small scale, making it hard to train a neural network. For BoW model, we simply use the word vectors pre-trained on GoogleNews[3] to infer the document vectors. For PV model, we employ the Thomson Reuters Text Research Collection (TRC2) in Reuters Corpora (Lewis et al., 2004) to train the PV model first, then fine-tune it on the documents in DUC2006 and DUC2007 datasets to learn more precise representation. The entire training set contains 215 million tokens, 1.3 million word types and 1.8 million documents. The python package gensim[4] is used for training PV model and the parameters are set to default.

Compared Methods Since we focus on unsupervised extractive summarization task in this work, we compare our model *DocRebuild* with several unsupervised extraction-based algorithms. As the same reason as He et al. (2012), we don't compare with supervised methods (Toutanova et al., 2007; Haghighi and Vanderwende, 2009; Çelikyilmaz and Hakkani-Tür, 2010; Lin and Bilmes, 2011) on DUC2006 and DUC2007 here.

1. **Random** randomly selects sentences from the original document set.
2. **Lead** (Wasson, 1998) sorts the documents chronologically and selects the leading sentences one by one.
3. **DSDR** (He et al., 2012) reconstructs all the sentences in the document set by linearly combining summary sentences and selects sentences to minimize reconstruction error with sparse coding.
4. **SpOpt** (Yao et al., 2015) uses a sparse representation model simultaneously selecting sentences and doing sentence compression, subject to the diversity constraint.

Among these methods, Random and Lead are weaker baselines, DSDR is the original reconstruction method, and SpOpt is a state-of-the-art summarization method which considers sentence compression at the same time. Note that we re-implement the above methods[5] to filter sentences and generate candidate sets for our method. Then we construct our reconstruction model on each candidate set. We use term BoW(*) to denote the versions with BoW model and term PV(*) for those with PV model.

4.2 Experimental Result

Overall Performance Table 1 shows the system comparison results on the two datasets. The parameters of *DocRebuild* are set as follows: the dimensionality of document model is 300 for both BoW model and PV model, the candidate sentences are the top 10% sentences[6], and the beamsize is set to

[1] `http://duc.nist.org`

[2] ROUGE version 1.5.5 with options: -a -n 2 -x -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -t 0 -d -l 250

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

[4] `http://radimrehurek.com/gensim/index.html`

[5] Here we used the source code of SpOpt but failed to completely reproduce its results, which may be caused by document preprocessing and parameter setting.

[6] As for Lead, all the leading sentences are reserved as candidate.

	DUC2006			DUC 2007		
Algorithm	**Rouge-1**	**Rouge-2**	**Rouge-SU4**	**Rouge-1**	**Rouge-2**	**Rouge-SU4**
Baselines						
Random	0.34873	0.05447	0.11007	0.36573	0.06346	0.12097
Lead	0.35538	0.06419	0.11672	0.37807	0.07014	0.13111
DSDR	0.36418	0.06100	0.11948	0.38633	0.08052	0.13614
SpOpt	0.40418	0.08388	0.14232	0.41674	0.09905	0.15665
***DocRebuild* (different versions)**						
BoW(over Random)	0.37772	0.06584	0.12421	0.39769	0.07974	0.13704
BoW(over Lead)	0.37489	0.07210	0.12665	0.40086	0.08824	0.14070
BoW(over DSDR)	0.38638	0.07239	0.13013	0.41153	0.09236	0.14836
BoW(over SpOpt)	0.40098	0.07953	0.13868	0.42443	0.09768	0.15548
PV(over Random)	0.38117	0.06812	0.12705	0.41113	0.08733	0.14550
PV(over Lead)	0.37912	0.07632	0.13009	0.41423	0.10377	0.15514
PV(over DSDR)	0.40862	0.08485	0.14453	0.42726	0.10308	0.15810
PV(over SpOpt)	**0.42193**	**0.09314**	**0.15177**	**0.43426**	**0.10500**	**0.16246**

Table 1: Average F-measure performance on DUC2006 and DUC2007.

10 (parameters are tuned in the DUC2005). Among all the systems, Random and Lead unsurprisingly show the poorest performance, for they don't consider any semantic information. DSDR performs better by introducing reconstruction framework. SpOpt improves the performance by employing diversity constraint and doing sentence compression. *DocRebuild* obtains a significant[7] improvement on most of the corresponding baselines. Both PV(over DSDR) and PV(over SpOpt) outperform all the baselines and PV(over SpOpt) achieves the best performance. The result demonstrates the rationality of document-level reconstruction and the effectiveness of neural document model.

Besides, among all the versions of *DocRebuild*, PV model performs much better than BoW model, and the gap is more obvious on DUC2006 than on DUC2007. One possible reason is that PV model takes word order into consideration and this advantage is more apparent in the case of long documents (maximal length of documents is 5407 words in DUC2006 while 2663 words in DUC2007). It also can be observed that Rouge-1 score improves more obviously than Rouge-2 and Rouge-SU4 scores. This implicates our document models are more adept at handling words than n-grams since both document models do the prediction task on word-level.

Algorithm	**Rouge-1**	**Rouge-2**	**Rouge-SU4**
None	0.39048	0.07421	0.13383
+Sentence filtering	0.41576	0.08896	0.14847
+Beamsearch	**0.42193**	**0.09314**	**0.15177**

Table 2: Performance with different strategies on DUC2006.

Analysis on Sentence Selection Strategies We further discuss the separate influence of two sentence selection strategies with PV(over SpOpt) version on DUC2006. As shown in Table 2, None stands for greedily selecting sentences from all the sentences in the document set, sentence filtering and beam-search are added sequentially. We can see that sentence filtering has impressive effect on improving our method. This demonstrates that sentence filtering is necessary for making document model work well as document model may be weak in modeling noisy sentences with rare words or in bad format. In addition, beamsearch further improves the performance by considering more possible combination of sentences. The above results indicate these two strategies both work well.

[7]T-test with p-value ≤ 0.05

5 Conclusion and Future Work

In this paper, we introduced neural document model into multi-document summarization task and proposed a document-level reconstruction framework named *DocRebuild*. In this framework, we represent and reconstruct the main content of documents with summary sentences on neural document model and take the reconstruction error as objective. To obtain the summary, we use sentence filtering to generate a candidate set and select the summary sentences from the candidate set via beamsearch algorithm. The experiment results show that *DocRebuild* outperforms the state-of-the-art unsupervised algorithms and shows great potential in summarizing multiple documents. In future work, it would be of great interests to extend our model by two ways: (1) trying more complex neural networks to model the documents, and (2) designing new algorithm to improve sentence selection.

Acknowledgements

This work is partially supported by the National High Technology Research and Development Program of China (Grant No. 2015AA015403) and the National Natural Science Foundation of China (Grant No. 61170091). We would also like to thank the anonymous reviewers for their helpful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Asli Çelikyilmaz and Dilek Hakkani-Tür. 2010. A hybrid hierarchical model for multi-document summarization. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 815–824.

Günes Erkan and Dragomir R. Radev. 2004. Lexpagerank: Prestige in multi-document text summarization. In *Proceedings of EMNLP*, pages 365–371.

Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, and Jaime Carbonell. 1999. Summarizing text documents: Sentence selection and evaluation metrics. In *Proceedings of ACM SIGIR-1999*, pages 121–128.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, May 31 - June 5, 2009, Boulder, Colorado, USA*, pages 362–370.

Sanda Harabagiu and Finley Lacatusu. 2005. Topic themes for multi-document summarization. In *Proceedings of SIGIR*, pages 202–209.

Hilda Hardy, Nobuyuki Shimizu, Tomek Strzalkowski, Liu Ting, Xinyang Zhang, and G.Bowden Wise. 2002. Cross-document summarization by concept classification. In *Proceedings of SIGIR-02*, pages 121–128.

Zhanying He, Chun Chen, Jiajun Bu, Can Wang, Lijun Zhang, Deng Cai, and Xiaofei He. 2012. Document summarization based on data reconstruction. In *Proceedings of AAAI*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of ICML*, pages 1188–1196.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of ACL*, pages 1106–1115.

Hui Lin and Jeff A. Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of ACL-HLT*, pages 510–520.

Chin-Yew Lin and Eduard H. Hovy. 2002. From single to multi-document summarization. In *Proceedings of ACL*, pages 457–464.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of EMNLP*, page 899907.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL-04 Workshop*, pages 74–81.

He Liu, Hongliang Yu, and Zhi-Hong Deng. 2015. Multi-document summarization based on two-level sparse representation model. In *Proceedings of AAAI*, pages 196–202.

Rada Mihalcea and Paul Tarau. 2005. A language independent algorithm for single and multiple document summarization. In *Proceedings of IJCNLP*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Dragomir R. Radev, Hongyan Jing, Magorzata Sty, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Inf. Process. Manage.*, 40(6):919–938.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, page 31043112.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of EMNLP*, page 14221432.

Kristina Toutanova, Chris Brockett, Michael Gamon, Jagadeesh Jagarlamudi, Hisami Suzuki, and Lucy Vanderwende. 2007. The pythy summarization system: Microsoft research at duc 2007. *Proceedings of DUC-2007*.

Dingding Wang, Tao Li, Shenghuo Zhu, and Chris Ding. 2008. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In *Proceedings of SIGIR*, pages 307–314. ACM.

Mark Wasson. 1998. Using leading text for news summaries: Evaluation results and implications for commercial summarization applications. In *Proceedings of COLING-ACL*, pages 1364–1368.

Jinge Yao, Xiaojun Wan, and Jianguo Xiao. 2015. Compressive document summarization via sparse optimization. In *Proceedings of IJCAI*.

From OpenCCG to AI Planning: Detecting Infeasible Edges in Sentence Generation

Maximilian Schwenger[◇] and **Álvaro Torralba**[◇] and **Jörg Hoffmann**[◇]
David M. Howcroft[*] and **Vera Demberg**[◇]
[◇] Department of Computer Science, Saarland Informatics Campus
[*] Department of Language Science and Technology
Saarland University, Saarbrücken, Germany
schwenger@stud.uni-saarland.de, {torralba,hoffmann}@cs.uni-saarland.de
{howcroft,vera}@coli.uni-saarland.de

Abstract

The search space in grammar-based natural language generation tasks can get very large, which is particularly problematic when generating long utterances or paragraphs. Using surface realization with OpenCCG as an example, we show that we can effectively detect partial solutions (*edges*) which cannot ultimately be part of a complete sentence because of their syntactic category. Formulating the completion of an edge into a sentence as finding a solution path in a large state-transition system, we demonstrate a connection to AI Planning which is concerned with this kind of problem. We design a compilation from OpenCCG into AI Planning allowing the detection of infeasible edges via AI Planning dead-end detection methods (proving the absence of a solution to the compilation). Our experiments show that this can filter out large fractions of infeasible edges in, and thus benefit the performance of, complex realization processes.

1 Introduction

Surface generation is an NP-complete problem (Koller and Striegnitz, 2002). This is particularly problematic in practical terms when the sentence or text paragraph to be generated is long. Our aim in this paper is to improve the efficiency of surface realization in OpenCCG (White, 2006; White and Rajkumar, 2012), by detecting, early on during search, *infeasible* partial solutions (which can never be completed into a full sentence), through a new connection to techniques from *AI Planning*.

OpenCCG is based on *combinatory categorial grammar (CCG)*, where words from a lexicon are annotated with syntactic *categories*, and simple rules (e. g., forward/backward application) dictate category combination. The target of the realization process is a text that conveys the desired meaning, formalized as a conjunction of *elementary predicates*, where each must be covered exactly once. The search space (following a chart realization algorithm, e. g. (Kay, 1996; Cahill and van Genabith, 2006; Carroll and Oepen, 2005)) traverses collections of partial sentences (*edges*). We say that an edge is infeasible if it is not part of any complete sentence. This happens when the edge cannot be combined with other edges to a sentence in a way covering exactly the remaining semantic items. Our contribution is a new technique to automatically identify, and prune, such infeasible edges.

AI Planning (e. g. (Russell and Norvig, 1995; Ghallab et al., 2004)) is one of the oldest sub-areas of Artificial Intelligence (AI). It investigates general problem description languages, and general problem solving algorithms, where a "problem" comes in the form of an "initial state", a "goal", and a set of "actions" that can be applied to change states and thus, eventually, reach the goal. In other words, AI Planning is concerned with models of, and algorithms for, goal reachability testing in large state-transition systems. Connections between sentence generation and AI Planning were previously established for tree-adjoining grammars (Koller and Stone, 2007; Koller and Hoffmann, 2010; Koller and Petrick, 2011), showing how to formulate the entire generation problem as planning. Here we establish a new connection for combinatory categorial grammars, and we focus on the objective of identifying

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1524–1534, Osaka, Japan, December 11-17 2016.

infeasible edges, keeping the overall generation process in the hands of OpenCCG (which is better suited for surface realization as a planning compilation would be agnostic of quality measures such as n-grams).

We observe that edge feasibility in OpenCCG – the ability to complete an edge e_0 into a sentence – can be formulated in terms of a state-transition system. We design a compilation of that ability into an AI Planning task Π, where unsolvability of Π – the absence of a path to the planning goal – implies infeasibility of e_0. Applying dead-end detection algorithms from AI Planning (e. g. (Haslum and Geffner, 2000; Hoffmann and Nebel, 2001; Hoffmann et al., 2014; Steinmetz and Hoffmann, 2016)) to the compiled task Π, and doing so for every edge e_0 during the OpenCCG realization process, then allows the detection and filtering out of infeasible edges. We present two variants of the compilation, which we call optimistic and pessimistic. The optimistic compilation guarantees that every pruned edge is infeasible but it does not perform much pruning in practice. The pessimistic compilation may prune out some solutions. Our experiments show that the pessimistic compilation can filter out large fractions of infeasible edges in, and thus benefit the performance of, complex realization processes.

2 Background and State-Transition System Notation

We briefly introduce background and basic notations for OpenCCG and AI Planning, in a manner geared toward our compilation techniques.

2.1 OpenCCG

Combinatory categorial grammar (CCG) (Steedman, 2000; Steedman and Baldridge, 2011) is a grammar formalism which, in a nutshell, assigns *(syntactic) categories* to words or sequences thereof and provides a set of *combination rules* to combine these. Categories can be either atomic, e. g. noun phrase $\mathbf{NP}$, or complex, e. g. $\mathbf{NP}/\mathbf{N}$, where a slash indicates that the sequence $\mathbf{NP}/\mathbf{N}\ \mathbf{N}$ can be combined, via application of the *forward application* rule, to obtain a noun phrase. A backslash requires the combination partner, in *backward application*, to be on the left hand side. As an example, consider the sentence `Winter is coming`, where `Winter` as proper name has category $\mathbf{NP}$, `is` as verb modifier has category $\mathbf{S}\backslash\mathbf{NP}/(\mathbf{S}\backslash\mathbf{NP})$, and `coming` as intransitive verb has category $\mathbf{S}\backslash\mathbf{NP}$. We can combine `is coming` to acquire $\mathbf{S}\backslash\mathbf{NP}$, and combining that with `Winter` results in a sentence, i. e., in category $\mathbf{S}$. There are also unary rules to enable different combinations by allowing a word sequence to change its own category.

In OpenCCG's realization process, a formula in hybrid logical dependency semantics is flattened, i.e., transformed into a conjunction of elementary predications – the *semantic items* – and transformed into a sentence covering the semantic items. In this process, a *lexicon* provides entries – words associated with categories – potentially useful in terms of their semantics. Composed entries, enriched with additional information, are called *edges* during the search.

Towards our compilation, we next give notations for OpenCCG, and OpenCCG realization, already following AI Planning terminology. In doing so, we will not keep track of the word sequences in edges, and we will not incorporate any notion of word-sequence quality. This is because the purpose of our work merely is to filter out infeasible edges. We specify only those aspects relevant to that purpose.

We refer to the input of the realization process as an *OpenCCG task*, notated $\Omega = (C_0^\Omega, SI^\Omega, R^\Omega, s_I^\Omega, e_G^\Omega)$. Here, C_0^Ω is the finite set of atomic categories c_0. SI^Ω is the finite set of semantic items si. R^Ω is the set of combination rules. We will denote with C^Ω the set of *all* categories, that can be formed from C_0^Ω through applying the rules R^Ω. s^Ω is what we call a *state*, which consists of the set of *edges* already reached in that state. s_I^Ω specifically is the *initial state*. An edge e is a pair (c, σ) where $c \in C^\Omega$ is a category and $\sigma \subseteq SI^\Omega$ is the subset of semantic items *covered* by e (which we will also refer to as the edge's *coverage*). We denote the set of all edges by E^Ω. The initial state $s_I^\Omega \subseteq E^\Omega$ corresponds to the words in the lexicon that are semantically relevant for the sentence. Finally, e_G^Ω is the *goal edge*, defined as $e_G^\Omega = (\mathbf{S}, SI^\Omega)$.

Given this input, OpenCCG realization conducts a search – a chart realization process – over the possible constructions of new edges from previous ones. Each step of the search either applies a unary rule to an edge already reached, i. e., an edge contained in the current state; or applies a combination

rule to a pair of edges $e_1 = (c_1, \sigma_1)$ and $e_2 = (c_2, \sigma_2)$ from the current state, where c_1 and c_2 can be combined, and the truth value assignments have *empty overlap*, $\sigma_1 \cap \sigma_2 = \emptyset$. The resulting new edge is added into the outcome state. The details of how this search process is organized are not relevant to our purpose. Relevant to us are the states and their transitions, which we notate as Ω's *OpenCCG state space*, $\Theta^\Omega = (S^\Omega, T^\Omega, s_I^\Omega, S_G^\Omega)$. Here, $S^\Omega = \mathcal{P}(E^\Omega)$ is the set of all possible states (i.e., the powerset of all possible edges); $T^\Omega \subseteq S^\Omega \times S^\Omega$ contains the transitions over states, as just explained; s_I^Ω is the initial state; and $S_G^\Omega := \{s_G^\Omega \in S^\Omega \mid e_G^\Omega \in s_G^\Omega\}$, the *goal states*, are those containing e_G^Ω. We say that a state s^Ω is *reachable* in Θ^Ω if there is a transition path from s_I^Ω to s^Ω in Θ^Ω. The reachable states in Θ^Ω correspond to the OpenCCG search space. We say that Ω is *solvable* if Θ^Ω contains a reachable goal state.

Formulating our example above in this manner, say the semantic items SI^Ω are $\{\text{Winter}, \text{be}, \text{come}\}$. Say the lexicon contains exactly the three words needed, so that the initial state s_I^Ω contains the edges $(\mathbf{NP}, \{\text{Winter}\})$, $(\mathbf{S}\backslash\mathbf{NP}/(\mathbf{S}\backslash\mathbf{NP}), \{\text{be}\})$, and $(\mathbf{S}\backslash\mathbf{NP}, \{\text{come}\})$. A solution to Θ^Ω then is the path $s_I^\Omega \rightarrow s_1^\Omega \rightarrow s_2^\Omega$ where $s_1^\Omega = s_I^\Omega \cup \{(\mathbf{S}\backslash\mathbf{NP}, \{\text{be}, \text{come}\})\}$ and $s_2^\Omega = s_1^\Omega \cup \{(\mathbf{S}, \{\text{Winter}, \text{be}, \text{come}\})\}$.

We say that an edge e_0 is *feasible* in an OpenCCG task Ω iff, in the OpenCCG state space Θ^Ω, there is a reachable goal state $s_G^\Omega \in S_G^\Omega$ *containing a derived tree* T_0 for e_G^Ω where e_0 appears in T_0. Here, the derived tree T for an edge e is a tree combining edges to get from elements of s_I^Ω to e; and a state s^Ω contains T if all edges in T are elements of s^Ω. In other words, e_0 is feasible if it forms part of a derived tree for a complete sentence. Otherwise, e_0 is *infeasible*.

2.2 AI Planning

Many different variants of AI Planning problem variants have been devised. Here, we consider *STRIPS Planning* (Fikes and Nilsson, 1971), over Boolean variables ("facts"), extended with so-called *conditional effects* (Pednault, 1989). This planning variant is motivated by its wide-spread support in modern planning techniques, and by its match with the needs of our desired OpenCCG compilation.

A *planning task* is a tuple $\Pi = (F^\Pi, A^\Pi, s_I^\Pi, G^\Pi)$. Here, F^Π is a finite set of *facts*; $s_I^\Pi \subseteq F^\Pi$ is the *initial state* (the facts initially true); and G^Π is the *goal* (the facts we need to be true at the end). A^Π is a finite set of *actions*. Each action $a \in A^\Pi$ is a tuple $(pre_a, add_a, del_a, \mathit{CEff}_a)$ where $pre_a \subseteq F^\Pi$ is the action's *precondition*, $add_a \subseteq F^\Pi$ is the action's *add list*, $del_a \subseteq F^\Pi$ is the action's *delete list*, and CEff_a is the action's finite set of *conditional effects*. Each $e \in \mathit{CEff}_a$ is a triple (con_e, add_e, del_e) of fact sets, namely the effect's *condition*, *add list*, and *delete list* respectively.

Given a planning task Π, the task's *state space* is a tuple $\Theta^\Pi = (S^\Pi, T^\Pi, s_I^\Pi, S_G^\Pi)$. Here, $S^\Pi = \mathcal{P}(F^\Pi)$ is the set of all possible states, i. e., fact subsets interpreted as those facts currently true; s_I^Π is Π's initial state; and $S_G^\Pi := \{s_G \in S^\Pi \mid G^\Pi \subseteq s_G\}$ are the *goal states*, where Π's goal is true. The state transitions $T^\Pi \subseteq S^\Pi \times S^\Pi$ arise from action applications. Action a is *applicable* in state s if $pre_a \subseteq s$; in that case, the outcome state is defined as $s' := (s \cup add_a \cup \bigcup_{e \in \mathit{CEff}_a : con_a \subseteq s} add_e) \setminus (del_a \cup \bigcup_{e \in \mathit{CEff}_a : con_a \subseteq s} del_e)$. In other words, s' results from s by including the add lists of the action plus those effects whose condition holds in s, and afterwards removes the delete lists of the action and those effects. We say that Π is *solvable* if Θ^Π contains a reachable goal state.

3 Partial Compilation of OpenCCG Sentence Generation into AI Planning

There is a correspondence between AI Planning and OpenCCG realization – at the level of category combination rules and semantic item coverage – in that both require reaching a goal, from an initial state, in a transition system described in terms of actions/transition rules. We aim to exploit this connection, via a *compilation* from OpenCCG into AI Planning, for automatic filtering of infeasible edges.

Our compilation is *partial* in that it does not attempt to preserve OpenCCG edge reachability exactly. The compilation makes approximations – losing information – aimed at practical viability. It consists of (1) a finite approximation of the set of reachable categories; (2) a planning task capturing solvability of Ω, modulo approximation (1) plus an approximation of semantic coverage; and (3) a modified planning task capturing edge feasibility. We introduce these constructions in this order.

3.1 Finite Approximations of Reachable Categories

In CCG, combination rules specify how to create new categories from old ones. It is possible in principle to simulate this behavior in terms of AI Planning actions, designed to emulate the behavior of CCG combination rules. But this yields large and complex planning encodings, and it is not clear how to exploit those effectively. Therefore, we take a different approach here, pre-compiling all combined (non-atomic) categories that will be considered by the planning process. Our starting point is what we call the *category space*, capturing all possible categories and compositions:

Definition 1. *Let $\Omega = (C_0^\Omega, SI^\Omega, R^\Omega, s_I^\Omega, e_G^\Omega)$ be an OpenCCG task. The* category space *of Ω is the pair $(C^\Omega, \gamma^\Omega)$ where $\gamma^\Omega : C^\Omega \times C^\Omega \cup C^\Omega \mapsto \mathcal{P}(C^\Omega)$ is the partial function where $c' \in \gamma^\Omega(c)$ iff c can be transformed into c' using a unary rule from R^Ω, and $c' \in \gamma^\Omega(c_1, c_2)$ iff c_1 and c_2 can be combined into c' using a binary rule from R^Ω.*

Note that γ^Ω is a function onto subsets of possible outcome categories, rather than onto a unique outcome category, as several different rules may be applicable to the same input categories. Note further that, in the presence of unary rules (like type raising) which are always applicable in CCG, the category space is infinite. To compile it into a finite planning task, we need to restrict ourselves to a finite sub-space. We do so via a size-bound parameter k, in an optimistic vs. a pessimistic manner:

Definition 2. *Let $\Omega = (C_0^\Omega, SI^\Omega, R^\Omega, s_I^\Omega, e_G^\Omega)$ be an OpenCCG task. Let k be a natural number. For $c \in C^\Omega$, let the* degree *of c, denoted $\#(c)$, be the overall number of slashes and backslashes in c. By $C^\Omega[k] := \{c \in C^\Omega \mid \#(c) \leq k\} \cup \{*\}$, we denote the set of all categories whose degree is at most k, plus the* wildcard *symbol $*$. Two category spaces are defined as follows:*

(i) The pessimistic category space *of Ω given k is the pair $(C^\Omega[k], \gamma^{-\Omega})$ where $\gamma^{-\Omega}$ is defined like γ^Ω but replacing any category c' where $\#(c') > k$ by $*$.*

(ii) The optimistic category space *of Ω given k is the pair $(C^\Omega[k], \gamma^{+\Omega})$ where $\gamma^{+\Omega}$ is defined like γ^Ω but replacing any category c' where $\#(c') > k$ by $*$; and including $c' \in \gamma^\Omega(*)$ whenever $\gamma^\Omega(c) = c'$; and including $c' \in \gamma^\Omega(c_1, *)$ whenever $\gamma^\Omega(c_1, c_2) = c'$; and including $c' \in \gamma^\Omega(*, c_2)$ whenever $\gamma^\Omega(c_1, c_2) = c'$.*

In other words, we cut off the generation of categories once their degree exceeds a user-defined threshold k. In the pessimistic (under-approximating) variant, no further combinations are possible behind $*$. In the optimistic (over-approximating) variant, all combinations are possible behind $*$.[1]

For illustration, say in our example the lexicon contains only the words $(\mathbf{S}\backslash\mathbf{NP}/(\mathbf{S}\backslash\mathbf{NP}), \{\text{be}\})$ and $(\mathbf{S}\backslash\mathbf{NP}, \{\text{come}\})$. If we set $k := 3$, then $\gamma^{+\Omega}$ preserves γ^Ω sufficiently to determine that $\mathbf{S}$ cannot be reached from the initial state categories. For $k := 2$, however, $\mathbf{S}\backslash\mathbf{NP}/(\mathbf{S}\backslash\mathbf{NP})$ is replaced by $*$, and we can reach $\mathbf{S}$ by "pretending" that $*$ stands for $\mathbf{NP}$.

The optimistic approximation variant preserves solutions and can be used to provide guarantees, i. e., using our edge-feasibility compilation below, to prune only edges that are indeed infeasible. The pessimistic variant does not provide that guarantee, but tends to be more successful in practice as we will show in Section 5. Observe that the approximations approach γ^Ω from opposite sides, in the sense that they are coarsest for $k = 1$, and become more precise as k grows, $\gamma^{+\Omega}$ getting less optimistic and $\gamma^{-\Omega}$ getting less pessimistic. The approximations converge to γ^Ω in that, for any finite sub-space of $(C^\Omega, \gamma^\Omega)$, there is a k so that both approximations are exact. In terms of edge pruning, this means that the optimistic variant prunes more for larger k, and eventually is precise enough to find any edge that can be pruned; while the pessimistic variant prunes less for larger k, and eventually is precise enough to preserve any edge that cannot be pruned (in particular: precise enough to preserve any one solution).

[1] One can (and our implementation does) define $\gamma^{+\Omega}$ in a more fine-grained manner, replacing only the *sub*-categories behind the threshold k with $*$, and accordingly being less generous in the over-approximation of γ. As this refined version is cumbersome to spell out formally, and leads to similar results in practice, we omit this here.

3.2 Planning Compilation for Solvability

To capture solvability relative to the optimistic/pessimistic finite category space approximation, our compiled planning task combines facts keeping track of category creation with facts keeping track of semantic coverage. Here again we face a design choice: we could, in principle, keep track of the actual category/coverage pairs, i. e., of edges. This would allow us to check for empty overlap when combining two edges. However, that would (a) again yield rather large planning encodings, and (b) require the AI Planning dead-end detection method to be able to reason about delete lists. The most canonical dead-end detection method, that we employ here, does not qualify for (b), so we can just as well circumvent (a). We do so by abstracting from edges, associating a category c with a semantic item si if *at least one* reached edge has category c and covers si. We compile this into a planning task as follows:

Definition 3. *Let $\Omega = (C_0^\Omega, SI^\Omega, R^\Omega, s_I^\Omega, e_G^\Omega)$ be an OpenCCG task. Let k be a natural number. The* optimistic solvability-compilation *is the planning task $\Pi^{+\Omega}[k] = (F^\Pi, A^\Pi, s_I^\Pi, G^\Pi)$ where:*

(i) $F^\Pi = \{c \mid c \in C^\Omega[k]\} \cup \{c[si] \mid c \in C^\Omega[k], si \in SI^\Omega\}$.

(ii) $s_I^\Pi = \{c \mid e = (c, \sigma) \in s_I^\Omega\} \cup \{c[si] \mid e = (c, \sigma) \in s_I^\Omega, si \in \sigma\}$.

(iii) $G^\Pi = \{\mathbf{S}\} \cup \{\mathbf{S}[si] \mid si \in SI^\Omega\}$.

(iv) $A^\Pi = \{a[c, c'] \mid \gamma^{+\Omega}(c) = c'\} \cup \{a[c_1, c_2, c'] \mid \gamma^{+\Omega}(c_1, c_2) = c'\}$, *where:*

(a) $a[c, c'] := (\{c\}, \{c'\}, \emptyset, \{(\{c[si]\}, \{c'[si]\}, \emptyset) \mid si \in SI^\Omega\})$.

(b) $a[c_1, c_2, c'] := (\{c_1, c_2\}, \{c'\}, \emptyset, \{(\{c_j[si]\}, \{c'[si]\}, \emptyset) \mid j \in \{1, 2\}, si \in SI^\Omega\})$.

The pessimistic solvability-compilation *is the planning task $\Pi^{-\Omega}[k]$, defined like $\Pi^{+\Omega}[k]$ but using $\gamma^{-\Omega}$.*

Items (i)–(iii) should be easy to understand: in the compiled planning task, facts c indicate whether category c has been reached yet, and facts $c[si]$ indicate whether c covers si yet; in the initial state, these flags are set according to s_I^Ω, i. e., according to the words in the lexicon; the goal is to have a sentence covering the entire semantics. To understand item (iv), recall that actions have the form $(pre_a, add_a, del_a, CEff_a)$. In item (iv a), encoding unary rule applications $\gamma^{+\Omega}(c) = c'$, the precondition is $\{c\}$ and the (unconditional) add list is $\{c'\}$, effectively saying that, if c is already reached, then applying the action (the rule) yields c'. The conditional effects simply transfer, for each si, the coverage from c (if already reached) to c'. The encoding of binary rules in item (iv b) is similar.

Note that, as indicated above, the delete lists in the compilation are empty. On the one hand, this corresponds to the monotonic nature of the OpenCCG search space, where new edges are being added without removing the old ones. On the other hand, delete effects *would* be needed to capture empty coverage overlap in combination-rule applications. For example, consider that our example of Figure 1 where we have two edges with categories $\mathbf{NP}$. In case 1a, "Winter comes" and "It comes" are valid solutions. However, in case 1b it is impossible to convey all the semantics because it is not possible to use both "Winter" and "Summer" as required (assuming that there are no "and" connectives in our lexicon). Nevertheless, both cases are mapped into the same planning instance since there are edges $\mathbf{NP}$ with semantics 011. Yet, as explained, in the present approach we forsake that information as our dead-end detector would not be able to handle it anyhow.

Theorem 1. *Let Ω be an OpenCCG task. Let k be a natural number. If Ω is solvable, then so is $\Pi^{+\Omega}[k]$.*

Proof. The proof compares Ω's state space $\Theta^\Omega = (S^\Omega, T^\Omega, s_I^\Omega, S_G^\Omega)$ with that of $\Pi^{+\Omega}[k]$. Denote the latter by $\Theta = (S, T, s_I, S_G)$. Define the mapping $\alpha : S^\Omega \mapsto S$ as $\alpha(s^\Omega) := \{c \mid e = (c, \sigma) \in s^\Omega\} \cup \{c[si] \mid e = (c, \sigma) \in s^\Omega, si \in \sigma\}$. As $\alpha^{+\Omega}$ over-approximates the category combinations in α^Ω, it is easy to see that transitions are preserved by α, i. e., whenever $(s_1^\Omega, s_2^\Omega) \in T^\Omega$, we have $(\alpha(s_1^\Omega), \alpha(s_2^\Omega)) \in T$. Furthermore, goal states are preserved, i. e., whenever $s^\Omega \in S_G^\Omega$, we have $\alpha(s^\Omega) \in S_G$. Finally, $\alpha(s_I^\Omega) = s_I$. The claim follows. □

Given Theorem 1, if $\Pi^{+\Omega}[k]$ is *not* solvable, i. e., if an AI Planning dead-end detector is able to detect that this is so, then we can safely conclude that Ω is not solvable either. The pessimistic compilation $\Pi^{-\Omega}[k]$ does not give that guarantee.

String	Category	Semantics
"Winter"	**NP**	011
"It"	**NP**	011
"comes"	**S\NP**	100

(a) Case 1: Feasible

String	Category	Semantics
"Winter"	**NP**	010
"Summer"	**NP**	001
"comes"	**S\NP**	100

(b) Case 2: Unfeasible

Figure 1: Example of two cases that are compiled into the same planning task.

For illustration, consider again the example variant where the lexicon contains only the words $(\mathbf{S}\backslash\mathbf{NP}/(\mathbf{S}\backslash\mathbf{NP}), \{\text{be}\})$ and $(\mathbf{S}\backslash\mathbf{NP}, \{\text{come}\})$, so Ω is unsolvable. Then $\Pi^{+\Omega}[3]$ is unsolvable as $\mathbf{S}$ cannot be reached using $\gamma^{+\Omega}$, cf. above. $\Pi^{+\Omega}[2]$ also is unsolvable, because the semantic item "Winter" cannot be covered; but if we remove that semantic item from the OpenCCG task (and thus from $\Pi^{+\Omega}[2]$), then $\Pi^{+\Omega}[2]$ has a one-step solution combining the two initial-state categories.

3.3 Planning Compilation for Edge Feasibility

The above compilation provides a necessary criterion for an OpenCCG task to be solvable. However, our actual purpose requires a necessary criterion for an OpenCCG *edge* e_0 to be *feasible*, i. e., to form part of a solution. This can be achieved by a simple modification of the compilation, propagating markers to make sure that e_0's category is used in the solution:[2]

Definition 4. *Let $\Omega = (C_0^\Omega, SI^\Omega, R^\Omega, s_I^\Omega, e_G^\Omega)$ be an OpenCCG task, and let $e_0 = (c_0, \sigma_0)$ be an edge in Ω. Let k be a natural number. The* optimistic feasibility-compilation *is the planning task $\Pi^{+\Omega}[k, e_0] = (F^\Pi, A^\Pi, s_I^\Pi, G^\Pi)$ where:*

(i) $F^\Pi = \{c, c[0] \mid c \in C^\Omega[k]\} \cup \{c[si] \mid c \in C^\Omega[k], si \in SI^\Omega\}$.

(ii) $s_I^\Pi = \{c_0[0]\} \cup \{c \mid e = (c, \sigma) \in s_I^\Omega, \sigma \cap \sigma_0 = \emptyset\} \cup \{c[si] \mid e = (c, \sigma) \in s_I^\Omega, \sigma \cap \sigma_0 = \emptyset, si \in \sigma\}$.

(iii) $G^\Pi = \{\mathbf{S}, \mathbf{S}[0]\} \cup \{\mathbf{S}[si] \mid si \in SI^\Omega\}$.

(iv) $A^\Pi = \{a[c, c'] \mid \gamma^{+\Omega}(c) = c'\} \cup \{a[c_1, c_2, c'] \mid \gamma^{+\Omega}(c_1, c_2) = c'\}$, *where:*

(a) $a[c, c'] := (\{c\}, \{c'\}, \emptyset, \{(\{c[0]\}, \{c'[0]\}, \emptyset)\} \cup \{(\{c[si]\}, \{c'[si]\}, \emptyset) \mid si \in SI^\Omega\})$.

(b) $a[c_1, c_2, c'] := (\{c_1, c_2\}, \{c'\}, \emptyset, \{(\{c_1[0]\}, \{c'[0]\}, \emptyset), (\{c_2[0]\}, \{c'[0]\}, \emptyset)\} \cup$
$\{(\{c_j[si]\}, \{c'[si]\}, \emptyset) \mid j \in \{1, 2\}, si \in SI^\Omega\})$.

The pessimistic feasibility-compilation *is $\Pi^{-\Omega}[k, e_0]$, defined like $\Pi^{+\Omega}[k, e_0]$ but using $\gamma^{-\Omega}$.*

Relative to Definition 3, we add the $c[0]$ markers to keep track of whether an ancestor of c uses e_0's category. The initial state includes this marker only for e_0's own category c_0, the goal is for $\mathbf{S}$ to be marked. The actions propagate the markers through conditional effects, marking the outcome category c' if at least one of the input categories is already marked. The additions "$\sigma \cap \sigma_0 = \emptyset$" in (ii) introduce a limited form of empty coverage overlap reasoning, excluding in the initial state those edges whose semantics overlaps with e_0 (and that thus won't be used in a solution incorporating e_0).

Theorem 2. *Let Ω be an OpenCCG task, and let e_0 be an edge in Ω. Let k be a natural number. If e_0 is feasible in Ω, then $\Pi^{+\Omega}[k, e_0]$ is solvable.*

Proof. Say that e_0 is feasible in Ω. Then there is a solution θ to Ω using e_0, and not using any $e \in s_I^\Omega$ whose semantics overlaps with that of e_0. By Theorem 1, $\Pi^{+\Omega}[k]$ is solvable, via a transition path π corresponding to θ. By construction, π is a solution for $\Pi^{+\Omega}[k, e_0]$. □

Given Theorem 2, if an AI Planning dead-end detector proves $\Pi^{+\Omega}[k, e_0]$ to be unsolvable, then we can conclude that e_0 is infeasible. The pessimistic compilation $\Pi^{-\Omega}[k, e_0]$ does not give that guarantee.

For illustration, say in our example the lexicon contains the words $e_1 = (\mathbf{NP}, \{\text{Winter}\})$, $e_2 = (\mathbf{S}\backslash\mathbf{NP}/(\mathbf{S}\backslash\mathbf{NP}), \{\text{be}\})$, and $e_3 = (\mathbf{S}\backslash\mathbf{NP}, \{\text{come}\})$ as before, but contains also the transitive form

[2] In this definition, the modifications relative to Definition 3 are shown in red for the benefit of on-screen reading.

of "coming", $e_4 = ((\mathbf{S}\backslash\mathbf{NP})/\mathbf{NP}, \{\text{come}\})$, infeasible for our purposes. Consider the edge $e_0 = (\mathbf{S}\backslash\mathbf{NP}, \{\text{Winter}, \text{come}\})$, where we combined e_1 with e_4. Consider the compilation $\Pi^{+\Omega}[3, e_0]$: In the initial state, as e_1 overlaps e_0, e_1 is not included. But without **NP**, **S** is unreachable given $\gamma^{+\Omega}$ for $k = 3$, so $\Pi^{+\Omega}[3, e_0]$ is unsolvable and we correctly detect that e_0 is infeasible.[3]

4 Practical Compilation Use and Optimizations

Our idea is to create, and check the solvability of, the compiled planning task $\Pi^{+\Omega}[k, e_0]$ respectively $\Pi^{-\Omega}[k, e_0]$, every time a new edge e_0 is created during the OpenCCG realization process. If the compiled planning task is unsolvable, e_0 is deemed infeasible, and is discarded. This filtering method is provably sound when using $\Pi^{+\Omega}[k, e_0]$. When using $\Pi^{-\Omega}[k, e_0]$, it is a practical heuristic, and converges to sound pruning – eventually preserving the best solution – as k grows.

To realize this approach, we require a method for checking solvability of planning tasks. In general, however, deciding solvability ("plan existence") is **PSPACE**-complete (Bylander, 1994). For fast solvability detection, planning research therefore concentrates on polynomial-time solvable fragments of the plan existence problem. The most wide-spread such fragment is the one where all delete lists are required to be empty (e. g. (Bylander, 1994; Haslum and Geffner, 2000; Hoffmann and Nebel, 2001)). Hence the design of our compilation, which incorporates approximations resulting in empty delete lists. For planning tasks with empty delete lists, plan existence can be decided in time low-order polynomial in the size of the task, using so-called *relaxed planning graphs* (Hoffmann and Nebel, 2001).

Though polynomial time, testing delete-free plan existence does incur a runtime overhead, especially in our context where we need to do so for every edge during realization. Efficient implementation is therefore important. One key to this is the re-use of information/computation shared across individual tests. First, every call to sentence realization based on the same lexicon shares the same category space. Hence we can build the category space approximation, $(C^{\Omega}[k], \gamma^{+\Omega})$ respectively $(C^{\Omega}[k], \gamma^{-\Omega})$, *offline*, just once for the lexicon at hand, prior to realization. Second, the feasibility compilations for individual edges e_0 during the same realization process are identical except for their initial states. So, during a realization process, we create a compiled task just once and adapt it minimally for each test.

Finally, (a) the action set in $\Pi^{+\Omega}[k, e_0]$ respectively $\Pi^{-\Omega}[k, e_0]$ is fully determined by $\gamma^{+\Omega}$ respectively $\gamma^{-\Omega}$ along with the set of semantic items SI^{Ω}; while (b) for the maintenance of semantic coverage and $c[0]$ markers, instead of the compilation via conditional effects as specified, one can implement a simple special-case handling in the standard relaxed planning graph solvability test. Taking these two observations together, we can generate the action set completely offline. Online, prior to a realization process, we merely need to read in the actions and setup the marker-maintenance data structures.

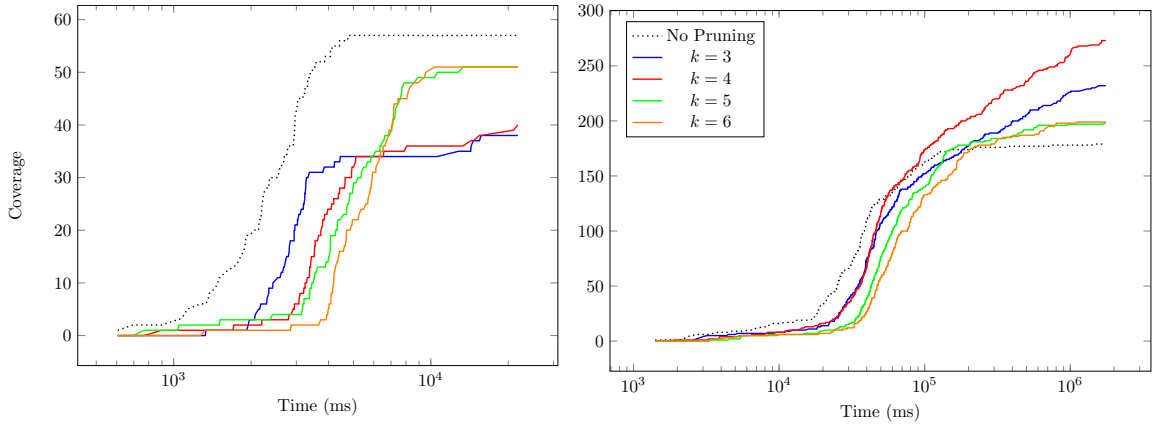
5 Experiments

As our main test base for experimentation, we used the SPaRKy Restaurant Corpus (we also ran preliminary experiments with some other test bases, which we get back to below). SPaRKy is one of the only published resources for NLG which provides intermediate representations in addition to system inputs and outputs and quality ratings for those outputs. Originally introduced by Walker et al. (2007), Nakatsu and White (2010) developed a CCG grammar for this dataset which spans both the sentence and the discourse levels. The domain of the corpus is restaurant descriptions, including prices, kind of food, decor, service, etc. In this work we use the a set of 431 test instances developed for the contrast-enhanced version of the grammar presented in Howcroft, Nakatsu, & White (2013). The lexicon in this testbed includes 193 words and the grammar is capable of producing a wide variety of texts of varying lengths. Of the 431 OpenCCG realization tasks, 61 *recommendation tasks* require generating a text recommending a single restaurant, while 370 *comparison tasks* require generating a text comparing two or more restaurants with each other. As these instances correspond to the generation of entire text paragraphs, they are complex enough to be interesting use cases for our techniques. This pertains in particular to

[3]Note that the same is not true for $k = 2$; and neither for e_4 because, there, ignoring overlap in rule applications means that e_1 could be used twice. These are weaknesses of our current approach, which could potentially be tackled by more informed compilations. We get back to this in the conclusion.

the comparison tasks, where the required text is longer (an average of 60 words with respect to 38 for recommendation tasks).

In preliminary tests with the optimistic approximation variant, the pruning was too weak to pay off, i. e., too few edges were pruned to get a benefit. We therefore concentrate here on pruning with the pessimistic approximation variant, where small values of k may prune too aggressively, while large values of k yield more reliable pruning yet incur a larger runtime overhead. All experiments were run on a cluster of machines with Intel Xeon E5-2660 processors running at 2.2 GHz. The runtime/memory limit was set to 30 minutes/4 GB for each sentence generation task, i. e., for each benchmark instance.



(a) Recommendation tasks (b) Comparison tasks

Figure 2: Coverage, i. e., the number of sentence generation tasks to which a solution was found, as a function of runtime on SPaRKy (a) recommendation tasks and (b) comparison tasks. A data point (x, y) means that y tasks are solved within a time limit of x milliseconds. "No pruning" is the baseline OpenCCG search without pruning; "$k = n$" considers our pessimistic pruning with parameter k, i. e., the $\Pi^{-\Omega}[k, e_0]$ compilation, on every edge e_0 during search, pruning e_0 if $\Pi^{-\Omega}[k, e_0]$ is unsolvable.

As a simple measure of performance, we focus on the runtime spent by the OpenCCG chart realization process until the first solution – the first edge of category **S** covering all semantic items – is generated. Figure 2 shows coverage, i. e., the number of benchmark instances where a solution was found, as a function of runtime. We distinguish between (a) recommendation vs. (b) comparison tasks as these are different in nature and yield very different performance profiles.

Regarding (a), we see that these tasks are essentially too easy for our pruning method to pay off: the runtime overhead of repeatedly checking the solvability of $\Pi^{-\Omega}[k, e_0]$ outweighs the gain from pruning, so that fewer tasks are solved within the same runtime limits. The restaurant comparison tasks (b), however, are more challenging (notice the different x-axis scales in (a) and (b)), and the picture is different: in the much larger search spaces, the pruning impact is stronger. For runtime limits > 56 seconds, the coverage of $k = 4$ pruning exceeds that without pruning. For runtime limits > 206 seconds, all settings of k exceed the baseline. Note here that the value of k controls the trade-off between accuracy (better with large k) and runtime overhead (better with small k). In SPaRKy restaurant comparison tasks, $k = 4$ is the sweet spot of that trade off. At our maximum time limit of $x = 30$ minutes, $k = 4$ pruning increases coverage from 179 instances without pruning, to 273 with pruning, an increase of 52%.

As the pessimistic compilation does not guarantee that pruned edges are actually infeasible, the pruning may adversely affect the quality of the sentences generated. As k gets larger, this danger decreases as the pruning becomes more accurate. In SPaRKy, it turns out that $k = 4$ is not only best in terms of runtime performance, but is also enough to avoid any deterioration in sentence quality. Of the 215 instances solved by both the baseline and $k = 4$ (across recommendation and comparison tasks), in 137 cases the two realizations are identical. In the remainining 78 cases, the realizations differ only in using the word "just" vs. the word "only", so that the version with pruning does exactly as well as the baseline.

Going beyond solutions, there also are cases where OpenCCG produces a *partial solution*, an edge of

category S that covers only a subset of the semantic items. This can still be useful if, e. g., four instead of five restaurants are being compared. Our $k = 4$ pruning has clear advantages in terms of the ability to find such partial solutions. Of the 97 cases where neither $k = 4$ nor the baseline find a complete solution, $k = 4$ provides a partial solution in 81 cases, the baseline in 52 cases. In all 21 cases where only the baseline finds a complete solution, $k = 4$ finds a partial solution. In contrast, of the 98 cases where only $k = 4$ finds a complete solution, in 59 cases the baseline does not manage to find a partial solution.

We also ran experiments on some other test bases (Vancoppenolle et al., 2011; Racioppa, 2011; Kruijff et al., 2010), yet as the text paragraphs to be generated were comparatively small, similarly to SPaRKy recommendation tasks our pruning methods generally did not pay off. Conversely, in the CCGBank (Hockenmaier and Steedman, 2007), our category space approximations consumed excessive amounts of memory. For practical viability in such large bases, either additional implementation tricks, or more intelligent abstractions (not just enumerating all categories up to a fixed degree), would be required.

6 Related Work

There are a number of approaches in the literature to speed-up sentence realization. The closest approach is *polarity filtering* for generation with Tree-Adjoining Grammar (TAG), proposed by Gardent and Kow (2005). The polarity filter avoids the combination of those sets of elementary trees that cannot possibly lead to a full solution, by making sure that the numbers and types of substitution nodes and elementary trees fit together. Compared to the approach of compiling into planning in general, this differs in that it specifies a concrete dead-end detector, rather than a connector to another area offering a rich set of potential such detectors. Compared to our particular compilation here, polarity-based filtering uses a different kind of abstraction, of CCG categories as input/output signatures, instead of the finite category space approximation we make based on bounded degree.

Within CCG-based surface realization approaches, there are several suggestions to reduce the size of the search space:

- *Chunking* (White, 2006) divides the semantic input into subproblems that are solved independently and then combined into a full sentence. This approach is different to ours since we remove edges that are deemed as infeasible due to the syntactic structure of the grammar, while *chunking* exploits the structure of the semantics in order to simplify the problem.

- *Hypertagging* (Espinosa et al., 2008) uses text-body statistics in order to select the relevant words from the lexicon and assign them a syntactic category. This may filter out irrelevant edges but it does not reason over category/semantic combinations.

- *Glue rules* (White, 2011) can be activated whenever the search fails to find a grammatically correct sentence, in order to relax the constraints imposed by the grammars. In our case, we try to speed-up search to be able to find correct sentences according to the grammar rules provided.

Besides OpenCCG, other surface realization approaches have been successfully used to for broad-coverage language generation. These include TAG-based structure-driven surface realization (Narayan and Gardent, 2012), head-driven phrase structure grammar approaches (Carroll and Oepen, 2005), and statistical approaches trained on the surface realization shared task data (Bohnet et al., 2010). Some of these approaches, in which certain grammar rules must be followed, may also benefit from detecting infeasible partial solutions. Hence, adapting our planning compilation to other grammar formalisms is an interesting line for future research.

Previous connections between sentence generation and AI Planning were previously established for tree-adjoining grammars (Koller and Stone, 2007; Koller and Hoffmann, 2010; Koller and Petrick, 2011). Our compilation, apart from starting from CCG rather than TAG, has a quite different purpose and properties. While the TAG-PDDL is an equivalent compilation whose size is "the same as" that of the TAG input, whereas here we have an over/under-approximation whose size grows exponentially in category space with the our size limit k.

7 Conclusion

Sentence generation as search relates deeply to AI Planning in that, at least as far as grammatical and semantic correctness is concerned, it is essentially a reachability problem in a large discrete transition system. This connection has been made before, and we herein propose a new variant and application, detecting infeasible edges in OpenCCG. Our empirical results show promise, though much remains to be done. In our view, the most interesting question is *how much information we can efficiently capture and exploit* in this kind of compilation. Our present approach is (a) inflexible in precomputing a category space approximation, and (b) conservative in targeting delete-free planning which is easy to handle. Both design choices sacrifice information, and both may be lifted through more intelligent compilations/abstractions, paired with more advanced dead-end detection on the AI Planning side.

From a broader point of view, we believe that planning/search techniques can be an important part of the quest for practical sentence generation with complex optimization objectives (Demberg et al., 2016).

Acknowledgements

This work was partially supported by the DFG excellence cluster EXC 284 "Multimodal Computing and Interaction" and the DFG collaborative research center SFB 1102 "Information Density and Linguistic Encoding".

References

Bernd Bohnet, Leo Wanner, Simon Mille, and Alicia Burga. 2010. Broad coverage multilingual deep sentence generation with a stochastic multi-level realizer. pages 98–106. Chinese Information Processing Society of China.

Tom Bylander. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1–2):165–204.

Aoife Cahill and Josef van Genabith. 2006. Robust pcfg-based generation using automatically acquired LFG approximations. In Nicoletta Calzolari, Claire Cardie, and Pierre Isabelle, editors, *Proceedings of the 21st International Conference on Computational Linguistics (ACL'06)*. ACL.

John A. Carroll and Stephan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. pages 165–176. Springer.

Vera Demberg, Jörg Hoffmann, David Howcroft, Dietrich Klakow, and Álvaro Torralba. 2016. Search challenges in natural language generation with complex optimization objectives. *KI – Künstliche Intelligenz*, 30:63–69.

Dominic Espinosa, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. pages 183–191.

Richard E. Fikes and Nils Nilsson. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

Claire Gardent and Eric Kow. 2005. Generating and selecting grammatical paraphrases. In *Proceedings of the 10th European Workshop on Natural Language Generation (ENLG'05)*, pages 49–57.

Malik Ghallab, Dana Nau, and Paolo Traverso. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

Patrik Haslum and Hector Geffner. 2000. Admissible heuristics for optimal planning. In S. Chien, R. Kambhampati, and C. Knoblock, editors, *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems (AIPS'00)*, pages 140–149, Breckenridge, CO. AAAI Press, Menlo Park.

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396, September.

Jörg Hoffmann and Bernhard Nebel. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.

Jörg Hoffmann, Peter Kissmann, and Álvaro Torralba. 2014. "Distance"? Who Cares? Tailoring merge-and-shrink heuristics to detect unsolvability. In Thorsten Schaub, editor, *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14)*, Prague, Czech Republic, August. IOS Press.

David Howcroft, Crystal Nakatsu, and Michael White. 2013. Enhancing the expression of contrast in the sparky restaurant corpus. In *Proceedings of the 14th European Workshop on Natural Language Generation (ENLG'13)*, pages 30–39.

Martin Kay. 1996. Chart generation. In Aravind K. Joshi and Martha Palmer, editors, *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 200–204. Morgan Kaufmann / ACL.

Alexander Koller and Jörg Hoffmann. 2010. Waking up a sleeping rabbit: On natural-language sentence generation with ff. In Ronen I. Brafman, Hector Geffner, Jörg Hoffmann, and Henry A. Kautz, editors, *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*. AAAI Press.

Alexander Koller and Ronald Petrick. 2011. Experiences with planning for natural language generation. *Computational Intelligence*, 27(1):23–40.

Alexander Koller and Matthew Stone. 2007. Sentence generation as planning. In *Proc. of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)*.

Alexander Koller and Kristina Striegnitz. 2002. Generation as dependency parsing. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 17–24. Association for Computational Linguistics.

Geert-Jan M Kruijff, Pierre Lison, Trevor Benjamin, Henrik Jacobsson, Hendrik Zender, Ivana Kruijff-Korbayová, and Nick Hawes. 2010. Situated dialogue processing for human-robot interaction. In *Cognitive Systems*, pages 311–364. Springer.

Crystal Nakatsu and Michael White. 2010. Generating with discourse combinatory categorial grammar. *Linguistic Issues in Language Technology*, 4(1).

Shashi Narayan and Claire Gardent. 2012. Structure-driven lexicalist generation. pages 2027–2042. Indian Institute of Technology Bombay.

Edwin P.D. Pednault. 1989. ADL: Exploring the middle ground between STRIPS and the situation calculus. In R. Brachman, H. J. Levesque, and R. Reiter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 1st International Conference (KR-89)*, pages 324–331, Toronto, ON, May. Morgan Kaufmann.

Stefania Racioppa. 2011. Italian ccg grammar for aliz-e. Technical report, DFKI.

Stuart Russell and Peter Norvig. 1995. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ.

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. *Non-Transformational Syntax: Formal and Explicit Models of Grammar. Wiley-Blackwell*.

Mark Steedman. 2000. *The syntactic process*, volume 24. MIT Press.

Marcel Steinmetz and Jörg Hoffmann. 2016. Towards clause-learning state space search: Learning to recognize dead-ends. In Dale Schuurmans and Michael Wellman, editors, *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*. AAAI Press, February.

Jean Vancoppenolle, Eric Tabbert, Gerlof Bouma, and Manfred Stede. 2011. A german grammar for generation in openccg. Number 96, pages 145–150.

Marilyn A. Walker, Amanda Stent, François Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research*, 30:413–456.

Michael White and Rajakrishnan Rajkumar. 2012. Minimal dependency length in realization ranking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 244–255.

Michael White. 2006. Efficient realization of coordinate structures in combinatory categorial grammar. *Research on Language and Computation*, 4(1):39–75.

Michael White. 2011. Glue rules for robust chart realization. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 194–199.

The Next Step for Multi-Document Summarization: A Heterogeneous Multi-Genre Corpus Built with a Novel Construction Approach

Markus Zopf, Maxime Peyrard and Judith Eckle-Kohler
Research Training Group AIPHES / Knowledge Engineering Group / UKP Lab
Department of Computer Science, Technische Universität Darmstadt
Hochschulstraße 10, 64289 Darmstadt, Germany
www.aiphes.tu-darmstadt.de, www.ukp.tu-darmstadt.de

Abstract

Research in multi-document summarization has focused on newswire corpora since the early beginnings. However, the newswire genre provides genre-specific features such as sentence position which are easy to exploit in summarization systems. Such easy to exploit genre-specific features are available in other genres as well. We therefore present the new ***h***MDS corpus for multi-document summarization, which contains heterogeneous source documents from multiple text genres, as well as summaries with different lengths. For the construction of the corpus, we developed a novel construction approach which is suited to build large and heterogeneous summarization corpora with little effort. The method reverses the usual process of writing summaries for given source documents: it combines already available summaries with appropriate source documents. In a detailed analysis, we show that our new corpus is significantly different from the homogeneous corpora commonly used, and that it is heterogeneous along several dimensions. Our experimental evaluation using well-known state-of-the-art summarization systems shows that our corpus poses new challenges in the field of multi-document summarization. Last but not least, we make our corpus publicly available to the research community at the corpus web page https://github.com/AIPHES/hMDS.

1 Introduction

Multi-document summarization (MDS) is the task of creating a summary from a topically related document collection. Existing corpora for the evaluation of MDS systems, most notably from the Document Understanding Conference (DUC) (Over et al., 2007) and from the Text Analysis Conference (TAC),[1] cover mostly MDS of news documents (Nenkova et al., 2011). While research on MDS has also considered genres other than newswire (e.g., opinionated blog posts in TAC 2008 or biomedical research papers in TAC 2014), MDS has almost exclusively focused on *homogeneous* document collections that belong to the same genre.

However, this homogeneous nature of the existing MDS benchmark corpora does not reflect application scenarios where topically related documents from different genres need to be summarized. An exemplary scenario of increasing importance is MDS on the web, where a user retrieves multiple online documents for a particular topic (cf. Rosner and Camilleri (2008), Nenkova and McKeown (2011)). These online documents may comprise news articles, blog posts, encyclopedic texts, or even scientific articles.

A related issue is the very high effort needed to create a new MDS corpus. Summarizing a set of documents is a demanding task for humans and requires expert abstractors, e.g., information analysts as in the DUC competitions. Attempts to obtain human-written summaries via crowdsourcing have failed (Lloret et al., 2013). When the set of documents to be summarized grows, the summarization task even becomes infeasible for humans.

We address these gaps and present (i) a large heterogeneous MDS corpus in English as a new challenging benchmark for summarization systems, and (ii) a novel approach for constructing such a corpus

[1] http://www.nist.gov/tac

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1535–1545, Osaka, Japan, December 11-17 2016.

at a large scale. Our methodology combines summaries from Wikipedia featured articles with human-retrieved source documents and can thus be generally applied in all languages where Wikipedias exist. Our detailed analysis and evaluation of the new MDS corpus shows that MDS from heterogeneous genres poses a new challenge and calls for future research.

2 Related Work

Related to our work are methods to create MDS corpora, i.e., pairs of topically related documents and reference summaries. In this section, we focus on the particular aspect of obtaining summaries of a given set of documents. We summarize the traditional approaches used in DUC and TAC, and previous work on alternative approaches.

Approach in DUC and TAC How have multi-document summaries been written in DUC and TAC? For example, the 2005 DUC Summary-Writing Task states that a human should first read the topic and all the 25 - 50 documents in the topic cluster. While reading the documents, information relevant for the topic should be highlighted and then used in a second step to write a 250-word summary of the documents. As observed in early DUC competitions, the agreement between different reference summaries for the task of generic summarization was low (cf. Harman and Over (2004)). Therefore, DUC 2005 offered 10 reference summaries for a subset of the 50 topics in order to cover a more representative sample of the diverging reference summaries.

Another approach to reduce the high variation between reference summaries was taken in TAC 2010, where the summarization task was made more specific and accompanied by guidelines describing a list of required aspects to be covered, e.g., summaries in the category accidents should cover aspects such as what, when, where why (Owczarzak and Dang, 2011).

Alternative Approaches Having trained humans, e.g., professional abstractors, write multidocument summaries is expensive, and for laymen, the task is highly demanding and time-consuming. Therefore, a recent project on the construction of a MDS corpus for European Portuguese took a semi-automatic approach where the documents within a topic were first filtered for relevance using a summarization system and then processed by human annotators who only had to remove sentences until the summary length was reached (Almeida et al., 2014). However, this approach is problematic in our context, because existing summarization systems have not been developed and evaluated on heterogeneous text types.

Crowdsourcing is another approach to reduce the effort for creating annotated data and it has been increasingly used in NLP for a range of different tasks. However, for multidocument summaries, it has been shown to lead to poor results (Lloret et al., 2013).

Exploiting user generated content on the web as a source of reference summaries is another option. For example, Aker and Gaizauskas (2010) used content available on the VirtualTourist platform to construct reference summaries for the task of summarizing location-related images. Even more appealing are collaboratively edited platforms such as Wikipedia where the content is refined and consolidated over time through a well-documented discussion and revision process. Recently, Wikipedia featured articles[2] have been used to create an evaluation dataset for the task of multilingual single-document summarization (Giannakopoulos et al., 2015). In Wikipedia featured articles, the first paragraph constitutes a summary of the article; hence Giannakopoulos et al. (2015) used it as a reference summary for the body of the Wikipedia article.

While Giannakopoulos et al. (2015) address the issue in DUC and TAC datasets of topic bias towards news-specific content, they consider only single-document summarization and sources from a single text type (i.e., encyclopedic text). In contrast to the DUC and TAC datasets, their multilingual SDS dataset provides only a single reference summary for each source document. Although the use of featured articles ensures a certain quality level of the summaries, there might be important aspects of the topic which are missing, in particular for topics that are discussed differently in different languages and cultures. For example, Filatova (2009) suggests to combine the Wikipedia articles across multiple languages for a specific topic in order to obtain a comprehensive summary.

[2] https://en.wikipedia.org/wiki/Wikipedia:Featured_articles

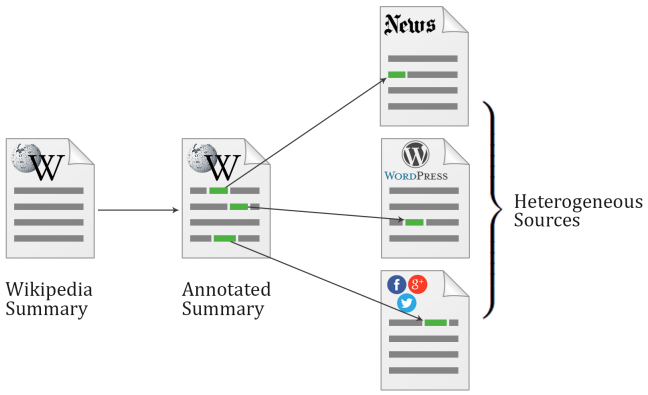


Figure 1: Illustration of the novel corpus construction approach. Left: already available Wikipedia summary; middle: Wikipedia lead with annotated information nuggets; right: a set of heterogeneous source documents which contain the information nuggets.

Our new corpus construction method addresses the bottleneck of human summary creation and the limitation of Giannakopoulos et al. (2015) to be restricted to the single-document summarization task.

3 Reversing Corpus Construction

Previous approaches for constructing MDS corpora start with specifying topics and collecting topic-related source documents; in a second step humans write a summary of the source documents for each of the topics. This approach has several disadvantages, in particular, the reading effort (a human has to read all source documents), the writing effort (the summary has to be written), and the subjectivity of the resulting summary (since the summary is written by only one human). Furthermore, good topics and suitable source documents that cover the topics have to be found in the first place, which can be a laborious and expensive task.

To address these issues, we propose to build MDS corpora by reversing the usual process: Instead of writing a summary for previously gathered source documents, we propose to use an already available high-quality summary and just search for suitable source documents which contain information about the topic. This simplifies the corpus construction process and reduces the effort to create pairs of source documents and summaries. The process is illustrated in Figure 1. In the rest of this section, we describe our novel methodology in detail.

3.1 Methodology

We propose to use the first section of Wikipedia featured articles (the so called *lead*) for this purpose. According to the Wikipedia featured article criteria,[3] these articles are (i) well-written, (ii) comprehensive, (iii) well-researched, (iv) neutral, and (v) stable. In particular, the lead of a featured article is supposed to summarize the topic (according to the guidelines). Since Wikipedia articles are written by many authors, we can consider the lead of a featured article as the consensus of many people regarding the important information about a particular topic. We can therefore consider the leads as high-quality summaries which are representative, because they combine contributions from many authors.

It is important to emphasize that our approach can easily be transferred to other languages where the respective Wikipedias contain featured articles, or even to completely different sources of already existing high-quality summaries.

Given a particular topic (i.e., Wikipedia article), the corpus construction process consists of two steps. First, we annotate topic-specific information nuggets in the summary, in order to be able to retrieve source documents in a systematic way. The information nuggets are supposed to represent atomic information units, i.e., omitting words would change the meaning of the nugget significantly. In the second

[3] https://en.wikipedia.org/wiki/Wikipedia:Featured_article_criteria

step, we search for heterogeneous source documents given the information nuggets. For each annotated information nugget, we search for one source document, which contains the information represented by the nugget, and we keep the text genre as document metadata (the set of possible text genres has been defined in advance, see section 3.2 for details). During our search for source documents, we try to find sources from as many text genres as possible, since we aim to build a heterogeneous corpus. Furthermore, we collect only source documents which are not too similar to the summary. This means that the sources should (i) not contain all information nuggets about the topic, and (ii) contain information which is irrelevant to the topic.

The first property enforces a *content distribution* across all source documents, which means that a summarization system has to consider all source documents to create a proper summary. This is different from previous corpora consisting of documents from a single text type (e.g., newswire), where each individual source document already provided most of the relevant information. In particular, this property reduces the redundancy in the source documents which is often used by multi-document summarization systems to detect important information. Consider as an example homogeneous newswire corpora where documents on a given topic typically have a high degree of redundancy due to the way journalists write news articles: in a typical hard news report, the reported news is first summarized in the lead section, and then specified and elaborated on in the body of the article (Thomson et al., 2008), which creates redundancy even in a single article. As a further consequence of the low redundancy in our collected source documents, the heuristics that the most frequent information is also the most important information (making frequency usually a strong feature in newswire MDS corpora (Nenkova et al., 2006)) is weakened.

The second property, the *presence of unimportant information*, is also different from prior corpora, where the sentences provide usually at least some information about the topic. Selecting the right sentences becomes therefore even more important in order to achieve a good scoring.

Our corpus construction approach is valid, since it is reasonable to assume that the retrieved documents do not introduce new important information about the topic. Wikipedia featured articles already contain the most important information related to a topic, and are furthermore continuously updated by the Wikipedia community. We therefore retrieve the version timestamp of each article and store it in the corpus metadata; the manual retrieval of sources for this version of the Wikipedia article was performed within a short time frame.

3.2 Corpus Construction

We applied our new corpus construction methodology to build ***h***MDS, a new, heterogeneous, multi-genre corpus for MDS. In this section, we describe details of the actual corpus construction process.

Since Wikipedia provides a large number of featured articles, we first selected a subset of the articles for our project. Based on the featured article overview page,[4] we selected the three broad domains

- *Art, Architecture, and Archaeology* (D1),
- *History* (D2), and
- *Law, Politics, and Government* (D3).

We asked three annotators to perform the steps described in the previous section. As part of the first step, they should tag and extract roughly 10 to 20 information nuggets from the lead of each Wikipedia article. For the topic *California Gold Rush*, examples of extracted nuggets are *1848-1855*, *period in American history*, *Sutter's mill*, *gold was found by James W. Marshall*, and *300,000 gold-seekers*.

In the second step, the annotators searched for source documents as described above, using well-known web search engines. Since they tried to find the nugget text verbatim in the source documents, ROUGE scoring is assumed to perform well in our corpus, which is confirmed in our experiments in section 5. The retrieved documents were archived in the Wayback Machine.[5] We provide more detailed

[4] `https://en.wikipedia.org/wiki/Wikipedia:Featured_articles`
[5] `http://archive.org/web/`

corpus information as well as the annotator guidelines and download scripts to retrieve the documents at the corpus web page `https://github.com/AIPHES/hMDS`. For the retrieval step, we used a predefined set of 10 text genres, which are shown in Table 1 along with short descriptions.

Since the source documents are web pages and not ready-to-use raw text documents, we asked the annotators to extract and store the relevant part of the web page, which leads to rather compact source documents. We also generate two larger versions of the corpus by performing boilerplate content removal with Boilerpipe (Kohlschütter et al., 2010). A second version contains all visible web page content. We use the shorthand notation ***h***MDS-M, ***h***MDS-A, and ***h***MDS-V to denote the manually extracted, the automatically extracted and the version with all the visible content, respectively. We also provide a version of the corpus where sentence splitting has already been applied, because most extractive summarizing systems extract sentences. This improves the reproducibility of summarization experiments since it removes one noisy preprocessing step. We use version 1.7.0 of the Stanford segmenter in the DKPro Core software (Eckart de Castilho and Gurevych, 2014) to produce the sentence segmentation.

Although the main purpose of our corpus is MDS, it can also be used for various other tasks. Since we store the genre of each source document, it can be used for text genre classification, or as dataset for training and evaluating boilerplate removal systems. Research in automatic source document retrieval can also use our new corpus.

4 Corpus Analysis

In the following, we summarize and analyze the result of our corpus construction effort. In particular, we analyze whether our corpus actually is as heterogeneous as we designed it to be via our corpus construction methodology. For this, we also compared the ***h***MDS corpus with two common MDS corpora, namely DUC2004[6] and TAC2008A.[7]

The ***h***MDS corpus contains 91 summary-source documents pairs (topics). In total, the annotators retrieved 1,265 source documents. We obtained 13.90 ± 3.09 (where $\pm$ indicates the standard deviation) source documents per topic in comparison to DUC2004 and TAC2008A which both have exactly 10 source documents per topic (i.e. 10 ± 0).

4.1 Text Genres

Since we asked the annotators to classify each source document according to the text genre it belongs to, we are able to provide a detailed analysis of the distribution of texts genres in our corpus. Table 1 provides an overview of the text genres which are present in our corpus, as well as their distributions.

		across domains			
Text Genre – Description (Example)	Count	D1	D2	D3	Avg. length (in words)
article – well-written text (high-quality blog post, news article)	524	0.37	0.42	0.47	1452.70 ± 1751.28
forum post – lack text structure (QA site, Youtube comment)	115	0.10	0.08	0.09	964.10 ± 1726.89
microblog – short, contains abbreviations (Twitter)	33	0.03	0.03	0.02	53.61 ± 14.44
organization – announcement, press release (any org./company)	99	0.11	0.06	0.06	749.29 ± 1119.21
encyclopedic short – encyc. source (Urban Dictionary, IMDB)	115	0.12	0.07	0.08	400.45 ± 362.88
encyclopedic long – encyc. source (Wikipedia)	137	0.11	0.14	0.07	3434.15 ± 5077.32
social media – post in social network (Facebook, Google+)	11	0.01	0.01	0.01	270.45 ± 250.67
scientific – contain citations and bibliography	119	0.07	0.08	0.14	5394.03 ± 9118.11
education – text book, tutorial	79	0.05	0.09	0.05	1568.76 ± 3020.62
dialogues – opinionated (interview, transcript, discussion)	33	0.02	0.03	0.03	3759.79 ± 4897.97
Total	1265	0.36	0.35	0.28	1863.59 ± 3928.91

Table 1: List of genres present in the ***h***MDS corpus along with their fractions in the different domains. The length details are computed for the M-version of our corpus.

We obtained a large amount of source documents for the "article" genre. The distribution of the other genres is rather uniform with most documents belonging to the encyclopedic, scientific, and forum post categories. The fraction of microblog documents, dialogues, and social media is considerably smaller.

[6] `http://duc.nist.gov/duc2004`
[7] `https://tac.nist.gov/2008`

On average, we obtained 5.39 $\pm$ 1.54 different genres per topic with a minimum of 3 and a maximum of 9 different genres per topic. These results substantiate our claim that the ***h***MDS corpus contains sources from very diverse genres. We also observe variations of the distributions of text genres across the three different domains.

4.2 Length of Source Documents and Summaries

Another property which we analyze is the length of both source documents and summaries. Table 1 provides information about the distribution of lengths across the different genres. We see that we obtained a wide variety of different lengths across the genres. Since Table 1 only provides information for ***h***MDS-M, we provide more details of the source document lengths in Table 2 where we can see that the variation of lengths increases strongly in the versions A and V of the ***h***MDS corpus. Compared to DUC2004 and TAC2008A, we obtained much longer source documents, as well as a much higher variance in length.

Corpus	Avg. length (in words)	Relative std
hMDS-M	1863.59 $\pm$ 3928.91	2.11
hMDS-A	2192.53 $\pm$ 8196.75	3.74
hMDS-V	2973.06 $\pm$ 8429.32	2.84
DUC2004	672.14 $\pm$ 506.32	0.75
TAC2008A	589.20 $\pm$ 480.33	0.82

Table 2: Length comparisons of source documents.

Corpus	Avg. length (in words)	Relative std
hMDS	245.55 $\pm$ 132.94	0.54
DUC2004	118.11 $\pm$ 6.38	0.05
TAC2008A	109.33 $\pm$ 7.01	0.06

Table 3: Length comparisons of summaries.

Regarding the summaries, we achieved a large difference to prior work as well, see Table 3. Our summaries are on average about twice as long as the summaries in DUC2004 and TAC2008A. The major difference, however, is the huge variance of lengths in our corpus which can be observed by both standard- and relative standard deviation. The minimum length of a summary in our corpus equals 72 words and the maximum length equals 657 words.

4.3 Textual Heterogeneity

The heterogeneity of our corpus also results from other textual properties. Heterogeneous documents are expected to use different wording and to have some topics shifts. In order to measure this textual heterogeneity, we use information theoretic metrics on word probability distributions.

In our experiments, we use the Jensen-Shannon (JS) divergence, which is a symmetric and always defined version of the Kullback Leibler (KL) divergence (Kullback and Leibler, 1951). It incorporates the idea that the distance between two distributions cannot be very different from the average of distances from their mean distribution. Its expression is $JS(P\|Q) = \frac{1}{2}KL(P\|A)+\frac{1}{2}KL(Q\|A)$ where $A = \frac{P+Q}{2}$ is the mean distribution of P and Q. Based on the JS divergence, we can define a measure of textual heterogeneity TH for a topic T composed of documents $d_1, \cdots, d_n$ as

$$TH_{JS}(T) = \frac{1}{n}\sum_{d_i \in T} JS(P_{d_i}, P_{T\setminus d_i}) \quad (1)$$

where P_{d_i} is the probability distribution of words in document d_i and $P_{T\setminus d_i}$ is the probability distribution of words in all other documents of the topic except d_i. TH_{JS} is the average divergence of documents with all the others and provides therefore a measure of diversity among documents of a given topic.

	DUC2004	TAC2008A	***h***MDS-M	***h***MDS-A	***h***MDS-V
TH_{JS}	0.3019	0.3188	0.3815	0.3358	0.3252

Table 4: Average TH_{JS} scores of classical corpora and our new datasets.

Table 4 shows the results of the analysis according to the TH metric. DUC2004 and TAC2008A have similar source documents in comparison to ***h***MDS-M according to TH. ***h***MDS-A and ***h***MDS-V are closer to the classical datasets than ***h***MDS-M. This is due to the fact that these versions contain much more boilerplate content compared to ***h***MDS-M, which makes them more similar again. Nevertheless, the heterogeneity of the main content in the source documents is verified by TH.

4.4 Distribution of Content

	ROUGE-1 Recall				ROUGE-2 Recall			
Corpus	full	n-1	n/2	1	full	n-1	n/2	1
hMDS-M	0.68	0.67 ± 0.03	0.63 ± 0.06	0.42 ± 0.12	0.48	0.46 ± 0.04	0.40 ± 0.07	0.17 ± 0.09
DUC2004	0.43	0.43 ± 0.01	0.43 ± 0.02	0.36 ± 0.05	0.16	0.15 ± 0.01	0.15 ± 0.01	0.09 ± 0.03
TAC2008A	0.46	0.46 ± 0.02	0.44 ± 0.02	0.35 ± 0.05	0.20	0.19 ± 0.01	0.17 ± 0.02	0.10 ± 0.03

Table 5: Results of the content distribution experiment. We present results of using all, n-1, n/2, and only 1 source document. The omitted documents were selected randomly.

As mentioned in section 3.1, our corpus construction aims for a distribution of content across all source documents. To evaluate this property, we conduct an experiment in which we use different fractions of the source documents as input for a greedy optimal summarization systems. Since we use a greedy optimal summarizer due to performance reasons, the optimal results differ slightly from the results in Table 6. We observe in Table 5 that the omission of one document has already an effect in our corpus, whereas the optimal performance stays nearly constant in DUC2004 and TAC2008A. The effect increases when only half of the source documents is considered. A rather large difference can be observed when only one source document is available to the summarizer. In DUC2004 and TAC2008A, the summarizer is still able to achieve 83.7% and 76.1% of the optimal score, whereas in our corpus, only 61.8% can be achieved according to ROUGE-1. For ROUGE-2, we observe 56.3% and 50.0% in DUC2004 and TAC2008A, compared to 35.4% in our corpus.

As we did not manually annotate the information nuggets in the source documents (due to the high annotation effort), we can investigate the differences in content distribution only regarding the ROUGE scores, which also considers high-frequent function words in the default setup. This might explain why the variation is not even higher.

4.5 Distribution of ROUGE Scores

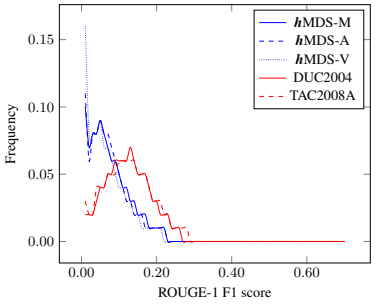


Figure 2: F1 score

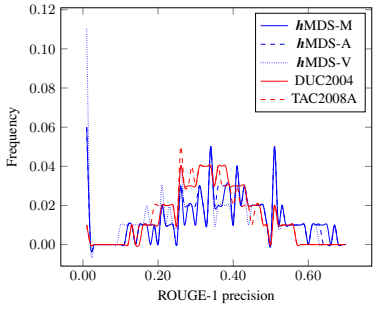


Figure 3: Precision

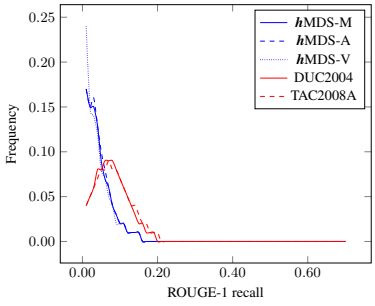


Figure 4: Recall

In this section, we investigate the distribution of ROUGE-1 scores of single sentences in our corpus compared to the DUC2004 and TAC2008A corpora. Figures 2, 3, and 4 provide the distribution of ROUGE-1 F1 measure, ROUGE-1 precision, and ROUGE-1 recall, respectively. The evaluation shows that the distribution according to ROUGE-1 precision is not much different, except for a large number of sentences with very low precision in our corpora. The ROUGE-1 recall curve shows that single sentences in DUC2004 and TAC2008A on average provide a higher recall compared to ***h***MDS. In combination, we see that there are a lot of sentences in ***h***MDS with both very low precision and very low recall. Thus, we can conclude that we indeed constructed a corpus containing sentences which do not contribute much to a good summary (see section 3.1, *presence of unimportant information*).

5 Summarization Experiments

In this section, we conduct experiments with well-known baselines and summarization systems to analyze our new corpus further. As evaluation metric, we apply the commonly used ROUGE scoring (Lin,

2004). All experiments were evaluated with ROUGE version 1.5.5 with standard parameters `-a -m -n 2 -x -c 95 -r 1000 -f A -p 0.5 -t 0 -l` l, which includes stemming without removing stopwords. Since our summaries have different length, we use a variable length parameter l for the evaluation, where l denotes the length of the reference summary. For the DUC2004 and TAC2008A datasets, we use $l = 100$. All results are averaged across all topics.

5.1 Summarization Systems

First, we describe several well-known extractive summarization approaches. By applying them to both the classical datasets and to our new corpus, we can test whether or not we succeeded in creating a new challenge for this research area. Furthermore, we can investigate different properties of our corpus in more detail, such as the strength of the sentence position and centrality features which are used by different summarizers.

Measure	Dataset	Optimal	Random	Lead	LexRank	ICSI	LSA	TF-IDF
ROUGE-1	DUC 2004	0.4535	0.2955	0.3424	0.3450	0.3778	0.2904	0.3318
	TAC 2008A	0.5067	0.2963	0.3315	0.3466	0.3675	0.3154	0.3236
	hMDS-M	0.6992	0.3754	0.4069	0.4192	0.5401	0.3447	0.3671
	hMDS-A	0.6962	0.3242	0.1041	0.4083	0.5370	0.3391	0.3439
	hMDS-V	0.7019	0.2847	0.0050	0.3133	0.5033	0.3228	0.3302
ROUGE-2	DUC 2004	0.1876	0.0435	0.0766	0.0715	0.0900	0.0430	0.0657
	TAC 2008A	0.2540	0.0458	0.0765	0.0773	0.1107	0.0696	0.0572
	hMDS-M	0.4960	0.0732	0.1237	0.1273	0.2293	0.0689	0.0939
	hMDS-A	0.4845	0.0594	0.0318	0.1192	0.2267	0.0652	0.0805
	hMDS-V	0.5018	0.0450	0.0018	0.0797	0.2082	0.0603	0.0766

Table 6: Performance according to ROUGE recall of various summarization approaches for DUC 2004 and our new MDS corpus.

Optimal provides an upper bound for the performance of the summarization systems by searching for the best combination of sentences that achieves the highest ROUGE score. It is no competitive summarization system, since it uses oracle knowledge (i.e. the reference summaries) to generate the best possible summary with an ILP solver. **Random** selects sentences randomly. Although it is a quite simple approach, it helps to compare the absolute ROUGE scores in different datasets. **Lead** is a simple, but quite strong baseline in newswire documents. It iteratively selects the first sentences of the source documents until the desired summary length is reached. Its strength in classical datasets derives from the fact that most important information are usually written first in news articles.

TF-IDF was introduced by Luhn (1958). It uses the typical word frequency assumption as a proxy for importance, which is a very strong feature in multi-document newswire corpora (Nenkova et al., 2006). In **LexRank** (Erkan and Radev, 2004), a similarity graph is constructed with sentences as nodes and the Cosine similarity between them as edge weights. Sentences are scored according to their PageRank score. LexRank relies on centrality to measure relevance. **LSA** performs single value decomposition on a terms-sentences matrix weighted by TF-IDF scores. The summary is made up of the sentences that represent well the most important topics. LSA is a topic-model approach which relies on frequency of co-occuring patterns. **ICSI** (Gillick and Favre, 2009) treats summarization as global linear optimization problem. It extracts a summary by solving a maximum coverage problem considering the most important bi-grams. The importance of bi-grams is estimated via their frequency in the source documents.

In our experiments, we use our own implementations of Optimal, Random, and Lead and implementations of TF-IDF, LexRank, and LSA provided by the sumy package.[8] We use the Python implementation of ICSI released by Boudin et al. (2015).

5.2 Results

The results of the summarization experiment are displayed in Table 6. We first observe that the optimal achievable score is much higher in our corpus, which means that a better fit of source documents and summary is possible. This might be due to the fact that we asked the annotators to search for documents

[8] `https://github.com/miso-belica/sumy`

containing the information nuggets verbatim. Having only one reference summary compared to multiple reference summaries in DUC2004 and TAC2008A might also have an impact. Since we observe different optimal scores, we present in the following relative ROUGE-1 scores according to the respective optimum. For ***h***MDS-V and ***h***MDS-A, we use the optimal score of ***h***MDS-M, which is a lower bound for the true score.

Selecting sentences randomly yields lower results in ***h***MDS compared to DUC2004 and TAC2008A in terms of the maximal possible result (e.g. in DUC2004, selecting sentences randomly yields 65.2% of the optimal score, in ***h***MDS-M we only achieve 54.7%). The lead baseline, which is strong in newswire, does not perform well in ***h***MDS-M (58.2%) and particularly bad in ***h***MDS-A (14.9%) and ***h***MDS-V (0.7%). Sentence position is therefore no longer a good feature in these corpora. ROUGE-2 scores support this result even stronger.

TF-IDF, LSA, LexRank, and ICSI use different approaches to model centrality. We observe that the performance of all approaches decreases when more noise is added to the corpus (versions A and V compared to M). ICSI, a strong state-of-the-art approach (Hong et al., 2014), performs best in our new corpus. We observe, similarly to Random, that it is better suited to summarize DUC2004 (83.3%) than ***h***MDS-M (77.2%), ***h***MDS-A (76.8%), and ***h***MDS-V (72.0%). LSA and TF-IDF have the lowest performance across the four non-baseline systems, and the LexRank results are between them and ICSI.

Our results show that the relative difference to the optimum is quite large in our corpora compared to the classical datasets.

6 Conclusion

In this paper, we presented ***h***MDS: a new, heterogeneous, multi-genre MDS corpus which provides new challenges for summarization systems and is therefore suited to drive research in new directions. To build the corpus, we proposed a novel corpus construction approach which reverses the classically applied approach and reduces the construction effort.

We provide detailed analyses of ***h***MDS which verify that our corpus is inherently different from DUC2004 and TAC2008A in various ways. ***h***MDS contains topics from three broad domains, the reference summaries and source documents have varying lengths, and the source documents belong to different genres, which results in the important information being distributed across all source documents. Thus, a system has to be able to deal with this high degree of heterogeneity in order to generate a proper summary. Furthermore, summaries in the corpus are not only written by one person, but are the consensus of a whole community and therefore provide a representative view of the importance of information. We provide results of several baselines and well-known summarization systems which indicate that our corpus poses new challenges for summarization systems. Last but not least, we make our corpus publicly available to the research community.

In future work, we are going to add another dimension of heterogeneity to the corpus by adding pairs of summary and source documents in German, based on German Wikipedia featured articles. Furthermore, we want to investigate if we can generate a very large corpus based on the proposed construction approach by automatically retrieving source documents instead of manually collecting them. The corpus built manually would then serve as a gold standard.

Acknowledgments

This work has been supported by the German Research Foundation (DFG) as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under grant No. GRK 1994/1, and via the German-Israeli Project Cooperation (DIP, grant No. GU 798/17-1). We thank Kamel Chelly, Matthias Hanreich, and Hannah Wieland for their contributions to the corpus construction.

References

Ahmet Aker and Robert Gaizauskas. 2010. Model Summaries for Location-related Images. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 3119–3124, Valletta, Malta. European Language Resources Association (ELRA).

Miguel B. Almeida, Mariana S. C. Almeida, André F. T. Martins, Helena Figueira, Pedro Mendes, and Cláudia Pinto. 2014. Priberam Compressive Summarization Corpus: A New Multi-Document Summarization Corpus for European Portuguese. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 146–152, Reykjavik, Iceland. European Language Resources Association (ELRA).

Florian Boudin, Hugo Mougard, and Benoît Favre. 2015. Concept-based Summarization using Integer Linear Programming: From Concept Pruning to Multiple Optimal Solutions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1914–1918, Lisbon, Portugal. The Association for Computational Linguistics.

Richard Eckart de Castilho and Iryna Gurevych. 2014. A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Elena Filatova. 2009. Multilingual Wikipedia, Summarization, and Information Trustworthiness. In *Proceedings of the SIGIR 2009 Workshop Information Access in a Multilingual World*, pages 19–24, Boston, Massachusetts USA.

George Giannakopoulos, Jeff Kubina, John Conroy, Josef Steinberger, Benoit Favre, Mijail Kabadjov, Udo Kruschwitz, and Massimo Poesio. 2015. MultiLing 2015: Multilingual Summarization of Single and Multi-Documents, On-line Fora, and Call-center Conversations. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 270–274, Prague, Czech Republic. Association for Computational Linguistics.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, ILP '09, pages 10–18, Boulder, Colorado. Association for Computational Linguistics.

Donna Harman and Paul Over. 2004. The Effects of Human Variation in DUC Summarization Evaluation. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 10–17, Barcelona, Spain.

Kai Hong, John Conroy, benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A Repository of State of the Art and Competitive Baseline Summaries for Generic News Summarization. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1608–1616, Reykjavik, Iceland. European Language Resources Association (ELRA).

Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, pages 441–450, New York City, NY USA.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of summaries. In *Proceedings of ACL workshop on Text Summarization Branches Out*, pages 74–81, Barcelona, Spain.

Elena Lloret, Laura Plaza, and Ahmet Aker. 2013. Analyzing the capabilities of crowdsourcing services for text summarization. *Language Resources and Evaluation*, 47(2):337–369.

Hans Peter Luhn. 1958. The Automatic Creation of Literature Abstracts. *IBM Journal of Research Development*, 2:159–165.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2-3):103–233.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A Compositional Context Sensitive Multi-document Summarizer: Exploring the Factors That Influence Summarization. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 573–580, Seattle, Washington, USA. ACM.

Ani Nenkova, Julia Hirchberg, and Yang Liu, editors. 2011. *Proceedings of the Workshop on Automatic Summarization for Different Genres, Media, and Languages*. Association for Computational Linguistics, Portland, Oregon.

Paul Over, Hoa Dang, and Donna Harman. 2007. DUC in Context. *Information Processing and Management*, 43(6):1506–1520.

Karolina Owczarzak and Hoa Dang. 2011. Who wrote What Where: Analyzing the content of human and automatic summaries. In *Proceedings of the Workshop on Automatic Summarization for Different Genres, Media, and Languages*, pages 25–32, Portland, Oregon. Association for Computational Linguistics.

Mike Rosner and Carl Camilleri. 2008. MultiSum: Query-Based Multi-Document Summarization. In *Proceedings of the workshop Multi-source Multilingual Information Extraction and Summarization at Coling 2008*, pages 25–32, Manchester, UK.

Elizabeth A. Thomson, Peter R. White, and Philip Kitley. 2008. "Objectivity" and "hard news" reporting across cultures: comparing the news report in English, French, Japanese and Indonesian journalism. *Journalism Studies*, 9(2):212–228.

SentiHood: Targeted Aspect Based Sentiment Analysis Dataset for Urban Neighbourhoods

Marzieh Saeidi
University College London
msaeidi@cs.ucl.ac.uk

Guillaume Bouchard
Bloomsbury AI
guillaume@bloomsbury.ai

Maria Liakata
University of Warwick
m.liakata@warwick.ac.uk

Sebastian Riedel
University College London
sriedel@cs.ucl.ac.uk

Abstract

In this paper, we introduce the task of targeted aspect-based sentiment analysis. The goal is to extract *fine-grained* information with respect to entities mentioned in user comments. This work extends both *aspect-based* sentiment analysis that assumes a single entity per document and *targeted* sentiment analysis that assumes a single sentiment towards a target entity. In particular, we identify the sentiment towards each aspect of one or more entities. As a testbed for this task, we introduce the SentiHood dataset, extracted from a question answering (QA) platform where urban neighbourhoods are discussed by users. In this context units of text often mention several aspects of one or more neighbourhoods. This is the first time that a generic social media platform in this case a QA platform, is used for fine-grained opinion mining. Text coming from QA platforms is far less constrained compared to text from review specific platforms which current datasets are based on. We develop several strong baselines, relying on logistic regression and state-of-the-art recurrent neural networks.

1 Introduction

Sentiment analysis is an important task in natural language processing. It has received not only a lot of interest in academia but also in industry, in particular for identifying customer satisfaction on products and services. Early research in the field (Das and Chen, 2001; Morinaga et al., 2002) of sentiment analysis only focused on identifying the overall sentiment or polarity of a given text. The underlying assumption of this work was that there is one overall polarity in the whole text.

Aspect-based sentiment analysis (ABSA) (Jo and Oh, 2011; Pontiki et al., 2015; Pontiki et al., 2016) relates to the task of extracting fine-grained information by identifying the polarity towards different aspects of an entity in the same unit of text, and recognizing the polarity associated with each aspect separately. The datasets for this task were mostly based on specialized review platforms such as Yelp where it is assumed that only one entity is discussed in one review snippet, but the opinion on multiple aspects can be expressed. This task is particularly useful because a user can assess the aggregated sentiment for each individual aspect of a given product or service and get a more fine-grained understanding of its quality.

Another line of research in this field is *targeted* (a.k.a. target-dependent) sentiment analysis (Jiang et al., 2011; Vo and Zhang, 2015). Targeted sentiment analysis investigates the classification of opinion polarities towards certain target entity mentions in given sentences (often a tweet). For instance in the sentence "People everywhere love Windows & vista. Bill Gates", polarity towards Bill Gates is "Neutral" but the positive sentiment towards Windows & vista will interfere with identifying it if the usual methods for sentiment analysis task are employed. However this task assumes only the *overall* sentiment for each entity. Moreover, the existing corpora for this task so far has contained only a single target entity per unit of text.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1546–1556, Osaka, Japan, December 11-17 2016.

Both settings are obviously limited, and there exists many scenarios in which sentiments towards different aspects of several entities are discussed in the same unit of text. As a running example, we use urban areas: choosing which area to live or to visit is an important task when moving or visiting a new city. Currently there are no dedicated platforms for reviewing and rating aspects of neighbourhoods of a city. However we can find many discussions and threads on several blogs and question answering platforms that discuss aspects of areas in many cities around the world. In general, these conversations are very comprehensible: they often contain specific information about several aspects of several neighbourhoods. One example is the following (area names are highlighted in bold and aspect related terms are underlined):

"*Other places to look at in South London are* ***Streatham*** *(good range of shops and restaurants, maybe a bit far out of central London but you get more for your money)* ***Brixton*** *(good transport links, trendy, can be a bit edgy)* ***Clapham*** *(good transport, good restaurants/pubs, can feel a bit dull, expensive) ...*"

The example above does not perfectly fit into the existing tasks in sentiment analysis mentioned earlier. In this work, we introduce a new task that not only subsumes the existing sub-fields of *targeted* and *aspect-based* sentiment analysis but it also makes less assumptions on the number of entities that can be discussed in the unit of text.

To compare with the existing aspect-based sentiment analysis task, take the following example from the restaurant dataset used by SemEval shared ABSA (Pontiki et al., 2016) task. "*The design of the space is good but the service is horrid!*". The ABSA task aims to identify that a positive sentiment towards the *ambiance* aspect is expressed (opinion target expression is "space"). Moreover, a negative sentiment is expressed towards the *service* aspect (opinion target expression is "service"). In this example, it is assumed that both of these opinions are expressed about a single restaurant which is not mentioned explicitly. However, take the following *synthetic* example that ABSA is not addressing:

"*The design of the space is good in* ***Boqueria*** *but the service is horrid, on the other hand, the staff in* ***Gremio*** *are very friendly and the food is always delicious.*"

In this example, more than one restaurant are discussed and restaurants for which opinions are expressed, are explicitly mentioned. We call these target entities. Current ABSA task can only recognise that positive and negative opinions towards aspect "service" are expressed. But it can not identify the target entity for each of these opinions (i.e. Germio and Boqueria respectively). Targeted aspect-based sentiment analysis handles extracting the target entities as well as different aspects and their relevant sentiments.

In the following, we argue that this task is both very relevant in practice, and raises interesting modelling questions. To facilitate research on this task we introduce the SentiHood dataset. SentiHood is based on the text from a QA platform in the domain of neighbourhoods of a city. Table 2 shows examples of input sentences and annotations provided.

Sentence	**Labels**
The cheap parts of London are **Edmonton** and **Tottenham** and they are all poor, crime ridden and crowded with immigrants	(Edmonton,price,Positive) (Tottenham,price,Positive) (Edmonton,safety,Negative) (Tottenham,safety,Negative)
Hampstead area, more expensive but a better quality of living than in **Tufnell Park**	(Hampstead,price,Negative) (Hampstead,live,Positive)

Table 1: Examples of input sentences and output labels in the system.

Our contributions in this paper can be summarised as follows:

- We introduce the task of *targeted aspect-based* sentiment analysis as a further step towards extracting more fine-grained information from more complex text in the field of sentiment analysis.
- We use the text from social media platforms, in particular QA, for fine-grained opinion mining. So far, all datasets in this field have utilised text from review specific platforms where certain assumptions can be made and data is more constrained and less noisy.
- We propose SentiHood, a benchmark dataset that is annotated for the task of targeted aspect-based sentiment analysis in the domain of urban neighbourhoods.
- We show that despite the fact that the texts in QA were not written with the goal of writing a review in mind, question answering platforms and online forums are in general rich in information.
- We provide strong baselines for the task using both logistic regression and Long Short Term Memory (LSTM) networks and analysis of the results.

2 SentiHood

SentiHood is a dataset for the task of *targeted aspect-based* sentiment analysis. It is based on the text taken from question answering platform of Yahoo! Answers that is filtered for questions relating to neighbourhoods of the city of London. In this section we explain the data collection and annotation process and summarise properties of the dataset.

2.1 Data Collection Process

Entities in the dataset are locations or neighbourhoods. Yahoo! Answers was queried using the name of each neighbourhood of the city of London. Location (entity) names were taken from the gazetteer GeoNames[1] and restricted to those within the boundaries of London. This list includes names of areas and boroughs and therefore entities are not always geographically exclusive (a borough contains several areas or neighbourhoods). The content of each question-answer pairs was aggregated and split into sentences. We keep only sentences that have a mention of a location entity name and discard other sentences.

2.2 Categories

The Number of location mentions in a single sentence in our dataset varies from one to over 50. To simplify the task, we only annotate sentences that contain one or two location mentions. These sentences were divided into two groups: sentences containing one location mention — Single, and sentences containing two location mentions — Multi. This is to observe the difficulty of annotating two groups by human annotators and by the models.

2.3 Aspects

Like existing work in the aspect-based sentiment analysis task (Brychcın et al., 2014), a pre-defined list of aspects is provided for annotators to choose from. These aspects are: *live*, *safety*, *price*, *quiet*, *dining*, *nightlife*, *transit-location*, *touristy*, *shopping*, *green-culture* and *multicultural*. Adding an additional aspect of *misc* was considered. However in the initial round of annotations, we realised that it had a negative effect on the decisiveness of annotators and it led to a lower overall agreement. Aspect *general* refers to a generic opinion about a location, e.g. "I love **Camden Town**".

2.4 Sentiment

For each selected aspect, annotators were required to select a polarity or sentiment. Most work in this area considers three sentiment categories of "Positive". "Negative" and "Neutral". In our annotation however, we only provided "Positive" and "Negative" sentiment labels. This is because in our data we rarely come across cases where aspects are discussed without a polarity.

[1] http://www.geonames.org/

2.5 Target Entity

Target entity is a location entity in which an opinion (aspect and sentiment) is expressed for. We also refer to target entity as target location.

2.6 Out of scope

For the sentences that do not comply with our schema, we define the two following special labels. Sentences marked with one of the these labels are removed from the dataset.

1. **Irrelevant**: When the identified name does not refer to a location entity: for example in the sentence "**Notting Hill** (1999) stars Julia Roberts and Hugh Grant use the characteristic features of the area as a backdrop to the action", "Notting Hill" refers to the movie and not the area.

2. **Uncertain**: When two contradicting sentiments are expressed for the same location and aspect, e.g. "Like any other area, **Camden Town** has good and bad parts". Moreover, when the opinion is expressed for an area without a direct mention in the sentences, e.g. "**It**'s a very trendy area and not too far from **King's Cross**".

2.7 Procedure:

We use the BRAT annotation tool (Stenetorp et al., 2012) to simplify the annotation task. Three annotators were initially selected for the task. None of the annotators are experts in linguistics. Annotators began by reading the guidelines and examples. Each annotator was then required to annotate a small subset of the data. After each round of annotation, agreements between annotators were calculated and discussed and this procedure continued until they reached a reasonable agreement. 10% of the whole dataset was randomly selected and annotated by all the three annotators. The annotator with the highest inter-annotator agreement was selected to annotate all the dataset.

Agreements: Cohen's Kappa coefficient(K) (COHEN, 1960) is often used for measuring the pairwise agreement between each two annotators for the task of aspect-based sentiment analysis (Gamon et al., 2005; Ganu et al., 2009) and other tasks (Liakata et al., 2010). The Kappa Coefficient is calculated over aspect-sentiment pairs per each location. Pairwise inter-annotator agreement for aspect categories measured using Cohen's Kappa is 0.73, 0.78 and 0.70, which is deemed of sufficient quality. It is worth mentioning that agreements on different aspect categories varied, with some aspects having a higher agreement rate. Agreements for aspect expressions are 0.93, 0.94, 0.93. These agreements indicate reasonably high inter-annotator agreements (Pavlopoulos, 2014).

Disagreements: Main disagreements between annotators occurred in detecting the aspect rather than detecting the sentiment, aspect expression or the target location. For instance, some annotators associated the expression "residential area" with a "Positive" sentiment for aspect "quiet" or "live" and others did not agree that "residential" implies quietness or desirable for living. In the case of disagreements, the vote of the majority was considered as the correct annotation.

Some ambiguity was also observed with respect to detecting the target location. This occurred mainly when a location is confined in another location. For instance the sentence "**Angel** in **Inslington** has many great restaurants for eating out" expresses a "Positive" sentiment for the aspect "dining" of area **Angel** which is within the borough of **Islington**. Some annotators suggested that the sentence also implies the same opinion for **Islington**. However at the end all annotators agreed that in such cases no implicit assumptions should be made and only confined area should be labeled.

2.8 Dataset

SentiHood currently contains annotated sentences containing one or two location entity mentions.[2] SentiHood contains 5215 sentences with 3862 sentences containing a single location and 1353 sentences containing multiple (two) locations. Figure 1 shows the number of sentences that are labeled with each aspect, breaking down on the sentiment "Positive" or "Negative". "Positive" sentiment is dominant for

[2]SentiHood data can be obtained at `http://annotate-neighborhood.com/download/download.html`

aspects such as dining and shopping. This shows that for some aspects, people usually talk about areas that are good for it as oppose to areas that are not. The *general* aspect is the most frequent aspect with over 2000 sentences while aspect *touristy* has occurred in less than 100 sentences. Notice that since each sentence can contain one or more opinions, the total number of opinions (5920) in the dataset is higher than the number of sentences.

Location entity names are masked by **location1** and **location2** in the whole dataset, so the task does not involve identification and segmentation of the named entities. We also provide the dataset with the original location entity names.

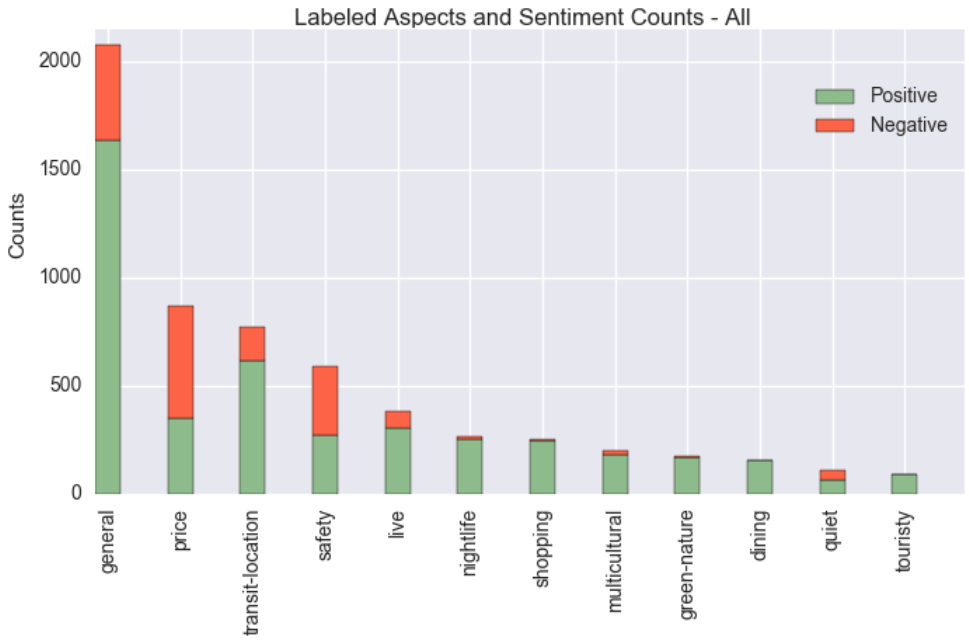


Figure 1: Number of annotated aspects and their sentiments.

3 Task

We define the task of targeted aspect-based sentiment analysis as follows: given a unit of text s (for example, a sentence), provide a list of tuples (labels) $\{(l, a, p)\}_{t=0}^{T}$, where p is the polarity expressed for the aspect a of entity l. Each sentence can have zero to T number of labels associated with it.

Within the current aspect-based sentiment analysis work, three tasks are defined (Brychcın et al., 2014): detecting the aspect, detecting the opinion target expression and detecting the sentiment, with detecting the opinion target expression being an intermediary task for identifying the sentiment of the aspect.

Here we focus on identifying only the aspect and sentiment for each entity. We identify each aspect, its relevant sentiment and the target location entity jointly by introducing a new polarity class called "None". "None" indicated that a sentence does not contain an opinion for the aspect a of location l. Therefore the overall task can be defined as a three-class classification task for each (l, a) pair with labels "Positive", "Negative", "None". Table 2 shows an example of the input sentence and output labels.

Sentence	**Labels**
location1 is very safe and **location2** is too far	(location1,safety,Positive) (location1,transit-location,None) (location2,safety,None) (location2,transit-location,Negative)

Table 2: Example of an input sentence and the output labels.

4 Evaluation

Most existing work in aspect-based sentiment analysis field, report F_1 measure for aspect detection task, and accuracy for sentiment classification. The scores can be calculated over 2-class or 3-class sentiments (Pontiki et al., 2015). In our results, F_1 score is calculated with a threshold that is optimized on validation set.

We also propose the AUC (area under the ROC curve) metric for both aspect and sentiment detection tasks. AUC captures the quality of the ranking of output scores and does not rely on a threshold.

5 Baseline

Here we propose baselines for the task. In all our methods, we treat the task as a three-class classification for each aspect and use a softmax function as follows:

$$p(y_{l,a} = c) = \text{softmax}(c) = \frac{exp(w_c.e_l + b_c)}{\sum_{c'=1}^{C} exp(w_{c'}.e_l + b_{c'})} \quad (1)$$

where $y_{l,a}$ is the sentiment label of aspect a for location l. w_c and b_C are the weights and the bias specific to each sentiment class c, respectively. e_l is a representation of location l. This representation can be a BoW or a distributional representation. Each method that we propose here define their own specific representation for e_l.

5.1 Logistic Regression

Many existing works in the aspect-based sentiment analysis task,[3] use a classifier, such as logistic regression or SVM, based on linguistic features such as n-grams, POS information or more hand-engineered features. We can think of these features as a sparse representation e_l that enter the softmax in equation 1. More concretely, we define the following sparse representations of locations:

Mask target entity n-grams: For each location, we define an n-gram representation over the sentence and mask the target location using a special token. This can help to differentiate between representations of two locations present in the same sentence.

Left-right n-grams: we create an n-gram representation for both the right and the left context around each location mention. We then concatenate these two representations to obtain one single feature vector.

Left right pooling: Previously embedding representations over the left and right context have been used for automatic feature detection in the targeted sentiment analysis task (Vo and Zhang, 2015). Inspired by this approach, we obtain max, min, average and standard deviation pooling over all the word embeddings for left and right context separately. We then combine the pooled embeddings of the left and right context to obtain a single feature vector. Word embeddings are obtained by running word2vec tool on a combination of our Yahoo! Answers corpus and a substantially big corpus from the web.[4]

5.2 Long Short-Term Memory (LSTM)

Inspired by the recent success of applying deep neural networks on language tasks, we use a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to learn a classifier for each of the aspects. Representations for a location (e_l) are obtained using one of the following two approaches:

Final output state (LSTM - Final): e_l is the output embedding of the bidirectional LSTM.

Location output state (LSTM - Location): e_l is the output representation at the index corresponding to the location entity as illustrated in Figure 2.

[3] including participants of SemEval ABSA tasks

[4] http://ebiquity.umbc.edu/redirect/to/resource/id/351/UMBC-webbase-corpus

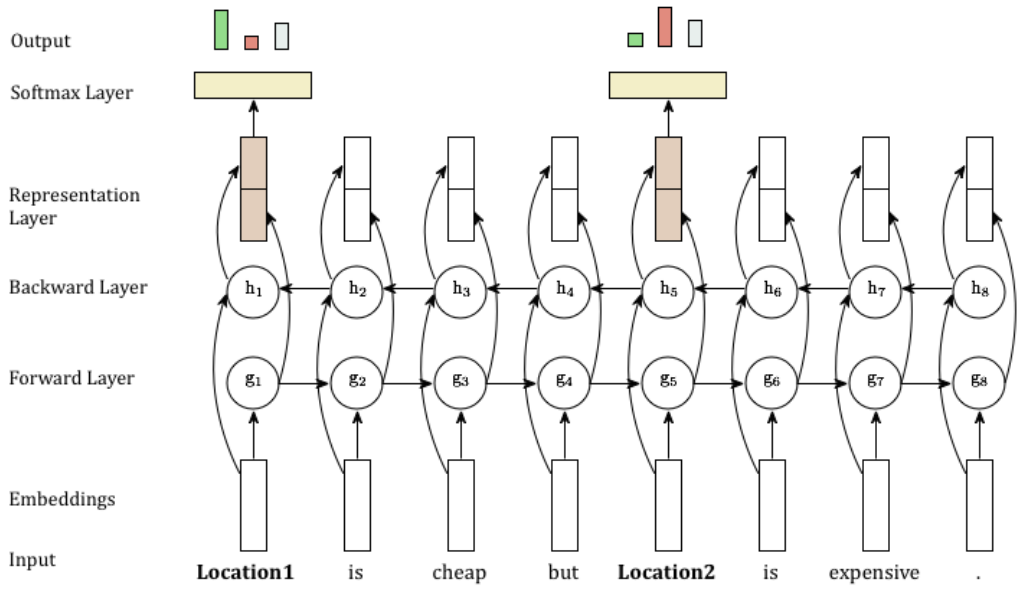


Figure 2: Bidirectional LSTM outputs a representation for each token in the sentence. The output state at the index of each location is then fed into a softmax layer to identify the sentiment class for the corresponding aspect. In this figure, LSTM is trained to identify the sentiment of aspect "price". Model should predict "Positive" for **location1** and "Negative" for **location2**

6 Experiments

In this paper, we select the four most frequent aspects from the dataset which are: "price", "safety", "transit-location" and "general" but the same approach can be applied to the remaining aspects. We divide each collection of single and multiple location mentions into train, dev and test set, with each having 70%, 10% and 20% of data respectively. We choose the best model with respect to the dev set.

In the case of the LSTM, we evaluate the loss on both training set and dev set after each iteration. We save the best model which has the lowest loss on the dev set over all the iterations. We then run this model on the test set and report the results. We report results separately on both categories of single location sentences and sentences with two locations and over all the data in the test set. Results on single location sentences mainly show the ability of the model to detect the correct sentiment for an aspect. On the other hand, results on two location sentences demonstrate the ability of the system not only on detecting the relevant sentiment of an aspect but also on recognising the target entity of the opinion.

Training LSTMs We implement our LSTM models using tensorflow (ten, 2015). To tackle the problem of having an unbalanced dataset (i.e. too many "None" instances), we train the LSTM model in batches with every batch having the same number of sentences selected randomly from each sentiment class. We tune the hyper parameters of the model on the dev set. The best model uses hidden units of size 50 and batch sizes of size 150. The Adam optimizer is used for optimization with a starting learning rate of 0.01 which is tuned to be the best performing on the dev set. Dropout is used both on initial word embeddings and on LSTM cells with the probability of 0.001. Tensorflow (ten, 2015) is used for the implementation of LSTM.

Training Logistic Regression Logistic regression models were based on implementations from scikit-learn.[5] Since we have an unbalanced dataset, we use a weighted logistic regression. To obtain the best weights, we cross-validate them on the development set. Weights inversely proportional to the size of each class result in the best performance.

[5] `http://scikit-learn.org/`

7 Results

Table 3 shows the results (averaged over all selected aspects) in terms of both F_1/accuracy and AUCs. It also shows the results of logistic regression based models versus LSTM models.

As we can see, the n-gram representation with location masking achieves slightly better results over the left-right context. N-grams include unigrams and bigrams. Also, by adding POS information, we gain an increase in the performance. We also experimented with adding tri-grams but it did not have a positive effect on the overall scores. Separating the left and the right context (LR-Left-Right) for BoW representation, does not improve the performance. Left-right pooling of dense embeddings performed weakly in comparison with other representations and therefore their results were omitted.

Amongst the two variations of LSTM, the model with final state embeddings does slightly better than the model where we use the embeddings at the location index, however they are not significantly different (with a p value less than 0.01). It is interesting to note that the best LSTM model is not superior to logistic regression model, especially in terms of AUC. This can be due to the fact that the amount of training data is not sufficient for LSTM to perform well. Moreover, while we provide some grammar information to logistic regression model through POS tags, such information is not incorporated into LSTM models. Another interesting observation is that the F_1 measure for logistic regression model with n-grams and POS information is very low while this model's performance is superior to other models in terms of AUC. This is because in general, it is easier to rank prediction scores than to assign predicted labels to instances by choosing a hard threshold.

Model	Aspect (F_1)	Sentiment (Accuracy)	Aspect (AUC)	Sentiment (AUC)
LR-Left-Right	0.683	0.847	0.903	0.875
LR-Mask(ngram)	**0.697**	0.853	0.918	0.885
LR-Mask(ngram+POS)	*0.393*	**0.875**	**0.924**	**0.905**
LSTM-Final	0.689	0.820	0.898	0.854
LSTM-Location	**0.693**	0.819	0.897	0.839

Table 3: Results of best logistic regression (LR) models and LSTM models.

Table 4 shows the average AUC (over aspect and sentimentclassification tasks) for two categories of data: Single — sentences that contain one location entity and Multi — sentences that contain two location entities. While logistic regression can perform slightly better on son Single location sentences, LSTM performs slightly better on Multi location sentences.

Model	Single	Multi
LR - Mask (n-gram + POS)	**0.916**	**0.907**
LSTM - Final	0.872	0.890

Table 4: Results of best logistic regression (LR) and LSTM models on sentences with a single location (Single) and multiple locations (Multi). AUC scores are averaged over aspect and sentiment classification tasks.

Table 5 shows the break down of average AUC scores for each aspect. We can see that aspects such as "safety" can be predicted with a better AUC score than aspect "general".

Model	Price	Safety	Transit	General
LR - Mask (n-gram + POS)	**0.940**	**0.960**	**0.879**	0.864
LSTM - Final	0.875	0.932	0.836	**0.869**

Table 5: LR and LSTM performance breakdown on aspects. AUC scores are averaged over aspect and sentiment detection.

Table 6 shows examples of correct and incorrect predictions using the best logistic regression model. The top part of the table contains examples that each contain a single location entity. At the bottom of the table, a sentence with two location entities is provided. The system correctly identifies that a "Positive" sentiment is expressed for the *general* aspect about **location2**. However, no sentiment is expressed for this aspect for **location1**.

Sentence	Aspect	Predicted	Label
location1 is not a nice cheap residential area to live trust me i was born and raised there	Price	Positive	Negative
I think you'd find it tough to find something affordable in **location1**	Price	Positive	Negative
I can't recommend **location1** for affordability	Price	Negative	Negative
I only know about **location1**, most people prefer location2	General	None	None
I only know about location1, most people prefer **location2**	General	Positive	Positive

Table 6: Examples of input sentences and predicted labels using the best system (LR - Mask (n-gram + POS). Target entity locations are highlighted in bold.

8 Related Work

The term sentiment analysis was first used in (et al, 2003). Since then, the field has received much attention from both research and industry. Sentiment analysis has applications in almost in every domain and it raised many interesting research questions. Furthermore, the availability of a huge volume of opinionated data on social media platforms has accelerated the development in this area.

In the beginning work on sentiment analysis mainly focused on identifying the overall sentiment of a unit of text. The unit of text varied from document (Pang et al., 2002; Turney, 2002), paragraph or sentences (Hu and Liu, 2004). However, only considering the overall sentiment fails to capture the sentiments over the aspects on which an entity can be reviewed or sentiment expressed toward different entities. Two remedy this, two new tasks have been introduced: *aspect-based* sentiment analysis and *targeted* sentiment analysis.

Aspect based sentiment analysis assumes a *single entity* per a unit of analysis and tries to identify sentiments towards different aspects of the entity (Lu et al., 2011; Lakkaraju et al., 2014; Alghunaim, 2015; Bagheri et al., 2013; Somprasertsri and Lalitrojwong, 2008; Alghunaim, 2015; Lu et al., 2011; Titov and McDonald, 2008; Brody and Elhadad, 2010). This task however does not consider more than one entity in the given text.

Targeted (target dependent) sentiment analysis is another task that identifies polarity towards a target entity (as opposed to over entire unit of text) (Mitchell et al., 2013; Jiang et al., 2011; Dong et al., 2014; Vo and Zhang, 2015; Zhang et al., 2016). (Jiang et al., 2011) was the first to propose targeted sentiment analysis on Twitter and demonstrates the importance of targets by showing that 40% of sentiment errors are due to not considering them in classification. However this task only identifies the *overall sentiment* and the existing corpora for the task consist only of text with one single entity per unit of analysis.

The task of targeted aspect-based sentiment analysis caters for more generic text by making fewer assumptions while extracting fine-grained information.

9 Conclusion

In this paper, we introduced the task of *targeted aspect-based* sentiment analysis and a new dataset. We also provide two strong baselines using logistic regression and LSTM. Ways to improve the baselines can involve using parse trees for identifying the context of each location. Data augmentation can be used

to make the models and especially LSTM more robust to variations in the data. We also like to provide more detailed analysis of what each system can achieve.

References

Abdulaziz Alghunaim. 2015. *A Vector Space Approach for Aspect-Based Sentiment Analysis*. Ph.D. thesis, Massachusetts Institute of Technology.

Ayoub Bagheri, Mohamad Saraee, and Franciska De Jong. 2013. Care more about customers: unsupervised domain-independent aspect detection for sentiment analysis of customer reviews. *Knowledge-Based Systems*, 52:201–213.

Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 804–812. Association for Computational Linguistics.

Tomáš Brychcın, Michal Konkol, and Josef Steinberger. 2014. Uwb: Machine learning approach to aspect-based sentiment analysis. *SemEval 2014*, page 817.

JACOB COHEN. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Sanjiv Das and Mike Chen. 2001. Yahoo! for amazon: Extracting market sentiment from stock message boards. In *Proceedings of the Asia Pacific finance association annual conference (APFA)*, volume 35, page 43. Bangkok, Thailand.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *ACL (2)*, pages 49–54.

Jeonghee Yi et al. 2003. Extracting sentiments about a given topic using natural language processing techniques; jeonghee yi et al; ibm. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03) 0-7695-1978-4/03*, volume 17.

Michael Gamon, Anthony Aue, Simon Corston-Oliver, and Eric Ringger. 2005. Pulse: Mining customer opinions from free text. In *Advances in Intelligent Data Analysis VI*, pages 121–132. Springer.

Gayatree Ganu, Noemie Elhadad, and Amélie Marian. 2009. Beyond the stars: Improving rating predictions using review text content. In *WebDB*, volume 9, pages 1–6. Citeseer.

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 151–160. Association for Computational Linguistics.

Yohan Jo and Alice H Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824. ACM.

Himabindu Lakkaraju, Richard Socher, and Chris Manning. 2014. Aspect specific sentiment analysis using hierarchical deep learning.

Maria Liakata, Simone Teufel, Advaith Siddharthan, Colin R Batchelor, et al. 2010. Corpora for the conceptualisation and zoning of scientific papers. In *LREC*. Citeseer.

Bin Lu, Myle Ott, Claire Cardie, and Benjamin K Tsou. 2011. Multi-aspect sentiment analysis with topic models. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 81–88. IEEE.

Margaret Mitchell, Jacqueline Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment.

Satoshi Morinaga, Kenji Yamanishi, Kenji Tateishi, and Toshikazu Fukushima. 2002. Mining product reputations on the web. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 341–349. ACM.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Ioannis Pavlopoulos. 2014. Aspect based sentiment analysis. *Athens University of Economics and Business*.

Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics, Denver, Colorado*, pages 486–495.

Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, Véronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeny Kotelnikov, Nuria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. SemEval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, June. Association for Computational Linguistics.

Gamgarn Somprasertsri and Pattarachai Lalitrojwong. 2008. Automatic product feature extraction from online product reviews using maximum entropy with lexical and syntactic features. In *Information Reuse and Integration, 2008. IRI 2008. IEEE International Conference on*, pages 250–255. IEEE.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics.

2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th international conference on World Wide Web*, pages 111–120. ACM.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 1347–1353.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *Thirtieth AAAI Conference on Artificial Intelligence*.

On the Impact of Seed Words on Sentiment Polarity Lexicon Induction

Dame Jovanoski, Veno Pachovski
University American College Skopje
UACS, Macedonia
{jovanoski,pachovski}@uacs.edu.mk

Preslav Nakov
Qatar Computing Research Institute
HBKU, Qatar
pnakov@qf.org.qa

Abstract

Sentiment polarity lexicons are key resources for sentiment analysis, and researchers have invested a lot of efforts in their manual creation. However, there has been a recent shift towards automatically extracted lexicons, which are orders of magnitude larger and perform much better. These lexicons are typically mined using bootstrapping, starting from *very few* seed words whose polarity is given, e.g., 50-60 words, and sometimes even just 5-6. Here we demonstrate that much higher-quality lexicons can be built by starting with hundreds of words and phrases as seeds, especially when they are in-domain. Thus, we combine (*i*) mid-sized high-quality manually crafted lexicons as seeds and (*ii*) bootstrapping, in order to build large-scale lexicons.

1 Introduction

The recent rise of social media has greatly democratized content creation. Facebook, Twitter, Skype, WhatsApp and LiveJournal are now commonly used to share thoughts and opinions about anything in the surrounding world. This proliferation of social media content has created new opportunities to study public opinion, with Twitter being especially popular for research due to its scale, representativeness, variety of topics discussed, as well as ease of public access to its messages.

Naturally, this abundance of data has attracted business and research interest from various fields including marketing, political science, and social studies, among many others, which are interested in questions like these: *Do people like the new Apple Watch? What do they hate about iPhone6? Do Americans support ObamaCare? What do Europeans think of Pope's visit to Palestine? How do we recognize the emergence of health problems such as depression? Do Germans like how Angela Merkel is handling the refugee crisis in Europe? What do republican voters in USA like/hate about Donald Trump?* Answering these questions requires studying the sentiment of opinions people express in social media, which has given rise to the fast growth of the field of sentiment analysis in social media.

Initially, sentiment analysis was addressed as a text classification problem, but it was soon realized that sizable performance gains can be obtained from using carefully built sentiment polarity lexicons as a source of external knowledge. Thus, researchers have invested a lot of efforts in the manual creation of such lexicons, which were typically of small to moderate size, e.g., less than 10,000 words. Recently, there has been a shift towards using automatically extracted lexicons, which are orders of magnitude larger and perform much better. These lexicons are typically mined using bootstrapping, starting from *very few* seed words whose polarity is given, e.g., 50-60 words, and sometimes even just 5-6.

Here, we demonstrate that sizable further performance gains can be observed by starting with mid-sized seeds (hundreds of words and phrases), thus getting the best of both worlds: (*i*) using high-quality mid-sized manually crafted lexicons as seeds, and (*ii*) extending them automatically using bootstrapping.

The remainder of the paper is organized as follows: Section 2 presents some related work. Section 3 describes our training and testing datasets. Section 4 presents the various lexicons we created for Macedonian. Section 5 gives details about our system, including the pre-processing steps and the features used. Section 6 describes our experiments and discusses the results. Section 7 concludes with possible directions for future work.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1557–1567, Osaka, Japan, December 11-17 2016.

2 Related Work

In this section, we first present work on sentiment analysis in general: methods used, work on sentiment analysis on Twitter, and relevant tasks at SemEval. Then, we present work on sentiment polarity lexicon induction, and finally, we discuss sentiment analysis for Macedonian.

2.1 Sentiment Analysis

Research in sentiment analysis started in the early 2000s. Initially, it was regarded as standard document classification into topics (Pang et al., 2002). However, researchers soon realized that it was quite different from standard document classification (Sebastiani, 2002), e.g., into categories such as business, sport and politics, and that sentiment analysis crucially needs external knowledge in the form of sentiment polarity lexicons. See for example the surveys by Pang and Lee (2008) and Liu and Zhang (2012) for more detail about research in sentiment analysis.

Around the same time, other researchers realized the importance of external sentiment lexicons, e.g., Turney (2002) proposed an unsupervised approach to learn the sentiment orientation of words/phrases: positive vs. negative. Later work studied the linguistic aspects of expressing opinions, evaluations, and speculations (Wiebe et al., 2004), the role of context in determining the sentiment orientation (Wilson et al., 2005), of deeper linguistic processing such as negation handling (Pang and Lee, 2008), of finer-grained sentiment distinctions (Pang and Lee, 2005), of positional information (Raychev and Nakov, 2009), etc. Moreover, it was recognized that in many cases, it is crucial to know not just the polarity of the sentiment, but also the topic towards which this sentiment is expressed (Stoyanov and Cardie, 2008).

Early sentiment analysis research focused on customer reviews of movies, and later of hotels, phones, laptops, etc. Later, with the emergence of social media, sentiment analysis in Twitter became a hot research topic. Unfortunately, research in that direction was hindered by the unavailability of suitable datasets and lexicons for system training, development and testing. While some Twitter-specific resources were developed, initially they were either small and proprietary, such as the i-sieve corpus (Kouloumpis et al., 2011), were created only for Spanish like the TASS corpus (Villena-Román et al., 2013), or relied on noisy labels obtained automatically based on emoticons and hashtags (Mohammad, 2012; Pang et al., 2002; Mohammad et al., 2013).

This situation changed with the shared task on *Sentiment Analysis on Twitter*, which was organized at SemEval, the International Workshop on Semantic Evaluation, a semantic evaluation forum previously known as SensEval. The task ran in 2013, 2014, 2015 and 2016, attracting over 40+ of participating teams in all four editions. While the focus was on general tweets, the task also featured out-of-domain testing on SMS messages, LiveJournal messages, as well as on sarcastic tweets.

SemEval-2013 task 2 (Nakov et al., 2013) and SemEval-2014 Task 9 (Rosenthal et al., 2014) focused on expression-level and message-level polarity. SemEval-2015 Task 10 (Rosenthal et al., 2015; Nakov et al., 2016b) featured topic-based message polarity classification, on detecting trends towards a topic, and on determining the out-of-context (a priori) strength of association of Twitter terms with positive sentiment. SemEval-2016 Task 4 (Nakov et al., 2016a) introduced a 5-point scale, which is popular and is commonly used for human review ratings on popular websites such as Amazon, TripAdvisor, Yelp, etc.; from a research perspective, this meant moving from classification to *ordinal regression*. Moreover, some subtasks of the general task focused on *quantification*, i.e., determining what proportion of a set of tweets on a given topic are positive/negative about it. It also featured a 5-point scale *ordinal quantification* subtask (Gao and Sebastiani, 2015).

Other related (mostly non-Twitter) tasks explored aspect-based sentiment analysis (Pontiki et al., 2014; Pontiki et al., 2015; Pontiki et al., 2016), sentiment analysis of figurative language on Twitter (Ghosh et al., 2015), implicit event polarity (Russo et al., 2015), stance in tweets (Mohammad et al., 2016), out-of-context sentiment intensity of phrases (Kiritchenko et al., 2016), and emotion detection (Strapparava and Mihalcea, 2007). Some of these tasks featured languages other than English.

2.2 Sentiment Polarity Lexicons

Despite the huge variety of knowledge sources explored in the literature, sentiment polarity lexicons remained the only universally recognized resource for the task of sentiment analysis. Until recently, such sentiment polarity lexicons were manually crafted, and were thus of small to moderate size, e.g., LIWC (Pennebaker et al., 2001) has 2,300 words, the General Inquirer (Stone et al., 1966) contains 4,206 words, Bing Liu's lexicon (Hu and Liu, 2004) includes 6,786 words, and MPQA (Wilson et al., 2005) has about 8000. Early efforts in building them automatically also yielded lexicons of moderate sizes such as the SentiWordNet (Esuli and Sebastiani, 2006; Baccianella et al., 2010).

However, recent results have shown that automatically extracted large-scale lexicons (e.g., up to a million words and phrases) offer important performance advantages, as confirmed at shared tasks on Sentiment Analysis on Twitter at SemEval 2013-2016 (Nakov et al., 2013; Rosenthal et al., 2014; Rosenthal et al., 2015; Nakov et al., 2016a), where over 40 teams participated four years in a row.

Using such large-scale lexicons was crucial for the performance of the top-performing systems. Similar observations were made in the related Aspect-Based Sentiment Analysis task at SemEval 2014 (Pontiki et al., 2014). In both tasks, the winning systems benefitted from building and using massive sentiment polarity lexicons (Mohammad et al., 2013; Zhu et al., 2014).[1] The two most popular large-scale lexicons were the Hashtag Sentiment Lexicon and the Sentiment140 lexicon, which were developed by the team of NRC Canada for their participation in the SemEval-2013 shared task on sentiment analysis on Twitter.

The importance of building sentiment polarity lexicons has resulted in a special subtask (Rosenthal et al., 2015) at SemEval-2015 (part of Task 4), and an entire task (Kiritchenko et al., 2016) at SemEval-2016 (namely, Task 7), on predicting the out-of-context sentiment intensity of words and phrases.[2]

These large-scale automatic lexicons are typically built using bootstrapping, starting with a small set of seeds of, e.g., 50-60 words, and sometimes even just two emoticons (Mohammad et al., 2013).

Here, we demonstrate that sizable further performance gains can be observed by starting with mid-sized seeds (i.e., hundreds of words and phrases), thus getting the best of both worlds: (*i*) using high-quality mid-sized manually crafted lexicons as seeds, and (*ii*) further extending them automatically using bootstrapping.

2.3 Sentiment Analysis for Macedonian

In our experiments below, we focus on Macedonian (tweets), for which we only know two publications on sentiment analysis, none of which is about Twitter.

Gajduk and Kocarev (2014) experimented with 800 posts from the Kajgana forum (260 positive, 260 negative, and 280 objective), using SVM and Naïve Bayes classifiers, and features such as bag of words, rules for negation, and stemming.

Uzunova and Kulakov (2015) experimented with 400 movie reviews[3] (200 positive, and 200 negative; no objective/neutral), and a Naïve Bayes classifier, using a small manually annotated sentiment lexicon of unknown size, and various preprocessing techniques such as negation handling and spelling/character translation.

Unfortunately, the datasets and the generated lexicons used in the above work are not publicly available, and/or are also from a different domain (i.e., not Twitter). As we are interested in sentiment analysis of Macedonian tweets, we had to build our own datasets. We have described these datasets and initial experiments with them in an earlier publication (Jovanoski et al., 2015), where the focus was on the datasets and on the classifier; in contrast, here we focus on assessing the impact of our proposed lexicon generation method. Below we will describe these datasets in detail, for the sake of self-containment of the present paper.

[1] Such lexicons proved useful for other tasks at SemEval, e.g., for SemEval-2016 Task 3 on Community Question Answering (Balchev et al., 2016).

[2] We should note though that the utility of using sentiment polarity lexicons for sentiment analysis probably needs to be revisited, as the best system at SemEval-2016 Task 4 could win without using any lexicons (Deriu et al., 2016).

[3] There have been also experiments on movie reviews for the closely related Bulgarian language (Kapukaranov and Nakov, 2015), but there the objective was to predict user rating, which was addressed as an ordinal regression problem.

3 Data

We downloaded half a million tweets in Macedonian, which we collected over a six-month period spanning from November 2014 to April 2015. We tried to download all Macedonian tweets based on the Twitter language classification. However, it turned out that in many cases, the returned tweets were in Bulgarian or Russian, which are also Slavic languages and share the same alphabet with Macedonian. Thus, we trained and used our own Naïve Bayes classifier, which achieved over 95% accuracy.[4] We used part of these tweets as training and testing data, and the rest for building automatic lexicons.

Table 1 shows statistics about the training and the testing datasets. We can see that they are somewhat balanced between positive and negative tweets, and that there is smaller proportion of neutral tweets.[5]

The *testing data* was annotated for sentiment at the tweet level (using *positive*, *negative*, and *neutral/objective* as labels[6]) by two annotators, both native speakers of Macedonian. The Cohen's Kappa statistics (Cohen, 1960) for the inter-annotator agreement was 0.64, which corresponds to *substantial* agreement (Landis and Koch, 1977). Our follow-up analysis has shown that the main disagreement was about distinguishing between negative and neutral tweets. In the final testing dataset, we discarded all tweets with disagreement (a total of 482 tweets).

The *training data* was annotated by a single annotator, one of those who annotated the testing dataset. In addition to producing tweet-level sentiment polarity annotations, the annotator further marked the positive and the negative phrases inside each tweet. We will use the set of these words and phrases, together with their polarities, as a sentiment lexicon, and also as seeds when bootstrapping a large-scale automatic sentiment lexicon from the remaining unannotated tweet messages.

Dataset	Positive	Neutral	Negative	Total
Train	2,610 (30%)	1,280 (15%)	4,693 (55%)	8,583
Test	431 (38%)	200 (18%)	508 (44%)	1,139
No annot.	–	–	–	0.5M

Table 1: Statistics about the datasets.

4 Sentiment Lexicons

A sentiment lexicon contains words and phrases annotated with positive and negative sentiment, sometimes with numerical intensity, e.g., *spectacular* could have positive strength of 0.91, while for *okay* that might be 0.3. Below we describe the sentiment lexicons we built and experimented with.

4.1 Manually-Crafted Lexicon

As we mentioned above, our training dataset was annotated with sentiment words and phrases, a total of 1,088: 459 positive and 629 negative. These terms form our manually-crafted lexicon.

4.2 Translated Lexicons

As no sentiment polarity lexicons are publicly available for Macedonian, we translated some popular English manually-crafted lexicons such as Bing Liu's lexicon (2,006 positive and 4,783 negative), and MPQA (2,718 positive and 4,912 negative), and a Bulgarian lexicon (5,016 positive and 2,415 negative), extracted from a movie reviews website (Kapukaranov and Nakov, 2015), which includes 694 positive and 2,966 negative English words. We used Google Translate, and we further manually corrected some of the results: we removed some bad translations and we corrected the grammar.

[4] At the 2015 Discriminating between Similar Languages (DSL) shared task (Zampieri et al., 2015), the participating systems distingushed Macedonian from Bulgarian with 100% accuracy, which shows that this is an easy task; as a result, this language pair was not included in the 2016 edition of the task (Malmasi et al., 2016). Our Naïve Bayes classifier achieved slightly lower accuracy as we deal with tweets, which are short and harder to categorize than the newswire texts in the DSL task.

[5] It was previously reported that most tweets are neutral, but this was for English, and for tweets about selected topics (Rosenthal et al., 2014). Here, we have no topic restriction. Moreover, there is an ongoing political crisis in Macedonia, and thus Macedonian tweeps express a lot of emotions rather than staying neutral.

[6] Following (Nakov et al., 2013), we merged *neutral* and *objective* as they are commonly confused by annotators.

4.3 Bootstrapped Lexicons

Various approaches have been proposed in the literature for bootstrapping sentiment polarity lexicons starting from a small set of seeds: positive and negative terms (words and phrases).

A very influential approach is that of Turney (2002), which uses pointwise mutual information and bootstrapping to build a large lexicon and to estimate the semantic orientation of each word in that lexicon. The idea is to start with a small set of seed positive (e.g., *excellent*) and negative words (*bad*), and then to use these words to induce sentiment polarity orientation for new words in a large unannotated set of texts (in his case, product reviews). The idea is that words that co-occur in the same text with positive seed words are likely to be positive, while those that tend to co-occur with negative words are likely to be negative. To quantify this intuition, Turney defines the notion of sentiment orientation (SO) for a term w as follows:

$$SO(w) = pmi(w, pos) - pmi(w, neg)$$

where PMI is the pointwise mutual information, pos and neg are placeholders standing for any of the seed positive and negative terms, respectively, and w is a target word/phrase from the large unannotated set of texts (here tweets).

A positive/negative value for $SO(w)$ indicates positive/negative polarity for w, and its magnitude shows the corresponding sentiment strength. In turn, $pmi(w, pos) = \frac{P(w,pos)}{P(w)P(pos)}$, where $P(w, pos)$ is the probability to see w with any of the seed positive words in the same tweet,[7] $P(w)$ is the probability to see w in any tweet, and $P(pos)$ is the probability to see any of the seed positive words in a tweet; $pmi(w, neg)$ is defined similarly.

The pointwise mutual information (PMI) is a notion from information theory: given two random variables A and B, the mutual information of A and B is the "amount of information" (in units such as bits) obtained about the random variable A, through the random variable B (Church and Hanks, 1990).

Let a and b be two values from the sample space of A and B, respectively. The *pointwise* mutual information between a and b is defined as follows:

$$pmi(a; b) = \log \frac{P(A = a, B = b)}{P(A = a) \cdot P(B = b)} = \log \frac{P(A = a|B = b)}{P(A = a)} \quad (1)$$

$pmi(a; b)$ takes values between $-\infty$, which happens when $P(A = a, B = b) = 0$, and $\min\{-\log P(A = a), -\log P(B = b)\}$, when $P(A = a|B = b) = P(B = b|A = a) = 1$.

In his experiments, Turney (2002) used five positive and five negative words as seeds. His PMI-based approach further served as the basis for the creation of the two above-mentioned large-scale automatic lexicons for sentiment analysis in Twitter for English, initially developed by NRC for their participation in SemEval-2013 (Mohammad et al., 2013). The *Hashtag Sentiment Lexicon* uses as seeds hashtags containing 32 positive and 36 negative words, e.g., `#happy` and `#sad`. Similarly, the *Sentiment140* lexicon uses smileys as seed indicators for positive and negative sentiment, e.g., `:)`, `:-)` and `:))` as positive seeds, and `:(` and `:-(` as negative ones.

Recently, Severyn and Moschitti (2015) proposed an approach to lexicon induction, which, instead of using PMI **(SO)**, assigns positive/negative labels to the unlabeled tweets (based on the seeds), and then trains an SVM classifier on them, using word n-grams as features. These n-grams are then used as lexicon entries with the learned classifier weights as polarity scores.

In our experiments below, we calculate $SO(w)$ using PMI or LR (Logistic Regression[8]), and we experiment with different seeds:

- the 1+1 seeds of Turney (2002), translated to Macedonian ("excellent" and "poor");
- the 7+7 seeds of Turney and Littman (2003), translated to Macedonian;

[7] Here we explain the method using tweets as this is how we are using it, but Turney (2002) actually used page hits in the AltaVista search engine.

[8] LR worked better than SVM in our experiments.

- our 5+5 seeds, manually selected words in Macedonian with strong sentiment;
- 30+30 seeds, which we obtained by translating the 32+36 seeds[9] used for the *Hashtag Sentiment Lexicon* lexicon (but we used these seeds as regular words, not as hashtags);
- the 3+2 smileys from above;
- the 459+629 terms from our manually-crafted lexicon (we further experiment with random proportional positive/negative subsets of 100, 200, and 500 words thereof).

Table 2 shows some statistics about the lexicons we built (unigrams + bigrams) on the unanotated 0.5M tweets. We can see that the larger the seed, the larger the bootstrapped lexicons. Note that the lexicon sizes for PMI and LR are the same as they are calculating the sentiment orientation for the exactly same terms; what differs is the way the weights are being calculated.

Type of seed	Seeds	Unigrams	Bigrams	Total
Smileys: NRC	5	128	2,163	2,291
Words: Turney	10	865	14,343	15,208
Words: NRC	60	1,669	32,459	34128
Words: MCL	100	1,926	40,242	42,168
Words: MCL	200	3,752	60711	64,463
Words: MCL	500	7,219	124,977	132,196
Words: MCL	1,088	9,746	160,526	170,272

Table 2: Statistics about the lexicons we built using bootstrapping with PMI and LR. MCL is the manually-crafted lexicon.

5 The System

Below we describe our baseline system: the preprocessing, and the features used.

5.1 Preprocessing

For pre-processing, we applied various algorithms, which we combined in order to achieve better performance. We used Christopher Potts' tokenizer,[10] and we had to be careful since we had to extract not only the words but also other tokens such as hashtags, emoticons, user names, etc. The pre-processing of the tweets goes as follows:

1. **URL and username removal**: tokens such as URLs and usernames (i.e., tokens starting with `@`) were removed.
2. **Stopword removal**: stopwords were filtered out based on a word list (146 words).
3. **Repeating characters removal**: consecutive character repetitions in a word were removed, e.g., 'какоооо' became 'како' ('what' in English); also were removed repetitions of a word in the same token, e.g., 'дададада' became 'да' ('yes' in English).
4. **Negation handling**: negation was addressed using a predefined list of negation tokens, then the prefix `NEG_CONTEXT_` was attached to the following tokens until a clause-level punctuation mark, in order to annotate it as appearing in a negated context, as suggested in (Pang et al., 2002). A list of 45 negative phrases and words was used to signal negation.

[9] We lost some terms in the process of translation, e.g., because some English words translated to the same Macedonian word.

[10] http://sentiment.christopherpotts.net/tokenizing.html

Seeds for bootstrapping	Source	PMI			LR			PMI + LR
		B	B+S	B+S+M	B	B+S	B+S+M	B+S+M
–	–	–	61.99	78.18	–	61.99	78.18	78.18
1+1 words	(Turney, 2002)	51.48	62.29	78.40	59.82	63.57	78.51	78.89
2+3 smileys	(Mohammad et al., 2013)	61.12	63.81	78.69	65.18	68.99	78.95	79.62
5+5 words	manually-selected	62.55	64.59	79.25	66.70	69.73	80.13	80.87
7+7 words	(Turney and Littman, 2003)	63.02	66.27	79.71	66.98	69.82	80.54	81.99
30+30 words	(Mohammad et al., 2013)	63.47	68.51	79.84	67.28	70.01	80.68	81.33
50+50 words	our MCL	67.11	72.48	80.89	69.79	74.15	81.96	82.73
100+100 words	our MCL	70.94	76.30	82.76	71.41	77.47	84.72	85.45
250+250 words	our MCL	72.25	84.72	92.23	73.76	85.89	93.47	93.55
459+629 words	our MCL	73.82	90.91	94.12	75.29	91.02	94.32	94.44

Table 3: Sentiment classification results (F-score) using lexicons bootstrapped with PMI, LR, or both to calculate $SO(w)$: B = using the bootstrapped lexicon only, B+S = also using non-lexicon features, B+S+M = also using our MCL. The first line shows results when no bootstrapped lexicon is used.

5. **Non-standard to standard word mapping**: non-standard words (slang) were mapped to an appropriate form, according to a manualy crafted predefined list of mappings.

6. **PoS tagging**: rule-based, using a dictionary.

7. **Tagging positive/negative words**: positive and negative words were tagged as `POS` and `NEG`, using sentiment lexicons.

8. **Stemming**: rule-based stemming was performed, which removes or replaces some prefixes and suffixes.

In sum, we started the transformation of an input tweet by converting it to lowercase, followed by removal of URLs and user names. We then normalized some words to Standard Macedonian using a dictionary of 173 known word transformations, and we also removed the stopwords (from a list of 146 words). As part of the transformation, we marked the words in a negated context.

We further created a rule-based stemming algorithm with a list of 65 rules for removing/replacing prefixes and suffixes, inspired by the Porter stemmer (Porter, 1980). We used two groups of rules: 45 rules for affix removal, and 20 rules for affix replacement. Developing a stemmer for Macedonian was challenging as this is a highly inflective language, rich in both inflectional and derivational forms.

5.2 Features

In order to evaluate the impact of the sentiment lexicon, we defined features that are fully or partially dependent on the lexicons. When using multiple lexicons at the same time, there are separate instances of these features for each lexicon. Here are the features we used: number of positive terms, number of negative terms, ratio of the number of positive terms to the number of positive+negative terms, ratio of the number of negative terms to the number of positive+negative terms, sum of all positive scores, sum of all negative scores, sum of all scores, both positive and negative.

For classification, we used logistic regression. Our basic features were TF.IDF-weighted unigrams and bigrams, and also emoticons. We further included additional features that focus on the positive and on the negative terms that occur in the tweet together with their scores in the lexicon. In case of two or more lexicons being used together, we had a copy of each feature for each lexicon.

6 Experiments and Evaluation

Our evaluation setup follows that of the SemEval 2013-2016 task on Sentiment Analysis on Twitter (Nakov et al., 2013; Rosenthal et al., 2014; Rosenthal et al., 2015; Nakov et al., 2016a), and uses an F-score that is the average of the F_1 score for the positive, and the F_1 score for the negative class. Note that, even though implicit, the neutral class still matters in this score. Note also that our focus here is on assessing the impact of our proposed lexicon generation method, and not the classifier itself.

Table 3 shows the results when using each of the bootstrapped lexicons from Table 2. The upper part of the table shows experiments with translations of the seeds used in related work, as described above, while the lower part shows results with (a random subset) of our manually crafted lexicon. We can see that all lexicons outperform the *no bootstrapped lexicons* baseline. The results indicate that our manual lexicon is more useful than the bootstrapping lexicons built using small seeds: it improves over the baseline by 16 points absolute, while the NRC-style or Turney-style lexicons only improve by 2-8 points.

Moreover, using our manually crafted lexicon as a seed for bootstrapping works better than using it as a lexicon: 90.91 vs. 78.18 (with PMI). Moreover, combining it with a bootstrapped lexicon built using all 1,088 words as seeds yields an F-score of 94.12 (with PMI). Note that LR performs better than PMI, by up to four points. Yet, as the last column shows, there is also gain when combining them.

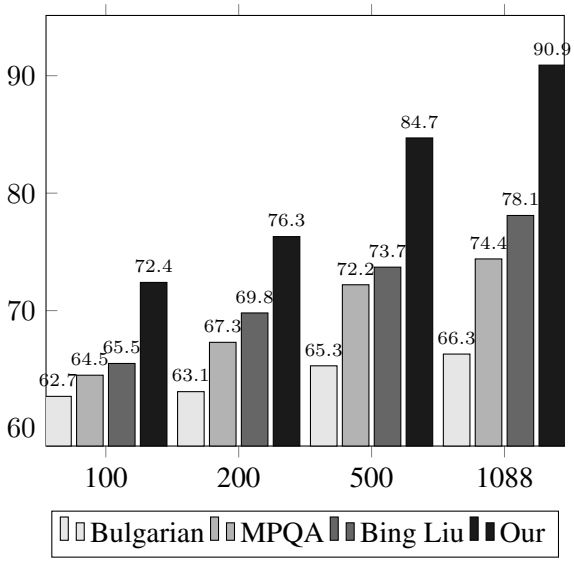


Figure 1: Sentiment polarity classification results (F-score) using different translated bootstrapped lexicons and numbers of seeds with PMI for $SO(w)$, and using B+S as features.

The B+S+M columns in Table 3 show results when using our manually-crafted 1,088-word lexicon as an additional lexicon in each experiment. We can see consistent improvement ranging from 3 to 15 points of F-score absolute on top of the performance of the bootstrapped lexicons. Most interestingly, the bigger the size of the seed, the better the performance of the resulting lexicon (improvement of up to 28 points). So, is it all about the size of the resulting lexicon (as Table 2 shows, bigger seeds yield bigger bootstrapped lexicons)? In order to test this hypothesis, we built bootstrapping lexicons with 100, 200, 500, and 1,088 seeds, with the seeds coming from our lexicons and from the three translated lexicons above. The results are shown in the Figure 1. There are consistent gains as the number of seeds increases, and this is true for seeds coming from our lexicon and also from translations of MPQA, Bing Liu's lexicon, and the Bulgarian lexicon.

However, not all seeds are created equal, even when they are of equal size, and we can see that it is much better to use our manually crafted lexicon as a source of seeds. Yet, if only using 100 seeds from our lexicon, the resulting bootstrapped lexicon would not be able to compete against one built using 1,088 seeds from MPQA or Bing Liu's lexicon. Next, we computed what proportion of each translated lexicon is contained in our lexicon – Bulgarian: 25%, MPQA: 37%, Bing Liu: 43%. We can see that the larger the overlap the better the lexicon, i.e., closer to our domain.

Thus, we can conclude that it is preferable to use (*i*) a *manually-crafted/in-domain lexicon* for the seeds, and (*ii*) a *mid-sized set of seeds*.

7 Conclusion and Future Work

We have presented experiments with different seeds for bootstrapping sentiment polarity lexicons. We have shown that it is best to use (*i*) a *mid-sized seed*, contrary to what is common practice, and (*ii*) a *manually-crafted/in-domain lexicon*, and (*iii*) a classifier such as LR rather than PMI. We have released all our Macedonian lexicons freely for research use.[11]

In future work, we plan experiments for other languages, other sets of seeds, other lexicons, and other learning methods. We further want to study the impact of the raw corpus size, e.g., we could only collect half a million tweets for Macedonian, while Mohammad et al. (2013) used 135 million English tweets. Also, we are interested not only in quantity but also in quality, i.e., in studying the impact of the quality of the individual words when used as seeds. An interesting work in that direction, even though in a different domain and context, is that of (Kozareva and Hovy, 2010).

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the International Conference on Language Resources and Evaluation*, LREC '10, Valletta, Malta.

Daniel Balchev, Yasen Kiprov, Ivan Koychev, and Preslav Nakov. 2016. PMI-cool at SemEval-2016 task 3: Experiments with PMI and goodness polarity lexicons for community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 844–850, San Diego, California, USA.

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

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37.

Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurelien Lucchi, Valeria De Luca, and Martin Jaggi. 2016. SwissCheese at SemEval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 1124–1128, San Diego, California, USA.

Andrea Esuli and Fabrizio Sebastiani. 2006. SENTIWORDNET: A publicly available lexical resource for opinion mining. In *Proceedings of the International Conference on Language Resources and Evaluation*, LREC '06, pages 417–422, Genoa, Italy.

Andrej Gajduk and Ljupco Kocarev. 2014. Opinion mining of text documents written in Macedonian language. *arXiv preprint arXiv:1411.4472*.

Wei Gao and Fabrizio Sebastiani. 2015. Tweet sentiment: From classification to quantification. In *Proceedings of the 7th International Conference on Advances in Social Network Analysis and Mining*, ASONAM '15, pages 97–104, Paris, France.

Aniruddha Ghosh, Guofu Li, Tony Veale, Paolo Rosso, Ekaterina Shutova, John Barnden, and Antonio Reyes. 2015. SemEval-2015 task 11: Sentiment analysis of figurative language in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval '15, pages 470–478, Denver, Colorado, USA.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, Seattle, Washington, USA.

Dame Jovanoski, Veno Pachovski, and Preslav Nakov. 2015. Sentiment analysis in Twitter for Macedonian. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, RANLP '15, pages 249–257, Hissar, Bulgaria.

Borislav Kapukaranov and Preslav Nakov. 2015. Fine-grained sentiment analysis for movie reviews in Bulgarian. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 266–274, Hissar, Bulgaria.

[11] `http://github.com/badc0re/sent-lex`

Svetlana Kiritchenko, Saif M Mohammad, and Mohammad Salameh. 2016. SemEval-2016 task 7: Determining sentiment intensity of English and Arabic phrases. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 42–51, San Diego, California, USA.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the OMG! In *Proceedings of the International Conference on Weblogs and Social Media*, ICWSM '11, Barcelona, Spain.

Zornitsa Kozareva and Eduard Hovy. 2010. Not all seeds are equal: Measuring the quality of text mining seeds. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL-HLT '10, pages 618–626, Los Angeles, California, USA.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–74, 3.

Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 415–463. Springer.

Shervin Malmasi, Marcos Zampieri, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, and Jörg Tiedemann. 2016. Discriminating between similar languages and Arabic dialect identification: A report on the third DSL shared task. In *Proceedings of the 3rd Workshop on Language Technology for Closely Related Languages, Varieties and Dialects*, VarDial '16, Osaka, Japan.

Saif Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the Seventh International Workshop on Semantic Evaluation Exercises*, SemEval '13, pages 321–327, Atlanta, Georgia, USA.

Saif M Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. SemEval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 31–41, San Diego, California, USA.

Saif Mohammad. 2012. #Emotional tweets. In *Proceedings of *SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task*, *SEM '12, pages 246–255, Montreal, Canada.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. SemEval-2013 task 2: Sentiment analysis in Twitter. In *Proceedings of the Seventh International Workshop on Semantic Evaluation*, SemEval '13, pages 312–320, Atlanta, Georgia, USA.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016a. SemEval-2016 Task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, pages 1–18, San Diego, California, USA.

Preslav Nakov, Sara Rosenthal, Svetlana Kiritchenko, Saif M. Mohammad, Zornitsa Kozareva, Alan Ritter, Veselin Stoyanov, and Xiaodan Zhu. 2016b. Developing a successful SemEval task in sentiment analysis of Twitter and other social media texts. *Language Resources and Evaluation*, 50(1):35–65.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, ACL '05, pages 115–124, Ann Arbor, Michigan, USA.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '02, pages 79–86, Philadelphia, Pennsylvania, USA.

James W. Pennebaker, Martha E. Francis, and Roger J. Booth. 2001. *Linguistic Inquiry and Word Count*. Lawerence Erlbaum Associates, Mahwah, NJ.

Maria Pontiki, Harris Papageorgiou, Dimitrios Galanis, Ion Androutsopoulos, John Pavlopoulos, and Suresh Manandhar. 2014. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, SemEval '14, pages 27–35, Dublin, Ireland.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. SemEval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval '2015, pages 486–495, Denver, Colorado, USA.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2016. SemEval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, USA.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Veselin Raychev and Preslav Nakov. 2009. Language-independent sentiment analysis using subjectivity and positional information. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, RANLP '09, pages 360–364, Borovets, Bulgaria.

Sara Rosenthal, Alan Ritter, Preslav Nakov, and Veselin Stoyanov. 2014. SemEval-2014 Task 9: Sentiment analysis in Twitter. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, SemEval '14, pages 73–80, Dublin, Ireland.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. SemEval-2015 task 10: Sentiment analysis in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval '15, pages 450–462, Denver, Colorado, USA.

Irene Russo, Tommaso Caselli, and Carlo Strapparava. 2015. SemEval-2015 task 9: CLIPEval implicit polarity of events. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval '15, pages 442–449, Denver, Colorado, USA.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March.

Aliaksei Severyn and Alessandro Moschitti. 2015. On the automatic learning of sentiment lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '15, pages 1397–1402, Denver, Colorado, USA.

Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, and Daniel M. Ogilvie. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. MIT Press.

Veselin Stoyanov and Claire Cardie. 2008. Topic identification for fine-grained opinion analysis. In *Proceedings of the 22nd International Conference on Computational Linguistics*, COLING '08, pages 817–824, Manchester, United Kingdom.

Carlo Strapparava and Rada Mihalcea. 2007. SemEval-2007 task 14: Affective text. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '07, pages 70–74, Prague, Czech Republic.

Peter D Turney and Michael L Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, 21(4):315–346.

Peter D. Turney. 2002. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, ACL '02, pages 417–424, Philadelphia, Pennsylvania, USA.

Vasilija Uzunova and Andrea Kulakov. 2015. Sentiment analysis of movie reviews written in Macedonian language. In *ICT Innovations 2014*, pages 279–288. Springer.

Julio Villena-Román, Sara Lana-Serrano, Eugenio Martínez-Cámara, and José Carlos González Cristóbal. 2013. TASS - Workshop on Sentiment Analysis at SEPLN. *Procesamiento del Lenguaje Natural*, 50:37–44.

Janyce Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Comput. Linguist.*, 30(3):277–308, September.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT-EMNLP '05, pages 347–354, Vancouver, British Columbia, Canada.

Marcos Zampieri, Liling Tan, Nikola Ljubešić, Jörg Tiedemann, and Preslav Nakov. 2015. Overview of the DSL shared task 2015. In *Proceedings of the Joint Workshop on Language Technology for Closely Related Languages, Varieties and Dialects*, LT4VarDial '15, pages 1–9, Hissar, Bulgaria.

Xiaodan Zhu, Svetlana Kiritchenko, and Saif M. Mohammad. 2014. NRC-Canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '14, pages 437–442, Dublin, Ireland.

Evaluating Argumentative and Narrative Essays using Graphs

Swapna Somasundaran, Brian Riordan, Binod Gyawali and Su-Youn Yoon
Educational Testing Service
660 Rosedale Road, Princeton
NJ 08541, USA
{ssomasundaran,briordan,bgyawali,SYoon}@ets.org

Abstract

This work investigates whether the development of ideas in writing can be captured by graph properties derived from the text. Focusing on student essays, we represent the essay as a graph, and encode a variety of graph properties including PageRank as features for modeling essay scores related to quality of development. We demonstrate that our approach improves on a state-of-the-art system on the task of holistic scoring of persuasive essays and on the task of scoring narrative essays along the development dimension.

1 Introduction

Development, elaboration and exemplification are important writing skills that come into play in many different genres of writing. In a persuasive essay, to produce good arguments, it is important to substantiate a stance. Similarly, in a story, it is important to develop plot, character and events. Consequently, skill in development is evaluated in essay writing tasks at all levels of education, from primary school to the graduate level. For example, on the Graduate Record Examination (GRE), the scoring guidelines recommend that the top score be assigned to persuasive essays that "develop the position fully with compelling reasons and/or persuasive examples"[1]. Primary school assessments of narrative writing (i.e., stories and personal experiences, real or imagined) also test this skill. For instance, the scoring rubric for one of the U.S. Common Core State Standards tests[2] has a dimension for "elaboration and development", where top scores are given to essays where "experiences, characters, settings and/or events are clearly developed".

Previous research has investigated techniques for the automated assessment of essays by evaluating aspects of writing such as grammar, fluency, and coherence (Shermis and Burstein, 2013; Miltsakaki and Kukich, 2004; Attali and Burstein, 2006; Rus and Niraula, 2012; Stab and Gurevych, 2014b; Somasundaran et al., 2014; Rahimi et al., 2015; Song et al., 2014; Farra et al., 2015). In this work, we investigate how to evaluate the development of ideas and exemplification. Specifically, we explore whether development is reflected in the structure of graphs constructed from the discourse proximity of essay concepts. We construct graphs for each essay where the essay's concepts comprise the nodes and links are formed from concepts occurring in adjacent sentences. We then use properties of each graph, such as PageRank, to predict essay quality, including the quality of development. Our hypothesis is that this novel graph representation can help distinguish essays with well-developed ideas from essays lacking development and elaboration.

We test the effectiveness of our approach on two different essay datasets: holistically scored persuasive essays and trait-scored narrative essays. Our results demonstrate that graph-based features are useful across genres for essay scoring, and improve the performance of models for both holistic scoring and trait scoring. With the addition of these features, we are able to improve on the state of the art in essay scoring.

[1] https://www.ets.org/gre/revised_general/prepare/analytical_writing/issue/scoring_guide

[2] http://sbac.portal.airast.org/wp-content/uploads/2015/03/Narrative-050814.pdf

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1568–1578, Osaka, Japan, December 11-17 2016.

The rest of the paper is organized as follows. Section 2 presents our intuitions, methods for graph construction, and features for scoring. Section 3 introduces our data. Experiments and analyses are in Sections 4 and 5, respectively. We discuss related work in Section 6 and conclude in Section 7.

2 Graphs for capturing development

In the process of developing a story or an argument, experienced writers provide detailed, illustrative examples. For example, consider a persuasive essay from Stab and Gurevych (2014a) responding to the prompt "In order to become integrated into society in their adopted countries, immigrants should abandon their old ways and adapt to local customs and codes of behavior. Do you agree or disagree?" The essay, a portion of which is shown in Example 1, takes a stance that immigrants should maintain their cultural identity.

Example 1. The last 50 years have seen an increasing number of immigrants to other countries... However I strongly believe that they are able to sustain their cultural identities and doing so help they keep their origin values.

Firstly, maintaining one's cultural identity is a key important rule to help individuals emerge in the new multicultural environments. Take Australia for example, ...

Secondly, it is crucial to keep ones identity for they need a connection back to their country as well as teach their children their value of origin. For instance, children immigrated to a new country will face social troubles ...

To conclude, ...

The writer first takes a stance that immigrants should sustain their identities. He then presents a claim ("maintaining one's cultural identity is a key important rule...") and then develops this idea ("Take Australia for example, ..."). Once this example is discussed, the writer moves on to another claim and example. The writer concludes by reiterating the main stance.

Intuitively, the linguistic correlates of development are new words related to sub-topics and allied topics that writers introduce into the discourse to support or illustrate their main points. Rather than repeating the same vocabulary over and over, writers enrich the vocabulary of the essay as they develop examples. The discourse flow is also affected by development – when claims and examples are developed, vocabulary associated with the main topic or stance is suspended, then revived when the discussion of the example is complete. In more detail, as the reader moves from sentence to sentence in an essay, he encounters concepts that either have been previously introduced or that are new to the discourse. When there is a repetition of ideas across the essay, the reader encounters the same concepts over and over again. On the other hand, when an example is introduced (e.g., to support a claim or to develop an idea), the reader encounters new concepts. As the example is developed further in successive sentences, the reader continues to move between concepts pertaining to this example and tends not to encounter concepts from the main topic of the essay. When the example is complete, the reader again encounters words pertaining to the main topic or stance. In a typical essay, this process repeats as the writer moves between presenting the main topic claims and detailed examples to develop the claims.

This type of development structure is also often evident in skillful narrative storytelling. After a character or situation is introduced, it may be described or fleshed out (via background information, vivid descriptions, etc.) before the writer returns to the main story line. The writer may briefly digress from the main goal of the story to add plot, character, and situation details.

2.1 Essay as a Graph

We construct graphs from essays by representing each content word in a sentence as a node in the graph. Content words are found by filtering out all words less than 4 characters long. This filtering removes function words such as "a" and "the" as well as punctuation. Other than this filtering, we use all the words in the essay, as is, to construct nodes. That is, we do not use lemmatization or synonymy to collapse similar words. Links are created from all nodes in one sentence to nodes in the following sentence to simulate the reader's movement from sentence to sentence. At this point, the nodes in the graph correspond to tokens in the essay, and any given node has as its neighbors all nodes from the previous and next sentence. Next, the nodes corresponding to the same word token are collapsed to a single node representing the word type, converting the graph of word tokens to a graph of word types. All links from collapsing nodes are added to their respective collapsed nodes. This collapsing process gives words that are repeated multiple times in the essay many more neighbors than words that occur

just once. Finally, multiple links between two nodes are collapsed into a single weighted link, where the weight is equal to the number of links between the two nodes.

We hypothesize that the structure of graphs constructed in this way can distinguish essays where there is no development from essays where ideas are developed in detail. For instance, if a writer repeats the same idea over and over, the corresponding graph will have nodes (corresponding to repeated ideas) that are heavily linked to other nodes. When an idea is not developed, the nodes corresponding to this idea will tend to have very few neighbors.

2.2 Graph Characteristics

Node degree The number of neighbors that a node has can indicate how it is connected to other concepts in the discourse. If a node has high degree, it is connected to more concepts, indicating that it has occurred in more contexts. Similarly, a node with low degree is sparsely connected to other nodes, corresponding to a concept that is mentioned in passing, without being developed. A large number of such sparsely connected nodes in an essay might indicate underdeveloped ideas.

PageRank PageRank (Brin and Page, 1998) emulates a "random surfer" on a graph. In our graph, this random surfer will move from concept to concept. PageRank is influenced by both the number of nodes and by the structure of the graph.

The number of nodes in the graph is influenced by the number of unique concepts in an essay. When an essay is characterized by development, we expect that more concepts get introduced into the discourse. On the other hand, mere repetition will not increase the number of nodes in the graph. The larger the graph, the lower the PageRank value assigned to individual nodes (if link characteristics remain constant).

Node count (i.e., unique concepts) remaining constant, the link structure of the graph is influenced by both development and repetition. When an example is well developed, the nodes corresponding to this development will form a well connected sub-graph. Consequently, there will be a shift in the distribution of PageRank values to reflect the fact that the random surfer is likely to visit these nodes with comparable probabilities as the nodes of the main theme or stance. When there is little development, PageRank values will be skewed and concentrated on a few nodes. Similarly, if an idea is repeated many times in the essay, the corresponding set of nodes will have very high connectivity, thus obtaining high PageRank values.

2.3 Graph Features

With the goal of encoding the graph characteristics described above, we developed 19 features: 6 features based on degree, one feature designed to capture the basic connectivity of the graph, and 12 features based on PageRank.

Features based on degree: We use three features to capture underdeveloped ideas: percentage of nodes in the graph with degree one (*perNodesDeg1*), percentage of nodes in the graph with degree two (*perNodesDeg2*), and percentage of nodes in the graph with degree three (*perNodesDeg3*). A node can get one neighbor if everything in the previous and next sentence (except for one word type) gets filtered out. Recall that, during graph construction, words with fewer than 4 characters are filtered out as non-content. For example, the sentence "I can say in the end it may be all good" will have only one resulting node in the graph.

Two features encode the degree of the two top most connected nodes in the graph: *degreeTop1* and *degreeTop2*. A third feature, *degreeMed*, is the median degree and encodes the general connectivity structure of the graph. Finally, the *starScore* feature aims to capture the extent to which the graph has a star-like appearance: the number of links from the node with the highest degree divided by the total number of links in the graph.

Features based on PageRank We have three features corresponding to the top three PageRank values in the graph (*prTop1*, *prTop2* and *prTop3*), and one feature, *prMed*, corresponding to the median PageRank value. As PageRank values tend to be very small numbers, we create negative log versions of the features: *prTop1Log*, *prTop2Log*, *prTop3Log* and *prMedLog*. Finally, the size of the graph can influence PageRank values. Specifically, given two nodes with the same type of linking structure, the node in a

larger graph (i.e., a graph with more nodes) will receive a smaller PageRank value. To mitigate this effect, we create a new set of PageRank-based features in which the values of the original PageRank features are multiplied by the total number of nodes in the graph, producing node count-normalized features. These features are: *prTop1NodeNorm*, *prTop2NodeNorm* and *prTop3NodeNorm* and *prMedNodeNorm*. While the 12 proposed features are correlated, in our experiments we use an Elastic Net learner (Zou and Hastie, 2005), which is designed for the case of groups of correlated features and performs automatic feature selection.

3 Data

To test our hypotheses, we carried out experiments in two different genres using two different datasets.

The first dataset, *persuasive*, is comprised of 1000 persuasive essays written by test takers from a high stakes assessment. The essays are holistically scored on an integer scale from 1 to 6 (with score point 6 assigned to excellent essays). The data distribution for each score point is as follows: score 0 = 0.3%; score 1 = 2%; score 2 = 12.3%; score 3 = 42.3%; score 4 = 33.6%; score 5 = 8.4%; score 6 = 1.1%. Essays were scored by expert human annotators trained in operational essay scoring. Holistic scores were assigned based on writing proficiency displayed in analyzing an issue topic. Scores take into account a number of factors such as organization, development, language fluency, use of proper grammar, mechanics, and language conventions. With this dataset, we investigate if and how development in the form of developing a persuasive stance is captured by our approach.

The second dataset, *narrative*, is obtained from the Criterion® writing evaluation system[3] and is comprised of 590 narrative essays written by middle and high school students. These essays were scored based on a rubric developed by the Smarter Balanced Assessment Consortium[4] for assessing narrative essays[5]. The rubrics assign three separate "trait" scores to each essay for the dimensions of Organization, Development, and Conventions. The scores for the Organization and Development traits are on an integer scale from 0 ("non-scorable") to 4 ("excellent"), and the score for Conventions is on a 3 point scale from 0 ('little or no command of conventions") to 2 ("excellent"). The data distribution is as follows:
Organization: score 0 = 4.9%; score 1 = 5.4%; score 2 = 27.3%; score 3 = 32.0%; score 4 = 30.4%.
Development: score 0 = 4.9%; score 1 = 6.6%; score 2 = 34.1%; score 3 = 28.5%; score 4 = 25.9%.
Conventions: score 0 = 9.7%; score 1 = 47.1%; score 2 = 43.2%.

Inter-annotator agreement is calculated using quadratic weighted kappa (QWK) (Cohen, 1968). QWK for the three dimensions was: Organization = 0.726; Development = 0.741; Conventions = 0.459. Correlation between the dimension scores (Pearson's r) are: Conventions and Organization = 0.41; Conventions and Development= 0.43; Organization and Development = 0.89. This dataset allows us to test if the proposed features work across different genres of writing. Additionally, as this data is scored separately along the development dimension, it allows us to test if our approach indeed captures development.

4 Experiments

We performed 10-fold cross validation experiments on the *persuasive* and *narrative* datasets to evaluate whether our proposed graph-based features help predict scores of essay quality and development. We used Gephi (Bastian et al., 2009) for generating the graph features listed in Section 2.3. All default parameters were used (epsilon, probability and number of iterations). We compared our proposed features with two baseline feature sets. Performance was measured with Quadratic Weighted Kappa (QWK) (Cohen, 1968), a common metric for measuring essay scoring performance. QWK scores are averaged across the 10 folds. We used a bootstrap significance test (Berg-Kirkpatrick et al., 2012; Zhang et al., 2004) to test if improvements over baselines are significant. We used the Elastic Net implementation from scikit-learn (Pedregosa et al., 2011)[6].

[3]https://criterion.ets.org/

[4]http://www.smarterbalanced.org

[5]http://sbac.portal.airast.org/wp-content/uploads /2015/03/Narrative-050814.pdf

[6]Results using Lasso and ridge regression were similar.

4.1 Baselines

e-rater Features (eRaterFeat) E-rater (Attali and Burstein, 2006), a state-of-the-art system for automatic essay scoring, uses a comprehensive set of features covering many aspects of writing quality, such as grammar, language use, mechanics, fluency, style, organization, and development. We use the e-rater feature set as a baseline and compare with our proposed features by employing the same learning algorithm over both feature sets. Development in eRaterFeat is captured by a feature that sums up the counts of the thesis, main points, supporting ideas, and conclusion elements in the essay, where the individual elements (e.g., supporting ideas) are identified in a separate step as described in Burstein et al. (2003).

We investigate how our graph-based features perform in comparison to the eRaterFeat feature set and in combination with it. Specifically, for holistic scoring, we hypothesize that graph features will improve over eRaterFeat performance by improving construct coverage. For trait scoring of development, we hypothesize that the graph-based features will perform as well or better than eRaterFeat since the graph-based feature set is designed to capture this trait.

Lexical Diversity Baseline (lexdiv) As discussed in Section 2, development and exemplification introduce new concepts into the discourse. A simple measure of lexical diversity is the type-token ratio, the ratio of the count of unique word types to the count of word tokens in the essay. To ascertain whether the graph features do more than capture lexical diversity in the essay, we employ the type-token ratio as a baseline. We refer to this feature as *lexdiv*. While we do not expect this single feature to perform well by itself for holistic scoring, our goal is to test if its combination with eRaterFeat produces similar or better performance to that achieved by the combination of graph features to eRaterFeat.

4.2 Results

All features (baseline and proposed) are extracted for all essays in both datasets. For the *persuasive* dataset, the features are used to predict the target holistic score. For the *narrative* dataset, the features are used to train three separate systems, one for predicting each trait score.

Table 1 reports the results of the feature sets (individually and in combination) for holistic scoring of persuasive essays. Statistically significant improvements over eRaterFeat (E) are indicated with * for $p < 0.05$ and ** for $p < 0.005$. All feature sets significantly outperform the lexdiv baseline ($p < 0.005$); to reduce clutter, we omit these significance results in the tables. At the level of individual feature sets, eRaterFeat is the best performer. This is expected, as eRaterFeat uses a suite of features capturing a variety of language proficiency measures that the holistic score represents, while the graph-based (and lexdiv) features capture only one aspect of writing. The lexdiv baseline is a poor performer, not substantially improving the performance of any system or system combination to which it is added.

Notably, when graph features are added to eRaterFeat features (eRaterFeat+graph), there is a significant boost in performance of 4 percentage points. This result is promising, as it indicates that graph features capture a part of the construct previously not covered in the state of the art.

Feature set	Holistic
Combinations	
all	0.73 (E**)
eRaterFeat+graph	0.73 (E**)
eRaterFeat+lexdiv	0.69 (E**)
graph+lexdiv	0.64
Individual	
graph	0.63
eRaterFeat	0.69
lexdiv	-0.01

Table 1: Performance (QWK) of feature sets on holistic scoring on the persuasive dataset. "all" = eRaterFeat+graph+lexdiv.

Feature set	Conv.	Org.	Dev.
Combinations			
all	0.34	0.50	0.57 (E**)
eRaterFeat+graph	0.36	0.51	0.57 (E**)
eRaterFeat+lexdiv	0.37	0.50	0.54
graph+lexdiv	0.23	0.51	0.55
Individual			
graph	0.24	0.51	0.56 (E*)
eRaterFeat	0.36	0.49	0.52
lexdiv	-0.01	0.11	0.12

Table 2: Performance (QWK) of feature sets on trait scoring (Conventions, Organization, Development) on the narrative dataset. "all" = eRaterFeat+graph+lexdiv.

Table 2 reports the performance of the different feature sets and their combinations on trait scoring of narrative essays. In general, the QWK values are lower than those in Table 1, suggesting that scoring of stories is a much harder task. As expected, eRaterFeat is the best performing individual feature set for Conventions; graph and lexdiv features are not designed to score this aspect of writing. There is a slight improvement in eRaterFeat performance when lexdiv is added (eRaterFeat+lexdiv), but this improvement is not significant over eRaterFeat individually.

For Organization the graph features are the best performing individual feature set. In combination with eRaterFeat (eRaterFeat+graph), QWK remains the same. However, in neither case is there a statistically significant improvement. Nevertheless, this result is interesting, as it shows that graph features, while intended to capture development, are able to predict Organization scores at least as well as a well-established system. We believe this is presumably because the Organization and Development traits are correlated.

Confirming our hypothesis, the graph feature set is the best performer for scoring Development. The graph feature set individually improves on eRaterFeat by 4 percentage points, a significant difference ($p < 0.05$). This result is notable as it shows that the graph features indeed capture the Development construct much more effectively than the features in eRaterFeat. When graph and eRaterFeat features are combined (eRaterFeat+graph), there is an additional 1 point improvement over eRaterFeat alone.

5 Analysis

Our results provide strong evidence that the graph-based feature set is useful for essay scoring. In this section, we examine in more detail how the graph features are related to the Development score from the *narrative* dataset.

5.1 Correlation Analysis

We employed correlation analysis to study the relationship of each of our graph features with Development trait scores. Table 3 lists the correlation (Pearson's *r*) between each degree-based graph feature and its correlation with Development scores. All features except *perNodeDeg1* and *perNodeDeg3* have medium (> 0.3) to low (between 0.1 and 0.3) correlation with scores (significant at $p < 0.005$). The features encoding values of the higher degree of nodes, *degreeTop1* and *degreeTop2*, are positively correlated with scores. This indicates that the higher the connection of central ideas to other concepts, the better the development of the essay. Interestingly, *starScore* is negatively correlated with Development scores, suggesting that essays where most of the links represent connections to a single idea have little or no development.

Feature set	Corr.	Partial Corr.
degreeTop1	0.41*	0.00
degreeTop2	0.39*	0.01
degreeMed	0.24*	0.09
perNodesDeg3	-0.04	0.05
perNodesDeg1	-0.07	-0.03
perNodesDeg2	-0.12*	-0.03
starScore	-0.31*	-0.12 *

Table 3: Correlation and partial correlation (controlling for length) of node degree-based features with Development trait scores. * = significant at $p < 0.005$.

Feature set	Corr.	Partial Corr.
prMedLog	0.63 *	0.35 *
prTop3Log	0.57 *	0.31 *
prTop2Log	0.56 *	0.30 *
prTop1Log	0.52 *	0.27 *
prTop3NodeNorm	0.49 *	0.09
prTop2NodeNorm	0.44 *	0.07
prTop1NodeNorm	0.41 *	0.05
prMedNodeNorm	0.06	0.10
prTop1	-0.33 *	-0.20 *
prTop2	-0.41 *	-0.22 *
prTop3	-0.48 *	-0.23 *
prMed	-0.48 *	-0.26 *

Table 4: Correlation and partial correlation (controlling for length) of PageRank-based features with Development trait scores. * = significant at $p < 0.005$.

In Table 4 we see that all features based on PageRank, with the exception of *prMedNodeNorm*, are moderately or highly correlated with development ($p < 0.005$). All of the raw PageRank features show a negative correlation with Development scores, which implies that the lower the highest PageRank values, better the development of the essay. This happens when: (1) there are more nodes in the graph (indicating a large number of concepts in the essay); (2) the links for a graph are more distributed. Such decentralization indicates that the writer emphasizes the detailed development of multiple concepts and tends not to repeat a single concept. The negative log and nodeNorm counterparts of the raw features are naturally positively correlated with the scores.

Indeed, it is impossible to develop an idea or character without using additional words. As a result, longer essays may receive higher scores. The graph-based features are also naturally affected by essay length. Nevertheless, it is informative to investigate if a feature has predictive value even after the effect of essay length is removed. Looking at the partial correlations (accounting for length) in Tables 3 and 4 (Partial Corr. column), the node degree-based *starScore* feature and all PageRank-based features with the exception of *nodeNorm features retain their respective correlations with Development scores ($p < 0.005$), albeit to a smaller degree.

5.2 Qualitative Analysis

To explore whether the graph features indeed encode the intuitions discussed in Section 2, we modified the contents of the sample essay discussed in Example 1 in various ways to simulate a lack of development and repetition of ideas. This essay is very similar to the persuasive essays in our dataset[7]. In each simulated essay, we maintained the length of the original as much as possible so that we could see the effect of the parameters we varied. We simulated different scenarios:

No development (no-dev): This simulates a scenario where the writer does not expand beyond the main stance or theme. We replaced the second and third paragraphs of the essay with the first paragraph (which contains the main stance).

Vague development (vague-dev): This simulates a scenario where the writer provides vague development for the main stance, but does not fully elaborate on the main claims. We deleted the examples in the second and third paragraphs (i.e., the text spans "Take Australia ..." and "For instance ...", respectively). Note that this results in an essay that is smaller than the original.

Vague development with repetition (vague-dev-rep): This simulates a scenario where the writer reiterates an idea multiple times, without substantially developing it. Similar to vague-dev, the text of the examples are deleted, but in this case they are replaced by the first sentence of the corresponding paragraph ("Firstly, ..." and "Secondly, ...", respectively). The number of words approximately matches the original, so that the simulated essay is similar in length to the original.

essay	length	degreeTop1	degreeTop2	starScore	prTop1Log	prTop2Log	prTop3Log	prMedLog
original	2025	67	65	0.073	3.460	3.485	3.813	4.742
vague-dev-rep	2033	34	29	0.102	3.093	3.373	3.400	3.862
vague-dev	1005	33	28	0.115	3.031	3.204	3.218	3.929
no-dev	2026	24	24	0.155	2.697	2.697	3.200	3.200

Table 5: Graph-based feature values under different simulated conditions of essay development. *length* = number of characters.

Table 5 shows several graph features computed from the different scenarios (we reproduce only a subset of features due to space limitations). Notice that all features show sensitivity to the type of development. The degree-based features (*degreeTop1*, *degreeTop2*) and PageRank-based features (*prTop1Log*, *prTop2Log*, *prTop3Log*) rank the essays from worst to best in the order: no-dev, vague-dev, vague-dev-rep, original. These features rank the vague development (vague-dev) essay higher than no development (no-dev) essay, even though the latter is a longer essay. The *starScore* feature ranks the essays in the same order, keeping in mind that this feature is negatively correlated with Development scores. The *prMedLog* feature, which is most highly correlated Development scores in Table 4, ranks vague develop-

[7]Due to the proprietary nature of our datasets, we are not able to reproduce actual essays.

ment (vague-dev) higher than vague development with additional repetition (vague-dev-rep), penalizing repetition that contains the same amount of development.

6 Related Work

Entity-based approaches to discourse coherence focus on the distribution of entities in discourse, under the assumption that a patterned focus on discourse entities is indicative of greater topic continuity. Barzilay and Lapata (2008) formalize these intuitions to predict coherence. Their work has been extended in a number of ways, such as adding sentence-to-sentence sequences based on predicted discourse relations (Lin et al., 2011; Feng et al., 2014), adding information about topics (Elsner and Charniak, 2011), and incorporating other linguistic features (Eisner and Charniak, 2011). Burstein et al. (2010) employ this idea for evaluating coherence in student essays. Morris and Hirst (1991) connect highly related words in the discourse to create chains, which indicate cohesion of ideas in text. This idea is employed in the context of essay evaluation by Somasundaran et al. (2014) to capture discourse coherence quality. Our approach does not evaluate discourse coherence and does not employ cohesion-based relations. Rather, we use discourse-based proximity relations to evaluate the development of ideas.

Hearst's TextTiling algorithm (Hearst, 1997) captures some elaboration of subtopics by measuring the differences of word profiles between discourse segments. Our work also captures the use of subtopics, but we do not address the issue of text segmentation.

Guinaudeau and Strube (2013) build a graph representation of entities and sentences to predict sentence ordering, summary coherence, and readability. Mesgar and Strube (2015) extend this graph representation by adding rhetorical relations, and use subgraph mining techniques improve readability classification accuracy. Petersen et al. (2015) leverage the graph representation of Guinaudeau and Strube to compute a variety of graph-based metrics to measure coherence, including PageRank, and show that these features improve the relevance of IR results. In contrast with these lines of work, our graph structure only contains nodes for words, not sentences, and edges are inserted for words in consecutive sentences. Additionally, our goal is to predict essay scores, focusing on capturing the development trait.

In argumentative writing, it has been observed that support and elaboration plays an important role in overall comprehensibility (Garing, 2014; Duterte-Angeles, 2005). Miltsakaki and Kukich (2004) developed the concept of "rough shifts" in discourse and found that more rough shifts and less elaboration negatively correlated with essay score. In related work, O'Rourke et al. (2011) develop a method to measure the semantic or topic flow of essays. Rahimi et al. (2015) tie together the concepts of discourse coherence, essay organization, and argumentation for scoring short answers. Farra et al. (2015) evaluate whether a given opinion is topically relevant to the persuasive goal in student essays. These works are complementary to ours: while these approaches do not penalize repetition and lack of development, our approach is not sensitive to topicality and organization.

Stab and Gurevych (2014b) propose methods for identifying argumentation components in persuasive essays, including claims and premises. Peldszus and Stede (2015) demonstrate how the resulting graphs of argument components and their relations can be parsed into discourse structure. Song et al. (2014) as well as Ghosh et al. (2016) explore features specific to argumentation and argumentation schemes. While our features capture elaboration in persuasive essays, they are not tied to the specifics of argumentation, and we demonstrate the applicability of these features across genres. Along similar lines, previous work in narrative evaluation (Evanini and Wang, 2013; Hassanali et al., 2013; Somasundaran et al., 2015) explores features that are complementary to ours.

Within the area of predicting text quality more generally, researchers have focused on readability and text complexity using a variety of features such as entity grid features (Pitler and Nenkova, 2008), language model-based features (Kate et al., 2010; Feng et al., 2010), grade-level features (Qumsiyeh and Ng, 2011). Jiang et al. (2015) uses a graph representation to predict readability.

7 Conclusion

Evaluation of development is a relatively less investigated aspect in writing evaluation. In this work, we examined whether writing development can be represented in the properties of simple graphs computed

from the structure of essays. From our graph representation of word types and sentence-adjacency links, to capture development, we computed both graph structural features based on node degree and PageRank features. To the best of our knowledge, this is the first work on employing graphs to capture this aspect of writing.

We performed experiments to show that our approach complements previously established features in essay scoring. Specifically, we demonstrated that our feature set significantly improves on the state of the art for holistic essay scoring. For trait scoring, we showed that our features are more effective in capturing writing development than existing feature sets. As part of this investigation, we also explored automated scoring of narratives, a relatively less explored genre of student writing. Results across genres (persuasive and narrative) and scoring granularity (holistic and trait) demonstrated that graph-based features are effective at capturing development in writing.

In future work we plan to explore ways to incorporate and represent more information in the graphs, such as discourse relations, morphological variants, equivalence classes of semantically similar words and synonyms, as well as examine performance trends on publicly available essay datasets.

References

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater v. 2.0. *Journal of Technology, Learning, and Assessment*, 4:3.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. 2009. Gephi: An open source software for exploring and manipulating networks. In *Proceedings of International AAAI Conference on Weblogs and Social Media*.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005. Association for Computational Linguistics.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of Seventh International World-Wide Web Conference (WWW 1998)*.

Jill Burstein, Daniel Marcu, and Kevin Knight. 2003. Finding the write stuff: Automatic identification of discourse structure in student essays. *IEEE Intelligent Systems*, 18(1):32–39.

Jill Burstein, Joel Tetreault, and Slava Andreyev. 2010. Using entity-based features to model coherence in student essays. In *Proceedings of Human language technologies: The 2010 annual conference of the North American chapter of the Association for Computational Linguistics*, pages 681–684. Association for Computational Linguistics.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4):213.

S Duterte-Angeles. 2005. *Coherence in the argumentative essays of ADZU college freshmen: A textual analysis of writing quality*. Ph.D. thesis.

Micha Eisner and Eugene Charniak. 2011. Extending the Entity Grid with Entity-specific Features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 125–129. Association for Computational Linguistics.

Micha Elsner and Eugene Charniak. 2011. Disentangling Chat with Local Coherence Models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1179–1189. Association for Computational Linguistics.

Keelan Evanini and Xinhao Wang. 2013. Automated speech scoring for non-native middle school students with multiple task types. In *Proceedings of Interspeech*, pages 2435–2439.

Noura Farra, Swapna Somasundaran, and Jill Burstein. 2015. Scoring Persuasive Essays Using Opinions and their Targets. In *Proceedings of the 10th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 64–74.

Lijun Feng, Martin Jansche, Matt Huenerfauth, and Noemie Elhadad. 2010. A Comparison of Features for Automatic Readability Assessment. In *Proceedings of the 23rd International Conference on Computational Linguistics*.

Vanessa Wei Feng, Ziheng Lin, Graeme Hirst, and Singapore Press Holdings. 2014. The impact of deep hierarchical discourse structures in the evaluation of text coherence. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 940–949.

Alphie G Garing. 2014. Coherence in Argumentative Essays of First-Year College of Liberal Arts Students at De La Salle University. In *DLSU Research Congress*.

Debanjan Ghosh, Aquila Khanam, Yubo Han, and Smaranda Muresan. 2016. Coarse-grained argumentation features for scoring persuasive essays. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 549–554.

Camille Guinaudeau and Michael Strube. 2013. Graph-based Local Coherence Modeling. In *Proceedings of the Association for Computational Linguistics*, pages 93–103.

Khairun-nisa Hassanali, Yang Liu, and Thamar Solorio. 2013. Using Latent Dirichlet Allocation for child narrative analysis. In *Proceedings of the 2013 Workshop on Biomedical Natural Language Processing*. Association for Computational Linguistics.

Marti A. Hearst. 1997. TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages. *Computational Linguistics*, 23:33–64.

Zhiwei Jiang. 2015. A Graph-based Readability Assessment Method using Word Coupling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 411–420.

Rohit J Kate, Xiaoqiang Luo, Siddharth Patwardhan, Martin Franz, Radu Florian, Raymond J Mooney, Salim Roukos, and Chris Welty. 2010. Learning to predict readability using diverse linguistic features. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 546–554. Association for Computational Linguistics.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 997–1006.

Mohsen Mesgar and Michael Strube. 2015. Graph-based Coherence Modeling for Assessing Readability. In *Proceedings of Lexical and Computational Semantics (*SEM 2015)*, pages 309–318.

Eleni Miltsakaki and Karen Kukich. 2004. Evaluation of text coherence for electronic essay scoring systems. *Natural Language Engineering*, 10(01):25–55.

Jane Morris and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational linguistics*, 17(1):21–48.

Stephen T. O'Rourke, Rafael A. Calvo, and Danielle S. McNamara. 2011. Visualizing Topic Flow in Students' Essays. *Educational Technology & Society*, 14:4–15.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in MST-style discourse parsing for argumentation mining. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 938–948.

Casper Petersen, Christina Lioma, Jakob Grue Simonsen, and Birger Larsen. 2015. Entropy and Graph Based Modelling of Document Coherence using Discourse Entities: An Application to IR. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 186–195.

Rani Qumsiyeh and Yiu-Kai Ng. 2011. ReadAid: A Robust and Fully-Automated Readability Assessment Tool. In *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence*, pages 539–546. IEEE.

Zahra Rahimi, Diane Litman, Elaine Wang, and Richard Correnti. 2015. Incorporating Coherence of Topics as a Criterion in Automatic Response-to-Text Assessment of the Organization of Writing. In *Proceedings of the 10th Workshop on Innovative Use of NLP for Building Educational Applications*.

Vasile Rus and Nobal Niraula. 2012. Automated detection of local coherence in short argumentative essays based on centering theory. In *Computational Linguistics and Intelligent Text Processing*, pages 450–461. Springer.

Mark D Shermis and Jill Burstein. 2013. *Handbook of automated essay evaluation: Current applications and new directions*. Routledge.

Swapna Somasundaran, Jill Burstein, and Martin Chodorow. 2014. Lexical Chaining for Measuring Discourse Coherence Quality in Test-taker Essays. In *Proceedings of the 25th International Conference on Computational Linguistics*.

Swapna Somasundaran, Chong Min Lee, Martin Chodorow, and Xinhao Wang. 2015. Automated scoring of picture-based story narration. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 42–48.

Yi Song, Michael Heilman, Beata Beigman Klebanov, and Paul Deane. 2014. Applying argumentation schemes for essay scoring. In *Proceedings of the First Workshop on Argumentation Mining*, pages 69–78, Baltimore, Maryland, June. Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2014a. Annotating argument components and relations in persuasive essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1501–1510.

Christian Stab and Iryna Gurevych. 2014b. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 46–56.

Ying Zhang, Stephan Vogel, and Alex Waibel. 2004. Interpreting bleu/nist scores: How much improvement do we need to have a better system? In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC)*, pages 2051–2054.

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

Selective Co-occurrences for Word-Emotion Association

Ameeta Agrawal and Aijun An
Department of Electrical Engineering and Computer Science
York University, Toronto, Canada
{ameeta,aan}@cse.yorku.ca

Abstract

Emotion classification from text typically requires some degree of word-emotion association, either gathered from pre-existing emotion lexicons or calculated using some measure of semantic relatedness. Most emotion lexicons contain a fixed number of emotion categories and provide a rather limited coverage. Current measures of computing semantic relatedness, on the other hand, do not adapt well to the specific task of word-emotion association and therefore, yield average results. In this work, we propose an unsupervised method of learning word-emotion association from large text corpora, called **Se**lective **Co**-occurrences (SECO), by leveraging the property of mutual exclusivity generally exhibited by emotions. Extensive evaluation, using just one seed word per emotion category, indicates the effectiveness of the proposed approach over three emotion lexicons and two state-of-the-art models of word embeddings on three datasets from different domains.

1 Introduction

Emotion detection from text is the task of identifying emotions from natural language data such as user reviews, blogs, news articles, etc. (Alm et al., 2005; Aman and Szpakowicz, 2007). Although there is no strict definition for emotion, most researchers agree that it is a particular feeling that characterizes the state of mind such as happiness, anger, sadness and so on. Emotion analysis is extremely popular in the field of market research, where brands tap into their customer base by analyzing user-generated data, readily available nowadays due to the rapid growth of social media (De Bondt et al., 2013; Shrum et al., 2013). Mohammad and Turney (2013) present a more extensive list of applications.

Emotion detection from text typically requires some degree of *word-emotion association*, i.e., knowing which words are more appropriately associated with which emotions. For example, the word *"accident"* can be considered associated with the emotion SADNESS. This association can be obtained from a pre-compiled emotion lexicon or calculated using some measure of semantic relatedness. The currently available manually annotated emotion lexicons tend to be restricted in size due to the expensive process of human annotation. This limited coverage leads to the undesirable effect of leaving too many words unassociated with any emotion category. Moreover, most lexicons contain a small fixed set of emotions which is unsuitable for a larger (Du et al., 2014) or a newly defined set of emotions (Facebook, 2016). On the other hand, while automatically computing word-emotion association scores from text corpora possibly provides a better coverage and more flexibility, the current techniques are ill-suited to the task of emotion detection and therefore, tend to yield average results.

To address these issues, in this work, we propose an unsupervised method of learning word-emotion association scores from text corpora, which we call **Se**lective **Co**-occcurences (SECO). By modifying conventional co-occurrence-based methods, we compute a uni-directional asymmetric association between a given word and an emotion seed word. The proposed approach is found to be better at capturing the association between words and emotions than general purpose measures. Extensive evaluation of word-emotion association scores derived from two large text corpora, Wikipedia and Amazon reviews,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1579–1590, Osaka, Japan, December 11-17 2016.

on three emotion datasets from very diverse domains demonstrates the effectiveness of employing selective co-occurrences. The proposed approach is particularly interesting as it requires no training data and can be applied to a flexible number of emotion categories.

The remainder of this paper is organized as follows. In Section 2, we survey the related work. In Section 3, we describe the learning of the word-emotion association scores and their use in the task of emotion classification. Section 4 describes the evaluation setup, while Section 5 analyzes the experimental results. Lastly, Section 6 concludes the paper with a brief look at future work.

2 Related Work

In this section, we present the existing emotion lexicons, manually annotated as well as those created using supervised machine learning, and also discuss the measures of semantic relatedness that have been previously employed to derive word-emotion association for emotion classification.

2.1 Emotion Lexicons

2.1.1 Manually Annotated Lexicons

One of the earliest and most popular emotion resources is the WordNet Affect (Strapparava and Valitutti, 2004), developed by manually labelling about 1,314 synsets with one or more of Ekman's (1992) six basic emotions. Using crowd-sourcing, one of the largest manually annotated emotion lexicons created to date is the NRC Emotion Lexicon (EmoLex) (Mohammad and Turney, 2010; Mohammad and Turney, 2013). It contains about 14,200 unigrams annotated with one or more of Plutchik's eight emotions (Plutchik, 2001). Another manually created lexicon, the Affect database (Neviarouskaya et al., 2007), contains a total of 2,440 entries (emoticons, acronyms, words and modifiers) annotated by three annotators using nine emotion labels as well as their intensities. Considering the fundamental role played by lexical resources in the task of emotion detection, the current options available due to manual lexicons seem rather limited in their coverage. Human annotation, including crowd-sourcing, requires considerable lexicographic expertise, time and effort. An alternative approach involves creating such lexical resources automatically.

2.1.2 Automatically Acquired Lexicons

More recently, DepecheMood (Staiano and Guerini, 2014), was created using supervised training by applying distributional semantics to a dataset of crowd-annotated news articles. This lexicon consists of 37,000 words and their emotion scores across seven emotions. A different approach of generating an emotion lexicon (18,000 words and 8 emotions) involves using the Google n-grams corpus to expand an existing, smaller human-annotated lexicon such as the EmoLex (Perrie et al., 2013). Although these approaches cover a larger vocabulary, they are limited to the emotion categories of the source corpus or the lexicon which they use for training.

2.2 Measures of Semantic Relatedness

Statistical approaches that leverage large text corpora provide an alternative way of acquiring word-emotion association scores, which can remedy the problem of unseen vocabulary to a large extent. As these approaches employ just a handful of emotion seed words to initialize the process, they are also applicable to a flexible number of emotion categories. Many models of computing word semantic relatedness exist, from the traditional count-based methods such as Pointwise Mutual Information (PMI) to the more recent neural-network inspired models of word embeddings such as Continuous Bag of Words (CBOW). While the models differ in their algorithms, they are fundamentally based on the intuitive assumption that co-occurring words tend to be related to each other.

Previously, PMI has been used to classify emotions in news headlines, where the probabilities of words were calculated using statistics collected from three search engines (Kozareva et al., 2007). Wikipedia has also proven a useful resource for calculating word frequencies using PMI to obtain word-emotion association scores (Agrawal and An, 2012). Alternatively, PMI has been used to first build an emotion lexicon which is then used for classification (Yang et al., 2007). Latent Semantic Analysis (LSA)

(Deerwester et al., 1990), which analyzes the statistical relationships among words in a corpus using Singular Value Decomposition (SVD) dimensionality reduction technique, was used to calculate word-emotion association scores to classify news headlines (Strapparava and Mihalcea, 2008).

Despite, and perhaps because of, its simplicity, PMI (Church and Hanks, 1990) has long been a popular measure of semantic relatedness. It estimates the similarity between two terms x and y as $PMI\,(x,y) = \log\left(\frac{p(x,y)}{p(x)p(y)}\right)$, where $p\,(x,y)$ is the probability that words x and y co-occur within a window of specific length, and $p\,(x)$ and $p\,(y)$ are the individual probabilities of word x and word y, respectively, in the corpus. To overcome a few well-known shortcomings of PMI (i.e., low frequency events receiving relatively high scores, lack of a fixed upper bound), Bouma (2009) proposed a normalized version of PMI (NPMI), where $NPMI\,(x,y) = \frac{PMI(x,y)}{-\log\,p(x,y)}$, with fixed orientation values: when two words only occur together, NPMI$(x,y) = 1$; when they are distributed as expected under independence, NPMI$(x,y) = 0$; and, when they occur separately but not together, NPMI$(x,y) = -1$.

More recently proposed neural-network based approaches, such as Continuous Bag-of-Words (CBOW) and Skip-Gram (SG) (Mikolov et al., 2013b; Mikolov et al., 2013a) output word embeddings which can then be used to compute the similarity between two words by calculating their cosine similarity. These methods implicitly factorize a word-context matrix whose cell values are in fact shifted PMI (Levy and Goldberg, 2014) and have outperformed traditional count-based methods such as PMI on many similarity-oriented tasks (Marco Baroni, Georgiana Dinu, 2014).

While these measures of semantic relatedness provide promising results on most word similarity tasks, they do not adapt well to emotion detection. We build upon these latest advancements by proposing a variant that is more suitable to the task at hand.

3 Word-emotion Association for Emotion Classification

Given a sentence s and a set $E = \{e_1, e_2, \ldots, e_g\}$ of g emotion categories, the objective is to label s with the best possible emotion $e \in E$. We first discuss our proposed method for deriving the word-emotion association scores between a word and an emotion category, and then use these scores to obtain an emotion label for each sentence.

3.1 Learning Word-emotion Association

Let $W = \{w_1, w_2, \ldots, w_n\}$ be a set of n cue words in an input sentence s, where $W \subset s$. A cue word is defined as any word within a sentence that could have some emotional connotation. Usually, these are the nouns, adjectives, verbs and adverbs. Let $E = \{e_1, e_2, \ldots, e_g\}$ be a set of g emotion categories. Each emotion category $e_j \subset E$ is represented by one or more emotion seed words. Let T be the set of all the seed words for all the emotion categories, $T_j \subset \mathrm{T}$ and $T_j = \{t_{j_1}, t_{j_2}, \ldots, t_{j_m}\}$ be the set of m emotion seed words for each emotion category $e_j \subset E$.

As an illustration, consider the sentence "*We are going to a party tonight*". Here, the set of cue words includes $W = \{going, party, tonight\}$. If the classification scheme follows, for example, Ekman's (1992) model of emotions, then $E = \{anger, disgust, fear, happiness, sadness, surprise\}$. The set of seed words for HAPPINESS could be $T_{happiness} = \{happy, joy, ...\}$ and ANGER's seed words could be $T_{anger} = \{angry, mad, \ldots\}$.

We adopt the $\rightarrow$ symbol to denote the association between a cue word w and an emotion seed word t. The first step is to derive the association scores between a cue word and every seed word, e.g., $Assoc.\,(party \rightarrow joy)$, $Assoc.\,(party \rightarrow angry)$. Since our main goal is to acquire *association scores for emotion classification*, the design choices for our proposed measure of learning word-emotion association scores, SECO, are largely motivated by the following observations:

- The intrinsic process of annotating an emotion dataset as well as that of classifying it is uni-directional, i.e., given a word or a sentence, the task is to label it with the emotion it evokes the most.
- Although expressions of emotions can sometimes be fuzzy, most words primarily evoke only one emotion in a particular context, i.e., the emotion categories are, for the most part, mutually exclusive.

In fact, in the emotion lexicon WordNet Affect (Strapparava and Valitutti, 2004), which has words annotated with more than one emotion, about 98.7% of the terms are labelled with just one emotion.

- Most importantly, unlike other word relatedness tasks, the second half of the association pair (i.e., emotion seed words) in this particular task are known in advance.

These observations lead us to hypothesize that an asymmetric measure of association, where a word's association with an emotion seed word (and therefore by extension, with an emotion category) may be more meaningful than trying to achieve a symmetric bi-directional association between the two. In fact, as Tversky (1977) notes, certain linguistic relationships are characteristically asymmetric . In one experiment to list the first meaningfully related word that comes to mind, for the cue word *fear*, 24% of the participants answered *scared*, while only 9% of them recalled *fear* when given the cue word *scared*, suggesting an inherent asymmetry in word associations (Altarriba et al., 1999).

As noted earlier, traditional co-occurrence based models of semantic relatedness consider two words as co-occurring when both the words appear within a specific window of text, no matter how far they are from each other. In reality, nearer words have been found to exhibit stronger relationships (Beeferman et al., 1997).

In similar essence, emotions generally exhibit the property of mutual exclusivity and therefore, we propose the concept of selective co-occurrences (SECO), where a cue word is considered as co-occurring with only one emotion's seed words within any particular window of text. Consider Fig. 1 containing an example window of text, the cue word "*party*", and two seed words "*angry*" and "*happy*", representing the two different emotion categories respectively. To apply selective co-occurrences, the cue word "*party*" is considered co-occurring with either "*angry*" or "*happy*", not both.

> Theater critic Michael Riedel (playing himself) also shows up, uninvited. Ivy is put out by this and gets **angry** at Michael about it. We hear but don't see Ivy singing "Bittersweet Symphony" at her **party**. Derek then walks in and gives her a present and wishes her **happy** birthday.

Figure 1: Sample window of text containing cue and seed words

When a context window contains multiple seed words from multiple emotion categories, three possible settings for selecting the most appropriate seed word as co-occurring with the cue word can be explored:

- *nearest* (SECO-NEAR): This is the most intuitive option, where the nearest seed word to the cue word is selected. For example, "*happy*" is counted as co-occurring with "*party*"; "*angry*" is ignored.
- *preceding* (SECO-PREC): To account for any positional predisposition, this variant considers only the closest preceding seed word to the cue word. For example, "*angry*" is considered as co-occurring with "*party*"; "*happy*" is ignored.
- *following* (SECO-FOLL): Similarly, this variant considers only the closest seed word that follows the cue word as co-occurring together. For example, "*happy*" is considered as co-occurring with "*party*"; "*angry*" is ignored.

The selective counting of the seed word's co-occurrence frequency with the cue word is what essentially makes our association measure asymmetric and therefore, the order of the cue and seed word in the association equation cannot be interchanged. That is, $Assoc.(w \rightarrow t)$ denotes the association between a cue word w and a seed word t, and $Assoc.(w \rightarrow t) \neq Assoc.(t \rightarrow w)$. Technically, SECO is applicable to any traditional co-occurrence-based word association measure that estimates the relatedness between two words by computing some function of the words' frequencies. To this end, we adopt three popular co-occurrence association measures, namely NPMI (Bouma, 2009), Dice (1945) and Jaccard (1912), into their SECO counterparts, SECO-NPMI, SECO-Dice and SECO-Jaccard, respectively.

When applying SECO, in a corpus of M words, $\#(w, t)$ denotes the number of times a cue word w co-occurs mutually exclusively with any one emotion seed word $t \subset \mathrm{T}$ within a context window of size

k, where k is the maximum distance between those two words; $\#(w)$ and $\#(t)$ denote the frequencies of w and t, respectively.

SECO-NPMI The normalized SECO-NPMI between w and t, within the range of $[-1, 1]$, is:

$$SECO\text{-}NPMI(w \rightarrow t) = \frac{\log\left(M \frac{\#(w,t)}{\#(w)\#(t)}\right)}{\log\left(\frac{M}{\#(w,t)}\right)}. \quad (1)$$

SECO-Dice Similarly, SECO-Dice between w and t is computed as:

$$SECO\text{-}Dice(w \rightarrow t) = \frac{2 \times \#(w,t)}{\#(w) + \#(t)}. \quad (2)$$

SECO-Jaccard Lastly, one of the earliest co-occurrence associations measures, Jaccard, can be transformed as follows:

$$SECO\text{-}Jaccard(w \rightarrow t) = \frac{\#(w,t)}{\#(w) + \#(t) - \#(w,t)}. \quad (3)$$

Empirical data shows that the association between words decays exponentially (Beeferman et al., 1997) and this property has been successfully exploited by adding a decaying factor which allows words that co-occur nearer to each other to be more related (Gao et al., 2002; Sahlgren, 2006; Brosseau-Villeneuve et al., 2010; Mikolov et al., 2013a). Drawing from previous research, we also apply a context weighting scheme whereby a seed word is linearly weighted according to its distance from the cue word as follows: in a window of size k, the n^{th} word from the cue word is weighted by the function $\frac{k-n+1}{k}$. For example, in a window of 5, the first word next to the cue word is weighted by $\frac{5}{5}$, while the fourth word away is of weight $\frac{2}{5}$. In other words, as the distance between two words increases, their weighted association score decreases.

Finally, the word-emotion association between w and an emotion category e_j is obtained by calculating the average mean of the association scores between w and all the seed words of e_j as:

$$Assoc.(w \rightarrow e_j) = \frac{1}{m} \sum_{k=1}^{m} Assoc.(w \rightarrow t_{j_k}) \quad (4)$$

where $Assoc.$ is any association measure such as SECO-NPMI.

3.2 Classifying Sentence Emotion

For each word w, its emotion vector ϕ_w is denoted as:

$$\phi_w = \langle Assoc.(w \rightarrow e_1), Assoc.(w \rightarrow e_2), \ldots, Assoc.(w \rightarrow e_g) \rangle$$

and the emotion vector ϕ_s of sentence s is obtained by averaging the emotion vectors of all its n cue words as $\phi_s = \frac{1}{n} \sum_{i=1}^{n} \phi_{w_i}$. Finally, the sentence is labelled with the emotion category $e \subset E$ with the maximum value in ϕ_s.

4 Evaluation Setup

In this section we describe the evaluation datasets, the text corpus used for learning the word-emotion association scores and the evaluation metric for this task.

4.1 Evaluation Datasets

Below described are the three popular emotion evaluation datasets, with some sample sentences presented in Table 1 and their summarized statistics in Table 2.

Aman: Consisting of highly informal blog data, this dataset includes 1290 sentences annotated with one of six emotions: *anger, disgust, fear, joy, sadness and surprise* (Aman and Szpakowicz, 2007).

Alm: Emotions are particularly significant in the literary genre of fairy tales and this dataset contains 1207 high-agreement sentences (i.e., all four annotators agreed with the same label) marked with one of five emotions: *angry-disgusted, fearful, happy, sad and surprised* (Alm, 2008).

Aman	Do you people not listen to the news or what? I had a blast in california hanging out with my family and friends.	*anger* *happiness*
Alm	Oh! cried the devil, "what are you doing?" Ha! what are you doing? cried the devil angrily.	*surprised* *angry-disgusted*
ISEAR	When I saw a ghost. Slaughtering of animals.	*fear* *disgust*

Table 1: Sample sentences from evaluation datasets

	ag	***dg***	***fr***	***hp***	***sd***	***sp***	**total**
Aman	179	172	115	536	173	115	1290
Alm	218		166	445	264	114	1207
ISEAR	1085	1072	1086	1089	1080	-	5412

Table 2: Details of evaluation datasets

ISEAR: Developed for studying the relationships among emotions and cultures, this corpus contains experiences evoking seven emotions: *anger, disgust, fear, joy, sadness, shame* and *guilt*, resulting in a total of 5412 sentences[1]. To the best of our knowledge, no existing lexicon contains *shame* or *guilt* categories, and therefore methods that depend on emotion lexicons cannot correctly classify sentences belonging to these emotions. However, unsupervised approaches such as ours, which can be initialized with as little as one seed word per emotion category, are easily applicable to such datasets.

4.2 Text Corpora

We derive the word-emotion association scores from two large text corpora of different domains, which are pre-processed by: a) converting to lowercase; b) stripping off all non-alphanumeric characters; c) removing stopwords; d) stemming, and e) removing words that occur less than 5 times in the corpus.

Wikipedia[2]: The large publicly available corpus of Wikipedia mainly consists of formal language structured text articles considered to be more "objective" in nature. Our clean corpus contains approximately 918.5 million tokens, with each article on one line.

Amazon: The text of all the product reviews, mostly consisting of informal language makes up our second corpus, considered to be of more "emotional" type. This data was extracted from the aggressively deduplicated dataset (McAuley et al., 2015), which contains 82.83 million product reviews from Amazon, spanning May 1996 - July 2014. Our clean corpus contains more than 3 billion tokens (three times the size of Wikipedia corpus), with one review per line.

4.3 Evaluation Metric

Following prior studies, we calculate the F-score for each emotion class e, where F-score is the harmonic mean of *precision* and *recall*, defined as $2\left(\frac{precision \times recall}{precision + recall}\right)$. *Precision* is the number of sentences correctly labeled as belonging to the class e divided by the total number of sentences labeled as belonging to e, and *recall* is the number of sentences correctly labeled as belonging to e divided by the total number of sentences that actually belong to e. We report the average F-score over all the classes.

5 Experiments and Analysis

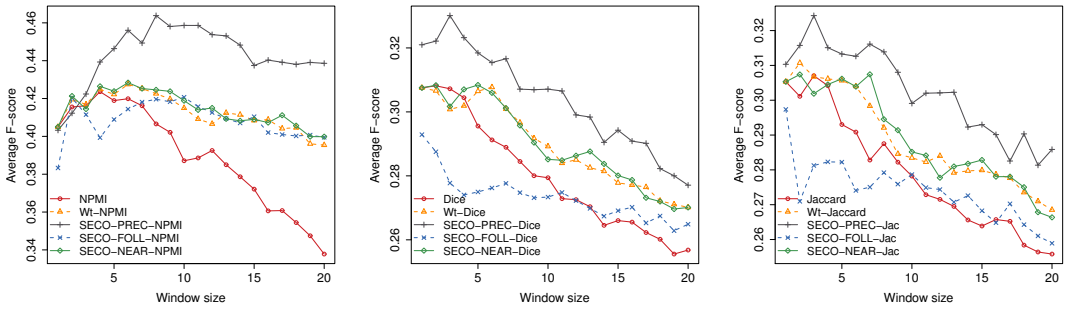
In what follows, we evaluate the performance of the proposed approach in several experiments and discuss their results.

5.1 How effective is selective co-occurrence?

We test the performance of the three proposed variants, `SECO-NEAR`, `SECO-PREC` and `SECO-FOLL`, against regular as well as weighted versions (where the same context weighting scheme as described in Section 3.1 is applied to regular association measures) in order to analyze the effect of selective co-occurrence in particular, and not just the advantage obtained using weighted contexts.

[1] http://www.affective-sciences.org/system/files/webpage/ISEAR_0.zip. Removed instances with [no response].

[2] http://dumps.wikimedia.org/enwiki/20140811/enwiki-20140811-pages-articles.xml.bz2



(a) NPMI (b) Dice (c) Jaccard

Figure 2: Selective co-occurrences and regular co-occurrences on Alm dataset

As observed from Figure 2, tested for context window sizes 1 to 20 and trained on the Wikipedia corpus, our proposed approach, *preceding* selective co-occurrences (SECO−PREC), where only the nearest seed word that precedes a given cue word within a context window is considered, exhibits the best performance when applied to all the three association measures (NPMI, Dice and Jaccard), on the Alm dataset[3]. In fact, the best average F-score from SECO−NPMI−PREC is almost 10% better than that of Wt-NPMI, leading us to conclude that the gain in performance is due to selective co-occurrences and not just weighted contexts. As expected, the weighted versions, Wt-NPMI, Wt-Dice and Wt-Jaccard perform much better than their regular unweighted counterparts, NPMI, Dice and Jaccard, respectively. SECO−NEAR has slight advantage over weighted association measures while SECO−FOLL and regular association measures (i.e., NPMI, Dice, Jaccard) are nearly always the poorest performing. Since SECO−PREC−NPMI (Figure 2a) yielded better average F-score than SECO−PREC−Dice or SECO−PREC−Jaccard, we further evaluate its performance in the next experiment.

5.2 How effective is SECO−PREC−NPMI?

In this experiment, we evaluate the performance of unsupervised SECO−PREC−NPMI against five baselines (in **bold**) described next.

5.2.1 Baselines

WordNet Affect (**WNA**), NRC Emotion Lexicon (**EmoLex**) and DepecheMood (**DM**) (Section 2.1) contain words and their association with various emotions. For each emotion category, WNA contains a simple list of words, which we interpret as a binary association; if a word exists in an emotion category, we assign $+1$ for that emotion. EmoLex and DM, on the other hand, contain association scores between a word and all the emotion categories. For instance, EmoLex lists the association between the word "*awful*" and 8 emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, trust) as: $1, 0, 1, 1, 0, 1, 0, 0$. In DM, each word is associated with a different set of emotions by a real valued score, summing upto 1. For instance, the word "*awe*" and 8 emotions (afraid, amused, angry, annoyed, dont_care, happy, inspired, sad) is listed as: $0.08, 0.12, 0.04, 0.11, 0.07, 0.15, 0.38, 0.05$. We use these emotion lexicons as a baseline by applying a keyword matching algorithm, to obtain $Assoc.(w \rightarrow e)$. For example, $Assoc.(awe \rightarrow sad) = 0.05$ from DM. Note that, since DM does not contain two of the emotions found in our evaluation datasets, i.e., disgust and surprise, we report its results on a subset of the datasets. Instead of directly comparing our word-emotion association scores with those of emotion lexicons, we extrinsically evaluate them in the task of emotion classification as there are significant differences between the various emotion lexicons and none can be considered to be a perfect benchmark.

Semantic similarity computed from two state-of-the-art word embedding algorithms, **CBOW** and **SG** (Mikolov et al., 2013b; Mikolov et al., 2013a) (Section 2.2), provide another baseline as they can be used to obtain unsupervised word-emotion association scores, which is closer in spirit to our goal. We

[3]Consistent results were obtained on Aman and ISEAR datasets, not included here due to limited space.

used the algorithms' recommended default parameter settings: *dimension size of feature vectors* = 300; *negative sampling* = 5. We present a comparison against only unsupervised methods as the focus of this paper is on unsupervised emotion detection methods.

Since the context window size can have a significant impact on the performance of an algorithm, we run each method of semantic relatedness on 20 different window sizes (1 to 20) on both the corpora (Wikipedia and Amazon) and report the average result with standard deviation for each setting in Table 3. To keep the process as unsupervised as possible, in this study only one seed word per emotion category is used to derive the association scores. The seed words "*angry, disgust, happy, scared, sad, surprise*" represent the six emotion categories "*anger, disgust, happiness, fear, sadness, surprise*", respectively.

	Aman	**Alm**	**ISEAR**
SG_{wiki}	0.242 ± 0.04	0.209 ± 0.05	0.259 ± 0.08
$CBOW_{wiki}$	0.382 ± 0.02	0.426 ± 0.03	0.446 ± 0.03
SECO-PREC-NPMI$_{wiki}$	**0.410** ± 0.01**	**0.443** ± 0.01**	0.488 ± 0.02**
SG_{amazon}	**0.410** ± 0.02*	0.406 ± 0.02	0.438 ± 0.04
$CBOW_{amazon}$	0.393 ± 0.02	0.373 ± 0.02	0.484 ± 0.03
SECO-PREC-NPMI$_{amazon}$	0.403 ± 0.01	0.409 ± 0.01**	**0.498** ± 0.02**

Table 3: Average F-scores (of windows 1 to 20) for three evaluation datasets. SG, CBOW and SECO-PREC-NPMI were run on Wikipedia and Amazon corpora for windows 1 to 20. The best average result for each dataset is in **bold**. $^{**}p < .00001$, $^{*}p < .01$ (one-way ANOVA test for each dataset results using the same training corpus, i.e., wiki or amazon)

5.2.2 Results

Usually it is difficult to determine the best window size in advance, and therefore, for window sizes 1 to 20, we summarize the average F-scores over all 20 window sizes in Table 3. The best results obtained using any particular window are presented in Table 4 and the details of different emotion category results are further shown in Table 5. Finally, Figure 3 presents the window sensitivity graphs.

i) The average F-score results summarized in Table 3 indicate that on average, SECO-PREC-NPMI yields better overall results, with SG_{amazon}obtaining competitive results on one dataset (Aman), suggesting the effectiveness of selective co-occurrences in this task. Interestingly, contrary to popular intuition, the "objective" text from Wikipedia training corpus yields better F-scores on average than the "subjective" Amazon reviews corpus for two out of the three evaluation datasets.

	Aman	**Alm**	**ISEAR**
WNA	0.286	0.362	0.343
EmoLex	0.316	0.341	0.318
DM	0.324	0.340	0.290
SG_{wiki}	0.338 (2)	0.345 (1)	0.433 (1)
$CBOW_{wiki}$	0.410 (15)	0.456 (11)	0.481 (19)
SECO-PREC-NPMI$_{wiki}$	0.422 (10)	**0.464** (8)	0.497 (10)
SG_{amazon}	**0.435** (6)	0.440 (1)	0.490 (5)
$CBOW_{amazon}$	0.411 (6)	0.399 (18)	0.510 (19)
SECO-PREC-NPMI$_{amazon}$	0.412 (11)	0.422 (15)	**0.512** (20)

Table 4: Details of best results for three evaluation datasets. The best result for each dataset is in **bold**. The window size is shown in parentheses.

ii) The results of Table 4 indicate that, on all three evaluation datasets, with just one seed word per emotion category used to derive the word-emotion association scores, all the unsupervised measures of semantic relatedness (SECO-PREC-NPMI, SG and CBOW) outperform all the emotion lexicons (WNA, EmoLex and DM) that were created using considerable human input and training data, indicating that semantic similarity approaches provide an effective unsupervised way of extracting meaningful word-emotion association scores. Within the emotion lexicons, WNA provides the best performance on two out

of the three datasets despite being the smallest in size. Unsupervised association measures demonstrate two significant advantages over emotion lexicons: firstly, association measures are able to provide a wider coverage by exploiting the inherent associations between words that are present in text corpora; and secondly, while lexicons have fixed pre-determined emotion categories, association measures can be flexibly extended to any number and types of emotions. As for the recommended window settings for each approach, it seems that SG works well with window size $= 1$ on Wikipedia corpus and around 5 on Amazon; CBOW usually does well on windows larger than 15 and `SECO-PREC-NPMI` is recommended to be used with window 10 on Wikipedia and larger than 15 on Amazon.

AMAN																		
	ag			*dg*			*fr*			*hp*			*sd*			*sp*			Avg
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	
SG_{amazon}	0.66	0.25	0.36	0.65	0.40	**0.50**	0.33	0.71	**0.45**	0.79	0.34	0.47	0.34	0.64	**0.44**	0.24	0.66	**0.34**	0.435
$CBOW_{amazon}$	0.38	0.39	0.38	0.48	0.51	0.49	0.22	0.70	0.33	0.82	0.39	0.53	0.33	0.46	0.38	0.47	0.24	0.32	0.411
$SECO\text{-}PREC\text{-}NPMI_{wiki}$	0.41	0.44	**0.43**	0.46	0.23	0.30	0.34	0.44	0.38	0.66	0.64	**0.65**	0.48	0.39	0.43	0.27	0.49	**0.34**	0.422

ALM																
	ag-dg			*fr*			*hp*			*sd*			*sp*			Avg
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	
SG_{amazon}	0.58	0.44	**0.50**	0.40	0.55	0.46	0.77	0.31	0.44	0.40	0.71	0.51	0.23	0.33	0.27	0.440
$CBOW_{amazon}$	0.48	0.47	0.47	0.31	0.73	0.43	0.76	0.31	0.44	0.39	0.59	0.47	0.50	0.08	0.14	0.399
$SECO\text{-}PREC\text{-}NPMI_{wiki}$	0.52	0.27	0.36	0.42	0.52	**0.47**	0.64	0.70	**0.67**	0.57	0.53	**0.55**	0.24	0.33	**0.28**	0.464

ISEAR																
	ag			*dg*			*fr*			*hp*			*sd*			Avg
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	
SG_{amazon}	0.67	0.27	0.38	0.73	0.45	**0.56**	0.49	0.64	0.55	0.50	0.44	**0.47**	0.36	0.65	0.46	0.490
$CBOW_{amazon}$	0.42	0.62	0.50	0.61	0.47	0.53	0.53	0.67	**0.59**	0.60	0.32	0.42	0.51	0.47	**0.49**	0.510
$SECO\text{-}PREC\text{-}NPMI_{amazon}$	0.48	0.54	**0.51**	0.78	0.44	**0.56**	0.52	0.69	**0.59**	0.55	0.34	0.42	0.41	0.57	0.48	0.512

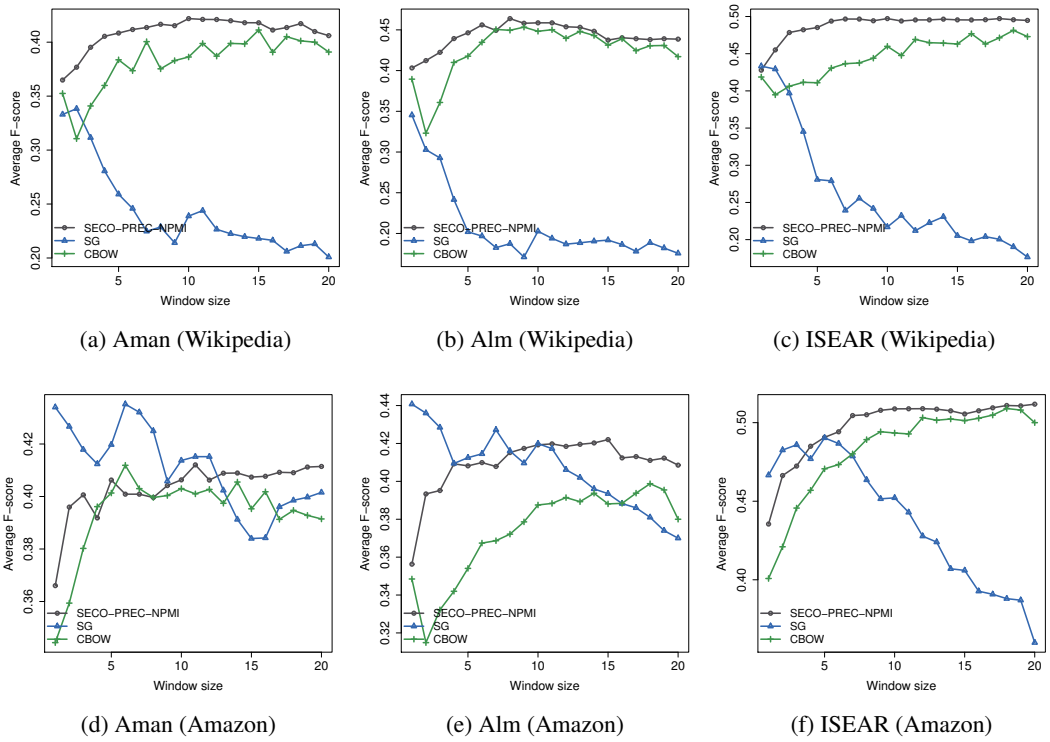
Table 5: Details of emotion category results for best window size/training corpus combination for three evaluation datasets. *ag* = anger, *dg* = disgust, *fr* = fear, *hp* = happy, *sd* = sad, *sp* = surprise. P = precision, R = recall, F = F-score.

iii) To analyze the individual emotion category results, we present the results of the best approach/training corpus combination in Table 5. In general, the *happiness* category obtains the highest results in two datasets while *fear* does best on the third. On the other hand, the most difficult category to be classified correctly seems to be *surprise*. One avenue of future work could include experimenting with various seed words to increase the accuracy of such emotions.

iv) Figure 3 summarizes the results of sensitivity of the three algorithms, SG, CBOW and `SECO-PREC-NPMI`, to different window parameter settings. On Wikipedia corpus, `SECO-PREC-NPMI` is consistently better than the others, whereas SG takes better advantage of the Amazon corpus. While `SECO-PREC-NPMI` and CBOW get better with bigger context windows, SG depicts the opposite trend, best on windows less than 5.

To summarize the results:

- Initialized using one seed word per emotion category, the measures of semantic relatedness yield better results than the emotion lexicons.
- When the window size is not known, in general, `SECO-PREC-NPMI` yields consistent promising results on all the evaluation datasets, while SG provides competitive results on one dataset.
- `SECO-PREC-NPMI` and CBOW yield better results with larger window sizes, whereas SG is best on windows less than 5.



(a) Aman (Wikipedia) (b) Alm (Wikipedia) (c) ISEAR (Wikipedia)

(d) Aman (Amazon) (e) Alm (Amazon) (f) ISEAR (Amazon)

Figure 3: Parameter sensitivity results for 20 window sizes

6 Conclusion

Emotion detection from text requires the degree of word-emotion association which is generally obtained from emotion lexicons or measures of semantic relatedness. While manual lexicons require considerable human time and effort, automatic techinques provide average performance. In this paper, we described a novel approach to automatically learning word-emotion association scores. Using just one seed word per emotion category, our proposed approach `SECO-PREC-NPMI` significantly outperformed three emotion lexicons and two state-of-the-art word embeddings models when trained using the Wikipedia text corpus.

As future work, we plan to further improve the accuracy of emotion classification by experimenting with a variety of seed words and also adapt `SECO` to other tasks that require association between two words.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable feedback. This research is funded in part by the Big Data Research, Analytics and Information Network (BRAIN) Alliance established by the Ontario Research Fund - Research Excellence Program (ORF-RE), Natual Sciences and Engineering Research Council of Canada (NSERC), and Social Sciences and Humanities Research Council of Canada (SSHRC).

References

Ameeta Agrawal and Aijun An. 2012. Unsupervised emotion detection from text using semantic and syntactic relations. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '12, pages 346–353, Washington, DC, USA. IEEE Computer Society.

Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: Machine learning for text-

based emotion prediction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 579–586, Stroudsburg, PA, USA. ACL.

Ebba Cecilia Ovesdotter Alm. 2008. *Affect in Text and Speech*. Ph.D. thesis, University of Illinois at Urbana-Champaign.

Jeanette Altarriba, Lisa M. Bauer, and Claudia Benvenuto. 1999. Concreteness, context availability, and imageability ratings and word associations for abstract, concrete, and emotion words. *Behavior Research Methods, Instruments, & Computers*, 31(4):578–602.

Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. In *Proceedings of the 10th International Conference on Text, Speech and Dialogue*, TSD'07, pages 196–205, Berlin, Heidelberg. Springer-Verlag.

Doug Beeferman, Adam Berger, and John Lafferty. 1997. A model of lexical attraction and repulsion. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, ACL '98, pages 373–380, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40.

Bernard Brosseau-Villeneuve, Jian-Yun Nie, and Noriko Kando. 2010. Towards an optimal weighting of context words based on distance. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 107–115. Association for Computational Linguistics.

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

Werner De Bondt, Rosa M Mayoral, and Eleuterio Vallelado. 2013. Behavioral decision-making in finance: An overview and assessment of selected research. *Spanish Journal of Finance and Accounting/Revista Española de Financiación y Contabilidad*, 42(157):99–118.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Shichuan Du, Yong Tao, and Aleix M Martinez. 2014. Compound facial expressions of emotion. *Proceedings of the National Academy of Sciences*, 111(15):E1454–E1462.

Facebook. 2016. Reactions now available globally. Accessed: 2016-03-05.

Jianfeng Gao, Ming Zhou, Jian-Yun Nie, Hongzhao He, and Weijun Chen. 2002. Resolving query translation ambiguity using a decaying co-occurrence model and syntactic dependence relations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 183–190, New York, NY, USA. ACM.

Zornitsa Kozareva, Borja Navarro, Sonia Vazquez, and Andres Montoyo. 2007. Ua-zbsa: A headline emotion classification through web information. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 334–337, Prague, Czech Republic, June. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc.

GermÃ¡n Kruszewski Marco Baroni, Georgiana Dinu. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, 1:238–247.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 43–52, New York, NY, USA. ACM.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Saif M. Mohammad and Peter D. Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, CAAGET '10, pages 26–34, Stroudsburg, PA, USA. Association for Computational Linguistics.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon. 29(3):436–465.

Alena Neviarouskaya, Helmut Prendinger, and Mitsuru Ishizuka. 2007. Textual affect sensing for sociable and expressive online communication. In *Proceedings of the 2Nd International Conference on Affective Computing and Intelligent Interaction*, ACII, pages 218–229, Berlin, Heidelberg. Springer-Verlag.

Jessica Perrie, Aminul Islam, Evangelos Milios, and Vlado Keselj. 2013. Using google n-grams to expand word-emotion association lexicon. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume 2*, CICLing'13, pages 137–148, Berlin, Heidelberg. Springer-Verlag.

Robert Plutchik. 2001. The nature of emotions human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice. *American Scientist*, 89(4):344–350.

Magnus Sahlgren. 2006. *The Word-space model*. Ph.D. thesis, University of Stockholm (Sweden).

LJ Shrum, Min Liu, Mark Nespoli, and Tina M Lowrey. 2013. Persuasion in the marketplace: How theories of persuasion apply to marketing and advertising. *The Sage Handbook of Persuasion: Developments in Theory and Practice*, pages 314–330.

Jacopo Staiano and Marco Guerini. 2014. Depechemood: a lexicon for emotion analysis from crowd-annotated news. *CoRR*, abs/1405.1605.

Carlo Strapparava and Rada Mihalcea. 2008. Learning to identify emotions in text. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, pages 1556–1560, New York, NY, USA. ACM.

Carlo Strapparava and Alessandro Valitutti. 2004. WordNet-Affect: An affective extension of WordNet. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 1083–1086. ELRA.

Changhua Yang, Kevin Hsin-Yih Lin, and Hsin-Hsi Chen. 2007. Building emotion lexicon from weblog corpora. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 133–136. Association for Computational Linguistics.

Weighted Neural Bag-of-n-grams Model: New Baselines for Text Classification

Bofang Li* , **Zhe Zhao***, **Tao Liu, Puwei Wang†, Xiaoyong Du**
School of Information, Renmin University of China
Key laboratory of Data Engineering and Knowledge Engineering, MOE
{libofang,helloworld,tliu,wangpuwei,duyong}@ruc.edu.cn

Abstract

NBSVM is one of the most popular methods for text classification and has been widely used as baselines for various text representation approaches. It uses Naive Bayes (NB) feature to weight sparse bag-of-n-grams representation. N-gram captures word order in short context and NB feature assigns more weights to those important words. However, NBSVM suffers from sparsity problem and is reported to be exceeded by newly proposed distributed (dense) text representations learned by neural networks. In this paper, we transfer the n-grams and NB weighting to neural models. We train n-gram embeddings and use NB weighting to guide the neural models to focus on important words. In fact, our methods can be viewed as distributed (dense) counterparts of sparse bag-of-n-grams in NBSVM. We discover that n-grams and NB weighting are also effective in distributed representations. As a result, our models achieve new strong baselines on 9 text classification datasets, e.g. on IMDB dataset, we reach performance of 93.5% accuracy, which exceeds previous state-of-the-art results obtained by deep neural models. All source codes are publicly available at https://github.com/zhezhaoa/neural_BOW_toolkit.

1 Introduction

Text representation is a core technology for many NLP tasks. Most text representation approaches fall into one of the two classes: sparse and distributed (dense) representations. One of the most popular sparse representations is bag-of-words (BOW), where each dimension represents the number of occurrences of a word in a text. Though simple, BOW enjoys the advantages of being efficient and surprisingly effective. Until now, BOW representation still serves as baselines on a range of NLP tasks.

In distributed representation, texts are represented by low-dimensional real vectors. Recently, there has been a surge of work proposing to learn distributed text representation through neural networks. There exists two lines of researches in neural models. The first is order/syntax-aware models, such as Convolutional Neural Networks (CNNs) (Kim, 2014; Kalchbrenner et al., 2014; Zhang et al., 2016), Recurrent Neural Networks (RNNs) (Dai and Le, 2015) and Recursive Neural Networks (RecNNs) (Socher et al., 2011; Socher et al., 2012). In these models, words are firstly embedded into low-dimensional real vectors (word embeddings) as the input, and then order/syntax-aware compositions upon words are learned by neural networks (Goldberg, 2015). Another line is neural bag-of-words models, where unordered compositions are learned upon word embeddings (Iyyer et al., 2015). The simplest neural bag-of-words model is word embeddings average. Compared to order/syntax-aware models, they are much more efficient in training (Iyyer et al., 2015) but lose the accuracies due to the ignorance of the order/syntax information.

To the best of our knowledge, most of the neural models map isolated words (uni-grams) to embeddings. Sometimes, consecutive words (n-grams), such as 'not like' and 'as good as' can convey semantics that are difficult to be obtained by simple compositions of individual words (Mikolov et al., 2013b). A natural extension is to embed n-grams into low-dimensional real vectors, e.g. the embedding of bi-gram

*Equal contribution. † Corresponding author.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1591–1600, Osaka, Japan, December 11-17 2016.

'not like' should be close to the embedding of word 'dislike'. The introduction of the n-gram embeddings takes the word order in short context into consideration, and can further enrich the semantics of text representations.

A distinct characteristic of neural models is their automatic feature extraction ability. Most neural models directly learn compositions upon word embeddings, and are reported to be powerful enough to learn high-quality distributed representations without human intervention. However, in sparse case, heuristic weighting techniques designed by humans are shown to be able to bring significant improvements over raw BOW representation (Wang and Manning, 2012; Martineau and Finin, 2009). For example, in sentiment classification, word 'amazing' is much more important than words like 'movie' and 'of', and should be given more weight in sparse BOW representation. Weighting technique has been successfully applied to sparse representation, but is still seldom used in neural models. Intuitively, neural models can also benefit a lot from knowing the importance of words in advance to guide the training processes.

In this paper, both n-grams and weighting techniques are introduced into the neural bag-of-words models. In fact, our models can be regarded as a neural or distributed counterparts of NBSVM (Wang and Manning, 2012). In NBSVM, these two techniques are shown to be very useful for sparse representations and strong baselines are achieved on a range of text classification tasks. In our work, we show how to transfer these two techniques to distributed representations, and discover that they are also very effective in distributed case.

We evaluate our models on 9 text classification tasks. Significant improvements are witnessed when n-gram and weighting techniques are introduced into neural bag-of-words models. As a result, new strong baselines are achieved on 5 document-level datasets and 4 sentence-level datasets. Most recently, state-of-the-art results on NLP tasks are dominated by deep neural models: CNNs exploit convolutional filters to extract n-grams features from texts; RNNs are reported to be able to capture long-distance patterns from natural languages. RecNNs even take syntactic information into consideration. In theory, these models are very powerful since complex compositionalies are learned upon word embeddings. However, experimental results in this paper give us further insights: though n-gram features have been studied in the NLP literature for decades and are usually viewed as baselines, they can still outperform features learned by the newly proposed deep neural models in many datasets if we can make use of them correctly. At least in text classification tasks, complex deep neural models do not show obvious superiority over our n-grams models on accuracies. What is more, deep neural models always require much more computational resources compared to bag-of-words (n-grams) models.

2 Related Work

Bag-of-words (n-grams)

Bag-of-words (n-grams) models treat a text as a set of words (n-grams), which ignore the fact that texts are essentially sequential data (Pang et al., 2002). Though the order information contained in word sequences is discarded, bag-of-words (n-grams) models are surprisingly effective, and also enjoy the advantages of being efficient and robust. They have been widely used in various kinds of NLP tasks such as information retrieval, question answering and text classification. Usually, sparse BOW features require weighting techniques to achieve better performance, where important words are given more weights while unimportant words are given less weights. For example, NBSVM (Wang and Manning, 2012) uses the ratio of the number of words in positive texts and negative texts to weight words, and achieves competitive results on a range of text classification tasks. However, traditional sparse BOW representations take each word or n-gram as a unit and ignore the internal semantics of them. As a result, they tend to generalize poorly compared with the newly proposed distributed representations.

Recently, distributed text representations are widely used in NLP tasks. The most fundamental work in the distributed representation literature is word embedding. In word embedding algorithms, syntactic and semantic information of words is encoded into low-dimensional real vectors and similar words tend to have close vectors (Bengio et al., 2003; Collobert et al., 2011; Mikolov et al., 2013a; Mikolov et al., 2013b). To obtain text embedding from word embedding, the simplest way is word embeddings

(vectors) average (VecAvg). This method belongs to neural bag-of-words models based on the fact that VecAvg also treats text as a set of words and discards order information totally. Other popular neural bag-of-words models include Deep Average Network (DAN) (Iyyer et al., 2015) and Paragraph Vector (PV) (Le and Mikolov, 2014). DAN constructs multiple neural layers upon the average of word vectors in the text. They show that more discriminative features can be extracted by deepening the layers of the neural networks. In PV, text embedding are trained to be useful to predict the words in the text. These neural bag-of-words models are reported to perform better than sparse BOW representations, and inherit the advantages of BOW models of being efficient and robust.

Complex Compositionality

Recently, many deep neural models are proposed on the basis of compositionality: the meanings of the expressions depend on their constituents. These models can learn complex compositionality upon words and achieve state-of-the-art results on a range of NLP tasks. Kim (2014) proposes to use convolutional neural networks (CNNs) to extract text features. N-grams information is extracted by convolutional layers and the most distinct features are selected by max pooling layers. Recurrent Neural Networks (RNNs) are sequential models and are suitable for texts data in nature. Hidden layers of RNNs can preserve the historical sequential information, and can be used as the representations of the texts (Dai and Le, 2015). However, both CNNs and RNNs are essentially 'flat' models, where structural information (e.g. syntactic parse tree) from texts are generally ignored. Socher et al. (2011) propose to use Recursive Neural Networks (RecNNs) to learn syntactic-aware compositionality upon words. They recursively combine neighbor nodes on parse trees in a bottom-up fashion until the root is reached.

Most recently, many researchers have focused on using the combinations of neural networks to achieve better text representations. Sutskever et al. (2014) propose to use multi-layer LSTM networks for text representation and significant improvement is achieved on machine translation when more layers of neural networks are added. Lai et al. (2015) use RNN-CNN for text representations, where RNN layer is used for extracting word sequences information and the most distinct features are selected by max-pooling layer. Tai et al. (2015) model the texts through tree-structured LSTM, which can be viewed as the combination of RecNN and RNN. To take the relationships among sentences in a document into consideration, some works have been done to learn document representation hierarchically (Kiros et al., 2015). For example, in (Li, 2014), RecNN is used for learning sentence embedding from word embedding, and RNN is used for learning document embedding from sentence embedding. Denil et al. (2014), Lin et al. (2015), Li et al. (2015b) and Tang et al. (2015) respectively propose to use CNN-CNN, RNN-RNN, LSTM-LSTM and CNN-LSTM to represent documents hierarchically.

These models are very powerful in theory since they exploit extensive information of texts such as word order, syntax and even relations among sentences. However, for text classification, a relatively simple task in the NLP community, deep neural models do not show their superiority over our n-grams models according to our experimental results. It still requires further explorations to demostrate if deep neural models can really exploit complex information beyond n-grams, or if complex information is really useful for text classification. What is more, most deep neural models require intensive training resources and usually need careful hyper-parameter tuning for specific datasets.

3 Models

In the following subsections, we present the framework of exploiting n-grams and weighting techniques for neural bag-of-words models. We use Paragraph Vector (PV) model (Le and Mikolov, 2014)[1] as a concrete case to illustrate the way of introducing these two techniques. Our method can be applied to other neural bag-of-words models straightforwardly.

Paragraph Vector (PV) is a popular method for text representation. In PV, text embedding is trained to be useful to predict words in the text. Formally, the objective is to maximize the conditional probabilities of words given their texts:

[1]Concretely, PV-DBOW (a variant of PV) is used in our paper.

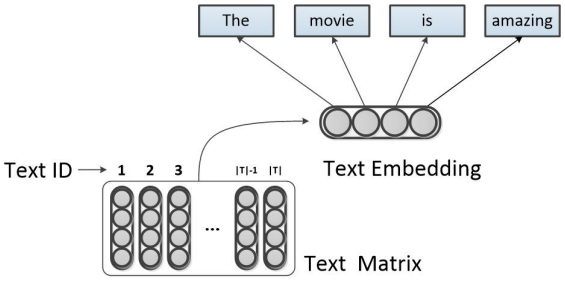


Figure 1: Illustration of the original Paragraph Vector model. The model only considers the uni-grams and they are equally treated in the model.

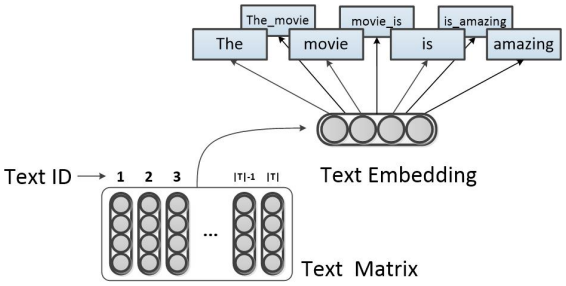


Figure 2: Illustration of n-gram PV model, where n-grams are predicted by the text embedding.

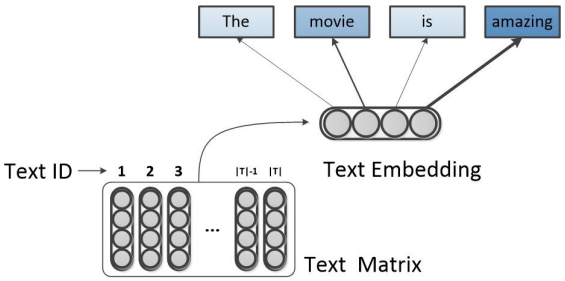


Figure 3: Illustration of weighted PV model, where important words are given more attention during the training process.

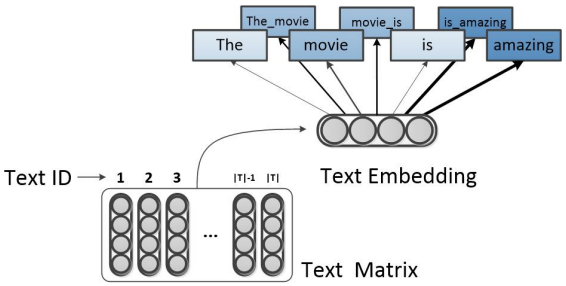


Figure 4: Combination of n-gram and weighting techniques. Text embeddings are trained to be useful to predict important n-grams during the training process.

$$\sum_{i=1}^{|T|}\sum_{j=1}^{|t_i|} logP(w_{i,j}|t_i) \tag{1}$$

where t_i=$\{w_{i,1},w_{i,2},.....,w_{i,|t_i|}\}$ denotes the i_{th} text and T=$\{t_1,t_2,.....,t_{|T|}\}$ denotes the whole dataset. In this paper, the conditional probability is defined by negative sampling (Mikolov et al., 2013b), which speeds up the training process significantly. Figure 1 clearly shows that, in PV, each word is predicted separately and the words in a text are equally treated.

3.1 N-gram Embedding

As we have mentioned above, n-grams can reflect semantics that can not be captured by looking at words individually. To introduce n-grams information into the neural models, a natural extension is to train n-gram embeddings (Li et al., 2015a). Each text is regarded as a set of n-grams (e.g. n = 1, 2, 3) and each n-gram is assigned a randomly initialized vector. The length of the text t is denoted by $|t|$. Obviously, text t consists of $|t|$ uni-grams, $|t|$-1 bi-grams and $|t|$-2 tri-grams.

During the training process, text embeddings are trained to be useful to predict n-grams in the text (as shown in figure 2). The following objective is optimized:

$$\sum_{i=1}^{|T|}\sum_{n=0}^{N-1}\sum_{j=1}^{|t_i|-n} logP(w_{i,j\sim j+n}|t_i) \tag{2}$$

where N denotes the number of consecutive words considered. When N equals to 3, uni-grams, bi-grams and tri-grams are predicted by the text. $w_{i,j\sim j+n}$ denotes *n+1* consecutive words $w_{i,j}, w_{i,j+1}, ..., w_{i,j+n}$. Word order in short context is captured by including n-grams information.

3.2 Naive Bayes Weighting

Intuitively, some n-grams are more important and should be paid more attention during the training process. In this paper, we only consider calculating weights of n-grams for binary classification. Without loss of generality, we use 'positive' and 'negative' to denote the name of two classes. Following works done by Wang and Manning (2012) and Martineau and Finin (2009), we use Naive Bayes feature to assign weight for each word. The weight of word *w* is calculated as follows:

$$(\frac{(\#POS(w)+\alpha)/|POS|}{(\#NEG(w)+\alpha)/|NEG|})^{\beta * \lfloor \#POS(w)/|POS| > \#NEG(w)/|NEG| \rfloor} \tag{3}$$

$$where \quad \lfloor \cdot \rfloor = \begin{cases} 1 & \cdot is True \\ -1 & \cdot is False \end{cases}$$

where ***#POS(w)*** denotes the number of positive texts that contain n-gram *w*, and ***#NEG(w)*** denotes the number of negative texts that contain n-gram *w*. α and β are two hyper-parameters for smoothing ratios (both of them are set to be 0.5 in this paper). $|POS|$ and $|NEG|$ denote the number of positive and negative texts respectively (including pseudo count α). To this end, words that have uneven distributions over classes are given more weights. Naive Bayes weighting is shown to be effective in sparse representation. To introduce weighting techniques into distributed case, weighted objective function is optimized to train text embeddings:

$$\sum_{i=1}^{|T|} \sum_{n=0}^{N-1} \sum_{j=1}^{|t_i|-n} Weight(w_{i,j\sim j+n}) logP(w_{i,j\sim j+n}|t_i) \tag{4}$$

where ***Weight(•)*** denotes the weight of n-gram •. As a result, text embeddings are trained to predict those important n-grams in larger probabilities rather than those words that have little discriminative information for classification. Figure 3 and 4 respectively demonstrate the overview of introducing weighting techniques into uni-gram and n-gram Paragraph Vector. The way we exploit n-grams and weighting information can be easily used in other neural bag-of-words models. For example, we can use the weighted average of n-gram embeddings as the input of the DAN neural networks (Iyyer et al., 2015).

In summary, this section introduces n-grams and NB weighting into neural bag-of-words models. These two techniques have shown their effectiveness on sparse representation in NBSVM. In the following Experiment Section, we demonstrate the effectiveness of n-grams and NB weighting for distributed representations.

4 Experiments

4.1 Datasets and Training Protocols

We evaluate our models on five document-level and four sentence-level text classification datasets. The details of the datasets are shown in table 1. The pre-processing of texts and the train/test, cross-validation splits strictly follow the implementation in NBSVM[2].

Our models are trained by stochastic gradient descent (SGD). The hyper-parameter setting follows the implementation in (Mesnil et al., 2014) [3] except that the training iterations are determined by validation set. When the training process is finished, the text representations are fed into a logistic regression classifier. Following the work done by Mesnil et al. (2014), we also combine the outputs of logistic classifiers (for sparse and dense representations) via linear interpolation. The weights are determined by validation set.

It is common to use pretrained vectors (Kim, 2014) and additional unlabeled texts (Mesnil et al., 2014; Li et al., 2015a)[4] to assist the training when size of the dataset is small. IMDB is a relatively large-scale

[2]https://github.com/sidaw/nbsvm

[3]https://github.com/mesnilgr/iclr15

[4]Mesnil et al. use the unlabeled data in their published implementation.

dataset and does not require pretrained vectors. For the rest of eight tasks, we use word2vec vectors to initialize the uni-grams. The additional unlabeled texts are added to RT-2k and RTs datasets, the details of which will be discussed in the following subsections.

Unless otherwise noted, we don't perform any data specific hyper-parameter tuning.

Type	Dataset	$\#(train+, train-, test+, test-)$	CV	$\lvert\bar{t}\rvert$	|V|
document-level	IMDB	(12500,12500,12500,12500)	N	231	392K
	RT-2k	(1000,1000)	10	787	51K
	AthR	(399,315,400,313)	N	345	22K
	Xgraph	(491,486,489,487)	N	261	32K
	BbCrypt	(497,496,497,495)	N	269	25K
sentence-level	RTs	(5331,5331)	10	21	21K
	CR	(2406,1366)	10	20	5713
	MPQA	(3316,7308)	10	3	6229
	Subj.	(5000,5000)	10	24	24K

Table 1: Datasets statistics. #(train+,train-,test+,test-): the number of positive and negative samples in train, test set respectively. For datasets that use cross-validation to evaluate models, column 3 only lists the number of positive and negative samples. CV: the number of cross-validation splits. N denotes train/test split. $|\bar{t}|$ denotes the average length of text samples. $|V|$ denotes the vocabulary size.

4.2 Comparison of Models on IMDB Dataset

IMDB dataset is one of the most popular benchmarks in text classification. A large amount of models are evaluated and compared on this dataset. In table 2, we compare our neural bag-of-words models with NBSVM. We can observe that dense representations consistently outperform their sparse counterparts. The n-grams and NB weighting are effective for both sparse and dense representations. Four percent improvement in accuracies is witnessed when n-grams and NB weighting are considered in dense representation. The best result is achieved by the ensemble of dense and sparse represntations (Mesnil et al., 2014).

In table 3, we compare our models with state-of-the-art methods which are dominated by deep neural models. To better compare different methods, we divide the existing models into three groups according to how they exploit information in the texts. Models in the first group treat a text as a bag of words or n-grams. Models in the second group treat texts as sequential data. In the third group, structural information such as syntactic parse tree and relationships among sentences is taken into consideration. In theory, accuracy should benefit from the sequential and structural information of texts. However, we surprisingly find that our models outperform other approaches, even though only n-gram information is exploited in our models.

Models	**Sparse**	**Dense**
Unigram	86.95	88.97
Unigram+NB	88.29	**90.10**
Bigram	89.16	91.27
Bigram+NB	91.22	**92.20**
Trigram	91.4	92.14
Trigram+NB	91.87	**92.95**
Ensemble	**93.51**	

Table 2: Comparison of sparse and dense representations. Dense representations consistently outperform sparse representations. N-grams and NB weighting are effective in both sparse and dense cases

Group	**Model**	**Accuracy**
bag-of-words	NBSVM-tri(Mesnil et al., 2014)	91.87
	Paragraph Vector(Mesnil et al., 2014)	88.73
	DAN(Iyyer et al., 2015)	89.40
	DV-tri(Li et al., 2015a)	92.14
	our model	92.95
	our ensemble	**93.51**
sequential	word2vec-LSTM(Dai and Le, 2015)	90.00
	SA-LSTM(Dai and Le, 2015)	92.76
	seq2-bown-CNN(Johnson and Zhang, 2015a)	92.33
	CNN+unsup3-tv(Johnson and Zhang, 2015b)	**93.49**
	Ensemble(Mesnil et al., 2014)	92.57
structural	DCNN(Denil et al., 2014)	89.40
	BENN(Li, 2014)	**91.00**
	RecRNN-RNN(Li, 2014)	87.00

Table 3: Comparison of state-of-the-art approaches, which are grouped according to how they exploit texts informtaion

4.3 Comparison of Models on Document-level Datasets

In this subsection, we continue evaluating our methods on document-level datasets. RT-2k dataset contains 2000 movie reviews. Texts in RT-2k and IMDB are both movie reviews and are classified according to whether they are positive or negative. Consequently, texts in IMDB are suitable alternative as additional unlabeled texts for RT-2k. AthR, XGraph and BbCrypt are classification pairs from 20-newsgroups. In these datasets, pretrained word2vec vectors are used to initialize uni-gram embeddings. N-gram embeddings are initialized randomly. To the best of our knowledge, state-of-the-art results on these four document-level datasets are still achieved by NBSVM. In table 4, we compare our models with their sparse counterparts NBSVM. We discover that dense representations benefit a lot from n-gram and weighting techniques. However, dense representations do not perform consistently better than sparse representations.

Not surprisingly, ensemble of the dense and sparse models achieves the best results.

Models	RT2k		AthR		XGraph		BcCrypt	
	Sparse	Dense	Sparse	Dense	Sparse	Dense	Sparse	Dense
Unigram	86.3	87.6	82.6	79.0	85.1	89.2	98.3	98.9
Unigram+NB	87.8	**88.7**	**87.9**	84.0	**91.2**	90.2	**99.7**	99.2
Bigram	87.4	89.2	83.7	81.1	86.2	90.5	97.7	99.1
Bigram+NB	89.5	**90.5**	**87.7**	86.1	90.7	**90.8**	99.5	**99.7**
Ensemble	**91.4**		**88.0**		**92.0**		**99.7**	

Table 4: Comparison of sparse and dense representations on document-level datasets.

4.4 Comparison of Models on Sentence-level Datasets

In this subsection, we demonstrate the effectiveness of our models on four sentence-level datasets: RTs, MPQA, CR and Subj. Pretrained word2vec vectors are used to initialize uni-gram embeddings. For RTs, data in IMDB dataset are used as additional unlabeled texts. From table 5, it can be observed that comparable results are achieved by dense and sparse representations. Similar with the conclusions obtained in above subsections: both n-grams and NB weighting improve the accuracies significantly. Ensemble of dense and sparse representations gives the best results.

In table 6, we make comparisons of state-of-the-art models, which are grouped according to their ways of exploiting text information. From the first row of table 6, we can observe that traditional PV performs poorly on sentence-level datasets compared to other state-of-the-art models. When n-grams, NB weighting and pretrained word embeddings are introduced, decent accuracies are achieved by PV. We can also observe that deep neural models show their superiority over bag-of-words models. Since n-grams information contained in sentence-level texts is limited, sequential and structural information is important for achieving better accuracies on these datasets. Nevertheless, most deep neural models in table 6 can not be extended to document-level texts. In contrast, our models can be used for texts of variable length.

Models	RTs		MPQA		CR		Subj.	
	Sparse	Dense	Sparse	Dense	Sparse	Dense	Sparse	Dense
Unigram	76.2	77.3	**86.1**	81.7	79.0	79.1	90.8	90.5
Unigram+NB	78.1	**78.7**	85.3	81.1	**80.5**	80.3	**92.4**	92.0
Bigram	77.7	78.5	**86.7**	82.0	80.8	80.1	91.7	91.2
Bigram+NB	79.4	**79.5**	86.3	82.1	**81.8**	81.1	**93.2**	92.8
Ensemble	**80.8**		**86.8**		**82.5**		**93.6**	

Table 5: Comparison of sparse and dense representations on sentence-level datasets.

Group	Model	RTs	MPQA	CR	Subj.
bag-of-words	Paragraph Vector(Kiros et al., 2015)	74.8	74.2	78.1	90.5
	NBSVM-bi(Wang and Manning, 2012)	79.4	86.3	81.8	93.2
	DAN(Iyyer et al., 2015)	80.3	-	-	-
	cBoW(Zhao et al., 2015)	77.2	86.4	79.9	91.3
	our model	79.5	82.1	81.1	92.8
	our ensemble	**80.8**	**86.8**	**82.5**	**93.6**
sequential	CNN(Kim, 2014)	81.5	89.6	**85.0**	93.4
	RNN(Zhao et al., 2015)	77.2	90.1	82.3	93.7
	BRNN(Zhao et al., 2015)	**82.3**	**90.3**	82.6	**94.2**
structural	combine-skip(Kiros et al., 2015)	76.5	87.1	80.1	93.6
	GrConv(Zhao et al., 2015)	76.3	84.5	81.3	89.5
	AdaSent(Zhao et al., 2015)	**83.1**	**93.3**	**86.3**	**95.5**

Table 6: Comparison of state-of-the-art approaches on sentence-level datasets.

4.5 Further Discussions

From tables in the above subsections, we can observe that n-gram features are still competitive if we can take full advantages of them. For document-level datasets, the ensemble of dense and sparse representation even outperforms the complex deep neural models which take complex compositions into consideration. For sentence-level datasets, competitive results are achieved by our models when pre-trained vectors or additional unlabelled data is added.

When taking efficiency and robustness into consideration, our n-gram models are better choices. Since our models are essentially bag-of-n-grams models, they only require a fraction of time compared to deep neural models. Our models are also robust for both sentence and document level datasets. In contrast, many deep neural models can not be extended to document-level datasets. Besides that, they usually require careful dataset-specific hyper-parameter tuning for better performance. While in our models, experimental setting is universal to all datasets except that the number of iterations are determined by validation set.

5 Conclusion

In this paper, we propose a framework of introducing n-grams and Naive Bayes weighting into neural bag-of-words models. These two techniques are effective for neural models and new strong baselines are achieved when they are used together. Though many state-of-the-art results in NLP tasks are achieved by deep neural models, we discover that for text classification, n-grams information is sufficient to achieve state-of-the-art accuracies. Moreover, our models inherit efficiency and robustness from bag-of-words representations: they only require a fraction of computational resources compared to deep neural models, and at the same time perform consistently well on a range of datasets without specific hyper-parameter tunings. Our source codes are organized as a text classification toolkit at `https://github.com/zhezhaoa/neural_BOW_toolkit`. We recommend to use the ensemble of dense and sparse representations implemented in our toolkit in real-world challenges.

Acknowledgements

This work is supported by National Natural Science Foundation of China with grant No. 61472428, the Fundamental Research Funds for the Central Universities, the Research Funds of Renmin University of China No. 14XNLQ06. This work is partially supported by ECNU-RUC-InfoSys Joint Data Science Lab and a gift from Tencent.

References

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

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, pages 3079–3087.

Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, and Nando de Freitas. 2014. Modelling, visualising and summarising documents with a single convolutional neural network. *CoRR*, abs/1406.3830.

Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *CoRR*, abs/1510.00726.

Mohit Iyyer, Varun Manjunatha, Jordan L. Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Volume 1: Long Papers*, pages 1681–1691.

Rie Johnson and Tong Zhang. 2015a. Effective use of word order for text categorization with convolutional neural networks. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112.

Rie Johnson and Tong Zhang. 2015b. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, pages 919–927.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, pages 3294–3302.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2267–2273.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning*, pages 1188–1196.

Bofang Li, Tao Liu, Xiaoyong Du, Deyuan Zhang, and Zhe Zhao. 2015a. Learning document embeddings by predicting n-grams for sentiment classification of long movie reviews. *CoRR*, abs/1512.08183.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015b. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Volume 1: Long Papers*, pages 1106–1115.

Jiwei Li. 2014. Feature weight tuning for recursive neural networks. *CoRR*, abs/1412.3714.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 899–907.

Justin Martineau and Tim Finin. 2009. Delta TFIDF: an improved feature space for sentiment analysis. In *Proceedings of the Third International Conference on Weblogs and Social Media*.

Grégoire Mesnil, Tomas Mikolov, Marc'Aurelio Ranzato, and Yoshua Bengio. 2014. Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. *CoRR*, abs/1412.5335.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems*, pages 3111–3119.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. *CoRR*, cs.CL/0205070.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 151–161.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL*, pages 1201–1211.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems*, pages 3104–3112.

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

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

Sida I. Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *The 50th Annual Meeting of the Association for Computational Linguistics, Volume 2: Short Papers*, pages 90–94.

Rui Zhang, Honglak Lee, and Dragomir R. Radev. 2016. Dependency sensitive convolutional neural networks for modeling sentences and documents. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1512–1521.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, pages 4069–4076.

A Deeper Look into Sarcastic Tweets Using Deep Convolutional Neural Networks

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Prateek Vij
Nanyang Technological University
50 Nanyang Ave, Singapore 639798
{sporia, cambria}@ntu.edu.sg
{devamanyu, prateek}@sentic.net

Abstract

Sarcasm detection is a key task for many natural language processing tasks. In sentiment analysis, for example, sarcasm can flip the polarity of an "apparently positive" sentence and, hence, negatively affect polarity detection performance. To date, most approaches to sarcasm detection have treated the task primarily as a text categorization problem. Sarcasm, however, can be expressed in very subtle ways and requires a deeper understanding of natural language that standard text categorization techniques cannot grasp. In this work, we develop models based on a pre-trained convolutional neural network for extracting sentiment, emotion and personality features for sarcasm detection. Such features, along with the network's baseline features, allow the proposed models to outperform the state of the art on benchmark datasets. We also address the often ignored generalizability issue of classifying data that have not been seen by the models at learning phase.

1 Introduction

Sarcasm is defined as "a sharp, bitter, or cutting expression or remark; a bitter gibe or taunt". As the fields of affective computing and sentiment analysis have gained increasing popularity (Cambria, 2016), it is a major concern to detect sarcastic, ironic, and metaphoric expressions. Sarcasm, especially, is key for sentiment analysis as it can completely flip the polarity of opinions. Understanding the ground truth, or the facts about a given event, allows for the detection of contradiction between the objective polarity of the event (usually negative) and its sarcastic characteristic by the author (usually positive), as in "*I love the pain of breakup*". Obtaining such knowledge is, however, very difficult.

In our experiments, we exposed the classifier to such knowledge extracted indirectly from Twitter. Namely, we used Twitter data crawled in a time period, which likely contain both the sarcastic and non-sarcastic accounts of an event or similar events. We believe that unambiguous non-sarcastic sentences provided the classifier with the ground-truth polarity of those events, which the classifier could then contrast with the opposite estimations in sarcastic sentences. Twitter is a more suitable resource for this purpose than blog posts, because the polarity of short tweets is easier to detect (as all the information necessary to detect polarity is likely to be contained in the same sentence) and because the Twitter API makes it easy to collect a large corpus of tweets containing both sarcastic and non-sarcastic examples of the same event.

Sometimes, however, just knowing the ground truth or simple facts on the topic is not enough, as the text may refer to other events in order to express sarcasm. For example, the sentence "*If Hillary wins, she will surely be pleased to recall Monica each time she enters the Oval Office :P :D*", which refers to the 2016 US presidential election campaign and to the events of early 1990's related to the US president Clinton, is sarcastic because Hillary, a candidate and Clinton's wife, would in fact not be pleased to recall her husband's alleged past affair with Monica Lewinsky. The system, however, would need a considerable amount of facts, commonsense knowledge, anaphora resolution, and logical reasoning to draw such a conclusion. In this paper, we will not deal with such complex cases.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1601–1612, Osaka, Japan, December 11-17 2016.

Existing works on sarcasm detection have mainly focused on unigrams and the use of emoticons (González-Ibánez et al., 2011; Carvalho et al., 2009; Barbieri et al., 2014), unsupervised pattern mining approach (Maynard and Greenwood, 2014), semi-supervised approach (Riloff et al., 2013) and n-grams based approach (Tsur et al., 2010; Davidov et al., 2010; Ptácek et al., 2014; Joshi et al., 2015) with sentiment features. Instead, we propose a framework that learns sarcasm features automatically from a sarcasm corpus using a convolutional neural network (CNN). We also investigate whether features extracted using the pre-trained sentiment, emotion and personality models can improve sarcasm detection performance. Our approach uses relatively lower dimensional feature vectors and outperforms the state of the art on different datasets. In summary, the main contributions of this paper are the following:

- To the best of our knowledge, this is the first work on using deep learning for sarcasm detection.
- Unlike other works, we exploit sentiment and emotion features for sarcasm detection. As user profiling is also an important factor for detecting sarcastic content, moreover, we use personality-based features for the first time in the literature.
- Pre-trained models are commonly used in computer vision. In the context of natural language processing (NLP), however, they are barely used. Hence, the use of pre-trained models for feature extraction is also a major contribution of this work.

The rest of the paper is organized as follows: Section 2 proposes a brief literature review on sarcasm detection; Section 4 presents the proposed approach; experimental results and thorough discussion on the experiments are given in Section 5; finally, Section 6 concludes the paper.

2 Related Works

NLP research is gradually evolving from lexical to compositional semantics (Cambria and White, 2014) through the adoption of novel meaning-preserving and context-aware paradigms such as convolutional networks (Poria et al., 2016a), recurrent belief networks (Chaturvedi et al., 2016), statistical learning theory (Oneto et al., 2016), convolutional multiple kernel learning (Poria et al., 2016b), and commonsense reasoning (Cambria and Hussain, 2015). But while other NLP tasks have been extensively investigated, sarcasm detection is a relatively new research topic which has gained increasing interest only recently, partly thanks to the rise of social media analytics and sentiment analysis.

An early work in this field was done by (Tsur et al., 2010) on a dataset of 6,600 manually annotated Amazon reviews using a kNN-classifier over punctuation-based and pattern-based features, i.e., ordered sequence of high frequency words. (González-Ibánez et al., 2011) used support vector machine (SVM) and logistic regression over a feature set of unigrams, dictionary-based lexical features and pragmatic features (e.g., emoticons) and compared the performance of the classifier with that of humans. (Reyes et al., 2013) described a set of textual features for recognizing irony at a linguistic level, especially in short texts created via Twitter, and constructed a new model that was assessed along two dimensions: representativeness and relevance. (Riloff et al., 2013) used the presence of a positive sentiment in close proximity of a negative situation phrase as a feature for sarcasm detection. (Liebrecht et al., 2013) used the Balanced Window algorithm for classifying Dutch tweets as sarcastic vs. non-sarcastic; n-grams (uni, bi and tri) and intensifiers were used as features for classification.

(Buschmeier et al., 2014) compared the performance of different classifiers on the Amazon review dataset using the imbalance between the sentiment expressed by the review and the user-given star rating. Features based on frequency (gap between rare and common words), written spoken gap (in terms of difference between usage), synonyms (based on the difference in frequency of synonyms) and ambiguity (number of words with many synonyms) were used by (Barbieri et al., 2014) for sarcasm detection in tweets. (Joshi et al., 2015) proposed the use of implicit incongruity and explicit incongruity based features along with lexical and pragmatic features, such as emoticons and punctuation marks. Their method is very much similar to the method proposed by (Riloff et al., 2013) except (Joshi et al., 2015) used explicit incongruity features. Their method outperforms the approach by (Riloff et al., 2013) on two datasets.

(Ptácek et al., 2014) compared the performance with different language-independent features and pre-processing techniques for classifying text as sarcastic and non-sarcastic. The comparison was done over three Twitter dataset in two different languages, two of these in English with a balanced and an imbalanced distribution and the third one in Czech. The feature set included n-grams, word-shape patterns, pointedness and punctuation-based features.

In this work, we use features extracted from a deep CNN for sarcasm detection. Some of the key differences between the proposed approach and existing methods include the use of a relatively smaller feature set, automatic feature extraction, the use of deep networks, and the adoption of pre-trained NLP models.

3 Sentiment Analysis and Sarcasm Detection

Sarcasm detection is an important subtask of sentiment analysis (Cambria et al., 2015). Since sarcastic sentences are subjective, they carry sentiment and emotion-bearing information. Most of the studies in the literature (Joshi et al., 2016; Bosco et al., 2013; Joshi et al., 2015; Farías et al., 2016) include sentiment features in sarcasm detection with the use of a state-of-the-art sentiment lexicon. Below, we explain how sentiment information is key to express sarcastic opinions and the approach we undertake to exploit such information for sarcasm detection.

In general, most sarcastic sentences contradict the fact. In the sentence "I love the pain present in the breakups" (Figure 1), for example, the word "love" contradicts "pain present in the breakups", because in general no-one loves to be in pain. In this case, the fact (i.e., "pain in the breakups") and the contradictory statement to that fact (i.e., "I love") express sentiment explicitly. Sentiment shifts from positive to negative but, according to sentic patterns (Poria et al., 2015b), the literal sentiment remains positive. Sentic patterns, in fact, aim to detect the polarity expressed by the speaker; thus, whenever the construction "I love" is encountered, the sentence is positive no matter what comes after it (e.g., "I love the movie that you hate"). In this case, however, the sentence carries sarcasm and, hence, reflects the negative sentiment of the speaker.

In another example (Figure 1), the fact, i.e., "I left the theater during the interval", has implicit negative sentiment. The statement "I love the movie" contradicts the fact "I left the theater during the interval"; thus, the sentence is sarcastic. Also in this case the sentiment shifts from positive to negative and hints at the sarcastic nature of the opinion.

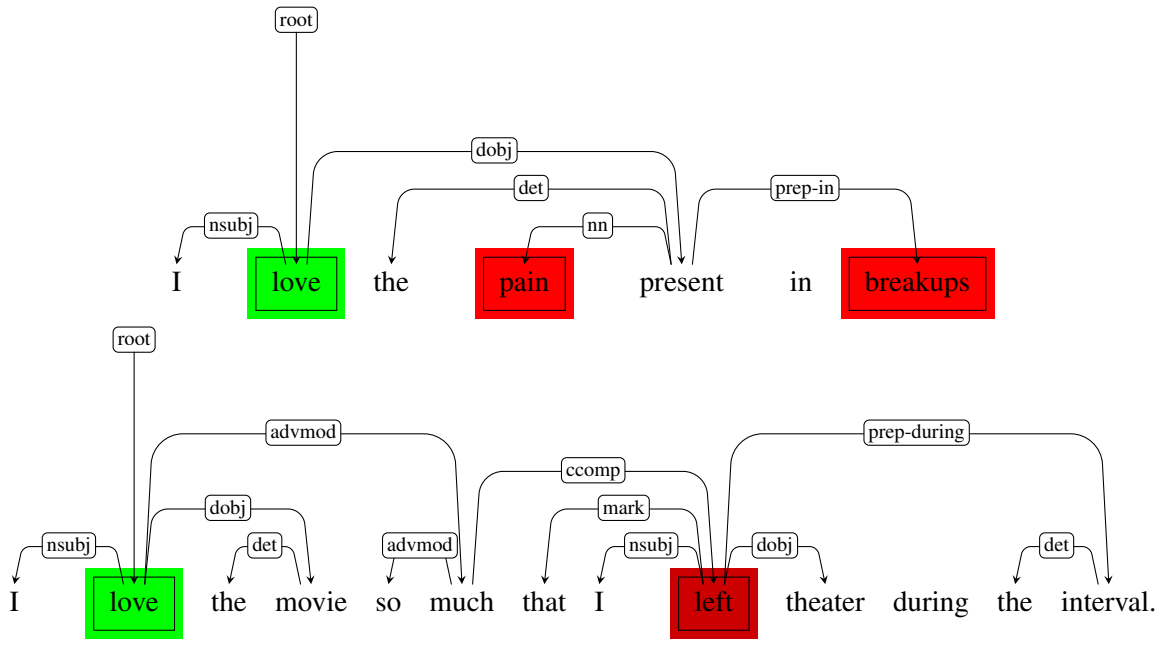


Figure 1: Sentiment shifting can be an indicator of sarcasm.

The above discussion has made clear that sentiment (and, in particular, sentiment shifts) can largely help to detect sarcasm. In order to include sentiment shifting into the proposed framework, we train a sentiment model for sentiment-specific feature extraction. Training with a CNN helps to combine the local features in the lower layers into global features in the higher layers. We do not make use of sentic patterns (Poria et al., 2015b) in this paper but we do plan to explore that research direction as a part of our future work. In the literature, it is found that sarcasm is user-specific too, i.e., some users have a particular tendency to post more sarcastic tweets than others. This acts as a primary intuition for us to extract personality-based features for sarcasm detection.

4 The Proposed Framework

As discussed in the literature (Riloff et al., 2013), sarcasm detection may depend on sentiment and other cognitive aspects. For this reason, we incorporate both sentiment and emotion clues in our framework. Along with these, we also argue that personality of the opinion holder is an important factor for sarcasm detection. In order to address all of these variables, we create different models for each of them, namely: sentiment, emotion and personality. The idea is to train each model on its corresponding benchmark dataset and, hence, use such pre-trained models together to extract sarcasm-related features from the sarcasm datasets.

Now, the viable research question here is - Do these models help to improve sarcasm detection performance?' Literature shows that they improve the performance but not significantly. Thus, do we need to consider those factors in spotting sarcastic sentences? Aren't n-grams enough for sarcasm detection? Throughout the rest of this paper, we address these questions in detail. The training of each model is done using a CNN. Below, we explain the framework in detail. Then, we discuss the pre-trained models. Figure 2 presents a visualization of the proposed framework.

4.1 General CNN Framework

CNN can automatically extract key features from the training data. It grasps contextual local features from a sentence and, after several convolution operations, it forms a global feature vector out of those local features. CNN does not need the hand-crafted features used in traditional supervised classifiers. Such hand-crafted features are difficult to compute and a good guess for encoding the features is always necessary in order to get satisfactory results. CNN, instead, uses a hierarchy of local features which are important to learn context. The hand-crafted features often ignore such a hierarchy of local features.

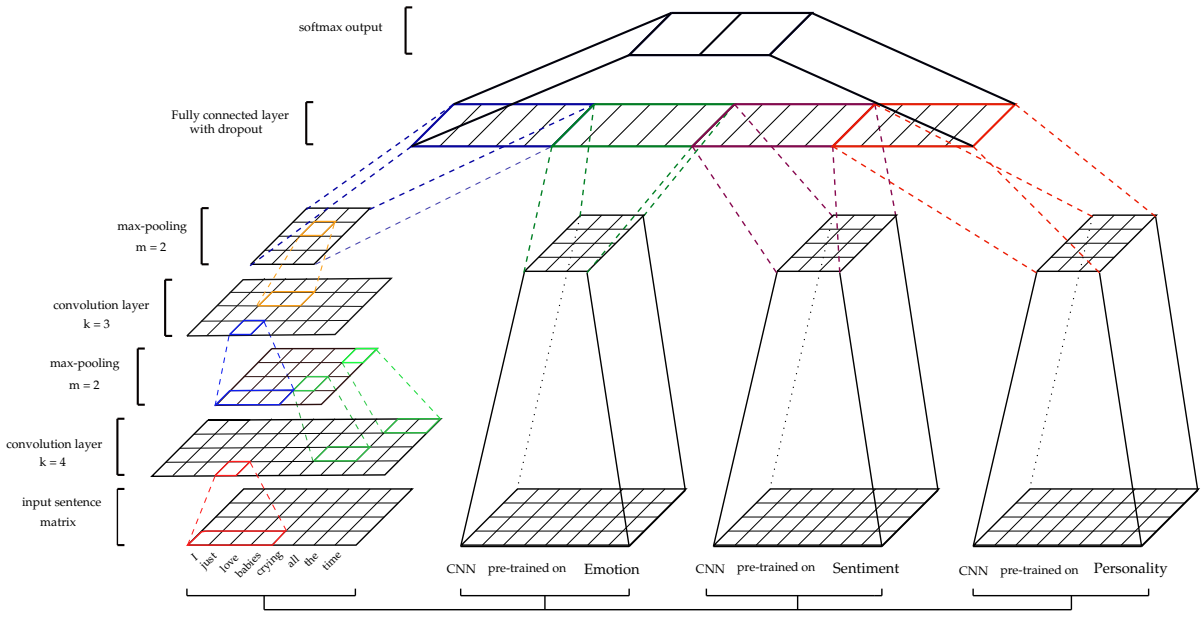


Figure 2: The proposed framework: deep CNNs are combined together to detect sarcastic tweets.

Features extracted by CNN can therefore be used instead of hand-crafted features, as they carry more useful information. The idea behind convolution is to take the dot product of a vector of k weights w_k also known as kernel vector with each k-gram in the sentence $s(t)$ to obtain another sequence of features $c(t) = (c_1(t), c_2(t), \ldots, c_{\mathrm{L}}(t))$.

$$c_j = {w_k}^T . \mathbf{x}_{i:i+k-1} \tag{1}$$

Thus, a max pooling operation is applied over the feature map and the maximum value $\hat{c}(t) = \max\{\mathbf{c}(t)\}$ is taken as the feature corresponding to this particular kernel vector. Similarly, varying kernel vectors and window sizes are used to obtain multiple features (Kalchbrenner et al., 2014). For each word x_i in the vocabulary, a d-dimensional vector representation is given in a look up table that is learned from the data (Mikolov et al., 2013). The vector representation of a sentence, hence, is a concatenation of vectors for individual words. Similarly, we can have look up tables for other features. One might want to provide features other than words if these features are suspected to be helpful. The convolution kernels are then applied to word vectors instead of individual words.

We use these features to train higher layers of the CNN, in order to represent bigger groups of words in sentences. We denote the feature learned at hidden neuron h in layer l as F_h^l. Multiple features may be learned in parallel in the same CNN layer. The features learned in each layer are used to train the next layer:

$$F^l = \sum_{h=1}^{n_h} w_k^h * F^{l-1} \tag{2}$$

where * indicates convolution and w_k is a weight kernel for hidden neuron h and n_h is the total number of hidden neurons. The CNN sentence model preserves the order of words by adopting convolution kernels of gradually increasing sizes that span an increasing number of words and ultimately the entire sentence. As mentioned above, each word in a sentence is represented using word embeddings.

Word Embeddings We employ the publicly available word2vec vectors, which were trained on 100 billion words from Google News. The vectors are of dimensionality 300, trained using the continuous bag-of-words architecture (Mikolov et al., 2013). Words not present in the set of pre-trained words are initialized randomly. However, while training the neural network, we use non-static representations. These include the word vectors, taken as input, into the list of parameters to be learned during training.

Two primary reasons motivated us to use non-static channels as opposed to static ones. Firstly, the common presence of informal language and words in tweets resulted in a relatively high random initialization of word vectors due to the unavailability of these words in the *word2vec* dictionary. Secondly, sarcastic sentences are known to include polarity shifts in sentimental and emotional degrees. For example, "*I love the pain present in breakups*" is a sarcastic sentence with a significant change in sentimental polarity. As *word2vec* was not trained to incorporate these nuances, we allow our models to update the embeddings during training in order to include them. Each sentence is wrapped to a window of n, where n is the maximum number of words amongst all sentences in the dataset. We use the output of the fully-connected layer of the network as our feature vector.

CNN-SVM We have done two kinds of experiments: firstly, we used CNN for the classification; secondly, we extracted features from the fully-connected layer of the CNN and fed them to an SVM for the final classification. The latter CNN-SVM scheme is quite useful for text classification as shown by Poria et al. (Poria et al., 2015a). We carry out n-fold cross-validation on the dataset using CNN. In every fold iteration, in order to obtain the training and test features, the output of the fully-connected layer is treated as features to be used for the final classification using SVM. Table 1 shows the training settings for each CNN model developed in this work. *ReLU* is used as the non-linear activation function of the network[1]. The network configurations of all models developed in this work are given in Table 1.

[1]We show the optimal training settings of the CNNs used in this work. Changing kernels' size or adding/removing layers does not improve results.

	Convolution Layer 1		1st Max Pooling	Convolution Layer 2		2nd Max-Pooling	FC Layer	Softmax Output
	Kernel Size	Feature Map		Kernel Size	Feature Map			
S	4,5	50	2	3	100	2	100	3
E	3,4,5	50	2	2	100	2	150	6
P	3,4,5	50	2	2	100	2	150	2
B	4,5	50	2	3	100	2	100	2

Table 1: Training settings for each deep model. Legenda: FC = Fully-Connected, S = Sentiment model, E = Emotion model, P = Personality model, B = Baseline model.

4.2 Sentiment Feature Extraction Model

As discussed above, sentiment clues play an important role for sarcastic sentence detection. In our work, we train a CNN (see Section 4.1 for details) on a sentiment benchmark dataset. This pre-trained model is then used to extract features from the sarcastic datasets. In particular, we use Semeval 2014 (Rosenthal et al., 2014) Twitter Sentiment Analysis Dataset for the training. This dataset contains 9,497 tweets out of which 5,895 are positive, 3,131 are negative and 471 are neutral. The fully-connected layer of the CNN used for sentiment feature extraction has 100 neurons, so 100 features are extracted from this pre-trained model. The final softmax determines whether a sentence is positive, negative or neutral. Thus, we have three neurons in the softmax layer.

4.3 Emotion Feature Extraction Model

We use the CNN structure as described in Section 4.1 for emotional feature extraction. As a dataset for extracting emotion-related features, we use the corpus developed by (Aman and Szpakowicz, 2007). This dataset consists of blog posts labeled by their corresponding emotion categories. As emotion taxonomy, the authors used six basic emotions, i.e., Anger, Disgust, Surprise, Sadness, Joy and Fear. In particular, the blog posts were split into sentences and each sentence was labeled. The dataset contains 5,205 sentences labeled by one of the emotion labels. After employing this model on the sarcasm dataset, we obtained a 150-dimensional feature vector from the fully-connected layer. As the aim of training is to classify each sentence into one of the six emotion classes, we used six neurons in the softmax layer.

4.4 Personality Feature Extraction Model

Detecting personality from text is a well-known challenging problem. In our work, we use five personality traits described by (Matthews and Gilliland, 1999), i.e., Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism, sometimes abbreviated as OCEAN (by their first letters). As a training dataset, we use the corpus developed by (Matthews and Gilliland, 1999), which contains 2,400 essays labeled by one of the five personality traits each.

The fully-connected layer has 150 neurons, which are treated as the features. We concatenate the feature vector of each personality dimension in order to create the final feature vector. Thus, the personality model ultimately extracts a 750-dimensional feature vector (150-dimensional feature vector for each of the five personality traits). This network is replicated five times, one for each personality trait. In particular, we create a CNN for each personality trait and the aim of each CNN is to classify a sentence into binary classes, i.e., whether it expresses a personality trait or not.

4.5 Baseline Method and Features

CNN can also be employed on the sarcasm datasets in order to identify sarcastic and non-sarcastic tweets. We term the features extracted from this network *baseline features*, the method as *baseline method* and the CNN architecture used in this baseline method as *baseline CNN*. Since the fully-connected layer has 100 neurons, we have 100 baseline features in our experiment. This method is termed baseline method as it directly aims to classify a sentence as sarcastic vs non-sarcastic. The baseline CNN extracts the inherent semantics from the sarcastic corpus by employing deep domain understanding. The process of using baseline features with other features extracted from the pre-trained model is described in Section 5.2.

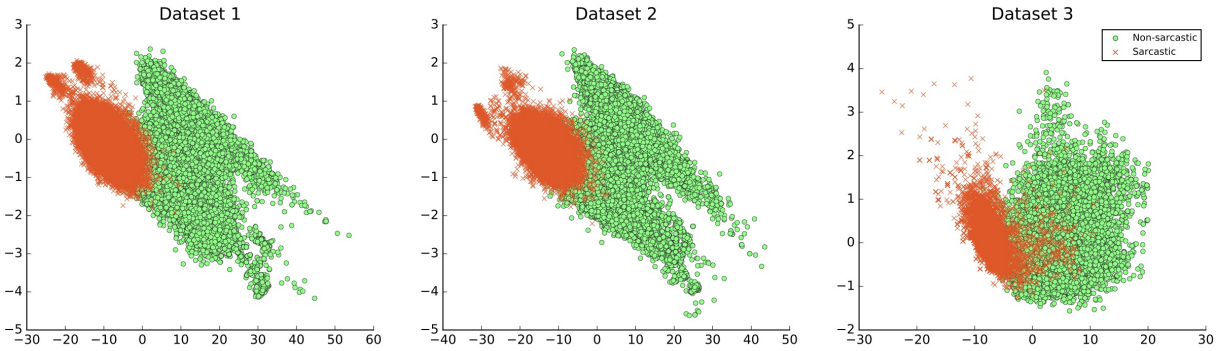


Figure 3: Visualization of the data.

5 Experimental Results and Discussion

In this section, we present the experimental results using different feature combinations and compare them with the state of the art. For each feature we show the results using only CNN and using CNN-SVM (i.e., when the features extracted by CNN are fed to the SVM). Macro-F1 measure is used as an evaluation scheme in the experiments.

5.1 Sarcasm Datasets Used in the Experiment

Dataset 1 (Balanced Dataset) This dataset was created by (Ptácek et al., 2014). The tweets were downloaded from Twitter using #sarcasm as a marker for sarcastic tweets. It is a monolingual English dataset which consists of a balanced distribution of 50,000 sarcastic tweets and 50,000 non-sarcastic tweets.

Dataset 2 (Imbalanced Dataset) Since sarcastic tweets are less frequently used (Ptácek et al., 2014), we also need to investigate the robustness of the selected features and the model trained on these features on an imbalanced dataset. To this end, we used another English dataset from (Ptácek et al., 2014). It consists of 25,000 sarcastic tweets and 75,000 non-sarcastic tweets.

Dataset 3 (Test Dataset) We have obtained this dataset from The Sarcasm Detector[2]. It contains 120,000 tweets, out of which 20,000 are sarcastic and 100,000 are non-sarcastic. We randomly sampled 10,000 sarcastic and 20,000 non-sarcastic tweets from the dataset. Visualization of both the original and subset data show similar characteristics.

Pre-processing A two-step methodology has been employed in filtering the datasets used in our experiments. Firstly, we identified and removed all the "user", "URL" and "hashtag" references present in the tweets using efficient regular expressions. Special emphasis was given to this step to avoid traces of hashtags, which might trigger the models to provide biased results. Secondly, we used *NLTK Twitter Tokenizer* to ensure proper tokenization of words along with special symbols and emoticons. Since our deep CNNs extract contextual information present in tweets, we include emoticons as part of the vocabulary. This enables the emoticons to hold a place in the word embedding space and aid in providing information about the emotions present in the sentence.

5.2 Merging the Features

Throughout this research, we have carried out several experiments with various feature combinations. For the sake of clarity, we explain below how the features extracted using difference models are merged.

- In the standard feature merging process, we first extract the features from all deep CNN based feature extraction models and then we concatenate them. Afterwards, SVM is employed on the resulted feature vector.

[2]http://thesarcasmdetector.com

- In another setting, we use the features extracted from the pre-trained models as the static channels of features in the CNN of the baseline method. These features are appended to the hidden layer of the *baseline CNN*, preceding the final output softmax layer.

For comparison, we have re-implemented the state-of-the-art methods. Since (Joshi et al., 2015) did not mention about the sentiment lexicon they use in the experiment, we used SenticNet (Cambria et al., 2016) in the re-implementation of their method.

5.3 Results on Dataset 1

As shown in Table 2, for every feature CNN-SVM outperforms the performance of the CNN. Following (Tsur et al., 2010), we have carried out a 5-fold cross-validation on this dataset. The baseline features (4.5) perform best among other features. Among all the pre-trained models, the sentiment model (F1-score: 87.00%) achieves better performance in comparison with the other two pre-trained models. Interestingly, when we merge the baseline features with the features extracted by the pre-trained deep NLP models, we only get 0.11% improvement over the F-score. It means that the baseline features alone are quite capable to detect sarcasm. On the other hand, when we combine sentiment, emotion and personality features, we obtain 90.70% F1-score. This indicates that the pre-trained features are indeed useful for sarcasm detection. We also compare our approach with the best research study conducted on this dataset (Table 3). Both the proposed baseline model and the *baseline + sentiment + emotion + personality* model outperform the state of the art (Joshi et al., 2015; Ptácek et al., 2014). One important difference with the state of the art is that (Ptácek et al., 2014) used relatively larger feature vector size (>500,000) than we used in our experiment (1,100). This not only prevents our model to overfit the data but also speeds up the computation. Thus, we obtain an improvement in the overall performance with automatic feature extraction using a relatively lower dimensional feature space.

In the literature, word n-grams, skipgrams and character n-grams are used as baseline features. According to Ptacek et al. (Ptácek et al., 2014), these baseline features along with the other features (sentiment features and part-of-speech based features) produced the best performance. However, Ptacek et al. did not analyze the performance of these features when they were not used with the baseline features. Pre-trained word embeddings play an important role in the performance of the classifier because, when we use randomly generated embeddings, performance falls down to 86.23% using all features.

5.4 Results on Dataset 2

5-fold cross-validation has been carried out on Dataset 2. Also for this dataset, we get the best accuracy when we use all features. Baseline features have performed significantly better (F1-score: 92.32%) than all other features. Supporting the observations we have made from the experiments on Dataset 1, we see CNN-SVM outperforming CNN on Dataset 2. However, when we use all the features, CNN alone (F1-score: 89.73%) does not outperform the state of the art (Ptácek et al., 2014) (F1-score: 92.37%). As shown in Table 3, CNN-SVM on the *baseline + sentiment + emotion + personality* feature set outperforms the state of the art (F1-score: 94.80%). Among the pre-trained models, the sentiment model performs best (F1-score: 87.00%).

B	S	E	P	Dataset 1		Dataset 2		Dataset 3	
				CNN	CNN-SVM	CNN	CNN-SVM	CNN	CNN-SVM
+				95.04%	97.60%	89.33%	92.32%	88.00%	92.20%
	+			-	87.00%	-	86.50%	-	73.50%
		+		-	76.30%	-	84.71%	-	72.10%
			+	-	75.00%	-	77.90%	-	74.41%
	+	+	+	-	**90.70%**	-	**90.90%**	-	**84.43%**
+	+			95.21%	97.67%	89.69%	94.60%	88.58%	93.12%
+		+		95.22%	97.65%	89.72%	94.50%	88.56%	92.63%
+			+	95.21%	97.64%	89.62%	93.60%	88.26%	92.50%
+	+	+	+	95.30%	**97.71%**	89.73%	**94.80%**	88.51%	**93.30%**

Table 2: Experimental Results. Legenda: B = Baseline, S = Sentiment, E = Emotion, P = Personality, 5-fold cross-validation is carried out for all the experiments.

Method	Dataset 1	Dataset 2	Dataset 3	D3 => D1
(Ptácek et al., 2014)	94.66%	92.37%	63.37%	53.02%
(Joshi et al., 2015)	65.05%	62.37%	60.80%	47.32%
Proposed Method (using all features)	97.71%	94.80%	93.30%	76.78%

Table 3: Performance comparison of the proposed method and the state-of-the-art approaches. Legenda: D3 => D1 is the model trained on Dataset 3 and tested on Dataset 1.

Table 2 shows the performance of different feature combinations. The gap between the F1-scores of only baseline features and all features is larger on the imbalanced dataset than the balanced dataset. This supports our claim that sentiment, emotion and personality features are very useful for sarcasm detection, thanks to the pre-trained models. The F1-score using sentiment features when combined with baseline features is 94.60%. On both of the datasets, emotion and sentiment features perform better than the personality features. Interestingly, using only sentiment, emotion and personality features, we achieve 90.90% F1-score.

5.5 Results on Dataset 3

Experimental results on Dataset 3 show the similar trends (Table 3) as compared to Dataset 1 and Dataset 2. The highest performance (F1-score 93.30%) is obtained when we combine baseline features with sentiment, emotion and personality features. In this case, also CNN-SVM consistently performs better than CNN for every feature combination. The sentiment model is found to be the best pre-trained model. F1-score of 84.43% is obtained when we merge sentiment, emotion and personality features.

Dataset 3 is more complex and non-linear in nature compared to the other two datasets. As shown in Table 3, the methods by (Joshi et al., 2015) and (Ptácek et al., 2014) perform poorly on this dataset. The TP rate achieved by (Joshi et al., 2015) is only 10.07% and that means their method suffers badly on complex data[3]. The approach of (Ptácek et al., 2014) has also failed to perform well on Dataset 3, achieving 62.37% with a better TP rate of 22.15% than (Joshi et al., 2015). On the other hand, our proposed model performs consistently well on this dataset achieving 93.30%.

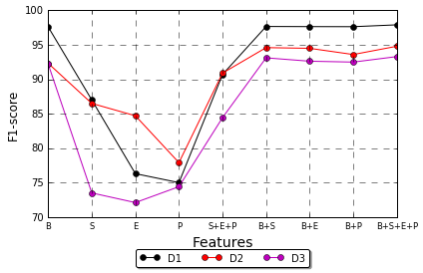
5.6 Testing Generalizability of the Models and Discussions

To test the generalization capability of the proposed approach, we perform training on Dataset 1 and test on Dataset 3. The F1-score drops down dramatically to 33.05%. In order to understand this finding, we visualize each dataset using PCA (Figure 3). It depicts that, although Dataset 1 is mostly linearly separable, Dataset 3 is not. A linear kernel that performs well on Dataset 1 fails to provide good performance on Dataset 3. If we use *RBF* kernel, it overfits the data and produces worse results than what we get using linear kernel. Similar trends are seen in the performance of other two state-of-the-art approaches (Joshi et al., 2015; Ptácek et al., 2014). Thus, we decide to perform training on Dataset 3 and test on the Dataset 1. As expected better performance is obtained[4] with F1-score 76.78%. However, the other two state-of-the-art approaches fail to perform well in this setting. While the method by (Joshi et al., 2015) obtains F1-score of 47.32%, the approach by (Ptácek et al., 2014) achieves 53.02% F1-score when trained on Dataset 3 and tested on Dataset 1. Below, we discuss about this generalizability issue of the models developed or referred in this paper.

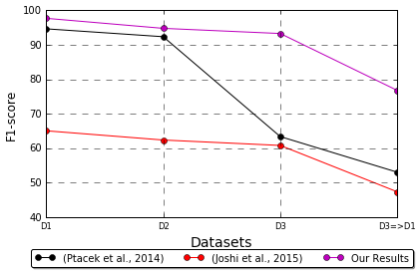
As discussed in the introduction, sarcasm is very much topic-dependent and highly contextual. For example, let us consider the tweet "I am so glad to see Tanzania played very well, I can now sleep well :P". Unless one knows that Tanzania actually did not play well in that game, it is not possible to spot the sarcastic nature of this sentence. Thus, an n-gram based sarcasm detector trained at time t_i may perform poorly to detect sarcasm in the tweets crawled at time t_j (given that there is a considerable gap between these time stamps) because of the diversity of the topics (new events occur, new topics are discussed) of the tweets. Sentiment and other contextual clues can help to spot the sarcastic nature in this kind of tweets. A highly positive statement which ends with a emoticon expressing joke can be sarcastic.

[3]We use *RBF* kernel, C=8 and gamma=0.01 to evaluate the method of Joshi et al. on Dataset 3 with 5-fold cross-validation.

[4]We report the result using all the features in this case.



(a) F1-score using different feature combinations.



(b) Comparison with the state of the art on benchmark datasets.

Figure 4: Plot of the performance of different feature combinations and methods.

State-of-the-art methods lack these contextual information which, in our case, we extract using pre-trained sentiment, emotion and personality models. Not only these pre-trained models, the baseline method (baseline CNN architecture) performs better than the state-of-the-art models in this generalizability test setting. In our generalizability test, when the pre-trained features are used with baseline features, we get 4.19% F1-score improvement over the baseline features. On the other hand, when they are not used with the baseline features, together they produce 64.25% F1-score.

Another important fact is that an n-grams model cannot perform well on unseen data unless it is trained on a very large corpus. If most of the n-grams extracted from the unseen data are not in the vocabulary of the already trained n-grams model, in fact, the model will produce a very sparse feature vector representation of the dataset. Instead, we use the word2vec embeddings as the source of the features, as word2vec allows for the computation of similarities between unseen data and training data.

5.7 Baseline Features vs Pre-trained Features

Our experimental results show that the baseline features outperform the pre-trained features for sarcasm detection. However, the combination of pre-trained features and baseline features beats both of themselves alone. It is counterintuitive, since experimental results prove that both of those features learn almost the same global and contextual features. In particular, baseline network dominates over pre-trained network as the former learns most of the features learned by the latter. Nonetheless, the combination of baseline and pre-trained classifiers improves the overall performance and generalizability, hence proving their effectiveness in sarcasm detection. Experimental results show that sentiment and emotion features are the most useful features, besides baseline features (Figure 4). Therefore, in order to reach a better understanding of the relation between personality features among themselves and with other pre-trained features, we carried out Spearman correlation testing. Results, displayed in Table 4, show that those features are highly correlated with each other.

	Sentiment	Happy	Fear	Openness	Conscientiousness
Sentiment	1.0	0.04*	0.03*	0.59*	0.83*
Happy		1.0	-0.48*	0.14*	0.12*
Fear			1.0	-0.10*	-0.09*
Openness				1.0	0.23*
Conscientiousness					1.0

Table 4: Spearman's correlations between different features. * Correlation is significant at the 0.05 level.

6 Conclusion

In this work, we developed pre-trained sentiment, emotion and personality models for identifying sarcastic text using CNN, which are found to be very effective for sarcasm detection. In the future, we plan to evaluate the performance of the proposed method on a large corpus and other domain-dependent corpora. Future work will also focus on analyzing past tweets and activities of users in order to better understand their personality and profile and, hence, further improve the disambiguation between sarcastic and non-sarcastic text.

References

Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. In *International Conference on Text, Speech and Dialogue*, pages 196–205. Springer.

Francesco Barbieri, Horacio Saggion, and Francesco Ronzano. 2014. Modelling sarcasm in Twitter, a novel approach. *ACL 2014*, pages 50–58.

Cristina Bosco, Viviana Patti, and Andrea Bolioli. 2013. Developing corpora for sentiment analysis and opinion mining: A survey and the Senti-TUT case study. *IEEE Intelligent Systems*, 28(2):55–63.

Konstantin Buschmeier, Philipp Cimiano, and Roman Klinger. 2014. An impact analysis of features in a classification approach to irony detection in product reviews. In *Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 42–49.

Erik Cambria and Amir Hussain. 2015. *Sentic Computing: A Common-Sense-Based Framework for Concept-Level Sentiment Analysis*. Springer, Cham, Switzerland.

Erik Cambria and Bebo White. 2014. Jumping NLP curves: A review of natural language processing research. *IEEE Computational Intelligence Magazine*, 9(2):48–57.

Erik Cambria, Soujanya Poria, Federica Bisio, Rajiv Bajpai, and Iti Chaturvedi. 2015. The CLSA model: A novel framework for concept-level sentiment analysis. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 3–22. Springer.

Erik Cambria, Soujanya Poria, Rajiv Bajpai, and Björn Schuller. 2016. SenticNet 4: A semantic resource for sentiment analysis based on conceptual primitives. In *COLING*.

Erik Cambria. 2016. Affective computing and sentiment analysis. *IEEE Intelligent Systems*, 31(2):102–107.

Paula Carvalho, Luís Sarmento, Mário J Silva, and Eugénio De Oliveira. 2009. Clues for detecting irony in user-generated contents: oh...!! it's so easy;-). In *International CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 53–56. ACM.

Iti Chaturvedi, Yew-Soon Ong, Ivor Tsang, Roy Welsch, and Erik Cambria. 2016. Learning word dependencies in text by means of a deep recurrent belief network. *Knowledge-Based Systems*, 108:144–154.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in Twitter and Amazon. In *Conference on computational natural language learning*, pages 107–116. Association for Computational Linguistics.

Delia Farías, Viviana Patti, and Paolo Rosso. 2016. Irony detection in Twitter: The role of affective content. *ACM Transactions on Internet Technology*, 16(3).

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in Twitter: A closer look. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 581–586. Association for Computational Linguistics.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 757–762.

Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. 2016. Automatic sarcasm detection: A survey. *arXiv preprint arXiv:1602.03426*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *CoRR*, abs/1404.2188.

Christine Liebrecht, Florian Kunneman, and Antal van den Bosch. 2013. The perfect solution for detecting sarcasm in tweets # not. In *Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*, pages 41–47.

Gerald Matthews and Kirby Gilliland. 1999. The personality theories of H.J. Eysenck and J.A. Gray: A comparative review. *Personality and Individual differences*, 26(4):583–626.

Diana Maynard and Mark A Greenwood. 2014. Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In *LREC*, pages 4238–4243.

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

Luca Oneto, Federica Bisio, Erik Cambria, and Davide Anguita. 2016. Statistical learning theory and ELM for big social data analysis. *IEEE Computational Intelligence Magazine*, 11(3):45–55.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2015a. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *EMNLP*, pages 2539–2544.

Soujanya Poria, Erik Cambria, Alexander Gelbukh, Federica Bisio, and Amir Hussain. 2015b. Sentiment data flow analysis by means of dynamic linguistic patterns. *IEEE Computational Intelligence Magazine*, 10(4):26–36.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2016a. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, 108:42–49.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Amir Hussain. 2016b. Convolutional MKL based multimodal emotion recognition and sentiment analysis. In *ICDM*, Barcelona.

Tomás Ptácek, Ivan Habernal, and Jun Hong. 2014. Sarcasm detection on Czech and English Twitter. In *COLING*, pages 213–223.

Antonio Reyes, Paolo Rosso, and Tony Veale. 2013. A multidimensional approach for detecting irony in Twitter. *Language resources and evaluation*, 47(1):239–268.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *EMNLP*, volume 13, pages 704–714.

Sara Rosenthal, Alan Ritter, Preslav Nakov, and Veselin Stoyanov. 2014. Semeval-2014 task 9: Sentiment analysis in Twitter. In *International workshop on semantic evaluation (SemEval 2014)*, pages 73–80.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. ICWSM-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *ICWSM*, pages 162–169.

Exploring Distributional Representations and Machine Translation for Aspect-based Cross-lingual Sentiment Classification

Jeremy Barnes Patrik Lambert* Toni Badia
Universitat Pompeu Fabra, Barcelona, Spain
*Webinterpret, Barcelona, Spain
{jeremy.barnes,toni.badia}@upf.edu, patrik.l@webinterpret.com

Abstract

Cross-lingual sentiment classification (CLSC) seeks to use resources from a source language in order to detect sentiment and classify text in a target language. Almost all research into CLSC has been carried out at sentence and document level, although this level of granularity is often less useful. This paper explores methods for performing aspect-based cross-lingual sentiment classification (aspect-based CLSC) for under-resourced languages. Given the limited nature of parallel data for under-resourced languages, we would like to make the most of this resource for our task. We compare zero-shot learning, bilingual word embeddings, stacked denoising autoencoder representations and machine translation techniques for aspect-based CLSC. We show that models based on distributed semantics can achieve comparable results to machine translation on aspect-based CLSC. Finally, we give an analysis of the errors found for each method.

1 Introduction

Sentiment analysis (SA) seeks to define the underlying sentiment of a text. The best results in SA require the use of a large number of resources; from tokenizers and parsers to large sentiment lexicons or hand-annotated corpora. The creation of these resources requires time, effort and a considerable monetary investment in order to ensure the quality and subsequent usefulness. Therefore, finding a way to perform sentiment analysis for under-resourced languages without having to repeat these efforts is an interesting endeavor. *Cross-lingual Sentiment Analysis* (CLSA) attempts to find methods to do just this. Most research on CLSA has been at document or sentence level. However, this does not capture the true granularity of opinionated text. Thus, in this paper we focus on CLSA at aspect-level[1]

Document- and sentence-level CLSA assume that an entire section of text expresses one sentiment towards one entity. This, however, is not always true. Aspect-based CLSA allows for multiple opinions towards multiple entities or aspects. Aspect-based CLSA can be decomposed into three subtasks: entity/aspect extraction, opinion holder extraction and sentiment classification. This last task, known as *Cross-lingual Sentiment Classification* (CLSC), has received little attention at aspect-level. Yet it would greatly benefit companies and government organizations that wish to gather information on the public opinions of their products or policies. In this paper we will only deal with improving or enabling aspect-based CLSC and leave the final goal of creating full aspect-based CLSA systems for under-resourced languages as future work.

Most research in CLSC has used Statistical Machine Translation (SMT) as a way of bridging the gap between languages, but there are drawbacks to this. First, an SMT system must be available for the language combination at hand. This requires a great deal of development and the quality of the sentiment analysis system used afterwards depends heavily on the quality of the SMT system. Secondly, study shows that even high quality SMT introduces noise into the data (Balahur and Turchi, 2014; Mohammad

[1]Here the term "aspect" refers to a feature of an entity. As an example taken from a hotel review, the sentence *"The rooms were great, but the service needs to improve"* contains two aspects (*rooms* and *service*) which pertain to the entity *hotel*. It is more useful to know that *rooms* is positive and *service* is negative than to know the overall sentiment towards *hotel*

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1613–1623, Osaka, Japan, December 11-17 2016.

et al., 2015). Finally, there are tasks in which systems which use distributed semantic representations to map between languages outperform SMT systems, e.g. cross-lingual document classification (Klementiev et al., 2012).

For this reason, a different representation of words and phrases, e.g. distributional vector representations, could prove to be a more effective approach and enable us to leverage information from resource-rich languages (English) to perform CLSA in a target language that lacks these resources (e.g. Spanish, Catalan, Basque).

This paper makes the following contributions:

- According to our knowledge, this is the most complete comparison of several types of distributed representations and machine translation for cross-lingual sentiment analysis.
- We give an analysis of the errors and possible ways to improve each system.
- We demonstrate that distributed representations can be competitive with machine translation for CLSC tasks.

2 Related Work

2.1 Monolingual Aspect-based Sentiment Analysis

Aspect-based sentiment analysis (ABSA) is a fine-grained approach to sentiment analysis. Many of the state-of-the-art ABSA systems in English require sophisticated NLP tools or hand-crafted sentiment lexicons. Hu and Liu (2004) propose WordNet-based methods for classifying aspect sentiment. Zhu et al. (2009) use sentiment lexicons. Moghaddam (2010) extracts an aspect and its nearest adjective and use a *k nearest neighbor* algorithm in order to estimate the rating of each aspect. Kiritchenko et al. (2014) use extracted features (part-of-speech tags, parsing features, sentiment lexicons, character-based information, n-grams) to train a *support vector machine* (SVM) for sentiment classification. These language-specific approaches do not lend themselves easily to CLSA because the target language often lacks the necessary resources.

2.2 Cross-lingual Sentiment Analysis

Aspect-based CLSA

In under-resourced languages, we lack resources and NLP tools which would allow us to create state-of-the-art systems similar to those mentioned. Therefore, the ability to leverage resources that already exist in English to perform sentiment analysis in other languages would be a great advantage. This would increase the performance of ABSA systems which are built using limited amounts of data in low-resourced languages and enable the creation of sentiment analysis systems in languages which have none at the moment.

Similarly, within CLSA, most researchers have worked at document- and sentence-level. In fact, there are only a handful of articles that deal with aspect-based CLSA. Zhou et al. (2012), Lin et al. (2014) and Klinger and Cimiano (2015) concentrate on extracting bilingual aspects but offer few ways to improve classification accuracy. Hass and Versley (2015) used machine translation and word alignment to map annotated syntactic nodes from English to German.

One of the difficulties at aspect-level is that the opinions attach to specific groupings of words, rather than a sentence or document. If we use SMT to create a new target language dataset, the opinionated units (e.g. opinion holder, opinion target, and opinion phrase) may be scattered or reordered. This would effectively reduce the usefulness of our new data because it would be difficult to project the opinion labels onto their corresponding word or phrase in the new dataset.

Lambert (2015) deals with this by using constrained SMT to translate the opinionated units within the context of the sentence. The classifiers trained on this SMT data achieve comparable results to their monolingual version. However, this is a state-of-the-art SMT system[2] which is not available in most language combinations. For these other languages, it would be useful to find alternative methods which do not require machine translation.

[2]The system achieves a *BLEU* score 45.3 in Spanish-English translation with true-case.

CLSA via Machine Translation

Advances in machine translation have made it possible to translate data from English to a target language or vice versa and use this data to train a classifying algorithm. There are reasons to believe that for well-resourced languages machine translation has reached a level that is useful for sentiment analysis (Banea et al., 2008; Duh et al., 2011; Balahur and Turchi, 2014; Mohammad et al., 2015). Much of the work has concentrated on the best combination of translation direction, classifiers and features (Banea et al., 2008; Banea et al., 2013; Balahur and Turchi, 2014). The advantage of this approach is that it is straight-forward to use a quality SMT system to create new resources by translating annotated corpora or sentiment lexicons (Mihalcea et al., 2007).

Nonetheless, there are disadvantages to using directly translated resources. Poor translation introduces a large amount of noise which hurts the performance of the classifier (Balahur and Turchi, 2014; Mohammad et al., 2015). It is clear that under-resourced languages are particularly susceptible to poor translation. Even with high quality machine translation, there is still a cross-lingual adaptation problem; the distribution of words and their polarity do not necessarily hold in cross-lingual contexts (Guo and Xiao, 2012; Mohammad et al., 2015). Therefore, we must find ways to minimize the undesirable effects of translation in cross-lingual sentiment analysis.

CLSA via Bilingual View

Another approach is to create a bilingual view of the data. The essence of this approach is to reduce the noise that translation introduces by presenting classifiers with complementary views. Wan (2009) creates a bilingual representation of the data through SMT and then uses co-training to take advantage of classifiers that commit complementary errors. This research seems promising, but there are some reasons to believe that the benefits of these techniques may have more to do with semi-supervised learning than cross-lingual transfer (Demirtas and Pechenizkiy, 2013).

Pan et al. (2011) use a bi-view non-negative matrix tri-factorization approach which allows for the incorporation of sentiment lexicon information. Lu et al. (2011) incorporate a joint bilingual model which makes use of unlabeled parallel or pseudo-parallel data in order to improve sentiment classification for both languages simultaneously.

CLSA via Latent View

Zhou et al. (2016) employ stacked denoising autoencoders to create a language independent representation of their data. This representation was then used as input for a linear SVM classifier.

For cross-lingual document classification, Prettenhofer and Stein (2011) use structural correspondence learning in order to find 'pivot' features. They use these features to create a representation of each document. These latent representations that encode the relationships betweens pivots and non-pivots are then used to train a linear classifier. Klementiev et al. (2012) create bilingual distributed representations as proposed by Bengio et al. (2003). They used these word vectors to classify cross-lingual documents.

The last two techniques are not entirely comparable to the first, since they perform cross-lingual document classification and not sentiment analysis. However, approaches which use latent representations are interesting because they have the potential to avoid some of the errors which are introduced by translation. To our knowledge, many techniques for creating latent bilingual representations (Chandar et al., 2014; Gouws et al., 2015; Vulić and Moens, 2016) have not been applied to CLSA. However, these techniques could provide a straightforward way to bridge the language gap.

3 Methodology

3.1 Datasets

The data used to train the sentiment analysis models are the English and Spanish OpeNER sentiment corpora (Agerri et al., 2013). We take a subset of these corpora which deal only with hotel reviews. Each review has annotations for opinion holders, opinion targets and opinion sentiment. We refer to this triplet (opinion holder, opinion target, opinion sentiment) as an opinion unit. The sentiment can be strong positive, positive, negative, or strong negative. A neutral category is not included. As such, when

training a classifier, rather than training on the complete sentence, we use the opinion unit. Table 1 shows the statistics for these corpora.

OpeNER Corpora	English	Spanish
Training Examples	2780	2991
Strong Pos	23.38%	29%
Pos	46.08%	50.34%
Neg	25.61%	17.41%
Strong Neg	4.93%	3.01%
Test examples	929	999
Strong Pos	23.36%	29.23%
Pos	46.07%	50.34%
Neg	25.62%	17.42%
Strong Neg	4.95%	3.00%

Table 1: Statistics of OpeNER Corpora

The corpora used to create the word embeddings are an English and Spanish Wikipedia corpus. These were taken from Wikipedia dumps in January 2016 and preprocessed to remove html markup and lowercase all words. We then performed sentence and word tokenization. We did not remove punctuation because this is often useful information for sentiment analysis. Table 2 gives the statistics for these corpora.

Wikipedia Corpora	English	Spanish
Number of sentences	118,900,197	26,777,415
Number of tokens	2,055,786,401	506,612,108

Table 2: Statistics of Wikipedia Corpora

The English-Spanish part of the Europarl v7 corpus[3] (Koehn, 2005) is used as parallel data. It contains around 2 million aligned sentences from the European Parliament. Table 3 shows the statistics for this corpus.

Europarl v7 Corpus	English	Spanish
Number of sentences	1,965,734	1,965,734
Number of tokens	49,093,806	51,575,784

Table 3: Statistics of Europarl v7 Corpus

3.2 Experiments

We performed a set of experiments in order to test different approaches for aspect-based CLSA. Each experiment requires a different amount of parallel data.

Representation of Training and Test Data for Sentiment Classification

For all experiments we use the same train and test split shown in Table 1. For each experiment, we trained a classifier on the English training data, performed the cross-lingual transfer on the Spanish test data and used this new data to test our classifier, as in Figure 1. One difficulty encountered when using vector representations is that the opinion units are variable length. This means that to train a classifier either we find a fixed-length representation for all opinion units or we use a classifier that accepts variable-length input. We decided to take an averaging approach, which has shown promise in other works (Iyyer et al., 2015). For each opinion unit we took the arithmetic mean of the words that compose the opinion unit,

[3]http://www.statmt.org/europarl

as shown in Figure 2, in order to create a fixed-length vector representation for each sentence. We then use these vectors to train a classifier. For the SMT transfer methods, we trained the classifier on unigram features. In all experiments, we used the *sequential minimal optimization* (SMO) classifier from the WEKA toolkit (Hall et al., 2009).

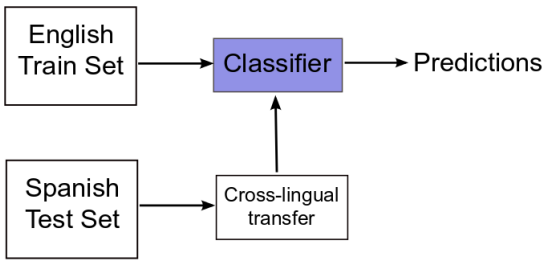


Figure 1: The process of cross-lingual sentiment classification. We assume that the opinion units have already been determined. The English train set is used to train a classifier. The Spanish test set is mapped accordingly and the classifier is tested on this cross-lingual test set.

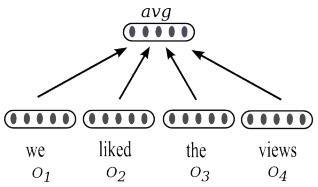


Figure 2: The representation of an opinion unit. For each word o_n in the opinion unit, we take its vector representation and average these vectors in order to create a fixed-length vector $avg = \sum_{i=1}^{n} \frac{o_i}{n}$, which we can use to train our classifier.

Zero-shot learning

Since we were interested in using the least amount of parallel data possible, we started with zero-shot learning. This is an approach which attempts to map between two monolingual vector representations using a "translation matrix" *W*. The desired effect is that the dot product of a Spanish word vector and *W* would be similar to the word vector of its English translation, as in Equation 1.

$$\mathbb{R}^{interesante} \circ W \approx \mathbb{R}^{interesting} \tag{1}$$

One could then perform a search for the most similar English vector and use this as a feature for classification. The only parallel data necessary is a bilingual dictionary. Given a pair of translated words and their associated vector representations $\{x_{eng}, x_{spa}\}$, we minimize the cost function in Equation 2 for our training vocabulary of length *n*:

$$\min_{W} \sum_{i=1}^{n} \parallel x_{eng} - x_{spa} \circ W \parallel^2 \tag{2}$$

Following Mikolov et al. (2013b) we created two sets of monolingual word embeddings using the Europarl v7 corpus (Koehn, 2005). We used the Skip-gram model (Mikolov et al., 2013a) and created 300 dimensional vectors using a window of 5 words, and 10 negative samples. We compiled a bilingual dictionary by taking the 8000 most common words in the English Wikipedia and translating them using Bing Translator[4]. Although Bing gives several options, we take only the first translation for use in our

[4]http://www.microsofttranslator.com/

bilingual dictionary. We then removed errors and ambiguous words and arrived at a final number of 4518 word pairs to train the matrix. Finally, we used stochastic gradient descent to optimize the translation matrix *W*. After creating the transition matrix *W*, we tested the effectiveness of this matrix translation to enable CLSC.

For each opinion unit in the corpora, we created a fixed-length vector representation, as shown in Figure 2. We now had a dataset with training instances such as $\{x_i : y_i\}$, where x_i was a 300 dimension vector and y_i was its corresponding label (Strong Positive, Positive, Negative, Strong Negative).

As a baseline, we trained and tested an SVM on the Spanish data from the OpeNER corpus as the Spanish test set has the same opinion units as the cross-lingual test set. We did the same with the English data, although this is not truly comparable[5]. Results are shown in Table 4.

We then created the cross-lingual test set by applying our translation matrix *W* to the Spanish test set. In order to find the most similar vector from the English word embeddings, we used a *k nearest neighbor* algorithm with cosine as the distance metric. Finally, we used the mean of the word embeddings as mentioned above to create the final fixed-length representation. We tested on the cross-lingual test set. The results are shown in Table 4.

From the results it is clear that our zero-shot approach did not yield any effective results. This may be a result of several factors. Mikolov et al. (2013b) were able to leverage a simple mapping strategy between word embeddings created from large monolingual datasets in order to fill the gaps in translation dictionaries. Given the poor results of this experiment, it seems unlikely that this same strategy can effectively capture the complex relationship between target and source language word vectors in a way that is useful for sentiment analysis. This is likely due to the different purposes of the mapping strategy in each approach. In Mikolov et al. (2013b), the success of this technique depended largely on using a small subset of the vocabulary and pairing it with other approaches. In our approach, however, all of the weight of correctly classifying a phrase fell on the accuracy of the mapping scheme. Therefore, it seems that any error in the mapping resulted in the propagation of error during classification.

Another problem that arose is that there were some words whose vector representation always appeared as the nearest neighbor of many other words, although they were not semantically similar with any of them. This problem is known as *hubness* and is an intrinsic problem with high-dimensional vector space. Our work seems to confirm the research of Lazaridou et al. (2015) and Georgiana Dinu et al. (2015), who showed that hubness is compounded when trying to create a linear mapping between two sets of word embeddings.

Bilingual Word Embeddings

The next set of experiments required the use of parallel sentences to create bilingual word embeddings (**BWEs**). Following the work of Luong et al. (2015), we created bilingual word embeddings using the Bilingual Skip-gram algorithm, which uses the Skip-gram model (Mikolov et al., 2013a) with an added bilingual objective. This algorithm creates vector representations in which words that appear in parallel sentences have similar representations. We used the Bilingual Skip-gram algorithm to train English and Spanish word vectors on the Europarl corpus (Koehn, 2005) and the corpus of parallel sentences in the hotel domain used in the work of Lambert (2015). We created the alignment using 3 iterations of the Berkley Aligner[6]. We then created 300 dimensional vectors with a window of 5 words, 10 negative samples and ran the algorithm for 3 epochs. This process gave us two sets of word embeddings in which words that often appear in parallel sentences have similar vector representations.

To train our classifier, we used our learned English embeddings and take the average of the vectors in each opinion unit in the English train set. We performed the same procedure with the learned Spanish embeddings and the Spanish test set. The results are shown in Table 4.

The results given by the bilingual word embeddings are not optimal, but are promising enough to warrant more research. There are problems with bilingual word embeddings which would need to be addressed in order to improve their usefulness for CLSC. First, there is the problem of ambiguity that affects all word embeddings. One way to correct this problem would be to disambiguate the word

[5]The monolingual English test set does not have the same examples as the Spanish one.

[6]https://code.google.com/archive/p/berkleyaligner/

senses prior to creating the word embeddings. Cheng et al. (2014) show that this technique improves the performance of distributional models for learning compositional models of meaning and it may improve the performance for sentiment analysis as well.

Secondly, due to the fact that they have similar distributions, antonyms are often given similar vector representations. This is not a problem for POS-taggers or parsers, but it is detrimental to sentiment analysis systems based on word embeddings because these words have opposing polarities and should therefore have different vector representations. To remedy this, one could add a classification task in the problem formulation that would better separate these antonyms into differing vector spaces (Tang et al., 2014). Another option is to decompose the word vectors into interpretable subspaces, train them to differentiate for a certain property, and use only these spaces as features (Rothe and Schütze, 2016).

Stacked Denoising Autoencoders

Following the work of Zhou et al. (2016) we trained a stacked bilingual denoising autoencoder (**SDBA**) on parallel sentences from the Europarl corpus. This approach aims to encode the parallel sentences into a common latent space. Given a vocabulary of length n, the autoencoder maps the sentences, which are represented as n-dimensional one-hot vectors, to a lower dimensional representation. These representations are then used to reconstruct the original sentences. In order to keep the autoencoder from simply learning the identity function, the lower dimensional representation of one of the sentences is corrupted, which causes the autoencoder to look for discriminative features to help reconstruct the original sentences. In this way, the autoencoder learns to find a lower dimensional representation that encodes as much information as possible needed to reconstruct the bilingual sentences.

We created source and target language autoencoders with 1000 hidden units, which were then mapped to 500 hidden units. The corruption level was set to 0.5. The 500 source and 500 target hidden units were then concatenated and normalized to unit length and fed to the bilingual autoencoder, again with the corruption level set to 0.5. After the autoencoder had been trained, we could use the learned weights to force any of our data into a latent bilingual space.

We then created our training data by mapping the opinion units from the English train set to this lower dimensional latent representation, which was a 500 dimensional vector. We trained a classifier on the mapped English training set. We tested on the similarly mapped Spanish test set. The results are shown in Table 4.

The stacked denoising autoencoder approach gave reasonably good results, despite the fact that it was designed for sentence-level CLSA. There are still ways which we could adapt this approach to aspect-based CLSA. By using word alignment, we could split sentences into parallel or pseudo-parallel n-grams and train the autoencoder with this data. This may improve its performance at aspect-level.

Statistical Machine Translation

For the final experiment we used statistical machine translation as a means of bridging the gap between languages. We compared Google Translate[7], a highly developed SMT system, as well as Constrained SMT (see section 2.2). First, we trained our monolingual English classifier and used Google Translate to create our cross-lingual test set. In this case, we translated only the opinionated phrases. This technique has the disadvantage that translation is done without context. We compared this with the Constrained SMT approach in Lambert (2015). This technique allows us to translate the opinion units in context, but without reordering or scrambling them. The language model used in this approach was trained with hotel domain data. All of this improved the quality of translation and resulted in more accurate CLSC results.

Finally, we trained our classifier on unigram features from the monolingual English training set. We created test sets by translating the Spanish test data with each SMT system. The results are shown in Table 4.

The constrained SMT approach is the most accurate approach and shows that, given a more refined treatment of less parallel data, one can achieve CLSA systems which are comparable to monolingual ones. It is interesting that Google Translate has a better BLEU score than constrained SMT[8], but the

[7] http://translate.google.com/

[8] Google Translate scores a 48.6 BLEU in English-Spanish true-case, versus 45.3 for constrained SMT.

performance on the classification task was lower. It also showed poorer results than the bilingual stacked denoising autoencoder.

Equal amounts of parallel data

Each of the previous experiments rely on different amounts of parallel data for optimal performance. Since we are interested in their performance on under-resourced languages, we ran all experiments again with the minimal amount of parallel data (measured at 15.9M English words). The results shown in Table 4 show that, despite a general decrease in precision, recall and F1, the performance of bilingual word embeddings remains stable with less data. The stacked denoising autoencoder, however, performs poorly with this amount of data.

4 Results

	English	Spanish	Zero-shot	Const. SMT	BWEs	SBDA	Google SMT
Parallel Data	-	-	4518	15.9M	15.9M	15.9M	-
Precision	-	-	.453	.779	.49	.254	-
Recall	-	-	.310	.758	.468	.4	-
F1 Score	-	-	.351	.755	.473	.338	-
Accuracy	-	-	48%	75.76%	62%	55%	-
Parallel Data	0	0	4518	15.9M	49M	49M	?
Precision	.824	.803	.517	.779	.632	.682	.670
Recall	.822	.809	.503	.758	.598	.590	.568
F1 Score	.820	.800	.434	.755	.567	.633	.615
Accuracy	82.22%	80.86%	50.25%	75.76%	59.76%	74.5%	72.81%

Table 4: Results of Crosslingual Experiments: Precision, recall and F1 are the weighted averages of all classes. The amount of parallel data is measure in the number of English tokens used in training the transfer method.

5 Discussion

Role of Parallel Data

It is interesting to see that there is not a direct correlation between the amount of parallel data used and the results. Constrained SMT uses less data than bilingual word embeddings or stacked denoising autoencoders and still outperforms both. However, this approach uses higher quality, in-domain data as well as tuning parameters which adapt it to this domain. The trend within representation and distributional approaches has been to use larger and larger datasets , but these results seem to suggest that using smaller, task-specific in-domain datasets which are automatically discovered from larger datasets may be key in improving performance in CLSC.

Representation

Besides using the average of the vectors in the opinion unit as a representation, we also experimented with summation and using Long Short-term Memory networks (LSTMs) as a way to deal with the different lengths of the opinion units for our vector-based methods. Although it lacks a strong theoretical motivation, summation is often used in distributional semantics as a baseline for combining vectors (Giorgiana Dinu and Baroni, 2014). Summation led to results that were slightly worse than averaging. This is likely due to the fact that longer opinion units result in vectors which are a magnitude larger than shorter opinion units. We discuss LSTMs below.

Classifiers

Apart from the SVM classifiers used in all experiments, we conducted further experiments using deep feed-forward networks during the zero-shot and bilingual word embeddings experiments. We used the

DAN model (Iyyer et al., 2015), with three hidden layers of 300 dimensions. This model performed better on the zero-shot learning experiments, but similar to SVMs on all other experiments.

We also experimented with an LSTM with a 400 dimensional hidden layer. The final state of the LSTM is used to output a softmax probability over the four classes. The results, however, were not competitive with the SVM for zero-shot learning or bilingual word embeddings. We believe the loss of information during transfer did not allow the LSTM to detect the same features during testing that it found while training. We also suspect that the dataset was not large enough to train an LSTM easily. This may limit the usefulness of LSTMs in cross-lingual settings where the dataset used to train the model is small. Given larger training sets, this effect may decrease.

6 Conclusion and Future Work

We have presented a comparison of aspect-based CLSA approaches using different amounts of parallel data. The results show that a simple zero-shot learning approach is currently ineffective for CLSA. We show that distributional vector representations are more promising and produce results that are comparable to simple SMT baselines, but still require more research.

In future work, we plan to investigate the role of prior disambiguation and ways to add sentiment-specific information to bilingual word embeddings. We believe this will make bilingual word embeddings more useful for aspect-based CLSA. Another approach that could improve the performance is to use ensemble classification, where we combine SMT and word vector information. Finally, we will extend these techniques to Catalan and Basque.

References

Agerri, Rodrigo, Montse Cuadros, Sean Gaines, and German Rigau. 2013. OpeNER: Open polarity enhanced named entity recognition. *Procesamiento del Lenguaje Natural*, 51(September):215-218.

Balahur, Alexandra and Marco Turchi. 2014. Comparative experiments using supervised learning and machine translation for multilingual sentiment analysis. *Computer Speech & Language*, 28(1):56-75.

Banea, Carmen, Rada Mihalcea, Janyce Wiebe, Samer Hassan. 2008. Multilingual subjectivity analysis using machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 127-135.

Banea, Carmen, Rada Mihalcea, Janyce Wiebe. 2013. Porting multilingual subjectivity analysis resources across languages. *IEEE Transactions on Affective Computing*, 4(2):211-225.

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

Chandar, Sarath, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. *Advances in Neural Information Processing Systems*, 27:1853-1861.

Cheng, Jianpeng, Dimitri Kartsaklis, and Edward Grefenstette. 2014. Investigating the role of prior disambiguation in deep-learning compositional models of meaning. In *Learning Semantics Workshop NIPS 2014*, 2(1):1-5.

Demirtas, Erkin and Mykola Pechenizkiy. 2013. Cross-lingual polarity detection with machine translation. In *Proceedings of the International Workshop on Issues of Sentiment Discovery and Opinion Mining -WISDOM '13*, pages 9:1-9:8.

Dinu, Georgiana and Marco Baroni. 2014. A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* , pages 90:99

Dinu, Georgiana, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *Proceedings of the 2015 International Conference on Learning Representations*

Duh, Kevin, Akinori Fujino, and Masaaki Nagata. 2011. Is machine translation ripe for cross-lingual sentiment classification? In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers* , 2:429-433.

Gouws, Stephan, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning* , pages 748-756.

Guo, Yuhong and Min Xiao. 2012. Cross language text classification via multi-view subspace learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)* , pages 1615–1622.

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10-18.

Hass, Michael and Yannick Versley. 2015. Subsentential sentiment on a shoestring: A crosslingual analysis of compositional classification. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 694-704.

Hu, Minqing and Bing Liu. 2004. Mining opinion features in customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004)*, pages 168-177.

Iyyer, Mohit, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daume III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681-1691.

Kiritchenko, Svetlana, Xiaodan Zhu, Colin Cherry, and Saif M. Mohammad. 2014. NRC-Canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 437-442.

Klementiev, Alexandre, Ivan Titov, and Binod Bhattarai. 2012. Inducing cross-lingual distributed representations of words. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012): Technical Papers*, pages 1459-1474.

Klinger, Roman and Phillip Cimiano. 2015. Instance selection improves cross-lingual model training for fine-grained sentiment analysis. In *Proceedings of the 19th Conference on computational Natural Language Learning*, pages 153-163.

Koehn, Philipp. 2005. Europarl: A parallel corpus for statistical machine translation. *MT Summit* .

Lambert, Patrik. 2015. Aspect-level cross-lingual sentiment classification with constrained SMT. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 781-787.

Lazaridou, Angeliki, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 781-787.

Lin, Zheng, Xiaolong Jin, Xueke Xu, Weiping Wang, Xueqi Cheng, and Yuanzhuo Wang. 2014. A cross-lingual joint aspect/sentiment model for sentiment analysis. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management - CiKM '14*, pages 1089-1098.

Lu, Bin, Chenhao Tan, Claire Cardie, and Benjamin K. Tsou. 2011. Joint bilingual sentiment classification with unlabeled parallel corpora. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 320-330.

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. *NAACL Workshop on Vector Space Modeling for NLP.*

Mihalcea, Rada, Carmen Banea, and Janyce Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 976-983.

Mikolov, Tomas, Greg Corraddo, Kai Chen, and Jeffrey Dean. 2013. Efficient estimation of word representation in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, pages 1-12.

Mikolov, Tomas, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. In arXiv: 1309.4168.

Moghaddam, Samaneh. 2010. Opinion Digger: An unsupervised opinion miner from unstructured product reviews. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10)*, pages 1825-1828.

Mohammad, Saif M., Mohammad Salameh, and Svetlana Kiritchenko. 2015. How translation alters sentiment. *Journal of Artificial Intelligence Research*, Volume 55, pages 95-130.

Pan, Junfeng, Gui-Rong Xue, Yong Yu, and Yang Wang. 2011. Cross-lingual sentiment classification via bi-view non-negative matrix tri-factorization. In *Proceedings of the 15th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 289-300.

Prettenhofer, Peter and Benno Stein. 2011. Cross-lingual adaptation using structural correspondence learning. *ACM Transactions on Intelligent Systems and Technology*, 3(1):95-130.

Rothe, Sacha and Hinrich Schütze. 2016. Word embedding calculus in meaningful ultradense subspace. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* , pages 512:517.

Tang, Duyu, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embeddings for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1555-1565.

Vulić, Ivan and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research*, Volume 55, pages 953-994.

Wan, Xiaojun. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Conference on Natural Language Processing*, pages 235-243.

Zhou, Guangyou, Zhiyuan Zhu, Tingting He, and Xiaohua Tony Hu. 2016. Cross-lingual sentiment analysis with stacked autoencoders. *Knowledge and Information Systems*, 47(1):27-44.

Zhou, Xinjie, Xiaojun Wan, and Jianguo Xiao. 2012. Cross-language opinion target extraction in review texts. In *Proceedings of the 12th IEEE International Conference on Data Mining*, pages 1200-1205.

Zhu, Jinbo, Huizhen Wang, Benjamin Tsou, and Muhua Zhu. 2009. Multi-aspect opinion polling from textual reviews. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009)*, pages 1799-1802.

A Bilingual Attention Network for Code-switched Emotion Prediction

Zhongqing Wang[†], Yue Zhang[†], Sophia Yat Mei Lee[‡], Shoushan Li[§] and Guodong Zhou[§]
[†] Singapore University of Technology and Design, Singapore
[‡]Department of Chinese and Bilingual Studies, The Hong Kong Polytechnic University
[§] Natural Language Processing Lab, Soochow University, China
wangzq.antony@gmail.com, yue_zhang@sutd.edu.sg
ym.lee@polyu.edu.hk, {lishoushan, gdzhou}@suda.edu.cn

Abstract

Emotions in code-switching text can be expressed in either monolingual or bilingual forms. However, relatively little research has placed emphasis on code-switching text. The challenges of this task include the exploration both monolingual and bilingual information of each post and capturing the informative words from the code-switching context. To address these challenges, we propose a Bilingual Attention Network (BAN) model to aggregate the monolingual and bilingual informative words to form vectors from the document representation, and integrate the attention vectors to predict the emotion. The experiments show the effectiveness of the proposed model. Visualization of the attention layers illustrates that the model selects informative words qualitatively.

1 Introduction

Microblogs such as Twitter and Facebook have gained tremendous popularity in the past decade, they often contain extremely current, even breaking, information about world events. However, the writing style of microblogs tends to be quite colloquial and nonstandard, unlike the style found in more traditional, edited genres (Li et al., 2015; Vo and Zhang, 2015). In addition, authors from multi-lingual communities tend to write code-switching posts frequently (Ling et al., 2013; Wang et al., 2015). These pose challenges for automatic emotion prediction tasks.

There has been some previous research focusing on both emotion analysis (Pang et al., 2002; Lee et al., 2014) and code-switching text analysis (Solorio and Liu, 2008; Ling et al., 2013; Jamatia et al., 2015). However, little research has focused on predicting emotion in code-switching text. Different from monolingual emotion prediction, the emotion in code-switching posts can be expressed in either monolingual or bilingual forms. In this study, we focus on Chinese and English mixed code-switching text from Chinese social media. Although Chinese is the major language, it has been shown that English words are critical for emotion expression (20.1% posts express emotion through English text). E1 - E4 show four examples of code-switching posts that contain both Chinese and English words.

E 1. 玩了一下午轮滑**so happy** ！
(I went rollerblading the whole afternoon, **so happy**!)

E 2. 开学以来，浮躁的情绪。不安稳的心态。确实该自己检讨一下了。。。**sigh** ~
(I have been grumpy and emotional since the first day of school, unstable mindset too. It's really time to self-evaluate ... **sigh**.)

E 3. 上了一天的课，嗓子**hold**不住了啊。
(I have been teaching the whole day, my throat **can't take it anymore**.)

E 4. 早起直接飙酒，喝多上车回校，回校一睁眼过站，多么**happy**的一天。
(I drank too much in the morning. I got drunk and went back to school by bus, and I missed my stop. Such a **happy** day.)

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1624–1634, Osaka, Japan, December 11-17 2016.

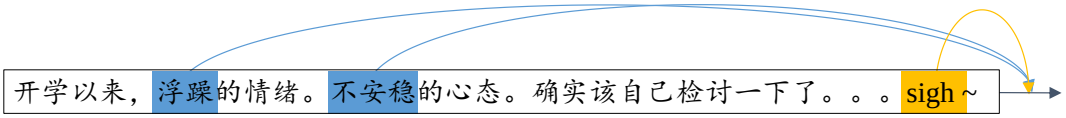


Figure 1: Example of attention mechanism result on E2.

These examples show that it is much more difficult to detect emotions in code-switching text than in the monolingual one, since emotions in code-switching posts can be expressed in either one or two languages. Take E2 for example, the sadness emotion is expressed by both Chinese and English text, and the mention of explicit emotion in English is triggered by the Chinese text. The sadness emotion of E3 is expressed by the mixed Chinese and English phrase "hold 不住". In addition, not all the words in the post are useful for predicting emotion, and the importance of words is highly context-dependent. For example in E2, only the words "浮躁" (*grumpy*) and "sigh" can be used to indicate the sadness emotion, yet the word "happy" in E4 indicates the opposite emotion without the context. Hence, the question how to explore both monolingual and bilingual information of each post post, and how to the informative words and phrases from the code-switching context are what constitute the challenging part.

To address the above challenges, we propose a Bilingual Attention Network (BAN) model to capture informative monolingual and bilingual emotion representations in the code-switching text. The attention mechanism (Bahdanau et al., 2014; Rocktäschel et al., 2015) used to aggregate the representation of informative words into a vector for emotion prediction, provides insight into which words contribute to the classification decision. In addition, a bilingual attention network is used to capture informative words from both monolingual and bilingual context. In particular, we first construct a document representation through a neural network model. Secondly, we project the document representation into three attention vectors by aggregating the representation of the informative words from both monolingual and bilingual context. Finally, a full-connected layer is used to integrate the three attention vectors and predict the emotion.

Our primary contribution is multi-lingual context sensitivity. The model considers both monolingual context and bilingual context, which allow the model to pay relevant attention to informative monolingual and bilingual words, respectively, when constructing relevant document representation. For example, consider the example in Figure 1. We can find that the informative Chinese and English words, such as "浮躁" (*grumpy*) and "sigh", have much more influence to the final decision. The key difference to previous work is that our system uses context to discover when a sequence of tokens is relevant rather than simply filtering for sequences of tokens, taken out of context. Evaluation shows the effectiveness of our proposed BAN model with both monolingual and bilingual information.

2 Related Works

2.1 Emotion Analysis

Over the last decade, there has been much work exploring various aspects of emotion analysis (Wiebe et al., 2005). While most focused on analyzing emotions in monolingual text. Some of these studies emotion lexicon building, for example, Rao et al. (2012) automatically built a word-emotion mapping dictionary for social emotion detection, Yang et al. (2014) proposed a novel emotion-aware topic model to build a domain-specific lexicon. For emotion classification, Liu et al. (2013) used a co-training framework to infer the news from readers' comments and writers' emotions collectively. Wen and Wan (2014) used class sequential rules for emotion classification of microblog texts by regarding each post as a data sequence. Li et al. (2015) proposed a factor graph based framework to incorporate both label and context dependency for emotion classification.

Deep neural networks have been proved effectiveness for many NLP tasks, including sentiment and emotion analysis (Vo and Zhang, 2015; Zhang et al., 2015). dos Santos and Gatti (2014) proposed a character-based deep convolutional neural network to predict sentiment of short text. Tang et al. (2015) proposed a neural network model to learn vector-based document representation. Zhang et al. (2015)

employed a neural network based CRFs for extracting opinion targets on open domains. Most of the previous studies focused on monolingual text, while our proposed bilingual attention network model focuses on exploring the monolingual and bilingual information collectively, and we also propose a new architecture to capture informative words from monolingual and bilingual contexts with attention mechanisms.

2.2 Research on Code-switching and Bilingual Text

Code-switched documents have received considerable attention in the NLP community (Adel et al., 2013; Garrette et al., 2015). Several studies have focused on code-switching identification and analysis, including mining translations in code-switched documents (Ling et al., 2013), predicting code-switched points (Solorio and Liu, 2008), identifying code-switched tokens (Lignos and Marcus, 2013), adding code-switched support to language models (Li and Fung, 2012), and POS tagging for code-switching text (Jamatia et al., 2015). There is relatively little work focus on predicting emotion in code-switching text. Wang et al. (2015) proposed a machine translation based approach to predict emotion in code-switching text with various external resources. Our approach departs from the previous work that we model the task by considering monolingual and bilingual information in both lexical and document level with neural network model and attention mechanism, while previous research only focused on lexical-level bilingual information. In addition, we do not use any external resource, such as bilingual and sentiment dictionary, to train our model.

More remotely connected, multilingual natural language processing has attracted increasing attention in the computational linguistic community due to its broad real-world applications. Relevant studies have been reported in various natural language processing tasks, such as parsing (Burkett and Klein, 2008), information retrieval (Gao et al., 2009), text classification (Amini et al., 2010), and so on. There are a number of studies on predicting sentiment polarity through multilingual text. Wan (2009) incorporated unlabeled data in the target language into classifier with co-training to improve classification performance. Wei and Pal (2010) regarded cross-lingual sentiment classification as a domain adaptation task and applied structural correspondence learning (SCL) to tackle this problem. Their approach achieves a better performance than the co-training algorithm. More recently, Meng et al. (2012) employed the parallel corpus for cross-lingual sentiment classification. They explored the case when no labeled data is available in the parallel corpus. However, such multi-lingual models do not explicitly consider code-switching, since their data sets are always parallel corpus. As the two languages are mixed in the code-switching text without parallel, code-switching corpus is more difficult to process.

3 Bilingual Attention Network

Given a post X with T words ($X =< w_1, w_2, ..., w_T >$), where each word w_t is represented with a K-dimensional embedding (Mikolov et al., 2013), our goal is to predict emotions for each post. Formally, for the post X with the emotion e, we need an objective integer variable y ($y \in \{0, 1\}$) to define if the emotion e is expressed in the post or not. Note that, we have five emotions in our emotion scheme[1], and we build the binary classifier to predict each emotion of the post individually.

There are three phases for predicting code-switched emotion using our bilingual attention network. Firstly, we use a long short-term memory network to build a document representation for each post. Secondly, we project the document representation into three vectors by aggregating the representation of the informative words from both monolingual and bilingual context. Note that, the attention vectors from monolingual context only consider the corresponding monolingual words from the document representation, and the vectors from bilingual context consider all the words from the document representation. Thirdly, a full-connected layer is used to integrate the three attention vectors, and predict the emotion with softmax function. The overall architecture of the Bilingual Attention Network (BAN) is shown in Figure 2.

[1]The emotions includes, *happiness*, *sadness*, *anger*, *fear*, and *surprise*. Please refer to Section 4.1 for more details.

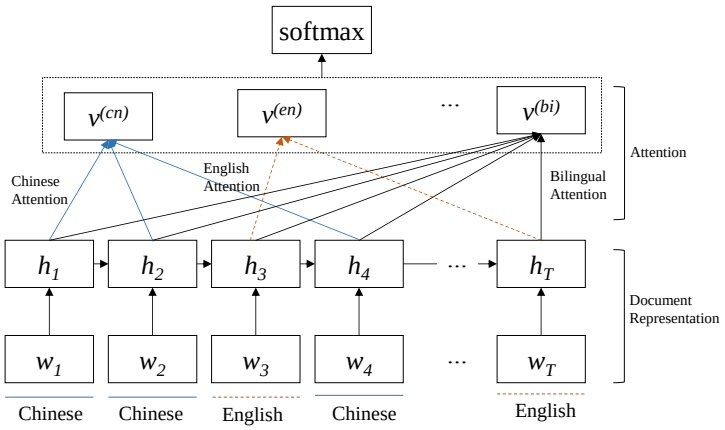


Figure 2: Overview of the bilingual attention network.

3.1 Document Representation

A Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) is used to obtain the document representation of each post. LSTM models a recurrent state transform sequence from a input sequence $\{x_1, x_2, ..., x_t\}$ of the post to a hidden state sequence $\{h_1, h_2, ..., h_t\}$. A LSTM represents each time step with an input, a memory and a output gate, denoted as i_t, f_t and o_t respectively. There are numerous variations to the LSTM model, and we choose one for which the hidden state h_t for each time-step t is given by:

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{i-1} + b^{(i)}) \tag{1}$$

$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) \tag{2}$$

$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}) \tag{3}$$

$$u_t = \tanh(W^{(u)} + U^{(u)}h_{t-1} + b^{(u)}) \tag{4}$$

$$c_t = i_t \odot u_t + f_t \odot c_{t-1} \tag{5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

where σ denotes the sigmoid function. After the LSTM process, we obtain an **annotation** h_t for a given word w_t.

3.2 Attention Mechanism

Not all words contribute equally to the representation of the meaning. Hence, we introduce an attention mechanism (Bahdanau et al., 2014; Yang et al., 2016) to extract the words that are important to the meaning of the post, and aggregate the representation of those informative words to form a vector. Since emotion can be expressed in either one or two languages in code-switching text, we build the vectors from monolingual and bilingual contexts respectively. For the monolingual case, we build two vectors $v^{(cn)}$ and $v^{(en)}$ to capture informative information from the Chinese and English contexts separately. For the bilingual case, we construct a vector $v^{(bi)}$ to capture the salient words from the mixed text.

Bilingual Attention. We use an attention function to aggregate the representation of the salient words to form the bilingual attention vector $v^{(bi)}$. Specifically,

$$u_t^{(bi)} = \tanh(W^{(bi)}h_t^{(bi)} + b^{(bi)}) \tag{7}$$

$$\alpha_t^{(bi)} = \frac{\exp(u_t^{(bi)\mathrm{T}}u^{(bi)})}{\sum_t \exp(u_t^{(bi)\mathrm{T}}u^{(bi)})} \tag{8}$$

$$v^{(bi)} = \sum_t \alpha_t^{(bi)}h_t^{(bi)} \tag{9}$$

In the above equations, we first feed the bilingual word annotations $h_t^{(bi)}$ through a one-layer perceptron to get $u_t^{(bi)}$ as a hidden representation of $h_t^{(bi)}$ (Eq. 7), and then measure the importance of the word by measuring the similarity of $u_t^{(bi)}$ with a word-level context vector $u^{(bi)}$ obtaining a normalized importance weight α_t (Eq. 8). After that, we compute the bilingual attention vector v^{bi} as a weighted sum of the word annotations based on the weight (Eq. 9). The context vector $u^{(bi)}$ can be seen as a high-level informative representation of the words in memory networks (Kumar et al., 2015). The word context vector $u^{(bi)}$ are randomly initialized and jointly learned during the training process.

Monolingual Attention. To reward to the most relevant words from the two monolingual contexts for emotion classification, we again use an attention functions on the monolingual context to measure their importance.

$$u_t^{(mo)} = \tanh(W^{(mo)} h_t^{(mo)} + b^{(mo)}) \tag{10}$$

$$\alpha_t^{(mo)} = \frac{\exp(u_t^{(mo)\mathrm{T}} u^{(mo)})}{\sum_t \exp(u_t^{(mo)\mathrm{T}} u^{(mo)})} \tag{11}$$

$$v^{(mo)} = \sum_t \alpha_t^{(mo)} h_t^{(mo)} \tag{12}$$

Since we only consider monolingual contexts, the input of attention function is the annotation h_t of Chinese or English words respectively. When Chinese is used as the monolingual context, $v^{(cn)}$ equals $v^{(mo)}$ in Eq. 12. and $v^{(cn)}$ equals $v^{(mo)}$ when considering English as the monolingual context.

3.3 Prediction

The monolingual vector ($v^{(cn)}$, $v^{(en)}$) and bilingual vector ($v^{(bi)}$) with the attention mechanism mentioned above are concatenated into a single vector $F = [v^{(cn)}, v^{(en)}, v^{(bi)}]$. We then use a softmax classifier to predict the label y give the inputs X. The classifier takes the feature vector $f \in F$ as input:

$$\widehat{p}_\theta(y|X) = \mathrm{softmax}(W^{(s)} f + b^{(s)}) \tag{13}$$

$$\widehat{y} = \arg\max_y \widehat{p}_\theta(y|X) \tag{14}$$

Training cost function is the negative log-likelihood of the true class labels $y^{(k)}$ at each labeled node ($k \in \{0, 1\}$):

$$J(\theta) = -\frac{1}{2} \sum_{k=0}^{1} \log \widehat{p}_\theta(y^{(k)}|X^{(k)}) + \frac{\lambda}{2} ||\theta||_2^2 \tag{15}$$

where the superscript k indicates the k^{th} labeled node, and λ is an L2 regularization hyperparameter.

We apply online training, where model parameters are optimized by using Adagrad (Duchi et al., 2011). In order to avoid over-fitting, dropout (Hinton et al., 2012) is used to the word embedding with a ratio of 0.2. For LSTM models, we empirically set size of the hidden layer is 32. We train the word embedding using the *Skip-gram* algorithm[2].

4 Experiments

4.1 Dataset and Statistics

We focus on emotion prediction in code-switching text which is defined as text that contains more than one language ("code"). We use Chinese and English code-switching posts for experimental study, and the data set is taken from Weibo.com, one of the popular Chinese social media websites. Our dataset is collected and annotated by Wang et al. (2015). Following their setting, five basic emotions are defined as candidate emotions, namely *happiness*, *sadness*, *fear*, *anger* and *surprise*. After removing those posts containing noise and advertisements, there are 3,530 posts express emotions.

[2] https://code.google.com/p/word2vec/

Emotion	Chinese	English	Both
Happiness	0.670	0.127	0.203
Sadness	0.835	0.046	0.119
Anger	0.706	0.068	0.226
Fear	0.901	0.026	0.073
Surprise	0.883	0.055	0.062

Table 1: Joint distribution of emotion and causal situations.

Emotions can be expressed through the two languages separately or collectively, and we focus four types of causal situations for each emotion, namely, *None*, *English*, *Chinese*, and *Both*. *None* means that the post does not contain any corresponding emotions (E5). *Chinese* means that the emotion is expressed through the Chinese text only (E6). *English* means that the emotion is expressed through the English text only (E1). *Both* means that the emotions of the post are expressed through both Chinese and English text (E2).

The joint distribution between emotions and caused situations is illustrated in Table 1. Each cell presents the conditional probability $p(l_j|e_i)$ of the emotion e_i based on the situation l_j. From the table, we can find that, most of emotional posts are expressed through Chinese text due to Chinese being the major language. Although English text contains relatively fewer words in each post, 20.1% of emotional posts are expressed through English. It indicates that English is of vital importance to emotion expression even in code-switching contexts dominated by Chinese. And more notably, 13.7% emotional posts are conducted in both Chinese and English. It indicates that Chinese and English text would be influenced by each other, and the bilingual context would also be effective for predicting emotion in code-switching text.

E 5. 还木有看《起风了》，**mark**一下，还有《虞美人盛开的山坡》。
(I've never seen the "Kaze tachinu" and "Kokuriko-zaka kara", **mark** as a note.)

E 6. 静静坐下来看别人**show**啦。刚刚在节目里看到妈咪和弟的视频真的很意外！
(I sat down quietly to watch someone else's **show**. To my surprise, both my mother and brother appeared on the programme.)

4.2 Experiment Setting

After constructing the dataset, we randomly selected half of the annotated posts as the training data and the other half as the testing data. We use FudanNLP[3] for Chinese word segmentation and adopt the F1-Measure (F1.) to evaluate the performance of emotion prediction.

4.3 Baselines

Our first group of experiments is to investigate whether our proposed approach improves emotion prediction in code-switching text compared with state-of-the-art monolingual emotion prediction methods. For fair comparison, the following models are implemented.

- *Term-Counting (TC)* counts the Chinese and English emotional cue words for each post to predict the emotion (Tunery, 2002).
- *SVM* is the basic model which uses all the Chinese and English text of each post as features[4], we consider bag-of-words as features.
- *BLP-BS* is proposed by Wang et al., (2015), employs a Bipartite graph based Label Propagation framework with lexical level Bilingual and Sentimental information. We re-implement their approach.

[3] https://github.com/FudanNLP/fnlp
[4] SVM^{light} is used as the implementation for the SVM classifier, http://svmlight.joachims.org

Emotion	TC	SVM	BLP-BS	LSTM	BAN
Happiness	0.258	0.591	0.638	0.662	0.678
Sadness	0.207	0.573	0.628	0.614	0.634
Anger	0.194	0.677	0.700	0.659	0.728
Fear	0.187	0.719	0.693	0.700	0.728
Surprise	0.211	0.548	0.560	0.575	0.594
Average	0.211	0.622	0.645	0.642	0.672

Table 2: Comparison with baselines.

	CN	EN	All	Comb
LSTM	0.627	0.579	0.642	0.656
Attention	0.635	0.590	0.663	0.672

Table 3: Influence of Different Factors.

- *LSTM* uses the mixed code-switching text as the input to train a basic LSTM model, using the last hidden state vector h_n directly for emotion prediction. This services as a neural network baseline without different monolingual and bilingual context.

- *BAN* is our proposed model, which uses attention to capture the informative words from both monolingual and bilingual context.

Table 2 shows the experimental results. From the table we can find that, 1) the performance of Term-counting is unacceptable since many emotions are expressed implicitly and the importance of words is highly context-dependent. 2) Since the neural network model can capture richer features automatically, LSTM outperforms the bag-of-word SVM model in most emotions. 3) Our proposed BAN model significantly outperforms all other approaches (p-value < 0.01). This indicates the effectiveness of attention mechanism from bilingual and monolingual context, compared to learning the monolingual information. Moreover, as BAN outperforms the BLP-BS model, it shows that our proposed model can automatically capture more valuable information. Note that, BLP-BS use many external resources, such as bilingual and sentimental dictionary, while our proposed model does not use any external resources.

4.4 Influence of Different Factors

Table 3 shows the influence of different factors on our proposed model. *LSTM* represents use basic LSTM as the prediction model, and *Attention* represents the LSTM model with attention. *CN* means only considering the Chinese text of each post as input, *EN* means only considering English text, *All* means using the mixed code-switching text as input. Specificlly, *LSTM-Comb* means merge the output of LSTM-CN, LSTM-EN and LSTM-All into the full connected layer, and then get the prediction results with softmax function, and *Attention-Comb* is same as the proposed BAN model that integrate both monolingual and bilingual attention mechanisms from the document representation.

The table shows us that 1) as the English is the minor language in our corpus, the results of LSTM-EN and Attention-EN are relatively weak. 2) the model using mixed code-switching text (*All*) always outperforms the model using the monolingual text individually, indicating the bilingual information is more useful than mere monolingual information. 3) Exploring monolingual and bilingual information collectively, the LSTM-Comb model can improve the accuracy over the basic LSTM model. It indicates both monolingual (i.e., Chinese and English) and bilingual texts are useful for predicting emotion in code-switching text. 4) By aggregating the representation of the informative words, the attention-based model always outperform the traditional LSTM model. 5) Our model Attention-Comb (BAN) that further utilizes attention mechanism to capture the informative words from both monolingual and bilingual context can outperforms the previous models.

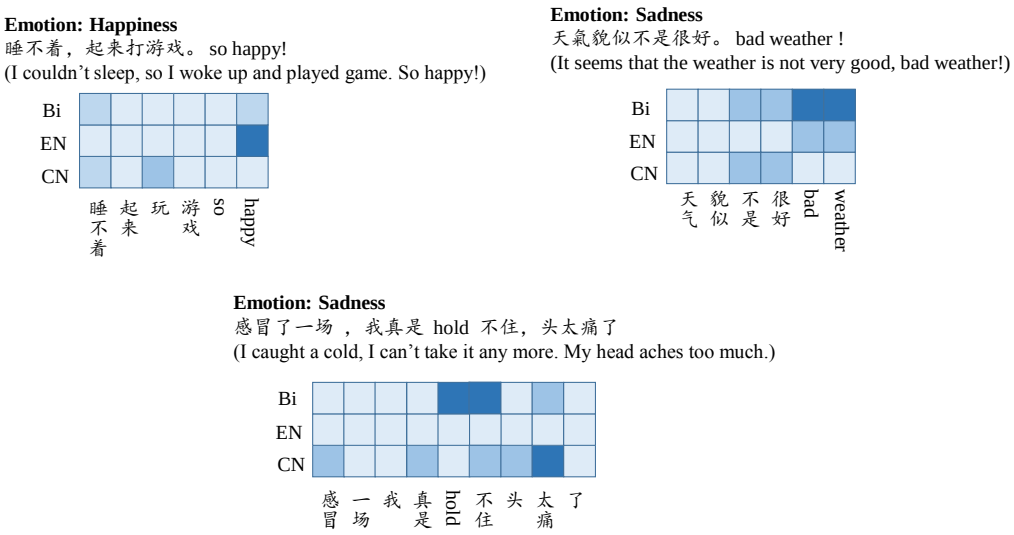


Figure 3: Example attention results.

4.5 Visualization of Attention

In order to validate that our model is able to select informative words in a post, we visualize the attention layers for several posts from our corpus in Figures 3 . Blue denotes the word weight. Since we have three attention mechanisms for monolingual and bilingual context, the first blue line denote the word weight for bilingual attention, and the other two lines denote the word weight for the Chinese and English attention respectively.

The figure shows that our model can select the words carrying strong sentiment signals such "happy", "bad", and "很好"(*very good*). In addition, since different attention functions consider different contexts, the monolingual attention functions always select the monolingual sentimental words with corresponding language such as, "happy", and "很好"(*very good*). The bilingual attention function can select mixed sentimental phrases, such as "hold不住" (*can't hold any more*). The joint attention mechanism can also deal with complex contexts. For example, in the first case, the weight of the sadness emotional word "睡不着" (*couldn't sleep*) is high with Chinese attention function, although the emotion of the whole post is happiness. However, by considering the English and bilingual attention functions, we can find the weight of word "happy" is higher than " 睡不着" (*couldn't sleep*), and it can lead us to the correct emotion.

5 Conclusion

In this paper, we addressed a novel yet important task, namely emotion detection in code-switching text. The challenges include that we need to consider both monolingual and bilingual information, and we need to extract the salient words from both monolingual and bilingual contexts. To address these challenges, a bilingual attention network model is proposed to capture the representations of monolingual and bilingual information in the code-switching text respectively. A LSTM model is used to construct the document-level representation of each post, and attention mechanism is used to capture the informative words from both monolingual and bilingual contexts. Empirical studies demonstrated that our model can significantly outperform several strong baselines.

6 Acknowledgments

Shoushan Li is the corresponding author. We thank our anonymous reviewers for prudent advice. The work is funded by an Early Career Scheme (ECS) sponsored by the Research Grants Council of Hong Kong (No. PolyU 5593/13H) and a PolyU Faculty Research Grant (No. 1-ZVEK), and supported by the

National Natural Science Foundation of China (No. 61273320, and No. 61375073) and the Key Project of the National Natural Science Foundation of China (No. 61331011).

References

Heike Adel, Ngoc Thang Vu, and Tanja Schultz. 2013. Combination of recurrent neural networks and factored language models for code-switching language modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*, pages 206–211.

Massih-Reza Amini, Cyril Goutte, and Nicolas Usunier. 2010. Combining coregularization and consensus-based self-training for multilingual text categorization. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, pages 475–482.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu, Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 877–886.

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 69–78.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Wei Gao, John Blitzer, Ming Zhou, and Kam-Fai Wong. 2009. Exploiting bilingual information to improve web search. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 1075–1083.

Dan Garrette, Hannah Alpert-Abrams, Taylor Berg-Kirkpatrick, and Dan Klein. 2015. Unsupervised code-switching for multilingual historical document transcription. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1036–1041.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

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

Anupam Jamatia, Björn Gambäck, and Amitava Das. 2015. Part-of-speech tagging for code-mixed english-hindi twitter and facebook chat messages. In *Recent Advances in Natural Language Processing, RANLP 2015, 7-9 September, 2015, Hissar, Bulgaria*, pages 239–248.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285.

Sophia Yat Mei Lee, Shoushan Li, and Chu-Ren Huang. 2014. Annotating events in an emotion corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014), Reykjavik, Iceland, May 26-31, 2014.*, pages 3511–3516.

Ying Li and Pascale Fung. 2012. Code-switch language model with inversion constraints for mixed language speech recognition. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, pages 1671–1680.

Shoushan Li, Lei Huang, Rong Wang, and Guodong Zhou. 2015. Sentence-level emotion classification with label and context dependence. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1045–1053.

Constantine Lignos and Mitch Marcus. 2013. Toward web-scale analysis of codeswitching. In *Proceedings of Annual Meeting of the Linguistic Society of America*.

Wang Ling, Guang Xiang, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2013. Microblogs as parallel corpora. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 176–186.

Huanhuan Liu, Shoushan Li, Guodong Zhou, Chu-Ren Huang, and Peifeng Li. 2013. Joint modeling of news reader's and comment writer's emotions. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*, pages 511–515.

Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Ge Xu, and Houfeng Wang. 2012. Cross-lingual mixture model for sentiment classification. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, pages 572–581.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. *CoRR*, cs.CL/0205070.

Yanghui Rao, Xiaojun Quan, Liu Wenyin, Qing Li, and Mingliang Chen. 2012. Building word-emotion mapping dictionary for online news. In *The 1st International Workshop on Sentiment Discovery from Affective Data.*

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664.

Thamar Solorio and Yang Liu. 2008. Learning to predict code-switching points. In *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu, Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 973–981.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1422–1432.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1347–1353.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 235–243.

Zhongqing Wang, Sophia Yat Mei Lee, Shoushan Li, and Guodong Zhou. 2015. Emotion detection in code-switching texts via bilingual and sentimental information. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 763–768.

Bin Wei and Christopher J. Pal. 2010. Cross lingual adaptation: An experiment on sentiment classifications. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, Short Papers*, pages 258–262.

Shiyang Wen and Xiaojun Wan. 2014. Emotion classification in microblog texts using class sequential rules. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 187–193.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210.

Min Yang, Dingju Zhu, and Kam-Pui Chow. 2014. A topic model for building fine-grained domain-specific emotion lexicon. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 421–426.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL 2016, 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, US*.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 612–621.

UTCNN: a Deep Learning Model of Stance Classification on Social Media Text

Wei-Fan Chen
Institute of Information Science,
Academia Sinica, Taipei, Taiwan.
vieric wf@iis.sinica.edu.tw

Lun-Wei Ku
Institute of Information Science,
Academia Sinica, Taipei, Taiwan.
lwku@iis.sinica.edu.tw

Abstract

Most neural network models for document classification on social media focus on text information to the neglect of other information on these platforms. In this paper, we classify post stance on social media channels and develop UTCNN, a neural network model that incorporates user tastes, topic tastes, and user comments on posts. UTCNN not only works on social media texts, but also analyzes texts in forums and message boards. Experiments performed on Chinese Facebook data and English online debate forum data show that UTCNN achieves a 0.755 macro-average f-score for supportive, neutral, and unsupportive stance classes on Facebook data, which is significantly better than models in which either user, topic, or comment information is withheld. This model design greatly mitigates the lack of data for the minor class without the use of oversampling. In addition, UTCNN yields a 0.842 accuracy on English online debate forum data, which also significantly outperforms results from previous work as well as other deep learning models, showing that UTCNN performs well regardless of language or platform.

1 Introduction

Deep neural networks have been widely used in text classification and have achieved promising results (Lai et al., 2015; Ren et al., 2016; Huang et al., 2016). Most focus on content information and use models such as convolutional neural networks (CNN) (Kim, 2014) or recursive neural networks (Socher et al., 2013). However, for user-generated posts on social media like Facebook or Twitter, there is more information that should not be ignored. On social media platforms, a user can act either as the author of a post or as a reader who expresses his or her comments about the post.

In this paper, we classify posts taking into account post authorship, likes, topics, and comments. In particular, users and their "likes" hold strong potential for text mining. For example, given a set of posts that are related to a specific topic, a user's likes and dislikes provide clues for stance labeling. From a user point of view, users with positive attitudes toward the issue leave positive comments on the posts with praise or even just the post's content; from a post point of view, positive posts attract users who hold positive stances. We also investigate the influence of topics: different topics are associated with different stance labeling tendencies and word usage. For example we discuss women's rights and unwanted babies on the topic of abortion, but we criticize medicine usage or crime when on the topic of marijuana (Hasan and Ng, 2014). Even for posts on a specific topic like nuclear power, a variety of arguments are raised: green energy, radiation, air pollution, and so on. As for comments, we treat them as additional text information. The arguments in the comments and the commenters (the users who leave the comments) provide hints on the post's content and further facilitate stance classification.

In this paper, we propose the user-topic-comment neural network (UTCNN), a deep learning model that utilizes user, topic, and comment information. We attempt to learn user and topic representations which encode user interactions and topic influences to further enhance text classification, and we also incorporate comment information. We evaluate this model on a post stance classification task on forum-style social media platforms. The contributions of this paper are as follows: 1. We propose UTCNN,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1635–1645, Osaka, Japan, December 11-17 2016.

a neural network for text in modern social media channels as well as legacy social media, forums, and message boards — anywhere that reveals users, their tastes, as well as their replies to posts. 2. When classifying social media post stances, we leverage users, including authors and likers. User embeddings can be generated even for users who have never posted anything. 3. We incorporate a topic model to automatically assign topics to each post in a single topic dataset. 4. We show that overall, the proposed method achieves the highest performance in all instances, and that all of the information extracted, whether users, topics, or comments, still has its contributions.

2 Related Work

2.1 Extra-Linguistic Features for Stance Classification

In this paper we aim to use text as well as other features to see how they complement each other in a deep learning model. In the stance classification domain, previous work has showed that text features are limited, suggesting that adding extra-linguistic constraints could improve performance (Bansal et al., 2008; Hasan and Ng, 2013a; Walker et al., 2012). For example, Hasan and Ng as well as Thomas et al. require that posts written by the same author have the same stance (Hasan and Ng, 2013b; Thomas et al., 2006). The addition of this constraint yields accuracy improvements of 1–7% for some models and datasets. Hasan and Ng later added user-interaction constraints and ideology constraints (Hasan and Ng, 2013a): the former models the relationship among posts in a sequence of replies and the latter models inter-topic relationships, e.g., users who oppose abortion could be conservative and thus are likely to oppose gay rights.

For work focusing on online forum text, since posts are linked through user replies, sequential labeling methods have been used to model relationships between posts. For example, Hasan and Ng use hidden Markov models (HMMs) to model dependent relationships to the preceding post (Hasan and Ng, 2013b); Burfoot et al. use iterative classification to repeatedly generate new estimates based on the current state of knowledge (Burfoot et al., 2011); Sridhar et al. use probabilistic soft logic (PSL) to model reply links via collaborative filtering (Sridhar et al., 2015). In the Facebook dataset we study, we use comments instead of reply links. However, as the ultimate goal in this paper is predicting not comment stance but post stance, we treat comments as extra information for use in predicting post stance.

2.2 Deep Learning on Extra-Linguistic Features

In recent years neural network models have been applied to document sentiment classification (Socher et al., 2012; Socher et al., 2013; Kalchbrenner et al., 2014; Johnson and Zhang, 2015; Huang et al., 2016). Text features can be used in deep networks to capture text semantics or sentiment. For example, Dong et al. use an adaptive layer in a recursive neural network for target-dependent Twitter sentiment analysis, where targets are topics such as *windows 7* or *taylor swift* (Dong et al., 2014a; Dong et al., 2014b); recursive neural tensor networks (RNTNs) utilize sentence parse trees to capture sentence-level sentiment for movie reviews (Socher et al., 2013); Le and Mikolov predict sentiment by using paragraph vectors to model each paragraph as a continuous representation (Le and Mikolov, 2014). They show that performance can thus be improved by more delicate text models.

Others have suggested using extra-linguistic features to improve the deep learning model. The user-word composition vector model (UWCVM) (Tang et al., 2015b) is inspired by the possibility that the strength of sentiment words is user-specific; to capture this they add user embeddings in their model. In UPNN, a later extension, they further add a product-word composition as product embeddings, arguing that products can also show different tendencies of being rated or reviewed (Tang et al., 2015a). Their addition of user information yielded 2–10% improvements in accuracy as compared to the above-mentioned RNTN and paragraph vector methods. We also seek to inject user information into the neural network model. In comparison to the research of Tang et al. on sentiment classification for product reviews, the difference is two-fold. First, we take into account multiple users (one author and potentially many likers) for one post, whereas only one user (the reviewer) is involved in a review. Second, we add comment information to provide more features for post stance classification. None of these two factors have been considered previously in a deep learning model for text stance classification. Therefore, we

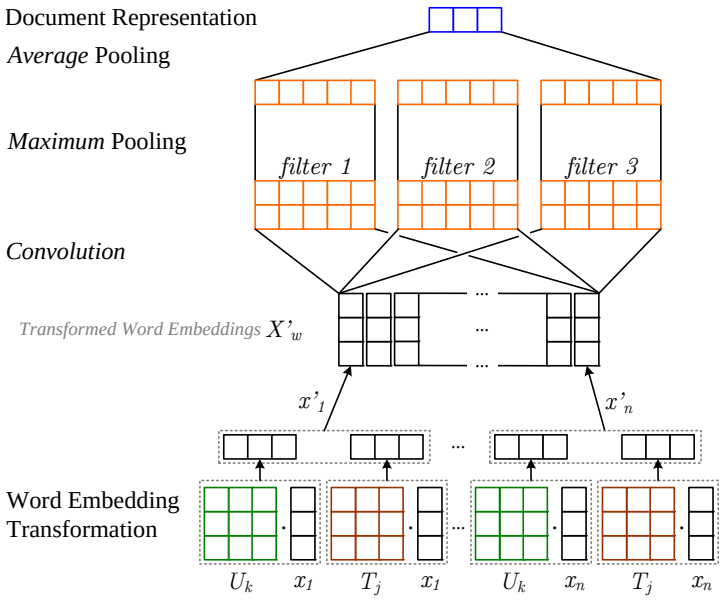


Figure 1: Document composition in a convolutional neural network with three convolutional filters and user- and topic-dependent semantic transformations. Respectively, x_w is the word embedding of word w, x'_w is the word embedding of word w after transformation, U_k and T_j are user and topic matrix embeddings for user k and topic j.

propose UTCNN, which generates and utilizes user embeddings for all users — even for those who have not authored any posts — and incorporates comments to further improve performance.

3 Method

In this section, we first describe CNN-based document composition, which captures user- and topic-dependent document-level semantic representation from word representations. Then we show how to add comment information to construct the user-topic-comment neural network (UTCNN).

3.1 User- and Topic-dependent Document Composition

As shown in Figure 1, we use a general CNN (Kim, 2014) and two semantic transformations for document composition [1] . We are given a document with an engaged user k, a topic j, and its composite n words, each word w of which is associated with a word embedding $x_w \in \mathbb{R}^d$ where d is the vector dimension. For each word embedding x_w, we apply two dot operations as shown in Equation 1:

$$x'_w = [U_k \cdot x_w; T_j \cdot x_w] \tag{1}$$

where $U_k \in \mathbb{R}^{d_u \times d}$ models the user reading preference for certain semantics, and $T_j \in \mathbb{R}^{d_t \times d}$ models the topic semantics; d_u and d_t are the dimensions of transformed user and topic embeddings respectively. We use U_k to model semantically what each user prefers to read and/or write, and use T_j to model the semantics of each topic. The dot operation of U_k and x_w transforms the global representation x_w to a user-dependent representation. Likewise, the dot operation of T_j and x_w transforms x_w to a topic-dependent representation.

After the two dot operations on x_w, we have user-dependent and topic-dependent word vectors $U_k \cdot x_w$ and $T_j \cdot x_w$, which are concatenated to form a user- and topic-dependent word vector x'_w. Then the transformed word embeddings $X'_w = [x'_1; x'_2; ...; x'_n]$ are used as the CNN input. Here we apply three convolutional layers on the concatenated transformed word embeddings $x'_c = [x'_m; x'_{m+1}; ...; x'_{m+l_{cf}-1}] \in \mathbb{R}^{d \cdot l_{cf}}$:

$$h_{cf} = f\left(W_{cf} \cdot x'_c + b_{cf}\right) \tag{2}$$

[1] Here by saying *document*, we mean the user-generated content in a post or a comment.

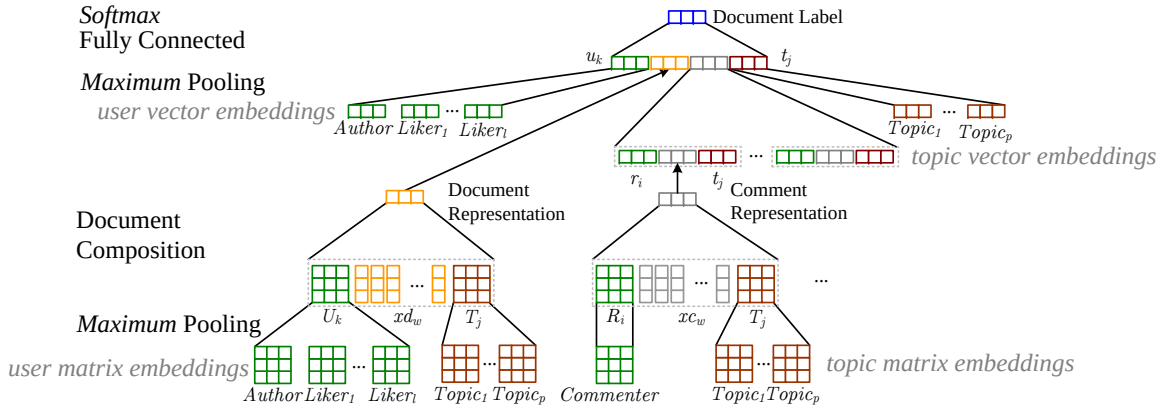


Figure 2: The UTCNN model. Assuming one post author, l likers and p topics, xd_w is the word embedding of word w in the document; xc_w is the word embedding of word w in the comments; U_k and u_k are the moderator matrix and vector embedding for moderator k; T_j and t_j are the topic matrix and vector embedding for topic j; R_i and r_i are the commenter matrix and vector embedding for commenter i. For simplicity we do not explicitly plot the topic vector embedding part for comments, but it does include a maximum pooling layer as with documents.

where m is the index of words; f is a non-linear activation function (we use $\tanh$[2]); $W_{cf} \in \mathbb{R}^{len \times d \cdot l_{cf}}$ is the convolutional filter with input length $d \cdot l_{cf}$ and output length len, where l_{cf} is the window size of the convolutional operation; and h_{cf} and b_{cf} are the output and bias of the convolution layer cf, respectively. In our experiments, the three window sizes l_{cf} in the three convolution layers are one, two, and three, encoding unigram, bigram, and trigram semantics accordingly.

After the convolutional layer, we add a maximum pooling layer among convolutional outputs to obtain the unigram, bigram, and trigram *n*-gram representations. This is succeeded by an average pooling layer for an element-wise average of the three maximized convolution outputs.

3.2 UTCNN Model Description

Figure 2 illustrates the UTCNN model. As more than one user may interact with a given post, we first add a maximum pooling layer after the user matrix embedding layer and user vector embedding layer to form a moderator matrix embedding U_k and a moderator vector embedding u_k for moderator k respectively, where U_k is used for the semantic transformation in the document composition process, as mentioned in the previous section. The term *moderator* here is to denote the pseudo user who provides the overall semantic/sentiment of all the engaged users for one document. The embedding u_k models the moderator stance preference, that is, the pattern of the revealed user stance: whether a user is willing to show his preference, whether a user likes to show impartiality with neutral statements and reasonable arguments, or just wants to show strong support for one stance. Ideally, the latent user stance is modeled by u_k for each user. Likewise, for topic information, a maximum pooling layer is added after the topic matrix embedding layer and topic vector embedding layer to form a joint topic matrix embedding T_j and a joint topic vector embedding t_j for topic j respectively, where T_j models the semantic transformation of topic j as in users and t_j models the topic stance tendency. The latent topic stance is also modeled by t_j for each topic.

As for comments, we view them as short documents with authors only but without likers nor their own comments[3]. Therefore we apply document composition on comments although here users are commenters (users who comment). It is noticed that the word embeddings x_w for the same word in the posts and comments are the same, but after being transformed to x'_w in the document composition process

[2]Some papers suggest using $ReLU$ as the activation function in deep CNNs with many layers. Nevertheless, we use $\tanh$ as the activation function, as our model is moderately deep and empirically we found the impact to be limited.

[3]Recently Facebook released a function allowing likes and comments on comments, but it was not available during the time we collected data. However, UTCNN works on this richer data, as comments are treated as posts under this framework.

Dataset	FBFans				CreateDebate							
Type	*Sup*	*Neu*	*Uns*	All	ABO		GAY		OBA		MAR	
					F	*A*	*F*	*A*	*F*	*A*	*F*	*A*
Training	7,097	19,412	245	26,754	770.4	622.4	700.8	400.0	420.8	367.2	355.2	145.6
Development	155	2,785	11	2,951	-	-	-	-	-	-	-	-
Testing	252	2,619	19	2,890	192.6	155.6	175.2	100.0	105.2	91.8	88.8	36.4
All	7,504	24,816	275	32,595	963.0	778.0	876.0	500.0	526.0	459.0	444.0	182.0

Table 1: Annotation results of FBFans and CreateDebate dataset.

Author \ Post	*Sup*	*Neu*	*Uns*
Sup	**58.5%**	**51.3**%	29.4%
Neu	33.9%	43.5 %	9.3%
Uns	7.6%	5.2%	**61.3**%

Table 2: Distribution of like behavior.

shown in Figure 1, they might become different because of their different engaged users. The output comment representation together with the commenter vector embedding r_i and topic vector embedding t_j are concatenated and a maximum pooling layer is added to select the most important feature for comments. Instead of requiring that the comment stance agree with the post, UTCNN simply extracts the most important features of the comment contents; they could be helpful, whether they show obvious agreement or disagreement. Therefore when combining comment information here, the maximum pooling layer is more appropriate than other pooling or merging layers. Indeed, we believe this is one reason for UTCNN's performance gains.

Finally, the pooled comment representation, together with user vector embedding u_k, topic vector embedding t_j, and document representation are fed to a fully connected network, and softmax is applied to yield the final stance label prediction for the post.

4 Experiment

We start with the experimental dataset and then describe the training process as well as the implementation of the baselines. We also implement several variations to reveal the effects of features: authors, likers, comment, and commenters. In the results section we compare our model with related work.

4.1 Dataset

We tested the proposed UTCNN on two different datasets: FBFans and CreateDebate. FBFans is a privately-owned[4], single-topic, Chinese, unbalanced, social media dataset, and CreateDebate is a public, multiple-topic, English, balanced, forum dataset. Results using these two datasets show the applicability and superiority for different topics, languages, data distributions, and platforms.

The FBFans dataset contains data from anti-nuclear-power Chinese Facebook fan groups from September 2013 to August 2014, including posts and their author and liker IDs. There are a total of 2,496 authors, 505,137 likers, 33,686 commenters, and 505,412 unique users. Two annotators were asked to take into account only the post content to label the stance of the posts in the whole dataset as *supportive*, *neutral*, or *unsupportive* (hereafter denoted as *Sup*, *Neu*, and *Uns*). *Sup*/*Uns* posts were those in support of or against anti-reconstruction; *Neu* posts were those evincing a neutral standpoint on the topic, or were irrelevant. Raw agreement between annotators is 0.91, indicating high agreement. Specifically, Cohen's Kappa for *Neu* and not *Neu* labeling is 0.58 (moderate), and for *Sup* or *Uns* labeling is

[4]Currently not released due to copyright and privacy issues.

0.84 (almost perfect). Posts with inconsistent labels were filtered out, and the development and testing sets were randomly selected from what was left. Posts in the development and testing sets involved at least one user who appeared in the training set. The number of posts for each stance is shown on the left-hand side of Table 1. About twenty percent of the posts were labeled with a stance, and the number of supportive (*Sup*) posts was much larger than that of the unsupportive (*Uns*) ones: this is thus highly skewed data, which complicates stance classification. On average, 161.1 users were involved in one post. The maximum was 23,297 and the minimum was one (the author). For comments, on average there were 3 comments per post. The maximum was 1,092 and the minimum was zero.

To test whether the assumption of this paper – posts attract users who hold the same stance to like them – is reliable, we examine the likes from authors of different stances. Posts in FBFans dataset are used for this analysis. We calculate the like statistics of each distinct author from these 32,595 posts. As the numbers of authors in the *Sup*, *Neu* and *Uns* stances are largely imbalanced, these numbers are normalized by the number of users of each stance. Table 4 shows the results. Posts with stances (i.e., not neutral) attract users of the same stance. Neutral posts also attract both supportive and neutral users, like what we observe in supportive posts, but just the neutral posts can attract even more neutral likers. These results do suggest that users prefer posts of the same stance, or at least posts of no obvious stance which might cause annoyance when reading, and hence support the user modeling in our approach.

The CreateDebate dataset was collected from an English online debate forum[5] discussing four topics: abortion (ABO), gay rights (GAY), Obama (OBA), and marijuana (MAR). The posts are annotated as *for* (F) and *against* (A). Replies to posts in this dataset are also labeled with stance and hence use the same data format as posts. The labeling results are shown in the right-hand side of Table 1. We observe that the dataset is more balanced than the FBFans dataset. In addition, there are 977 unique users in the dataset. To compare with Hasan and Ng's work, we conducted five-fold cross-validation and present the annotation results as the average number of all folds (Hasan and Ng, 2013b; Hasan and Ng, 2014).

The FBFans dataset has more integrated functions than the CreateDebate dataset; thus our model can utilize all linguistic and extra-linguistic features. For the CreateDebate dataset, on the other hand, the like and comment features are not available (as there is a stance label for each reply, replies are evaluated as posts as other previous work) but we still implemented our model using the content, author, and topic information.

4.2 Settings

In the UTCNN training process, cross-entropy was used as the loss function and *AdaGrad* as the optimizer. For FBFans dataset, we learned the 50-dimensional word embeddings on the whole dataset using GloVe[6] (Pennington et al., 2014) to capture the word semantics; for CreateDebate dataset we used the publicly available English 50-dimensional word embeddings, pre-trained also using GloVe. These word embeddings were fixed in the training process. The learning rate was set to 0.03. All user and topic embeddings were randomly initialized in the range of [-0.1 0.1]. Matrix embeddings for users and topics were sized at 250 (5×50); vector embeddings for users and topics were set to length 10.

We applied the LDA topic model (Blei et al., 2003) on the FBFans dataset to determine the latent topics with which to build topic embeddings, as there is only one general known topic: nuclear power plants. We learned 100 latent topics and assigned the top three topics for each post. For the CreateDebate dataset, which itself constitutes four topics, the topic labels for posts were used directly without additionally applying LDA.

For the FBFans data we report class-based f-scores as well as the macro-average f-score (F_1^{SNU}) shown in equation 3.

$$F_1^{SNU} = 2 \cdot \frac{P^{SNU} \cdot R^{SNU}}{P^{SNU} + R^{SNU}} \tag{3}$$

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

[6] http://nlp.stanford.edu/projects/glove/

Method	Features				F-score			F_1^{SNU}
	Content	User	Topic	Comment	*Sup*	*Neu*	*Uns*	
Majority					.000	.841	.000	.280
SVM -UniBiTrigram	√				.721	.967	.091	.640
SVM -UniBiTrigram	√			√	.610	.938	.156	.621
SVM -AvgWordVec	√				.631	.952	.114	.579
SVM -AvgWordVec	√			√	.526	.100	.165	.336
SVM -AvgWordVec (transformed)	√	√	√		.571	.920	.229	.637
SVM -AvgWordVec (transformed)	√	√	√	√	.597	.963	.210	.642
CNN (Kim, 2014)	√				.738	.967	.171	.637
CNN (Kim, 2014)	√			√	.726	.964	.222	.648
RCNN (Lai et al., 2015)	√				.669	.951	.079	.606
RCNN (Lai et al., 2015)	√			√	.628	.944	.096	.605
UTCNN without user	√		√	√	**.748**	**.973**	.000	.580
UTCNN without topic	√	√		√	.643	.944	.476	.706
UTCNN without comment	√	√	√		.632	.940	.480	.707
UTCNN shared user embedding	√	√	√	√	.625	.969	.531	.732
UTCNN (full)	√	√	√	√	.698	.957	**.571**	**.755***

Table 3: Performance of post stance classification on the FBFans dataset.

*UTCNN (full) results are statistically significant (p-value < 0.005) with respect to all other methods except for UTCNN shared user embedding.

where P^{SNU} and R^{SNU} are the average precision and recall of the three class. We adopted the macro-average f-score as the evaluation metric for the overall performance because (1) the experimental dataset is severely imbalanced, which is common for contentious issues; and (2) for stance classification, content in minor-class posts is usually more important for further applications. For the CreateDebate dataset, accuracy was adopted as the evaluation metric to compare the results with related work (Hasan and Ng, 2013a; Hasan and Ng, 2013b; Sridhar et al., 2015).

4.3 Baselines

We pit our model against the following baselines: 1) SVM with unigram, bigram, and trigram features, which is a standard yet rather strong classifier for text features; 2) SVM with average word embedding, where a document is represented as a continuous representation by averaging the embeddings of the composite words; 3) SVM with average transformed word embeddings (the $x'w$ in equation 1), where a document is represented as a continuous representation by averaging the transformed embeddings of the composite words; 4) two mature deep learning models on text classification, CNN (Kim, 2014) and Recurrent Convolutional Neural Networks (RCNN) (Lai et al., 2015), where the hyperparameters are based on their work; 5) the above SVM and deep learning models with comment information; 6) UTCNN without user information, representing a pure-text CNN model where we use the same user matrix and user embeddings U_k and u_k for each user; 7) UTCNN without the LDA model, representing how UTCNN works with a single-topic dataset; 8) UTCNN without comments, in which the model predicts the stance label given only user and topic information. All these models were trained on the training set, and parameters as well as the SVM kernel selections (linear or RBF) were fine-tuned on the development set. Also, we adopt oversampling on SVMs, CNN and RCNN because the FBFans dataset is highly imbalanced.

4.4 Results on FBFans Dataset

In Table 3 we show the results of UTCNN and the baselines on the FBFans dataset. Here Majority yields good performance on *Neu* since FBFans is highly biased to the neutral class. The SVM models

perform well on *Sup* and *Neu* but perform poorly for *Uns*, showing that content information in itself is insufficient to predict stance labels, especially for the minor class. With the transformed word embedding feature, SVM can achieve comparable performance as SVM with *n*-gram feature. However, the much fewer feature dimension of the transformed word embedding makes SVM with word embeddings a more efficient choice for modeling the large scale social media dataset. For the CNN and RCNN models, they perform slightly better than most of the SVM models but still, the content information is insufficient to achieve a good performance on the *Uns* posts. As to adding comment information to these models, since the commenters do not always hold the same stance as the author, simply adding comments and post contents together merely adds noise to the model.

Among all UTCNN variations, we find that user information is most important, followed by topic and comment information. UTCNN without user information shows results similar to SVMs — it does well for *Sup* and *Neu* but detects no *Uns*. Its best f-scores on both *Sup* and *Neu* among all methods show that with enough training data, content-based models can perform well; at the same time, the lack of user information results in too few clues for minor-class posts to either predict their stance directly or link them to other users and posts for improved performance. The 17.5% improvement when adding user information suggests that user information is especially useful when the dataset is highly imbalanced. All models that consider user information predict the minority class successfully. UCTNN without topic information works well but achieves lower performance than the full UTCNN model. The 4.9% performance gain brought by LDA shows that although it is satisfactory for single topic datasets, adding that latent topics still benefits performance: even when we are discussing the same topic, we use different arguments and supporting evidence. Lastly, we get 4.8% improvement when adding comment information and it achieves comparable performance to UTCNN without topic information, which shows that comments also benefit performance. For platforms where user IDs are pixelated or otherwise hidden, adding comments to a text model still improves performance. In its integration of user, content, and comment information, the full UTCNN produces the highest f-scores on all *Sup*, *Neu*, and *Uns* stances among models that predict the *Uns* class, and the highest macro-average f-score overall. This shows its ability to balance a biased dataset and supports our claim that UTCNN successfully bridges content and user, topic, and comment information for stance classification on social media text. Another merit of UTCNN is that it does not require a balanced training data. This is supported by its outperforming other models though no oversampling technique is applied to the UTCNN related experiments as shown in this paper. Thus we can conclude that the user information provides strong clues and it is still rich even in the minority class.

We also investigate the semantic difference when a user acts as an author/liker or a commenter. We evaluated a variation in which all embeddings from the same user were forced to be identical (this is the UTCNN shared user embedding setting in Table 3). This setting yielded only a 2.5% improvement over the model without comments, which is not statistically significant. However, when separating authors/likers and commenters embeddings (i.e., the UTCNN full model), we achieved much greater improvements (4.8%). We attribute this result to the tendency of users to use different wording for different roles (for instance author vs commenter). This is observed when the user, acting as an author, attempts to support her argument against nuclear power by using improvements in solar power; when acting as a commenter, though, she interacts with post contents by criticizing past politicians who supported nuclear power or by arguing that the proposed evacuation plan in case of a nuclear accident is ridiculous. Based on this finding, in the final UTCNN setting we train two user matrix embeddings for one user: one for the author/liker role and the other for the commenter role.

4.5 Results on CreateDebate Dataset

Table 4 shows the results of UTCNN, baselines as we implemented on the FBFans datset and related work on the CreateDebate dataset. We do not adopt oversampling on these models because the CreateDebate dataset is almost balanced. In previous work, integer linear programming (ILP) or linear-chain conditional random fields (CRFs) were proposed to integrate text features, author, ideology, and user-interaction constraints, where text features are unigram, bigram, and POS-dependencies; the author

Method	Features		Topics				AVG
	Text	User	ABO	GAY	OBA	MAR	
Majority			.549	.634	.539	.695	.604
SVM -UniBiTrigram	√		.592	.569	.565	.673	.600
SVM -AvgWordVec	√		.559	.637	.548	.708	.613
SVM -AvgWordVec (transformed)	√	√	.859	.830	.800	.741	.808
CNN (Kim, 2014)	√		.553	.636	.557	.709	.614
RCNN (Lai et al., 2015)	√		.553	.637	.534	.709	.608
ILP (Hasan and Ng, 2013a)	√		.614	.626	.581	.669	.623
ILP (Hasan and Ng, 2013a)	√	√	.749	.709	.727	.754	.735
CRF (Hasan and Ng, 2013b)	√	√	.747	.699	.711	.754	.728
PSL (Sridhar et al., 2015)	√	√	.668	.727	.635	.690	.680
UTCNN without topic	√	√	.824	**.851**	.743	**.814**	.808
UTCNN without user	√		.617	.627	.599	.685	.632
UTCNN (full)	√	√	**.878**	.850	**.857**	.782	**.842***

Table 4: Accuracies of post stance classification on CreateDebate dataset.
*UTCNN results were statistically significant (p-value < 0.001) with respect to other UTCNN settings.

constraint tends to require that posts from the same author for the same topic hold the same stance; the ideology constraint aims to capture inferences between topics for the same author; the user-interaction constraint models relationships among posts via user interactions such as replies (Hasan and Ng, 2013a; Hasan and Ng, 2013b).

The SVM with n-gram or average word embedding feature performs just similar to the majority. However, with the transformed word embedding, it achieves superior results. It shows that the learned user and topic embeddings really capture the user and topic semantics. This finding is not so obvious in the FBFans dataset and it might be due to the unfavorable data skewness for SVM. As for CNN and RCNN, they perform slightly better than most SVMs as we found in Table 3 for FBFans.

Compared to the ILP (Hasan and Ng, 2013a) and CRF (Hasan and Ng, 2013b) methods, the UTCNN user embeddings encode author and user-interaction constraints, where the ideology constraint is modeled by the topic embeddings and text features are modeled by the CNN. The significant improvement achieved by UTCNN suggests the latent representations are more effective than overt model constraints.

The PSL model (Sridhar et al., 2015) jointly labels both author and post stance using probabilistic soft logic (PSL) (Bach et al., 2015) by considering text features and reply links between authors and posts as in Hasan and Ng's work. Table 4 reports the result of their best AD setting, which represents the full joint stance/disagreement collective model on posts and is hence more relevant to UTCNN. In contrast to their model, the UTCNN user embeddings represent relationships between authors, but UTCNN models do not utilize link information between posts. Though the PSL model has the advantage of being able to jointly label the stances of authors and posts, its performance on posts is lower than the that for the ILP or CRF models. UTCNN significantly outperforms these models on posts and has the potential to predict user stances through the generated user embeddings.

For the CreateDebate dataset, we also evaluated performance when not using topic embeddings or user embeddings; as replies in this dataset are viewed as posts, the setting without comment embeddings is not available. Table 4 shows the same findings as Table 3: the 21% improvement in accuracy demonstrates that user information is the most vital. This finding also supports the results in the related work: user constraints are useful and can yield 11.2% improvement in accuracy (Hasan and Ng, 2013a). Further considering topic information yields 3.4% improvement, suggesting that knowing the subject of debates provides useful information. In sum, Table 3 together with Table 4 show that UTCNN achieves promising performance regardless of topic, language, data distribution, and platform.

5 Conclusion

We have proposed UTCNN, a neural network model that incorporates user, topic, content and comment information for stance classification on social media texts. UTCNN learns user embeddings for all users with minimum active degree, i.e., one post or one like. Topic information obtained from the topic model or the pre-defined labels further improves the UTCNN model. In addition, comment information provides additional clues for stance classification. We have shown that UTCNN achieves promising and balanced results. In the future we plan to explore the effectiveness of the UTCNN user embeddings for author stance classification.

Acknowledgements

Research of this paper was partially supported by Ministry of Science and Technology, Taiwan, under the contract MOST 104-2221-E-001-024-MY2.

References

Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2015. Hinge-loss markov random fields and probabilistic soft logic. *arXiv preprint arXiv:1505.04406*.

Mohit Bansal, Claire Cardie, and Lillian Lee. 2008. The power of negative thinking: Exploiting label disagreement in the min-cut classification framework. In *COLING (Posters)*, pages 15–18.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3:993–1022.

Clinton Burfoot, Steven Bird, and Timothy Baldwin. 2011. Collective classification of congressional floor-debate transcripts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1506–1515. Association for Computational Linguistics.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014a. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 49–54. Association for Computational Linguistics.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2014b. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence*. AAAI.

Kazi Saidul Hasan and Vincent Ng. 2013a. Extra-linguistic constraints on stance recognition in ideological debates. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 816–821. Association for Computational Linguistics.

Kazi Saidul Hasan and Vincent Ng. 2013b. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1348–1356. Association for Computational Linguistics.

Kazi Saidul Hasan and Vincent Ng. 2014. Why are you taking this stance? identifying and classifying reasons in ideological debates. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 751–762.

Minlie Huang, Yujie Cao, and Chao Dong. 2016. Modeling rich contexts for sentiment classification with lstm. *arXiv preprint arXiv:1605.01478*.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751. Association for Computational Linguistics.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence*, pages 2267–2273. AAAI.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543. Association for Computational Linguistics.

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Context-sensitive twitter sentiment classification using neural network. In *Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence*. AAAI.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, volume 1631, page 1642. Association for Computational Linguistics.

Dhanya Sridhar, James Foulds, Bert Huang, Lise Getoor, and Marilyn Walker. 2015. Joint models of disagreement and stance in online debate. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Duyu Tang, Bing Qin, and Ting Liu. 2015a. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1014–1023. Association for Computational Linguistics.

Duyu Tang, Bing Qin, Ting Liu, and Yuekui Yang. 2015b. User modeling with neural network for review rating prediction. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 1340–1346.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 327–335. Association for Computational Linguistics.

Marilyn A Walker, Pranav Anand, Robert Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 592–596. Association for Computational Linguistics.

The Role of Intrinsic Motivation in Artificial Language Emergence: a Case Study on Colour

Miquel Cornudella[1,2] **Thierry Poibeau**[2] **Remi van Trijp**[1]

[1]**Sony Computer Science Laboratory Paris**
6 rue Amyot, 75005
Paris, France
{cornudella,remi}@csl.sony.fr

[2]**Laboratoire LATTICE-CNRS**
ENS & U. Paris 3 & CNRS
PSL and USPC
1 rue Maurice Arnoux, 92120
Montrouge, France
thierry.poibeau@ens.fr

Abstract

Human languages have multiple strategies that allow us to discriminate objects in a vast variety of contexts. Colours have been extensively studied from this point of view. In particular, previous research in artificial language evolution has shown how artificial languages may emerge based on specific strategies to distinguish colours. Still, it has not been shown how several strategies of diverse complexity can be autonomously managed by artificial agents . We propose an intrinsic motivation system that allows agents in a population to create a shared artificial language and progressively increase its expressive power. Our results show that with such a system agents successfully regulate their language development, which indicates a relation between population size and consistency in the emergent communicative systems.

1 Introduction

Over the past two decades, language evolution studies have attracted the attention of researchers working on domains such as biology, anthropology, artificial life or linguistics. This multitude of perspectives provides a rich variety of techniques on how to address this issue, including including agent-based modelling, which consists in studying the emergence and evolution of *artificial languages*, i.e. human-like communicative systems, in a population of artificial agents through recurrent peer-to-peer interactions (Smith et al., 2003; Steels, 2012). Results using this approach have shed light on the emergence of spatial relations (Spranger, 2013), case systems (van Trijp, 2013), colour categories (Bleys, 2010) or syntax (Garcia-Casademont and Steels, 2016).

In most of these experiments the control of the complexity relies on the experimenter, who carefully selects the stages of the experiment, constraining the language development. Insights from different models dealing with complexity come from research in AI and robotics, where a number of studies where a number of studies tried to specify how agents regulate the complexity of their actions in an autonomous way. Several models have been proposed, including error reduction (Andry et al., 2001), prediction (Marshall et al., 2004), interest (Merrick and Maher, 2009) or curiosity (Oudeyer et al., 2007).

These systems have been deeply inspired by psychological studies on the role of *motivation* (see Graham (1996) for an overview). According to Ryan and Deci (2000), motivation can be defined as "to be *moved* to do something". Psychologists further distinguish two types of motivation, depending on the reasons to perform an action: *extrinsic*, when the interest relies on the outcome of the action, or *intrinsic*, when the action itself results inherently enjoyable. Both types of motivation can incite to take part in the same activity. For example, a tennis player can work on improving her slice shots because she wants to win her next tournament or because she enjoys improving her technique.

This paper presents a simulation experiment to study how a population provided with the *Autotelic principle* (2004), a computational motivation system inspired on the *Flow theory* (1990), is able to self-organize and extend the expressive power of a shared communication system for the continuous domain of colour. Agents play a language game[1] and they use this motivation system to decide on the complexity

[1]Interested readers in the methodology used in this experiment to study the emergence of artificial language are referred to Steels (2012).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1646–1656, Osaka, Japan, December 11-17 2016.

of both the context of the interaction and the utterances they formulate and comprehend.

In the next section of the paper, we describe Flow Theory and an operational computational version, the Autotelic Principle, that allow agents to autonomously regulate their development. Section 3 briefly reviews artificial language evolution research on the domain of colour, the case study in which the motivational system is tested. Section 4 presents the experiment design in detail: how agents interact, the different communicative tasks, how the context of interactions is chosen and the different operational mechanisms agents use to emerge and align a shared communicative system. Finally, sections 5 and 6 presents the experimental results and conclusions.

2 Flow theory and architecture

Csíkszentmihályi wanted to understand what moves people to be absorbed in complex activities that do not provide an external reward, such as rock climbing, painting or sculpting. He found that the reason was that participants found these activities inherently enjoyable. He called these activities *autotelic*, as the motivational driving force (*telos*) comes from the individual itself (*auto*).

Based on these observations he developed the *Flow theory* (1990). Autotelic activities can be explained based on the relation of two dimensions (Figure 1a): *challenge*, a certain task to be done, and *skill*, the abilities that a person has to tackle that task. This relation accounts for the range of different mental states that people experience when they are involved in an autotelic activity: *boredom*, when the skills are too high for the current challenge, *anxiety*, when the challenge is too difficult given the current skills, and *flow*, when there is a balance between both. He identified the latest as the optimal state of experience. This state provides the best scenario for further enlarging their skills. As a consequence, the state of flow is in in continuous motion, as skills evolve over time. Participants seek to stay in flow state, therefore becoming self-motivated.

Inspired by the work of Csíkszentmihályi, Steels (2004) proposed the *Autotelic principle*, an operational version of the Flow theory to provide artificial agents with a system to self-regulate their development. As in the Flow theory, the balance between challenge and skills acts as the motivational driving force in agents. Agents use this relation to identify their internal state (*boredom*, *anxiety* or *flow*) and consequently react to it increasing or decreasing their challenges.

2.1 Challenge management

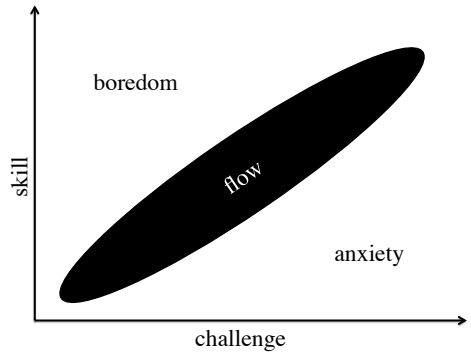
The principle establishes two different phases, *operational* and *shake-up*. The first one corresponds to the state of flow: agents explore a particular challenge and try to develop the abilities required to cope with it. The latest is reached when agents are either in a state of anxiety or boredom. It acts as a trigger to adjust the challenge to be addressed in order to look for a more balanced challenge-skills relation. In case of boredom a more demanding challenge should be attempted. In contrast, a more accessible challenge should be tackled in an anxiety state.

Challenges are characterised as a set of parameters: given a multi-dimensional parameter space P, a challenge p_i is defined as a vector $< p_{i,1}, p_{i,2}, ..., p_{i,n} >$, where $p_{i,j}$ corresponds to the value of the parameter j in the challenge i. Agents are able to generate more complex or manageable challenges by changing the specific configuration of a challenge. The space of possible challenges depends on the number of parameters of the set and the different values each parameter can have.

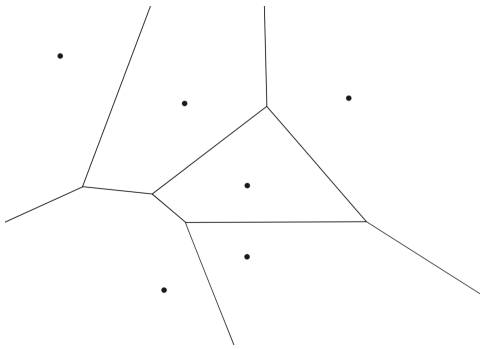
2.2 Skill evaluation

Agents indirectly measure their skills based on their performance. Each challenge is monitored with a value in the range [0,1] where 1 represents optimal performance for that task. This value is used to compute the *confidence* an agent has in its skill level to accomplish a particular challenge. A confidence value of 1 is interpreted as boredom as it indicates that the agent has acquired the skills to handle the current task. On the other hand, a steady value of 0 is interpreted as anxiety as it shows that the agent is not succeeding in expanding its abilities to cope with the actual challenge.

After each interaction, the confidence on the current challenge is updated taking into account the individual competences of the agent and the outcome of the interaction. When the interaction is a



(a) Flow theory



(b) Visual representation of the partition of the colour space into basic colour categories

Figure 1: According to Csíkszentmihályi (Figure 1a), individuals enter a state of *flow* when they correctly balance their skills against the selected challenges when performing particular activities. A mismatch of this balance may lead to anxiety (i.e. the challenge is too big) or boredom (i.e. the challenge is too easy). Figure 1b illustrates the division of the colour space into basic categories. Image extracted from Bleys (2010).

success, the confidence value is updated as follows: $conf_i = conf_{i-1} + \delta increase$, where $conf_i$ and $conf_{i-1}$ are the current and previous confidence values and $\delta increase$ is set to 0.005. On the contrary, when the outcome of an interaction is a failure the confidence value is updated in this way: $conf_i = conf_{i-1} - \delta decrease + ind_{comp}$, where $\delta decrease$ is set to 0.02 and ind_{comp} is the value in the range [0,0.015] that corresponds to the evaluation of the individual competences of the agent.

This motivation system has been used before in experiments of language emergence in discrete domains (Steels and Wellens, 2007; Cornudella et al., 2015). The work presented here differs from previous experiments in that it tests the autotelic principle in a continuous domain. In this experiment agents need to self-organize a communicative system but also agree on the meanings associated with their lexicons: colours are no longer discrete values but rather points in a three dimensional feature space and membership prototypes are values in the range [0,1].

3 A case study on colour

Research on the domain of colour has been of great interest to a lot of researchers, due to the differences observed in how colours are described in human languages (Berlin and Kay, 1969). It is commonly accepted that a *colour space*, the space of colours that can be perceived, is organised in different *colour categories*, subsets of this space (Figure 1b). *Colour prototypes* are the best representation of a particular colour category in a colour space (Rosch, 1973). Formally, a colour space is composed of a set of colour prototypes $\{c_1, c_2, ..., c_n\}$. Given the colour prototype c_k, its associated cell R_k, which determines the associated colour category, contains every point whose distance to c_k is shorter or equal to the distance to any other prototype c_i.

Although the research in this domain is extensive, most studies have focused on the use of single terms to describe colours. Experiments by Simpson and Tarrant (1991) and Lin et al. (2001) showed that only 15% of colour samples were described using a single colour term when human subjects were asked to describe colour samples without any restriction. These results provide evidence to the fact that usually people prefer to express more information about the colours they are describing instead of only employing single terms.

Colour has been of particular interest to researchers working on artificial language evolution. The majority of models have focused on the emergence of single colour terms (Steels et al., 2005; Belpaeme and Bleys, 2007; Baronchelli et al., 2010; Baronchelli et al., 2015), but some attempts to model more complex descriptions also exist. In this respect, the most advanced contribution is the work of Bleys (2009; 2009; 2012). In his doctoral thesis (2010) different *language strategies*, a particular method to express one area of meaning, are explained and studied. These language strategies are then tested on

artificial language evolution experiments, showing how these models can emerge and be learned by a population of artificial agents.

4 Experiment

In this section the design of the experiment is explained: the particular language game agents play, the different communicative challenges of the experiment, how the contexts of interactions are determined and the operational mechanisms agents use to develop and align their language. The experiment is implemented in Babel2[2], a multi-agent experiment framework (Loetzsch et al., 2008).

4.1 Language Game

The experiment consists in recurrent communicative interactions in a population of artificial agents equipped with the autotelic principle situated in a particular context. In every interaction a randomly selected pair of agents is picked from the population. One of them assumes the role of *speaker* and the other the role of *hearer*. The goal of the interacting agents is to communicate about one colour sample from the context.

The specific language game that agents play is called multi-word guessing game. The speaker selects the context of the interaction, based on the challenge it is currently addressing, and randomly picks a colour sample as topic[3]. When the speaker is able to discriminatingly conceptualise the topic into a meaning predicate it uses its language component to formulate an utterance which is transmitted as text to the hearer. The hearer tries to comprehend the utterance and constructs hypotheses about the topic. If the hearer has only one hypothesis, it points to the interpreted topic.

If the hypothesis corresponds to the topic, the speaker gives positive feedback and the interaction ends. On the other hand, if the hypothesis does not correspond to it, the speaker gives negative feedback to the hearer and points to the intended topic. When the hearer has no or multiple hypotheses it signs to the speaker that it could not identify the topic. The speaker then gives feedback by pointing to the intended topic. The interaction is a success only when the hearer has one hypothesis about the topic that corresponds with the topic selected by the speaker. Otherwise, the result of the interaction is a failure.

4.2 Challenges

Agents can use different language strategies to communicate about colour samples. These strategies are identified as the three communicative challenges of the experiment and were previously analysed in Bleys (2010): *basic colour*, *graded membership* and *graded category combination*[4]. A parameter *level* is associated to each challenge, according to its complexity. Agents use this parameter to move between challenges, depending on their internal state. Agents are able to perform three operations on the colour space: add a colour prototype, compute the distance between a colour sample and its closer colour prototype (Figure 2a) and transform the colour space towards a colour prototype (Figure 2b).

In the basic colour strategy a single term is used to describe a colour sample. Agents use this term to refer to the closest colour prototype. An example for English would be to use a term as “green” to describe a colour sample. In the experiment, agents are initialised with an empty vocabulary and colour space. This means that they need to converge both on a classification of the colour space into colour prototypes and on the terms associated to each colour prototype.

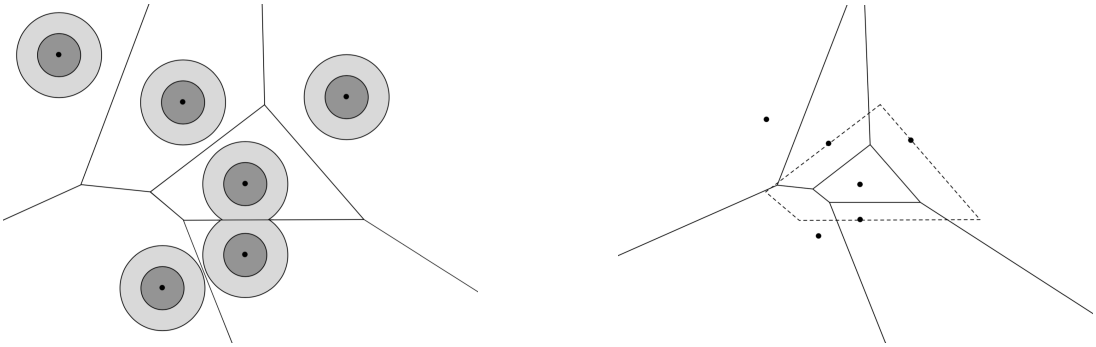
The graded membership strategy characterises a colour sample by expressing both the closest colour prototype and the distance between the colour sample to it. This strategy is observed, for instance, in English, where it is possible to describe a colour sample by combining a basic colour term with adverbs such as “very” or the postfix “-ish”, as in “very blue” or “greenish”. When addressing this challenge, agents also need to agree on both the membership prototypes and the terms associated with them.

Lastly, the graded category combination strategy describes a colour sample by referring to two colour and one membership prototypes. Agents first identify the closest colour prototype to the colour sample

[2]Babel2 is available as open-source software at `www.emergent-languages.org`.

[3]Interested readers in the impact of active selection of the topic are pointed to Schueller and Oudeyer (2015).

[4]The language strategies used in this paper have been replicated from Bleys (2010), who granted us access to the original implementation.



(a) Graded membership operation

(b) Category combination operation

Figure 2: Visual representation of available operations. In a graded membership operation (Figure 2a) the distance between the colour sample and the closer colour prototype is computed. In a category combination operation (Figure 2b) the colour space is transformed towards the main colour prototype of the colour sample in order to perform a second classification. Images extracted from Bleys (2010).

and transform the colour space towards that prototype. Agents classify again the colour sample on the transformed colour space, obtaining a second colour prototype[5]. Finally, they express how close the colour sample is to the identified colour prototype in the transformed colour space using a graded membership term. "Very dark green" or "blueish purple" are examples of this strategy in English.

The complexity of a communicative task is determined by its number of *cognitive operations*: algorithms that encode a particular cognitive function used in conceptualisation and interpretation. This number differs among the different challenges and is used to determine its level. Moreover, more complex colour descriptions can reuse skills developed on earlier stages.

4.3 Context

The world consist of 268 different colour samples in the CIE 1967 $L^*A^*B^*$ colour space. Colour samples are represented in three dimensions: the L^* dimension represents lightness, the A^* dimension roughly redness-greenness and the B^* approximately yellowness-blueness. The difference between two colour samples is determined by their Euclidean distance. The world contains the focal colours and the consensus samples[6] for English and colour samples created when combining two focal colours in different percentages: 25%, 45%, 55% and 75%, respectively.

(a) Example of a context for the basic colour challenge

(b) Example of a context for the graded membership challenge

(c) Example of a context for the graded category combination challenge

Figure 3: Example of the different contexts speakers can create, depending on their current challenge.

In each interaction the speaker selects the context, which is a subset of the colour samples present in the world. It chooses both the size of the context and the different colour samples that are part of it. The choice depends on the current challenge of the speaker. In the basic colour challenge the context is created by randomly picking three focal colours of English. In the graded membership challenge the speaker chooses five random samples from the consensus samples for English. Finally, in the graded category combination challenge the speaker picks six colour samples that correspond to the combination of two focal colours for English. Figure 3 provides an example of each context.

[5]In the experiment agents can classify the colour sample to the same colour prototype twice, before and after transforming the colour space. For instance, they can create utterances as "blueish blue" if the colour sample is very close to a certain colour prototype.

[6]Colour samples that were consistently named in English by all participants. See Sturges & Whitfield (1995).

4.4 Operational mechanisms

Agents create and learn *constructions*, which can be seen as form-meaning pairs. Constructions are stored in the *construction inventory* of the agent, which defines its vocabulary and grammar. Agents make use of their construction inventory to *formulate*, verbalise a conceptualised meaning, and *comprehend*, extract the meaning representation of an input utterance. Agents start the experiment with an empty vocabulary and enlarge it using different *diagnostics*, used to identify problems during an interaction, and *repairs*, processes to solve diagnosed problems.

Invention: the speaker cannot find a discriminating colour or membership prototype in formulation, caused by the lack of a relevant colour or membership prototype.

- *Diagnostic*: the speaker cannot come up with a discriminative conceptualization of the topic.
- *Repair for lack of relevant colour prototype*: the speaker creates a colour prototype C and sets the colour sample of the topic as its colour prototype. Additionally, the speaker invents a new term t for the colour prototype and creates a new construction relating C with t.
- *Repair for lack of relevant membership prototype*: the speaker creates a new membership prototype M and sets its value to the distance between the colour sample and the prototype of its closest colour category. Additionally, the speaker invents a new term t for the membership prototype and creates a new lexical construction relating M with t.

Adoption: the hearer cannot identify the topic due to an unknown word t, which can refer to either a colour or a membership prototype.

- *Diagnostic*: the hearer encounters an unknown word in the input utterance.
- *Repair for unknown word that refers to a colour prototype*: the hearer uses the feedback from the speaker to create a colour prototype C with the colour sample of the topic as its colour prototype. Additionally, the hearer creates a new construction relating C with t.
- *Repair for unknown word that refers to a membership prototype*: the hearer uses the feedback from the speaker to create a new membership prototype M and sets its value to the distance between the colour sample and the prototype of its closest colour category. Additionally, the hearer creates a new lexical construction relating M with t.

Moreover, agents can also create and learn *grammatical constructions*. These constructions allow agents to express meaning predicates not captured by lexical constructions and restrict the ambiguity of multi-word sentences by imposing form constraints.

4.5 Alignment

In the previous subsection we have introduced *adoption*, which allows hearers to learn new word-meaning associations for both colour and membership prototypes. When adopting an unknown word agents have to decide between adding the observed colour or membership prototype as a new prototype to their inventory or associate the unknown word to an existing one. The decision is based on how close the observed prototype is from the closest prototype in the inventory. A new prototype will be added when the Euclidean distance between both prototypes is bigger than 0.05. When the hearer associates the word to an already existing prototype it introduces competition in its construction inventory, as at least two constructions refer to the same prototype. For instance, the terms “blue” and “azul” would be competitors if they are associated to the same colour prototype.

Agents are provided with a mechanism called *alignment* to manage the competition between constructions in their construction inventory. Each construction has a score with a value between 0.0 and 1.0, and is initialized at 0.5. Scores are used by agents to decide which constructions apply to express one meaning, selecting the one with the highest score. After each interaction the scores of the constructions of the interacting agents are updated using an alignment method called *lateral inhibition* (De Vylder and

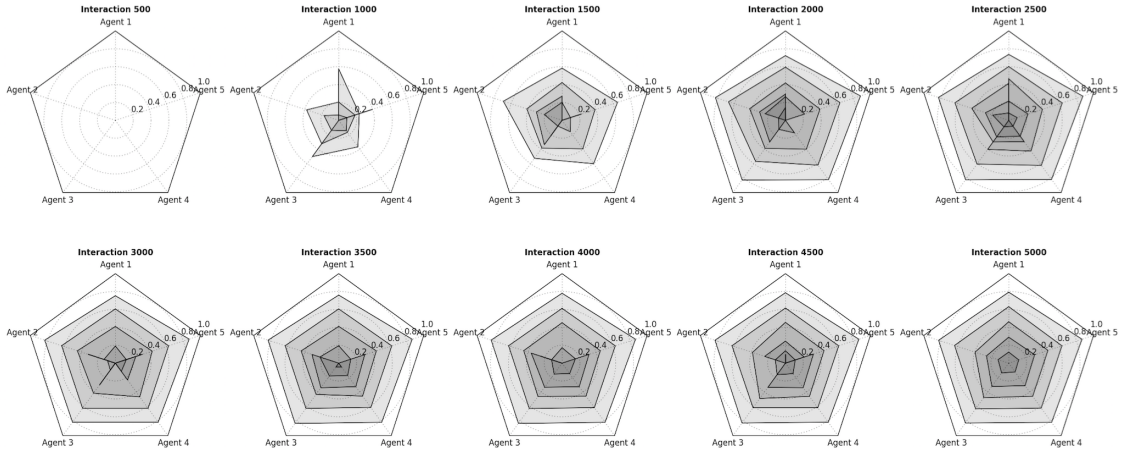


Figure 4: Example of the alignment of membership prototypes for a population of 5 agents. Initially each agent has different prototype values. After each successful interaction the involved membership values of the interacting agents are adjusted. At the end of the simulation the population converges to similar prototypes.

Tuyls, 2006). When a construction has reached a score of 0.0 is removed from the construction inventory of the agent.

Alignment takes into account the outcome of the interaction (i.e. communicative success or failure) to update the scores of the constructions. In a successful interaction both speaker and hearer increase the constructions used by a score $\delta_{increase}$ and punish their competing constructions by a score of $\delta_{decrease}$. If the result of the interaction is a failure, the speaker punishes the constructions used by a score of $\delta_{decrease}$. In the experiment both $\delta_{increase}$ and $\delta_{decrease}$ are set to 0.1.

Agents also align their prototypes. After a successful interaction that involved a membership prototype *M*, both speaker and hearer update the value m associated to that prototype as follows: $m_i = m_{i-1} - \delta rate(m_{i-1} - act_i)$, where m_i and m_{i-1} are the current and previous values of M and act_i the activation of the topic. In the experiment $\delta rate$ is set to 0.05. Figure 4 illustrates the alignment of membership prototypes in a population of five agents.

5 Experimental results

All experimental results have been tested on ten runs, to ensure the consistency of the results. In each trial agents start with an empty construction inventory and colour and membership inventories. The following measures are reported:

- *Communicative success* measures the average performance of the population in the communicative task. When the communication is successful a value of 1.0 is recorded, 0.0 otherwise.
- *Alignment success* measures the average cohesion of the construction inventory on the population. A value of 1.0 is recorded when there was communicative success and both agents would use the same constructions to refer to the topic of that interaction, 0.0 otherwise.
- *Lexical stability* measures the average scores of lexical constructions of the population. A value of 1.0 means that all lexical constructions on each agent have the maximum score.
- *Confidence in challenge* measures the average confidence that the population has for a certain challenge level. It has a value between 0.0 and 1.0.

The resulting dynamics of the experiment with a population of ten agents are shown in Figure 5. Agents start addressing the first challenge, on which the population has to create and coordinate their

construction inventory for basic colour terms. Agents gain confidence for this challenge fast, as both the communicative success and the confidence value for the first challenge rapidly increase. An abrupt drop occurs around interaction 2000, when agents start to reach maximum confidence. As it corresponds to the internal state of boredom, agents enter in the shake-up phase and move to the second challenge. In the course of the second challenge agents are exposed to a bigger diversity of contexts, which makes the alignment of membership prototypes and its associated lexical constructions more difficult. This is also reflected in the evolution of lexical stability, as the average score of lexical constructions drops despite the fact that agents are converging to an optimal lexicon for basic colour terms.

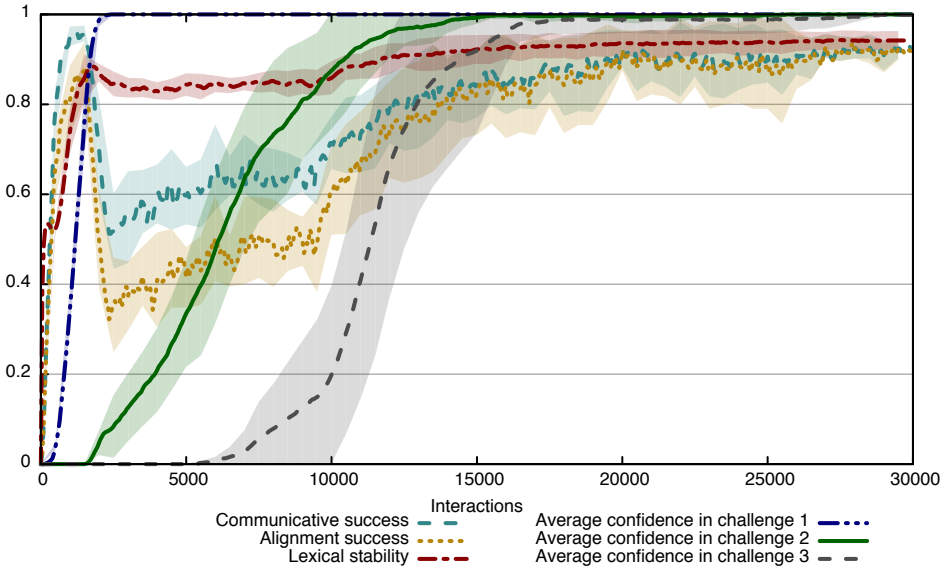


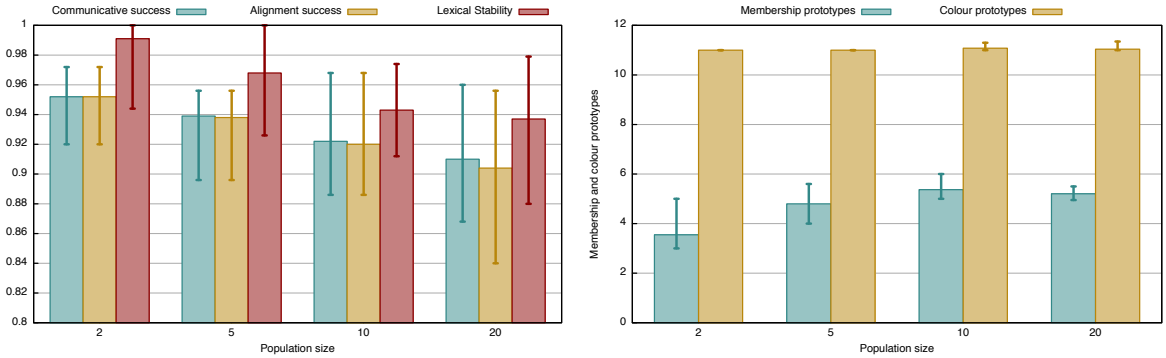
Figure 5: Resulting dynamics of the experiment for a population of 10 agents averaged over 10 runs of 30000 interactions. By the end of the simulation all agents in the population reach a steady communicative success value above 90% and maximum confidence for the three challenges. Error bars represent the maximum and minimum across the different experimental runs.

An overlap between the second and third challenge starts approximately at interaction 5000 where a fraction of the population has already reached maximum confidence for the second challenge. At this point agents identify their internal state as boredom and are motivated to attempt the third challenge. Communication success progressively improves as population succeeds in aligning their construction inventory and membership prototypes. As a consequence of this alignment, alignment success also increases and reaches the same value as communicative success. By interaction 30000 all agents in the population have reached maximum confidence for the three challenges and a steady communicative success value above 90%.

Bleys (2010) showed that agents using these strategies cannot come up with a discriminative conceptualisation in certain situations, which explains why the population does not reach a 100% communicative success even when all agents have reached the highest confidence score for all challenges. However, lexical stability settles to a value around 95%, which means that not all lexical constructions in the population have a score of 1.0. This is caused by different membership categories no longer used but still in the lexicon of some agents. Therefore, population has not fully converged to a minimal lexicon, although they manage to communicate successfully in most contexts.

We have studied the relation between lexical stability and communicative success by testing the same configuration on different populations. Figure 6a presents the resulting communicative success, alignment and lexical stability for a population of two, five, ten and twenty agents (3000, 10000, 50000 and 150000 interactions, respectively). Results show a slight reduction of communicative success as population size increases. More importantly, a little discrepancy between communicative success and alignment is observed in bigger populations. This gap occurs because in some interactions agents prefer distinct

discriminative conceptualisations for certain topics. In other words, different prototypes are triggered as more accurate conceptualisations of the topic in a particular context and therefore agents select different terms to describe the same colour sample.



(a) Resulting communicative success, alignment and lexical stability for different population sizes averaged over 10 runs.

(b) Resulting membership and colour categories for different population sizes averaged over 10 runs.

Figure 6: Effect of population size. Figure 6a presents the resulting communicative success, alignment and lexical stability scores in a population of two, five, ten and twenty agents. The scale on the Y-axis is set to the range [0.8,1]. Figure 6b displays the average membership and colour prototypes for the same populations.

This effect can be explained by the fact that bigger populations converge to systems with more membership prototypes (Figure 6b). An increased number of membership prototypes requires more time to align: this helps prototypes which are not spread over the population to stay longer in individual inventories as they are less used. The decrease in communicative success is therefore explained by (a) a lower alignment of agents' construction inventories and membership prototypes and (b) longer presence of non spread membership prototypes among the population that are used in conceptualisation. These results suggest that smaller populations could be able to arise more consistent communicative systems for the domain of colour.

6 Conclusions

In this paper we have studied how a population of agents provided with a motivation system to regulate their complexity is able to develop an artificial language of increasing expressive power to refer to colours. Agents using this system develop to their construction inventory in three progressive stages: they first (a) converge on a language for colour prototypes and then extend its expressive power by (b) developing categories to express degrees of similarity between a colour sample and a colour prototype and (c) combining colour prototypes.

The results obtained show that a population of agents equipped with an architecture of flow successfully manages to progressively develop its communicative skills when trying to remain in a state of flow. Moreover, simulations with different population sizes show that bigger populations converge to systems with more membership prototypes on average.

Acknowledgements

This research was conducted at and funded by the Sony Computer Science Laboratory Paris. Miquel Cornudella is partially supported by a CIFRE grant (agreement no. 2013/0730). We would like to thank those who provided help and feedback, particularly Paul Van Eecke, and three anonymous referees for very constructive remarks after a careful reading of the manuscript. Special thanks go to Joris Bleys, who granted us access to the original implementation of the language strategies used in his experiments.

References

Pierre Andry, Philippe Gaussier, Sorin Moga, Jean-Paul Banquet, and Jacqueline Nadel. 2001. Learning and communication via imitation: An autonomous robot perspective. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 31(5):431–442.

Andrea Baronchelli, Tao Gong, Andrea Puglisi, and Vittorio Loreto. 2010. Modeling the emergence of universality in color naming patterns. *Proceedings of the National Academy of Sciences*, 107(6):2403–2407.

Andrea Baronchelli, Vittorio Loreto, and Andrea Puglisi. 2015. Individual biases, cultural evolution, and the statistical nature of language universals: The case of colour naming systems. *PLoS ONE*, 10(5):1–19, 05.

Tony Belpaeme and Joris Bleys. 2007. Language, perceptual categories and their interaction: Insights from computational modelling. In *Emergence of Communication and Language*, pages 339–353. Springer.

Brent Berlin and Paul Kay. 1969. *Basic color terms: Their universality and evolution*. University of California Press, Berkeley.

Joris Bleys and Luc Steels. 2009. Linguistic selection of language strategies. In *European Conference on Artificial Life*, pages 150–157. Springer.

Joris Bleys, Martin Loetzsch, Michael Spranger, and Luc Steels. 2009. The Grounded Color Naming Game. In *Proceedings of the 18th IEEE International Symposium on Robot and Human Interactive Communication (Ro-man 2009)*.

Joris Bleys. 2010. *Language Strategies for the Domain of Colour*. Ph.D. thesis, Vrije Universiteit Brussel.

Joris Bleys. 2012. Language strategies for color. In Luc Steels, editor, *Experiments in Cultural Language Evolution*, pages 61 – 85. John Benjamins.

Miquel Cornudella, Paul Van Eecke, and Remi van Trijp. 2015. How intrinsic motivation can speed up language emergence. In *Proceedings of the European Conference on Artificial Life 2015*, pages 571–578.

Mihaly Csíkszentmihályi. 1990. *Flow: The psychology of optimal experience*. Harper and Row, New York.

Bart De Vylder and Karl Tuyls. 2006. How to reach linguistic consensus: A proof of convergence for the naming game. *Journal of Theoretical Biology*, 242(4):818 – 831.

Emilia Garcia-Casademont and Luc Steels. 2016. Insight grammar learning. *Journal of Cognitive Science*, 17(1):27–62.

Sandra Graham. 1996. Theories and principles of motivation. *Handbook of educational psychology*, 4:63–84.

Helen Lin, M Ronnier Luo, Lindsay W MacDonald, and Arthur WS Tarrant. 2001. A cross-cultural colour-naming study. part i: Using an unconstrained method. *Color Research & Application*, 26(1):40–60.

Martin Loetzsch, Pieter Wellens, Joachim De Beule, Joris Bleys, and Remi van Trijp. 2008. The Babel2 manual. Technical Report AI-Memo 01-08, AI-Lab VUB, Brussels, Belgium.

James B Marshall, Douglas Blank, and Lisa Meeden. 2004. An emergent framework for self-motivation in developmental robotics. In *Proceedings of the 3rd international conference on development and learning (ICDL 2004), Salk Institute, San Diego*, volume 10.

Kathryn Merrick and Mary Lou Maher. 2009. Motivated learning from interesting events: adaptive, multitask learning agents for complex environments. *Adaptive Behavior*, 17(1):7–27.

Pierre-Yves Oudeyer, Frédéric Kaplan, and Verena Vanessa Hafner. 2007. Intrinsic motivation systems for autonomous mental development. *Evolutionary Computation, IEEE Transactions on*, 11(2):265–286.

Eleanor H Rosch. 1973. Natural categories. *Cognitive psychology*, 4(3):328–350.

Richard M Ryan and Edward L Deci. 2000. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25(1):54–67.

William Schueller and Pierre-Yves Oudeyer. 2015. Active learning strategies and active control of complexity growth in naming games. In *the 5th International Conference on Development and Learning and on Epigenetic Robotics*.

Jean Simpson and Arthur WS Tarrant. 1991. Sex-and age-related differences in colour vocabulary. *Language and speech*, 34(1):57–62.

Kenny Smith, Simon Kirby, and Henry Brighton. 2003. Iterated learning: A framework for the emergence of language. *Artificial Life*, 9(4):371–386.

Michael Spranger. 2013. Evolving grounded spatial language strategies. *KI-Künstliche Intelligenz*, 27(2):97–106.

Luc Steels and Pieter Wellens. 2007. Scaffolding language emergence using the autotelic principle. In *Artificial Life, 2007. ALIFE'07. IEEE Symposium on*, pages 325–332. IEEE.

Luc Steels, Tony Belpaeme, et al. 2005. Coordinating perceptually grounded categories through language: A case study for colour. *Behavioral and brain sciences*, 28(4):469–488.

Luc Steels. 2004. The autotelic principle. In I. Fumiya, R. Pfeifer, L. Steels, and K. Kunyoshi, editors, *Embodied Artificial Intelligence*, volume 3139 of *Lecture Notes in AI*, pages 231–242. Springer Verlag, Berlin.

Luc Steels. 2012. Self-organization and selection in cultural language evolution. In Luc Steels, editor, *Experiments in Cultural Language Evolution*, pages 1 – 37. John Benjamins, Amsterdam.

Julia Sturges and TW Allan Whitfield. 1995. Locating basic colours in the munsell space. *Color Research & Application*, 20(6):364–376.

Remi van Trijp. 2013. Linguistic assessment criteria for explaining language change: A case study on syncretism in german definite articles. *Language Dynamics and Change*, 3(1):105–132.

Predicting the Evocation Relation between Lexicalized Concepts

Yoshihiko Hayashi
Faculty of Science and Engineering, Waseda University
2-4-12 Ohkubo, Shinjuku, Tokyo 169-0072, Japan
yshk.hayashi@aoni.waseda.jp

Abstract

Evocation is a directed yet weighted semantic relationship between lexicalized concepts. Although evocation relations are considered potentially useful in several semantic NLP tasks, the prediction of the evocation relation between an arbitrary pair of concepts remains difficult, since evocation relationships cover a broader range of semantic relations rooted in human perception and experience. This paper presents a supervised learning approach to predict the strength (by regression) and to determine the directionality (by classification) of the evocation relation that might hold between a pair of lexicalized concepts. Empirical results that were obtained by investigating useful features are shown, indicating that a combination of the proposed features largely outperformed individual baselines, and also suggesting that *semantic relational vectors* computed from existing semantic vectors for lexicalized concepts were indeed effective for both the prediction of strength and the determination of directionality.

1 Introduction

Evocation, defined as the extent to which one concept (the *source* concept, s) brings to mind another (the *target* concept, t), is a directed yet weighted semantic relationship between semantic units (Boyd-Graber et al., 2006). As in the previous work (Boyd-Graber et al., 2006; Ma, 2013), the present work also considers evocation to be a semantic relationship between lexicalized concepts, rather than a relation between words. The weight of an evocation relation instance should measure the strength of the directed association from s to t, which we cannot directly observe nor compute from corpora.

Although evocation relations are potentially useful in several semantic NLP tasks, such as the measurement of textual similarity/relatedness and the lexical chaining in discourse, the prediction of the evocation relation between an arbitrary pair of concepts remains more difficult than measuring conventional similarities (synonymy, as well as hyponymy/hypernymy) or relatednesses (further including antonymy, meronymy/holonymy, as well as predicate-argument relations and maybe more), since evocation relationships cover a far broader range of semantic relationships (Cramer, 2008). Besides, as Ma (2013) argues, some types of evocation relations might be rooted in human perception and experience, implying that the acquisition of evocation relations solely from textual corpora is rather difficult, as they are the outcome of already accomplished activities of language production.

To the best of our knowledge, this paper is the first to present a supervised learning approach to predict the strength of an evocation as a regression task, and to determine the directionality as a classification task. We utilize Princeton WordNet (PWN) (Fellbaum, 1998) as the inventory of lexicalized concepts (synsets). Our empirical results show that combining a range of features is effective as intended, and that the *semantic relational vectors* (detailed in section 3.2.3) computed from existing semantic vectors for word senses and lexicalized concepts are indeed effective for both the prediction of strength and the determination of directionality. This definitely highlights the effectiveness of the semantic vectors derived for WordNet lexemes and synsets (Rothe and Schütze, 2015) (henceforth, Autoextend lexeme/synset semantic vector) that were utilized in the present work.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1657–1668, Osaka, Japan, December 11-17 2016.

The proposed method (section 3) utilizes several types of features: their effectiveness was examined by a series of experiments that employed the strength data provided by (Boyd-Graber et al., 2006) (PWN evocation data), and the directionality data made available by (Ma, 2013) (Ma's evocationNet data) (section 2). Although the empirical results (section 4) we obtained were rather promising, there remains considerable room for improvement. Thus, the paper concludes with possible future directions (section 6) after a brief review of some related research (section 5).

2 Evocation Relationship and the Resources

Evocation is the outcome of a kind of psychological phenomenon rooted in human perception and experience (Ma, 2013). This strongly implies that human-rated resources are required to uncover the underlying psycholinguistic mechanisms and to develop a computational mechanism to predict the evocation relationship between an arbitrary pair of lexicalized concepts.

To date, two resources have been publicized to facilitate research associated with the evocation relationship: one is the Princeton WordNet (PWN) evocation dataset (Boyd-Graber et al., 2006) that collects human ratings of the evocation strength; the other is Ma's evocationNet dataset (Ma, 2013) that provides directionality judgments.

2.1 PWN evocation dataset

The PWN evocation dataset[1] is a collection of ratings of evocation strength for 119,652 PWN synset pairs (Boyd-Graber et al., 2006). Each synset pair was judged by at least three raters, where each rating ranges between 0 and 100. We simply averaged the ratings in this work. As the synset pairs had been *randomly* selected from the *core synsets*, two thirds of them (80,343) are rated as zero, which means "no evocation." The mean and the standard deviation of the ratings greater than zero are 8.389 and 12.00, respectively, showing that the evocation strengths vary considerably.

For example, for some of the positively rated synset pairs (39,309), the evocation from `prize.n.01` to `honor.n.02` records the highest strength of 100.0, whereas that from `critical.a.04` to `obstruct.v.01` shows the lowest, namely 0.0625. Figure 1 further displays the distribution of the positive evocation ratings while binning them into five classes: $\{b0 : r > 0, b1 : r \geq 1, b25 : r \geq 25, b50 : r \geq 50, b75 : r \geq 75\}$.

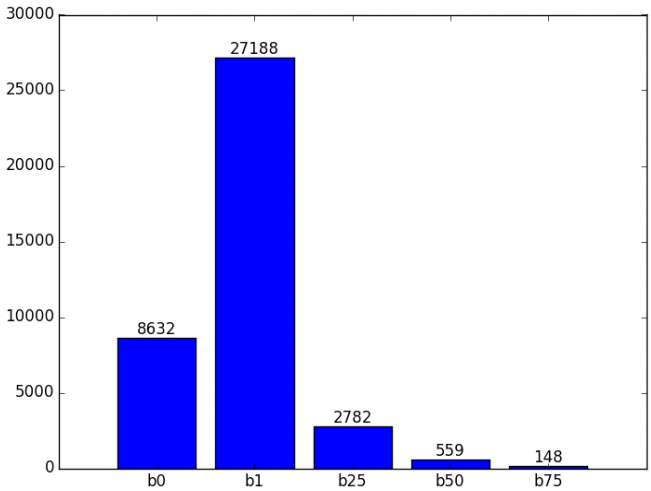


Figure 1: Distribution of the evocation ratings in the PWN evocation data.

Reports from as early as in (Boyd-Graber et al., 2006) showed that major similarity measures could not reproduce the evocation ratings well: the best result reported in the literature was as low as $\rho = 0.131$ (ρ:

[1] `http://wordnet.cs.princeton.edu/downloads.html`

Directionality	Count
$x \rightarrow y$ (outbound)	172,126
$x \leftarrow y$ (inbound)	123,147
$x \leftrightarrow y$ (bidirectional)	43,459
no-evocation (original)	9,715
no-evocation (incorporated from PWN)	90,058
Total	428,790

Table 1: Distribution of the directionality categories in Ma's evocationNet dataset (augmented by PWN Evocation data).

the Spearman correlation coefficient), which was achieved with semantic vectors derived by applying the Latent Semantic Analysis (LSA) method (Deerwester et al., 1990) toward the British National Corpus. Remind here, however, that the work (Boyd-Graber et al., 2006) did not intend to develop a method to predict evocation strengths, rather their intention was to explore a different type of semantic relationship that could be incorporated into PWN.

2.2 Ma's evocationNet dataset

Ma (2013) presented a method to create a dataset of concept pairs that are in evocation relation, and she publicized the dataset, which we refer to as Ma's evocationNet[2]. This dataset provides directionality annotations for a number of PWN synset pairs, but not their evocation strength ratings. She created this dataset by converting the word-based association data given in *The University of South Florida Free Word Association Norms* (Nelson et al., 2004) into *word sense*-based data by first applying an automatic word sense disambiguation process, and then a manual verification process. Because of this creation process, Ma's evocationNet dataset contains a number of duplications in the synset-level, but we did not exclude these duplicated data in the present research.

The dataset consists of 13,975 files from which we extracted 348,447 synset pairs. Each of the files is designated by a pair consisting of a word and a synset. As a simple example, the content of the file named `banana{banana.n.01}.txt` (for word: banana, synset: `banana.n.01`) is given below, showing this synset `banana.n.01` (*food* sense) is linked to the synsets `apple.n.01` as well as `orange.n.01` and `orange.n.02`, whereas it is co-linked with the synset `banana.n.02` (*plant* sense).

```
apple {apple.n.01}++
orange {orange.n.01}++
oranges {orange.n.01}++
banana {banana.n.02}+=
```

That is, the symbol '++' denotes an outbound link, whereas '+=' indicates a bidirectional link. Other categories that appeared in the dataset are: '+-' (inbound) and '==' (no evocation).

Table 1 counts the frequencies of the directionality categories. As displayed in the table, Ma's original dataset contains a relatively small number of "no-evocation" instances that causes the problem of skewed distribution. Thus, as a remedy, we added the synset pairs of which the evocation strength was rated as zero in the PWN dataset to this category, finally giving us a data set of 428,790 synset pairs.

3 Supervised Learning Approach

3.1 Machine-Learning Frameworks

Since the psychological mechanism underlying evocation or association is not yet well understood (De Deyne and Storms, 2015), it is natural to use an exploratory approach to build a computational method that exploits potentially effective features obtained from several resources. As detailed in the

[2] `http://kettle.ubiq.cs.cmu.edu/~xm/DataSet/webpage/evocationNet/`

previous section, the data for evocation strength are numerical ratings, and the data for evocation directionality are discrete categories. We therefore naturally defined the prediction of evocation strength as a regression task, and the determination of evocation direction as a classification task.

We adopt a supervised machine-learning approach, and compare a feed-forward neural network (NN) with the Random Forest™(RF) algorithm as the basic learning framework. Since the submitted paper is not intended as a contribution to the machine-learning field, we simply applied straightforward or off-the-shelf classifiers/regressors in the experiments. The hyperparameters were determined through a series of pre-experiments.

Neural Network: We adopted simple perceptron with two hidden layers, both for the regression task and the classification task. We applied *dropout* and employed *ReLu* as the activation functions. *Mean squared error* and *Adam algorithm* were utilized for the optimization in the regression tasks. Almost the same architecture was adopted for the classification tasks, where *softmax cross entropy* was adopted as the error function, and the number of nodes in the output layer was equal to the number of classes (that is, four). We utilized a Python-based framework known as chainer[3] for the implementation.

Random Forest: We employed another Python-based framework named scikit-learn[4] for the Random Forest classifiers and regressors. The hyperparameters we used were similar to the default parameters of the system, except that the number of estimators was boosted to 125 from the default of 10.

3.2 Features

We integrate potentially effective features, which can be divided into the following three groups.

3.2.1 Similarity/relatedness features

Even for an asymmetric semantic relationship such as evocation, *symmetric* similarity/relatedness would provide a certain *bias* or *basis* (De Deyne et al., 2013). With this motivation, we utilize four similarity/relatedness features as shown below. Note here that (c) and (d) are synset-based, whereas (a) and (b) are word-based. We were able to incorporate these word-based similarities, as the utilized data explicitly specify the focused word for each of the synsets. In addition, notice that (b) and (d) rely on distributed representation vectors, whereas (a) and (c) do not.

- (a) $ldaSim$ provides the cosine similarity between the word vectors created by applying the Latent Dirichlet Allocation (LDA) algorithm (Hoffman et al., 2010). We trained an LDA model from the enwik9 Wikipedia corpus[5] while using the gensim[6] Python library for topic modeling. The dimensionality of the vectors is 300, which is the same as the vectors employed by (b) and (d).

- (b) $w2vSim$ provides the cosine similarity between Word2Vec-induced word embedding vectors (Mikolov et al., 2013a). We used the pre-trained 300-dimensional vectors available at Google's Word2Vec site[7]. Note that these vectors were created by applying the continuous bag-of-words (CBOW) model.

- (c) $wupSim$ computes Wu-Palmer similarity (Budanitsky and Hirst, 2006) defined by the formula shown below: here, $depth(s)$ gives the depth of node s from the root; $lcs(s,t)$ computes the least common subsumer node of s and t. Wu-Palmer similarity is convenient in the sense that the similarity ranges $0 < wupSim(s,t) \leq 1$. To cope with cross-POS evocation relations, we assumed a virtual root node that integrates PWN's POS-oriented subtrees.

$$wupSim(s,t) = \frac{2 \times depth(lcs(s,t))}{depth(s) + depth(t)}$$

[3] http://chainer.org/
[4] http://scikit-learn.org/
[5] http://mattmahoney.net/dc/text.html
[6] https://radimrehurek.com/gensim/
[7] https://code.google.com/p/word2vec/

- (d) $autoexSim$ provides the cosine similarity between Autoextend synset semantic vectors. These 300-dimensional vectors were created by the method proposed in (Rothe and Schütze, 2015), and made available on the author's site[8]. Recall that these vectors were induced by using the same Word2Vec CBOW embedding vectors described above. We adopt the Autoextend method, since it conveniently utilizes the semantic-relational structure that resides in an existing lexical-semantic resource (in this case, PWN), while exploiting ready-made word embedding vectors.

3.2.2 Lexical resource features

Lexical resources, such as PWN, can be utilized as a precious source of features. In the present work, we employed the three features listed below. Note that all of these features have been incorporated in expectation of contributing to capturing some *asymmetric* aspects of evocation relationships.

- $posSem$ is introduced to dictate some of the fundamental attributes of the query synset pair. It concatenates the feature vector of the source concept with that of the target concept in this order. Each feature vector for a concept consists of a *1-of-k* encoding of the part-of-speech sub-vector (five dimensions, corresponding to: *a*, *s*, *n*, *r*, and *v*) and a 45-dimensional sub-vector for the coarse-level semantic classification. This eventually provides us with a 100-dimensional ($(5 + 45) \times 2$) vector to represent the query synset pair. As for the labels of the coarse-level semantic classes, we utilize the names of the lexicographer files in PWN, such as `noun.artifact`, `noun.process`, and `verb.motion`.

- $lexNW$ attempts to dictate the difference in graph-theoretic *influence* of the source/target concepts in the underlying PWN lexical-semantic network. The rationale behind this is: the evocation relation from a less important concept to a weighty concept may be more likely than in the reverse direction. More specifically, we compute the betweenness and load centralities (Barthélemy, 2004) for each synset node, and dispose the values in the order of source concept and target concept, resulting in a four-dimensional vector for the query synset pair.

  The betweenness centrality $bc(v)$ defined by the formula below measures the influence of the designated node in terms of network flow. In the formula, $npaths(a, b, c)$ counts the number of the shortest paths from a through c to b. The load centrality also assesses the importance of a node by using load distribution.

$$bc(v) = \sum_{s \neq v \neq t} \frac{npaths(s, t, v)}{npaths(s, t, *)}$$

  We computed these graph-theoretic metrics simply by using the NetworkX[9] Python library for processing complex networks.

- $dirRel$ also exploits the network structure of PWN, attempting to mimic the notion of *feature inclusion*: "an object with many features is judged as less similar to a sparser object than vice versa" (Gawron, 2014). In the present work, sets of neighboring concept nodes in the lexical-semantic network are considered as features.

  Figure 2 illustrates the notion of $dirRel(s, t, k)$, suggesting that this measure is associated with paths connecting s to t. In the present work, we set $k = 3$ as it performed best in the preliminary experiments. Note that this means that we considered the semantic paths in PWN of which the length is at most six as effective features.

  The defining formula shown below simply quantifies the overlap in *k*-neighbor nodes in the PWN lexical-semantic network, in which $nb(x, k)$ denotes the set of *k*-neighbor nodes of node x.

$$dirRel(s, t, k) = \frac{|nb(s, k) \cap nb(t, k)|}{|nb(s, k)|}$$

[8] `http://www.cis.lmu.de/~sascha/AutoExtend/embeddings.zip`
[9] `https://networkx.github.io/`

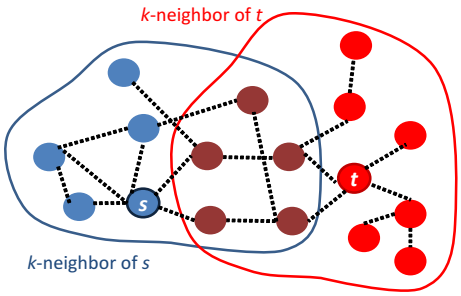


Figure 2: Notion of $dirRel(s, t, k), k = 2$.

Notice that this formula is a version of the well-known Tversky index $TI(X, Y)$ (Tversky, 1977) (where $\alpha = 1$ and $\beta = 0$), which dictates the asymmetric similarity of two given sets, X and Y.

$$TI(X, Y) = \frac{|X \cap Y|}{|X \cap Y| + \alpha|X - Y| + \beta|Y - X|}$$

3.2.3 Semantic relational vectors

One of the prominent advantages introduced by distributional word embedding vectors is the tendency: "all pairs of words sharing a particular relation are related by the same constant (vector) *offset*" (Mikolov et al., 2013b). Following this result, a number of research groups have tried to capture the characteristics of a semantic relationship from the offset vectors. Among them, for example, (Fu et al., 2014) successfully learned the hypernymy relationship by estimating the projection matrices that map words to the hypernyms.

Although it is not clear whether a similar approach would be effective for potentially complex semantic relationships such as evocation, we, in the present work, incorporated the offset semantic vectors $relVec(s, t)$ (referred to as *semantic relational vectors* in this research) as a vectorial feature. That is, we concatenated the obtained 300-dimensional relational vector with the vector obtained from other features described so far.

$relVec(s, t)$ could be defined as follows, provided an adequate semantic path from s to t is given by $path(s, t)$ that sequences edges. Here, each edge from a to b may carry a lexical-semantic relation r, and it could be associated with a relation weight $w(r)$.

$$relVec(s, t) = \sum_{(a,b,r) \in path(s,t)} w(r)(semVec(b) - semVec(a))$$

This formula can be simplified as follows by assuming a uniform relation weight ($w(r) = 1, \forall r$). This means that we see *an evocation relation as a short cut of a potential path* in the lexical-semantic network.

$$relVec(s, t) = semVec(t) - semVec(s)$$

In the formula, $SemVec(x)$ basically denotes the Autoextend synset semantic vector for synset x, but, as detailed in the experimental section, we experimentally altered it to other types of vectors.

4 Experiments and the Results

We assessed the proposed framework and investigated the effectiveness of the proposed features by conducting a series of experiments, where a five-fold cross-validation was employed in each of the experimental settings. The performances are measured by computing the Pearson (r) and Spearman (ρ) correlation coefficients between the gold data and the predictions for the regression tasks[10], and by the standard Precision/Recall/F1/Accuracy measures for the classification tasks, respectively. The results achieved by the combination of the proposed features were compared with each individual baseline case. Additionally we performed ablation tests in order to assess the importance of each feature.

[10]We included these two correlation measures, although we realized that the relative ordering captured by the Spearman was more adequate to compare.

4.1 Results: Prediction of strength

The whole combination of the proposed features yielded the best results of $r = 0.4391; \rho = 0.4000$ with NN, which significantly outperformed the results with RF, $r = 0.3695; \rho = 0.3291$. Thus, in the following, we only discuss the results achieved by NN.

Table 2 displays the strength prediction results in r and ρ, where each foo indicates an individual baseline with feature foo. Table 3 additionally displays the results of ablation tests, where each $-foo$ indicates the ablated feature foo. Both tables show two baseline results: one is the figure shown in (Boyd-Graber et al., 2006); the other is the result achieved by a simple baseline which uniformly assigns the average strength (2.756) computed from the entire PWN Evocation dataset.

Feature	r	ρ
All	**0.4391**	**0.4000**
$ldaSim$	0.1559	0.1441
$w2vSim$	0.2472	0.1841
$wupSim$	0.0907	0.0663
$autoexSim$	0.2395	0.1924
$posSem$	0.2442	0.2489
$lexNW$	0.1379	0.1211
$dirRel$	0.0839	0.0622
$relVec$	**0.2931**	**0.2763**
(Boyd-Graber et al., 2006)	NA	0.131
$Average$	0.0	NA

Table 2: Results: Prediction of evocation strength (individual features).

Feature	r	ρ
All	**0.4391**	**0.4000**
$-ldaSim$	0.4378	0.3994
$-w2vSim$	0.4370	0.3991
$-wupSim$	0.4387	0.3997
$-autoexSim$	0.4333	0.3962
$-posSem$	0.4269	0.3837
$-lexNW$	0.4379	0.3999
$-dirRel$	0.4385	0.4000
$-relVec$	**0.3959**	**0.3534**
(Boyd-Graber et al., 2006)	NA	0.131
$Average$	0.0	NA

Table 3: Results: Prediction of evocation strength (ablation tests).

The results listed in Table 2 and Table 3 can be summarized as follows.

- None of the individual baseline features could outperform the feature combination case *All*.
- The semantic relatedness measures based on distributed representation ($w2vSim$ and $autoexSym$) performed relatively well, suggesting that distributed representation of meaning would be promising. Besides, it is shown that even asymmetric evocation relationships could be somewhat recovered by symmetric semantic relatedness.
- A further effective feature was $relVec$, which in this case is a vector computed from the corresponding Autoextend semantic synset vectors. This definitely highlights the effectiveness of distributed representation at the concept level.
- Surprisingly, $posSem$, which essentially is a sparse representation of a synset pair, was a useful feature by itself, implying that characterization at a coarse semantic level could capture some aspects of evocation relationships.
- The contributions of $lexNW$ and $dirRel$, unfortunately, were not remarkable as expected. Indeed, $dirRel$ might have suffered from the relatively sparse connective structure of PWN.

4.2 Results: Determination of directionality

The combination of all the proposed features yielded 0.8703 in total accuracy with RF, which is considerably more accurate than 0.7642 with NN. Thus, in the following, we only discuss the RF results[11].

Table 4 displays the results of the directionality determination for the overall classification accuracy with individual features, whereas Table 5 displays additional results of the ablation test. The tables also

[11]The results are markedly different from the tendency observed in the regression results, where NN is superior to RF. Although, in general, RF algorithms are said to not perform well in regression tasks, this difference should be further examined.

list the accuracy figures obtained by two baselines: *Most frequent* always assigns the most frequent label (*outbound*), whereas *Random* randomly assigns a directionality label while observing the distribution shown in Table 1.

Feature	Accuracy
All	0.8703
$ldaSim$	0.4574
$w2vSim$	0.4872
$wupSim$	0.4014
$autoexSim$	0.4460
$posSem$	0.4674
$lexNW$	*0.7084*
$dirRel$	0.4400
$relVec$	**0.7939**
Most frequent	0.4014
Random	0.2860

Table 4: Results: Determination of evocation directionality (individual features).

Feature	Accuracy
All	0.8703
$-ldaSim$	0.8741
$-w2vSim$	0.8771
$-wupSim$	0.8709
$-autoexSim$	0.8704
$-posSem$	0.8684
$-lexNW$	0.8670
$-dirRel$	0.8704
$-relVec$	**0.7047**
Most frequent	0.4014
Random	0.2860

Table 5: Results: Determination of evocation directionality (ablation tests).

The results in Table 4 and Table 5 can be summarized as follows.

- Similar to the strength prediction problem, the semantic relational vectors $relVec$ played the most significant role: Without this feature, the accuracy drops by almost 17%. This clearly indicates that the semantic offset vectors are also useful in this classification task.
- Most of the individual features, including the asymmetric features ($posSem$ and $dirRel$) did not perform well by themselves. However, somewhat surprisingly, the graph-theoretic metric $lexNW$ played a greater role by itself (Accuracy=0.7084), implying that this simple NW-based metric could capture some aspect of the directionality of evocation relationships.
- On the contrary, contributions of the similarity/relatedness features are not very prominent even comparing to the random baseline. This might be however reasonable, given that these similarity/relatedness measures are innately symmetric.

Table 6 breaks down the results for the *All* feature case. It shows that the performance is generally good, but that distinguishing "no-evocation" from the other features is rather difficult. We may need to incorporate features that explicitly model some *irrelevancy* between the concept pair.

Directionality	Precision	Recall	F1
outbound	0.8311	0.9272	0.8765
inbound	0.9008	0.8029	0.8491
bidirectional	0.9600	0.9992	0.9792
no-evocation	0.8716	0.7913	0.8295

Table 6: Breakdown of the results for *All* features (Accuracy=0.8703).

4.3 Results: Types of semantic relational vectors

It is now evident that semantic relational vectors can be employed as a highly effective feature in both task types, suggesting that the characteristics of semantic relations, even though they are largely vague relationships such as evocations, can be captured to some extent by offset semantic vectors. The results reported thus far were achieved by utilizing Autoextend synset semantic vectors. However, alternatives

Vector type	r	ρ	Accuracy
synset	0.4391	0.4000	**0.8703**
w2v	**0.4551**	**0.4158**	0.7535
lexeme	0.4267	0.3880	0.7636

Table 7: Comparison of relational vector types.

would be possible: Word2Vec embedding vectors and the Autoextend lexeme semantic vectors can be assayed.

Table 7 thus compares the results when the type of semantic relational vector was altered. The most prominent observation in this table is: the synset-based relational vectors are far more effective than other types of vectors in the directionality classification task (0.8703 compared to around 0.76 in accuracy). This may indicate that a concept-level representation is more adequate in the classification task, which essentially is a coarser-level task compared to the strength prediction task. On the other hand, the strength prediction task may benefit from word-oriented features, as $relVec(w2v)$ produced the most accurate results (although the differences are subtle).

Interestingly however, the in-between representation $relVec(lexeme)$ failed to perform better than any of these. Although the reason should be further examined, it might reflect the mechanism of Autoextend that employs auto-encoders having the layer of lexemes as the middle layer.

5 Related Work

Among the efforts to enrich wordnets, similar to the work of (Boyd-Graber et al., 2006; Ma, 2013) are (Nikolova et al., 2012) and (Lebani and Pianta, 2012): The former, in the context of ViVA project, populated PWN with evocation links collected by using a crowd sourcing service, which may help people with anomic aphasia to navigate among words and concepts; the latter also extended PWN, but with more semantically relevant feature descriptions acquired from human subjects. Note also that these two projects may share the purpose: assisting people with verbal disorders.

Although there is a scarcity of research into a computational mechanism for predicting evocation relationships, some related research can be found in the areas of psycholinguistic semantic association and asymmetric semantic relatedness. Among a number of psychological/cognitive research studies on mental lexicons (Maki et al., 2004; De Deyne et al., 2013; De Deyne and Storms, 2015), De Deyne and Storms (2015) argue that "the entire set of connections between a pair of words in a large network of knowledge may determine the associative strength between them." Although the $dirRel$ feature proposed in this paper partly reflects this indication (since it virtually considers multiple short paths between the synsets), we could probe the network structures of the underlying lexical-semantic resources even further.

Since an evocation relation is a type of asymmetric semantic relation between lexicalized concepts, it is naturally associated with the issue of asymmetric similarities (Tversky, 1977). As already mentioned in this paper, the central notion behind the asymmetric similarity/relatedness is *feature inclusion*. In particular, (Kotlerman et al., 2010) and (Gawron, 2014) make use of dependency parses acquired from textual corpora as features. These features could potentially also be effective in predicting the evocation relations in part.

In addition to these areas, the use of semantic relational vectors (offset semantic vectors) (Fu et al., 2014; Bollegala et al., 2015; Necsulescu et al., 2015; Vylomova et al., 2016) would be worth pursuing further, as the present results strongly insist that these vectors could be an effective source of features. Our preliminary attempts (not discussed in this paper) to simply cluster the set of offset vectors into groups, however, have been proven ineffective in the present task. This may confirm that an evocation relation may be a composite of elementary semantic relations (Boyd-Graber et al., 2006). In this regard, we would need to further explore the semantic relational vectors, while considering the semantic paths connecting the source and target concepts in lexical-semantic networks. Presumably, deciding the edge weight $w(r)$ for each lexical-semantic relation type in the formula given in 3.2.3 would be a key to this issue. A closely related approach found in the literature is the "bag-of-edges" approach (Shwartz et al.,

2015) for automatically selecting an optimized subset of resource relations in a structured resource, given a target lexical inference task.

6 Concluding Remarks

This paper proposed a supervised learning approach to predict the strength and to determine the directionality of the evocation relation between lexicalized concepts. The empirical results evidently showed that the combination of the proposed features largely outperformed the individual baselines, and insisted that the *semantic relational vectors* computed from existing semantic synset embedding vectors (Rothe and Schütze, 2015) are quite useful in both tasks.

Although the achieved performances ($\rho \approx 0.42$ in regression; $Accuracy \approx 0.87$ in classification) are substantially superior to the figures reported in the literature, they could be further improved by applying more sophisticated machine learning frameworks. Once we get better performance figures, the proposed mechanism could be utilized with semantic NLP downstream applications, such as the measurement of textual similarity/relatedness and the lexical chaining in discourse.

Possible research directions for breakthroughs are (at least) threefold. First, we could explore more appropriate representations for words/lexemes/lexicalized-concepts. Although we proved in this research that the AutoExtend vectors were excellent, other approaches might be more effective. Among the possibilities we are interested in are the incorporation of perceptual features such as those acquired from images (Silberer and Lapata, 2014; Kiela and Bottou, 2014), as some evocation relationships might be rooted in human perception.

Second, we should incorporate more relational features. In particular, asymmetric similarity features that can be acquired from corpora (Kotlerman et al., 2010; Gawron, 2014) would be beneficial, as some of the evocation relations are rather direct asymmetric relations, such as hypernymy and meronymy. To facilitate this line of research, we would start with error analysis of the present results.

Finally, but not least important, we can exploit rich but latent information from a range of semantic resources, such as those with a networked structure; not necessarily limited to PWN. As an evocation relation could be considered as a shortcut for some longer semantic/conceptual association chains, paths in the semantic networks that link the source and target concepts could be utilized as a useful source of features.

Acknowledgments

The present research was supported by JSPS KAKENHI Grant Numbers JP26540144 and JP25280117.

References

Marc Barthélemy. 2004. Betweenness centrality in large complex networks. *European Physical Journal B*, 38:163–168.

Danushka Bollegala, Takanori Maehara, and Ken ichi Kawarabayashi. 2015. Embedding semantic relations into word representations. In *Proc. of International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 1222 – 1228.

Jordan Boyd-Graber, Christiane Fellbaum, Daniel Osherson, and Robert Schapire. 2006. Adding dense, weighted connections to wordnet. *Proceedings of the third international WordNet conference*, pages 29–36.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational. Linguistics.*, 32(1):13–47.

Irene Cramer. 2008. How well do semantic relatedness measures perform? A meta-study. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, volume 1, pages 59–70.

Simon De Deyne and Gert Storms. 2015. Word associations. In John R. Taylor, editor, *The Oxford Handbook of the Word*, pages 466–480. Oxford University Press.

Simon De Deyne, Daniel J. Navarro, and Gert Storms. 2013. Associative strength and semantic activation in the mental lexicon: evidence from continued word associations. In *Annual Conference of the Cognitive Science Society edition:35*, pages 2142–2147.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *JOURNAL of the American Society for Information Science*, 41(6):391–407.

Christiane Fellbaum, editor. 1998. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London, May.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1199–1209.

Jean Mark Gawron. 2014. Improving sparse word similarity models with asymmetric measures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 296–301.

Matthew D. Hoffman, David M. Blei, and Francis Bach. 2010. Online learning for latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, volume 24.

Douwe Kiela and Léon Bottou. 2014. Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16:359–389.

Gianluca E. Lebani and Emanuele Pianta. 2012. Encoding commonsense lexical knowledge into wordnet. In *Proceedings of the 6th International Global WordNet Conference (GWC2012)*, pages 159–166.

Xiaojuan Ma. 2013. Evocation: Analyzing and propagating a semantic link based on free word association. *Language Resources and Evaluation*, 47(3):819–837.

William S. Maki, Lauren N. McKinley, and Amber G. Thompson. 2004. Semantic distance norms computed from an electronic dictionary (wordnet). *Behavior Research Methods, Instruments, & Computers*, 36(3):421–431.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Silvia Necsulescu, Sara Mendes, David Jurgens, Núria Bel, and Roberto Navigli. 2015. Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 182–192, Denver, Colorado, June. Association for Computational Linguistics.

Douglas L. Nelson, Cathy L. McEvoy, and Thomas A. Schreiber. 2004. The university of south florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, and Computers*, 36:402–407.

Sonya Nikolova, Jordan Boyd-Graber, and Christiane Fellbaum, 2012. *Collecting Semantic Similarity Ratings to Connect Concepts in Assistive Communication Tools*, pages 81–93. Springer Berlin Heidelberg, Berlin, Heidelberg.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1793–1803.

Vered Shwartz, Omer Levy, Ido Dagan, and Jacob Goldberger. 2015. Learning to exploit structured resources for lexical inference. In *Proceedings of the 19th Conference on Computational Natural Language Learning, CoNLL 2015*, pages 175–184.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 721–732, Baltimore, Maryland.

Amos Tversky. 1977. Features of similarity. *Psychological Review*, 84:327–352.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1671–1682, Berlin, Germany, August. Association for Computational Linguistics.

Collecting and Exploring Everyday Language for Predicting Psycholinguistic Properties of Words

Gustavo Henrique Paetzold and **Lucia Specia**
Department of Computer Science
University of Sheffield, UK
{g.h.paetzold,l.specia}@sheffield.ac.uk

Abstract

Exploring language usage through frequency analysis in large corpora is a defining feature in most recent work in corpus and computational linguistics. From a psycholinguistic perspective, however, the corpora used in these contributions are often not representative of language usage: they are either domain-specific, limited in size, or extracted from unreliable sources. In an effort to address this limitation, we introduce SubIMDB, a corpus of everyday language spoken text we created which contains over 225 million words. The corpus was extracted from 38,102 subtitles of family, comedy and children movies and series, and is the first sizeable structured corpus of subtitles made available. Our experiments show that word frequency norms extracted from this corpus are more effective than those from well-known norms such as Kucera-Francis, HAL and $\text{SUBTLEX}_{\text{us}}$ in predicting various psycholinguistic properties of words, such as lexical decision times, familiarity, age of acquisition and simplicity. We also provide evidence that contradict the long-standing assumption that the ideal size for a corpus can be determined solely based on how well its word frequencies correlate with lexical decision times.

1 Introduction

Large corpora of text are certainly one of the most fundamental resources in the field of Computational Linguistics. In Psycholinguistics, it has been long established that word frequencies from corpora play a very important role in cognitive processes. Brysbaert and New (2009) points out that frequently occurring words are often much more easily perceived, recalled and associated than rare words (Balota and Chumbley, 1984; Rayner and Duffy, 1986). In Text Simplification, researchers have found a strong relationship between frequencies and word simplicity (Devlin and Tait, 1998).

An inherent limitation of work based on word frequency analysis is that the type of resource used as a corpus is often built for a specific communication purpose, such as news (Burgess and Livesay, 1998). This is however not representative of everyday language usage, particularly from a psycholinguistic perspective. The other extreme of the spectrum features resources compiled from user-generated content, such as micro-blogs. However, these resources often suffer from grammar errors and misspellings, excessive use of acronyms and shortenings, partly due to the constrains of the publication means (e.g. limited number of characters) (Pak and Paroubek, 2010).

This is particularly concerning given that previous research has shown that the source from which a corpus was extracted is one of its most important defining traits. For example, the experiments of Brysbaert and New (2009) and Shardlow (2013) reveal that frequencies from spoken text have a much stronger correlation with psycholinguistic word properties than those from other sources. Their findings greatly highlight the potential of spoken language text, but there are very few examples of resources of this kind available for English. $\text{SUBTLEX}_{\text{us}}$ is a notable exception: it contains texts extracted from 8,388 subtitles of American movies, and is freely available for download. The OpenSubtitles2016 corpus (Lison and Tiedemann, 2016) is another example, featuring sentences extracted from numerous subtitle files aligned at sentence level across 60 languages.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1669–1679, Osaka, Japan, December 11-17 2016.

However, since the subtitles in these corpora are not restricted with respect to genre or domain, their proficiency in capturing everyday language can also be limited. Movies and series span from lighthearted productions for toddlers to historic dramas targeting older audiences, with very distinct vocabulary used. In this paper, we explore the use of everyday language corpora in psycholinguistic applications. In an effort to address the lack of reliable everyday language corpora for English, we create SubIMDB, the first structured corpus of subtitles in the literature. SubIMDB is composed of subtitles of movies and series written for the "average audience", and can be downloaded in useful formats. In the sections that follow, we describe the resources and procedures used to build SubIMDB, and evaluate its performance in various tasks.

2 Building SubIMDB

Our goal in creating SubIMDB was to compile and provide freely a large, structured corpus of everyday language. As a data type, we chose subtitles of movies and series, since they are available for dozens of languages. Another advantage of using subtitles as opposed to, for example, chat logs or podcast transcripts, is that movies and series are subject to production standards, and hence the subtitles created for them tend to be composed of linguistically correct constructs.

2.1 Acquiring Subtitles

To create a reliable corpus of subtitles one must take into account that movies and series can be of many different genres, and may target very distinct audiences. The compilation of $\text{SUBTLEX}_{\text{us}}$ involved the download of 8,388 subtitles of U.S films and series released between 1900-2007, with no restriction with respect to genre. We took a different approach when creating SubIMDB. We use OpenSubtitles[1] as a data source. One can download subtitles from their API by providing with a production's unique IMDb[2] identifier.

As the first step in creating SubIMDB, we queried the IMDb platform searching for identifiers of six types of content: family movies, family series, comedy movies, comedy series, movies for children and series for children. We chose these genres because productions of this kind tend to target viewers of either young or all ages, and hence tend to use accessible language. Our hypothesis is that word usage statistics from this type of content correlate better with psycholinguistic properties of words, such as lexical decision times and age of acquisition.

To obtain the identifiers, we used the IMDb engine[3] to search for and parse all pages under the family and comedy feature film pages, as well as the ones under the family and comedy series categories. Since IMDb does not contain a category specific for children movies and series, we resorted to 15 movies and series lists created by IMDb users to obtain them. In total, we obtained the IMDb identifiers of 9,709 family movies, 8,008 family series, 66,411 comedy movies, 24,776 comedy series, 745 children movies and 124 children series.

We then queried the online OpenSubtitles API for each of these 109,773 IMDb identifiers. Surprisingly, we were only able to find subtitles for 12,618 movies and series. On the other hand, since series are comprised of various episodes, we downloaded subtitles for each episode of every season available in OpenSubtitles. A total of 38,102 subtitles were collected in this way.

2.2 Processing Subtitles

In order to make their content more easily accessible, we first tokenized all lines in the subtitles and removed any HTML tags. A filtering algorithm was then applied to discard subtitle lines which:

1. **Refer to metadata or timing indicators**: These lines do not contain meaningful information.

2. **Have more than 80 characters**: In most cases, lines with close to or more than 80 characters are composed of sequences of random spurious characters.

[1] http://www.opensubtitles.org
[2] http://www.imdb.com
[3] http://www.imdb.com/search

Size	HF	LF	All
10M	$\mathbf{-0.393}$	-0.391	-0.576
20M	-0.393	-0.433	-0.601
30M	-0.392	-0.454	-0.613
40M	-0.390	-0.471	-0.624
50M	-0.391	-0.465	-0.620

Size	HF	LF	All
60M	-0.390	-0.468	-0.622
70M	-0.390	-0.469	-0.622
80M	-0.391	-0.470	-0.623
90M	-0.392	-0.470	-0.623
100M	-0.392	$\mathbf{-0.471}$	$\mathbf{-0.624}$

Table 1: Pearson correlation of decision times and high and low frequency words per corpus size.

3. **Have at least one word with more than 15 characters**: Lines with unusually long words tend to be incorrectly formatted sentences.

4. **Contain advertisement**: These lines refer to credits attributed to the creators of the subtitles in question. Some examples of expressions targeted are "synched by" and "opensubtitles.org".

The resulting corpus contains 225,847,810 words in 38,643,849 lines, which is 4.5 times bigger than $\text{SUBTLEX}_{\text{us}}$.

2.3 Reliability Assessment

One of the most popular strategies for frequency norm quality assessment is to evaluate how well they predict lexical decision times. A very popular task in the field of Psycholinguistics, *lexical decision*, also known as *lexical reaction time*, refers to the process of deciding whether or not a given sequence of characters is a real word of the language in question (Balota et al., 2007). Previous work has measured the time taken by subjects to make such a decision for certain words, then used correlation metrics to assess how well their frequencies can predict them (Balota et al., 2004; Van Heuven et al., 2014; Vega et al., 2011; Brysbaert and New, 2009). In this section, we evaluate the reliability of SubIMDB by replicating some of the lexical decision experiments of Brysbaert and New (2009) and Burgess and Livesay (1998).

Brysbaert and New (2009) reveal that the size of a spoken text corpus plays a role in its utility. In general, but not always, larger corpora tend to capture psycholinguistic properties of words more effectively, given that they tend to feature a broader vocabulary and a wider array of distinct contexts from which to extract word usage statistics. But going beyond the "the bigger, the better" assumption, Burgess and Livesay (1998) propose that the ideal corpus size depends on the frequency of the words which one aims to predict the lexical decision times for.

To replicate their experiments, we first sample SubIMDB in portions containing 10 to 100 million words from sentences selected at random. As our test set, we use the MRC psycholinguistic Database (Coltheart, 1981), which provides lexical decision times for 40,468 words. Like in (Brysbaert and New, 2009), we consider only the subset of 38,130 lowercase words in order to avoid most abbreviations and proper nouns.

We split these 38,130 words in two sets: high and low frequency words. A word is considered high frequency (HF) if it is among the 1% most frequently occurring words in SubIMDB, otherwise, it is considered low frequency (LF). This methodology resembles that of Burgess and Livesay (1998). The Pearson correlation between word frequencies and lexical decision times for each corpus size are presented in Table 1.

The scores support the hypothesis in (Burgess and Livesay, 1998): the Pearson correlation for high frequency words peaks at 10 million words, while the correlation for low frequency words continuously grows from 10 to 100 million words. The increase in corpus size reflects positively on the overall performance of SubIMDB for all words, contradicting the results obtained by Brysbaert and New (2009), which suggest that a corpus does not need to have more than 16 million words in order to be cost effective.

As discussed in (Hauk and Pulvermüller, 2004), word length can also influence lexical decision times. Intuitively, one would expect to take longer to read a ten character word than a three character word, for example. Inspecting the word frequencies from SubIMDB, we found that larger words tend to benefit from larger corpora. Table 2 shows the Pearson correlation scores with lexical decision times obtained by SubIMDB samples in different sizes with respect to word length in characters.

	2	3	4	5	6	7	8	9
Count:	38M	85M	77M	18M	11M	8M	5M	3M
10M	**−0.736**	**−0.591**	−0.606	−0.576	−0.552	−0.529	−0.498	−0.455
20M	−0.728	−0.580	−0.608	−0.584	−0.564	−0.545	−0.522	−0.482
30M	−0.727	−0.584	−0.612	**−0.588**	−0.571	−0.556	−0.531	−0.498
40M	−0.716	−0.586	**−0.617**	−0.570	−0.559	−0.546	−0.532	−0.506
50M	−0.723	−0.583	−0.615	−0.583	−0.571	−0.556	−0.535	−0.505
60M	−0.721	−0.581	−0.615	−0.582	**−0.572**	**−0.557**	−0.536	−0.506
70M	−0.712	−0.579	−0.616	−0.581	−0.570	−0.554	**−0.537**	−0.506
80M	−0.713	−0.579	−0.617	−0.581	−0.569	−0.554	−0.535	**−0.508**
90M	−0.714	−0.581	−0.617	−0.579	−0.568	−0.552	−0.536	−0.508
100M	−0.714	−0.581	−0.617	−0.578	−0.568	−0.553	−0.537	−0.508

Table 2: Pearson correlation of decision times and word size per corpus size. Columns represent word length, rows represent corpus size, and cells depict Pearson correlation scores.

Table 2 shows that the scores for long words tend to require larger corpora. This could be explained by the hypothesis of Burgess and Livesay (1998), since the words' length and frequency in SubIMDB are inversely proportional. As illustrated in the second row of the table, shorter words occur much more frequently than longer words in SubIMDB.

Our findings also agree with the ones of Brysbaert and New (2009), who observed that, contrary to norms obtained from news articles and web content, spoken language text norms are better at predicting lexical decision times for shorter words. Notice that, while the correlation for shorter words tend to peak around -0.6, the correlation for longer words peaks around -0.5. This also applies to words with lengths beyond 9 characters: at around 15 characters, correlation values peak around -0.3.

Table 3 shows Pearson correlation scores for the HAL (Burgess and Livesay, 1998) and SubIMDB corpora with respect to word length. Unlike SubIMDB, the HAL corpus is composed of news articles. Much like what is observed in (Brysbaert and New, 2009), while the SubIMDB norm considerably outperforms HAL for words with 2-4 characters, the HAL corpus gives more reliable norms for words with 5+ characters. The reason behind SubIMDB's disadvantage with longer words is explained by the fact that words with 2-4 characters compose 80% of SubIMDB's content. Although this observation may seem puzzling at first, it can be easily explainable. Take, for an example, the sentences "what have you done?" and "what do you mean?". Both these sentences are composed entirely of words between two and four characters, and occur very frequently in SubIMDB. Other notable examples are "come on", "I got to go now" and "have a good one". This difference between HAL and SubIMDB suggests that combining frequency norms from different sources could be a good way of creating even more reliable norms.

	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**
HAL	−0.660	−0.543	−0.598	**−0.604**	**−0.605**	**−0.584**	**−0.574**	**−0.544**
SubIMDB	**−0.716**	**−0.586**	**−0.616**	−0.570	−0.559	−0.546	−0.532	−0.506
	••	•••	•••	•••	•••	•••	•••	•••

Table 3: Correlation comparison between frequencies and decision times on a word size basis. The last line indicates a statistically significant difference with SubIMDB given $p < 0.1$ (•), $p < 0.01$ (••) or $p < 0.001$ (• • •) (F-test).

In sections to come, we compare the performance of SubIMDB and numerous other corpora in various psycholinguistic tasks.

3 Predicting Lexical Decision Times

In this experiment we assess how well frequencies from different sets of SubIMDB subtitles fair against other well-known corpora in how they correlate with lexical decision times. For this experiment, we extracted word frequencies from various SubIMDB subcorpora, as shown in Table 4.

All SubIMDB (SubIMDB)	All Comedy content (SubCOM)	Comedy movies (SubCOM-M)
All movies (SubMOV)	All children content (SubCHI)	Comedy series (SubCOM-S)
All series (SubSER)	Family movies (SubFAM-M)	Children movies (SubCHI-M)
All Family content (SubFAM)	Family series (SubFAM-S)	Children series (SubCHI-S)

Table 4: Subcorpora from SubIMDB used to predict lexical decision times

We compare ours to six frequency norms:

- **KF**: Oldest and most widely used frequency norm, calculated over the Brown corpus (Rudell, 1993; Francis and Kucera, 1979).
- **HAL**: *Hyperspace Analogue to Language* word frequency norm, calculated over the HAL corpus, which contains over 131 million words from Usenet newsgroups (Burgess and Livesay, 1998).
- **Wiki**: Word frequencies from Wikipedia, with 97 million words (Kauchak, 2013).
- **SimpleWiki**: Word frequencies from Simple Wikipedia, with 9 million words (Kauchak, 2013).
- **SUBTLEX**: Word frequencies from $\text{SUBTLEX}_{\text{us}}$, with 51 million words (Brysbaert and New, 2009).
- **Open2016**: Word frequencies from OpenSubtitles2016, with 2 billion words (Lison and Tiedemann, 2016).

We regularise all norms using Equation 1, in which f is the frequency norm value of a word w. This transformation has shown to best represent the relationship between word frequencies and lexical decision times (Balota et al., 2004).

$$norm(f(w)) = \log_{10}(f(w) + 1) \tag{1}$$

We use the same lexical decision dataset from our previous experiments as our test set. The results in Table 5 reveal that, while SubIMDB in its entirety yields the highest Spearman (ρ) correlation scores, the SubMOV corpus, which contains only subtitles of movies, yields the highest Pearson (r) correlation. F-tests show a statistically significant difference between frequencies from SubIMDB and all other corpora.

Norm	Size	ρ	r	F-test
KF	1M	-0.517	-0.486	•••
HAL	131M	-0.641	-0.616	•••
Wiki	97M	-0.531	-0.506	•••
SimpleWiki	9M	-0.560	-0.530	•••
SUBTLEX	62M	-0.653	-0.619	•••
Open2016	2B	-0.657	-0.602	•••
SubIMDB	225M	$\mathbf{-0.659}$	-0.624	-
SubMOV	125M	-0.657	$\mathbf{-0.626}$	•••
SubSER	100M	-0.652	-0.620	•••

Norm	Size	ρ	r	F-test
SubFAM	34M	-0.649	-0.614	•••
SubCOM	199M	-0.657	-0.624	•••
SubCHI	17M	-0.634	-0.592	•••
SubFAM-M	17M	-0.640	-0.596	•••
SubFAM-S	17M	-0.632	-0.590	•••
SubCOM-M	107M	-0.655	-0.623	•••
SubCOM-S	91M	-0.651	-0.618	•••
SubCHI-M	8M	-0.625	-0.572	•••
SubCHI-S	8M	-0.606	-0.556	•••

Table 5: Lexical decision prediction correlation scores. The last column indicates a statistically significant difference with SubIMDB given $p<0.1$ (•), $p<0.01$ (••) or $p<0.001$ (•••) (F-test).

Unlike what was reported in (Brysbaert and New, 2009), the HAL norm achieved lower correlation scores than the SUBTLEX norm, despite the fact that the HAL corpus is twice as large as $\text{SUBTLEX}_{\text{us}}$. This contrast highlights the potential of spoken language corpora in lexical decision prediction.

Our results also indicate a poor performance for the Kucera-Francis coefficient. Despite its use in numerous previous contributions (Burgess and Livesay, 1998; Zevin and Seidenberg, 2002; Brysbaert and New, 2009), more modern resources proved more effective. We believe this is caused by the fact that these coefficients are calculated from a corpus that is very small when compared to the other resources presented in this paper.

4 Predicting Psycholinguistic Properties

In addition to lexical decision times, other psycholinguistic properties of words have been studied in terms of their correlation with frequency norms (Paetzold and Specia, 2016a). In this experiment, we evaluate how well the norms described in Section 3 correlate with four psycholinguistic properties extracted from the MRC psycholinguistic Database:

- **Familiarity**: Available for 9,392 words – frequency with which a word is seen, heard or used daily.
- **Age of Acquisition**: Available for 3,503 words – age at which a word is learned.
- **Concreteness**: Available for 8,228 words – how "palpable" the object the word refers to is.
- **Imagery**: Available for 9,240 words – intensity with which a word arouses images.

The results in Table 6 reveal that SubFAM-M (family movies) performs better than all other norms in predicting age of acquisition and concreteness, although it is 117 times smaller than OpenSubtitles2016 (Open2016). F-tests reveal a statistically significant difference between SubIMDB and all other corpora.

		Age of Acquisition		Familiarity		Concreteness		Imagery	
	Size	r	F-test	r	F-test	r	F-test	r	F-test
KF	1M	-0.447	•••	0.669	•••	$\mathbf{-0.180}$	•••	-0.045	•••
HAL	131M	-0.511	•••	0.732	•••	-0.064	•••	0.086	•••
Wiki	97M	-0.412	•••	0.676	•••	-0.043	•••	0.084	•••
SimpleWiki	9M	-0.486	•••	0.667	•••	0.011	•••	0.129	•••
SUBTLEX	62M	-0.676	•••	0.774	•••	0.017	•••	0.190	•••
Open2016	2B	-0.666	•••	**0.799**	•••	-0.003	•••	0.185	•••
SubIMDB	225M	-0.698	-	0.781	-	0.037	-	0.213	-
SubMOV	125M	-0.705	•••	0.777	•••	0.031	•••	0.212	•••
SubSER	100M	-0.687	•••	0.777	•••	0.038	•••	0.207	•••
SubFAM	34M	-0.723	•••	0.758	•••	0.038	•••	0.217	•••
SubCOM	199M	-0.696	••	0.781	•••	0.037	•••	0.211	•••
SubCHI	17M	-0.709	•••	0.735	•••	0.028	•••	0.201	•••
SubFAM-M	17M	$\mathbf{-0.746}$	•••	0.742	•••	0.043	•••	**0.220**	•••
SubFAM-S	17M	-0.685	•••	0.743	•••	0.007	•••	0.178	•••
SubCOM-M	107M	-0.698	•••	0.777	•••	0.027	•••	0.207	•••
SubCOM-S	91M	-0.690	•••	0.777	•••	0.042	•••	0.209	•••
SubCHI-M	8M	-0.728	•••	0.723	•••	0.026	•••	0.191	•••
SubCHI-S	8M	-0.670	•••	0.704	•••	-0.006	•••	0.158	•••

Table 6: Pearson correlation of norms with respect to psycholinguistic properties. Columns following correlation scores indicate a statistically significant difference with SubIMDB given $p<0.1$ (•), $p<0.01$ (••) or $p<0.001$ (•••) (F-test).

Perhaps most surprising is the performance of the SubIMDB subset of children movies (SubCHI-M) in predicting age of acquisition. Despite its small size, its performance is still much superior than almost

all corpora, including OpenSubtitles2016, which is over 250 times larger. Comparing word frequencies from SubCHI-M with the ones in OpenSubtitles2016, we found interesting differences. Table 7 shows the most over and underrepresented words in SubCHI-M based on percentages of variance with respect to OpenSubtitles2016.

It can be noticed that while overrepresented words ("turtles", "hedgehog", etc.) are mostly innocent in nature, underrepresented words describe mostly sexual and/or thought-provoking concepts ("vagina", "abortion", etc.). These differences reveal that, although subtitle corpora may share traits in general, the domain from which the subtitles are extracted plays an important role. This highlights the often disregarded advantages of a structured, raw text subtitle corpora like the one we collected here. By making subtitles available in their raw form along with metadata about their source of origin, future research can explore different ways of building the ideal corpus for a given task, e.g. by employing clever subtitle selection and filtering techniques.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Over	hoagy	flintstone	turtles	potter	fantasia	hedgehog	hiccup	dialogue
Under	vagina	abortion	cartel	intercourse	rapist	overdose	porn	pimp

Table 7: Representation contrast between the SubCHI-M and OpenSubtitles2016 corpora.

Inspecting our data, we also found further evidence that, unlike what was found by Brysbaert and New (2009), it is unfeasible to predict the ideal size of a corpus by simply looking at frequency correlation with lexical decision times. Table 8 illustrates Pearson correlation scores of different SubIMDB sample sizes for all aforementioned psycholinguistic properties. The correlation scores all behave differently: while familiarity benefits from larger corpora, the remaining properties do not.

	Age of Acquisition		Familiarity		Concreteness		Imagery	
	ρ	r	ρ	r	ρ	r	ρ	r
10M	-0.686	-0.703	0.770	0.724	0.067	0.018	0.225	0.186
20M	$\mathbf{-0.691}$	$\mathbf{-0.711}$	0.782	0.745	$\mathbf{0.072}$	$\mathbf{0.032}$	$\mathbf{0.234}$	0.207
30M	-0.688	-0.710	0.796	0.761	0.070	0.031	0.233	0.211
40M	-0.677	-0.698	0.804	0.768	0.069	0.030	0.227	$\mathbf{0.213}$
50M	-0.683	-0.706	0.805	0.769	0.066	0.030	0.229	0.211
60M	-0.680	-0.703	0.808	0.772	0.065	0.030	0.229	0.211
70M	-0.679	-0.701	0.809	0.772	0.063	0.028	0.228	0.209
80M	-0.678	-0.701	0.811	0.774	0.063	0.028	0.227	0.209
90M	-0.677	-0.700	0.811	0.774	0.063	0.029	0.227	0.210
100M	-0.676	-0.700	$\mathbf{0.811}$	$\mathbf{0.775}$	0.063	0.029	0.227	0.210

Table 8: Pearson correlation per corpus size for different psycholinguistic properties.

5 Predicting Simplicity

Everyday language corpora can also be useful in predicting word simplicity. In this experiment, we evaluate how well SubIMDB fairs against other corpora when employed as a solution to Lexical Simplification.

As our test set, we use the one from the English Lexical Simplification task of SemEval 2012, which contains 1,710 instances composed of a sentence, a target word, and candidate substitutions ranked by simplicity. This dataset has been widely used and hence allows the comparison of SubIMDB against state-of-the-art solutions for the task. For evaluation, we use Spearman (r) and Pearson (ρ) correlation, as well as the TRank metric proposed by Specia et al. (2012), which measures the rate with which a candidate substitution with the highest gold rank i.e. the simplest, was ranked first by the system.

We compare the performance of all frequency norms described in Section 3 to Google 1T, a corpus composed of over 1 trillion words (Evert, 2010), and the winner system in the SemEval 2012 task, which

Norm	r	ρ	TRank	F-test
KF	0.619	0.626	0.589	•••
HAL	0.630	0.633	0.598	•••
Wiki	0.575	0.583	0.516	•••
SimpleWiki	0.626	0.632	0.570	•••
SUBTLEX	0.649	0.649	0.619	•••
Open2016	0.650	0.647	0.619	•••
SubIMDB	0.654	0.652	0.622	-
SubMOV	**0.660**	**0.658**	**0.623**	•••
SubSER	0.648	0.647	0.619	•••
SubFAM	0.649	0.650	0.615	•••

Norm	r	ρ	TRank	F-test
SubCOM	0.655	0.653	**0.623**	•
SubCHI	0.643	0.645	0.611	•••
SubFAM-M	0.653	0.653	0.618	•••
SubFAM-S	0.647	0.650	0.620	•••
SubCOM-M	**0.660**	**0.658**	**0.623**	•••
SubCOM-S	0.647	0.648	0.618	•••
SubCHI-M	0.650	0.654	0.600	•••
SubCHI-S	0.640	0.644	0.608	•••
Google 1T	N/A	N/A	0.585	-
Best SemEval	N/A	N/A	0.602	-

Table 9: Correlation and TRank scores for frequency norms with respect to simplicity. The fifth column indicates a statistically significant difference with SubIMDB given $p<0.1$ (•), $p<0.01$ (••) or $p<0.001$ (•••) (F-test).

employs a Support Vector Machine ranker that uses a wide array of features (Jauhar and Specia, 2012). The results in Table 9[4] reveal that SubIMDB outperforms all baselines, including Google 1T and the former state-of-the-art for the task in TRank. Nonetheless, some SubIMDB subcorpora are even more effective than using our corpus in its entirety, despite being much smaller.

Work in Text Simplification has, however, explored more than single-word frequency norms, considering for example raw n-gram frequencies and language model probabilities (Horn et al., 2014; Baeza-Yates et al., 2015; Paetzold and Specia, 2016b). Table 10 shows TRank scores obtained on the SemEval 2012 task when using 3-gram and 5-gram raw frequencies and language model probabilities extracted from various corpora. The 3-grams and 5-grams consist in a candidate substitution surrounded by one and two tokens, respectively. For probabilities, we trained 5-gram language models using SRILM (Stolcke, 2002). For the Kucera-Francis (KF) norms we use the Brown corpus (Francis and Kucera, 1979). The HAL corpus is not available for download and hence it could not be tested here.

Table 10 shows that single word frequencies are more effective than both 3-grams or 5-grams in the SemEval 2012 task. We believe that the reason for this lies in the fact that almost all candidate substitutions in each instance of the dataset perfectly fit the context in which the target word was found, both with respect to grammaticality and meaning preservation. This setup disregards the need to account for context, which hence makes the use of n-grams less crucial. Since the representative sparsity of a corpus inherently grows as sequences of words become longer, n-grams with $n \geq 1$ are consequently much less reliable than single-word frequencies for this task in particular. This hypothesis is also supported by the fact that 3-gram frequencies achieved considerably higher scores than 5-gram frequencies.

Nonetheless, there is a clear advantage to using language model probabilities as opposed to raw frequencies for larger n-grams, since language models employ sophisticated smoothing techniques to reduce issues due to sparsity. These findings highlight again how important it is for corpora to be released in raw format to make it possible to train language models.

6 Conclusions

In this paper we presented a study on the application of everyday language corpora in the prediction of psycholinguistic properties of words. For our experiments, we created SubIMDB: a large structured corpus of subtitles of movies and series for the average audience. It contains 38,102 subtitles, each individually annotated with metadata about the movie or series for which they were created. Altogether, our corpus has 225,847,810 words in 38,643,849 lines, which is 4.5 times larger than the widely used $\text{SUBTLEX}_{\text{us}}$ corpus (Brysbaert and New, 2009).

We found that word frequencies from SubIMDB capture lexical decision times more effectively than various other frequency norms. Additionally, we found that using only certain types of subtitles can yield

[4]Specia et al. (2012) only provides results for TRank.

		Frequency				Probability			
		3-grams		5-grams		3-grams		5-grams	
Norm	Size	TRank	F-Test	TRank	F-Test	TRank	F-Test	TRank	F-Test
KF	1M	0.234	●●●	0.234	●●●	0.234	●●●	0.234	●●●
Wiki	97M	0.388	○	0.257	○	0.528	●●●	0.520	●●●
SimpleWiki	9M	0.354	●●●	0.247	●●●	0.557	●●●	0.560	●●●
SUBTLEX	62M	0.402	●●●	0.261	●	**0.588**	○	0.586	○
Open2016	2B	**0.461**	●●●	0.234	●●●	0.564	○	0.550	○
SubIMDB	225M	0.425	-	**0.264**	-	0.582	-	0.564	-
SubMOV	125M	0.401	●●	0.262	○	0.582	○	0.580	○
SubSER	100M	0.399	●●●	0.254	●	0.575	●●●	0.567	●●●
SubFAM	34M	0.379	●●●	0.251	●●	0.577	○	0.569	○
SubCOM	199M	0.416	○	0.261	○	0.577	●●●	0.566	●●●
SubCHI	17M	0.354	●●●	0.246	●●●	0.572	○	0.572	○
SubFAM-M	17M	0.357	●●●	0.248	●●●	0.589	○	**0.587**	○
SubFAM-S	17M	0.364	●●●	0.246	●●●	0.574	○	0.574	○
SubCOM-M	107M	0.398	●●●	0.259	●	0.582	●●●	0.572	●●●
SubCOM-S	91M	0.396	●●●	0.253	●	0.570	●●●	0.564	●●●
SubCHI-M	8M	0.329	●●●	0.242	●●●	0.572	○	0.569	●
SubCHI-S	8M	0.334	●●●	0.243	●●●	0.569	○	0.569	○

Table 10: TRank scores for n-grams. Columns following TRank scores indicate a statistically significant difference with SubIMDB given $p < 0.1$ (●), $p < 0.01$ (●●) or $p < 0.001$ (●●●) (F-test).

noticeable increase in performance. The same was observed for the prediction of other psycholinguistic properties, such as age of acquisition.

Our experiments provided evidence to support (Burgess and Livesay, 1998)'s hypothesis, which states that the ideal size of a corpus depends on the overall frequency of the words which one aims to predict lexical decision times for. Nonetheless, our results also reveal that, unlike what is claimed by Brysbaert and New (2009), one should not attempt to quantify the ideal corpus size based solely on correlation scores with lexical decision times.

Finally, we found that in English Lexical Simplification both word frequencies and language model probabilities from SubIMDB outperform the ones extracted from all other corpora available, as well as the state-of-the-art method for the task. Through these findings, we hope to encourage other researchers to collect and release corpora in more flexible, useful forms rather than simply providing with pre-computed single-word frequency counts.

In future work, we aim to add other types of subtitles to SubIMDB and to study smarter subtitle selection and filtering strategies. We also intend to study the use of other types of spoken text corpora, such as tweets and conversations from Facebook (Herdağdelen and Marelli, 2016), in improving the performance of Natural Language Processing tasks. We released the SubIMDB corpus in both raw form, containing subtitles individually annotated with metadata, and in compiled form. Both versions are freely available for download at `http://ghpaetzold.github.io/subimdb`.

Acknowledgements

This work has been partially supported by the European Commission project SIMPATICO (H2020-EURO-6-2015, grant number 692819).

References

Ricardo Baeza-Yates, Luz Rello, and Julia Dembowski. 2015. CASSA: A context-aware synonym simplification algorithm. In *Proceedings of the 2015 NAACL*, pages 1380–1385.

David A Balota and James I Chumbley. 1984. Are lexical decisions a good measure of lexical access? the role of word frequency in the neglected decision stage. *Journal of Experimental Psychology: Human perception and performance*, 10:340.

David A Balota, Michael J Cortese, Susan D Sergent-Marshall, Daniel H Spieler, and MelvinJ Yap. 2004. Visual word recognition of single-syllable words. *Journal of Experimental Psychology: General*, 133(2):283.

David A Balota, Melvin J Yap, Keith A Hutchison, Michael J Cortese, Brett Kessler, Bjorn Loftis, James H Neely, Douglas L Nelson, Greg B Simpson, and Rebecca Treiman. 2007. The english lexicon project. *Behavior research methods*, 39:445–459.

Marc Brysbaert and Boris New. 2009. Moving beyond kucera and francis: a critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english. *Behavior research methods*, 41:977–90.

Curt Burgess and Kay Livesay. 1998. The effect of corpus size in predicting reaction time in a basic word recognition task: Moving on from kučera and francis. *Behavior Research Methods, Instruments, & Computers*, 30:272–277.

Max Coltheart. 1981. The mrc psycholinguistic database. *The Quarterly Journal of Experimental Psychology*, 33(4):497–505.

Siobhan Devlin and John Tait. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic Databases*, pages 161–173.

Stefan Evert. 2010. Google web 1t 5-grams made easy (but not for the computer). In *Proceedings of the 2010 NAACL*, pages 32–40.

W Nelson Francis and Henry Kucera. 1979. Brown corpus manual. *Brown University*.

Olaf Hauk and F Pulvermüller. 2004. Effects of word length and frequency on the human event-related potential. *Clinical Neurophysiology*, 115:1090–1103.

Amaç Herdağdelen and Marco Marelli. 2016. Social media and language processing: How facebook and twitter provide the best frequency estimates for studying word recognition. *Cognitive Science*.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a Lexical Simplifier Using Wikipedia. In *Proceedings of the 52nd ACL*, pages 458–463.

S. Jauhar and L. Specia. 2012. Uow-shef: Simplex–lexical simplicity ranking based on contextual and psycholinguistic features. In *Proceedings of the 1st SemEval*, pages 477–481.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st ACL*, pages 1537–1546.

Pierre Lison and Jrg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the 10th LREC*.

Gustavo Henrique Paetzold and Lucia Specia. 2016a. Inferring psycholinguistic properties of words. In *Proceedings of the 2016 NAACL*, pages 435–440.

Gustavo Henrique Paetzold and Lucia Specia. 2016b. Unsupervised lexical simplification for non-native speakers. In *Proceedings of The 30th AAAI*.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of 2010 LREC*, volume 10, pages 1320–1326.

Keith Rayner and Susan A Duffy. 1986. Lexical complexity and fixation times in reading: Effects of word frequency, verb complexity, and lexical ambiguity. *Memory & Cognition*, 14(3):191–201.

Allan P. Rudell. 1993. Frequency of word usage and perceived word difficulty: Ratings of kuera and francis words. *Behavior Research Methods*, pages 455–463.

Matthew Shardlow. 2013. A comparison of techniques to automatically identify complex words. In *Proceedings of the 51st ACL Student Research Workshop*, pages 103–109.

Lucia Specia, Sujay Kumar Jauhar, and Rada Mihalcea. 2012. Semeval-2012 task 1: English lexical simplification. In *Proceedings of the 1st SemEval*, pages 347–355.

Andreas Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Proceedings of the 2002 ICSLP*, pages 257–286.

Walter JB Van Heuven, Pawel Mandera, Emmanuel Keuleers, and Marc Brysbaert. 2014. Subtlex-uk: A new and improved word frequency database for british english. *The Quarterly Journal of Experimental Psychology*, 67:1176–1190.

Fernando Cuetos Vega, María González Nosti, Analía Barbón Gutiérrez, and Marc Brysbaert. 2011. Subtlex-esp: Spanish word frequencies based on film subtitles. *Psicológica: Revista de metodología y psicología experimental*, 32:133–143.

Jason D Zevin and Mark S Seidenberg. 2002. Age of acquisition effects in word reading and other tasks. *Journal of Memory and language*, 47:1–29.

Using Argument Mining to Assess the Argumentation Quality of Essays

Henning Wachsmuth, Khalid Al-Khatib, and Benno Stein
Faculty of Media, Bauhaus-Universität Weimar, Germany
{henning.wachsmuth,khalid.alkhatib,benno.stein}@uni-weimar.de

Abstract

Argument mining aims to determine the argumentative structure of texts. Although it is said to be crucial for future applications such as writing support systems, the benefit of its output has rarely been evaluated. This paper puts the analysis of the output into the focus. In particular, we investigate to what extent the mined structure can be leveraged to assess the argumentation quality of persuasive essays. We find insightful statistical patterns in the structure of essays. From these, we derive novel features that we evaluate in four argumentation-related essay scoring tasks. Our results reveal the benefit of argument mining for assessing argumentation quality. Among others, we improve the state of the art in scoring an essay's organization and its argument strength.

1 Introduction

Argument mining aims to determine the argumentative structure of natural language texts. Usually, this structure is composed of different types of argumentative discourse units, such as premises and conclusions, that together form one or more arguments in favor of or against some thesis.

One of the main proposed downstream applications of argument mining is writing support including automated grading, which will extend the capabilities of massive open online courses (MOOCs), thereby contributing to unlimited access and participation in education. To aid argumentative writing, we envision a writing support system to proceed in three major steps: (1) The *mining* of argumentative structure, (2) the *assessment* of specific quality dimensions based on the mined structure, and (3) the *synthesis* of suggestions for quality improvements. Figure 1 visualizes the resulting process. Several approaches to the mining step have been developed and evaluated in terms of the effectiveness of the mined structure. So far, however, the benefit of this structure remains largely unexplored (see Section 2 for details).

This paper puts the assessment step into the focus. We ask if, to what extent, and how the output of argument mining can be leveraged to assess the argumentation quality of a text. In particular, we consider these questions for persuasive student essays. Such an essay seeks to justify a thesis on a given topic via a composition of arguments. Different quality dimensions related to argumentation have been studied for persuasive essays, such as the clarity of the justified thesis (Persing and Ng, 2013). Also, argument mining has already been performed effectively on persuasive essays (Stab and Gurevych, 2014b).

We build on the outlined research in that we use argument mining to assess an essay's argumentation quality. First, we adapt a state-of-the-art approach for mining argumentative discourse units (Section 3). Then, we apply the approach to all essays from the International Corpus of Learner English (Granger et al., 2009) in order to analyze their argumentative structure. We find statistically reliable patterns that yield insights into how students argue in essays. From these, we derive novel solely structure-oriented features for machine learning (Section 4). Finally, we tackle essay scoring for four argumentation-related quality dimensions: organization, thesis clarity, prompt adherence, and argument strength. In systematic experiments, we compare our features to strong baselines and to the state of the art (Section 5). The observed results provide clear evidence for the impact of argumentative structure on argumentation quality: Our features consistently do best among all structure-oriented approaches. Moreover, we outperform the state of the art of scoring the organization and the argument strength of persuasive essays.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1680–1691, Osaka, Japan, December 11-17 2016.

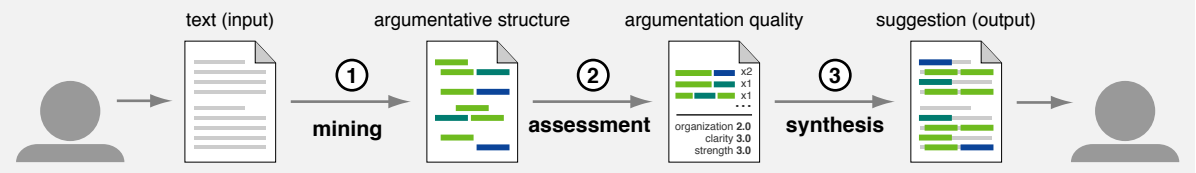


Figure 1: The three major steps of the envisioned process of writing support systems.

Contributions Altogether, with this paper we provide the following contributions to research:

1. We examine the use of argument mining for assessing argumentation quality for the first time.
2. We reveal common patterns in the argumentative structure of persuasive essays statistically.
3. We provide the new state of the art approach to two argumentation-related essay scoring tasks.

2 Related Work

Several approaches to argument mining have been introduced, often grounded in argumentation theory: Matching the argumentation schemes of Walton et al. (2008), Mochales and Moens (2011) model each argument in legal cases as a conclusion with a set of premises. Based on (Freeman, 2011), Peldszus and Stede (2015) capture support and attack relations between argumentative discourse units of microtexts. Habernal and Gurevych (2015) adapt the fine-grained argument model of Toulmin (1958) for web texts. As detailed in Section 3, we rely on the essay-oriented model of Stab and Gurevych (2014a). For us, mining is a preprocessing step only, though. For statistical reliability, we restrict our view to the units of arguments. Like Moens et al. (2007), we classify units on the sentence level, but we consider four different unit types. This results in a sequential structure comparable to argumentative zones (Teufel et al., 2009). The latter have also been exploited for downstream applications (Contractor et al., 2012).

Our focus is the *analysis* of argumentative structure. Related structures have been analyzed before: To measure text coherence, Feng et al. (2014) build on discourse structure (Mann and Thompson, 1988), which is connected but not equivalent to argumentative structure (Peldszus and Stede, 2013). Faulkner (2014) classifies the stance of essays using argument representations derived from dependency parse trees. For essay scoring, Persing et al. (2010) detect the discourse function of each paragraph in an essay in order to align the resulting function sequence with known function sequences. Similarly, we capture a review's overall structure in (Wachsmuth et al., 2014a) by comparing the local sentiment flow in the review to a set of common flow patterns that are learned through clustering. In (Wachsmuth et al., 2015), we further abstract the flows to optimize their domain generality in global sentiment analysis. Discourse structure, discourse functions, and sentiment flows serve as baselines in our experiments in Section 5. Unlike all mentioned approaches, however, we analyze the output of argument mining.

In particular, we use the mined structure to assess argumentation quality. While there is no common definition of such quality, Blair (2012) specifies the goals of relevance, acceptability, and sufficiency for arguments. To find accepted arguments in debate portals, Cabrio and Villata (2012) analyze attack relations between arguments based on the framework of Dung (1995). Rinott et al. (2015) detect three types of evidence in Wikipedia articles, and Boltužić and Šnajder (2015) seek for the prominent arguments in online debates. Here, we are not interested in the quality of single arguments but rather in the quality of a complete argumentation, namely, the argumentation found in a persuasive essay.

We target quality dimensions of persuasive essays that are directly related to argumentation: organization (Persing et al., 2010), thesis clarity (Persing and Ng, 2013), prompt adherence (Persing and Ng, 2014), and argument strength (Persing and Ng, 2015). In all four publications, sophisticated features are engineered to address a respective essay scoring task. The argument strength approach adopts ideas from the approach of Stab and Gurevych (2014b), but it finds structure heuristically only and, thus, does not perform argument mining. In the paper at hand, we fill this gap, i.e., we exploit the output of an argument mining approach trained on ground-truth data to assess the four quality dimensions.

In general, numerous approaches exist that assess essay quality. Classical essay scoring often focuses on grammar, vocabulary, and similar (Dikli, 2006), partly employing structural features like discourse markers (Burstein et al., 1998). In contrast, Song et al. (2014) study whether essays comply with critical

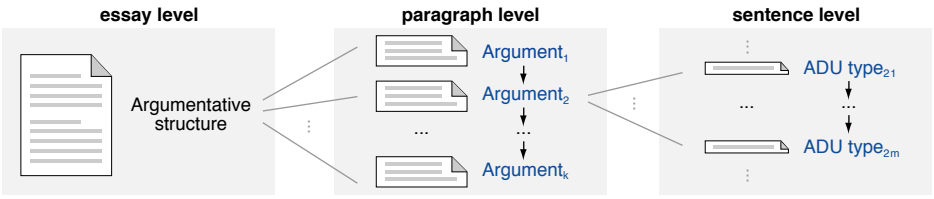


Figure 2: Application-oriented model of the argumentative structure of essays. Each paragraph is seen as an argument, defined as a sequence of sentence-level ADU types $\in \{$*Thesis, Conclusion, Premise, None*$\}$.

questions of an applied argumentation scheme. On manual annotations, they find correlations between an essay's score and the number of answered questions. Closer to our work, Ong et al. (2014) analyze argumentative discourse units found with a simple heuristic algorithm. And Ghosh et al. (2016) even derive features from argument mining, although they hardly exploit structure. Either way, all these approaches assign overall essay scores only, leaving unclear to what extent argumentation quality is captured.

3 Mining Argumentative Structure

This paper does *not* aim at new approaches to argument mining. Still, the effectiveness of mining as well as the underlying argumentation model directly affect the analysis of argumentative structure. Therefore, we summarize our mining approach in the following.[1]

3.1 An Application-Oriented Model of Argumentative Structure

We focus on the argumentative structures of persuasive student essays. Such an essay states and justifies a thesis on some topic that is introduced by a given prompt. To capture an essay's structure, we build on the work of Stab and Gurevych (2014a) who presented both an argumentation model for persuasive essays and an annotated corpus. By training a mining approach on this corpus, we expect to minimize the usual out-of-domain effectiveness drop (Blitzer et al., 2008), when using the approach on other essays.

Stab and Gurevych (2014a) distinguish four types of argumentative discourse units (called ADUs from here on) within essays: *Thesis*, *Conclusion*, *Premise*, and *None*.[2] The authors define an ADU loosely as a statement covering an entire sentence or less. Each conclusion in an essay supports or attacks a thesis, and each premise supports or attacks a thesis, conclusion, or other premise. Implicitly, these relations specify the essay's arguments. In their corpus, less than 15% of all relations are attacks.

For our purposes, we simplify the model of Stab and Gurevych (2014b) in two respects: (1) We define each sentence in an essay to correspond to exactly one ADU. Thereby, we avoid the need to segment essays into ADUs.[3] (2) We define each paragraph in an essay to correspond to exactly one argument. Thereby, we avoid the need to identify relations between ADUs. As a result, we represent the argumentative structure of an essay as a sequence of arguments and each argument as a sequence of ADU types. Figure 2 sketches this application-oriented model.

The justification for our simplification is twofold: (1) We aim to capture argumentative structure only on an abstraction level that allows assessing argumentation quality. Abstraction reduces the search space of argument structures to explore, which benefits pattern recognition, but it also takes away information. While the right level is unknown, we hypothesize that students largely organize essays sequentially. This is in line with our previous research (Wachsmuth et al., 2015). (2) For successful pattern recognition, we need to mine argumentative structure effectively. Therefore, we omit potentially helpful structure such as attack relations, as all available data seems insufficient for reliably training respective approaches.

3.2 Approach

For tokenization, sentence splitting, and paragraph splitting, we apply our own algorithms from previous work (Wachsmuth, 2015), while we use the TreeTagger for part-of-speech tagging (Schmid, 1995). Given the sentences and paragraphs of an essay, our model then requires only to classify the ADU type of

[1] The source code for reproducing all experiments from Sections 3 to 5 can be found here: http://www.arguana.com/software

[2] Stab and Gurevych (2014a) use other names for the ADU types than we do, such as *Major claim* instead of *Thesis*.

[3] The approach proposed by Stab and Gurevych (2014b) also does not deal with the segmentation of an essay into ADUs, but merely because it classifies ADUs simply based on the ground-truth segmentation.

ADU Type	Training	Test	Total	AAE Total
Thesis	72	18	90	90
Conclusion	325	93	418	429
Premise	652	181	833	1033
None	185	53	238	327
All types	1234	345	1579	1879

Table 1: Distribution of ADU type annotations in the modified dataset. Notice that the difference to the distribution in the original Argument Annotated Essays (AAE) corpus is moderate only.

#	Feature Type	Accuracy	F_1-score
1	Prompt similarity	44.9	41.8
2	Token n-grams	47.8	48.0
3	POS n-grams	41.2	43.5
4	General Inquirer classes	42.3	44.5
5	1st token n-grams	33.6	35.0
6	Sentence position	64.9	66.9
1–6	**Complete feature set**	74.5	**74.5**
	Majority baseline	52.5	36.1
	Stab and Gurevych (2014b)	**77.3**	72.6

Table 2: Effectiveness of our features in classifying ADU types compared to (Stab and Gurevych, 2014b).

each sentence. As Stab and Gurevych (2014b), we tackle this 4-class classification task with supervised machine learning. We employ six feature types that capture the content, style, and position of a sentence:[4]

Prompt Similarity The cosine, Euclidean, Manhattan, and Jaccard similarity of the sentence to the prompt of the given essay, once for all words and once for all non-function words.

Token n-Grams The frequency of each token 1- to 3-gram occurring in $\geq 1\%$ of the training sentences.

POS n-Grams The frequency of each part-of-speech 1- to 3-gram occurring in $\geq 5\%$ of these sentences.

General Inquirer Classes The frequency of each word class specified by the General Inquirer.[5]

1st Token n-Grams Indicators whether the first token 1-, 2-, and 3-gram of the sentence match those 1-, 2-, and 3-grams that are first in $\geq 0.5\%$ of all training ADUs.

Sentence Position Indicators whether a sentence is the first, second, or last within a paragraph and what its relative position is. The same for the sentence and the covering paragraph within the complete essay.

3.3 Experimental Set-up

We evaluated our approach to classify all ADU types in a persuasive essay based on the following set-up:

Data As indicated, we processed the Argument Annotated Essays (AAE) corpus of Stab and Gurevych (2014a), containing 90 persuasive student essays (72 for training, 18 for testing). In each essay, all theses, conclusions, and premises are annotated as ADUs of the respective types. Since we do not tackle ADU segmentation, we enlarged the annotations to span the whole covering sentence. If a sentence contained more than one ADU, we favored rarer classes to benefit training, i.e., we preferred *Thesis* over *Conclusion* over *Premise*. All unannotated sentences from an essay's body were assigned the type *None*. Unlike Stab and Gurevych (2014b), we ignored the titles of the 90 essays as *None* instances; classifying a title based on its position is trivial, but it causes errors on essays without titles. Table 1 compares the numbers of annotations in our modified dataset to those of the original AAE corpus. Besides the ignored titles, the two resources differ considerably only in the number of premises.

Experiments For supervised learning, we used the default configuration of the SMO classifier in Weka 3.7 (Hall et al., 2009). We turned off its feature normalization, though, because we generally normalize all our feature values to the range $[0, 1]$. On the training set of the derived dataset, we trained one classifier for each single feature type and for the complete feature set. We did not optimize any hyperparameters but simply measured the accuracy and weighted average F_1-score of the default SMO on the test set.

Comparison As a rough estimate, we compare the results of our approach to those of Stab and Gurevych (2014b) on the AAE corpus. While the comparability is only limited due to the slightly modified corpus, we do not primarily aim to outperform existing mining approaches but rather to imitate them. In order to ease the global interpretation of our results, we also report on the *majority baseline*.

3.4 Results

Table 2 presents the classification effectiveness of each evaluated feature type. The sentence position features dominate all other types with an accuracy of 64.9 and an F_1-score of 66.9. Still, the others add

[4]The strongest type in (Stab and Gurevych, 2014b) uses the length of an ADU as well as the tokens in its covering sentence. As we classify complete sentences, these features help less here.

[5]For more information on the General Inquirer classes, see http://www.wjh.harvard.edu/∼inquirer/.

Paragraph	Premise: *Secondly, most violent crimes are related to the abuse of guns, especially in some countries where guns are available for people.* Conclusion: *Eventually, guns will create a violent society if the trend continues.* Premise: *Take an example, in American, young adults and even juveniles can get access to guns, which leads to the tragedies of school gun shooting.* Premise: *What is worse, some terrorists are able to possess more advanced weapons than the police, which makes citizens always live in danger.*
ADU flow	(1x Premise, 1x Conclusion, 2x Premise)
ADU change flow	(Premise, Conclusion, Premise)

Figure 3: The ADU flow and the ADU change flow for one paragraph of the AAE corpus (see Section 3).

to the effectiveness of the complete feature set. The complete feature set performs a little worse than Stab and Gurevych (2014b) in terms of accuracy (74.5 vs. 77.3) but better in terms of F_1-score (74.5 vs. 72.6). Thus, we conclude that our mining approach is at eye level with (Stab and Gurevych, 2014b). Moreover, our results appear reasonable within a 4-class classification task. We will see whether they suffice to recognize discriminative argumentative structures and to leverage them for quality assessment.

4 Analyzing Argumentative Structure

This section analyzes the output of our mining approach to find statistically reliable patterns in the argumentative structure of persuasive essays. From these, novel features for machine learning are derived.

4.1 Statistically Reliable Patterns of Argumentative Structure

A persuasive essay is meant to compose a set of arguments in favor of or against a thesis, each combining a set of premises with a conclusion (Stab and Gurevych, 2014a). Such a tree-like structure allows for much variance, rendering a reliable pattern recognition hard. Above, we have hypothesized that essays largely argue sequentially. Given the model from Section 3, we hence restrict our view to the sequences of types of argumentative discourse units (ADUs) in essays. In accordance with our work on sentiment flows from (Wachsmuth et al., 2014b), we look at two kinds of patterns, both exemplified in Figure 3:

ADU Flow The sequence of all ADU types within one paragraph on an essay.

ADU Change Flow The sequence of all different ADU types within one paragraph on an essay.

4.2 Experimental Set-up

To get reliable insights into the structure of persuasive essays, we performed a straightforward analysis:

Data We took the International Corpus of Learner English (ICLE, version 2), containing 6085 English essays from students of 16 mother tongues (Granger et al., 2009). On average, an ICLE essay spans 7.6 paragraphs (standard deviation $\pm$ 5.2) and 33.8 sentences ($\pm$ 16.5) according to our preprocessing.

Experiments We applied the mining approach from Section 3 to all ICLE essays. Then, we computed the relative frequencies of all ADU flows and ADU change flows. In addition, we tested how much these frequencies differ within an essay's first and last paragraph.

4.3 Results

The top part of Table 3 lists the ten most frequent of the 2593 ADU flows found in the ICLE corpus. They cover about half of all paragraphs. The first two ADU flows consist of conclusions only, whereas the others show "real" argumentative structure. After a conclusion, two premises follow most often (5.4%). Still, the number of premises varies, bringing up the question whether a particular number benefits argumentation quality. Patterns such as *(1x Conclusion, 2x Premise, 1x Conclusion)* may refer to restated conclusions but also to paragraphs that combine two arguments.

Overall, we see that all top ten ADU flows begin with a conclusion, i.e., our analysis reveals that students tend to (or are taught to) first state a claim and then argue for it. This is also supported by the top ten ADU change flows in Table 4: Every fourth paragraph matches the pattern *(Conclusion, Premise)*, while only 2.9% order all premises first. Similarly, *None* serves for beginning a paragraph, while theses rather appear at the end. In total, the abstraction of ADU change flows seems to capture much diversity of arguments in persuasive essays: Together, the top ten represent 87.4% of all ICLE paragraphs, and all ADU types occur in at least one combination. Still, we found 319 different ADU change flows.

Both Table 3 and 4 highlight the special roles of the first and last paragraph of an essay, which clearly deviate from the average: The first is mostly made up of *None* and *Thesis*, underlining its introductory nature. In contrast, the last often ends with a conclusion—making the argumentation's final point.

¶'s	#	ADU Flow	Frequency
all	1	(1x Conclusion)	14.5%
	2	(2x Conclusion)	7.1%
	3	(1x Conclusion, 2x Premise)	5.4%
	4	(1x Conclusion, 1x Premise)	4.9%
	5	(1x Conclusion, 3x Premise)	4.2%
	6	(1x Conclusion, 1x Premise, 1x Conclusion)	4.2%
	7	(1x Conclusion, 2x Premise, 1x Conclusion)	3.4%
	8	(1x Conclusion, 4x Premise)	3.0%
	9	(1x Conclusion, 3x Premise, 1x Conclusion)	2.3%
	10	(1x Conclusion, 5x Premise)	2.0%
1st	1	(2x None)	9.7%
	2	(3x None)	8.6%
	3	(2x None, 1x Thesis)	6.2%
	4	(4x None)	6.2%
	5	(3x None, 1x Thesis)	5.7%
last	1	(1x Conclusion)	16.9%
	2	(2x Conclusion)	12.6%
	3	(1x Conclusion, 1x Premise, 1x Conclusion)	8.2%
	4	(1x Conclusion, 2x Premise, 1x Conclusion)	5.7%
	5	(1x Conclusion, 3x Premise, 1x Conclusion)	3.4%

Table 3: The most frequent ADU flows in *all* ICLE paragraphs as well as in the *1st* and *last* paragraphs.

¶'s	#	ADU Change Flow	Frequency
all	1	(Conclusion, Premise)	25.1%
	2	(Conclusion)	22.4%
	3	(Conclusion, Premise, Conclusion)	17.0%
	4	(None)	5.8%
	5	(Premise)	4.3%
	6	(None, Thesis)	3.4%
	7	(Premise, Conclusion)	2.9%
	8	(None, Premise)	2.7%
	9	(Conclusion, Premise, Conclusion, Premise)	2.0%
	10	(None, Premise, Conclusion)	1.8%
1st	1	(None)	42.7%
	2	(None, Thesis)	25.9%
	3	(Thesis)	5.7%
	4	(None, Premise, None)	4.4%
	5	(None, Conclusion)	4.3%
last	1	(Conclusion)	31.6%
	2	(Conclusion, Premise, Conclusion)	27.2%
	3	(Conclusion, Premise)	13.1%
	4	(None, Premise, Conclusion)	4.4%
	5	(Premise, Conclusion)	2.7%

Table 4: The most frequent ADU change flows in the ICLE paragraphs (ignoring type repititions).

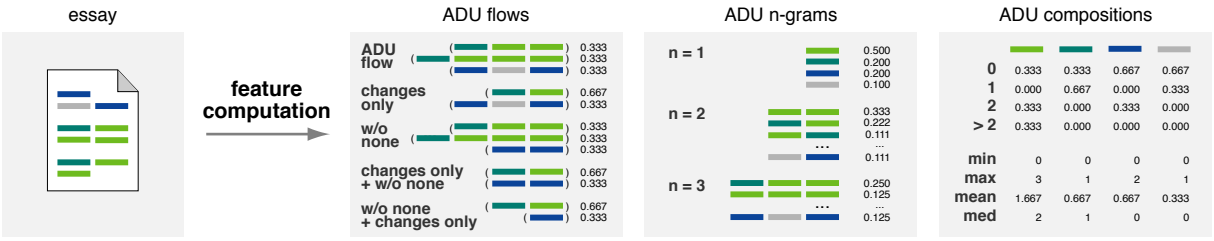


Figure 4: Sketch of the three feature types that we propose based on the output of argument mining.

4.4 Shallow Features for Statistical Significance

The found patterns suggest that persuasive student essays differ in the combination, ordering, and number of ADU types. For the assessment of argumentation quality, we capture these structural variations in the following three novel feature types. The types are kept shallow in order to benefit statistical significance:

ADU Flows The frequencies of all ADU flows in an essay. The hypothesis is that certain flows are favorable. We also examine two flow abstractions: (1) considering changes only, as above, and (2) ignoring the non-argumentative type *None*. Arranging 0 to 2 of these abstractions allows for five flow variations.

ADU n-Grams The frequencies of all ADU type n-grams in an essay for some $n \geq 1$. The hypothesis is that certain combinations of ADU types are favorable.

ADU Compositions The proportions of paragraphs in an essay with a particular number of occurrences of a particular ADU type as well as summary statistics about each type (such as the minimum or mean). The hypothesis is that certain numbers of certain ADU types are favorable.

Figure 4 illustrates the computation of feature values of each type for a sample essay with three paragraphs. The exact feature type configuration that we used in our experiments is specified in Section 5.

5 Assessing Argumentation Quality

Finally, we analyze the benefit of mining argumentative structure for assessing argumentation quality. In particular, we evaluate the three presented feature types in argumentation-related essay scoring.

5.1 Essay Scoring Tasks

We consider four essay scoring tasks that were introduced in successive papers, each of which capturing a particular dimension of argumentation quality. These tasks can be summarized as follows:

Organization Score the quality of an essay's organization. A high score is assigned to essays, which introduce their topic, take and argue for a position on the topic, and conclude (Persing et al., 2010).

Thesis Clarity Score the clarity of the explanation of the thesis that an essay argues for. A high score is assigned to essays, which make their thesis easy to understand (Persing and Ng, 2013).

Prompt Adherence Score the adherence of an essay's content to the essay's prompt. A high score is assigned to essays that consistently remain on the topic of the prompt (Persing and Ng, 2014).

Argument Strength Score the strength of the argument that an essay makes for its thesis. A high score is assigned to essays, which would convince most readers of their thesis (Persing and Ng, 2015).

Our proposed feature types solely focus on the argumentative structure of an essay—as opposed to the essay's content or linguistic style. Accordingly, we hypothesize that the feature types are particularly successful in the organization task. To a minor extent, we expect that they also help for argument strength, because argument strength should emerge from all aspects of an essay. In contrast, the scoring of thesis clarity and prompt adherence rather seems to require an analysis of content and style respectively.

5.2 Approach

Analogue to the authors of the four mentioned papers, we tackle essay scoring with supervised regression. For this purpose, we consider our proposed feature types as well as several baseline features:

ADU Features (a) In terms of the feature types from Section 4, we rely on the following configurations:

- a_1 *ADU flows*. The frequency of each ADU flow that occurs in $\geq 1\%$ of all training essays. All five flow variations described in Section 4 are taken into account.
- a_2 *ADU n-grams*. The frequency of each ADU 1-, 2-, and 3-gram that occurs in $\geq 5\%$ of all training essays. + Indicators that capture the first and the last ADU 1-, 2-, and 3-gram.
- a_3 *ADU compositions*. The percentages of paragraphs with $\{0 \mid 1 \mid 2 \mid >2\}$ occurrences of the type $\{$*Thesis* | *Conclusion* | *Premise* | *None*$\}$. + The {minimum | maximum | mean | median} of each of these ADU types per paragraph. + The percentage of each type in the first and in the last paragraph.

Flow Features (b) Persing et al. (2010) aligned sequences of four paragraph discourse functions: *Body* (own argument), *Rebuttal*, *Introduction*, and *Conclusion*. Since we cannot access their original approach, we approximate it—and also add further strong structure-oriented baseline approaches: In (Wachsmuth et al., 2014a) and (Wachsmuth et al., 2015), we captured the overall structure of a review by comparing the review's sentiment flow to a set of common flow patterns and flow abstractions. Both the patterns and the abstractions were found in a training set before. To model the paragraph-level argumentative structure of persuasive essays, we adapt these patterns and abstractions in the following features:

- b_1 *Function flows*. All flow features defined in (Wachsmuth et al., 2014a) and (Wachsmuth et al., 2015) based on paragraph discourse functions. Functions are found with the heuristic algorithm of Persing et al. (2010). *Body* is mapped to 1.0, *Rebuttal* to 0.0, and the remaining two functions to 0.5, in order to allow for numerical comparison between the flows.
- b_2 *Sentiment flows*. All flow features based on paragraph-level sentiment. A paragraph is assigned the numerical sentiment value 1.0 (0.0), if it contains a positive (negative) but no negative (positive) sentence, otherwise 0.5. We find sentence sentiment with the algorithm of Socher et al. (2013).
- b_3 *Relation flows*. All flow features based on sentence-level discourse relations. Ten relation types from (Mann and Thompson, 1988) are found with our rule-based algorithm (Wachsmuth et al., 2014a). For lack of an adequate mapping, we compare relation flows based on nominal differences only.

Standard Features (c) In order to be able to assess the impact of argumentative structure, we compare all structure-oriented features to two standard types of content and style features:

- c_1 *Content*. The frequency of each token 1-, 2-, and 3-gram that occurs in $\geq 10\%$ of all training essays. + The minimum, maximum, and average prompt similarity (see Section 3.2) over all sentences.
- c_2 *POS n-grams*. The frequency of each part-of-speech 1-, 2-, and 3-gram that occurs in $\geq 10\%$, $\geq 20\%$, and $\geq 40\%$ of all training essays respectively.

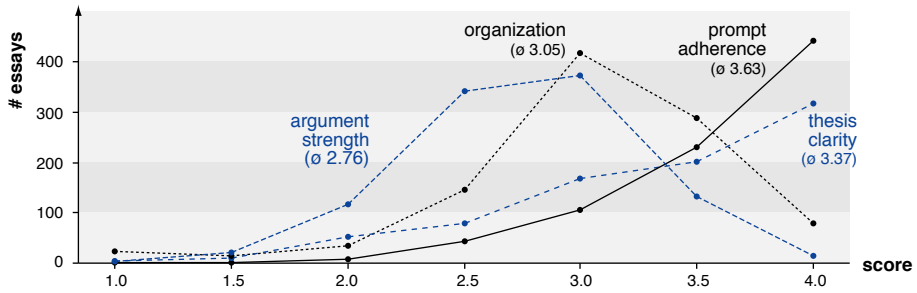


Figure 5: Distribution of essays over the possible scores from [1.0, 4.0] in the datasets of the four tasks.

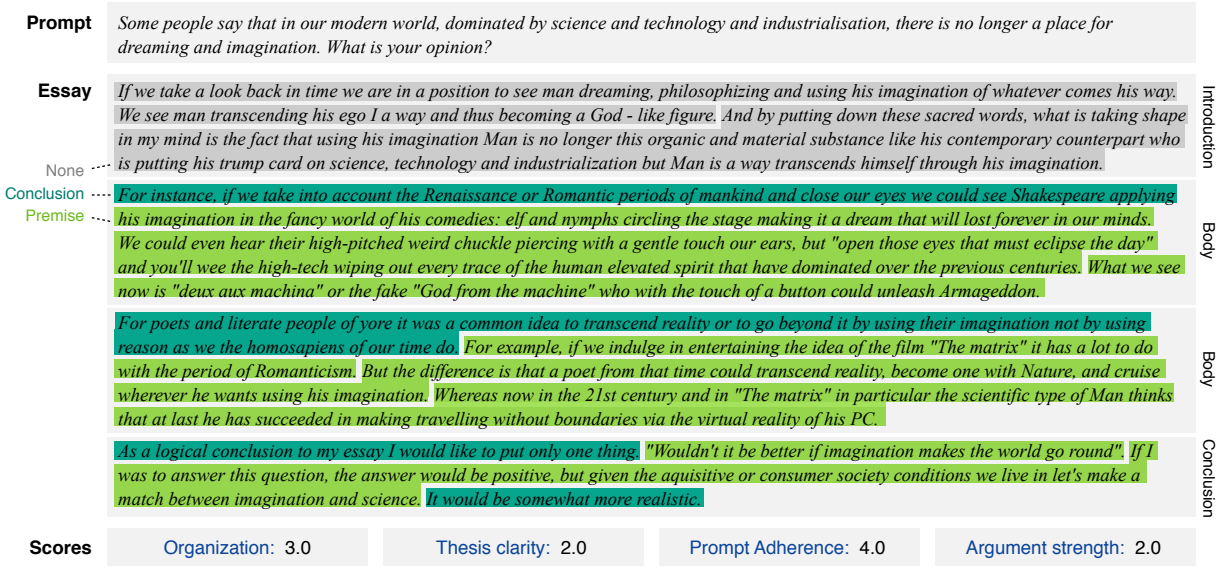


Figure 6: One essay from the four datasets together with its manually assigned scores as well as the ADU types (colored background) and discourse functions (vertical) automatically annotated by our algorithms.

5.3 Experimental Set-up

For direct comparison, we replicated the original experimental set-up of the authors of the aforementioned papers on the datasets they provide for the four essay scoring tasks:

Data For each task, one distinct subset of the ICLE corpus (see Section 4) is manually annotated with half-point scores between 1.0 (worst) and 4.0 (best). These datasets cover 1003 (organization), 830 (thesis clarity, prompt adherence), and 1000 essays (argument strength) respectively. For a rough overview, Figure 5 plots the numbers of scores in each dataset, indicating that only the organization and argument strength scores are Gaussian-like distributed. Exact numbers are found in the original papers. Figure 6 shows one essay included in all datasets with its scores and the annotations created by our algorithms.

Experiments We used linear ε-SVR support vector machine regression from LibSVM in Weka 3.7 (Hall et al., 2009; Chang and Lin, 2011).[6] Each dataset has five predefined folds. As in the original set-up, we performed cross-validation on these folds, training one LibSVM for each feature type and for different type combinations. Accordingly, we then also measured the mean absolute error (MAE) and the mean squared error (MSE) of regression.[7] Different from the original set-up, we omitted a real optimization of the LibSVM cost hyperparameter, but we simply set it permanently to 0.1 after a few initial tests.

Comparison We compare the proposed feature types to the described baseline features and to two general baselines: (**d**) The *average baseline*, which assigns the mean score of the training essays to all test essays. Although trivial, **d** is quite strong under given the score distributions in Figure 5. (**e**) The lowest MAE and MSE values reported by the authors of the four tasks, called *Persing et al. best* below. To our knowledge, these results have not been beaten so far and, thus, define the state of the art until now.

[6] Persing and Ng (2014; 2015) relied on LibSVM, too. In the other two papers, SVM^{light} was used (Joachims, 1999).

[7] From the practical viewpoint of applying automatic essay scoring in MOOCs or similar, the most important requirement is to avoid outliers (in terms of utterly wrong scores) as far as possible. In this regard, the MSE is the more meaningful measure.

#	Feature Type	Organization MAE	Organization MSE	Thesis Clarity MAE	Thesis Clarity MSE	Prompt Adherence MAE	Prompt Adherence MSE	Argument Strength MAE	Argument Strength MSE
a_1	ADU flows	0.367 ±.022	0.234 ±.017	0.530 ±.030	0.461 ±.061	0.373 ±.033	0.247 ±.056	0.399 ±.010	**0.242** ±.018
a_2	ADU n-grams	0.369 ±.024	0.225 ±.031	0.530 ±.032	0.466 ±.065	0.372 ±.031	0.265 ±.053	0.398 ±.012	**0.243** ±.017
a_3	ADU compositions	0.347 ±.026	0.194 ±.020	0.529 ±.034	0.457 ±.062	0.365 ±.032	0.239 ±.046	**0.390** ±.015	**0.239** ±.023
a	**ADU features**	0.336 ±.022	0.184 ±.020	0.537 ±.031	0.470 ±.056	0.368 ±.029	0.241 ±.044	0.392 ±.016	**0.242** ±.023
b_1	Function flows	0.368 ±.025	0.220 ±.037	0.541 ±.032	0.478 ±.063	0.370 ±.029	0.255 ±.042	0.403 ±.006	0.251 ±.011
b_2	Sentiment flows	0.369 ±.014	0.228 ±.022	0.536 ±.031	0.481 ±.064	0.372 ±.035	0.257 ±.053	0.410 ±.009	0.259 ±.013
b_3	Relation flows	0.426 ±.013	0.351 ±.033	0.541 ±.052	0.475 ±.074	0.368 ±.027	0.255 ±.048	0.408 ±.012	0.260 ±.014
b	**Flow features**	0.355 ±.023	0.207 ±.026	0.559 ±.040	0.512 ±.059	0.377 ±.020	0.255 ±.038	0.415 ±.009	0.269 ±.013
c_1	Content	0.415 ±.013	0.336 ±.025	0.501 ±.033	0.425 ±.064	0.362 ±.026	0.231 ±.046	0.395 ±.011	**0.236** ±.018
c_2	POS n-grams	0.407 ±.013	0.326 ±.032	0.528 ±.036	0.461 ±.063	0.361 ±.034	0.231 ±.048	**0.387** ±.014	**0.233** ±.019
c	**Standard features**	0.408 ±.013	0.324 ±.027	0.505 ±.036	0.429 ±.065	0.357 ±.026	0.222 ±.040	**0.384** ±.015	**0.230** ±.020
	a + b_1	**0.321** ±.020	**0.171** ±.019	0.551 ±.036	0.494 ±.064	0.372 ±.025	0.245 ±.038	0.400 ±.011	0.250 ±.019
	a + b_2	0.328 ±.016	**0.174** ±.016	0.546 ±.024	0.495 ±.049	0.374 ±.026	0.244 ±.043	0.402 ±.013	0.253 ±.018
	a + b_3	0.341 ±.019	0.189 ±.018	0.544 ±.035	0.482 ±.067	0.367 ±.027	0.240 ±.043	0.405 ±.022	0.254 ±.022
	a + **b**	0.329 ±.018	0.179 ±.020	0.565 ±.026	0.484 ±.057	0.379 ±.021	0.250 ±.037	0.421 ±.016	0.275 ±.016
	a + b_1 + c_1	**0.315** ±.018	**0.168** ±.020	0.524 ±.031	0.456 ±.057	0.361 ±.021	0.227 ±.033	**0.391** ±.010	**0.242** ±.019
	a + b_1 + c_2	**0.315** ±.017	**0.167** ±.017	0.546 ±.031	0.492 ±.056	0.362 ±.025	0.229 ±.033	**0.389** ±.011	**0.241** ±.020
*	**a** + b_1 + **c**	**0.314** ±.018	**0.167** ±.018	0.520 ±.032	0.450 ±.056	0.362 ±.021	0.226 ±.030	**0.387** ±.012	**0.238** ±.021
	a + b_2 + c_1	**0.320** ±.016	**0.167** ±.016	0.520 ±.023	0.457 ±.049	0.367 ±.024	0.230 ±.039	0.395 ±.013	0.247 ±.019
*	**a** + b_2 + c_2	**0.316** ±.015	**0.164** ±.013	0.548 ±.020	0.496 ±.046	0.364 ±.029	0.232 ±.041	0.393 ±.012	0.246 ±.019
	a + b_2 + **c**	**0.320** ±.014	**0.167** ±.014	0.520 ±.025	0.456 ±.051	0.363 ±.026	0.228 ±.037	0.392 ±.011	**0.243** ±.020
	a + b_3 + c_1	0.333 ±.020	0.182 ±.017	0.520 ±.031	0.443 ±.060	0.360 ±.023	0.228 ±.037	0.397 ±.014	0.247 ±.023
	a + b_3 + c_2	0.326 ±.018	0.176 ±.014	0.537 ±.030	0.477 ±.057	0.363 ±.027	0.232 ±.039	0.393 ±.014	0.244 ±.024
	a + b_3 + **c**	0.328 ±.017	0.177 ±.014	0.515 ±.032	0.441 ±.054	0.359 ±.023	0.226 ±.037	0.390 ±.014	**0.243** ±.024
	a + **b** + c_1	**0.321** ±.021	**0.172** ±.018	0.543 ±.029	0.484 ±.055	0.376 ±.018	0.242 ±.033	0.413 ±.008	0.265 ±.014
	a + **b** + c_2	**0.315** ±.019	**0.169** ±.016	0.557 ±.024	0.512 ±.049	0.375 ±.021	0.243 ±.033	0.411 ±.008	0.262 ±.014
	a + **b** + **c**	**0.315** ±.018	**0.169** ±.015	0.543 ±.029	0.486 ±.054	0.375 ±.017	0.241 ±.032	0.409 ±.008	0.259 ±.014
	a_3 + c_1	0.335 ±.022	0.182 ±.018	0.505 ±.036	0.431 ±.061	0.356 ±.029	0.221 ±.043	**0.383** ±.016	**0.230** ±.024
**	a_3 + c_2	0.329 ±.018	0.177 ±.013	0.529 ±.033	0.463 ±.062	0.354 ±.033	0.221 ±.042	**0.380** ±.016	**0.229** ±.024
**	a_3 + **c**	0.330 ±.017	0.178 ±.014	0.508 ±.036	0.435 ±.061	0.352 ±.027	0.216 ±.038	**0.378** ±.017	**0.226** ±.025
	b + c_1	0.344 ±.024	0.196 ±.026	0.531 ±.014	0.464 ±.066	0.372 ±.015	0.242 ±.015	0.406 ±.012	0.257 ±.015
	b + c_2	0.336 ±.018	0.189 ±.022	0.552 ±.013	0.450 ±.056	0.369 ±.022	0.242 ±.017	0.404 ±.015	0.255 ±.015
	b + **c**	0.336 ±.018	0.189 ±.021	0.531 ±.015	0.465 ±.064	0.368 ±.017	0.237 ±.016	0.400 ±.015	0.250 ±.015
d	**Average baseline**	0.425 ±.016	0.349 ±.030	0.545 ±.036	0.469 ±.084	0.370 ±.038	0.291 ±.055	0.407 ±.014	0.266 ±.018
e	**Persing et al. best**	0.323	0.175	0.483	0.369	0.348	0.197	0.392	0.244

Table 5: The mean average error (MAE) and the mean squared error (MSE) ± their standard deviations for each evaluated feature type and type combination in the four essay scoring tasks. All values marked in bold outperform the former state of the art (called *Persing et al. best* here). The most significant results for organization and argument strength each are marked with * and ** respectively.

5.4 Results

Table 5 lists the two mean regression errors and their standard deviations for each single feature type in isolation and for a selection of feature type combinations, averaged over the respective five folds for each of the four evaluated essay scoring tasks.[8]

With respect to the single feature types in the upper part, we see that the ADU compositions (a_3) consistently perform best among all structure-oriented features (a_1–a_3, b_1–b_3) in all four tasks. The ADU flows (a_1) and ADU n-grams (a_2) behave comparable to the function flows (b_1) and sentiment flows (b_2). The relation flows (b_3) compete with most others only in scoring prompt adherence, possibly suggesting that the impact of discourse structure on argumentation quality is limited.[9]

Matching our hypothesis from Section 5.1, the structure-oriented features clearly outperform the standard features (**c**) in scoring *organization*, as shown in the left part of Table 5. In fact, c_1 and c_2 hardly improve over the average baseline (**d**). In isolation, the complete ADU feature set (**a**) produces the lowest errors; an MAE of 0.336 and an MSE of 0.184. Since the quality of the organization of an essay naturally depends on the essay's argumentative structure, this result underpins the adequacy of our features. Moreover, **a** outperforms the best values we measured without using ADU features (those of **b** + **c**). Combined with function flows (**a** + b_1) or with sentiment flows (**a** + b_2), the ADU features already beat Persing et

[8] We did not perform explicit feature ablation tests, because they would provide limited insights only: As we did not optimize the LibSVM cost hyperparameter, leaving out one feature type does not necessarily lead to an increase of the regression errors.

[9] One reason for the low impact of b_3 lies in the varying number of sentences of essays (see Section 4), which impairs flow pattern recognition. A detailed analysis of the use of discourse relations for quality assessment is out of the scope of this paper.

al. best (**e**). The smallest MSE is achieved by the ADU features with sentiment flows and POS n-grams ($\mathbf{a} + b_2 + c_2$). According to a one-sided student t-test, the value 0.164 is significant at $p < 0.1$.

As expected, all structure-oriented features fail in case of *thesis clarity*, being only slightly better than the average baseline (**d**) if at all. The lowest errors (MAE 0.501, MAE 0.425) are observed for the content features (c_1). Still, c_1 cannot compete with **e**—the respective approach of Persing and Ng (2013) employs keyword features that were manually derived from the prompts of all essays.

For *prompt adherence*, at least the errors produced by the ADU compositions (a_3) are close to those of c_1 and c_2, which is why we additionally tested a_3 in combination with the standard feature types. As shown in the bottom part of Table 5, $a_3 + \mathbf{c}$ performs best with an MAE of 0.352 and an MSE of 0.216. These values are not significantly worse than the state of the art (**e**).

$a_3 + \mathbf{c}$ also minimizes the errors in scoring *argument strength*. Both the MAE of 0.378 and the MSE of 0.226 are significantly better than Persing et al. best (**e**) at $p < 0.1$. Again, this observation supports our hypothesis: The strength of an essay's argumentation will hardly ever be independent from the essay's content, but it still benefits from a good argumentative structure. Interestingly, even a_3 alone improves over **e** with an MAE of 0.390 (vs. 0.392) and an MSE of 0.239 (vs. 0.244).

We conclude that our approach denotes the new state of the art for two essay scoring tasks. Under the assumption that the manual score annotations in the processed datasets are adequate, our hypothesis that the benefit of argument mining is high for scoring an essay's organization turns out true. Compared to the findings of Persing et al. (2010), the obtained results thereby reveal that organization is not only about the ordering of discourse functions, but also about argumentative structure. In particular, the novel features that we proposed capture only such structural aspects. Accordingly, their impact is low for thesis clarity and also only fair for prompt adherence, underlining that these tasks are rather related to the content and style of an argumentation. In contrast, argument strength brings together structure and content, and this is indeed reflected by the moderate but significant benefit of our structure-oriented features there.

6 Conclusion

Although argument mining has become a hot topic, the question of what practical benefits it provides for applications has hardly been examined yet. In the paper at hand, we have approached this question for a specific but important task, namely, we have used argument mining to assess argumentation quality. Our results for persuasive student essays underpin the benefit of argument mining, revealing that the mined argumentative structure is particularly helpful for structure-related quality dimensions: Without putting emphasis on the content of arguments, we have improved the state of the art in scoring an essay's organization and even in scoring its argument strength. Our best-performing features capture the composition of types of units in arguments (such as premises and conclusions).

Similar to existing approaches, the mining algorithm we trained and applied in this paper misclassifies about one out of four units. So far, we could not analyze the impact of mining errors on the effectiveness of our essay scoring approaches, since ground-truth data is needed before that brings together argumentative structure and argumentation quality. A question that remains open in this regard is what model of argumentative structure proves most suitable. As adequate training data is still limited, we have modeled shallow unit types only, but we expect that considering attack and support relations, evidence types, or argumentation schemes will prove useful for quality assessment.

Naturally, other quality dimensions of argumentation will depend more on content, so our analysis of argumentative structure does not solve the assessment of argumentation quality in general. Also, essay structure is quite conventionalized, i.e., a transfer of our findings to other argumentative text genres requires further investigation. We plan to continue our research in this regard based on our new corpus for the analysis of argumentation strategies in news editorials (Al-Khatib et al., 2016).

In practice, our approach in its given form most notably contributes to educational applications that analyze argumentative texts, such as automatic grading and writing support systems. These systems need not only mine argumentative structure, but also evaluate the mined structure.[10] To support argumentation quality, another step is then to synthesize suggestions for improvements. We leave this to future work.

[10] A demo application based on our presented approaches is found at: http://webis16.medien.uni-weimar.de/essay-scoring

References

Khalid Al-Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen, and Benno Stein. 2016. A News Editorial Corpus for Mining Argumentation Strategies. In *Proceedings of the 26th International Conference on Computational Linguistics*.

J. Anthony Blair. 2012. Relevance, Acceptability and Sufficiency Today. In *Groundwork in the Theory of Argumentation*, pages 87–100.

John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jenn Wortman. 2008. Learning Bounds for Domain Adaptation. In *Advances in Neural Information Processing Systems 21*. MIT Press.

Filip Boltužić and Jan Šnajder. 2015. Identifying Prominent Arguments in Online Debates Using Semantic Textual Similarity. In *Proceedings of the Second Workshop on Argumentation Mining*, pages 110–115.

Jill Burstein, Karen Kukich, Susanne Wolff, Chi Lu, and Martin Chodorow. 1998. Enriching Automated Essay Scoring Using Discourse Marking. In *Discourse Relations and Discourse Markers*, pages 15–21.

Elena Cabrio and Serena Villata. 2012. Combining Textual Entailment and Argumentation Theory for Supporting Online Debates Interactions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers*, pages 208–212.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Danish Contractor, Yufan Guo, and Anna Korhonen. 2012. Using Argumentative Zones for Extractive Summarization of Scientific Articles. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 663–678.

Semire Dikli. 2006. An Overview of Automated Scoring of Essays. *Journal of Technology, Learning, and Assessment*, 5(1).

Phan Minh Dung. 1995. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artificial Intelligence*, 77(2):321–357.

Adam Robert Faulkner. 2014. *Automated Classification of Argument Stance in Student Essays: A Linguistically Motivated Approach with an Application for Supporting Argument Summarization*. Dissertation, City University of New York.

Vanessa Wei Feng, Ziheng Lin, and Graeme Hirst. 2014. The Impact of Deep Hierarchical Discourse Structures in the Evaluation of Text Coherence. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 940–949.

James B. Freeman. 2011. *Argument Structure: Representation and Theory*. Springer.

Debanjan Ghosh, Aquila Khanam, Yubo Han, and Smaranda Muresan. 2016. Coarse-grained Argumentation Features for Scoring Persuasive Essays. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 549–554.

Sylviane Granger, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. 2009. International Corpus of Learner English (Version 2).

Ivan Habernal and Iryna Gurevych. 2015. Exploiting Debate Portals for Semi-Supervised Argumentation Mining in User-Generated Web Discourse. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2127–2137.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1):10–18.

Thorsten Joachims. 1999. Advances in Kernel Methods. chapter Making Large-scale Support Vector Machine Learning Practical, pages 169–184. MIT Press, Cambridge, MA, USA.

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

Raquel Mochales and Marie-Francine Moens. 2011. Argumentation Mining. *Artificial Intelligence and Law*, 19(1):1–22.

Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau, and Chris Reed. 2007. Automatic Detection of Arguments in Legal Texts. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law*, pages 225–230.

Nathan Ong, Diane Litman, and Alexandra Brusilovsky. 2014. Ontology-Based Argument Mining and Automatic Essay Scoring. In *Proceedings of the First Workshop on Argumentation Mining*, pages 24–28.

Andreas Peldszus and Manfred Stede. 2013. From Argument Diagrams to Argumentation Mining in Texts: A Survey. *International Journal of Cognitive Informatics and Natural Intelligence*, 7(1):1–31.

Andreas Peldszus and Manfred Stede. 2015. Joint Prediction in MST-style Discourse Parsing for Argumentation Mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948.

Isaac Persing and Vincent Ng. 2013. Modeling Thesis Clarity in Student Essays. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics - Volume 1: Long Papers*, pages 260–269.

Isaac Persing and Vincent Ng. 2014. Modeling Prompt Adherence in Student Essays. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics - Volume 1: Long Papers*, pages 1534–1543.

Isaac Persing and Vincent Ng. 2015. Modeling Argument Strength in Student Essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 543–552.

Isaac Persing, Alan Davis, and Vincent Ng. 2010. Modeling Organization in Student Essays. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 229–239.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show Me Your Evidence – An Automatic Method for Context Dependent Evidence Detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 440–450.

Helmut Schmid. 1995. Improvements in Part-of-Speech Tagging with an Application to German. In *Proceedings of the ACL SIGDAT-Workshop*, pages 47–50.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Yi Song, Michael Heilman, Beata Beigman Klebanov, and Paul Deane. 2014. Applying argumentation schemes for essay scoring. In *Proceedings of the First Workshop on Argumentation Mining*, pages 69–78.

Christian Stab and Iryna Gurevych. 2014a. Annotating Argument Components and Relations in Persuasive Essays. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1501–1510.

Christian Stab and Iryna Gurevych. 2014b. Identifying Argumentative Discourse Structures in Persuasive Essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 46–56.

Simone Teufel, Advaith Siddharthan, and Colin Batchelor. 2009. Towards Discipline-independent Argumentative Zoning: Evidence from Chemistry and Computational Linguistics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1493–1502.

Stephen E. Toulmin. 1958. *The Uses of Argument*. Cambridge University Press.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, and Gregor Engels. 2014a. Modeling Review Argumentation for Robust Sentiment Analysis. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 553–564.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014b. A Review Corpus for Argumentation Analysis. In *Proceedings of the 15th International Conference on Intelligent Text Processing and Computational Linguistics*, pages 115–127.

Henning Wachsmuth, Johannes Kiesel, and Benno Stein. 2015. Sentiment Flow – A General Model of Web Review Argumentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 601–611, Lisbon, Portugal.

Henning Wachsmuth. 2015. *Text Analysis Pipelines—Towards Ad-hoc Large-scale Text Mining*, volume 9383 of *Lecture Notes in Computer Science*. Springer.

Douglas Walton, Christopher Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.

Grammatical Templates: Improving Text Difficulty Evaluation for Language Learners

Shuhan Wang
Department of Computer Science
Cornell University
forsona@cs.cornell.edu

Erik Andersen
Department of Computer Science
Cornell University
eland@cs.cornell.edu

Abstract

Language students are most engaged while reading texts at an appropriate difficulty level. However, existing methods of evaluating text difficulty focus mainly on vocabulary and do not prioritize grammatical features, hence they do not work well for language learners with limited knowledge of grammar. In this paper, we introduce *grammatical templates*, the expert-identified units of grammar that students learn from class, as an important feature of text difficulty evaluation. Experimental classification results show that grammatical template features significantly improve text difficulty prediction accuracy over baseline readability features by 7.4%. Moreover, we build a simple and human-understandable text difficulty evaluation approach with 87.7% accuracy, using only 5 grammatical template features.

Keywords text difficulty evaluation, education, grammatical templates, language learners.

1 Introduction

Evaluating *text difficulty*, or *text readability*, is an important topic in natural language processing and applied linguistics (Zamanian and Heydari, 2012; Pitler and Nenkova, 2008; Fulcher, 1997). A key challenge of text difficulty evaluation is that linguistic difficulty arises from both vocabulary and grammar (Richards and Schmidt, 2013). However, most existing tools either do not sufficiently take the impact of grammatical difficulty into account (Smith III et al., 2014; Sheehan et al., 2014), or use traditional syntactic features, which differ from what language students actually learn, to estimate grammatical complexity (Schwarm and Ostendorf, 2005; Heilman et al., 2008; François and Fairon, 2012). In fact, language courses introduce grammar constructs together with vocabulary, and grammar constructs vary in frequency and difficulty just like vocabulary (Blyth, 1997; Manzanares and López, 2008; Waara, 2004). Ideally, we would like to have better ways of estimating the grammatical complexity of a sentence.

To make progress in this direction, we introduce *grammatical templates* as an important feature in text difficulty evaluation. These templates are what language teachers and linguists have identified as the most important units of grammatical understanding at different levels, and what students actually learn in language lessons. We also demonstrate that grammatical templates can be automatically extracted from the dependency-based parse tree of a sentence.

To evaluate, we compare the difficulty prediction accuracy of grammatical templates with existing readability features in Japanese language placement tests and textbooks. Our results show that grammatical template features slightly outperform existing readability features. Moreover, adding grammatical template features into existing readability features significantly improves the accuracy by 7.4%. We also propose a multilevel linear classification algorithm using only 5 grammatical features. We demonstrate that this simple and human-understandable algorithm effectively predicts the difficulty level of Japanese texts with 87.7% accuracy.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1692–1702, Osaka, Japan, December 11-17 2016.

2 Related Work

Text difficulty evaluation has been widely studied over the past few decades (Nelson et al., 2012; Sinha et al., 2012; Hancke et al., 2012; Jameel et al., 2012; Gonzalez-Dios et al., 2014; Sinha et al., 2014). Researchers have developed over 200 metrics of text difficulty (Collins-Thompson and Callan, 2004). For example, *Lexile* measures text complexity and readability with word frequency and sentence length (Smith III et al., 2014). *ATOS*[1] includes two formulas for texts and books, both of which take into account three variables to predict text difficulty: word length, word grade level and sentence length. *TextEvaluator* is a comprehensive text analysis system designed to help teachers and test developers evaluate the complexity characteristics of reading materials (Sheehan et al., 2014). It incorporates more vocabulary features, such as meaning and word type, as well as some sentence and paragraph-level features.

Nevertheless, most of these methods provide limited consideration of grammatical difficulty, which is a major challenge for foreign language learners (Callan and Eskenazi, 2007). In fact, text readability not only depends on sentence lengths or word counts, but on 'the grammatical complexity of the language used' as well (Richards and Schmidt, 2013). Based on this fact, recent readability evaluation systems improved performance by incorporating syntactic features like parse tree depth (Schwarm and Ostendorf, 2005) and subtree patterns (Heilman et al., 2008) to measure grammatical complexity. Moreover, researchers have developed an unified framework of text readability evaluation, which combines lexical, syntactic and discourse features, and predicts readability with outstanding accuracy (Pitler and Nenkova, 2008). The relationship between text readability and reading devices was also studied in the past two years (Kim et al., 2014). However, most of these approaches are intended for native speakers and use texts from daily news, economic journals or scientific publications, which are too hard to read for beginning and intermediate language learners. Ideally, we would have specific features and approaches for text difficulty evaluation for language learners.

Recently, language educational researchers conducted a bunch of studies on text readability evaluation for language learners in different languages, such as English, German, Portuguese and French (Blyth, 1997; Waara, 2004; Manzanares and López, 2008; Vajjala and Meurers, 2012; François and Fairon, 2012; Xia et al., 2016). However, they use traditional syntactic features such as sentence length, part of speech ratios, number of clauses and average parse tree height, which differ from the grammatical knowledge that students actually learn in language lessons. For example, Curto et al. measured text difficulty using traditional vocabulary and syntactic features, to predict text difficulty levels for Portuguese language learners (Curto et al., 2015). Unfortunately, 75% accuracy in 5-level classification with 52 features is not satisfactory. Instead, we extract grammatical features from *grammatical templates*, the knowledge units that language students actually learn in classes and that expert language instructors have identified and highlighted in textbooks. We also propose a novel technique that has a simpler and human-interpretable structure, uses only 5 grammatical template features, and predicts text difficulty with 87.7% accuracy in 5-level classification.

3 Grammatical Template Analysis

A key challenge in modeling text difficulty is to specify all prerequisite knowledge required for understanding a certain sentence. Traditional methods measure text difficulty mostly by evaluating the complexity of vocabulary (word count, word frequency, word type, etc.). This is effective for native speakers, who typically understand the grammar of their language but vary in mastery of vocabulary. However, these vocabulary-based methods underperform for language learners who have limited knowledge of grammar (Callan and Eskenazi, 2007; Curto et al., 2015).

To resolve this, we focus our research on grammatical difficulty. We introduce the idea of *grammatical templates*, units of grammar that expert language instructors and linguists have identified as the most important grammatical knowledge, and are typically emphasized as key points in every textbook lesson (Banno et al., 2011; People's Education Press, 2013). Since these grammatical templates are

[1] http://www.renaissance.com/Products/Accelerated-Reader/ATOS/ATOS-Analyzer-for-Text

taught explicitly in language lessons and learned directly by language students, we believe they reflect the conceptual units of grammar more closely than parse trees.

Grammatical templates play an important role in language understanding because:

- Many grammatical templates suggest sentence structure. For example, "hardly ... when ..." in English, "nicht nur ..., sondern auch ..." (not only ... but also ...) in German, and "必ずしも ... とはいえない" (it is not necessarily the case that ...) in Japanese;
- For languages like Chinese and Japanese, lacking knowledge of some grammatical templates will cause difficulties in segmentation. For example, consider the Japanese template "...つ...つ" (two opposite behaviors occuring alternately) in the phrase "行きつ戻りつ" (to walk back and forth), and the Chinese template "越...越好" (the more ... the better) in "越早越好"(the earlier the better);
- Some grammatical templates may refer to special meanings that cannot be understood as the combination of individual words. For example, "in terms of", "such that" in English, "mit etwas zu tun haben" (have something to do with ...) in German, and "... ことはない" (no need to ...) in Japanese.

We show some simple examples of grammatical templates for Japanese in Table 1[2]. Line 2 shows the pronunciation of the templates, line 3 shows the translations, and the uppercase letters in line 4 are provided for notation. We also provide examples of how the grammar of a sentence can be described as combinations of these grammatical templates in Table 2.

3.1 Difficulty Evaluation Standard

To evaluate the difficulty of texts and grammatical templates, we follow the standard of the Japanese-Language Proficiency Test (JLPT). The JLPT is the most widely used test for measuring proficiency of non-native speakers, with approximately 610,000 examinees in 62 countries and areas worldwide in 2011[3]. It has five different levels, ranging from from N5 (beginner) to N1 (advanced). A summary of the levels can be found at JLPT website [4].

3.2 Grammatical Template Library

Due to their significance in Japanese education, grammatical templates are well-studied by Japanese teachers and researches. Grammatical templates are summarized and collected for both Japanese learners (common templates) and native speakers (templates used in very formal Japanese or old Japanese). We referenced 3 books about grammatical templates for Japanese learners (Sasaki and Kiko, 2010; Xu and Reika, 2015; Liu and Ebihara, 2012), all of which divide their templates into N1-N5 levels, for generating our template library at each corresponding level.

Although not common, books may have different opinions on the difficulty of the same template. For example, an N1 template in book A may be recognized as an N2 template in book B. In order to conduct our experiments on a reliable template library, we only pick the templates recognized as the same level by at least two of the three books. For example, if both book A and C recognized template t as an N3 template, we can incorporate template t into our N3 template library. Ultimately, we collected 147 N1 templates, 122 N2 templates, 74 N3 templates, 95 N4 templates and 128 N5 templates in our library. All selected grammatical templates are stored in the format of regular expressions for easy matching in parse trees.

3.3 Grammatical Template Extraction

The framework of grammatical template extraction is shown in Algorithm 1. The program requires the dependency-based parse tree of a sentence as input, runs from bottom to top and returns a set of

[2]A long list of Japanese grammatical templates with English translations can be accessed at the JGram website: http://www.jgram.org/pages/viewList.php. There is also a nice and comprehensive book of Japanese grammatical templates, written by Japanese linguists, with English, Korean and Chinese translations: (Tomomatsu Etsuko and Masako, 2010).

[3]http://www.jlpt.jp/e/about/message.html

[4]http://www.jlpt.jp/e/about/levelsummary.html

Template	– は	– の	– を	– では ない	–(名詞) に	–(動詞連用形) に
Pronunciation	– *wa*	– *no*	– *o*	– *dewa nai*	–(noun) *ni*	–(verb, i-form) *ni*
Translation	(topic)	(genitive)	(object)	is not	to (location)	for (purpose)
Notation	A	B	C	D	E	F

Table 1: Grammatical Templates in Japanese, with hyphens denoting words to be filled in. Note that some grammatical templates may impose requirements of some properties (e.g. part of speech or form) on the missing words.

Sentence	彼	**は**	すぐ	東京	**に**	到着する	
Pronunciation	kare	**wa**	sugu	*toukyou*	**ni**	touchakusuru	
Translation	he	**(topic)**	soon	*Tokyo*	**to (location)**	arrive	
Templates		A			E		
	“ he will soon arrive in Tokyo ”						
Sentence	僕	**の**	彼女	**を**	見	**に**	行く
Pronunciation	boku	**no**	kanojo	**o**	*mi*	**ni**	iku
Translation	I	**(genitive)**	girlfriend	**(object)**	*see*	**for (purpose)**	go
Templates		B		C		F	
	“ I go to see my girlfriend ”						
Sentence	これ	**は**	君	**の**	本	**では**	**ない**
Pronunciation	kore	**wa**	kimi	**no**	hon	**dewa**	**nai**
Translation	this	**(topic)**	you	**(genitive)**	book	**is**	**not**
Templates		A		B		D	
	“ this is not your book ”						

Table 2: Identified grammatical templates of Japanese sentences. In sentences, pronunciations and translations, grammatical templates are in bold. The word *toukyou* in the first sentence is a noun (Tokyo,東京), as characterized by template E. The word *mi* (to see, 見) in the second sentence is the i-form (動詞連用形) of a verb, as required by template F.

	N1 Texts	N2 Texts	N3 Texts	N4 Texts	N5 Texts
N1 Templates	0.902%	0.602%	0.077%	0.074%	0.056%
N2 Templates	2.077%	1.571%	1.072%	0.298%	0.056%
N3 Templates	4.070%	3.679%	1.531%	0.894%	0.222%
N4 Templates	16.635%	15.449%	13.323%	12.071%	1.832%
N5 Templates	76.316%	78.699%	83.997%	86.662%	97.834%

Table 3: Distribution of grammatical templates of level N1(hard)-N5(easy)

	N1 Texts	N2 Texts	N3 Texts	N4 Texts	N5 Texts
N1 Templates	3.536	2.342	0.295	0.230	0.146
N2 Templates	8.141	6.110	4.130	0.922	0.146
N3 Templates	15.954	14.308	5.900	2.765	0.582
N4 Templates	65.214	60.081	51.327	37.327	4.803
N5 Templates	299.178	306.059	323.599	267.972	256.477

Table 4: Number of templates of level N1(hard)-N5(easy) per 100 sentences

all identified grammatical templates $\mathbf{T}(node_0)$. Line 7 extracts the templates in the children of $node_0$ (and ignores the descendants of the children), by matching the phrase associated with the child nodes $[node_1, node_2, \cdots]$ to all templates stored in terms of regular expressions in our library. The matching is based on both the structure of the phrases and the properties of the words. Line 8 shows $\mathbf{T}(node_0)$ covers all templates identified in subtrees rooted at $node_0$'s children and the templates extracted in the phrase associated with the child nodes $[node_1, node_2, \cdots]$.

Algorithm 1 Grammatical Progression Extraction

Require: A dependency-based parse tree of the sentence

Ensure: $\mathbf{T}(node_0)$ = set of identified grammatical templates in (sub)parse tree rooted at $note_0$.

1: **if** $node_0$ is leaf node **then**
2: return $\mathbf{T}(node_0) = \{\}$
3: **end if**
4: $node_1, node_2, \cdots \leftarrow$ children of $node_0$
5: Calculate: $\mathbf{T}(node_1), \mathbf{T}(node_1), \cdots$ // templates identified in subtrees rooted at $node_0$'s children
6: $\mathbf{T_1}(node_0) \leftarrow \mathbf{T}(node_1) \cup \mathbf{T}(node_2) \cup \cdots$
7: $\mathbf{T_2}(node_0) \leftarrow$ identified templates in phrase $[node_1, node_2, \cdots]$
8: return $\mathbf{T}(node_0) = \mathbf{T_1}(node_0) \cup \mathbf{T_2}(node_0)$

We use Cabocha (Kudo and Matsumoto, 2002) for parsing Japanese sentences. This tool generates the hierarchical structure of the sentence as well as some properties (e.g. base form, pronunciation, part of speech, etc.) of each word. We execute Algorithm 1 on the parse tree to extract all identified templates of a Japanese sentence.

4 Statistics of Grammatical Templates

4.1 Corpus

We build our corpus from two sources: past JLPT exams and textbooks. The reading texts from JLPT exams are ideal for difficulty evaluation experiments since all of them are tagged authoritatively with difficulty levels, and JLPT problem sets before 2010 are publicly released[5]. We also collected reading texts from two popular series of Japanese textbooks: *Standard Japanese* (People's Education Press, 2013) and *Genki* (Banno et al., 2011). *Standard Japanese I* and *Genki I* are designed for the N5 level (the first semester) and *Standard Japanese II* and *Genki II* are designed for the N4 level (the second semester). Ultimately, our corpus consists of 220 texts (150 from past JLPT exams and 70 from textbooks), totaling 167,292 words after segmentation.

4.2 Results

For texts with different difficulties, we calculate the distribution of N1-N5 grammatical templates, which are shown in Table 3. We can see that N1 texts have higher portion of N1 and N2 templates than N2 texts, implying that the difficulty boosts from N2 to N1 are derived from increasing usage of advanced grammar. It is also clear that even in the texts of advanced levels, the majority of the sentences are organized by elementary grammatical templates, and the advanced ones are only used occasionally for formality or preciseness.

We also calculate the per-100-sentence number of templates at each level, which are shown in Table 4. When comparing any two adjacent levels (e.g. N2 and N3), the templates at those levels or above seem to be the most significant. For instance, N1/N2 texts differ in numbers of N1 and N2 templates while they have similar numbers of N3-N5 templates, and the numbers of N1, N2 and N3 templates differentiate

[5]For example, the second exam in 2009 is published in (Japan Educational Exchanges et al., 2010).

the N2/N3 texts while the numbers of N4 and N5 templates seem relatively similar. This phenomenon inspires us to build a simple and effective approach to differentiate the texts of two adjacent levels.

5 Difficulty Level Prediction

5.1 Multilevel Linear Classification

We differentiate two adjacent levels by looking at the knowledge 'on the boundary' and 'outside the boundary'. Concretely, when judging whether a text is harder than level N_i, we consider a grammatical template as:

- *within the boundary*, if the template is easier than N_i (N_{i+1} to N_5);
- *on the boundary*, if the template is exactly at N_i level;
- *outside the boundary*, if the template is harder than N_i (N_1 to N_{i-1}).

We found that texts of adjacent levels are nearly linear-separable with two features: templates 'on the boundary' and templates 'outside the boundary'. For example, Figure 1 shows how N1 and N2 texts are linearly separated based on the numbers of N1 and N2 templates: we can easily obtain a two-dimensional linear classifier separating N1 and N2 texts with 83.4% accuracy. This phenomenon is even more obvious at lower levels. Figure 2 shows N4 and N5 texts are almost perfectly linearly separated with two features: 'number of N5 templates per 100 sentences' (on the boundary) and 'number of N1-N4 templates per 100 sentences' (outside the boundary).

Taking advantage of this phenomenon, we build 4 linear classifiers for 4 pairs of adjacent levels. For example, the N4 classifier judges whether a text is harder than N4 (N1-N3). Our *Multilevel Linear Classification (MLC)* algorithm combines all 4 linear classifiers: A text is judged by the N5 classifier first. If it is no harder than N5, it will be labeled as an N5 text; otherwise, it will be passed to the N4 classifier in order to decide if it is harder than N4. The process continues similarly, until if it is judged to be harder than N2, it will be labeled as an N1 text. Figure 3 shows how the algorithm works.

5.2 Features

We conduct our experiments on the following 4 feature sets:

First, our *grammatical template feature set* has only 5 features:

- Average number of N1-N5 grammatical templates per sentence

We compare our work with recent readability evaluation studies (Kim et al., 2014; Pitler and Nenkova, 2008). In our experiments, the *baseline readability feature set* consists of the following 12 features:

- Number of words in a text
- Number of sentences in a text
- Average number of words per sentence
- Average parse tree depths per sentence
- Average number of noun phrases per sentence
- Average number of verb phrases per sentence
- Average number of pronouns per sentence
- Average number of clauses per sentence
- Average cosine similarity between adjacent sentences
- Average word overlap between adjacent sentences
- Average word overlap over noun and pronoun only
- Article likelihood estimated by language model

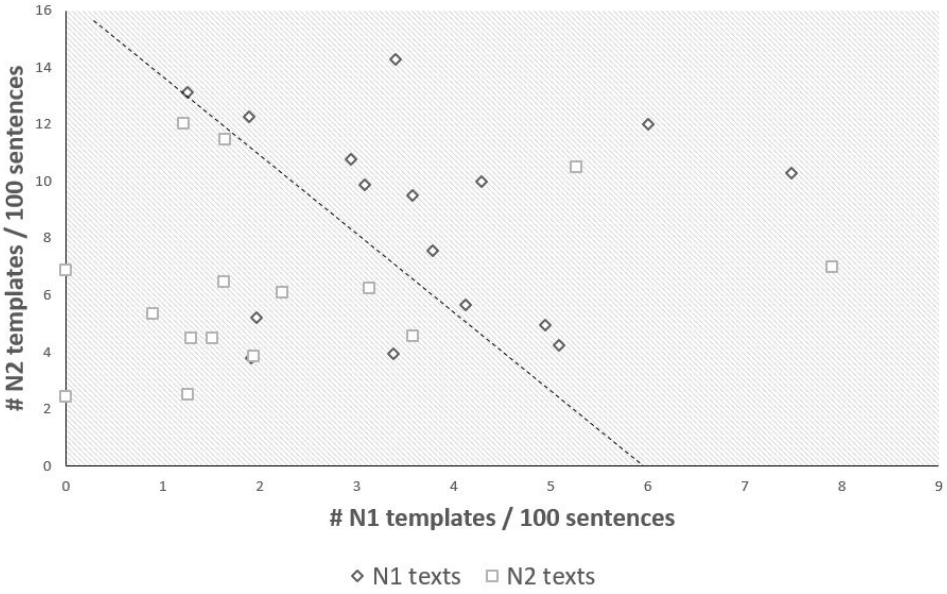


Figure 1: Grammatical difficulty in the N1/N2 texts

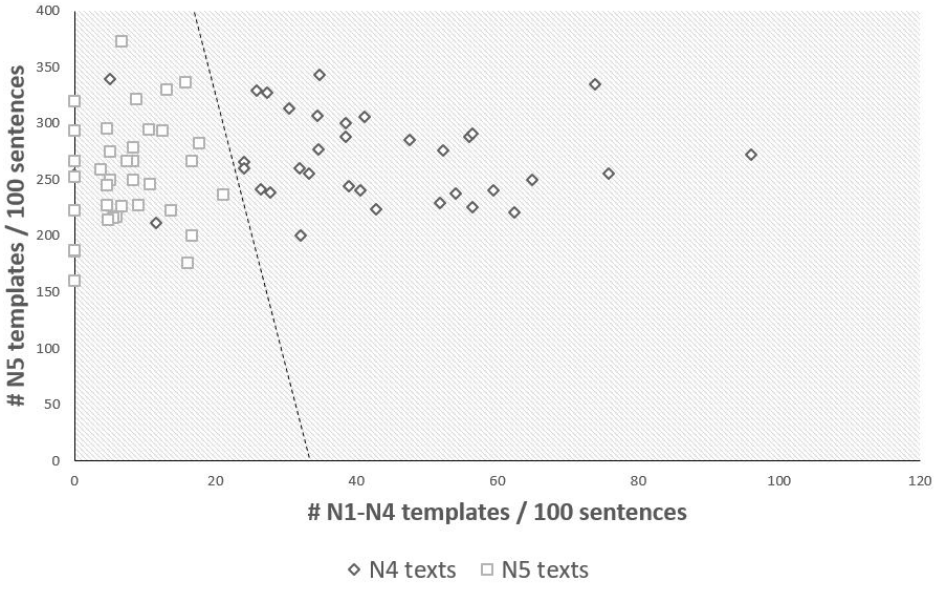


Figure 2: Grammatical difficulty in the N4/N5 texts

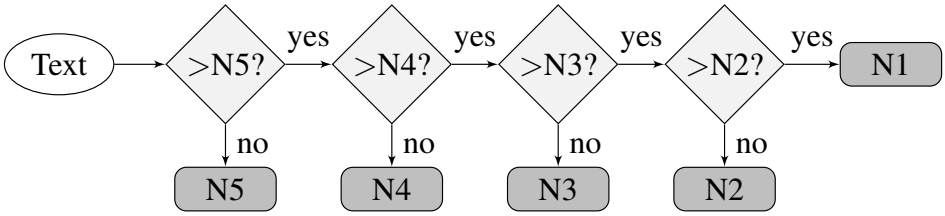


Figure 3: Multilevel Linear Classification (MLC). '>N5?' represents the linear classifier judging whether a text is harder than N5. The classifiers are similar for the other levels.

Feature Set (number of features)	Algorithm	Accuracy
TF-IDF Features (5100)	kNN	69.1%
	SVM	80.5%
Baseline Readability Features (12)	kNN	72.3%
	SVM	80.9%
Grammatical Template Features (**5**)	kNN	78.0%
	SVM	81.1%
	MLC	**87.7%**
Hybrid Features (17)	kNN	85.7%
	SVM	**88.5%**

Table 5: Accuracies of classifying N1-N5 texts

Moreover, we combine these 12 traditional readability features with our 5 grammatical template features, forming a *'hybrid' feature set*, since we would like to see if grammatical template features are really able to improve text difficulty evaluation.

Since the text difficulty level prediction can be regarded as a special text classification problem, we also extract *TF-IDF features* (Sparck Jones, 1972) (Nelson et al., 2012) as an extra baseline, in order to see how general text classification techniques work on text difficulty evaluation.

5.3 Result

We test k-Nearest Neighbor and Support Vector Machines for each feature set. The implementations of these two popular classification algorithms are provided by the WEKA toolkit (Hall et al., 2009) and LibSVM (Chang and Lin, 2011). The SVMs use RBF kernels (Chang et al., 2010). We also test our Multilevel Linear Classification (MLC) algorithm on the grammatical template feature set. We use 5-fold cross validation to avoid overfitting. Table 5 shows the results.

Comparing the results of kNN and SVM across the four different feature sets in Table 5, it is clear that TF-IDF features have the largest feature set yet lowest accuracy, indicating the general word-based text classification techniques do not work well on text difficulty level prediction. Compared with baseline readability features, our grammatical template features have smaller number of features but higher accuracy (slightly higher with SVM but significantly higher with kNN). Moreover, the hybrid features, which combine baseline readability features with grammatical template features, decisively outperform baseline readability features, confirming our expectation that adding grammatical template features to existing readability techniques improves text difficulty evaluation for language learners.

Additionally, our Multilevel Linear Classification algorithm achieves excellent accuracy with only 5 grammatical template features. An accuracy of 87.7% , although slightly lower than hybrid features + SVM (more features, more complexity), still significantly outperforms baseline readability techniques. In conclusion, the Multilevel Linear Classification algorithm has high accuracy, a small number of features, and a simple, human-understandable structure.

6 Conclusions and Future Work

We proposed a new approach for evaluating text difficulty that focuses on grammar and utilizes expert-identified grammatical templates, the grammar knowledge that students actually learn in language lessons. This approach significantly improved the accuracy of text difficulty evaluation for Japanese language learning. We also introduced a simple, human-understandable, and effective text difficulty evaluation approach using only five grammatical template features.

In future work, we are interested in extending our work to other languages like English, and adapting grammatical templates for various languages. To achieve this, we need to itemize the grammar knowledge that students learn from language lessons. We can also develop a machine learning system that can automatically discover discriminative grammatical templates from texts. Moreover, we would like

to study if the topic of a text has considerable impact on text difficulty for language learners, just like vocabulary and grammar.

We also hope to use our approach to recommend reading texts to individual learners at appropriate difficulty levels. For instance, Japanese news articles could be good learning materials for advanced Japanese language learners. We want to build an online tool to collect reading texts from current news reports in specific target languages, and select appropriate ones for language learners, especially intermediate and advanced learners.

Finally, we plan to leverage some novel ideas from Human-Computer Interaction and educational technology (Andersen et al., 2013) to build an *adaptive* Computer-Assisted Language Learning (CALL) system. Using our new approach introduced in this paper, we can decompose the difficulty of a text into several basic skills (grammatical templates), and model the internal hierarchical structure of a sequence of texts with a partial ordering graph. Using this structure, we can comprehensively assess a student's knowledge and tailor optimal learning progressions for individual students.

Acknowledgements

Special thanks to Xiang Long for his help during the writing of this paper.

References

Erik Andersen, Sumit Gulwani, and Zoran Popovic. 2013. A trace-based framework for analyzing and synthesizing educational progressions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 773–782. ACM.

Eri Banno, Yoko Ikeda, and Yutaka Ohno. 2011. *GENKI: An Integrated Course in Elementary Japanese*. Japan Times and Tsai Fong Books.

Carl Blyth. 1997. A constructivist approach to grammar: Teaching teachers to teach aspect. *The Modern Language Journal*, 81(1):50–66.

Jamie Callan and Maxine Eskenazi. 2007. Combining lexical and grammatical features to improve readability measures for first and second language texts. In *Proceedings of NAACL HLT*, pages 460–467.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. 2010. Training and testing low-degree polynomial data mappings via linear svm. *The Journal of Machine Learning Research*, 11:1471–1490.

Kevyn Collins-Thompson and James P Callan. 2004. A language modeling approach to predicting reading difficulty. In *HLT-NAACL*, pages 193–200.

Pedro Curto, Nuno Mamede, and Jorge Baptista. 2015. Assisting european portuguese teaching: Linguistic features extraction and automatic readability classifier. In *Computer Supported Education*, pages 81–96. Springer.

Thomas François and Cédrick Fairon. 2012. An ai readability formula for french as a foreign language. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 466–477. Association for Computational Linguistics.

Glenn Fulcher. 1997. Text difficulty and accessibility: Reading formulae and expert judgement. *System*, 25(4):497–513.

Itziar Gonzalez-Dios, María Jesús Aranzabe, Arantza Díaz de Ilarraza, and Haritz Salaberri. 2014. Simple or complex? assessing the readability of basque texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 334–344, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Julia Hancke, Sowmya Vajjala, and Detmar Meurers. 2012. Readability classification for German using lexical, syntactic, and morphological features. In *Proceedings of COLING 2012*, pages 1063–1080, Mumbai, India, December. The COLING 2012 Organizing Committee.

Michael Heilman, Kevyn Collins-Thompson, and Maxine Eskenazi. 2008. An analysis of statistical models and features for reading difficulty prediction. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, pages 71–79. Association for Computational Linguistics.

Shoaib Jameel, Xiaojun Qian, and Wai Lam. 2012. N-gram fragment sequence based unsupervised domain-specific document readability. In *Proceedings of COLING 2012*, pages 1309–1326, Mumbai, India, December. The COLING 2012 Organizing Committee.

Japan Educational Exchanges, Services, and Japan Foundation. 2010. *The 2009-2 Japanese Language Proficiency Test Level 1 and 2: Questions and Correct Answers*. Bonjinsha Inc.

A-Yeong Kim, Hyun-Je Song, Seong-Bae Park, and Sang-Jo Lee. 2014. Device-dependent readability for improved text understanding. In *EMNLP*, pages 1396–1404.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–7. Association for Computational Linguistics.

Wenzhao Liu and Hiroshi Ebihara. 2012. *New JLPT N1 Grammar Description*.

Javier Valenzuela Manzanares and Ana María Rojo López. 2008. What can language learners tell us about constructions? *APPLICATIONS OF COGNITIVE LINGUISTICS*, 9:197.

Jessica Nelson, Charles Perfetti, David Liben, and Meredith Liben. 2012. Measures of text difficulty: Testing their predictive value for grade levels and student performance. *Council of Chief State School Officers, Washington, DC2012*.

People's Education Press. 2013. *Standard Japanese of China-Japan Exchanges for Beginners*. Mitsumura Tosho Publishing Co.Ltd.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the conference on empirical methods in natural language processing*, pages 186–195. Association for Computational Linguistics.

Jack C Richards and Richard W Schmidt. 2013. *Longman dictionary of language teaching and applied linguistics*. Routledge.

Hitoko Sasaki and Matsumoto Kiko. 2010. *Japanese Language Proficiency Test N1 GRAMMAR Summary*.

Sarah E Schwarm and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 523–530. Association for Computational Linguistics.

Kathleen M Sheehan, Irene Kostin, Diane Napolitano, and Michael Flor. 2014. The textevaluator tool: Helping teachers and test developers select texts for use in instruction and assessment. *The Elementary School Journal*, 115(2):184–209.

Manjira Sinha, Sakshi Sharma, Tirthankar Dasgupta, and Anupam Basu. 2012. New readability measures for Bangla and Hindi texts. In *Proceedings of COLING 2012: Posters*, pages 1141–1150, Mumbai, India, December. The COLING 2012 Organizing Committee.

Manjira Sinha, Tirthankar Dasgupta, and Anupam Basu. 2014. Influence of target reader background and text features on text readability in bangla: A computational approach. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 345–354, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Malbert Smith III, Anne Schiano, and Elizabeth Lattanzio. 2014. Beyond the classroom. *Knowledge Quest*, 42(3):20.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Miyamoto Atsushi Tomomatsu Etsuko and Waguri Masako. 2010. *Essential Japanese Expression Dictionary: A Guide to Correct Usage of Key Sentence Patterns (New Edition)*. ALC Press.

Sowmya Vajjala and Detmar Meurers. 2012. On improving the accuracy of readability classification using insights from second language acquisition. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 163–173. Association for Computational Linguistics.

Renee Waara. 2004. Construal, convention, and constructions in l2 speech. *Cognitive linguistics, second language acquisition and foreign language pedagogy*, pages 51–75.

Menglin Xia, Ekaterina Kochmar, and Ted Briscoe. 2016. Text readability assessment for second language learners. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 12–22. Association for Computational Linguistics.

Xiaoming Xu and Reika. 2015. *Blue Book All-in-one: JLPT N1-N5 Grammar*.

Mostafa Zamanian and Pooneh Heydari. 2012. Readability of texts:: State of the art. *Theory and Practice in Language Studies*, 2(1):43.

Still not there? Comparing Traditional Sequence-to-Sequence Models to Encoder-Decoder Neural Networks on Monotone String Translation Tasks

Carsten Schnober[†‡], Steffen Eger[†], Erik-Lân Do Dinh[†], and Iryna Gurevych[†‡]

[†]Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
[‡]Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research and Educational Information
https://www.ukp.tu-darmstadt.de/

Abstract

We analyze the performance of encoder-decoder neural models and compare them with well-known established methods. The latter represent different classes of traditional approaches that are applied to the monotone sequence-to-sequence tasks OCR post-correction, spelling correction, grapheme-to-phoneme conversion, and lemmatization. Such tasks are of practical relevance for various higher-level research fields including *digital humanities*, automatic text correction, and speech recognition. We investigate how well generic deep-learning approaches adapt to these tasks, and how they perform in comparison with established and more specialized methods, including our own adaptation of pruned CRFs.

1 Introduction

Encoder-decoder neural models (Sutskever et al., 2014) are a generic deep-learning approach to sequence-to-sequence translation (Seq2Seq) tasks. They encode an input sequence into a vector representation from which the decoder generates an output. These models have shown to achieve state-of-the-art or at least highly competitive results for various NLP tasks including machine translation (Cho et al., 2014), conversation modeling (Vinyals and Le, 2015), question answering (Yin et al., 2016), and, more generally, language correction (Schmaltz et al., 2016; Xie et al., 2016).

We have noticed that, given the enormous interest currently surrounding neural architectures, recent research appears to somewhat over-enthusiastically praise the performance of encoder-decoder approaches for Seq2Seq tasks. For example, while the encoder-decoder G2P model by Rao et al. (2015) achieves an extremely low error rate on the CMUdict dataset (Kominek and Black, 2004), the neural architecture itself has a mediocre performance and only outperforms traditional models *in combination* with a weighted finite state transducer. Similarly, Faruqui et al. (2016) report on "par or better" performance of their inflection generation neural architecture. However, a closer inspection of their results suggests that their system is sometimes worse and sometimes better than traditional approaches.

Here, we aim for a more balanced comparison on three exemplary monotone[1] Seq2Seq tasks, namely *spelling correction*, *G2P conversion*, and *lemmatization*. Monotone Seq2Seq tasks such as morphological analysis/lemmatization, grapheme-to-phoneme conversion (G2P) (Yao and Zweig, 2015; Rao et al., 2015), transliteration (Sherif and Kondrak, 2007), and spelling correction (Brill and Moore, 2000) have been fundamental problem classes in natural language processing (NLP) ever since the origins of the field. Their simplicity vis-à-vis non-monotonic problems such as machine translation renders them as particularly tractable testbeds of technological progress. Unlike previous work, which has typically focussed on only one specific subproblem of monotone Seq2Seq tasks at a time, we consider model performances on *three* such tasks simultaneously. This leads to a more balanced view on the relative performance of different models.

We compare three variants of encoder-decoder models — including attention-based models (Bahdanau et al., 2014; Luong et al., 2015) and the model proposed by Faruqui et al. (2016) — to three very

[1]We call our tasks, described below, monotone because relationships between input and output sequence characters typically obey monotonicity. That is, unlike in machine translation, there are no 'crossing edges' in corresponding alignments.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1703–1714, Osaka, Japan, December 11-17 2016.

well-established baselines for monotone Seq2Seq, namely Sequitur (Bisani and Ney, 2008), DirecTL+ (Jiampojamarn et al., 2010), and Phonetisaurus (Novak et al., 2012). We also offer our own contribution[2], which may be considered a variation of the principles underlying DirecTL+. For that purpose, we have adapted higher-order pruned conditional random fields (PCRFs) (Müller et al., 2013; Lafferty et al., 2001) to handle generic monotone Seq2Seq tasks.

We find that traditional models appear to still be on par with or better than encoder-decoder models in most cases, depending on factors such as training data size and the complexity of the task at hand. We show that neural models unfold their strengths as soon as more complex phenomena need to be learned. This becomes clearly visible in the comparison between lemmatization and the other tasks we have investigated. Lemmatization is the only task at hand in which neural models outperform all established systems — as it is the only one which systematically exhibits long-range dependencies, particularly through Finnish *vowel harmony* (see Section 5). We are thus able to contrast the different challenges imposed by different tasks and show how these differences have significant impact on the performance of encoder-decoder models in comparison to established Seq2Seq models.

To our best knowledge, no systematic comparison with regard to the suitability of these encoder-decoder neural models for a wider and more generic selection of tasks has been conducted.

2 Task Description

Throughout, we denote individual tokens in a sequence by ordinary letters x, and a sequence of symbols by $\vec{x}$. Hence a string of length s is denoted as $\vec{x} = x_1 \dots x_s$. Real-valued vectors are denoted by bold-faced letters, $\mathbf{x}$.

Spelling correction is the problem of converting an 'erroneous' input sequence $\vec{x}$ into a corrected version $\vec{y}$. In terms of errors committed by humans (typos), spelling correction often deals with errors that are due to keyboard adjacency of characters and grapho-phonemic mismatch (e.g. $emergancy \rightarrow emergency$, $wuld \rightarrow would$).

OCR post-correction can be seen as a special case of spelling correction. OCR (optical character recognition) is the process of digitizing printed texts automatically, often applied to make text data from the pre-electronic age digitally available (Springmann et al., 2014). Depending on various factors including paper and scan quality, typeface, and OCR engine, OCR error rate can be extraordinarily high (Reynaert, 2014). OCR post-correction is of particular practical importance in the field of *digital humanities*. Here, paper quality, which is often bleached and tainted, and "unusual" typefaces typically cause major problems. Unlike in human spelling correction, OCR errors often arise due to *visual* similarity of character sub-sequences such as $rn \rightarrow m$ or $li \rightarrow h$.

Previous works in OCR post-correction apply noisy-channel models (Brill and Moore, 2000) and various extensions (Toutanova and Moore, 2002; Cucerzan and Brill, 2004; Gubanov et al., 2014), generic string-to-string substitution models (Xu et al., 2014), discriminative models (Okazaki et al., 2008; Farra et al., 2014), and user-interactive approaches (Reffle and Ringlstetter, 2013). Neural network designs including auto-encoders (Raaijmakers, 2013) and recurrent neural networks (Chrupała, 2014) were also investigated in previous works.

G2P conversion is the problem of converting orthographic representations into sound representations. It is the prime example of a monotone Seq2Seq task, which — as a fundamental building block for speech recognition, speech synthesis, and related tasks — has been researched for decades. It differs from the previous two tasks in that input and output strings are defined over different alphabets.

Lemmatization is the task of deriving the lemma from an inflected word form such as *atmest→atmen*. The problem is relatively simple for morphologically poor languages like English, but much harder for languages like Finnish. The task can be seen as the inverse to inflection generation (Durrett and DeNero, 2013; Ahlberg et al., 2014; Nicolai et al., 2015; Faruqui et al., 2016), where an inflected form is generated from a lemma plus an inflection tag.

[2]Our implementation of PCRF-Seq2Seq is available at:
`https://github.com/UKPLab/coling2016-pcrf-seq2seq`

3 Data

Here we detail the data sets used in our experiments; examples are provided in Table 1. These datasets reflect the different Seq2Seq tasks we aim to investigate.

The **Text+Berg** corpus (Bubenhofer et al., 2015) contains historic proceedings of the *Schweizer Alpenclub* ("Swiss Alpine Club") from the years 1864–1899 in Swiss German and French. The data has been digitized and OCR errors have been corrected manually. The corpus contains 19,024 pages, 17,186 of which are in Swiss German, and 1,838 are in French. We have extracted 88,302 unique misrecognized words along with their manually corrected counterparts.

For our experiments, we have used randomly selected 72K entries for training and test our models on another 9K entries. Furthermore, we report results for each model trained on a reduced training set (10K entries).

Twitter Typo Corpus[3]: We use a corpus of 39,172 spelling mistakes extracted from English Tweets with their respective corrections. The manually corrected mistakes come with a context word on both sides. Again, we have split the data randomly, using a training set of 31K entries and 4K for testing. We also report results on the same test set when using a reduced training set with 10K entries.

The **Combilex** data set (Richmond et al., 2009) provides mappings from English graphemes to phonetic representations (Table 1). We use different subsets for training, with 2K, 5K, 10K, and 20K entries respectively. Furthermore, we employ a test set with 26,609 entries.

P'reunde	→	**F**reunde	(misrecognition)	to_**y**ork_from	→	to_**w**ork_from
Thal wand	→	Thalwand	(segmentation)	before_**tt**_was	→	before_**i**t_was
S**lutl**erfi**m**	→	S**tud**erfi**rn**	(multiple errors)	with_my_daug**th**er	→	with_my_daug**ht**er
kinaatte	→	kinata		Waterloo	→	wOtBr5u
kinaavat	→	kinata		barnacles	→	bArn@k@5z

Table 1: Training data examples from the four corpora used: OCR detection errors for Text+Berg (top left), Twitter Typo Corpus (top right), Wiktionary Morphology Dataset (Finnish) (bottom left), and Combilex G2P mappings (bottom right).

For lemmatization, we use the **Wiktionary Morphology Dataset** (Durrett and DeNero, 2013). The data set contains inflected forms for different languages and parts of speech, corresponding lemmas, and detailed inflection information, including mood, case, and tense. We conduct experiments on the German and Finnish verb datasets and further reduce the size of the latter by considering present tense indicative verb forms in active voice only. Note that our results are not comparable to the ones presented by Durrett and DeNero (2013), Ahlberg et al. (2014), Nicolai et al. (2015), and Faruqui et al. (2016) because we focus on lemmatization, not inflection generation, as mentioned. We do so because this produces less overhead — e.g., Faruqui et al. (2016) train 27 different systems for German verbs, one for each inflection type — and there is a priori not much difference in whether we transform an inflected form to a lemma or vice versa. Hence, the relative ordering of the systems we survey should not be affected by this change of direction in the morphological analysis. In total, we have used training sets of size 43,929 entries for German verbs and 41,094 entries for Finnish verbs, and dev set and test set sizes of 5,400 (German) and 1,200 (Finnish) entries each.

4 Model Description

In this section, we briefly describe encoder-decoder neural models, pruned CRFs, and our three baselines.

4.1 Encoder-Decoder Neural Models

We compare three variants of encoder-decoder models: the 'classic' variant and two modifications:

- `enc-dec`: Encoder-decoder models using recurrent neural networks (RNNs) for Seq2Seq tasks were introduced by Cho et al. (2014) and Sutskever et al. (2014). The encoder reads an input $\vec{x}$ and

[3]Twitter typo corpus: `http://luululu.com/tweet/`

generates a vector representation $\mathbf{e}$ from it. The decoder predicts the output $\vec{y}$ one time step t at a time, based on $\mathbf{e}$. The probability for each output symbol y_t hence depends on $\mathbf{e}$ and all previously generated output symbols: $p(\vec{y}|\mathbf{e}) = \prod_{t=1}^{T'} p(y_t|\mathbf{e}, y_1 \cdots y_{t-1})$ where T' is the length of the output sequence. In NLP, most implementations of encoder-decoder models employ LSTM (long short-term memory) layers as hidden units, which extend generic RNN hidden layers with a memory cell that is able to "memorize" and "forget" features. This addresses the 'vanishing gradients' problem and allows to catch long-range dependencies.

- `attn-enc-dec`: We explore the attention-based encoder-decoder model proposed by Bahdanau et al. (2014) (Figure 1). It extends the encoder-decoder model by learning to align and translate jointly. The essential idea is that the current output unit y_t does not depend on all input units in the same way, as captured by a 'global' vector $\mathbf{e}$ encoding the input. Instead, y_t may be conditioned upon local context in the input (to which it pays *attention*).

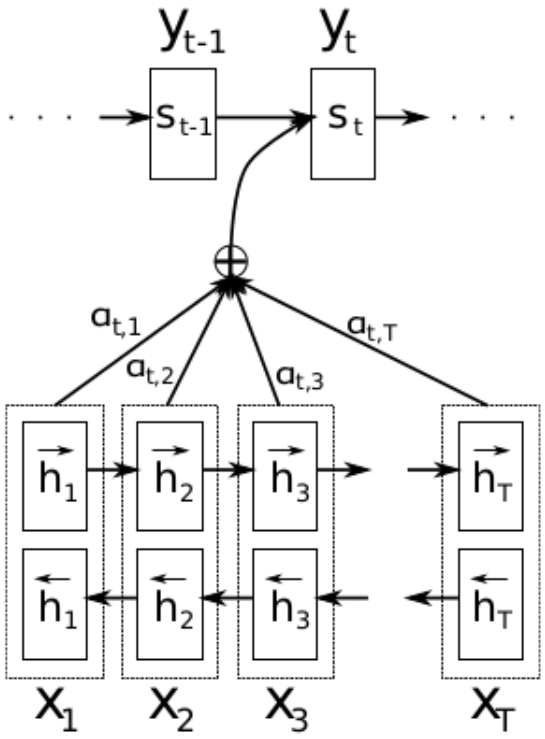


Figure 1: In the encoder-decoder model, the encoder (bottom) generates a representation of the input sequence $\vec{x}$ from which the decoder (top) generates the output sequence $\vec{y}$. The attention-based mechanism (shown here) enables the decoder to "peek" into the input at every decoding step through multiple input representations a_t. Illustration from Bahdanau et al. (2014).

- `morph-trans`: Faruqui et al. (2016) present a new encoder-decoder model designed for morphological inflection, proposing to feed the input sequence directly into the decoder. This approach is motivated by the observation that input and output are usually very similar in problems such as morphological inflection. Similar ideas have been proposed in Gu et al. (2016) in their so-called "CopyNet" encoder-decoder model (which they apply to text summarization) that allows for portions of the input sequence to be simply copied to the output sequence, without modifications. A priori, this observation seems to apply to our tasks too: at least in spelling correction, the output usually differs only marginally from the input.

For the tested neural models, we follow the same overall approach as Faruqui et al. (2016): we perform decoding and evaluation of the test data using an **ensemble** of $k = 5$ independently trained models in order to deal with the non-convex nature of the optimization problem of neural networks and the risk of

running into a local optimum (Collobert et al., 2011). The total probability p_{ens} for generating an output token y_t is estimated from the individual model output probabilities: $p_{\text{ens}}(y_t|\cdot) = \frac{1}{Z}\prod_{i=1}^{k} p_i(y_t|\cdot)^{\frac{1}{k}}$ with a normalization factor Z.

4.2 Pruned Conditional Random Fields

Conditional random fields (CRFs) were introduced by Lafferty et al. (2001) and have been a major workhorse for many sequence labeling tasks such as part-of-speech tagging and named entity recognition during the 2000s. Unfortunately, training and decoding time depend polynomially on the tag set size and exponentially on the *order* of the CRF. Here, order refers to the dependencies on the label side. This makes higher-order CRFs impractical for large training data sizes, which is the reason why virtually only first-order (linear chain) CRFs were used until recently.

Müller et al. (2013) introduced pruned CRFs (PCRFs) that approximate the CRF objective function using coarse-to-fine decoding (Charniak and Johnson, 2005). PCRFs require much shorter runtime and are thus able to make use of higher orders. Higher orders, in turn, have been shown to be highly beneficial for coarse and fine-grained part-of-speech tagging, outperforming first-order models.

For our tasks, we have adapted the implementation from Müller et al. (2013) — originally designed for sequence labeling — to general monotone Seq2Seq tasks. Sequence labeling assumes that an input sequence of length N is mapped to an output sequence of identical length N, while in Seq2Seq tasks, input string lengths may be shorter, longer, or equal to output string lengths.

We address this by first *aligning* input and output sequences as exemplified in

S	l	u	t	l	e	r	f	i	m
S	t	u	d	∅	e	r	f	i	rn

This alignment matches up character subsequences from both strings. It may include 1-to-zero matches (e.g. $l \rightarrow \emptyset$) and 1-to-many matches (e.g. $m \rightarrow rn$). We disallow many-to-1 or many-to-many matches, as they cause a problem during decoding: at test time, it is unclear how to segment a new input string into parts with size ≥ 1. A naïve 'pipeline' approach (first segment, then translate the segmented string) leads to error propagation. More sophisticated 'joint' approaches (Jiampojamarn et al., 2010) are considerably more computationally expensive.

Once the data is aligned as above, input and (modified) output sequences are of equal lengths and we can directly apply higher-order PCRFs. Below, we show that orders up to 5 (and possibly beyond) are beneficial for the Seq2Seq tasks we consider.[4] We refer to this model as **PCRF-Seq2Seq** in the remainder.

Features Conditional random fields are feature-based, so we need to decide which features we use. In view of the end-to-end nature of neural techniques, requiring little linguistic knowledge, we also minimize feature-engineering effort for the traditional approaches and thus only include very simple features. For each position p to tag, we include all consecutive character m-grams (m ranges from 1 to a maximum order of N) within a *window* of size w around p; i.e., in total our window covers $2w + 1$ positions. In our experiments below, we report results for windows of size $w = 4$ and $w = 6$. For simplicity, we set $N = w$ in each case.

4.3 Further Baseline Systems

Considering the similarity of G2P conversion, spelling correction, and lemmatization with regard to their innate monotonicity (Eger et al., 2016; Nicolai et al., 2015; Eger, 2015), we explore for all our datasets three further approaches that were originally designed for G2P conversion.

Sequitur (Bisani and Ney, 2008) is a 'joint' model for Seq2Seq in the sense of the classic distinction between joint and discriminative models. Its core architecture is a model over 'joint n-grams', also termed 'graphones' in the original publication (that is, pairs of substrings of the $\vec{x}$ and $\vec{y}$ sequence).

DirecTL+ (Jiampojamarn et al., 2010) is a discriminative model for monotone Seq2Seq that integrates joint n-gram features. It jointly learns input segmentation, output prediction, and sequence modeling.

[4] In our experiments, higher order CRFs (> 3) substantially outperformed first-order models. Typical performance differences were from about 4 to 7% between first-order and fifth-order models. For brevity, we omit results for orders ≤ 3.

	Text+Berg (72K)	**Text+Berg (10K)**	**Twitter (31K)**	**Twitter (10K)**
attn-enc-dec (1 layer, size 100)	66.80%	61.71%	66.25%	60.99%
attn-enc-dec (1 layer, size 200)	68.29%	**63.00%**	67.81%	59.36%
attn-enc-dec (2 layers, size 100)	68.30%	62.27%	69.29%	63.31%
attn-enc-dec (2 layers, size 200)	69.74%	62.87%	69.70%	62.01%
enc-dec (1 layer, size 100)	50.96%	39.99%	60.91%	52.90%
enc-dec (1 layer, size 200)	53.65%	41.52%	63.39%	55.76%
enc-dec (2 layers, size 100)	56.94%	42.53%	65.94%	56.50%
enc-dec (2 layers, size 200)	59.01%	46.01%	63.70%	53.74%
morph-trans (1 layer, size 100)	55.96%	49.37%	41.46%	39.19%
morph-trans (1 layer, size 200)	54.22%	49.63%	36.28%	30.74%
morph-trans (2 layers, size 100)	56.11%	47.35%	44.42%	42.02%
morph-trans (2 layers, size 200)	49.27%	45.55%	30.33%	29.61%
PCRF-Seq2Seq (order 4, $w = 6$)	**74.67%**	62.24%	73.52%	59.97%
PCRF-Seq2Seq (order 5, $w = 6$)	74.22%	62.47%	74.03%	60.19%
PCRF-Seq2Seq (order 4, $w = 4$)	74.55%	62.75%	**74.87%**	**63.59%**
Phonetisaurus ($n = 8$)	60.89%	51.84%	69.52%	55.76%
Sequitur	68.04%	57.30%	70.74%	58.90%

Table 2: Word accuracies (WACs) for all encoder-decoder models, PCRF-Seq2Seq, and baselines for the **OCR post-correction** task and for the **spelling correction** task. Best configurations for each model are underlined, overall best results are bold-faced.

Since it is based on ordinary CRFs, it is virtually impossible to use this system with higher orders for all practically relevant datasets due to very long training times. Moreover, the system is generally very slow because it jointly learns to segment and translate, as mentioned. For this reason, we have only tested it on the Combilex dataset (Table 3), run with comparable parametrizations (context size, etc.) as PCRF-Seq2Seq.

Phonetisaurus (Novak et al., 2012) implements a weighted finite state transducer (WFST) to align input and output tokens. The EM-driven algorithm is capable of learning multiple-to-multiple alignments where we restrict both sides to a maximum of 2. The alignments learned from the training data are subsequently used to train a character-based n-gram language model. For brevity, we only report results for models with $n = 8$, which outperformed lower-order models in our experiments.

5 Results and Analysis

5.1 Model Performances

We report the results of all our experiments in terms of *word accuracy* (WAC), i.e., the fraction of completely correctly predicted output sequences. Table 2 lists WACs for all our systems on the OCR post-correction task (*Text+Berg*, full and reduced training set) and on the spelling correction task (Twitter, full and reduced training set). Table 3 reports WAC of all tested models on the Combilex dataset with models trained on training sets of different sizes. Table 4 reports WAC for the lemmatization task on the morphology dataset.

For the encoder-decoder models, we report the results with one and two layers of sizes 100 and 200 each. We have additionally conducted sample experiments with larger networks which have shown that neither increasing the number of layers nor the size of the layers leads to further improvements. For the PCRF-Seq2Seq models, we report results for windows of sizes $w = 4$ and $w = 6$. We note that, a priori, more training data tends to favor larger context size w, whereas a large w may lead to overfitting when training data is small. The same holds for model order.

	Combilex (20K)	**Combilex (10K)**	**Combilex (5K)**	**Combilex (2K)**
attn-enc-dec (1 layer, size 100)	57.39%	54.40%	41.19%	35.68%
attn-enc-dec (1 layer, size 200)	61.86%	57.92%	46.72%	38.31%
attn-enc-dec (2 layers, size 100)	66.74%	60.52%	55.89%	44.13%
attn-enc-dec (2 layers, size 200)	67.36%	59.62%	55.07%	44.26%
enc-dec (1 layer, size 100)	54.03%	48.25%	36.62%	18.17%
enc-dec (1 layer, size 200)	55.27%	49.81%	36.19%	18.41%
enc-dec (2 layers, size 100)	57.77%	51.91%	38.97%	16.68%
enc-dec (2 layers, size 200)	56.95%	50.69%	39.01%	17.83%
morph-trans (1 layer, size 100)	48.82%	43.30%	33.63%	18.97%
morph-trans (1 layer, size 200)	49.72%	43.15%	32.42%	18.76%
morph-trans (2 layers, size 100)	49.58%	42.05%	28.69%	15.13%
morph-trans (2 layers, size 200)	44.36%	35.14%	23.08%	12.74%
PCRF-Seq2Seq (order 4, $w = 6$)	72.14%	64.39%	55.66%	42.82%
PCRF-Seq2Seq (order 5, $w = 6$)	72.23%	64.32%	55.58%	42.62%
PCRF-Seq2Seq (order 4, $w = 4$)	71.74%	64.71%	**56.89%**	**44.74%**
DirecTL+	72.23%	**65.09%**	55.75%	42.95%
Phonetisaurus ($n = 8$)	**72.29%**	64.14%	55.28%	42.21%
Sequitur	70.57%	62.57%	54.03%	41.94%

Table 3: WACs for all encoder-decoder models, PCRF-Seq2Seq, and baselines for the **G2P** task. Best configurations for each model are underlined, overall best results are bold-faced.

While more training data obviously increases WAC for every model, the specific impact varies. In general, (attention-based) encoder-decoder models deal relatively well with limited test data in our experiments, achieving WACs comparable to PCRF-Seq2Seq. In contrast, they appear to benefit less from increasing data sizes than CRFs do. On the Twitter dataset, for instance, the best-performing encoder-decoder model increases WAC by 7.7 percentage points when tripling the training data size. At the same time, the 5th-order PCRF-Seq2Seq WAC increases by 13.8. When a large amount of training data is available, CRFs therefore consistently outperform neural models, and so do the specialized baseline systems on the G2P conversion tasks (Table 3).

Summarizing, we find that PCRF-Seq2Seq performs best among the tested systems for the two spelling correction tasks when large training data is available. The best performance of PCRF-Seq2Seq is roughly 6-7 percentage points better than the best performance of an encoder-decoder model for both Twitter 31K and Text+Berg 72K. For small training set sizes, PCRF-Seq2Seq and the encoder-decoder models are on a similar level. For the G2P task, an analogous pattern emerges. Moreover, here, all classical systems appear to perform similarly, with DirecTL+ and PCRF-Seq2Seq marginally outperforming the others. For lemmatization, the overall picture looks different. For Finnish verbs, we observe the only case in which attention-based encoder-decoder systems clearly outperform all other approaches. For German, neural models also achieve the best results, albeit only marginally above PCRF-Seq2Seq.

Previous works that employed encoder-decoder models successfully focused on tasks like machine translation and grammar correction in which more challenging linguistic phenomena such as long-range dependencies and 'crossing edges' (re-ordering) occur frequently. In our experiments, too, neural models only outperform traditional ones when long-range dependencies become relevant, namely in lemmatization. In all other tasks at hand, in contrast, neural models perform worse or equal.

The afore-mentioned, more complex linguistic phenomena intuitively require a more global view on long input sequences which is hard to impossible to model for approaches that cannot look beyond a statically defined context. Spelling mistakes, OCR errors, and G2P, however, largely depend on a very local context. For instance, OCR systems typically do not consider more than a small context when

	German Verbs (44K)	Finnish Verbs (41K)
attn-enc-dec (1 layer, size 100)	93.67%	**98.00%**
attn-enc-dec (1 layer, size 200)	93.17%	96.92%
attn-enc-dec (2 layers, size 200)	92.39%	97.08%
attn-enc-dec (2 layers, size 200)	**94.83%**	96.42%
enc-dec (1 layer, size 100)	77.31%	94.83%
enc-dec (1 layer, size 200)	82.00%	94.67%
enc-dec (2 layers, size 200)	79.50%	95.67%
enc-dec (2 layers, size 200)	76.30%	95.58%
morph-trans (1 layer, size 100)	91.89%	96.17%
morph-trans (1 layer, size 200)	93.24%	96.75%
morph-trans (2 layers, size 200)	93.02%	97.08%
morph-trans (2 layers, size 200)	93.63%	96.75%
PCRF-Seq2Seq (order 4, $w = 6$)	94.22%	94.08%
PCRF-Seq2Seq (order 5, $w = 6$)	93.77%	94.00%
PCRF-Seq2Seq (order 4, $w = 4$)	93.44%	93.33%
Phonetisaurus ($n = 8$)	86.62%	93.42%
Sequitur	85.63%	92.92%

Table 4: WACs for all encoder-decoder models, PCRF-Seq2Seq, and baselines for the **lemmatization task**. Best configurations for each model are underlined, overall best results are bold-faced.

estimating the probability of a character. Regarding the G2P task, phonetics is generally independent of characters that occur more than two or three positions before of after, at least in most cases and in English. The same is presumably true for human typos, where a mistaken key stroke may be the result of a previous key's position, but does not correlate to any key that was hit several time steps before. Hence, neural networks are unable to benefit from their often advantageous capability of modeling long-range dependencies here.

Especially the Finnish lemmatization experiments confirm that the capability of dealing with long-range dependencies plays an important role. Finnish *vowel harmony* makes a vowel control other vowels in the word, potentially across multiple syllables; see Faruqui et al. (2016) for more detailed explanation.

As a side note, our results are in line with the common notion that the specific impact of a neural network's size (number and sizes of layers) is almost unpredictable. Our results can only confirm the general rule-of-thumb that larger networks are better for larger training sets, while models with fewer parameters outperform larger ones when training data is smaller.

5.2 Training Time

Another potentially limiting factor for the applicability of a model in real-world scenarios, especially for large datasets, is training time. Under all circumstances, weighted finite state transducers (Phonetisaurus) are trained by magnitudes faster than all other approaches. Training times range between as little as 6 seconds for the smallest training set and 247 seconds for the full *Text+Berg* training set (72K entries).

In comparison, training times for the encoder-decoder models range from 2 to 80 hours (without using GPUs) for 30 epochs, depending on the sizes of the networks and the training data. Furthermore, there is no noticeable difference between either of the three encoder-decoder variations. Training time increases approximately linearly with the number of layers, the size of the layers, and the training data size. All these factors add up, meaning that doubling both the number of layers and the size of the layers approximately quadruples training time.

Contrasting DirecTL+ with PCRF-Seq2Seq, both of which rest on similar principles and also perform similarly in our experiments on the G2P task, we find that training PCRF-Seq2Seq was a factor of 30 or 50 times faster than DirecTL+ on Combilex (2K) and Combilex (5K), respectively. In general, training

for PCRF-Seq2Seq across our datasets was in the order of minutes to (few) hours.

5.3 Error Analysis

We divided three of our test sets (*Text+Berg*, Twitter, and Combilex) by input string lengths and evaluated PCRF-Seq2Seq and encoder-decoder neural models on these subsets of the test data. As illustrated in Figures 2 and 3, we observe a consistent tendency: PCRF-Seq2Seq performs relatively robustly over input strings of different lengths, while the performance of the encoder-decoder models plummets more drastically with sequences becoming longer, in particular those without attention-mechanism.

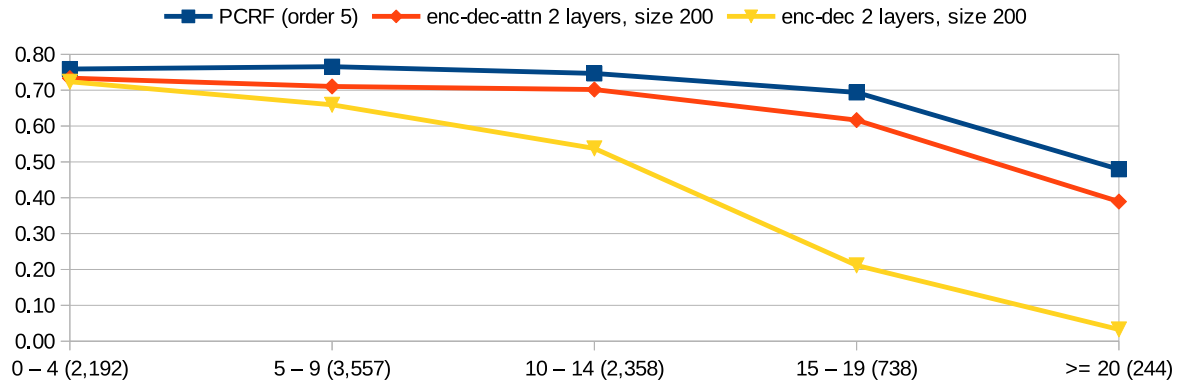


Figure 2: WAC of PCRF-Seq2Seq and encoder-decoder neural models with and without attention-based mechanisms as a function of input string length (number of training samples) on *Text+Berg* **OCR post-correction**.

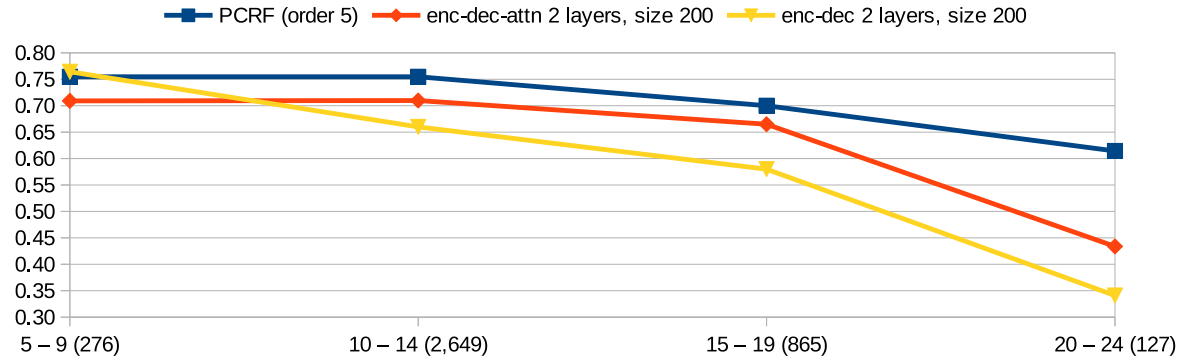


Figure 3: WAC of PCRF-Seq2Seq and encoder-decoder neural models with and without attention-based mechanisms as a function of input string length (number of training samples) on the Twitter **spelling correction** task.

For shorter sequences, we observe that standard encoder-decoder models even slightly outperform their attention-based counterparts as well as PCRF-Seq2Seq on both the Twitter spelling correction task (Figure 3) and on G2P conversion, in contrast to their rather low performance on the full datasets. On the *Text+Berg* data, all systems achieve approximately equal WAC for short sequences (Figure 2).

For longer sequences, the performance of the encoder-decoder models drops dramatically on all data sets. This effect is also visible, albeit less strong, for the attention-based variant. This can be seen particularly well on the OCR post-correction task (Figure 2), where the test set contains numerous long sequences: the accuracy rate for the standard encoder-decoder model drops from 73.32% on very short sequences to below 10% for very long ones ($\geq$ 20), whereas the attention-based model drops less drastically to 38.93% (from 76.92%). At the same time, PCRF-Seq2Seq behaves more stably, particularly on the Twitter data (Figure 3). For the Combilex data, the picture looks very similar — we omit these results for brevity.

6 Conclusions

The generality of neural networks makes them appealing for a wide range of possible tasks. In the scope of this work, we have applied encoder-decoder neural models to monotone Seq2Seq tasks. We have shown that they can perform comparably to more specialized models in some cases, but cannot (yet) consistently outperform established approaches, and are sometimes still substantially below them. Furthermore, the advantage of having rendered feature engineering and hyper-parameter optimization in the traditional sense unnecessary is notoriously substituted by the search for optimal neural network topologies.

At first sight, our analyses based on string lengths are in line with those reported by Bahdanau et al. (2014). They state that — for the field of machine translation — the attention mechanism leads to improvements over the standard encoder-decoder model on longer sentences. We also observe this positive impact for our tasks, where the attention-based mechanism alleviates the drastic performance drop of the standard encoder-decoder models on long sequences to some extent. At the same time, we see that very performance drop persisting — CRFs still outperform encoder-decoder models on long sequences, even when employing attention-mechanisms. As described in Section 5.3, neural models are only able to successfully compete when more complex phenomena occur, on which traditional models fail. Nevertheless, previous works such as Vukotic et al. (2015) also indicate that even in more complex sequence labeling tasks such as spoken language understanding, neural networks are not guaranteed to outperform CRFs.

The task-specific extensions to the encoder-decoder proposed by Faruqui et al. (2016) have been shown to produce mostly bad results in our settings. This is particularly surprising for the OCR data, for which input and output sequences are usually very similar, so that we had expected that re-feeding the input to the decoder should be equally beneficial in that domain. As discussed, one explanation might be that OCR, or spelling correction generally, putatively exhibits few long-range dependencies. This might explain why the `morph-trans` approach works quite well and competitive in morphological analysis tasks, as re-confirmed in our experiments. Thus, long-range dependencies might actually be a more crucial aspect for the performance of the model presented by Faruqui et al. (2016) than the similarity between input and output sequence.

We conclude that neural networks are far from completely replacing established methods at this point, as the latter can be both faster and more accurate, depending on the properties of the task at hand. A systematic analysis of the complexities and challenges a particular task imposes, remains unavoidable. At the same time, one can argue that encoder-decoder neural models are a relatively recent development and might continue to improve much over the next years. Being very generic and largely task-agnostic, they are already able to outperform traditional and specialized approaches under certain circumstances.

Acknowledgements

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant № I/82806, and by the German Institute for Educational Research (DIPF), as part of the graduate program "Knowledge Discovery in Scientific Literature" (KDSL).

References

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, Gothenburg, Sweden 26–30 April 2014*, pages 569–578.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*, September.

Maximilian Bisani and Hermann Ney. 2008. Joint-Sequence Models for Grapheme-to-Phoneme Conversion. *Speech Communication*, 50(5):434–451, May.

Eric Brill and Robert C. Moore. 2000. An Improved Error Model for Noisy Channel Spelling Correction. In *Proceedings of ACL '00*, pages 286–293, Stroudsburg, PA, USA. Association for Computational Linguistics.

Noah Bubenhofer, Martin Volk, Fabienne Leuenberger, and Daniel Wüest, editors. 2015. *Text+Berg-Korpus (Release 151_v01)*. Institut für Computerlinguistik, Universität Zürich. Digitale Edition des Jahrbuch des SAC 1864–1923, Echo des Alpes 1872–1924, Die Alpen, Les Alpes, Le Alpi 1925–2014, The Alpine Journal 1969–2008. Published: XML-Format.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of ACL '05*, pages 173–180, Ann Arbor, MI, USA. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078 [cs, stat]*, June.

Grzegorz Chrupała. 2014. Normalizing Tweets with Edit Scripts and Recurrent Neural Embeddings. In *Proceedings of ACL '14*, pages 680–686, Baltimore, MD, USA. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *The Journal of Machine Learning Research*, 12:2493–2537, February.

Silviu Cucerzan and Eric Brill. 2004. Spelling Correction as an Iterative Process that Exploits the Collective Knowledge of Web Users. In *Proceedings of EMNLP '04*, pages 293–300, Barcelona, Spain. Association for Computational Linguistics.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of NAACL-HLT '13*, pages 1185–1195, Atlanta, GA, USA. Association for Computational Linguistics.

Steffen Eger, Tim vor der Brück, and Alexander Mehler. 2016. A Comparison of Four Character-Level String-to-String Translation Models for (OCR) Spelling Error Correction. *The Prague Bulletin of Mathematical Linguistics*, 105(1):77–99, April.

Steffen Eger. 2015. Designing and comparing g2p-type lemmatizers for a morphology-rich language. In *Systems and Frameworks for Computational Morphology - Fourth International Workshop, SFCM 2015, Stuttgart, Germany, September 17-18, 2015, Proceedings*, pages 27–40.

Noura Farra, Nadi Tomeh, Alla Rozovskaya, and Nizar Habash. 2014. Generalized Character-Level Spelling Error Correction. In *Proceedings of ACL '14*, pages 161–167, Baltimore, MD, USA. Association for Computational Linguistics.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological Inflection Generation Using Character Sequence to Sequence Learning. In *Proceedings of NAACL-HLT '16*, pages 634–643, San Diego, CA, USA. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Sergey Gubanov, Irina Galinskaya, and Alexey Baytin. 2014. Improved Iterative Correction for Distant Spelling Errors. In *Proceedings of ACL '14*, pages 168–173, Baltimore, MD, USA. Association for Computational Linguistics.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2010. Integrating Joint n-gram Features into a Discriminative Training Framework. In *Proceedings of NAACL-HLT '10*, pages 697–700, Los Angeles, CA, USA. Association for Computational Linguistics.

John Kominek and Alan W Black. 2004. The CMU Arctic speech databases. In *Fifth ISCA Workshop on Speech Synthesis*, pages 223–224, Pittsburgh, PA, USA.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML '01*, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of EMNLP '15*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient Higher-Order CRFs for Morphological Tagging. In *Proceedings of EMNLP '13*, pages 322–332, Seattle, WA, USA. Association for Computational Linguistics.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *Proceedings of NAACL-HLT '15*, pages 922–931, Denver, CO, USA. Association for Computational Linguistics.

Josef R. Novak, Nobuaki Minematsu, and Keikichi Hirose. 2012. WFST-Based Grapheme-to-Phoneme Conversion: Open Source tools for Alignment, Model-Building and Decoding. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, pages 45–49, Donostia, Spain. Association for Computational Linguistics.

Naoaki Okazaki, Yoshimasa Tsuruoka, Sophia Ananiadou, and Jun'ichi Tsujii. 2008. A Discriminative Candidate Generator for String Transformations. In *Proceedings of EMNLP '08*, pages 447–456, Honolulu, HI, USA. Association for Computational Linguistics.

Stephan Raaijmakers. 2013. A Deep Graphical Model for Spelling Correction. In *Proceedings of the 25th Benelux Conference on Artificial Intelligence*, pages 160–167, Delft, Netherlands. Delft University of Technology.

Kanishka Rao, Fuchun Peng, Hasim Sak, and Françoise Beaufays. 2015. Grapheme-to-Phoneme Conversion Using Long Short-Term Memory Recurrent Neural Networks. In *Proceedings of ICASSP '15*, pages 4225–4229, South Brisbane, QLD, Australia. Institute of Electrical and Electronics Engineers.

Ulrich Reffle and Christoph Ringlstetter. 2013. Unsupervised Profiling of OCRed Historical Documents. *Pattern Recognition*, 46(5):1346–1357, May.

Martin Reynaert. 2014. On OCR Ground Truths and OCR Post-correction Gold Standards, Tools and Formats. In *Proceedings of DATeCH '14*, pages 159–166, New York, NY, USA. ACM.

Korin Richmond, Robert A.J. Clark, and Susan Fitt. 2009. Robust LTS rules with the Combilex speech technology lexicon. In *Proceedings of INTERSPEECH '09*, pages 1295–1298, Brighton, UK. ISCA.

Allen Schmaltz, Yoon Kim, Alexander M Rush, and Stuart M Shieber. 2016. Sentence-Level Grammatical Error Identification as Sequence-to-Sequence Correction. *arXiv preprint arXiv:1604.04677*.

Tarek Sherif and Grzegorz Kondrak. 2007. Substring-Based Transliteration. In *Proceedings of ACL '07*, pages 944–951, Prague, Czech Republic. Association for Computational Linguistics.

Uwe Springmann, Dietmar Najock, Hermann Morgenroth, Helmut Schmid, Annette Gotscharek, and Florian Fink. 2014. OCR of Historical Printings of Latin Texts: Problems, Prospects, Progress. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, pages 71–75, New York, NY, USA. ACM.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of Neural Information Processing Systems 2014*, pages 3104–3112, Montréal, Québec, Canada. Curran Associates, Inc.

Kristina Toutanova and Robert C. Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of ACL '02*, pages 144–151, Philadelphia, PA, USA. Association for Computational Linguistics.

Oriol Vinyals and Quoc Le. 2015. A Neural Conversational Model. In *Proceedings of the 31st International Conference on Machine Learning, JMLR: W&CP*, volume 37, Lille, France.

Vedran Vukotic, Christian Raymond, and Guillaume Gravier. 2015. Is it time to switch to Word Embedding and Recurrent Neural Networks for Spoken Language Understanding? In *InterSpeech-2015*, pages 130–134, Dresden, Germany.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural Language Correction with Character-Based Attention. *arXiv preprint arXiv:1603.09727*.

Gu Xu, Hang Li, Ming Zhang, and Ziqi Wang. 2014. A Probabilistic Approach to String Transformation. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1063–1075, May.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-Sequence Neural Net Models for Grapheme-to-Phoneme Conversion. In *Proceedings of INTERSPEECH '15*, pages 3330–3334, Dresden, Germany. ISCA.

Wenpeng Yin, Sebastian Ebert, and Hinrich Schütze. 2016. Attention-Based Convolutional Neural Network for Machine Comprehension. In *Proceedings of the Workshop on Human-Computer Question Answering*, pages 15–21, San Diego, CA, USA. Association for Computational Linguistics.

Towards Time-Aware Knowledge Graph Completion

Tingsong Jiang, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li and **Zhifang Sui**
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
Collaborative Innovation Center for Language Ability, Xuzhou 221009 China
{tingsong,tianyu0421,taoge,shalei,chbb,lisujian,szf}@pku.edu.cn

Abstract

Knowledge graph (KG) completion adds new facts to a KG by making inferences from existing facts. Most existing methods ignore the time information and only learn from time-unknown fact triples. In dynamic environments that evolve over time, it is important and challenging for knowledge graph completion models to take into account the temporal aspects of facts. In this paper, we present a novel *time-aware* knowledge graph completion model that is able to predict links in a KG using both *the existing facts* and *the temporal information of the facts*. To incorporate the happening time of facts, we propose a time-aware KG embedding model using temporal order information among facts. To incorporate the valid time of facts, we propose a joint time-aware inference model based on Integer Linear Programming (ILP) using temporal consistency information as constraints. We further integrate two models to make full use of global temporal information. We empirically evaluate our models on time-aware KG completion task. Experimental results show that our time-aware models achieve the state-of-the-art on temporal facts consistently.

1 Introduction

Knowledge graphs (KGs) such as Freebase (Bollacker et al., 2008) and YAGO (Fabian et al., 2007) are extremely useful resources for many NLP related applications such as relation extraction and question answering, etc. Although KGs are large in size, they are far from complete (West et al., 2014). Knowledge graph completion, i.e., automatically inferring missing facts between entities in a knowledge graph, has thus become an increasingly important task. Recently a promising approach called KG embedding aims to embed the components (entities and relations) of a KG into a continuous vector space while preserving the inherent structure of a knowledge graph (Nickel et al., 2011; Bordes et al., 2011). This kind of approach has shown good effectiveness and scalability for KG completion.

However, most existing KG embedding models ignore the temporal information of facts. In the real world, many facts are not static but highly ephemeral. For example, (*Steve_Jobs,* `diedIn`*, California*) happened on 2011-10-05; (*Ronaldo,* `playsFor`*, A.C.Milan*) is true only during 2007-2008. Intuitively, temporal aspects of facts should play an important role when we perform KG completion. In this paper, we focus on time-aware KG completion. Specially, we incorporate two kinds of temporal information for KG completion: (a) *temporal order information* and (b) *temporal consistency information*. By *temporal order information*, we mean that many facts have temporal dependencies on others according to the time that they happened. For example, the facts involving a person *P* may follow the following timeline: (*P,* `wasBornIn`*,* _) → (*P,* `graduateFrom`*,* _) → (*P,* `workAt`*,* _) → (*P,* `diedIn`*,* _). Given the time after *P* died , it's not proper to predict relations like `workAt`. By *temporal consistency information*, we mean that many facts are only valid during a short time period. For example, a person's marriage may be valid for a short period. Besides, the periods of a person's different marriages should not overlap. Without considering the temporal aspects of facts, the existing KG embedding methods may make mistakes. It is also non-trivial for existing KG embedding methods to incorporate such temporal information.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1715–1724, Osaka, Japan, December 11-17 2016.

To deal with the issues of the existing KG embedding methods, we propose two *time-aware KG completion* models to incorporate the above two kinds of temporal information, respectively. The extensive experimental results show the effectiveness of the two proposed models. We further propose a joint model that achieves better results. Our contributions include the following:

- To the best of our knowledge, this is the first work for time-aware KG completion. To incorporate the temporal order information, we propose a novel time-aware embedding (TAE) model that encodes the temporal order information as a regularizer on the geometric structure of the embedding space. To incorporate more temporal consistency information, we propose using Integer Linear Programming (ILP) to encode the temporal consistency information as constraints.
- We further propose a joint framework to unify the two complementary time-aware models seamlessly. ILP model considers more temporal constraints than TAE model, while TAE model generates more accurate embeddings for the objective function of ILP model. Our framework can be generalized to many KG embedding models such as TransE (Bordes et al., 2013) model and its extensions.
- We create real-world temporal data sets based on YAGO2 and Freebase for time-aware KG completion. The evaluation results show that our models outperform the start-of-the-art approaches and it confirms the effectiveness of incorporating temporal information.

The rest of the paper is organized as follows. Section 2 and Section 3 describe two time-aware KG completion models, respectively. Experiments, related work and conclusion are shown in Section 4-6.

2 Time-Aware KG Embedding Model

Time-aware KG embedding aims to automatically learn entity and relation embeddings by exploiting both observed triple facts and temporal order information among facts.

2.1 Time-Aware KG Completion Task

We represent facts with temporal annotations by quadruples, *quads* for short. We use (e_i, r, e_j, t) to denote the fact that e_i and e_j has relation r during the time interval $t = [t_b, t_e]$ with $t_b < t_e$. Although our reasoning framework supports arbitrary continuous intervals over real number, for simplicity, we assume time intervals range over **years**. For example, the interval [1980, 1999] starts in 1980 and ends in 1999. For some facts that happened at a certain time and did not last, we have $t_b = t_e$. For some facts that does not end yet, we represent t as $t = [t_b, +\infty]$.

KG completion is the task of predicting whether a given edge (e_i, r, e_j) exists in the graph or not. However, most facts are time-dependent and hold only for a given time period. For example, the fact of George W. Bush's presidency is only meaningful from 2001 to 2009. To incorporate temporal information for a more accurate representation, we extend this task to include the time dimension of the facts and call it *time-aware KG completion*, i.e., to complete the quad (e_i, r, e_j, t) when e_i, r or e_j is missing given a specific time interval t. For example, we can answer the question *"Who is the president of USA in 2010?"* by predicting head entity in *(?,* `presidentOf` *, USA*, [2010,2010]).

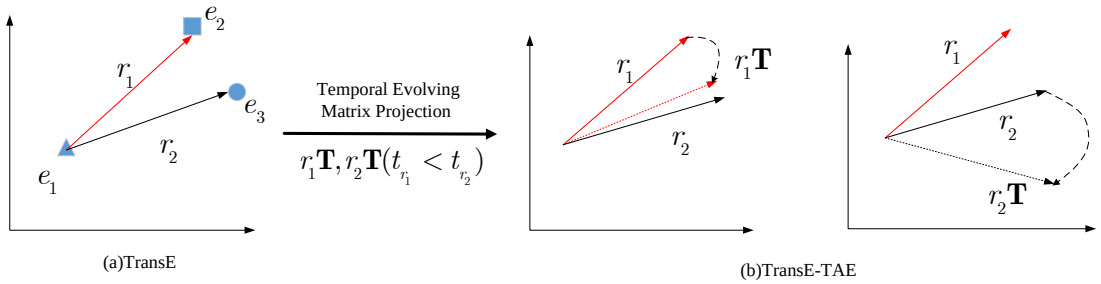
2.2 Traditional KG Embedding Methods

Traditional KG embedding methods use only the observed time-unknown facts (triples) to learn entity and relation representations. TransE (Bordes et al., 2013) is an efficient and simple model among them. The basic idea behind TransE is that the relation between two entities $\mathbf{e}_i, \mathbf{e}_j \in \mathbb{R}^n$ corresponds to a translation vector $\mathbf{r} \in \mathbb{R}^n$ between them, i.e., $\mathbf{e}_i + \mathbf{r} \approx \mathbf{e}_j$ when (e_i, r, e_j) holds. The scoring function is defined as measuring its plausibility in the vector space:

$$f(e_i, r, e_j) = \|\mathbf{e}_i + \mathbf{r} - \mathbf{e}_j\|_{\ell_1/\ell_2}, \tag{1}$$

where $\|\cdot\|_{\ell_1/\ell_2}$ denotes the ℓ_1-norm or ℓ_2-norm. A margin-based ranking loss is optimized to derive the entity and relation representations:

$$\min \sum_{x^+ \in \Delta} \sum_{x^- \in \Delta'} [\gamma + f(x^+) - f(x^-)]_+. \tag{2}$$



(a)TransE (b)TransE-TAE

Figure 1: Simple illustration of Temporal Evolving Matrix $\mathbf{T}$ in the time-aware embedding (TAE) space. For example, r_1=$wasBornIn$ happened before r_2=$diedIn$. After projection by $\mathbf{T}$, we get prior relation's projection $\mathbf{r}_1\mathbf{T}$ near subsequent relation $\mathbf{r}_2$ in the space, i.e.,$\mathbf{r}_1\mathbf{T} \approx \mathbf{r}_2$, but $\mathbf{r}_2\mathbf{T} \neq \mathbf{r}_1$.

Here, $x^+ \in \Delta$ is the observed (i.e., positive) triple, and $x^- \in \Delta'$ is the negative triple constructed by replacing entities in x^+. γ is the margin separating positive and negative triples and $[z]_+ = \max(0, z)$. Please refer to (Wang et al., 2014a; Lin et al., 2015b) for TransH, TransR and other models.

After we obtain the embeddings, the plausibility of a missing triple can be predicted by using the scoring function. In general, triples with higher plausibility are more likely to be true.

2.3 Time-Aware KG Embedding Model

TransE assumes that each relation is time independent and entity/relation representation is only affected by structural patterns in KGs. To better model knowledge evolution, we assume temporal ordered relations are related to each other and evolve in a time dimension. For example, for the same person, there exists a temporal order among relations `wasBornIn`→ `graduatedFrom` → `diedIn`. In time dimension, `wasBornIn` can evolve into `graduateFrom` and `diedIn`, but `diedIn` cannot evolve into `wasBornIn`.

To compare temporal orders, we define a pair of temporal ordering relations sharing the same head entity [1] as **temporal ordering relation pair**, e.g., ⟨`wasBornIn`, `diedIn`⟩. We define the relation happening earlier, e.g., `wasBornIn`, as **prior relation** and the other as **subsequent relation**. We define ⟨prior relation, subsequent relation⟩ as positive temporal ordering pairs and ⟨subsequent relation, prior relation⟩ as negative ones.

To capture the temporal order of relations, we further define a *temporal evolving matrix* $\mathbf{T} \in \mathbb{R}^{n \times n}$ to model relation evolution, where n is the dimension of relation embedding. $\mathbf{T}$ is a parameter to be learned by the model from the data. We assume that prior relation can evolve into subsequent relation through the temporal evolving matrix. The more frequent they have temporal orders, the more they can evolve. Specially, as in Figure 1, prior relation $\mathbf{r}_1$ projected by $\mathbf{T}$ should be near subsequent relation $\mathbf{r}_2$, i.e., $\mathbf{r}_1\mathbf{T} \approx \mathbf{r}_2$, while $\mathbf{r}_2\mathbf{T}$ should be far from $\mathbf{r}_1$. In this way, we are able to separate prior relation and subsequent relation automatically during training.

We formulate time-aware KG completion as an optimization problem based on a regularization term. Given any positive training quad $(e_i, r_k, e_j, t_{r_k}) \in \Delta_t$, we can find a temporally related quad $(e_i, r_l, e_m, t_{r_l}) \in \Delta_t$ sharing the same *head entity* and a temporal ordering relation pair $\langle r_k, r_l \rangle$. If $t_{r_k} < t_{r_l}$, we have a positive temporal ordering relation pair $y^+ = \langle r_k, r_l \rangle$ and the corresponding negative relation pair $y^- = \langle r_k, r_l \rangle^{-1} = \langle r_l, r_k \rangle$ by inverse. Our optimization requires that positive temporal ordering relation pairs should have lower scores (energies) than negative pairs. Therefore, we define a **temporal** scoring function as

$$g(\langle r_k, r_l \rangle) = \|\mathbf{r}_k\mathbf{T} - \mathbf{r}_l\|_{\ell_1/\ell_2}, \tag{3}$$

which is expected to give a low score when the temporal ordering relation pair is in chronological order, and a high score otherwise. Note that $\mathbf{T}$ is asymmetric and the loss function is also asymmetric so as to capture temporal order information.

[1] We only consider relations sharing the same head entity because most temporal facts and temporal relations are partially ordered around a common *protagonist* (usually the head entity), e.g., "wasBornIn", "workAt", and "diedIn" are temporally ordered with a common person. Temporal relations that are ordered with a common tail entity could be transformed by replacing the relation with its inverse relation and exchanging the head and tail entity.

To make the embedding space compatible with the observed triples, we make use of the fact triples set Δ and follow the same strategy adopted in previous methods. Specially, we apply the same **fact** scoring function $f(e_i, r_k, e_j)$ in Equation (1) to each candidate triple. The optimization is to minimize the **joint** scoring function,

$$L = \sum_{x^+ \in \Delta} \Big[\sum_{x^- \in \Delta'} [\gamma_1 + f(x^+) - f(x^-)]_+ + \lambda \sum_{y^+ \in \Omega_{e_i,t_{r_k}}, y^- \in \Omega'_{e_i,t_{r_k}}} [\gamma_2 + g(y^+) - g(y^-)]_+ \Big], \tag{4}$$

where $x^+ = (e_i, r_k, e_j) \in \Delta$ is a positive triple, $x^- = (e'_i, r_k, e'_j) \in \Delta'$ is the corresponding negative triple by replacing entities. The positive temporal ordering relation pair set with respect to (e_i, r_k, e_j, t_{r_k}) is defined as

$$\begin{aligned} \Omega_{e_i,t_{r_k}} = \{\langle r_k, r_l \rangle | (e_i, r_k, e_j, t_{r_k}) \in \Delta_t, (e_i, r_l, e_m, t_{r_l}) \in \Delta_t, t_{r_k} < t_{r_l}\} \\ \cup \{\langle r_l, r_k \rangle | (e_i, r_k, e_j, t_{r_k}) \in \Delta_t, (e_i, r_l, e_m, t_{r_l}) \in \Delta_t, t_{r_k} > t_{r_l}\} \end{aligned} \tag{5}$$

where r_k and r_l share the same head entity e_i. $\Omega'_{e_i,t_{r_k}}$ are the corresponding negative relation pairs by inverse the relation pairs. In experiments, our constrains are $\|\mathbf{e}_i\|_2 \leq 1$, $\|\mathbf{r}_k\|_2 \leq 1$, $\|\mathbf{r}_l\|_2 \leq 1$, $\|\mathbf{e}_j\|_2 \leq 1$, $\|\mathbf{r}_k\mathbf{T}\|_2 \leq 1$, and $\|\mathbf{r}_l\mathbf{T}\|_2 \leq 1$ to avoid overfitting similarly to previous work.

The first term in Equation (1) enforces the generated embedding space compatible with all the observed triples, and the second term further requires the space to be temporally consistent and more accurate. Hyperparameter λ strikes a trade-off between the two cases. Stochastic gradient descent (in mini-batch mode) is adopted to solve the minimization problem.

3 Joint Inference for Time-Aware KG Completion

In this section, we incorporate temporal information as temporal consistency constraints for KG completion. We take advantage of temporal logic transitivity and use ILP to derive more accurate predictions.

3.1 Temporal Consistency Constraints

The candidate predictions we obtained in the traditional KG embedding inevitably include many incorrect predictions. By applying temporal consistency constraints, we can identify and then discard such errors to produce more accurate results.

As the complexity of resolving conflicts strictly depends on the constraints to apply, we need to choose them with great care. In the following, we consider three kinds of temporal constraints.

Temporal Disjointness. The time intervals of any two facts with a common functional relation and a common head entity are non-overlapping. For example, a person can only be spouse of one person at a time: $(e_1, \texttt{wasSpouseOf}, e_2, [1990, 2010)) \wedge (e_1, \texttt{wasSpouseOf}, e_3, [2005, 2013)) \wedge e_2 \neq e_3 \rightarrow$ false.

Temporal Ordering. For some temporal ordering relations, one fact always happens before another fact. For example, a person must be born before he graduated: $(e_1, \texttt{wasBornIn}, e_2, t_1) \wedge (e_1, \texttt{graduateFrom}, e_3, t_2) \wedge t_1 > t_2 \rightarrow$ false.

Temporal Spans. Some facts are true only during a specific time span. In general, the fact is invalid for other time periods outside the range of its time span in KGs. For example, given time interval t' outside the range t in $(e_1, \texttt{presidentOf}, e_2, t) \in$ KG, the fact $(e_1, \texttt{presidentOf}, e_2, t')$ is invalid.

3.2 Integer Linear Program Formulation

We formulate the time-aware inference as an ILP problem with temporal constraints. Traditional KG embedding methods can capture the intrinsic properties of data, which can be treated as a probability to predict unseen facts. For each candidate fact (e_i, r_k, e_j), we use $w_{ij}^{(k)} = f(e_i, r_k, e_j)$ to represent the plausibility predicted by an embedding model, and introduce a Boolean decision variable $x_{ij}^{(k)}$ to indicate whether the fact (e_i, r_k, e_j, t) is true or not for time t. Our aim is to find the best assignment to the decision variables, maximizing the overall plausibility while complying with all the temporal constraints. The objective function can be written as:

$$\max \sum_{x_{ij}^{(k)}} w_{ij}^{(k)} x_{ij}^{(k)}. \tag{6}$$

We add the constraints described in Section 3.1 for the above objective function.

The temporal disjointness constraints avoid the disagreement between the predictions of two facts sharing the same head entity and relation. These constraints can be represented as:

$$x_{ij}^{(k)} + x_{il}^{(k)} \leq 1, \forall k \in \mathcal{C}^d, t_{x_{ij}^{(k)}} \cap t_{x_{il}^{(k)}} \neq \emptyset \tag{7}$$

where $\mathcal{C}^d$ are functional relations described such as `wasSpouseOf` and $t_{x_{ij}^{(k)}}, t_{x_{il}^{(k)}}$ are time intervals for two facts, respectively.

The Temporal Ordering constraints ensure the occurring order for some relation pairs. These constraints can be represented as:

$$x_{ij}^{(k)} + x_{il}^{(k')} \leq 1, \forall (k, k') \in \mathcal{C}^o, t_{x_{ij}^{(k)}} \geq t_{x_{il}^{(k)}} \tag{8}$$

where $\mathcal{C}^o = \{\langle r_k, r'_k \rangle\}$ are relation pairs that have precedent orders such as $\langle$`wasBornIn`,`diedIn`$\rangle$. These relation pairs are discovered automatically in the training set by statistics and finally manually calibrated.

The temporal span constraints ensure the specific time span when the corresponding fact is true. These constraints can be represented as:

$$x_{ij}^{(k)} = 0, \forall k \in \mathcal{C}^s, t_{x_{ij}^{(k)}} \cap t_\Delta = \emptyset \tag{9}$$

where $\mathcal{C}^s$ are those relations valid for only a specific time span such as `presidentOf` and t_Δ is the valid time span in KG.

Using ILP, we can combine the ability of capturing the intrinsic properties of KG data and the temporal constraints that are embedded into global consistencies of the relations together. As shown in Eq.(10), any unseen fact's plausibility is encoded in scores w_{ij}^k which captures the intrinsic properties of KG data. Temporal consistency constraints are formulated as Eq.(7)-(9) and apply to the objective function naturally. By solving Eq.(10), we will obtain a list of selected candidate entities or relations for a missing fact as our final output.

3.3 Integrating Two Time-Aware Models

As mentioned above, the two time-aware models are complementary for each other: ILP model considers more temporal constraints than TAE model while TAE model generates more accurate embeddings for the ILP objective function.

For each unseen quad (e_i, r_k, e_j, t), we use a Boolean decision variable $x_{ij}^{(k,t)}$ to indicate whether it's true or not. We can use the embeddings of TAE model in Section 2.3 to calculate the plausibility $v_{ij}^{(k,t)}$ for the ILP objective function. The objective function is

$$\max \sum_{x_{ij}^{(k,t)}} v_{ij}^{(k,t)} x_{ij}^{(k,t)}. \tag{10}$$

Eq.(7)-(9) remain the same.

4 Experiments

We use similar evaluation metrics as traditional KG completion methods (Bordes et al., 2013) for time-aware KG completion.

4.1 Data Sets

To create temporal KG data sets, we need to decide whether a fact has temporal information. We categorize relations into time-sensitive relations and time-unsensitive relations according to YAGO2 (Hoffart et al., 2013). For example, `diedIn` is time-sensitive, but `hasNeighbor` is not. We extract temporal annotations for time-sensitive facts from YAGO2 and Freebase[2].

In YAGO2, temporal facts are in the form (*factID,occurSince,*t_b), (*factID,occurUntil,*t_e*)* indicating the fact is true during $[t_b, t_e]$. Here *factID* denotes a specific fact (e_i, r, e_j). We directly represent these temporal facts as quads $(e_i, r, e_j, [t_b, t_e])$. We selected 10 frequent time-sensitive relations to make a pure temporal data set. Then we selected the subset of entities which have at least two mentions in temporal

[2]www.freebase.com

Dataset	#Rel	#Ent	#Train/#Valid/#Test	#Quads
YG15k	10	9513	13345/1320/1249	15914
YG36k	10	9513	29757/3252/3058	15914
FB42	42	8376	23827/2173/2610	28610
FB87	87	8844	142598/14848/17566	175012

Table 1: Statistics of data sets.

DataSets	YG15k				YG36k				FB42				FB87			
Metric	MeanRank		Hits@10(%)		MeanRank		Hits@10(%)		MeanRank		Hits@10(%)		MeanRank		Hits@10(%)	
	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter
TransE	990	971	26.6	29.5	179	163	65.7	75.6	383	341	39.5	46.6	105	54	47.7	70.2
TransE-TAE	**245**	**244**	**34.4**	**35.3**	**62**	**58**	**75.4**	**81.9**	**328**	**300**	**45.0**	**51.3**	**92**	**50**	**53.8**	**77.5**
TransE-ILP	-	-	**40.5**	**41.9**	-	-	**80.1**	**85.4**	-	-	**53.5**	**55.2**	-	-	**62.1**	**82.7**
TransE-TAE-ILP	-	-	**44.8**	**46.1**	-	-	**84.7**	**89.4**	-	-	**65.1**	**70.2**	-	-	**65.7**	**85.3**
TransH	986	966	25.7	28.0	174	158	65.3	77.8	378	333	40.3	48.1	102	58	45.3	71.8
TransH-TAE	**243**	**241**	**33.4**	**34.7**	**63**	**58**	**75.3**	**81.6**	**320**	**291**	**46.4**	**52.7**	**93**	**52**	**55.3**	**78.5**
TransH-ILP	-	-	**41.7**	**42.6**	-	-	**81.5**	**85.6**	-	-	**53.7**	**56.4**	-	-	**63.5**	**81.1**
TransH-TAE-ILP	-	-	**43.3**	**46.6**	-	-	**85.4**	**88.7**	-	-	**65.3**	**71.4**	-	-	**67.2**	**86.0**
TransR	976	955	29.5	30.2	175	153	68.3	80.1	371	325	42.5	49.2	96	52	49.3	72.1
TransR-TAE	**253**	**251**	**33.5**	**33.9**	**56**	**45**	**79.5**	**86.9**	**318**	**282**	**47.2**	**54.9**	**88**	**47**	**56.5**	**79.9**
TransR-ILP	-	-	**41.9**	**44.3**	-	-	**82.6**	**82.5**	-	-	**57.8**	**57.1**	-	-	**63.4**	**87.9**
TransR-TAE-ILP	-	-	**45.4**	**47.7**	-	-	**85.8**	**89.5**	-	-	**66.5**	**72.3**	-	-	**68.2**	**88.2**

Table 2: Evaluation results on entity prediction.

facts. This resulted in 15,914 triples (quadruples) which were randomly split with the ratio shown in Table 1. This data set is denoted *YG15k*. Although YAGO2 has many temporal annotations for facts, a lot of temporal annotations are still missing for time-sensitive facts. We consider the data set *YG36k* consisting of half facts with temporal annotations and the other half missing temporal annotations to evaluate whether partial temporal information of data improves the performance or not. The relationship set is the same in YG15k and YG36k.

We extracted temporal facts mainly from FB15k (Bordes et al., 2013), a subset of Freebase consisting of 1345 relations. Among them, 707 relations are long relations in the form "$r_1.r_2$" concatenating short relations r_1 and r_2. Long relations do not exist in the original schema of Freebase. Many associated facts in Freebase are organized as a CVT structure (similar to an event), e.g., (Einstein,`hasWonPrize`, Nobel) is stored as (Einstein, */award/award_winner/awards_won*, x), (x,*/award/award_honor/award,Nobel*) in Freebase, where x is called *mediator* and not a real entity. FB15k facts are created by concatenating two relations: (*Einstein,/award/award_winner/awards_won. /award/award_honor/award,Nobel*). We extracted temporal annotations from the original Freebase CVT structure for these facts with long relations. For short relations such as */film/director/film*, we used creation/destruction dates of head or tail entity as their time, e.g., the released date of the film. This resulted in 42 time-sensitive relations and 28,610 temporal facts. We denoted the data set as *FB42*. We further added triples without time annotations and created *FB87*. In FB15k, there are about 50% temporal facts in our setting. The data set will be publicly available. All experiments are repeated five times by drawing new training/validation/test splits, and results averaged over the five rounds are reported.

4.2 Time-aware KG Completion

Time-aware KG completion (link prediction) is to complete the triple (e_i, r, e_j, t) when e_i, r or e_j is missing given a specific time interval t. We divided the stage into two sub-tasks, i.e., entity prediction and relation prediction.

4.2.1 Entity Prediction

Evaluation protocol. For each test triple with missing head or tail entity, various methods are used to compute the scores for all candidate entities and rank them in descending order. We use two metrics for our evaluation as in (Bordes et al., 2013): the mean of correct entity ranks (Mean Rank) and the proportion of valid entities ranked in top-10 (Hits@10). As mentioned in (Bordes et al., 2013), the metrics are desirable but flawed when a corrupted triple exists in the KG. As a countermeasure, we may filter out all these corrupted triples which have appeared in KG before ranking. We name the first evaluation set as *Raw* and the second as *Filter*.

For each test quad (triple), we replace the head/tail entity e_i by those entities with *compatible types* as removing triples with incompatible types during test time leads to better results (Chang et al., 2014;

Wang et al., 2015). Entity type information is easy to obtain for YAGO and Freebase. Then we rank the generated corrupted triples in descending order, according to the plausibility (for baselines and TAE model) or the decision variables (for time-aware ILP model). Then we check whether the original correct triple ranks in top-10. To calculate Hit@10 for ILP model, for each test quad, we add additional constraints that at most 10 corrupted are true: $\sum_{i,j} x_{e_i e_j}^{(r_1)} \leq 10$. Mean Rank is missing for ILP method as we could not rank the binary decision variables.

Baseline methods. For comparison, we select TransE (Bordes et al., 2013), its extensions TransH (Wang et al., 2014b) and TransR (Lin et al., 2015b) as our baselines. We then compare time-aware embedding and time-aware ILP inference with each baseline. For example, TransE with TAE and time-aware ILP is denoted as "TransE-TAE" and "TransE-ILP", respectively. The combined model of the two time-aware models are denoted as "TransE-TAE+ILP".

Implementation details. For all embedding methods, we create 100 mini-batches on each data set. The dimension of the embedding n is set in the range of {20,50,100}, the margin γ_1 and γ_2 are set in the range {1,2,4,10}. The learning rate is set in the range {0.1, 0.01, 0.001}. The regularization hyperparameter λ is tuned in $\{10^{-1},10^{-2},10^{-3},10^{-4}\}$. The best configuration is determined according to the mean rank in validation set. For YAGO data set, the optimal configurations are $n = 100$,$\gamma_1 = \gamma_2 = 4$,$\lambda = 10^{-2}$, learning rate is 0.001 and taking ℓ_1-norm; For Freebase data set, the optimal configurations are $n = 100$,$\gamma_1 = \gamma_2 = 1$,$\lambda = 10^{-1}$, learning rate is 0.001 and taking ℓ_1-norm.

We then incorporate temporal constraints into the six models with optimal parameter settings using ILP. To generate the objective function of ILP, plausibility predicted by embedding models is normalized by $w'_{ij} = (w_{ij} - \text{MIN})/(\text{MAX} - \text{MIN})$, where MAX and MIN are max/min scores for each corrupted test triple. We use the lp_solve package[3] to solve the ILP problem.

Results. Table 2 reports the results for each data set. From the results, we can see that 1) TAE methods outperform all the baselines on all the data sets and with all the metrics. The improvements are quite significant. The Mean Rank drops by about 75%, and Hits@10 rises about 19% to 30%. This demonstrates the superiority and generality of our method. 2) Adding more temporal facts improve the performance for TAE models. YG15k consists of 100% temporal facts while YG36k consists of 50% temporal facts. All the temporal information in YG15k is utilized to model temporal associations and make the embeddings more accurate. Therefor, it obtains larger improvement for TAE than YG36k. 3) Improvement for YAGO is larger than Freebase because YAGO data set contains more temporal ordering relation pairs than Freebase data set.

As we can see from Table2, the time-aware ILP method improves each baseline model by about 10% to 16%. This demonstrates the effectiveness of incorporating temporal consistency constraints. Combining two time-aware models further improves the performance by 2% to 3%. This indicates that 1) although TAE models encode temporal order information, only pair-wise temporal ordering relations are optimized during each training iteration. ILP can take advantage of global temporal transitivity which pair-wise methods can't. 2) Adding time span information in the ILP model can remove more false predictions.

4.2.2 Relation Prediction

Relation prediction aims to predict relations between two entities. Evaluation results are shown in Table 3 on YG15K and FB87 due to space limit, and here we report Hits@1 instead of Hits@10. For ILP models, we report Hits@1 for the same reason in entity prediction. Again, two time-aware models improve baselines greatly.

The ILP models improve the precision by about 10%, showing that incorporating temporal constraints directly is better for this task. The main reason is that our temporal constraints are designed to better handle temporal conflicts in relations. Relation prediction and relation extraction from text have common multi-label problems that the same entity pair may have multiple relation labels. For example, *(Obama,US)* could have two valid relations: *wasPresidentOf, wasBornIn*. Through temporal constraints, we are aware that the two relations have different valid time, and therefore we could remove the false one to

[3] http://lpsolve.sourceforge.net/5.5/

Data Sets	YG15K				FB87			
Metric	Mean Rank		Hits@1 (%)		Mean Rank		Hits@1 (%)	
	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter
TransE	1.5	1.4	69.4	73.0	1.9	1.7	60.0	73.8
TransE-TAE	**1.4**	**1.3**	**71.4**	**75.7**	**1.7**	**1.6**	**63.4**	**76.7**
TransE-ILP	-	-	**81.6**	**85.4**	-	-	**71.0**	**82.9**
TransE-TAE-ILP	-	-	**82.5**	**86.5**	-	-	**72.3**	**83.1**
TransH	1.5	1.3	69.7	73.4	1.8	1.6	61.3	75.6
TransH-TAE	**1.3**	**1.3**	**74.6**	**76.9**	**1.4**	**1.30**	**64.2**	**77.2**
TransH-ILP	-	-	**81.1**	**85.7**	-	-	**71.7**	**83.1**
TransH-TAE-ILP	-	-	**83.2**	**86.2**	-	-	**73.2**	**84.4**
TransR	1.4	1.2	71.1	74.3	1.6	1.5	62.1	77.3
TransR-TAE	**1.2**	**1.1**	**74.5**	**78.9**	**1.2**	**1.1**	**64.3**	**79.6**
TransR-ILP	-	-	**82.8**	**86.6**	-	-	**72.2**	**83.2**
TransR-TAE-ILP	-	-	**83.1**	**88.3**	-	-	**73.8**	**85.1**

Table 3: Evaluation results on relation prediction.

Testing quads	TransE	TransE-TAE	TransE-ILP
(Stanford_Moore,?,New_York_City,[1982,1982])	wasBornIn,**diedIn**	**diedIn**,wasBornIn	**diedIn**,wasBornIn
(John_Schoenherr,?,Caldecott_Medal,[1988,1988])	owns,**hasWonPrize**	**hasWonPrize**,created	**hasWonPrize**,created
(John_G._Thompson,?,University_of_Cambridge,[1968,1994])	graduatedFrom,**worksAt**	**worksAt**,graduatedFrom	**worksAt**,graduatedFrom
(Tommy_Douglas,?,New_Democratic_Party,[1961,1972])	isMarriedTo,**isAffiliatedTo**	**isAffiliatedTo**,isMarriedTo	**isAffiliatedTo**,isMarriedTo
(Carmen Electra,?,Owen Wilson,[2004,2005])	isMarriedTo,**sameAward_winner**	isMarriedTo,**sameAward_winner**	**sameAward_winner**,isMarriedTo

Table 4: Examples of relation prediction in descending order. Correct predictions are in **bold**.

improve Hit@1 accuracy.

Qualitative analysis. Examples of relation prediction for TransE, TransE-TAE and TransE-ILP are compared in Table 4. From the results we have the following two conclusions. 1) Temporal order information is useful to distinguish similar relations. For example, when testing *(Stanford_Moore, ? , Chicago, [1982, 1982])*, it's easy for TransE to mix relations `wasBornIn` and `diedIn` as they behave similarly for a person and a place. But knowing that he `graduated` in 1935 from the training set, and TransE-TAE have learnt temporal order that *wasBornIn→graduated→diedIn*, the regularization term $|\mathbf{r}_{graduate}\mathbf{T} - \mathbf{r}_{died}|$ and $|\mathbf{r}_{graduate}\mathbf{T} - \mathbf{r}_{born}|$ helps rank `diedIn` higher than `wasBornIn`. TransE-ILP also benefits from such temporal order constraints and obtains more accurate predictions. 2) Time span information is useful to make accurate predictions. For example, TransE and TransE-TAE both predict (*Carmen Electra,?,Owen Wilson*,[2004,2005]) has `wasMarriedTo` relation. Temporal order constraints don't work for this example. But the time span constraints help TransE-ILP to remove `wasMarriedTo` because *Carmen Electra* was married to *Dave Navarro* during [2003,2008] and a person cannot marry two people at the same time.

5 Related Work

There are two lines of research related to our work.

Knowledge Graph Completion. Nickel et al. (2016) provide a broad overview of machine learning models for KG completion. These models predict new facts in a given knowledge graph using information from existing entities and relations. The most related work from this line of work is KG embedding models (Nickel et al., 2011; Bordes et al., 2013; Socher et al., 2013). Aside from fact triples, external information is employed to improve KG embedding such as combining text (Riedel et al., 2013; Wang et al., 2014a; Zhao et al., 2015), entity type and relationship domain (Guo et al., 2015; Chang et al., 2014), relation path (Lin et al., 2015a; Gu et al., 2015), and logical rules (Wang et al., 2015; Rocktäschel et al., 2015). However, these methods have not utilized temporal information among facts.

Temporal Information Extraction. This line of work mainly falls into two categories: methods that extract temporal facts from web (Ling and Weld, 2010; Wang et al., 2011; Artiles et al., 2011; Garrido et al., 2012) and methods that infer temporal scopes from aggregate statistics in large Web corpora (Talukdar et al., 2012b; Talukdar et al., 2012a). The TempEval task (Pustejovsky and Verhagen, 2009) and systems (Chambers et al., 2007; Bethard and Martin, 2007; Chambers and Jurafsky, 2008; Cassidy et al., 2014) have been successful in extracting temporally related events. Temporal reasoning is also explored to solve temporal conflicts in KG (Dylla et al., 2011; Wang et al., 2010). This paper differs from this line of work as we directly use temporal information from KG to perform KG completion.

6 Conclusion and Future Work

In this paper, we propose two novel time-aware KG completion models. Time-aware embedding (TAE) model imposes temporal order constraints on the geometric structure of the embedding space and enforces it to be temporally consistent and accurate. Time-aware joint inference with ILP framework considers global temporal constraints as well as KG embeddings. It naturally preserves the benefits of embedding models and is more accurate with respect to various temporal constraints. We further integrate two models to make full use of temporal information.

As future work: 1) Many temporal facts are not stored by current KGs (about 30% facts in YAGO and 50% in Freebase lack temporal annotations), we will extract more temporal information from texts. 2) We will consider using our time-aware KG completion model to predict temporal scopes of new facts.

Acknowledgements

This research is supported by National Key Basic Research Program of China (No.2014CB340504) and National Natural Science Foundation of China (No.61375074,61273318). The contact author for this paper are Baobao Chang and Zhifang Sui.

References

Javier Artiles, Qi Li, Taylor Cassidy, Suzanne Tamang, and Heng Ji. 2011. Cuny blender tackbp2011 temporal slot filling system description. In *Proceedings of Text Analysis Conference (TAC)*.

Steven Bethard and James H Martin. 2007. Cu-tmp: Temporal relation classification using syntactic and semantic features. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 129–132. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, number EPFL-CONF-192344.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Taylor Cassidy, Bill McDowell, Nathanael Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. In *ACL*.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. *ACL*, 94305:789–797.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 173–176. Association for Computational Linguistics.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *EMNLP*, pages 1568–1579.

Maximilian Dylla, Mauro Sozio, and Martin Theobald. 2011. Resolving temporal conflicts in inconsistent rdf knowledge bases. In *BTW*, pages 474–493.

MS Fabian, K Gjergji, and W Gerhard. 2007. Yago: A core of semantic knowledge unifying wordnet and wikipedia. In *16th International World Wide Web Conference, WWW*, pages 697–706.

Guillermo Garrido, Anselmo Penas, Bernardo Cabaleiro, and Alvaro Rodrigo. 2012. Temporally anchored relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 107–116. Association for Computational Linguistics.

Kelvin Gu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. *arXiv preprint arXiv:1506.01094*.

Shu Guo, Quan Wang, Bin Wang, Lihong Wang, and Li Guo. 2015. Semantically smooth knowledge graph embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 84–94.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2015a. Modeling relation paths for representation learning of knowledge bases. *arXiv preprint arXiv:1506.00379*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion.

Xiao Ling and Daniel S Weld. 2010. Temporal information extraction. In *AAAI*, volume 10, pages 1385–1390.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

James Pustejovsky and Marc Verhagen. 2009. Semeval-2010 task 13: evaluating events, time expressions, and temporal relations (tempeval-2). In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 112–116. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. 2012a. Acquiring temporal constraints between relations. In *CIKM*.

Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. 2012b. Coupled temporal scoping of relational facts. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 73–82. ACM.

Yafang Wang, Mohamed Yahya, and Martin Theobald. 2010. Time-aware reasoning in uncertain knowledge bases. In *MUD*, pages 51–65. Citeseer.

Yafang Wang, Bin Yang, Lizhen Qu, Marc Spaniol, and Gerhard Weikum. 2011. Harvesting facts from textual web sources by constrained label propagation. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 837–846. ACM.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014a. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).*, pages 1591–1601.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014b. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1859–1865.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*, pages 515–526. ACM.

Yu Zhao, Zhiyuan Liu, and Maosong Sun. 2015. Representation learning for measuring entity relatedness with rich information. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1412–1418. AAAI Press.

Learning to Weight Translations using Ordinal Linear Regression and Query-generated Training Data for Ad-hoc Retrieval with Long Queries

Javid Dadashkarimi **Masoud Jalili Sabet** **Azadeh Shakery**
School of Electrical and Computer Engineering, College of Engineering,
University of Tehran, Tehran, Iran
{dadashkarimi,jalili.masoud,shakery}@ut.ac.ir

Abstract

Ordinal regression which is known with learning to rank has long been used in information retrieval (IR). Learning to rank algorithms, have been tailored in document ranking, information filtering, and building large aligned corpora successfully. In this paper, we propose to use this algorithm for query modeling in cross-language environments. To this end, first we build a query-generated training data using pseudo-relevant documents to the query and all translation candidates. The pseudo-relevant documents are obtained by top-ranked documents in response to a translation of the original query. The class of each candidate in the training data is determined based on presence/absence of the candidate in the pseudo-relevant documents. We learn an ordinal regression model to score the candidates based on their relevance to the context of the query, and after that, we construct a query-dependent translation model using a softmax function. Finally, we re-weight the query based on the obtained model. Experimental results on French, German, Spanish, and Italian CLEF collections demonstrate that the proposed method achieves better results compared to state-of-the-art cross-language information retrieval methods, particularly in long queries with large training data.

1 Introduction

The multilingual environment of the Web has long required the researchers in information retrieval (IR) to introduce powerful algorithms for bridging the gaps between the languages (Nie, 2010; Ganguly et al., 2012; Dadashkarimi et al., 2016). Generally, these algorithms can be categorized as follows: *(1)* translating the query of the user to the language of the documents (Ganguly et al., 2012), *(2)* translating all of the documents into the language of the user (Oard, 1998), *(3)* translating the query and the documents into a third language (Kishida and Kando, 2005), *(4)* bringing the query and the documents into a shared low-dimensional space (Vulic and Moens, 2015; Dadashkarimi et al., 2016), and *(5)* using semantic/concept networks (Franco-Salvador et al., 2014). Usually the query translation approach has been opted as the most efficient and effective approach in the literature (Vulic and Moens, 2015; Nie, 2010). Ma et al. (2012), have shown that cross-language information retrieval (CLIR) takes more advantage of weighting all translations than selecting the most probable ones. But, building this translation model demands a statistical analysis of translation candidates over an aligned corpus or a single target collection (Talvensaari et al., 2007; Liu et al., 2005; Ganguly et al., 2012).

Aligned corpora have been exploited in CLIR successfully (Rahimi et al., 2016; Talvensaari et al., 2007). But, these resources are either scarce in some languages or specific to a few number of domains. Therefore, recently query-dependent collections have been shown to be more effective and are available to many languages (Dadashkarimi et al., 2016; Ganguly et al., 2012). Pseudo-relevant documents are useful resources to this end. In this paper we propose to use pseudo-relevant documents to build a query-dependent translation model. To this aim, first we take top-ranked documents retrieved in response to a simple translation of the query as a pseudo-relevant collection; we expect relevant translations to appear in the collection by accepting a limited amount of noise. Thus we build a training data based

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1725–1733, Osaka, Japan, December 11-17 2016.

on presence/absence of the translations in the collection and a number of embedded features. At the next step we aim to learn an ordinal regression model over the translation candidates and then build a translation model for the query using a softmax function. The final model is used in the second retrieval run.

Since this model requires rather large training data, it is expected to be more useful for long queries, where there is enough information about the user intention. Experimental results on French, Spanish, German, and Italian CLEF collections demonstrate that the proposed method performs better than state-of-the-art dictionary-based CLIR methods particularly in long queries.

In Section 2 we provide an overview on related works and then we propose the method and all the formulations in Section 3. Experimental results and related discussions are provided in Section 4. We conclude the paper and provide future works in Section 5.

2 Previous Works

2.1 Query Translation in CLIR

Query translation is opted as an efficient way for bridging the gap between the source language of the query q^s and the language of a target collection $\mathcal{C} = \{d_1, d_2, .., d_{|\mathcal{C}|}\}$ in CLIR (Nie, 2010). In statistical language modeling, a query translation is defined as building a translation model $p(w_t|q_i^s; q^s)$ where w_t is a translation candidate and q_i^s is a query term. Monz and Dorr (2005) introduced an expectation maximization algorithm for estimating this probability: $p(w_t|q_i^s)^n = p(w_t|q_i^s)^{n-1} + \sum_{w_{t'}} a_{w_t,w_{t'}}.p(w_{t'}|q_i^s)$ where $a_{w_t,w_{t'}}$ is a mutual information of a couple of translations. This probability is computed iteratively and then is used for building query model $p(w_t|q^s)$. Dadashkarimi et al. (2014) and Cao et al. (2008), employed similar methods with bigram probabilities $p(w_t|w_{t'})$. On the other hand, Pirkola et al. (2001) introduced structured queries for CLIR in which each translation of a query term can be considered as a member of a synonym set. Structured queries use a number of operators for building this set. For example $\#sum(\#syn(w_1, .., w_k)\#syn(w'_1, .., w'_{k'}))$ treats occurrences of w_t in a document as occurrences of its set and then sums over all the sets for estimating score of a document. There are also selection-based methods that consider only a limited subset of translations in their retrieval task. Nie (2010), demonstrated that these approaches suffer from lower coverage compared to the weighting approaches.

2.2 Pseudo-relevance Feedback for Query Modeling

Top-ranked documents $F = \{d_1, d_2, .., d_{|F|}\}$ in response to the query of a user have long been considered as informative resources for query modeling (Lavrenko and Croft, 2001; Zhai and Lafferty, 2001; Lv and Zhai, 2014). Relevance models are proposed by (Lavrenko et al., 2002; Lavrenko and Croft, 2001) in both monolingual and cross-lingual environments for language modeling. To this end, Zhai and Lafferty (2001) proposed the mixture model for monolingual environments based on an expectation maximization algorithm. Lv and Zhai (2014) proposed a divergence minimization algorithm that outperforms most of the competitive baselines. There are also a further number of powerful algorithms based on machine learning methods in this area (Liu, 2009). Dadashkarimi et al. (2016), employed a divergence minimization framework for pseudo-relevance feedback using embedded features of words from a positive and a negative sample set of feedback documents. Liu et al. (2005), introduced maximum coherence model for query translation whose aim is to estimate overall coherence of translations based on their mutual information. Dadashkarimi et al. (2016), recently published another work for query translation using low-dimensional vectors of feedback terms from a couple of pseudo-relevant collections. The cross-lingual word embedding translation model (CLWETM) first learns the vectors of feedback terms separately and then aims at finding a query dependant transformation matrix $\mathbf{W}$ for projecting the source vectors to their equivalents in the target language. The projected vectors $\mathbf{W}^T\mathbf{v}_w$ are then used to build a translation model for the query. The authors have shown that CLWETM outperforms the state-of-the-art dictionary-based cross-lingual relevance models.

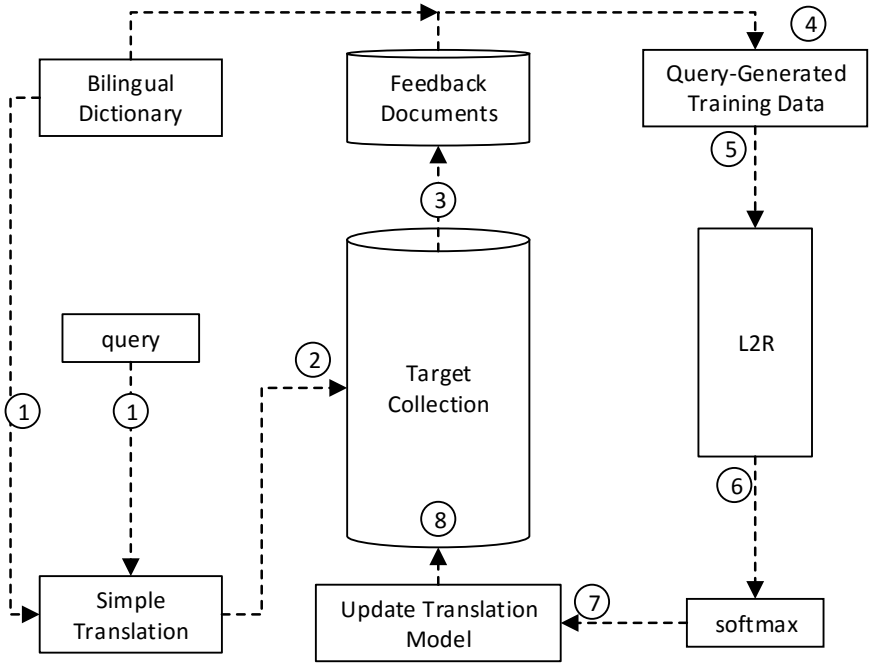


Figure 1: The whole process of building translation model using ordinal linear regression and query-generated training data.

3 Learning to Weight Translations using Query-generated Training Data and Embedded Features

In this section we propose a learning approach for weighting translations of query terms. To this end we first elaborate on building a query-generated training data in Section 3.1. In Section 3.2, we introduce the formulations of the proposed method and finally in Section 3.3 we introduce a number of embedded features used in the learning process.

3.1 Query-generated Training Data for Ordinal Regression

Let $q = \{q_1, .., q_m\}$ be the query and let $q^t = \{w_1, ..w_n\}$ be all the translation candidates of q. We expect correct translations to appear in pseudo-relevant collection F by accepting a limited amount of noise (see Section 2.2). As an example, let the query be $q = \{world, cup, 2018\}$ and assume that $q^t = \{[monde, univers], [coupe, tasse], [2018]\}$ is the set of translation candidates in French. By using a uniform distribution of weights over translation words, $q^t = \{[(1/2, monde), (1/2, univers)], [(1/2, coupe), (1/2, tasse)], [(1, 2018)]\}$ could be a simple query model in the target language. Since $\{monde, coupe, 2018\}$ are conceptually better translations, we expect them to appear in F. Thus, the presence/absence of the translations in F can be indicators of their relevance to the query. We use this information for building a query-generated training data to learn an ordinal regression model for scoring the translations. Let $y_i \in \{-1, +1\}$ indicates the presence/absence of w_i represented by feature vector $\mathbf{x}_i \in \mathbb{R}^n$, and then assume that $D = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^{|\mathbf{x}_i|} \times \{-1, +1\}\}$ is the training data. D is then be used as the training data for our regression model.

3.2 Learning to Rank for Ordinal Translation Regression

We aim to find $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, where $\mathbf{w} \in \mathbb{R}^{|\mathbf{x}|}$ is the weight vector and b is a bias both specific to a query, satisfying the following constraint:

$$f(\mathbf{x}_i) > f(\mathbf{x}_j) \Longleftrightarrow y_i > y_j \quad \forall(\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j) \in D \tag{1}$$

Table 1: Descriptions of the features in **x**.

Feature	Description
$[\mathbf{u}_{w_j}]_k$	the $k-$th dimension of w_j in its low dimensional vector $\mathbf{u}_{w_j} \in \mathbb{R}^{c\times 1}$
$p(w_j\|C)$	the maximum likelihood probability of w_j in the collection
$p(w_j\|\theta_F)$	the maximum likelihood probability of w_j in the feedback documents
$p(w_j\|q^t)$	the maximum likelihood probability of w_j in the simple translation of the query
$\sum_{w_{j'}\notin q_{w_j}} p(w_j, w_{j'})$	sum of the bi-gram probability of w_j with all translations of $q_{w_{j'}} \neq q_{w_j}$

where $f(\mathbf{x})$ should give higher rank to a pseudo relevant translation w_i compared to a non-relevant translation w_j. If we define the set of all translation words' pairs with $P = \{(i,j) : y_i > y_j\}$, finding $f(\mathbf{x})$ requires minimizing the following loss function:

$$\mathcal{L}(\theta) = \frac{1}{2}\mathbf{w}^T\mathbf{w} \qquad s.t. \;\; \forall (i,j) \in P : (\mathbf{w}^T\mathbf{x}_i) \geq (\mathbf{w}^T\mathbf{x}_j) \tag{2}$$

Generally speaking, Equation 2 shows loss-function of an ordinal regression with parameter **w** (Herbrich et al., 1999; Joachims, 2006). Here, the goal is to score $w \in q^t$ based on the embedded feature vectors $\mathbf{x}_{1:n}$ and build a translation model as follows:

$$p(w_j|q) = \frac{1}{m}\frac{\delta_{w_j} e^{\mathbf{w}^T \mathrm{x}_j + b}}{\sum\limits_{w_{j'}} \delta_{w_{j'}} e^{\mathbf{w}^T \mathrm{x}_{j'} + b}} \tag{3}$$

where δ_{w_j} is a weight function specific to each word and m is the number of query terms. We choose $\delta_{w_j} = c(w_j, F)^{\frac{1}{2}}$ equal to the count of w_j in F to the power of $\frac{1}{2}$. This power is for rewarding rare words and penalizing the common ones (Goldberg and Levy, 2014). Figure 1 shows the whole process of building training data and weighting the translations.

3.3 Embedded Features

In Section 3.1 we proposed a query-dependant training data. In this section, we shed light on **x**, the feature vectors in D. As shown in Table 1, we exploited two categories of features: query-dependent features and query-independent features. $p(w_j|C)$ and $[\mathbf{u}_{w_j}]_k$ are independent of the query and capture the frequency of w_j in the collection and the semantic information of w_j in the target language respectively. On the other hand, the other features are specific to the q. $p(w_j|\theta_F)$ captures frequency of w_j in the pseudo-relevant documents. For example in $q = \{world, cup, 2018\}$, although the frequency of $[tasse]$ in collection is more than $[coupe]$, but in F, $[coupe]$ is a more frequent translation compared to $[tasse]$. $p(w_j|q^t)$ is a useful feature for long queries where there are multiple instances of a topical term in the query. According to (Dadashkarimi et al., 2014; Gao et al., 2005), $\sum_{w_{j'}\notin q_{w_j}} p(w_j, w_{j'})$ captures coherence of w_j with the context of the query.

4 Experiments

4.1 Experimental Settings

Details of the used collections are provided in Table 2. As shown in the table we provided experiments on four European languages. For each collection we experiment on both short queries, derived from title of the topics, and long queries, derived from title and description of the topics. We used Lemur toolkit in all experiments[1]. All the queries and documents are stemmed using the Porter stemmer (Porter, 1997). The collections are also normalized and purified from stopwords[2]. We used Dirichlet smoothing method with prior $\mu = 1000$ in a statistical language modeling framework with KL-divergence similarity measure.

[1] http://www.lemurproject.org/

[2] `http://www.unine.ch/info/clef/`

Table 2: Collection Characteristics

ID	Lang.	Collection	Queries (title+description)	#docs	#qrels
IT	Italy	La Stampa 94, AGZ 94	CLEF 2003-2003, Q:91-140	108,577	4,327
SP	Spanish	EFE 1994	CLEF 2002, Q:91-140	215,738	1,039
DE	German	Frankfurter Rundschau 94, SDA 94, Der Spiegel 94-95	CLEF 2002-03, Q:91-140	225,371	1,938
FR	French	Le Monde 94, SDA French 94-95	CLEF 2002-03, Q:251-350	129,806	3,524

Table 3: Comparison of different query translation methods for short queries. Superscripts 1/2/3/4/5/6 indicate that the MAP improvements over the corresponding methods are statistically significant (2-tail t-test, $p \leq 0.05$). * indicates $0.05 \leq p \leq 0.1$ (compared to the proposed method L2R).

		FR (short)			DE (short)			ES (short)			IT (short)		
	ID	MAP	P@5	P@10	MAP	P@5	P@10	MAP	P@5	P@10	MAP	P@5	P@10
1	MONO	0.3262	0.412	0.374	0.2675	0.432	0.369	0.3518	0.496	0.432	0.2949	0.368	0.311
2	TOP-1	0.2211	0.312	0.273	0.2015	0.253	0.233	0.2749	0.367	0.326	0.1566	0.221	0.190
3	UNIF	0.1944	0.269	0.236	0.2148	0.282	0.237	0.236	0.294	0.249	0.1526	0.200	0.156
4	STRUCT	0.1677	0.250	0.226	0.1492	0.227	0.204	0.2472	0.335	0.328	0.0994	0.133	0.118
5	BiCTM	0.2156	0.314	0.275	0.2126	0.282	0.261	0.2652*	0.343	0.316	0.1504	0.217	0.177
6	CLWETM	**0.2312**	**0.331**	0.281	0.2158	0.282	0.255	**0.2915**	**0.384**	**0.337**	0.1630	0.221	**0.194**
7	L2R	0.2296^{2-5}	0.312	**0.288**	$\mathbf{0.2170}^{2-4}$	**0.290**	**0.265**	0.2749^{2-4}	0.380	0.320	$\mathbf{0.1638}^{2-5}$	**0.229**	0.190

The embedding features $[\mathbf{u}_{w_j}]_k$ are computed with word2vec introduced in (Mikolov et al., 2013) on each collection; size of the window, number of negative samples and size of the vectors are set to typical values of 10, 45, and 100 respectively. We also used the svm-rank toolkit for learning $\mathbf{w}$ (Joachims, 2006)[3].

As shown in Table 3 and Table 4 we have the following experimental runs: *(1)* Monolingual retrieval run (MONO). It is the primary comparison baseline for CLIR in the literature (Pirkola et al., 2001; Levow et al., 2005); *(2)* translating by top-ranked translation of a bilingual dictionary (TOP-1) (Ma et al., 2012; Esfahani et al., 2016; Dadashkarimi et al., 2014); *(3)* uniform weighting of translations in the query language modeling (UNIF); *(4)* structured query using $\#syn$ operator as described in Section 2.1 (STRUCT); *(5)* binary coherence translation model (BiCTM) introduced in (Dadashkarimi et al., 2014); cross-lingual word embedding translation model (CLWETM) recently introduced by (Dadashkarimi et al., 2016); and *(6)* the proposed learning to rank (L2R) algorithm. We used the simple STRUCT method for our initial retrieval run to build the query-generated training data as described in Equation 3.1.

4.2 Performance Comparison and Discussion

All the experimental results are provided in Table 3 and Table 4. As shown in Table 3, although L2R outperforms most of the baselines with short queries, the improvements with respect to CLWETM, the most competitive baseline, are marginal. The first reason for these outcomes could be the lower number of training data as shown in Table 6. L2R reaches 70.39%, 81.46%, 78.14%, and 55.54% of performances of the monolingual run in FR, DE, ES, and IT collections respectively.

On the other hand, the proposed L2R outperforms all the baselines with long queries in almost all the metrics. According to Table 4, L2R reaches 77.77%, 70.11%, 77.84%, 61.79% of performance of the monolingual run in FR, DE, ES, and IT collections respectively. Although CLWETM, the state-of-the-art dictionary-based translation model, takes advantage of a couple of collections in the source and target language, L2R successfully outperforms CLWETM with only one collection in the target. Nevertheless, the authors did not exploit comparable corpora for their evaluations and used a pool of multiple news agencies in the source language instead.

[3] https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

Table 4: Comparison of different query translation methods for long queries. Superscripts 1/2/3/4/5/6 indicate that the MAP improvements over the corresponding methods are statistically significant (2-tail t-test, $p \leq 0.05$). $n - m$ indicates all methods in range $[n, .., m]$.

		FR (long)			DE (long)			ES (long)			IT (long)		
	ID	MAP	P@5	P@10	MAP	P@5	P@10	MAP	P@5	P@10	MAP	P@5	P@10
1	MONO	0.4193	0.535	0.473	0.3938	0.528	0.478	0.5281	0.672	0.596	0.3947	0.502	0.436
2	TOP-1	0.3077	0.396	0.343	0.2242	0.308	0.250	0.3762	0.480	0.432	0.2195	0.280	0.262
3	UNIF	0.2709	0.356	0.309	0.2425	0.284	0.254	0.3243	0.368	0.334	0.2095	0.231	0.200
4	STRUCT	0.1800	0.265	0.239	0.2103	0.252	0.250	0.2951	0.400	0.376	0.1942	0.244	0.224
5	BiCTM	0.3050	0.390	0.350	0.2442	0.328	0.278	0.3841	0.464	0.434	0.2172	0.262	0.242
6	CLWETM	0.3167	0.410	0.366	0.2622	0.348	0.308	0.4029	0.500	**0.462**	0.2380	0.298	**0.267**
7	L2R	**0.3261**$^{2-6}$	**0.428**	**0.368**	**0.2761**$^{2-6}$	**0.364**	**0.328**	**0.4111**$^{2-6}$	**0.504**	0.446	**0.2439**$^{2-6}$	**0.302**	0.262

Table 5: Translation model for the English topic 'Brain-Drain Impact' to French.

	UNIF		BiCTM		CLWETM		L2R	
term	candidate	$p(w\|q)$	candidate	$p(w\|q)$	candidate	$p(w\|q)$	candidate	$p(w\|q)$
impact	effet	0.125	effet	0.074646	effet	0.11913	effet	0.143442
impact	impact	0.125	impact	1.35E-03	impact	1.07E-07	impact	0.15437
impact	choc	0.125	choc	1.16E-03	choc	1.07E-07	choc	0.042613
impact	enfonc	0.125	enfonc	4.26E-04	enfonc	1.07E-07	enfonc	0.068032
impact	frapper	0.125	frapper	0.513367	frapper	0.855057	frapper	0.050397
impact	incident	0.125	incident	3.91E-01	incident	1.07E-07	incident	0.377201
impact	porte	0.125	porte	0.017560	porte	0.025813	porte	0.120816
impact	influer	0.125	influer	5.51E-05	influer	1.07E-07	influer	0.04313
brain	tete	0.340	tete	0.999197	tete	0.993176	tete	0.556568
brain	cerveau	0.340	cerveau	0.000758	cerveau	0.003412	cerveau	0.357755
brain	cervelle	0.340	cervelle	4.53E-05	cervelle	0.003412	cervelle	0.085677
drain	pert	0.143	pert	0.192359	pert	0.189706	pert	0.371849
drain	evacu	0.143	evacu	0.227306	evacu	0.216075	evacu	0.318367
drain	epuis	0.143	epuis	0.043371	epuis	0.044900	epuis	0.028666
drain	purg	0.143	purg	0.536827	purg	0.538518	purg	0.112147

Table 5 shows three translation models for the topic 'Brain-Drain Impact' based on UNIF, BiCTM, CLWETM, and L2R. As shown in the table BiCTM and CLWETM are more likely to be trapped in a local optimum. BiCTM originally estimates the query model based on co-occurrences of translations through a collection and thus does not use the pseudo-relevant data. Therefore, it is possible that some translations are co-occurred with each other in the collection but not in a query-dependent collection. On the other hand, CLWETM considers semantic information of the query using low-dimensional vectors of the candidates in top-ranked documents and then combines the obtained translation model with a collection dependent model. CLWETM expects this combination to prevent the final model to be biased to each of the query-dependent/independent collection. This expectation works well in very short queries in which there is a limited information about the intention of the user (e.g., bi-gram queries). But when the original query has an informative knowledge about the intention of the user (i.e., long queries), it is better to consider statistics of the original query as a number of feature alongside the other query-dependent/independent features. For example in Table 5 [*tete*] absorbed all translation weight of 'brain' and then prevented the model to have more coverage/recall. On the other hand, appearing [*cerveau*] as a relevant observation in D, lead L2R to distribute translation probability more justly between [*tete*] and [*cerveau*]. Therefore, we believe that L2R defines a reliable hyperplane discriminating between the context words and the noisy ones more effectively.

4.3 Parameter Sensitivity

$|D|$ is the only parameter in the proposed L2R method. For each collection, we opted $|D|$ that gives the optimum MAP on L2R over a small subset of queries and then tested on remaining topics (Gao et al.,

Table 6: Expected number of query terms ($|q|$) and size of the query-generated training data ($|D|$).

FR				DE				ES				IT			
short		long		short		long		short		long		short		long	
$\|q\|$	$\|D\|$	$\|q\|$	$\|D\|$	$\|q\|$	$\|D\|$	$\|q\|$	$\|D\|$	$\|q\|$	$\|D\|$	$\|q\|$	$\|D\|$	$\|q\|$	$\|D\|$	$\|q\|$	$\|D\|$
2.76	10.62	11.58	53.44	2.8	15.62	11.54	82.7	2.8	11.6	11.56	59.9	2.82	11.14	11.73	60.76

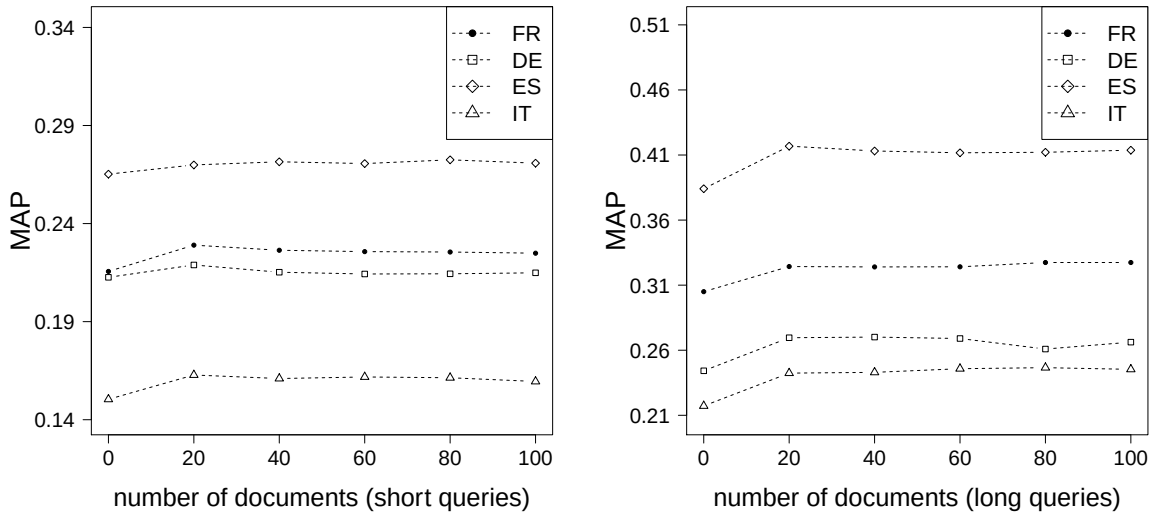


Figure 2: MAP sensitivity of L2R to the number of feedback documents in short and long queries respectively.

2005; Dadashkarimi et al., 2016). As shown in Figure 2, the proposed method works stably in all the collections. In long queries, amount of the improvements are clearly larger than the short ones (see the amounts of jumps from $|D| = 0$ to $|D| = 20$).

5 Conclusion and Future Works

In this paper we proposed a learning to rank method based on ordinal regression on a query-generated training data. We built the query-generated training data of translation words by using their presence/absence in pseudo-relevant documents as labels. This training data consists of embedded features representing each translation word. The result of the regression model was used in the scoring function to weight the translation words.

The method was tested on four different collections in four European languages. The experiments showed that the proposed method outperforms the state-of-the-art dictionary-based CLIR methods, especially in long queries, and it reached up to 81.46% of the performance in the monolingual task. As a future work, the authors would like to test the model on multi-lingual information filtering.

References

Guihong Cao, Stephen Robertson, and Jian-Yun Nie. 2008. Selecting query term alternations for web search by exploiting query contexts. In *ACL-08: HLT*, pages 148–155, Columbus, Ohio, June. Association for Computational Linguistics.

Javid Dadashkarimi, Azadeh Shakery, and Heshaam Faili. 2014. A Probabilistic Translation Method for Dictionary-based Cross-lingual Information Retrieval in Agglutinative Languages. In *CCL '14*, Tehran, Iran.

Javid Dadashkarimi, Mahsa S Shahshahani, Amirhossein Tebbifakhr, Heshaam Faili, and Azadeh Shakery. 2016. Dimension projection among languages based on pseudo-relevant documents for query translation. *arXiv preprint arXiv:1605.07844*.

Hossein Nasr Esfahani, Javid Dadashkarimi, and Azadeh Shakery. 2016. Profile-based translation in multilingual expertise retrieval. In *MultilingMine@ECIR '16*.

Marc Franco-Salvador, Paolo Rosso, and Roberto Navigli. 2014. A knowledge-based representation for cross-language document retrieval and categorization. In *EACL '14*, pages 414–423.

Debasis Ganguly, Johannes Leveling, and Gareth Jones. 2012. Cross-Lingual Topical Relevance Models. In *COLING '12'*, pages 927–942.

Jianfeng Gao, Haoliang Qi, Xinsong Xia, and Jian-Yun Nie. 2005. Linear discriminant model for information retrieval. In *SIGIR '05*, pages 290–297. ACM.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 1999. Support vector learning for ordinal regression. In *ICANN '99*, volume 1, pages 97–102. IET.

Thorsten Joachims. 2006. Training linear svms in linear time. In *SIGKDD '06*, pages 217–226. ACM.

Kazuaki Kishida and Noriko Kando. 2005. A hybrid approach to query and document translation using a pivot language for cross-language information retrieval. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 93–101. Springer.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance-based language models. In *SIGIR '01*, pages 120–127.

Victor Lavrenko, Martin Choquette, and W. Bruce Croft. 2002. Cross-lingual relevance models. In *SIGIR '02*, pages 175–182.

Gina-Anne Levow, Douglas W. Oard, and Philip Resnik. 2005. Dictionary-based Techniques for Cross-language Information Retrieval. *IP&M*, 41(3):523–547.

Yi Liu, Rong Jin, and Joyce Y. Chai. 2005. A maximum coherence model for dictionary-based cross-language information retrieval. In *SIGIR '05*, pages 536–543, Salvador, Brazil.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

Yuanhua Lv and ChengXiang Zhai. 2014. Revisiting the divergence minimization feedback model. In *CIKM '14*, pages 1863–1866.

Yanjun Ma, Jian-Yun Nie, Hua Wu, and Haifeng Wang. 2012. Opening machine translation black box for cross-language information retrieval. In *CIRT '12*, pages 467–476.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS '13'*, pages 3111–3119.

Christof Monz and Bonnie J. Dorr. 2005. Iterative Translation Disambiguation for Cross-language Information Retrieval. In *SIGIR '05*, pages 520–527.

Jian-Yun Nie. 2010. *Cross-Language Information Retrieval*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Douglas W Oard. 1998. A comparative study of query and document translation for cross-language information retrieval. In *Conference of the Association for Machine Translation in the Americas*, pages 472–483. Springer.

Ari Pirkola, Turid Hedlund, Heikki Keskustalo, and Kalervo Järvelin. 2001. Dictionary-based cross-language information retrieval: Problems, methods, and research findings. *Information Retrieval*, 4(3-4):209–230.

M. F. Porter. 1997. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Rzieh Rahimi, Azadeh Shakery, Javid Dadashkarimi, Mozhdeh Ariannezhad, Mostafa Dehghani, and Hossein Nasr Esfahani. 2016. Building a multi-domain comparable corpus using a learning to rank method. *Natural Language Engineering*, 22(04):627–653.

Tuomas Talvensaari, Jorma Laurikkala, Kalervo Järvelin, Martti Juhola, and Heikki Keskustalo. 2007. Creating and exploiting a comparable corpus in cross-language information retrieval. *ACM Transactions on Information Systems (TOIS)*, 25(1):4.

Ivan Vulic and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *SIGIR '15*, pages 363–372.

ChengXiang Zhai and John Lafferty. 2001. Model-based Feedback in the Language Modeling Approach to Information Retrieval. In *CIKM '01*, pages 403–410, Atlanta, Georgia, USA.

Neural Attention for Learning to Rank Questions in Community Question Answering

Salvatore Romeo, Giovanni Da San Martino,
Alberto Barrón-Cedeño and **Alessandro Moschitti**
Qatar Computing Research Institute, HBKU, Doha, Qatar
{sromeo, gmartino, albarron, amoschitti}@qf.org.qa

Yonatan Belinkov, Wei-Ning Hsu,
Yu Zhang, **Mitra Mohtarami** and **James Glass**
MIT Computer Science and Artificial Intelligence Laboratory
Cambridge, MA 02139, USA
{belinkov, wnhsu, yzhang87, mitram, glass}@mit.edu

Abstract

In real-world data, e.g., from Web forums, text is often contaminated with redundant or irrelevant content, which leads to introducing noise in machine learning algorithms. In this paper, we apply Long Short-Term Memory networks with an attention mechanism, which can select important parts of text for the task of similar question retrieval from community Question Answering (cQA) forums. In particular, we use the attention weights for both selecting entire sentences and their subparts, i.e., word/chunk, from shallow syntactic trees. More interestingly, we apply tree kernels to the filtered text representations, thus exploiting the implicit features of the subtree space for learning question reranking. Our results show that the attention-based pruning allows for achieving the top position in the cQA challenge of SemEval 2016, with a relatively large gap from the other participants while greatly decreasing running time.

1 Introduction

Previous work on modeling high-level semantic tasks, e.g., paraphrasing, recognizing textual entailment, and question answering (QA), has shown that syntactic and semantic structures are essential for boosting the accuracy of the applied machine learning algorithm. However, when dealing with real-world data, automatic parsers typically decrease their accuracy. Additionally, there is often a considerable amount of meaningless text for the task, contributing to generating noisy structures. This kind of phenomena can be clearly observed in new Web applications, such as community Question Answering (cQA), where the presence of informal, redundant, and often unrelated text constitutes a major challenge for learning the automatic detection of related topics. For example, given the original question:

Which all places are there for tourists to Qatar? My nephew 18 years on visit.

a cQA system has to infer whether the following question is related:

What are the tourist places in Qatar? I'm likely to travel in the month of june. Just wanna know some good places to visit.

This implies that the system has to (i) recognize the sentences, *My nephew 18 years on visit* and *I'm likely to travel in the month of june*, as irrelevant for the classification task; and (ii) deal with non-standard writing (e.g., *wanna*). If the above steps are carried out with *good* accuracy, complex semantic models can be applied to the selected text to achieve better performance. In particular, such models have to encode structural relations between the constituents of the two questions, e.g., *places are there for tourists* and *some good places to visit*, in the learning algorithm. This is a challenging task as we do not know which relations might be useful, whereas explicitly including all of them is an intractable problem.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1734–1745, Osaka, Japan, December 11-17 2016.

One solution for handling this problem for the answer selection task of standard QA was proposed in (Severyn and Moschitti, 2012). They applied tree kernels (TK) to relational syntactic structures, i.e., a graph representing both question and answer structures. TKs allow for using all the substructures of the relational structures as features in the learning algorithm. However, in Web forums the TK performance is downgraded by the presence of noise and insignificant information, which also makes TKs too slow for processing large datasets. This suggests that text selection (TS) models, applied before TKs in both learning and classification phases have great potential to positively impact the final performance.

In this paper, we study several methods for TS: (i) unsupervised methods based on scalar products with and without TF×IDF weights and (ii) supervised approaches based on attention weights learned by a Long Short-Term Memory (LSTM) network. Additionally, we apply the techniques above with two different strategies reflecting two different granularity levels for: (i) selecting the set of sentences constituting the question trees, and (ii) filtering out tree constituents, where the latter represent pairs of input questions to TKs. We measure the benefit of our TS models against a strong baseline based on TKs and domain specific features for the task of question reranking for cQA, which was recently proposed in SemEval 2016 (Nakov et al., 2016).

Our extensive experiments on the official SemEval dataset produced the following results. First, our basic model, which is based on a combination of (*i*) features using traditional text similarity measures, e.g., bag-of-word models, (*ii*) the initial rank provided by the search engine (providing the initial question rank), and (*iii*) TKs applied to the syntactic structure of the sentences of original and retrieved questions, is state of the art as it ranked 2nd at the official SemEval 2016 challenge (Nakov et al., 2016). Second, TS significantly impacts the performance of TKs by feeding them with better representations. In particular, when using supervised methods, i.e., attention model weights, in sentence selection and tree pruning strategies both provide better representations, and higher results than the models using all sentences or those selected by TF×IDF approaches. Finally, our TS-based system outperforms the top system of the SemEval cQA challenge.

The rest of the paper is organized as follows. Section 2 overviews related work. Section 3 describes our learning-to-rank approach. Section 4 describes the application of LSTMs in TK-based ranking models. Section 5 describes our text selection strategies. Section 6 discusses our experiments and the obtained results. Finally, Section 7 concludes the paper.

2 Related Work

Question ranking in cQA has been central in the research community practically since the begining of cQA system design. Beside "standard" similarity measures, different characterizations and models have been explored. For instance, Cao et al. (2008) proposed a question recommendation system based on the questions' topic and Duan et al. (2008) added the question's focus into the formula. A different approach using topic modeling for question retrieval was introduced by Ji et al. (2012) and Zhang et al. (2014). Here, the authors use LDA topic modeling to learn the latent semantic topics that generate question/answer pairs and use the learned topic distribution to retrieve similar historical questions.

Various methods rely on machine-translation models. For instance, Jeon et al. (2005) and Zhou et al. (2011) used monolingual phrase-based translation models to compare the questions. Jeon et al. (2005) built their translator from a collection of previously identified similar questions whereas Zhou et al. (2011) used question–answer pairs.

Other approaches are based on syntactic representations. This is the case of Wang et al. (2009), who consider the number of common substructures of parse trees to estimate the similarity between two questions. Both Barrón-Cedeño et al. (2016) and Filice et al. (2016) use parse trees as well. The difference is that they use them directly within a tree kernel, with the use of the KeLP platform (Filice et al., 2015a). The latter two models were applied on the SemEval 2016 Task 3 challenge on cQA (Nakov et al., 2016), which proposed a task on question ranking (together with one on answer ranking). The best-performing system in this task was the one from Franco-Salvador et al. (2016), which used SVM^{rank} (Joachims, 2006) on a manifold of features, including distributed representations and semantic resources. To our knowledge, the only work exploring text selection for improving cQA or QA systems is (Barrón-Cedeño

Original Question			
$\mathbf{q}_o$: What are the tourist places in Qatar? I'm likely to travel in the month of June. Just wanna know some good places to visit.			
G	*GS*	*R*	**Retrieved Questions**
1	-1	8	The Qatar banana island will be transfered by the end of 2013 to 5 stars resort called Anantara. Has anyone seen this island? Where is it? Is it near to Corniche?
2	+1	2	Is there a good place here where I can spend some quality time with my friends?
3	-1	7	Where is the best beach in Qatar? Maybe a silent and romantic bay? Where to go for it?
4	-1	9	Any suggestions on what are the happenings in Qatar on Holidays? Something new and exciting suggestions please?
5	-1	3	Where in Qatar is the best place for Snorkeling? I'm planning to go out next friday but don't know where to go.
6	-1	6	Can you give me some nice places to go or fun things to do in Doha for children 17-18 years old? Where can we do some watersports (just for once, not as a member), or some quad driving? Let me know please. Thanks.
7	+1	1	Which all places are there for tourists to Qatar? My nephew 18 years on visit.
8	-1	10	Could you suggest the best holiday destination in the world?
9	-1	5	I really would like to know where the best place to catch fish here in Qatar is. But of course from the beach. I go every week to Umsaeed but rerly i catch somthing! So experianced people your reply will be appreciated.

Table 1: A re-ranking example: we report the Google rank (G), the gold standard relevance (GS) and our rank (R) for each question.

et al., 2016), which exploits tree kernel function itself to auto-filter the non relevant subtrees. The main difference with the approach we present in the current paper is the use of neural networks for learning attention weights and thus modeling sentence or word pruning.

Neural Approaches Recent work has shown the effectiveness of neural models for answer selection (Severyn and Moschitti, 2015; Tan et al., 2015; Feng et al., 2015) and question similarity (dos Santos et al., 2015) in community question answering. For instance, dos Santos et al. (2015) used CNN and bag-of-words (BOW) representations of original and related questions in order to compute cosine similarity scores. Recently, Bahdanau et al. (2014) presented a neural attention model for machine translation and showed that the attention mechanism is helpful for addressing long sentences. We use an LSTM model (Hochreiter and Schmidhuber, 1997) with an attention mechanism for capturing long dependencies in questions for the question similarity task. The major difference with previous work is that we exploit the weights learned by the attention model for selecting important text segments (words, chunks and sentences) for improving syntactic tree-kernel models.

3 Learning to Rank Questions in cQA

This section describes the question reranking problem that we study in this paper and provides state-of-the-art models for its solution. As shown by some methods presented in SemEval, TKs are important for achieving top results.

Class	train	dev	test	overall
Relevant	1,083	214	233	1,530
Irrelevant	1,586	286	467	2,339
Total	2,669	500	700	3,869

Table 2: Class distribution in the training, development, and test partitions.

3.1 Problem Description

The task we focus on is defined as follows: given an original question, q_o, and l candidate questions, q_s, retrieved with a search engine, rerank them with respect to their relevance to q_o.

We use the SemEval 2016 cQA dataset (Nakov et al., 2016), which is composed of 386 user questions, each of which includes 10 potentially related questions. At construction time, the Google search engine, which represents also the strong baseline for the task[1], was used to select potentially relevant forum questions. Table 1 shows an example of the data: an original question on top, followed by several questions retrieved by Google. Nakov et al. crowdsourced the manual annotation of the relevance of the questions. Table 2 shows the amount of relevant and irrelevant instances in the different partitions of the corpus, whereas Table 3 illustrates the distribution of relevant/irrelevant forum questions per ranking position.

[1]The task assumes that the Google Rank is not optimal.

R	train	dev	test	overall	R	train	dev	test	overal
1	0.21 ± 0.05	0.24 ± 0.07	0.40 ± 0.11	0.25 ± 0.07	6	0.08 ± 0.02	0.09 ± 0.02	0.05 ± 0.01	0.08 ± 0.02
2	0.14 ± 0.03	0.18 ± 0.02	0.12 ± 0.02	0.14 ± 0.03	7	0.08 ± 0.02	0.07 ± 0.01	0.05 ± 0.01	0.07 ± 0.02
3	0.11 ± 0.02	0.10 ± 0.01	0.08 ± 0.01	0.10 ± 0.02	8	0.06 ± 0.01	0.04 ± 0.01	0.03 ± 0.00	0.05 ± 0.01
4	0.12 ± 0.03	0.08 ± 0.01	0.10 ± 0.03	0.11 ± 0.03	9	0.07 ± 0.02	0.06 ± 0.01	0.04 ± 0.01	0.07 ± 0.02
5	0.09 ± 0.02	0.09 ± 0.01	0.08 ± 0.02	0.09 ± 0.02	10	0.05 ± 0.01	0.05 ± 0.01	0.04 ± 0.01	0.05 ± 0.01

Table 3: Average fraction (and standard deviation) of Relevant questions at different ranking positions.

Although relevant questions tend to concentrate towards the top of the Google-generated ranking, they are fairly spread on the entire ranking scale.

3.2 The Reranking Approach

Reranking can be modeled by a scoring function $r : \mathcal{Q} \times \mathcal{Q} \rightarrow \mathbb{R}$, where $\mathcal{Q}$ is the set of questions. In turn, r can be modeled as a linear function $r(q_o, q_s) = \vec{w} \cdot \phi(q_o, q_s)$, where $\vec{w}$ is the model and $\phi()$ provides a feature vector representation of the pair.

Binary classifiers, such as SVMs (Joachims, 1999), can be applied for implementing r, where the ranked list is derived from the prediction scores.[2] We model $\phi(q_o, q_s)$ with different advanced feature sets: (*i*) TKs applied to the syntactic structures of question pairs, (*ii*) similarity features computed between q_o and q_s, and (*iii*) a rank feature, i.e., the rank assigned to the question by the search engine.

3.3 Tree Kernels for Question-to-Question Similarity

Kernelized SVMs can express $\vec{w}$ as $\sum_{i=1}^{n} \alpha_i y_i \phi(q_o^i, q_s^i)$, where n are the number of training examples, α_i are weights, y_i are the example labels, $\phi(q_o^i, q_s^i)$ is the representation of pairs of the original and candidate questions. This leads to the following scoring function:

$$r(q_o, q_s) = \sum_{i=1}^{n} \alpha_i y_i \phi(q_o, q_s) \cdot \phi(q_o^i, q_s^i) = \sum_{i=1}^{n} \alpha_i y_i K\big(\langle q_o, q_s \rangle, \langle q_o^i, q_s^i \rangle\big),$$

where the kernel, $K(\cdot, \cdot)$, intends to capture the similarity between pairs of objects constituted by the original and retrieved questions.

The definition of effective Ks for QA and other relational learning tasks, e.g., textual entailment and paraphrasing, has been studied in a large body of work, e.g., (Zanzotto and Moschitti, 2006; Filice et al., 2015b). Given the high similarity between question ranking in cQA and passage ranking in QA, we opted for the state-of-the-art model proposed by Severyn and Moschitti (2012). It should be noted that we apply TK models to pairs of questions rather than questions with their passages.

Figure 1 displays an example of the structure we used for representing the original question, q_o and the seventh candidate question, q_s, in Table 1. The graph is composed by two macro-trees, one for each question, which in turn are constituted by the syntactic trees of the sentences composing the two questions[3]. Additionally, we link the two macro-trees by connecting phrases, e.g., NP, VP, PP, when there is at least lexical match between the phrases of q_o and q_s. Such links are marked with the presence of a REL tag. Finally, we apply the following kernel:

$$K\big(\langle q_o, q_s \rangle, \langle q_o^i, q_s^i \rangle\big) = TK(t(q_o, q_s), t(q_o^i, q_s^i)) + TK(t(q_s, q_o), t(q_s^i, q_o^i)) \ ,$$

where TK is a tree kernel function, e.g., the partial tree kernel by Moschitti (2006) and $t(x, y)$ returns the syntactic tree from the text x, enriching it with the REL tags computed with respect to the syntactic tree of y.

[2] In a set of preliminary experiments, we compared, a true re-ranker, SVM^{rank} (Joachims, 2002), with a standard SVM, the results were comparable.

[3] We have used the OpenNLP tool to build the trees: `https://opennlp.apache.org`.

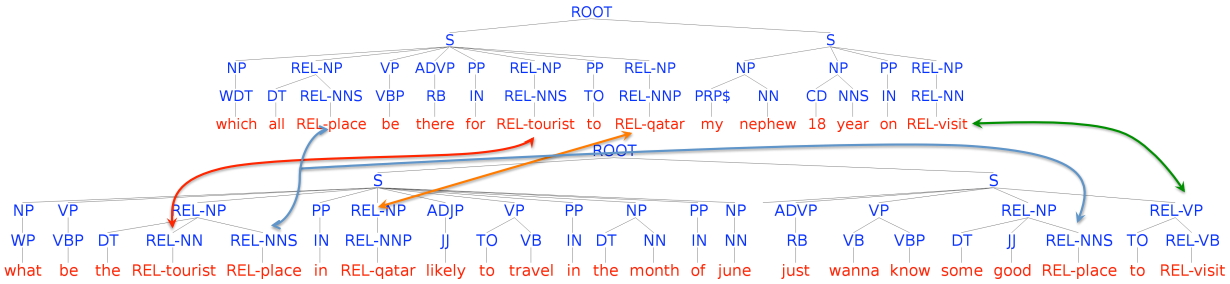


Figure 1: Representation of two questions as syntactic trees. Related nodes are enriched with REL links.

3.4 Feature Vectors

We combine the kernel above with an RBF kernel applied to feature vectors composed of similarity features. These are computed between the original and the related question and the Google rank. Such **text similarity features** (***sim***) are 20 similarities $sim(q_o, q_s)$ using word n-grams ($n = [1, \ldots, 4]$), after stopword removal, using greedy string tiling (Wise, 1996), longest common subsequences (Allison and Dix, 1986), Jaccard coefficient (Jaccard, 1901), word containment (Lyon et al., 2001), and cosine similarity. We also add a structural similarity obtained by comparing the syntactic trees of the questions of an example pair using the partial tree kernel, i.e., $TK(t(q_o, q_s), t(q_s, q_o))$. Note that the operands of the kernel function are members of the same pair. The **ranking-based feature** (***rank***) is computed using the ranking generated by the baseline Google search engine system. Each candidate question is located in one position in the range $[1, \ldots, 10]$. We exploit this information as the inverse of the position.

4 Long Short-Term Memory Networks for TK-based Reranking

As shown in Section 2, several neural approaches have been successfully applied to QA tasks. Unfortunately, question retrieval in cQA is heavily affected by a large amount of noise and a rather different domain, which make it difficult to effectively use out-of-domain embeddings to pre-train neural networks. This probably prevented the participants to SemEval tasks from achieving satisfactory results with such models (Nakov et al., 2016). In this work, we also tried to exploit neural models using their top-level representations for the (q_o, q_s) pair and fed them into the TK classifier as proposed by Tymoshenko et al. (2016), but this simple combination proved to be ineffective as well. In contrast, neural embeddings and weights can be useful for selecting better representations for TK models. In the reminder of this section, we present LSTM networks for question retrieval and our approach for incorporating them into TK-based rerankers.

We approach question ranking as a classification task: given a pair (q_o, q_s), we need to classify q_s as relevant or irrelevant. In order to evaluate the neural classifiers on our ranking task, we can rank candidates, q_s, according to their posterior probability. Among the different models, we were interested in having feedback on the most important pieces of text, thus we opted for LSTM (Hochreiter and Schmidhuber, 1997), which can easily incorporate attention models. LSTMs have proven to be useful in a number of language understanding tasks. Recently, Rocktäschel et al. (2016) adapted an attentional LSTM model (Bahdanau et al., 2014) to textual entailment, and a similar model has been applied to cQA (Hsu et al., 2016). We follow the same setup of the latter: given a pair (q_o, q_s), we learn two serial LSTM models. First, LSTM$_o$ reads the words' vectors of q_o, one by one, and records the corresponding memory cells and hidden states. Second, the final memory cell is used to initialize LSTM$_s$, which reads the words' vectors of q_s. The final hidden state of LSTM$_s$, $\vec{h}_{s,N}$, is used as a feature vector to feed a multi-layer perceptron with one hidden layer, followed by a softmax classifier. The objective function is the cross-entropy objective over binary relevant/irrelevant target labels. We refer to (Hsu et al., 2016) for more details on the architecture and only define the attention model here, as it will be used for TS.

Given the hidden states produced by LSTM$_o$, we compute a weighted representation of q_o:

$$\vec{h}_o = \sum_{i=1}^{L} \beta_i \vec{h}_{o,i} \ ,$$

where $\vec{h}_{o,i}$ are the hidden states corresponding to the words of the original question, and the attention weights are computed as:

$$\beta_i = \frac{exp(a(\vec{h}_{o,i}, \vec{h}_{s,N}))}{\sum_{j=1}^{L} exp(a(\vec{h}_{o,j}, \vec{h}_{s,N}))} \ .$$

Here $a()$ is parameterized as multi-layered perceptron (MLP) with one hidden layer and a *tanh* non-linearity (Rocktäschel et al., 2016). The input to the MLP is then a concatenation of $\vec{h}_o$ and $\vec{h}_{s,N}$. We also concatenate a one-hot vector encoding of the search engine rank (this worked better than scaling the rank into a real-valued feature as $\frac{1}{\text{rank}}$).

Intuitively, β_i assigns a higher weight to words in q_o, if they are useful for determining the relation to q_s. As we will see, these attention weights turn out to be useful for selecting important parts of the questions for the TK models. Note also that the attention here is one-sided, only on q_o. In practice, we train another model, with attention on q_s, and use its weights.

5 Text Selection

Our goal is to select important sentences and/or their constituents to improve the representation of the TK-based system proposed in Section 3. We consider two strategies: selecting a subset of sentences among those composing each question or pruning tree nodes in a bottom-up fashion. For each strategy, we consider both supervised methods by LSTM attention weights and unsupervised methods for TS.

5.1 Sentence selection

Given an original question q_o, with its sentences $\{s^o_1, ..., s^o_n\}$, and a related question q_s, with its sentences $\{s^s_1, ...s^s_m\}$, we aim at selecting those sentences that are most relevant for determining the similarity between q_o and q_s. For this purpose, we create two ranked sentence lists, Σ_o and Σ_s, such that the top-k (from each set) will constitute the sets to be parsed and used to build the macro-trees described in Section 3.3. We experiment with different values for the k parameter. Our sorting algorithm is rather simple: we (i) find the most similar pair of sentences, $(\sigma_o, \sigma_s) \in q_o \times q_s$, using a scoring function, $sim(\sigma_o, \sigma_s)$; (ii) store them individually in Σ_o and Σ_s, respectively, by also preserving the insertion order; and (iii) remove such sentences from q_o and q_s. We continue executing these steps until $q_o \times q_s$ is empty. The scoring (similarity) function can be modeled with both supervised and unsupervised methods.

Unsupervised methods We generate the vector representation of each sentence in q_o and q_s by averaging 300-dimensional word2vec (Mikolov et al., 2013a; Mikolov et al., 2013b) vector representations after stopword removal (we use the Google News default model). Then, we compute sim as the standard cosine similarity. We call this model **sim(nw)**. As a second approach, we apply the same algorithm above by weighing the word2vec vectors with TF×IDF (**sim(tf-idf)**), where IDF is derived from the entire dataset. Note that averaging has the consequence of ignoring the word order in the sentence. We leave the exploration of more sophisticated sentence representation models (e.g., Skip-Thought (Kiros et al., 2015)) for future work.

Supervised Methods We conjectured that using the information encoded by question labels to guide TS may improve our ability to measure question similarity. For this purpose, we exploit the attention weights learned by our LSTM model (Section 4). In more detail, the sentence scores for σ_o and σ_s are computed with the average attention weights β_i over sentence words. Since the attention model is one-sided, it generates weights for either the original or the related question. We thus train two separate models, for q_o and q_s, and rank each sentence list independently, on the basis of the attention weights[4].

[4]In practice, we found it useful to consider only the top-m weights in the average weights computation. In our experiments, we set $m = 3$, which worked well in preliminary experiments.

Model	**MAP**	AvgRec	MRR
Google baseline	71.35	86.11	76.67
Kernel-based			
sim	64.80	82.52	73.73
sim + rank	69.82	85.91	77.17
sim + rank + TK	73.58	89.10	79.83
LSTM	68.06	84.22	74.33

(a) Performance on the development set.

Model	**MAP**	AvgRec	MRR
Google baseline	74.75	88.30	83.79
Kernel-based			
sim	70.70	85.78	80.58
sim + rank	74.58	89.09	83.57
sim + rank + TK	76.15	90.79	84.76
LSTM	67.96	85.03	76.63

(b) Performance on the test set

Table 4: Different feature combination methods and learning models on the development and test sets.

We call this model ***attention***. Additionally, similarly to the unsupervised approaches, we compute a cosine similarity score for sentence pairs, where each sentence is represented as a bag-of-lemmas vector whose entries are the corresponding attention weights. Note that the attention model may assign different weights to identical words that appear more than once in the sentence. In this case, we use the average attention weight over identical lemmas. We name this model ***sim(att)***.

5.2 Tree pruning

Our second approach to TS works on the syntactic tree nodes and it is illustrated by Algorithm 1. Its main idea is to filter out the leaf nodes of the parse tree corresponding to words associated with weights lower than a user-defined threshold, where the word weights are provided by either β_i or TF×IDF. The most important step of Algorithm 1 is the recursive function *pruneNode*, which is initially invoked for the root node of the tree. Function *pruneNode* checks whether the node n is a leaf (Line 4) and then applies the appropriate strategy: (i) for non-leaf nodes, *pruneNode* is invoked for the children of n, then n is removed if all of its children are removed; and (ii) a leaf node is removed if its weight is lower than the user-defined threshold, h. Additionally, since the REL tagging has proved to be effective for paraphrasing and textual entailment tasks (Filice et al., 2015b), we experiment with the simple rule no REL-tagged node (see Section 3.3) is removed, independently of its weight and number of children.

```
1  Function PruneTree (T, h);
   Input  : a tree T;
            a pruning threshold h;
   Output: a pruned version of T
2  pruneNode(root(T), h);
3  Function pruneNode (n, h);
4  if |children(n)| > 0 then
5      for c ∈ children(n) do
6          pruneNode(c, h);
7      end
8      if |children(n)| = 0 && !REL_Node(n)) then
9          remove (n, T);
10     end
11 else
12     if n.weight < h && !REL_Node(n)) then
13         remove (n, T);
14     end
15 end
```

Algorithm 1: Function *PruneTree* for pruning a tree according to tf-idf or attention weights.

Finally, it should be noted that different thresholds determine different percentages of pruned nodes.

6 Experiments

In these experiments, we first evaluate our state-of-the-art reranker to establish a strong baseline. Then, we measure the impact of our different TS models with respect to accuracy and speed.

6.1 Testing the Baseline Models

We use binary SVMs to generate our rankings, where TKs are enabled by the KeLP toolkit[5]. KeLP allows for combining our three types of features within different kernels, namely RBF for the similarity features, TKs for the parse trees, and RBF kernels for the ranking-based feature[6]. We set the C parameter of SVMs to 1 in all experiments whereas we used the default values for TK and RBF kernel parameters. MAP is computed over all questions and then averaged.

[5] https://github.com/SAG-KeLP

[6] RBF proved superior than linear kernels for both similarity and ranking features in our internal experiments.

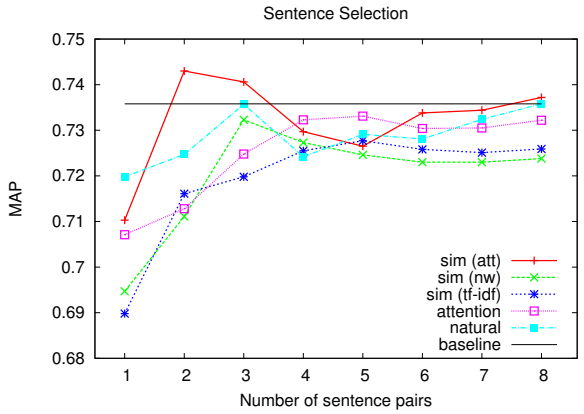


Figure 2: MAP evaluated on the dev. set according to the number of selected sentences. The different text selection methods are the natural order, the average attention weight, and cosine similarity computed with three weighting models: attention (att), no weight (nw), and tf-idf. The *baseline* uses all the sentences.

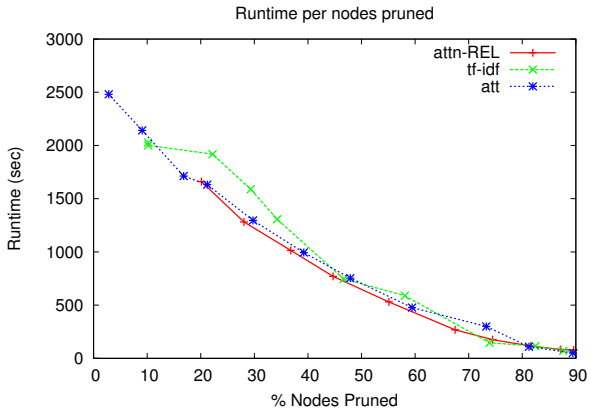


Figure 3: Reranker training time with respect to the percentage of pruned nodes using: attention weights (attn), attention weights without pruning REL nodes (attn-REL), and TF×IDF weights without pruning REL nodes (tf-idf).

Tree Kernels and Feature Combinations We first experimented with the features defined in Section 3.4: similarities (*sim*) and the *rank* feature. Additionally, we combine these features with the TKs described in Section 3.3. Tables 4a and 4b report the obtained performance on the development and test datasets. We note that (i) the Google rank baseline is rather strong; (ii) the MAP of *sim* alone is below the baseline; (iii) combining *rank* and *sim* produces an increase but the result is still below the baseline; and (iv) only the full combination, *sim+rank+TK*, improves on Google. Table 5b reports the three top systems of the SemEval cQA competition, where ConvKN (Da San Martino et al., 2016; Barrón-Cedeño et al., 2016) is the model we described in Sec. 3 whereas KeLP (Filice et al., 2016) shares a number of features with ConvKN along with the TK-based approach.

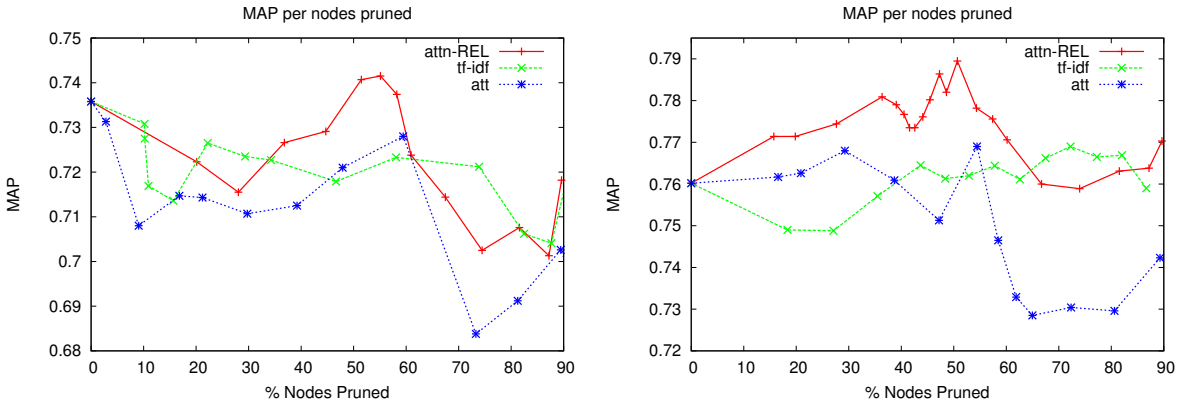
Neural Networks Tables 4a and 4b also show the results obtained with the LSTM, which does not improve upon the strong Google baseline, even though its rank is incorporated as an additional feature. This is in line with (Hsu et al., 2016), where a combination method was necessary to obtain a better performance. The low MAP can be attributed to the small dataset size (only 2,669 training examples). Nevertheless, the attention weights learned by the neural model are still useful for TS (see next section).

6.2 Measuring the Impact of the Text Selection Methods

In these experiments, we focus on reducing the text used for generating the trees (the other features are computed on the standard text). We use sentence selection and tree pruning, as described in Section 5.

Sentence selection results We test different algorithms for sorting sentences for both original and related questions, i.e., to obtain the sorted lists, Σ_o and Σ_s (see Sec. 5.1), and then use the top k sentences to build the TK representations. Figure 2 shows the results on the dev. set of our reranker using different sorting algorithms: *natural* is the natural order (it can be considered as a competitive baseline), *attention* uses the average attention weight to sort sentences, and *sim(att)*, *sim(nw)* and *sim(tf-idf)* apply cosine similarity, where the sentence vector weights are constituted by attention weights, no weight (nw), and TF×IDF weights, respectively. The system *baseline* uses all the sentences, i.e., it is the best model tested in the previous section.

We note that (i) all the models using only one sentence perform lower than the *baseline*, i.e., using all sentences. (ii) The first sentence is intuitively very important as *natural* with one sentence is the best model. (iii) As soon as the number of sentences increases, the supervised models, i.e., *sim(att)* and *attention*, perform better than the unsupervised approaches, i.e., *sim(nw)* and *sim(tf-idf)*. In particular,



(a) On the development set.

(b) On the test set.

Figure 4: MAP scores after pruning the input texts using: attention weights (attn), attention weights without pruning REL-tagged nodes (attn-REL), and TF×IDF weights without pruning REL-tagged nodes (tf-idf).

sim(att) outperforms the *baseline*, e.g., 74.30 vs. 73.58 (although only when using 2 or 3 sentences). In any case, almost all the results are interesting since, when the number of sentences decreases, the computational complexity of TK applied to the reduced trees becomes much lower. We measured a decrease of 5 times of the training time required by the reranker. However, sentence selection may be considered too coarse-grained to be effective.

Tree pruning results Given word-level weights, either by TF×IDF or the attention model, we prune the parse trees according to the strategies described in Section 5.2: pruning using the weights of the (i) attention model (*attn*); (ii) attention model but retaining all REL-tagged nodes (*attn-REL*); (iii) TF×IDF model preserving all REL nodes (*tf-idf*). The experiments with TF×IDF without REL-tagging are not reported due to poor performance.

Figure 4 compares several pruning thresholds in terms of MAP on both the development and the test sets. The x-axis values indicate the percentage of nodes that were removed after pruning the question parse trees. Pruning results rather beneficial: after an initial performance decrease, MAP improves the *baseline* model (state of the art). For example, according to the results on the dev. set, *attn-REL*, which uses attention weights and also preserves the REL nodes, allows us to reduce the size of the trees by 55%, obtaining a MAP value of 74.15 (+0.57 with respect to using the full trees). This improvement is also statistically significant at 95% with respect to the $baseline$. The increase in performance is much more evident on the test set, where the best MAP of *attn-REL* is 78.95, obtained by pruning 50% of the nodes. Note that the best pruning threshold on the development set, i.e., corresponding to 55% of filtering, when used for the test set, achieves a MAP of 77.82, which, even if lower than the top result, 78.95, still outperforms the 76.70 MAP of the winner system of Semeval 2016 (Nakov et al., 2015); compare tables 5a and 5b. The other two models, based on TF×IDF and attention weights without preserving REL nodes, do not improve MAP, although they can produce a great speedup with a small loss in MAP. Figure 3 reports the running times of the three pruning models above with respect to the percentage of pruned nodes. All pruning strategies can reduce the size of the trees significantly, and consequently reduce the running time, for all weight sources. For example, the most accurate model on the dev. set (see Figure 4) filters 55% of the nodes, speeding up training by 5 times, i.e., 530 versus 2,580 seconds.

7 Conclusions

In this paper, we showed that TK-based models achieve the state of the art in cQA. Nevertheless, such models are affected by noise in the syntactic structure and redundant information, typically added by Web users when formulating or answering forum questions. We proposed to alleviate this problem selecting more significant text segments from the question text. For this purpose, we used unsupervised meth-

Model	**MAP**	AvgRec	MRR
sim+rank+TK (*baseline*)	76.15	90.79	84.76
Tree Pruning (attn+REL)	**77.82**	91.31	84.64
sim(tf-idf)	76.28	90.05	83.38
sim(att)	75.85	89.95	81.31

(a) Results of selected models on the test set using the best pruning threshold of the development set.

Model	**MAP**	AvgRec	MRR
UH-PRHLT	76.70	90.31	83.02
ConvKN	76.02	90.70	84.64
KeLP	75.83	91.02	82.71

(b) SemEval top 3 systems on the test set (Nakov et al., 2016).

Table 5: Comparison between the best systems of this paper and the SemEval Challenge.

ods and supervised methods based on LSTM with attention weights. The results show that supervised text selection can improve unsupervised models, thus enhancing the performance of a tree-kernel-based reranker. Additionally, using less text produces a boost in the speed of tree kernels —up to five times the speed of the model using all the sentences— while also improving MAP. Finally, our model achieves a top result with respect to the top performing systems submitted to the SemEval 2016 task on cQA. In the future, we would like to experiment with more advanced techniques that have been shown successful for traditional text summarization, e.g., using discourse structure.

Acknowledgements

This research was performed by the Arabic Language Technologies (ALT) group at the Qatar Computing Research Institute (QCRI), HBKU, part of Qatar Foundation, within the Interactive sYstems for Answer Search (Iyas) project, which is developed in collaboration with MIT-CSAIL.

References

Lloyd Allison and Trevor Dix. 1986. A Bit-string Longest-common-subsequence Algorithm. *Inf. Process. Lett.*, 23(6):305–310, December.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Shafiq Joty, Alessandro Moschitti, Fahad Al-Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. ConvKN at SemEval-2016 Task 3: Answer and Question Selection for Question Answering on Arabic and English Fora. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 896–903, San Diego, California, June. Association for Computational Linguistics.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Salvatore Romeo, and Alessandro Moschitti. 2016. Selecting sentences versus selecting tree constituents for automatic question ranking. In *Proceedings of the 26th International Conference on Computational Linguistics*, Osaka, Japan.

Yunbo Cao, Huizhong Duan, Chin-Yew Lin, Yong Yu, and Hsiao-Wuen Hon. 2008. Recommending Questions Using the Mdl-based Tree Cut Model. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 81–90, New York, NY, USA. ACM.

Giovanni Da San Martino, Alberto Barrón-Cedeño, Salvatore Romeo, and Alessandro Moschitti. 2016. Learning to re-rank questions in community question answering using advanced features. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, CIKM '16, Indianapolis, IN, October. Association for Computational Linguistics.

Cicero dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning Hybrid Representations to Retrieve Semantically Equivalent Questions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 694–699, Beijing, China, July. Association for Computational Linguistics.

Huizhong Duan, Yunbo Cao, Chin-Yew Lin, and Yong Yu. 2008. Searching Questions by Identifying Question Topic and Question Focus. In *ACL*.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying Deep Learning to Answer Selection: A Study and An Open Task. *CoRR*, abs/1508.01585.

Simone Filice, Giuseppe Castellucci, Danilo Croce, Giovanni Da San Martino, Alessandro Moschitti, and Roberto Basili. 2015a. KeLP: a Kernel-based Learning Platform in Java. In *The workshop on Machine Learning Open Source Software (MLOSS): Open Ecosystems*, Lille, France, July. International Conference of Machine Learning.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2015b. Structural Representations for Learning Relations between Pairs of Texts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1003–1013, Beijing, China, July. Association for Computational Linguistics.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KeLP at SemEval-2016 Task 3: Learning Semantic Relations between Questions and Answers. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1116–1123, San Diego, California, June. Association for Computational Linguistics.

Marc Franco-Salvador, Sudipta Kar, Thamar Solorio, and Paolo Rosso. 2016. UH-PRHLT at SemEval-2016 Task 3: Combining Lexical and Semantic-based Features for Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 814–821, San Diego, California, June. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

Wei-Ning Hsu, Yu Zhang, and James R. Glass. 2016. Recurrent Neural Network Encoder with Attention for Community Question Answering. *CoRR*, abs/1603.07044.

Paul Jaccard. 1901. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding Similar Questions in Large Question and Answer Archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM.

Zongcheng Ji, Fei Xu, Bin Wang, and Ben He. 2012. Question-answer topic model for question retrieval in community question answering. In *CIKM*.

Thorsten Joachims. 1999. Making Large-scale Support Vector Machine Learning Practical. In *Advances in Kernel Methods*. MIT Press, Cambridge, MA, USA.

Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. KDD.

Thorsten Joachims. 2006. Training Linear SVMs in Linear Time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 217–226, New York, NY, USA. ACM.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Caroline Lyon, James Malcolm, and Bob Dickerson. 2001. Detecting short passages of similar text in large document collections. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, EMNLP '01, pages 118–125, Pittsburgh, PA, USA.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Alessandro Moschitti. 2006. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Computer Science*, pages 318–329. Springer Berlin Heidelberg.

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2015. SemEval-2015 Task 3: Answer Selection in Community Question Answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 269–281, Denver, Colorado, June. Association for Computational Linguistics.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. SemEval-2016 Task 3: Community Question Answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, June. Association for Computational Linguistics.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about Entailment with Neural Attention. In *International Conference on Learning Representations*.

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer reranking. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 741–750, Portland, Oregon, USA.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 373–382, New York, NY, USA. ACM.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. LSTM-based Deep Learning Models for non-factoid answer selection. *CoRR*, abs/1511.04108.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016. Convolutional Neural Networks vs. Convolution Kernels: Feature Engineering for Answer Sentence Reranking. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1268–1278, San Diego, California, June. Association for Computational Linguistics.

Kai Wang, Zhaoyan Ming, and Tat-Seng Chua. 2009. A syntactic tree matching approach to finding similar questions in community-based qa services. In *SIGIR*.

Michael Wise. 1996. YAP3: Improved Detection of Similarities in Computer Program and Other Texts. In *Proceedings of the Twenty-seventh SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '96, pages 130–134, New York, NY, USA.

Fabio Massimo Zanzotto and Alessandro Moschitti. 2006. Automatic learning of textual entailments with cross-pair similarities. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 44, page 401.

Kai Zhang, Wei Wu, Haocheng Wu, Zhoujun Li, and Ming Zhou. 2014. Question retrieval with high quality answers in community question answering. In *CIKM*.

Guangyou Zhou, Li Cai, Jun Zhao, and Kang Liu. 2011. Phrase-based translation model for question retrieval in community question answer archives. In *ACL*.

Simple Question Answering by Attentive Convolutional Neural Network*

Wenpeng Yin*, **Mo Yu**†, **Bing Xiang**†, **Bowen Zhou**†, **Hinrich Schütze***
*Center for Information and Language Processing, LMU Munich, Germany
wenpeng@cis.lmu.de
†IBM Watson, Yorktown Heights, NY, USA
{yum,bingxia,zhou}@us.ibm.com

Abstract

This work focuses on answering single-relation factoid questions over Freebase. Each question can acquire the answer from a single fact of form (subject, predicate, object) in Freebase. This task, simple question answering (SimpleQA), can be addressed via a two-step pipeline: entity linking and fact selection. In fact selection, we match the *subject entity in a fact candidate* with the entity mention in the question by a *character-level* convolutional neural network (char-CNN), and match the *predicate in that fact* with the question by a *word-level* CNN (word-CNN). This work makes two main contributions. (i) A simple and effective entity linker over Freebase is proposed. Our entity linker outperforms the state-of-the-art entity linker over SimpleQA task. [1] (ii) A novel attentive maxpooling is stacked over word-CNN, so that the predicate representation can be matched with the predicate-focused question representation more effectively. Experiments show that our system sets new state-of-the-art in this task.

1 Introduction

Factoid question answering (QA) over knowledge bases such as Freebase (Bollacker et al., 2008) has been intensively studied recently (e.g., Bordes et al. (2014), Yao et al. (2014), Bast and Haussmann (2015), Yih et al. (2015), Xu et al. (2016)). Answering a question can require reference to multiple related facts in Freebase or reference to a single fact. This work studies simple question answering (SimpleQA) based on the *SimpleQuestions* benchmark (Bordes et al., 2015) in which answering a question does not require reasoning over multiple facts. Single-relation factual questions are the most common type of question observed in various community QA sites (Fader et al., 2013) and in search query logs. Even though this task is called "simple", it is in reality not simple at all and far from solved.

In SimpleQA, a question, such as "what's the hometown of Obama?", asks a single and direct topic of an entity. In this example, the entity is "Obama" and the topic is hometown. So our task is reduced to finding one fact (subject, predicate, object) in Freebase that answers the question, which roughly means the *subject* and *predicate* are the best matches for the topical entity "Obama" and for the topic description "what's the hometown of", respectively. Thus, we aim to design a method that picks a fact from Freebase, so that this fact matches the question best. This procedure resembles answer selection (Yu et al., 2014) in which a system, given a question, is asked to choose the best answer from a list of candidates. In this work, we formulate the SimpleQA task as a fact selection problem and the key issue lies in the system design for how to match a fact candidate to the question.

The first obstacle is that Freebase has an overwhelming number of facts. A common and effective way is to first conduct entity linking of a question over Freebase, so that only a small subset of facts remain as candidates. Prior work achieves entity linking by searching word n-grams of a question among all entity names (Bordes et al., 2015; Golub and He, 2016). Then, facts whose subject entities match those n-grams are kept. Our first contribution in this work is to present a simple while effective entity linker

*This work was conducted during the first author's internship at IBM Watson Group.

[1] *We release our entity linking results at*: https://github.com/Gorov/SimpleQuestions-EntityLinking

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1746–1756, Osaka, Japan, December 11-17 2016.

to this task. Our entity linker first uses each word of a question (or of an entity mention in the question) to search in the entity vocabulary, all entities are kept if their names *contain one of the query words*. Then, we design three simple factors to give a raw ranking score for each entity candidate: (i) the ratio of words in the entity name that are covered by the question; (ii) the ratio of words in the question that are covered by the entity name; (iii) the position of the entity mention in the question. We choose top-N ranked entities as candidates. Our entity linker does not consider the semantics or topic of an entity; it considers only the string surface. Nevertheless, experiments show that these three factors are the basis for a top-performing entity linker for SimpleQA.

Based on entity linking results, we consider each fact as a fact candidate that has one of the entity candidates as subject. Then our system solves the task of *fact selection*, i.e., matching the question with each fact candidate and picking the best one. Our system is built based on two observations. (i) Surface-form match between a subject entity and its mention in the question provides more straight-forward and effective clue than their semantic match. For example, "Barack Obama" matches with "Obama" in surface-form, which acts as a fundamental indicator that the corresponding fact and the question are possibly about the same "Obama". (ii) Predicate in a fact is a paraphrase of the question's pattern where we define the *pattern* to be the topic asked by the question about the entity, and represent it as the question in which the entity mention has been replaced by a special symbol. Often the predicate corresponds to a keyword or a rephrased token of the pattern, this means we need to create a flexible model to handle this relationship.

These observations motivate us to include two kinds of convolutional neural networks (CNN, LeCun et al. (1998)) in our deep learning system. (i) A character-level CNN (*char-CNN*) that models the match between an Freebase entity and its mention in the question on surface-form. We consider CNN over character-level rather than the commonly-used word-level, so that the generated representation is more robust even in the presence of typos, spaces and other character violations. (ii) A word-level CNN (*word-CNN*) with attentive maxpooling that learns the match of the Freebase predicate with the question's pattern. A Freebase predicate is a predefined relation, mostly consisting of a few words: "place of birth", "nationality", "author editor" etc. In contrast, a pattern is highly variable in length and word choice, i.e., the subsequence of the question that represents the predicate in a question can take many different forms. Convolution-maxpooling slides a window over the input and identifies the best matching subsequence for a task, using a number of filters that support flexible matching. Thus, convolution-maxpooling is an appropriate method for finding the pattern subsequence that best matches the predicate description. *We add attention to this basic operation of convolution-maxpooling.* Attentions are guided by the predicate over all n-gram phrases in the pattern, finally system pools phrase features by considering the feature values as well as the attentions towards those features. *Phrases more similar to the predicate, i.e., with higher attention values, will be selected with higher probability than other phrases to represent the pattern.*[2]

Our overall approach is for the entity linker to identify top-N entity candidates for a question. All facts that contain one of these entities as subject are then the fact search space for this question. Char-CNN and word-CNN decompose each question-fact match into an entity-mention surface-form match and a predicate-pattern semantic match. Our approach has a simple architecture, but it outperforms the state-of-the-art, a system that has a much more complicated structure.

2 Related Work

As mentioned in Section 1, factoid QA against Freebase can be categorized into single-relation QA and multi-relation QA. Much work has been done on multi-relation QA in the past decade, especially after the release of benchmark WebQuestions (Berant et al., 2013). Most state-of-the-art approaches (Berant et al., 2013; Yahya et al., 2013; Yao and Van Durme, 2014; Yih et al., 2015) are based on semantic parsing, where a question is mapped to its formal meaning representation (e.g., logical form) and then translated to a knowledge base (KB) query. The answers to the question can then be retrieved simply

[2]Surface-form entity linking has limitations in candidate collection as some entities have the same names. We tried another word-CNN to match the pattern to the entity description provided by Freebase, but no improvement is observed.

by executing the query. Other approaches retrieve a set of candidate answers from KB using relation extraction (Yao and Van Durme, 2014; Yih et al., 2014; Yao, 2015; Bast and Haussmann, 2015) or distributed representations (Bordes et al., 2014; Dong et al., 2015; Xu et al., 2016). Our method in this work explores CNN to learn distributed representations for Freebase facts and questions.

SimpleQA was first investigated in (Fader et al., 2013) through PARALEX dataset against knowledge base Reverb (Fader et al., 2011). Yih et al. (2014) also investigate PARALEX by a system with some similarity to ours – they employ CNNs to match entity-mention and predicate-pattern. Our model differs in two-fold. (i) They use the same CNN architecture based on a word-hashing technique (Huang et al., 2013) for both entity-mention and predicate-pattern matches. Each word is first preprocessed into a count vector of character-*trigram* vocabulary, then forwarded into the CNN as input. We treat entities and mentions as character sequences. Our char-CNN for entity-mention match is more end-to-end without data preprocessing. (ii) We introduce attentive maxpooling for better predicate-pattern match.

The latest benchmark SimpleQuestions in SimpleQA is introduced by Bordes et al. (2015). Bordes et al. (2015) tackle this problem by an embedding-based QA system developed under the framework of Memory Networks (Weston et al., 2015; Sukhbaatar et al., 2015). The setting of the SimpleQA corresponds to the elementary operation of performing a single lookup in the memory. They investigate the performance of training on the combination of SimpleQuestions, WebQuestions and Reverb training sets. Golub and He (2016) propose a character-level attention-based encoder-decoder framework to encode the question and subsequently decode into (subject, predicate) tuple. Our model in this work is much simpler than these prior systems. Dai et al. (2016) combine a unified conditional probabilistic framework with deep recurrent neural networks and neural embeddings to get state-of-the-art performance.

Treating SimpleQA as fact selection is inspired by work on answer selection (e.g., Yu et al. (2014), Yin et al. (2016b), Santos et al. (2016)) that looks for the correct answer(s) from some candidates for a given question. The answer candidates in those tasks are raw text, not structured information as facts in Freebase are. We are also inspired by work that generates natural language questions given knowledge graph facts (Seyler et al., 2015; Serban et al., 2016). It hints that there exists a kind of match between natural language questions and FB facts.

3 Task Definition and Data Introduction

We first describe the Freebase (Bollacker et al., 2008) and SimpleQuestions task (Berant et al., 2013).

Freebase is a structured knowledge base in which entities are connected by predefined predicates or "relations". All predicates are directional, connecting from the subject to the object. A triple (subject, predicate, object), denoted as (h, p, t), describes a fact; e.g., (U.S. Route 2, major_cities, Kalispell) refers to the fact that U.S. Route 2 runs through the city of Kalispell.

SimpleQuestions benchmark, a typical SimpleQA task, provides a set of single-relation questions; each question is accompanied by a ground truth fact. The object entity in the fact is the answer by default. The dataset is split into train (75,910), dev (10,845) and test (21,687) sets. This benchmark also provides two subsets of Freebase: FB2M (2,150,604 entities, 6,701 predicates, 14,180,937 atomic facts), FB5M (4,904,397 entities, 7,523 predicates, 22,441,880 atomic facts). While single-relation questions are easier to handle than questions with more complex and multiple relations, single-relation question answering is still far from being solved. Even in this restricted domain there are a large number of paraphrases of the same question. Thus, the problem of mapping from a question to a particular predicate and entity in Freebase is hard.

The task assumes that single-relation questions can be answered by querying a knowledge base such as Freebase with a single subject and predicate argument. Hence, only the tuple (h, p) is used to match the question. The evaluation metric is accuracy. Only a fact that matches the ground truth in both subject and predicate is counted as correct.

4 Entity Linking

Given a question, the entity linker provides a set of top-N entity candidates. The input of our deep learning model are (subject, predicate) and (mention, pattern) pairs. Thus, given a question, two problems

we have to solve are (i) identifying candidate entities in Freebase that the question refers to and (ii) identifying the span (i.e., mention) in the question that refers to the entity. Each problem can be handled before the other, which results in two entity linkers. (i) **Passive Entity Linker**: First search for entity candidates *by all question words*, then use returned entities to guide the mention detection; (ii) **Active Entity Linker**: First identify the entity mention in the question, then *use the mention span* to search for entity candidates. We now introduce them in detail.

Passive Entity Linker. We perform entity linking by deriving the *longest consecutive common subsequence* (LCCS) between a question and entity candidates and refer to it as σ. Given a question q and all entity names from Freebase, we perform the following three steps.

(i) Lowercase/tokenize entity names and question

(ii) Use each component word of q to retrieve entities whose names contain this word. We refer to the set of all these entities as C_e.

(iii) For each entity candidate e in C_e, compute its LCCS σ with the question q. Let p be the position of the last token of σ in q. Compute $a = |\sigma|/|q|$, $b = |\sigma|/|e|$ and $c = p/|q|$ where $|\cdot|$ is length in words. Finally, entity candidate e is scored by the weighted sum $s_e = \alpha a + \beta b + (1 - \alpha - \beta)c$. Parameters α and β are tuned on dev. Top-N ranked entities are kept for each question.

Discussion. Factor $a = |\sigma|/|q|$ means we prefer candidates that cover *more consecutive words* of the question. Factor $b = |\sigma|/|e|$ means that we prefer the candidates that cover *more consecutive words of the entity*. Factor $c = p/|q|$ means that we prefer candidates that appear *close to the end of the question*; this is based on the observation that most entity mentions are far from the beginning of the question. Despite the simplicity of this passive entity linker, it outperforms other state-of-the-art entity linkers of this SimpleQuestions task by a big margin. Besides, this entity linker is unsupervised and runs fast. We will show its promise and investigate the individual contributions of the three factors in experiments.

Each question q is provided top-N entity candidates from Freebase by entity linker. Then for *mention detection*, we first compute the LCCS σ *on word level* between q and entity e. If the entity is longer than σ and has l (resp. r) words on the left (resp. right) of σ, then we extend σ in the question by l left (resp. r right) words and select this subsequence as the candidate mention. For example, entity "U.S. Route 2" and question "what major cities does us route 2 run through" have LCCS σ "route 2". The FB entity "U.S. Route 2" has one extra word "u.s." on the left of σ, so we extend σ by one left word and the candidate mention is "us route 2". We do this so that the mention has the same word size as the entity string.[3]

In rare cases that the LCCS on the word level has length 0, we treat both entity string and question as character sequence, then compute LCCS σ on character level. Finally, mention is formed by expanding σ on both sides up to a space or the text boundary.

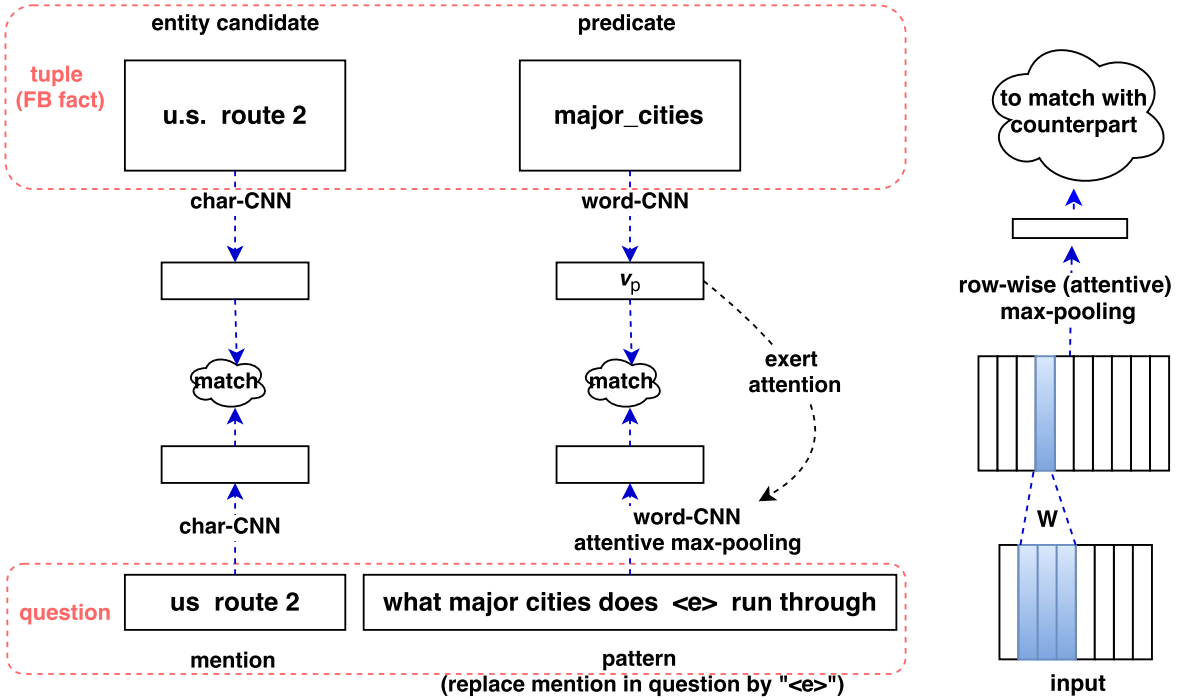
For each question, this approach to mention detection usually produces *more than one (mention, pattern) pair*.

Active Entity Linker. In the training set of SimpleQuestions, the topic entity of each question is labeled. Active entity linker is then achieved by detecting mention in a question by sequential labeling. The key idea is to train a model to predict the text span of the topic entity which can match the gold entity. This is inspired by some prior work. For example, Dai et al. (2016) map the gold entity back to the text to label the text span for each question and then train a BiGRU-CRF model to do the mention detection. Golub and He (2016) propose a generative model which generates the topic entity based on character-level text spans with soft attention scores. Similar to the work (Dai et al., 2016), we trained a BiLSTM-CRF model to detect the entity mentions.

This approach to mention detection produces *only one (mention, pattern) pair* for each question. Then, based on this detected mention, *we use each word of it* to search for the entity candidates via the three steps in "Passive Entity Linker".

We presented two styles of mention detection in questions – passive or active. In passive mention detection, the mention of a question depends on the entity candidates returned by an entity linker. Due

[3]Only using LCCS as mention performed worse.



(a) The whole system. Question: what major cities does us route 2 run through; Tuple: ("u.s. route 2", "major_cities")

(b) Convolution for representation learning

Figure 1: CNN System for SimpleQA

to the different furface-forms of entity candidates, a question can be detected in different spans as mentions. Instead, active mention detection is conducted in a similar way with Name Entity Recognition. Hence, the mention does not depend on the returned entity candidates, a single-relation question has only one mention. Our experiments will show that active entity linker bring better coverage of ground truth entities, nevertheless this method requires the availability of entity-labeled questions as training data.

After mention detection, we then convert the question into the tuple (mention, pattern) where pattern is created by replacing the mention in the question with <e>.

5 Fact Selection

Entity linker provides top-N entity candidates for each question. All facts having those entities as subject form a fact pool, then we build the system to seek the best.

Our whole system is depicted in Figure 1(a). It consists of match from two aspects: (i) a CNN on character level (char-CNN) to detect the similarity of entity string and the mention string in surface-form (the left column); (ii) a CNN with attentive maxpooling (AMP) in word level (word-AMPCNN) to detect if the predicate is a paraphrase of the pattern.

Word-AMPCNN is motivated by the observation that the FB predicate name is short and fixed whereas the corresponding pattern in the question is highly variable in length and word choice. Our hypothesis is that the predicate-pattern match is best done based on keywords in the pattern (and perhaps humans also do something similar) and that the CNN therefore should identify helpful keywords. Traditional maxpooling treats *all n-grams equally*. In this work, we propose *attentive maxpooling* (AMP). AMP gives *higher weights to n-grams that better match the predicate*. As a result, the predicate-pattern match computed by the CNN is more likely to be correct.

Next, we introduce the CNN combined with maxpooling for both char-CNN and word-CNN, then present AMPCNN. Figure 1(b) shows the common framework of char-CNN and word-CNN; only input granularity and maxpooling are different.

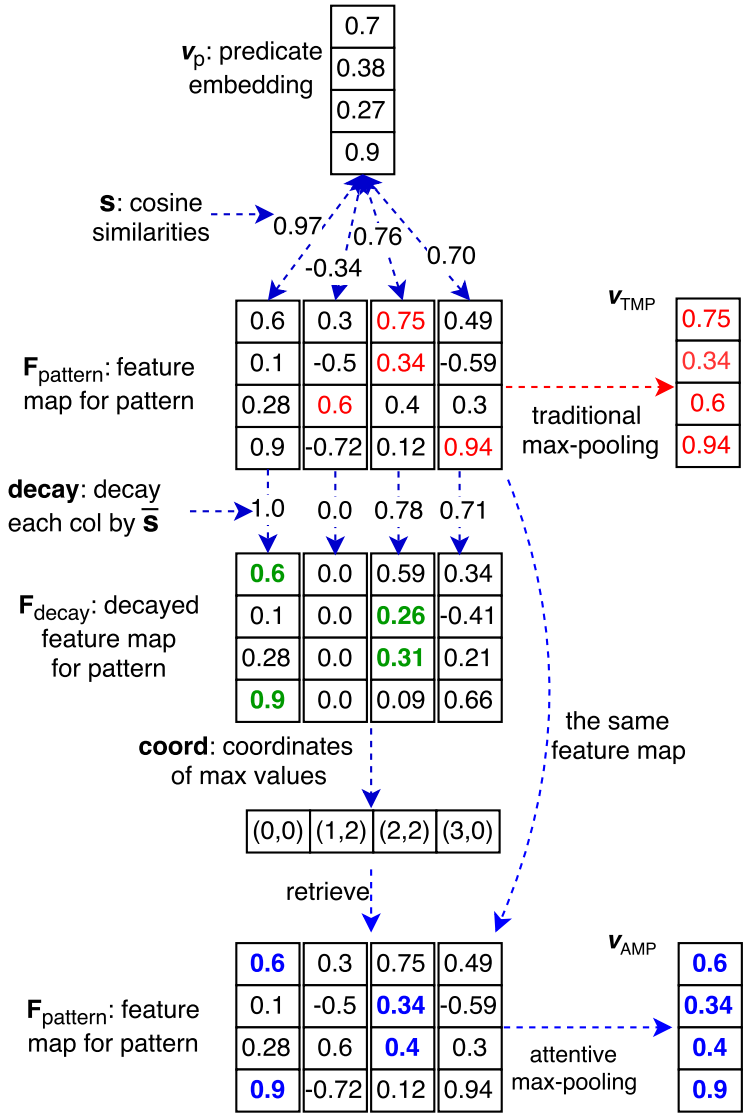


Figure 2: Traditional maxpooling vs. Attentive maxpooling

5.1 Framework of CNN-Maxpooling

Both char-CNN and word-CNN have two weight-sharing CNNs, as they model two pieces of text. In what follows, we use "entry" as a general term for both character and word.

The **input layer** is a sequence of entries of length s where each entry is represented by a d-dimensional randomly initialized embedding; thus the sequence is represented as a feature map of dimensionality $d \times s$. Figure 1(b) shows the input layer as the lower rectangle with multiple columns.

Convolution Layer is used for representation learning from sliding n-grams. For an input sequence with s entries: $v_1, v_2, \ldots, v_s$, let vector $\mathbf{c}_i \in \mathbb{R}^{nd}$ be the concatenated embeddings of n entries $v_{i-n+1}, \ldots, v_i$ where n is the filter width and $0 < i < s + n$. Embeddings for v_i, $i < 1$ or $i > s$, are zero padded. We then generate the representation $\mathbf{p}_i \in \mathbb{R}^d$ for the n-gram $v_{i-n+1}, \ldots, v_i$ using the convolution weights $\mathbf{W} \in \mathbb{R}^{d \times nd}$:

$$\mathbf{p}_i = \tanh(\mathbf{W} \cdot \mathbf{c}_i + \mathbf{b}) \tag{1}$$

where bias $\mathbf{b} \in \mathbb{R}^d$.

Maxpooling. All n-gram representations $\mathbf{p}_i$ $(i = 1 \cdots s+n-1)$ are used to generate the representation of input sequence $\mathbf{s}$ by maxpooling: $\mathbf{s}_j = \max(\mathbf{p}_{j1}, \mathbf{p}_{j2}, \cdots)$ $(j = 1, \cdots, d)$.

5.2 AMPCNN: CNN-Attentive-Maxpooling

Figure 2 shows TMP (Traditional MaxPooling) and AMP (Attentive MaxPooling) as we apply them to SimpleQA. Recall that we use standard CNNs to produce (i) the predicate representation $\mathbf{v}_p$ (see Figure 1(a)) and (ii) a feature map of the pattern, i.e., a matrix with columns denoting n-gram representations (shown in Figure 1(b), the matrix below "row-wise (attentive) maxpooling"). In Figure 2, we refer to the feature map as $\mathbf{F}_{\text{pattern}}$ and to the predicate representation as $\mathbf{v}_p$.

TMPCNN, i.e., traditional maxpooling, outputs the vector shown as $\mathbf{v}_{\text{TMP}}$; the same $\mathbf{v}_{\text{TMP}}$ is produced for different $\mathbf{v}_p$. The basic idea of AMPCNN is to let the predicate $\mathbf{v}_p$ *bias the selection and weighting of subsequences of the question to compute the representation of the pattern*. The first step in doing that is to compute similarity scores $\mathbf{s}$ between the predicate representation $\mathbf{v}_p$ and each column vector of $\mathbf{F}_{\text{pattern}}$:

$$\mathbf{s}_i = \mathbf{cos}(\mathbf{v}_p, \mathbf{F}_{\text{pattern}}[:, i]) \tag{2}$$

These cosines are then transformed into *decay* values by setting negative values to 0 (negatively correlated column vectors are likely to be unrelated to the predicate) and normalizing the positive values by dividing them by the largest cosine (.97 in this case), so that the largest decay value is 1.0. This is shown as "decay" and $\bar{\mathbf{s}}$ in the figure. Finally, we compute the reweighted feature map $\mathbf{F}_{\text{decay}}$ as follows:

$$\mathbf{F}_{\text{decay}}[:, i] = \mathbf{F}_{\text{pattern}}[:, i] * \bar{\mathbf{s}}_i \tag{3}$$

In $\mathbf{F}_{\text{decay}}$, the matrix with four green values, we can locate the maximal values in each dimension. Notice that they are not the true features by CNN any more, instead, *they convey the original feature values as well as their importance to be considered*. In $\mathbf{F}_{\text{decay}}$, we can see that the maximal values in each dimension mostly come from the first column and the third column which have relatively higher similarity scores 0.97 and 0.76 respectively to the predicate. *We use the coordinates of those maximal values to retrieve features from* $\mathbf{F}_{pattern}$ as a final pattern representation $\mathbf{v}_{\text{AMP}}$, the blue column vector[4].

In summary, TMP has no notion of context. The novelty of AMP is that it is guided by attentions from the context, in this case attentions from the predicate. In contrast to TMP, we expect AMP to mainly extract features that come from n-grams that are related to the predicate.

6 Experiments

6.1 Training Setup

Our fact pool consists of all facts whose subject entity is in the top-N entity candidates. For train, we sample 99 negative facts for each ground truth fact; for dev and test, all fact candidates are kept.

Figure 1(a) shows two-way match between a tuple t and a question q: entity-mention match by char-CNN (score m_e), predicate-pattern match by word-AMPCNN (score m_r). The overall ranking score of the pair is $s_t(q, t) = m_e + m_r + s_e$ where s_e is the entity ranking score in entity linking phase.

Our objective is to minimize ranking loss:

$$l(q, t^+, t^-) = \max(0, \lambda + s_t(q, t^-) - s_t(q, t^+)) \tag{4}$$

where λ is a constant.

We build word and character vocabularies on train. OOV words and characters from dev and test are mapped to an OOV index. Then, words (resp. characters) are randomly initialized into d_{word}-dimensional (resp. d_{char}-dimensional) embeddings. The output dimensionality in convolution, i.e., Equation 1, is the same as input dimensionality. We employ Adagrad (Duchi et al., 2011), L_2 regularization and diversity regularization (Xie et al., 2015). Hyperparameters (Table 1) are tuned on dev. For active mention detection, we trained a two-layer BiLSTM followed by a CRF, the hidden layer sizes of both BiLSTM are 200.

[4]We tried max-pooling over $\mathbf{F}_{\text{decay}}$ as $\mathbf{v}_{\text{AMP}}$ directly, but much worse performance was observed.

d_{word}	d_{char}	lr	L_2	div	k	λ
500	100	0.1	.0003	.03	[3,3]	0.5

Table 1: Hyperparameters. d_{word}/d_{char}: embedding dimensionality; lr: learning rate; L_2: L_2 normalization; div: diversity regularizer; k: filter width in char/word-CNN. λ: see Eq. 4

	baseline		**Ours**				
N	**raw**	**rerank**	**passive-linker**	**-a**	**-b**	**-c**	**active-linker**
1	40.9	52.9	**56.6**	11.0	34.9	52.3	**73.6**
5	–	–	**71.1**	29.5	49.5	67.7	**85.0**
10	64.3	74.0	**75.2**	40.7	56.6	72.8	**87.4**
20	69.3	77.8	**81.0**	63.3	62.4	78.6	**88.8**
50	75.7	82.0	**85.7**	77.1	67.1	84.2	**90.4**
100	79.6	85.4	**87.9**	81.2	70.4	87.0	**91.6**

Table 2: Experimental results for entity linking

6.2 Entity Linking

In Table 2, we compare our (passive and active) entity linkers with the state-of-the-art entity linker (Golub and He, 2016) in this SimpleQA task. Golub and He (2016) report the coverage of ground truth by top-N cases ($N \in \{1, 10, 20, 50, 100\}$). In addition, they explore a reranking algorithm to refine the entity ranking list.

Table 2 first shows the *overall* performance of our passive entity linker and its performance without factor a, b or c (-a, -b, -c). Our passive entity linker outperforms the baseline's *raw* results by big margins and is 2–3 percent above their *reranked* scores. This shows the outstanding performance of our passive entity linker despite its simplicity. The table also shows that all three factors (a, b, c) matter. Observations: (i) Each factor matters more when N is smaller. This makes sense because when N reaches the entity vocabulary size, all methods will have coverage 100%. (ii) The position-related factor c has less influence. From top1 to top100, its contribution decreases from 4.3 to .9. Our linker still outperforms the reranked baseline for $N \geq 20$. (iii) Factor a is dominant for small N, presumably because it chooses the longer one when two candidates exist, which is critical for small N. (iv) Factor b plays a more consistent role across different N.

The last column of Table 2 shows the overall results of our active entity linker, which are significantly better than the results of baseline linker and our passive linker. *We release our entity linking results for follow-up work to make better comparison.*

6.3 SimpleQuestions

Table 3 compares AMPCNN with two baselines. (i) **MemNN** (Bordes et al., 2015), an implementation of memory network for SimpleQuestions task. (ii) **Encoder-Decoder** (Golub and He, 2016), a character-level, attention-based encoder-decoder LSTM (Hochreiter and Schmidhuber, 1997) model. (iii) **CFO** (Dai et al., 2016), the state-of-the-art system in this task with CNN or BiGRU subsystem.

We report results for both passive entity linker and active entity linker. Furthermore, we compare AMPCNN to TMPCNN, i.e., we remove attention and representations for the predicate-pattern match are computed without attention. We choose top-20 (i.e., $N = 20$) entities returned by entity linker. Table 3 shows that AMPCNN with active entity linker has optimal performance for FB2M and FB5M. Performance on FB5M is slightly lower than on FB2M, which should be mainly due to the lower coverage for entity linking on FB5M – about 2% below that on FB2M. In addition, our CNN can still get competitive performance even if the attention mechanism is removed (TMPCNN result). This hints that CNN is promising for SimpleQA.

Settings		Methods	FB2M	FB5M
passive entity linker	baseline	random guess	4.9	4.9
		MemNN	62.7	63.9
		CFO w/ CNN	-	56.0
		CFO w/ BiGRU	-	62.6
	CNN	TMPCNN	**67.5***	**66.6***
		AMPCNN	**68.3***	**67.2***
active entity linker	baseline	Encoder-Decoder	70.9	70.3
		CFO w/ CNN	-	71.1
		CFO w/ BiGRU	-	75.7
	CNN	TMPCNN	75.4	74.6
		AMPCNN	**76.4***	**75.9**

Table 3: Experimental results for SimpleQuestions. Significant improvements over top baseline are marked with * (test of equal proportions, $p < .05$).

	RC	Para
OWA-HABCNN (Yin et al., 2016a)	.847	0
OWA-ABCNN (Yin et al., 2016b)	.902	0
OWA-APCNN (Santos et al., 2016)	.905	$\mathbb{R}^{d\times d}$
AMPCNN	.913	0

Table 4: Comparing different attention schemes of CNN in terms of RC, *extra* parameters brought (Para).

6.4 Effect of Attentive Maxpooling (AMP)

We compare AMP (one main contribution of this work) with three CNN attention mechanisms that are representative of related work in modeling two pieces of text: (i) **HABCNN**: Hierarchical attention-based CNN (Yin et al., 2016a); (ii) **ABCNN**: Attention-based CNN (Yin et al., 2016b); (iii) **APCNN**: CNN with attentive pooling (Santos et al., 2016).

Since attentive matching of predicate-pattern is only one part of our jointly trained system, it is hard to judge whether or not an attentive CNN performs better than alternatives. We therefore create a relation classification (RC) subtask to compare AMP with baseline schemes directly. RC task is created based on SimpleQuestions: label each question (converted into a pattern first) with the ground truth predicate; all other predicates of the gold subject entity are labeled as negative. The resulting datasets have sizes 72,239 (train), 10,310 (dev) and 20,610 (test). It is worth mentioning that this relation classification task is not unspecific to the SimpleQA task, as RC is actually the predict-pattern match part. Hence, this RC subtask can be viewed to check how well the predict-pattern subsystem performs within the whole architecture, and the effectiveness of various attention mechanisms is more clear.

In the three baselines, two pieces of text apply attention to each other. We adapt them into *one-way attention* (OWA) as AMP does in this work: fix predicate representation, and use it to guide the learning of pattern representation. To be specific, ABCNN first gets predicate representation by mean pooling, then uses this representation to derive similarity scores of each n-gram in pattern as attention scores, finally averages all n-gram embeddings weighted by attentions as pattern representation. HABCNN first gets predicate representation by max pooling, then computes attention scores the same way as ABCNN, finally does maxpooling over representations of top-k similar n-grams. APCNN is similar to ABCNN except that the similarity scores are computed by a nonlinear bilinear form.

Table 4 shows that AMPCNN performs well on relation classification, outperforming the best baseline APCNN by 0.8%. AMPCNN also has fewer parameters and runs faster than APCNN.

7 Conclusion

This work explored CNNs for the SimpleQA task. We made two main contributions. (i) A simple and effective entity linker that brings higher coverage of ground truth entities. (ii) An attentive maxpooling stacked above convolution layer that models the relationship between predicate and question pattern more effectively. Our model shows outstanding performance on both simpleQA and relation classification.

Acknowledgments. Wenpeng Yin and Hinrich Schütze were partially supported by DFG (grant SCHU 2246/8-2).

References

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of CIKM*, pages 1431–1440.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*, pages 1533–1544.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*, pages 1247–1250.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of EMNLP*, pages 615–620.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Zihang Dai, Lei Li, and Wei Xu. 2016. CFO: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of ACL*, pages 800–810.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of ACL-IJCNLP*, volume 1, pages 260–269.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of EMNLP*, pages 1535–1545.

Anthony Fader, Luke S Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of ACL*, pages 1608–1618.

David Golub and Xiaodong He. 2016. Character-level question answering with attention. In *Proceedings of EMNLP*.

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

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of CIKM*, pages 2333–2338.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of ACL*, pages 588–598.

Dominic Seyler, Mohamed Yahya, and Klaus Berberich. 2015. Generating quiz questions from knowledge graphs. In *Proceedings of WWW*, pages 113–114.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Proceedings of NIPS*, pages 2431–2439.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of ICLR*.

Pengtao Xie, Yuntian Deng, and Eric Xing. 2015. On the generalization error bounds of neural networks under diversity-inducing mutual angular regularization. *arXiv preprint arXiv:1511.07110*.

Kun Xu, Yansong Feng, Siva Reddy, Songfang Huang, and Dongyan Zhao. 2016. Enhancing freebase question answering using textual evidence. *arXiv preprint arXiv:1603.00957*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2013. Robust question answering over the web of linked data. In *Proceedings of CIKM*, pages 1107–1116.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with Freebase. In *Proceedings of ACL*, pages 956–966.

Xuchen Yao, Jonathan Berant, and Benjamin Van Durme. 2014. Freebase QA: Information extraction or semantic parsing? In *Proceedings of ACL Workshop on Semantic Parsing*, pages 82–86.

Xuchen Yao. 2015. Lean question answering over freebase from scratch. In *Proceedings of NAACL-HLT*, pages 66–70.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of ACL*, pages 643–648.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of ACL*, pages 1321–1331.

Wenpeng Yin, Sebastian Ebert, and Hinrich Schütze. 2016a. Attention-based convolutional neural network for machine comprehension. In *Proceedings of NAACL Human-Computer QA Workshop*.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016b. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *TACL*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *Proceedings of ICLR Workshop*.

Recurrent Dropout without Memory Loss

Stanislau Semeniuta[1] **Aliaksei Severyn**[2] **Erhardt Barth**[1]
[1]Universität zu Lübeck, Institut für Neuro- und Bioinformatik
{stas,barth}@inb.uni-luebeck.de
[2]Google Research
severyn@google.com

Abstract

This paper presents a novel approach to recurrent neural network (RNN) regularization. Differently from the widely adopted dropout method, which is applied to *forward* connections of feed-forward architectures or RNNs, we propose to drop neurons directly in *recurrent* connections in a way that does not cause loss of long-term memory. Our approach is as easy to implement and apply as the regular feed-forward dropout and we demonstrate its effectiveness for Long Short-Term Memory network, the most popular type of RNN cells . Our experiments on three NLP benchmarks show consistent improvements even when combined with conventional feed-forward dropout.

1 Introduction

Recurrent Neural Networks, LSTMs in particular, have recently become a popular tool among NLP researchers for their superior ability to model and learn from sequential data. These models have shown state-of-the-art results on various public benchmarks ranging from sentence classification (Wang et al., 2015; Irsoy and Cardie, 2014; Liu et al., 2015) and various tagging problems (Dyer et al., 2015) to language modelling (Kim et al., 2015; Zhang et al., 2015), text generation (Zhang and Lapata, 2014) and sequence-to-sequence prediction tasks (Sutskever et al., 2014).

Having shown excellent ability to capture and learn complex linguistic phenomena, RNN architectures are prone to overfitting. Among the most widely used techniques to avoid overfitting in neural networks is the dropout regularization (Hinton et al., 2012). Since its introduction it has become, together with the L2 weight decay, the standard method for neural network regularization. While showing significant improvements when used in feed-forward architectures, e.g., Convolutional Neural Networks (Krizhevsky et al., 2012), the application of dropout in RNNs has been somewhat limited. Indeed, so far dropout in RNNs has been applied in the same fashion as in feed-forward architectures: it is typically injected in input-to-hidden and hidden-to-output connections, i.e., along the input axis, but not between the recurrent connections (time axis). Given that RNNs are mainly used to model sequential data with the goal of capturing short- and long-term interactions, it seems natural to also regularize the recurrent weights. This observation has led us and other researchers (Moon et al., 2015; Gal, 2015) to the idea of applying dropout to the recurrent connections in RNNs.

In this paper we propose a novel *recurrent dropout* technique and demonstrate how our method is superiour to other recurrent dropout methods recently proposed in (Moon et al., 2015; Gal, 2015). Additionally, we answer the following questions which helps to understand how to best apply recurrent dropout: (i) how to apply the dropout in recurrent connections of the LSTM architecture in a way that prevents possible corruption of the long-term memory; (ii) what is the relationship between our *recurrent dropout* and the widely adopted dropout in input-to-hidden and hidden-to-output connections; (iii) how the dropout mask in RNNs should be sampled: once per step or once per sequence. The latter question of sampling the mask appears to be crucial in some cases to make the recurrent dropout work and, to

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1757–1766, Osaka, Japan, December 11-17 2016.

the best of our knowledge, has received very little attention in the literature. Our work is the first one to provide empirical evaluation of the differences between these two sampling approaches.

Regarding empirical evaluation, we first highlight the problem of information loss in memory cells of LSTMs when applying *recurrent dropout*. We demonstrate that previous approaches of dropping *hidden state* vectors cause loss of memory while our proposed method to use dropout mask in *hidden state update* vectors does not suffer from this problem. We experiment on three widely adopted NLP tasks: word- and character-level Language Modeling and Named Entity Recognition. The results demonstrate that our *recurrent dropout* helps to achieve better regularization and yields improvements across all the tasks, even when combined with the conventional feed-forward dropout. Furthermore, we compare our dropout scheme with the recently proposed alternative recurrent dropout methods and show that our technique is superior in almost all cases.

2 Related Work

Neural Network models often suffer from overfitting, especially when the number of network parameters is large and the amount of training data is small. This has led to a lot of research directed towards improving their generalization ability. Below we primarily discuss some of the methods aimed at improving regularization of RNNs.

Pham et al. (2013) and Zaremba et al. (2014) have shown that LSTMs can be effectively regularized by using dropout in forward connections. While this already allows for effective regularization of recurrent networks, it is intuitive that introducing dropout also in the hidden state may force it to create more robust representations. Indeed, Moon et al. (2015) have extended the idea of dropping neurons in forward direction and proposed to drop cell states as well showing good results on a Speech Recognition task. Bluche et al. (2015) carry out a study to find where dropout is most effective, e.g. input-to-hidden or hidden-to-output connections. The authors conclude that it is more beneficial to use it once in the correct spot, rather than to put it everywhere. Bengio et al. (2015) have proposed an algorithm called scheduled sampling to improve performance of recurrent networks on sequence-to-sequence labeling tasks. A disadvantage of this work is that the scheduled sampling is specifically tailored to this kind of tasks, what makes it impossible to use in, for example, sequence-to-label tasks. Gal (2015) uses insights from variational Bayesian inference to propose a variant of LSTM with dropout that achieves consistent improvements over a baseline architecture without dropout.

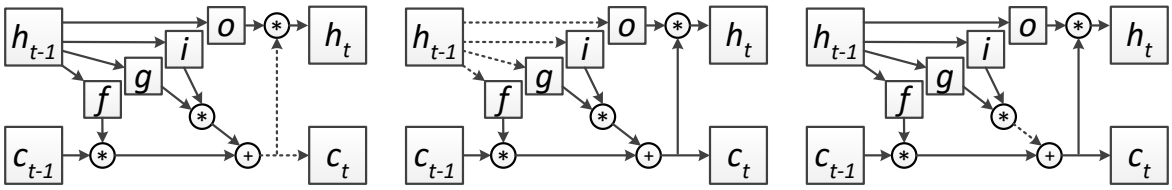
The main contribution of this paper is a new *recurrent dropout* technique, which is most useful in gated recurrent architectures such as LSTMs and GRUs. We demonstrate that applying dropout to arbitrary vectors in LSTM cells may lead to loss of memory thus hindering the ability of the network to encode long-term information. In other words, our technique allows for adding a strong regularizer on the model weights responsible for learning short and long-term dependencies without affecting the ability to capture long-term relationships, which are especially important to model when dealing with natural language. Finally, we compare our method with alternative *recurrent dropout* methods recently introduced in (Moon et al., 2015; Gal, 2015) and demonstrate that our method allows to achieve better results.

3 Recurrent Dropout

In this section we first show how the idea of feed-forward dropout (Hinton et al., 2012) can be applied to recurrent connections in vanilla RNNs. We then introduce our *recurrent dropout* method specifically tailored for gated architectures such as LSTMs and GRUs. We draw parallels and contrast our approach with alternative recurrent dropout techniques recently proposed in (Moon et al., 2015; Gal, 2015) showing that our method is favourable when considering potential memory loss issues in long short-term architectures.

3.1 Dropout in vanilla RNNs

Vanilla RNNs process the input sequences as follows:



(a) Moon et al., 2015 (b) Gal, 2015 (c) Ours

Figure 1: Illustration of the three types of dropout in recurrent connections of LSTM networks. Dashed arrows refer to dropped connections. Input connections are omitted for clarity.

$$\mathbf{h_t} = f(\mathbf{W_h}[\mathbf{x_t}, \mathbf{h_{t-1}}] + \mathbf{b_h}), \tag{1}$$

where $\mathbf{x_t}$ is the input at time step t; $\mathbf{h_t}$ and $\mathbf{h_{t-1}}$ are hidden vectors that encode the current and previous states of the network; $\mathbf{W_h}$ is parameter matrix that models input-to-hidden and hidden-to-hidden (recurrent) connections; $\mathbf{b}$ is a vector of bias terms, and f is the activation function.

As RNNs model sequential data by a fully-connected layer, dropout can be applied by simply dropping the previous hidden state of a network. Specifically, we modify Equation 1 in the following way:

$$\mathbf{h_t} = f(\mathbf{W_h}[\mathbf{x_t}, d(\mathbf{h_{t-1}})] + \mathbf{b_h}), \tag{2}$$

where d is the dropout function defined as follows:

$$d(\mathbf{x}) = \begin{cases} mask * \mathbf{x}, & \text{if train phase} \\ (1-p)\mathbf{x} & \text{otherwise,} \end{cases} \tag{3}$$

where p is the dropout rate and $mask$ is a vector, sampled from the Bernoulli distribution with success probability $1 - p$.

3.2 Dropout in LSTM networks

Long Short-Term Memory networks (Hochreiter and Schmidhuber, 1997) have introduced the concept of gated inputs in RNNs, which effectively allow the network to preserve its memory over a larger number of time steps during both forward and backward passes, thus alleviating the problem of vanishing gradients (Bengio et al., 1994). Formally, it is expressed with the following equations:

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \sigma(\mathbf{W}_i \left[\mathbf{x}_t, \mathbf{h}_{t-1}\right] + \mathbf{b}_i) \\ \sigma(\mathbf{W}_f \left[\mathbf{x}_t, \mathbf{h}_{t-1}\right] + \mathbf{b}_f) \\ \sigma(\mathbf{W}_o \left[\mathbf{x}_t, \mathbf{h}_{t-1}\right] + \mathbf{b}_o) \\ f(\mathbf{W}_g \left[\mathbf{x}_t, \mathbf{h}_{t-1}\right] + \mathbf{b}_g) \end{pmatrix} \tag{4}$$

$$\mathbf{c}_t = \mathbf{f}_t * \mathbf{c}_{t-1} + \mathbf{i}_t * \mathbf{g}_t \tag{5}$$

$$\mathbf{h}_t = \mathbf{o}_t * f(\mathbf{c}_t), \tag{6}$$

where $\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t$ are input, output and forget gates at step t; $\mathbf{g}_t$ is the vector of cell updates and $\mathbf{c}_t$ is the updated cell vector used to update the hidden state $\mathbf{h}_t$; σ is the sigmoid function and $*$ is the element-wise multiplication.

Gal (2015) proposes to drop the previous *hidden state* vectors when computing values of gates and updates of the current step, where he samples the dropout mask once for every sequence:

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \sigma(\mathbf{W}_i \left[\mathbf{x}_t, d(\mathbf{h}_{t-1})\right] + \mathbf{b}_i) \\ \sigma(\mathbf{W}_f \left[\mathbf{x}_t, d(\mathbf{h}_{t-1})\right] + \mathbf{b}_f) \\ \sigma(\mathbf{W}_o \left[\mathbf{x}_t, d(\mathbf{h}_{t-1})\right] + \mathbf{b}_o) \\ f(\mathbf{W}_g \left[\mathbf{x}_t, d(\mathbf{h}_{t-1})\right] + \mathbf{b}_g) \end{pmatrix} \tag{7}$$

Moon et al. (2015) propose to apply dropout directly to the cell values and use per-sequence sampling as well:

$$\mathbf{c}_t = d(\mathbf{f}_t * \mathbf{c}_{t-1} + \mathbf{i}_t * \mathbf{g}_t) \tag{8}$$

In contrast to dropout techniques proposed by Gal (2015) and Moon et al. (2015), we propose to apply dropout to the *cell update* vector $\mathbf{g}_t$ as follows:

$$\mathbf{c}_t = \mathbf{f}_t * \mathbf{c}_{t-1} + \mathbf{i}_t * d(\mathbf{g}_t) \tag{9}$$

Different from methods of (Moon et al., 2015; Gal, 2015), our approach does not require sampling of the dropout masks once for every training sequence. On the contrary, as we will show in Section 4, networks trained with a dropout mask sampled per-step achieve results that are at least as good and often better than per-sequence sampling. Figure 1 shows differences between approaches to dropout.

The approach of (Gal, 2015) differs from ours in the overall strategy – they consider network's hidden state as input to subnetworks that compute gate values and cell updates and the purpose of dropout is to regularize these subnetworks. Our approach considers the architecture as a whole with the hidden state as its key part and regularize the whole network. The approach of (Moon et al., 2015) on the other hand is seemingly similar to ours. In Section 3.3 we argue that our method is a more principled way to drop recurrent connections in gated architectures.

It should be noted that while being different, the three discussed dropout schemes are not mutually exclusive. It is in general possible to combine our approach and the other two. We expect the merge of our scheme and that of (Gal, 2015) to hold the biggest potential. The relations between recurrent dropout schemes are however out of scope of this paper and we rather focus on studying the relationships of different dropout approaches with the conventional forward dropout.

Lastly, we note that our dropout is also applicable to the recently introduced Gated Recurrent Unit (GRU) networks (Cho et al., 2014). GRU networks are built on the same design principles as LSTM networks and our dropout technique applies in a similar fashion.

3.3 Dropout and memory

We found that an intuitive idea to drop previous hidden states directly, as proposed in Moon et al. (2015), produces mixed results. We have observed that it helps the network to generalize better when not coupled with the forward dropout, but is usually no longer beneficial when used together with a regular forward dropout.

The problem is caused by the scaling of neuron activations during inference. Consider the hidden state update rule in the test phase of an LSTM network. For clarity, we assume every gate to be equal to 1:

$$\mathbf{h}_t = (\mathbf{h}_{t-1} + \mathbf{g}_t)p, \tag{10}$$

where $\mathbf{g}_t$ are update vectors computed by Eq. 4 and p is the probability to not drop a neuron. As h_{t-1} was, in turn, computed using the same rule, we can rewrite this equation as:

$$\mathbf{h}_t = ((\mathbf{h}_{t-2} + \mathbf{g}_{t-1})p + \mathbf{g}_t)p \tag{11}$$

Recursively expanding $\mathbf{h}$ for every timestep results in the following equation:

$$\mathbf{h}_t = ((((\mathbf{h}_0 + \mathbf{g}_0)p + \mathbf{g}_1)p + ...)p + \mathbf{g}_t)p \tag{12}$$

Pushing p inside parenthesis, Eq. 12 can be written as:

$$\mathbf{h}_t = p^{t+1}\mathbf{h}_0 + \sum_{i=0}^{t} p^{t-i+1}\mathbf{g}_i \tag{13}$$

Since p is a value between zero and one, sum components that are far away in the past are multiplied by a very low value and are effectively removed from the summation. Thus, even though the network is

able to learn long-term dependencies, it is not capable of exploiting them during test phase. Note that our assumption of all gates being equal to 1 helps the network to preserve hidden state, since in a real network gate values lie within (0, 1) interval. In practice trained networks tend to saturate gate values (Karpathy et al., 2015) what makes gates to behave as binary switches. The fact that Moon et al. (2015) have achieved an improvement can be explained by the experimentation domain. Le et al. (2015) have proposed a simple yet effective way to initialize vanilla RNNs and reported that they have achieved a good result in the Speech Recognition domain while having an effect similar to the one caused by Eq. 13. One can reduce the influence of this effect by selecting a low dropout rate. This solution however is partial, since it only increases the number of steps required to completely forget past history and does not remove the problem completely.

One important note is that the dropout function from Eq. 3 can be implemented as:

$$d(\mathbf{x}) = \begin{cases} mask * \mathbf{x}/p, & \text{if train phase} \\ \mathbf{x} & \text{otherwise} \end{cases} \tag{14}$$

In this case the above argument holds as well, but instead of observing exponentially decreasing hidden states during testing, we will observe exponentially increasing values of hidden states during training.

Our approach addresses the problem discussed previously by dropping the update vectors $\mathbf{g}$. Since we drop only candidates, we do not scale the hidden state directly. This allows for solving the scaling issue, as Eq. 13 becomes:

$$\mathbf{h}_t = p\mathbf{h}_0 + \sum_{i=0}^{t} p\ \mathbf{g}_i = p\mathbf{h}_0 + p\sum_{i=0}^{t} \mathbf{g}_i \tag{15}$$

Moreover, since we only drop differences that are added to the network's hidden state at each time-step, this dropout scheme allows us to use per-step mask sampling while still being able to learn long-term dependencies. Thus, our approach allows to freely apply dropout in the recurrent connections of a gated network without hindering its ability to process long-term relationships.

We note that the discussed problem does not affect vanilla RNNs because they overwrite their hidden state at every timestep. Lastly, the approach of Gal (2015) is not affected by the issue as well.

4 Experiments

First, we empirically demonstrate the issues linked to memory loss when using various dropout techniques in recurrent nets (see Sec. 3.3). For this purpose we experiment with training LSTM networks on one of the synthetic tasks from (Hochreiter and Schmidhuber, 1997), specifically the Temporal Order task. We then validate the effectiveness of our *recurrent dropout* on three public benchmarks: word and character-level Language Modeling and Named Entity Recognition comparing directly to alternative *recurrent dropout* methods from (Moon et al., 2015; Gal, 2015).

4.1 Synthetic Task

Data. In this task the input sequences are generated as follows: all but two elements in a sequence are drawn randomly from {C, D} and the remaining two symbols from {A, B}. Symbols from {A, B} can appear at any position in the sequence. The task is to classify a sequence into one of four classes ({AA, AB, BA, BB}) based on the order of the symbols. We generate data so that every sequence is split into three parts with the same size and emit one meaningful symbol in first and second parts of a sequence. The prediction is taken after the full sequence has been processed. We use two modes in our experiments: `Short` with sequences of length 15 and `Medium` with sequences of length 30.

Setup. We use LSTM with one layer that contains 256 hidden units and *recurrent dropout* with 0.5 strength. Network is trained by SGD with a learning rate of 0.1 for 5k epochs. The networks are trained on 200 mini-batches with 32 sequences and tested on 10k sequences.

Results. Table 1 reports the results on the Temporal Order task when *recurrent dropout* is applied using our method and methods from (Moon et al., 2015) and (Gal, 2015). Using dropout from (Moon et al., 2015) with per-sequence sampling, networks are able to discover the long-term dependency, but fail to

Sampling	Moon et al. (2015)				Gal (2015); Ours			
	short sequences		medium sequences		short sequences		medium sequences	
	Train	Test	Train	Test	Train	Test	Train	Test
per-step	100%	100%	25%	25%	100%	100%	100%	100%
per-sequence	100%	25%	100%	<25%	100%	100%	100%	100%

Table 1: Accuracies on the Temporal Order task.

Dropout rate	Sampling	Moon et al. (2015)		Gal (2015)		Ours	
		Valid	Test	Valid	Test	Valid	Test
0.0	–	130.0	125.2	130.0	125.2	130.0	125.2
0.25	per-step	**113.0**	**108.7**	119.8	114.2	106.1	100.0
0.5	per-step	124.0	116.5	**118.3**	**112.5**	102.8	98.0
0.25	per-sequence	121.0	113.0	120.5	114.0	106.3	100.7
0.5	per-sequence	137.7	126.2	125.2	117.9	**103.2**	**96.8**
0.0	–	**94.1**	**89.5**	94.1	89.5	94.1	89.5
0.25	per-step	113.5	105.8	**92.9**	**88.4**	**91.6**	**87.0**
0.5	per-step	140.6	130.1	98.6	92.5	100.6	95.5
0.25	per-sequence	105.7	99.9	94.5	89.7	92.4	87.6
0.5	per-sequence	125.4	117.4	98.4	92.5	107.8	101.8

Table 2: Perplexity scores of the LSTM network on word level Language Modeling task (lower is better). Upper and lower parts of the table report results without and with forward dropout respectively. Networks with forward dropout use 0.2 and 0.5 dropout rates in input and output connections respectively. Values in bold show best results for each of the recurrent dropout schemes with and without forward dropout.

use it on the test set due to the scaling issue. Interestingly, in `Medium` case results on the test set are worse than random. Networks trained with per-step sampling exhibit different behaviour: in `Short` case they are capable of capturing the temporal dependency and generalizing to the test set, but require 10-20 times more iterations to do so. In `Medium` case these networks do not fit into the allocated number of iterations. This suggests that applying dropout to hidden states as suggested in (Moon et al., 2015) corrupts memory cells hindering the long-term memory capacity of LSTMs.

In contrast, using our *recurrent dropout* methods, networks are able to solve the problem in all cases. We have also ran the same experiments for longer sequences, but found that the results are equivalent to the `Medium` case. We also note that the approach of (Gal, 2015) does not seem to exhibit the memory loss problem.

4.2 Word Level Language Modeling

Data. Following Mikolov et al. (2011) we use the Penn Treebank Corpus to train our Language Modeling (LM) models. The dataset contains approximately 1 million words and comes with pre-defined training, validation and test splits, and a vocabulary of 10k words.

Setup. In our LM experiments we use recurrent networks with a single layer with 256 cells. Network parameters were initialized uniformly in [-0.05, 0.05]. For training, we use plain SGD with batch size 32 with the maximum norm gradient clipping (Pascanu et al., 2013). Learning rate, clipping threshold and number of Backpropagation Through Time (BPTT) steps were set to 1, 10 and 35 respectively. For the learning rate decay we use the following strategy: if the validation error does not decrease after each epoch, we divide the learning rate by 1.5. The aforementioned choices were largely guided by the work of Mikolov et al. (2014). To ease reproducibility of our results on the LM and synthetic tasks, we have

Dropout rate	Sampling	Moon et al. (2015)		Gal (2015)		Ours	
		Valid	Test	Valid	Test	Valid	Test
0.0	–	1.460	1.457	1.460	1.457	1.460	1.457
0.25	per-step	1.435	1.394	1.345	1.308	**1.338**	**1.301**
0.5	per-step	1.610	1.561	1.387	1.348	1.355	1.316
0.25	per-sequence	**1.433**	**1.390**	**1.341**	**1.304**	1.356	1.319
0.5	per-sequence	1.691	1.647	1.408	1.369	1.496	1.450
0.0	–	**1.362**	**1.326**	**1.362**	**1.326**	1.362	1.326
0.25	per-step	1.471	1.428	1.381	1.344	**1.358**	**1.321**
0.5	per-step	1.668	1.622	1.463	1.425	1.422	1.380
0.25	per-sequence	1.455	1.413	1.387	1.348	1.403	1.363
0.5	per-sequence	1.681	1.637	1.477	1.435	1.567	1.522

Table 3: Bit-per-character scores of the LSTM network on character level Language Modelling task (lower is better). Upper and lower parts of the table report results without and with forward dropout respectively. Networks with forward dropout use 0.2 and 0.5 dropout rates in input and output connections respectively. Values in bold show best results for each of the recurrent dropout schemes with and without forward dropout.

released the source code of our experiments[1].

Results. Table 2 reports the results for LSTM networks. We also present results when the dropout is applied directly to hidden states as in (Moon et al., 2015) and results of networks trained with the dropout scheme of (Gal, 2015). In addition, we report results of networks trained with no regularization and with dropout in only forward connetions in first rows of upper and lower parts of the table respectivelly. We make the following observations: (i) our approach shows better results than the alternatives; (ii) per-step mask sampling is better when dropping hidden state directly; (iii) on this task our method using per-step sampling seems to yield results similar to per-sequence sampling; (iv) in this case forward dropout yields better results than any of the three recurrent dropouts; and finally (v) both our approach and that of (Gal, 2015) are effective when combined with the forward dropout, though ours is more effective.

4.3 Character Level Language Modeling

Data. We train our networks on the dataset described in the previous section. It contains approximately 6 million characters, and a vocabulary of 50 characters. We use the provided partitions train, validation and test partitions.

Setup. We use networks with 1024 units to solve the character level LM task. The characters are embedded into 256 dimensional space before being processed by the LSTM. All parameters of the networks are initialized uniformly in [-0.01, 0.01]. We train our networks on non-overlapping sequences of 100 characters. The networks are trained with the Adam (Kingma and Ba, 2014) algorithm with initial learning rate of 0.001 for 50 epochs. We decrease the learning rate by 0.97 after every epoch starting from epoch 10. To avoid exploding gradients, we use MaxNorm gradient clipping with threshold set to 10.

Results. Results of our experiments are given in Table 3. Note that on this task regularizing only the recurrent connections is more beneficial than only the forward ones. In particular, LSTM networks trained with our approach and the approach of (Gal, 2015) yield a lower bit-per-character (bpc) score than those trained with forward dropout onlyWe attribute it to pronounced long term dependencies. In addition, our approach is the only one that improves over baseline LSTM with forward dropout. The overall best result is achieved by a network trained with our dropout with 0.25 dropout rate and per-step sampling, closely followed by network with Gal (2015) dropout.

[1] https://github.com/stas-semeniuta/drop-rnn

Dropout rate	Sampling	Moon et al. (2015)		Gal (2015)		Ours	
		Valid	Test	Valid	Test	Valid	Test
0.0	–	88.56	84.46	88.56	84.46	88.56	84.46
0.25	per-step	**88.79**	**84.80**	88.95	84.34	89.27	84.78
0.5	per-step	88.68	84.43	88.66	84.33	89.06	84.39
0.25	per-sequence	88.71	84.33	**88.54**	**84.88**	**89.32**	**84.95**
0.5	per-sequence	88.06	83.92	89.05	84.22	88.94	84.32
0.0	–	90.53	86.99	90.53	86.99	90.53	86.99
0.25	per-step	90.86	87.19	91.06	87.05	91.02	87.03
0.5	per-step	90.71	87.03	**90.76**	**87.23**	90.78	87.31
0.25	per-sequence	**90.73**	**87.32**	90.86	86.89	**90.99**	**87.33**
0.5	per-sequence	89.61	86.39	90.76	86.68	90.40	86.82

Table 4: F1 scores (higher is better) of the LSTM network on NER task (average scores over 3 runs). Upper and lower parts of the table report results without and with forward dropout respectively. Values in bold show best results for each of the recurrent dropout schemes with and without forward dropout.

4.4 Named Entity Recognition

Data. To assess our recurrent Named Entity Recognition (NER) taggers when using *recurrent dropout* we use a public benchmark from CONLL 2003 (Tjong Kim Sang and De Meulder, 2003). The dataset contains approximately 300k words split into train, validation and test partitions. Each word is labeled with either a named entity class it belongs to, such as `Location` or `Person`, or as being not named. The majority of words are labeled as not named entities. The vocabulary size is about 22k words.

Setup. Previous state-of-the-art NER systems have shown the importance of using word context features around entities. Hence, we slightly modify the architecture of our recurrent networks to consume the context around the target word by preprocessing the inputs by a convolutional layer. The size of the convolutional kernel is fixed to 5 words (the word to be labeled, two words before and two words after) and the number of filters is fixed to 256. The recurrent layer size is 1024 units. The network inputs include word embeddings (initialized with pretrained word2vec embeddings (Mikolov et al., 2013) and kept static) and capitalization features. For training we use the Adam algorithm (Kingma and Ba, 2014) with initial learning rate of 0.001. We train for 50 epochs and multiply the learning rate by 0.95 after every epoch starting at epoch 10. We also combine our *recurrent dropout* with the conventional forward dropout with the rate 0.2 in input and 0.5 in output connections. Lastly, we found that using $relu(x) = max(x, 0)$ nonlinearity resulted in higher performance than $tanh(x)$. We train our network on randomly extracted samples up to 15 words long and use full sentences for testing.

Results. Table 4 reports the results of networks trained with and without forward dropout and compares our algorithm to approaches of (Moon et al., 2015) and (Gal, 2015). We make the following observations: (i) forward dropout provides a much bigger improvement than recurrent one, what can be explained by the fact that long term dependencies are much less important in the NER task, in contrast to the Language Modeling; (ii) the results of our approach and dropout of (Gal, 2015) are comparable and both better than those of (Moon et al., 2015); and (iii) all three approaches consistently outperform baseline networks without dropout in recurrent connections.

5 Conclusions

This paper presents a novel *recurrent dropout* method specifically tailored to the gated recurrent neural networks. Our approach is easy to implement and is even more effective when combined with conventional forward dropout. We have shown that applying dropout to arbitrary cell vectors results in suboptimal performance. We discuss in detail the cause of this effect and propose a simple solution to overcome it. The effectiveness of our approach is verified on three public NLP benchmarks.

Our findings along with our empirical results help us to answer the questions posed in Section 1: (i) while is straight-forward to use dropout in vanilla RNNs due to their strong similarity with the feed-forward architectures, its application to LSTM networks is not so straightforward. We demonstrate that *recurrent dropout* is most effective when applied to *hidden state update* vectors in LSTMs rather than to *hidden states*; (ii) we observe an improvement in the network's performance when our *recurrent dropout* is coupled with the standard forward dropout, though the extent of this improvement depends on the values of dropout rates; (iii) per-step mask sampling is at least as good as per-sequence mask sampling when using our *recurrent dropout* method, with the most pronounced difference in the character level LM experiments, while the results of (Moon et al., 2015) and (Gal, 2015) are mixed.

Acknowledgments

This project has received funding from the European Union's Framework Programme for Research and Innovation HORIZON 2020 (2014-2020) under the Marie Skodowska-Curie Agreement No. 641805. Stanislau Semeniuta thanks the support from Pattern Recognition Company GmbH. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research.

References

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR*, abs/1506.03099.

Theodore Bluche, Christopher Kermorvant, and Jérôme Louradour. 2015. Where to apply dropout in recurrent neural networks for handwriting recognition? In *13th International Conference on Document Analysis and Recognition, ICDAR 2015, Tunis, Tunisia, August 23-26, 2015*, pages 681–685.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and A. Noah Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *ACL*, pages 334–343. Association for Computational Linguistics.

Yarin Gal. 2015. A theoretically grounded application of dropout in recurrent neural networks. *arXiv:1512.05287*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

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

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 720–728. Association for Computational Linguistics.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models. *CoRR*, abs/1508.06615.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *CoRR*, abs/1504.00941.

Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *ACL*. Association for Computational Linguistics.

T. Mikolov, S. Kombrink, L. Burget, J.H. Cernocky, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531, May.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michaël Mathieu, and Marc'Aurelio Ranzato. 2014. Learning longer memory in recurrent neural networks. *CoRR*, abs/1412.7753.

Taesup Moon, Heeyoul Choi, Hoshik Lee, and Inchul Song. 2015. Rnndrop: A novel dropout for rnns in asr. *Automatic Speech Recognition and Understanding (ASRU)*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1310–1318.

Vu Pham, Christopher Kermorvant, and Jérôme Louradour. 2013. Dropout improves recurrent neural networks for handwriting recognition. *CoRR*, abs/1312.4569.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xin Wang, Yuanchao Liu, Chengjie SUN, Baoxun Wang, and Xiaolong Wang. 2015. Predicting polarities of tweets by composing word embeddings with long short-term memory. In *ACL*, pages 1343–1353. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680. Association for Computational Linguistics.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2015. Tree recurrent neural networks with application to language modeling. *CoRR*, abs/1511.00060.

Modeling topic dependencies in semantically coherent text spans with copulas

Georgios Balikas *
Université Grenoble-Alpes
Computer Science Laboratory

Hesam Amoualian
Université Grenoble-Alpes
Computer Science Laboratory

Marianne Clausel
Université Grenoble-Alpes
Department of Statistics

Eric Gaussier
Université Grenoble-Alpes
Computer Science Laboratory

Massih-Reza Amini
Université Grenoble-Alpes
Computer Science Laboratory

Abstract

The exchangeability assumption in topic models like Latent Dirichlet Allocation (LDA) often results in inferring inconsistent topics for the words of text spans like noun-phrases, which are usually expected to be topically coherent. We propose copulaLDA, that extends LDA by integrating part of the text structure to the model and relaxes the conditional independence assumption between the word-specific latent topics given the per-document topic distributions. To this end, we assume that the words of text spans like noun-phrases are topically bound and we model this dependence with copulas. We demonstrate empirically the effectiveness of copulaLDA on both intrinsic and extrinsic evaluation tasks on several publicly available corpora.

1 Introduction

Probabilistic topic models, such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003), are generative models that describe the content of documents by discovering the latent topics underlying them.

A limitation inherent from the bag-of-words representation in such state-of-the-art models concerns the independence assumption: given their topics, words are assumed to occur independently. While this exchangeability assumption greatly impacts the involved computations and, in particular, the calculations of the conditional probabilities, it is rather naive and unrealistic (Heinrich, 2005). As another limitation caused by the exchangeability assumption, the grouping of words in topically coherent spans, that is contiguous text spans like sentences, is lost.

On the other hand, text structure generally contains useful information that could be leveraged in inference process. Sentences or phrases, for instance, are by definition text spans complete in themselves that convey a concise statement. To better illustrate how text structure could help in topic identification, consider the example of Figure 1. It illustrates the topics inferred by LDA for the words (excluding stop-words) of a sentence drawn from a Wikipedia page. At the sentence level, one could argue that the sentence is generated by the "Cinema" topic since it discusses a film and its authors. LDA, however, fails and assigns several topics to the words of the sentence. Importantly, several of those topics like "Elections" and "Inventions" are unrelated. In finer text granularity, LDA also fails to assign consistent topics in noun-phrases like "film noir classic" and entities like "Brian Donlevy". A binding mechanism among the topics of the words of a sentence, or a phrase, could have prevented those limitations and taking simple text structure into account would be beneficial.

Motivated by the previous example, we propose to incorporate text structure in the form of sentence or phrase boundaries as an intermediate structure in LDA. We plan to model this binding mechanism with copulas. Copulas have been found to be a flexible tool to model dependencies in the fields of

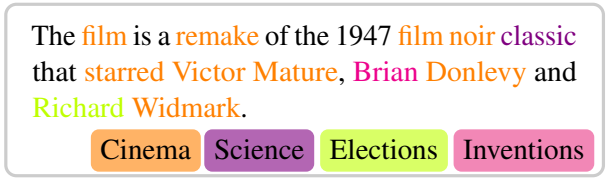


Figure 1: Applying LDA on Wikipedia documents.

* The author is also affiliated with Coffreo, Clermont Ferrand. The authors of the paper can be contacted at firstname.lastname@imag.fr

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1767–1776, Osaka, Japan, December 11-17 2016.

risk management and finance (Embrechts et al., 2002). They are a family of distribution functions that offer a flexible way to model the joint probability of random variables using only their marginals. This results in decoupling the marginal distributions by the underlying dependency. These properties make them appealing and some preliminary studies have started investigating their integration into different learning tasks (Wilson and Ghahramani, 2010; Tran et al., 2015; Amoualian et al., 2016).

The remainder of the paper is organized as follows: Section 2 presents the related work. The main contribution of this article is presented in Section 3, in which we propose to bind the latent topics that generate the words of a segment using copulas. We show that sampling word topics from copulas offers an elegant way to impose different levels and types of correlation between them. Section 4 then illustrates the behavior of *copulaLDA*, the copula-based version of LDA introduced in Section 3, while Section 5 concludes the paper.

2 Related Work

Despite the success that vector-space models (Salton et al., 1975) have enjoyed, they come with a number of limitations. We mention, for instance, their inability to model synonymy and polysemy and the sparse, high-dimensional induced representations. Many research studies have researched these problems, and Probabilistic Latent Semantic Analysis (Hofmann, 1999) was among the first attempts to model textual corpora using latent topics. In our work, we build on LDA (Blei et al., 2003), which is often used as a building block for topic models. In its context, the corpus is associated with a set of latent topics, and each document is associated with a random mixture of those topics. The words are assumed exchangeable, that is their joint probability is invariant to their permutation. Previous work proposed a variety of extensions to LDA in order to incorporate additional information such as class labels (Blei and McAuliffe, 2008) and temporal dependencies between stream documents (Wang et al., 2012). Here, our goal is to extend LDA by incorporating simple text structure in its generative and inference processes using copulas.

One may identify two lines of research to address the limitations due to the exchangeability assumption in LDA: extensions to account for the boundaries of text spans like sentences and extensions to account for the word order. With respect to the first line, (Wang et al., 2009) combine a unigram language model with topic models over sentences so that the latent topics are represented by sentences instead of terms. In (Griffiths et al., 2004), the authors investigate a combination of a topic model with a Hidden Markov Model (HMM). They assume that the HMM generates the words that handle the long-range dependencies (semantic dependencies) and the topic model the words that handle the short range dependencies (syntactic dependencies). Also, (Boyd-Graber and Blei, 2009) proposed the Syntactic Topic Model whose goal is to integrate the text semantics and the syntax in a non-parametric topic model. In another effort, (Zhu et al., 2006) propose *TagLDA*, where they replace the unigram word distributions by a factored representation that is conditioned on the topic and the part-of-speech tag of a term. Recently, (Balikas et al., 2016) introduced *senLDA*, that assumes that the terms occurring within a sentence are generated by the same topic. In our work here, we integrate part of the text structure in LDA by relying only on the boundaries of contiguous text spans like sentences, which can be obtained without deep linguistic analysis like the one required in the Syntactic Topic Model. Also, differently from *senLDA*, we do not restrict the words of the spans to be generated by the same topic. Instead, using copulas we pose correlations between those topics, which is more flexible.

The second line of research investigates how topic models can be extended to incorporate word order. In (Shafiei and Milios, 2006), the authors propose a four-level hierarchical structure where the latent topics of paragraphs are decided after performing a nested word-based LDA operation. In a similar context, (Wang et al., 2007) study how the word order in the form of n-grams can be leveraged to better capture a document's topical content. Their topical n-gram model extends LDA by determining unigram words and phrases based on context and assigning mixture of topics to both individual words and n-gram phrases.

Another interesting line of research studied the task of discovering and partitioning text in topically coherent spans. In (Du et al., 2010; Du et al., 2013) the authors rely on hierarchical Bayesian models to accomplish it. In this work, contrary to identifying such spans, we assume them to be topically coherent *a priori*, and we investigate how to leverage and incorporate this information to LDA.

Lately, there is an increasing interest over the integration of copulas in machine learning applications (Elidan, 2013) such as classification (Elidan, 2012) or structure learning (Liu et al., 2009). Interestingly, (Wilson and Ghahramani, 2010) have shown how to incorporate copulas in Gaussian processes in order to model the dependency between random variables with arbitrary marginals with a practical application on predicting the standard deviation of variables in the financial sector (volatility estimation). In another generic framework, (Tran et al., 2015) have shown the benefits of using copulas to model complex dependencies between latent variables in the general variational inference setting. The idea of using copulas with topic models was recently investigated in (Amoualian et al., 2016). In the context of document streams they proposed a topic model where the dependencies between the topic distributions of two consecutive documents are captured by copulas.

3 Integrating text structure to LDA using copulas

In this section we develop *copulaLDA* (hereafter *copLDA*), that extends LDA by integrating simple text structure in the model using copulas. We assume that the topics that generate the terms of coherent text spans are bound. A strong binding signifies high probability for the terms to have been generated by the same topic. Therefore, as we show, the conditional independence of topics given the per-document topic distributions does not hold. Before presenting the generative and inference processes of *copLDA*, we shortly discuss the idea of *coherent text spans*.

Each sentence is a coherent, meaningful segment of text and we consider them as coherent text spans in this study. However, each sentence can be further decomposed into smaller segments through syntactic analysis. Figure 2 illustrates the output of a shallow parsing step of the example sentence of Figure 1, generated using the Stanford Parser.[1] Among these different segments, noun phrases play a particular role as they are, for instance, at the basis of terminology extraction that aims at capturing concepts from a document. Noun phrases usually constitute a semantic unit, pertaining to a given concept related to few, related topics. For this reason, we also consider noun phrases as coherent text spans in this study. Another advantage of the two types of coherent text spans we consider (whole sentences and noun phrases) is that they can be easily extracted using shallow parsing techniques, and one needs not resort to complex syntactic analysis in practice.

The film is a remake of *the 1947 film noir classic* that starred *Victor Mature*, *Brian Donlevy* and *Richard Widmark*.

Figure 2: Shallow parsing using the Stanford Parser. Contiguous words in italics denote a noun-phrase.

3.1 Copulas and random variables

Copulas are interesting because they separate the dependency structure of random variables from their marginals. Formally (Nelsen, 2007; Trivedi and Zimmer, 2007), a p-dimensional copula C is a p-variate distribution function with $C : \mathbb{I}^p = [0,1]^p \rightarrow [0,1]$ whose univariate marginals are uniformly distributed on $\mathbb{I}$ and $C(u_1, \ldots, u_p) = P(U_1 \leq u_1, \ldots, U_p \leq u_p)$. Copulas allow one to explicitly relate joint and marginal distributions, through Sklar's theorem (Sklar, 1959):

Theorem 3.1 *Let F be a p-dimensional distribution function with univariate margins $F_1, \ldots, F_p$. Let A_j denote the range of F_j. Then there exists a copula C such that for all $(x_1, \ldots, x_p) \in \mathbb{R}^p$*

$$F(x_1, \ldots, x_p) = C(F_1(x_1), \ldots, F_d(x_p)) \tag{1}$$

Furthermore, when $F_1, \ldots, F_p$ are all continuous, then C is unique.

As a result any multivariate distribution F can be decomposed into its marginals $F_i, i \in \{1, \ldots, p\}$ and a copula, allowing to study the multivariate distribution independently of the marginals. Sklar's theorem also provides a way of sampling multivariate distributions with a large number of random variables using copulas: $F(x_1, \ldots, x_p) = F\left(F_1^{-1}(u_1), \ldots, F_p^{-1}(u_p)\right) = P[U_1 \leq u_1, \ldots, U_p \leq u_p] = C(u_1, \ldots, u_p)$. Hence, to sample F it suffices to sample the dependence structure modeled by copulas and then transform

[1] http://nlp.stanford.edu/software/lex-parser.shtml

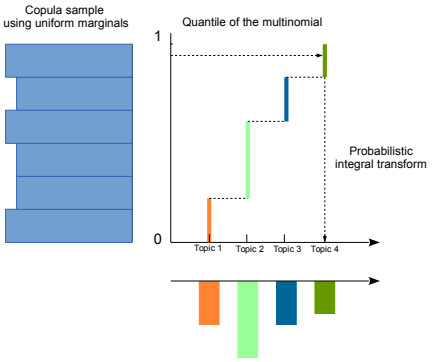


Figure 3: The transformation of a random variate to multinomial (or arbitrary) marginals. The arrows illustrate the generalized inverse; the histograms in y (resp. x) axis depict the distributions of the initial (resp. transformed) samples.

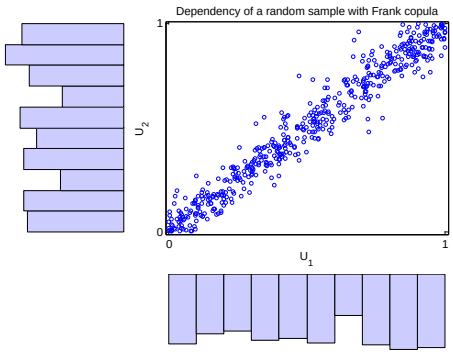


Figure 4: The positive correlation imposed to two random variates when sampling from a Frank copula with $\lambda = 25$. The histograms in x (resp. y) axis show the distributions of each of the variates that generate the scatterplot.

the obtained sample in the marginals of interest using the probabilistic integral transform. We illustrate this transformation for one variable in Figure 3. Sampling the copula returns, for each variate, a sample as the one indicated in the histogram of the y axis. One can then transform the sample using the quantile (F^{-1}) of an arbitrary marginal.

Before proceeding further, we visit some extreme conditions of dependence illustrating the respective copulas that model them: (1) *Independence*, which is a frequently assumed simplification in topic models and is obtained with $\prod_{i=1}^{p} u_i$, and (2) *Co-monotonicity*, which is the complete, positive correlation between the random variables u_p, obtained with $\min(u_1, \ldots, u_p)$.

In the rest of our development we will be using a particular family of copulas, the Archimedean copulas. Archimedean copulas are widely used copulas and are defined with respect to a generator function ψ. They take the form: $C(u_1, \cdots, u_d) = \psi^{-1}(\psi(u_1) + \cdots + \psi(u_d))$. A special case of Archimedean copulas corresponds to Frank copulas, which are obtained by setting: $\psi_\lambda(u) = \frac{-1}{\lambda}\log(1-(1-e^{-\lambda})e^{-u})$. When $\lambda \to 0$, the Frank copula approaches the independency copula; when $\lambda \to \infty$ it approaches the co-monotonicity copula. Hence, the Frank copula allows one to model all dependencies between complete independence to perfect dependence while varying λ from 0 to ∞. Therefore, λ can be seen as an additional hyper-parameter to be tuned or learned from the data. Figure 4 illustrates the positive dependence between two random variables sampled from a Frank copula with $\lambda = 25$. To sample from the Archimedean copulas, we rely on the algorithm proposed by (Marshall and Olkin, 1988), which was further improved in (McNeil, 2008; Hofert, 2011) and implemented in the R language (Hofert et al., 2011).

3.2 Extending LDA with copulas

As mentioned above, copulas provide a nice way to bind random variables. We are making use of them here to bind word-specific topics (the z variables in LDA) within coherent text spans, the rationale being that coherent text spans can not be generated by many different, uncorrelated topics. This leads us to the following generative model:

- For each topic $k \in [1, K]$, choose a per-word distribution: $\phi_k \sim Dir(\beta)$, with $\phi_k, \beta \in \mathbb{R}^{|V|}$
- For each document d_i, $i \in \{1, \ldots, D\}$:
  - Choose a per-document topic distribution: $\theta_i \sim Dir(\alpha)$, with $\theta_i, \alpha \in \mathbb{R}^{|K|}$
  - Sample number of segments in d_i: $S_i \sim Poisson(\xi)$;
  - For each segment $s_{i,j}, j \in \{1, \ldots, S_i\}$:

* Sample number of words: $N_{i,j} \sim Poisson(\xi_d)$;
* Sample topics $\mathcal{Z}_{i,j} = (z_{i,j,1}, \ldots, z_{i,j,N_{i,j}})$ from a distribution admitting $Mult(1, \theta_i)$ as margins and C as copula;
* Sample words $W_{i,j} = (w_{i,j,1}, \ldots, w_{i,j,N_{i,j}})$: $w_{i,j,n} \sim Mult(1, \phi_{z_{i,j,n}})$, $1 \leq n \leq N_{i,j}$.

There are two main differences between *copLDA* and LDA. Firstly, the former assumes a hierarchical structure in the documents: the topics that generate the words in the coherent segments exhibit topical correlation, hence the conditional independence assumption between the terms of a segment given the document per-topic distribution (θ_i) no longer holds. Secondly, this topical correlation is modeled using copulas. Figure 5 provides the graphical model for *copLDA*. For clarity, we draw each word in a coherent segment S $(w1, \ldots, w_N)$ to make the dependencies explicit. Notice how the topics of those words depend on both the copula parameter λ and the per-document topic distribution θ.

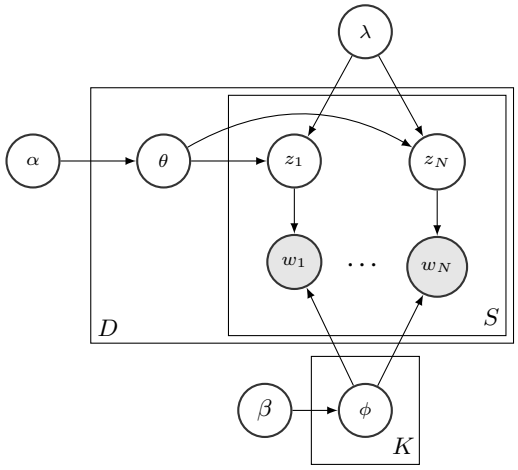


Figure 5: The *copLDA* generative model. We model the dependency between the topics underlying a segment with copulas.

The hyper-parameters α and β correspond to priors of the model. Following (Blei et al., 2003), we assume them here to be symmetric and we fix them to $\frac{1}{K}$, with K the number of topics retained. The hyper-parameter λ is chosen after exploration of a grid of possible values, and is the same for the whole corpus. We choose the value that minimizes perplexity.

3.3 Inference with Gibbs sampling

The parameters of the above model, that are ϕ, θ and the topics of each segment $\mathcal{Z}_{i,j} = (z_{i,j,1}, \cdots, z_{i,j,N_{i,j}})$, can be directly estimated through Gibbs sampling. Denoting Ω and Ψ the count matrices such that $\Omega = (\Omega_{i,k})$ (resp. $\Psi = (\Psi_{k,v})$) represents the count of word belonging to topic k assigned to document d_i (resp. the count of word v being assigned to topic k), the Gibbs updates for θ and ϕ are the same as the ones for the standard LDA model (Blei et al., 2003):

$$\theta_i \sim Dir(\alpha + \Omega_i) \quad \text{and} \quad \phi_k \sim Dir(\beta + \Psi_k) \tag{2}$$

The update for the variables z is obtained as follows:

$$p(\mathcal{Z}_{i,j}|\mathcal{Z}_{-i,j}, W, \Theta, \Phi, \alpha, \beta, \lambda) = \frac{p(\mathcal{Z}_{i,j}, \mathcal{Z}_{-i,j}, W|\Theta, \Phi, \alpha, \beta, \lambda)}{p(\mathcal{Z}_{-i,j}, W|\Theta, \phi, \alpha, \beta, \lambda)} =$$
$$\frac{p(\mathcal{Z}_{i,j}, W_{i,j}|\Theta, \Phi, \lambda)p(\mathcal{Z}_{-i,j}, W_{-i,j}|\Theta, \Phi, \lambda)}{p(W_{i,j}|\Theta, \phi)p(\mathcal{Z}_{-i,j}, W_{-i,j}|\Theta, \Phi, \lambda)} = \frac{p(\mathcal{Z}_{i,j}, W_{i,j}|\Theta, \Phi, \lambda)}{\sum_{\mathcal{Z}_{i,j}} p(\mathcal{Z}_{i,j}, W_{i,j}|\Theta, \Phi, \lambda)} =$$
$$\frac{p(W_{i,j}|\mathcal{Z}_{i,j}, \Phi)p(\mathcal{Z}_{i,j}|\Theta, \lambda)}{\sum_{\mathcal{Z}_{i,j}} p(W_{i,j}|\mathcal{Z}_{i,j}, \Phi)p(\mathcal{Z}_{i,j}|\Theta, \lambda)} \sim p(W_{i,j}|\mathcal{Z}_{i,j}, \Phi)p(\mathcal{Z}_{i,j}|\Theta, \lambda) = p(\mathcal{Z}_{i,j}|\Theta, \lambda) \prod_{n=1}^{N_{i,j}} \phi_{w_{i,j,n}, z_{i,j,n}} \tag{3}$$

where W, Θ and Φ stand for the whole parameter set of w, θ and ϕ and the probability outside the product in the last step admits a copula C_λ and $Mult(1, \theta_i)$ as margins. As is standard in topic models, the notation $-i, j$ means excluding the information for i, j. Note that in case where $\lambda \to 0$, the words of a segment become conditionally independent given the per-document distribution and one recovers the non collapsed Gibbs sampling updates of LDA.

From the expression of Eq. (3), a simple acceptance/rejection algorithm can be formulated: (1) Sample a random variable of pdf $p(\mathcal{Z}_{i,j}|\Theta, \lambda)$ using copula, and, (2) Accept the sample with probability $p(W_{i,j}|\mathcal{Z}_{i,j}, \Phi) = \prod_{n=1}^{N_{i,j}} \phi_{w_{i,j,n}, z_{i,j,n}}$. Algorithm 1 summarizes the inference process.

3.4 Computational Considerations

As the values of $\phi_{w_{i,j,1},z_{i,j,1}} \times \cdots \times \phi_{w_{i,j,n},z_{i,j,n}}$ tend to be very low, the acceptance/rejection sampling step described above is very slow in practice (see below). We propose here to speed it up by considering, for each word $w_{i,j,n}$ in a given segment, not the exact probability of $z_{i,j,n}$, but its mean (noted M) over all the other words in the segment:

$$M(z_{i,j,n}|\mathcal{Z}_{-i,j}, W, \Theta, \Phi, \alpha, \beta, \lambda) = \sum_{w_{ij,l}, l \neq n} \sum_{z_{ij,l}, l \neq n} P(\mathcal{Z}_{i,j}|\mathcal{Z}_{-i,j}, W, \Theta, \Phi, \alpha, \beta, \lambda) \propto \phi_{w_{i,j,n}} \theta_{d,z_{i,j,n}}$$

as $\sum_{w_{ij,l}} \phi_{w_{i,j,l}} = 1$. Note that the above form is a marginalization of $P(\mathcal{Z}_{i,j}|\mathcal{Z}_{-i,j}, W, \Theta, \Phi, \alpha, \beta, \lambda)$ and thus defines a valid probability and a valid Gibbs sampler, even though on a joint distribution that slightly differs from the original one.

Figure 6 compares the perplexity scores achieved in 200 documents from the Wikipedia dataset "Wiki46" of Table 1 by the copLDA model, when considering noun-phrases as coherent spans, with and without rejection sampling. We repeat the experiment 10 times and also plot the standard deviation. We first note that approximating Algorithm 1 by ignoring the rejection sampling step results in slightly worse performance. On the other hand, without the rejection sampling, copLDA converges faster in terms of iterations. Furthermore, the cost in terms of running time of a single iteration is significantly smaller: for instance, for 30 iterations with rejection sampling, the algorithm needs almost 6 hours, that is 100 times more than the 3.5 minutes needed without the rejection sampling. Hence, in the rest of the study, for scaling purposes, we adopt the above mean approximation.

Algorithm 1: A Gibbs Sampling iteration for *copLDA*

```
Input: documents' words grouped in segments, α, β, K, Copula
family and its parameter λ
//Initialize counters Ψ, Ω
for document d_i, i ∈ [1, D] do
    for segment s_{i,j} : j ∈ {1, ..., S_i} do
        Draw a random vector U = (U_1, ..., U_{N_{i,j}}) that
        admits a copula C_λ
        do /* If the mean approximation is used, the loop is
           done once, ignoring the acceptance condition */
            for words w_{i,j,k}, k ∈ [1, W_{N_{i,j}}] in s_{i,j} do
                Decrease counter variables Ψ, Ω
                Get z_{i,j,k} by transforming U_k to Mult.
                marginals with the generalized inverse
                Assign topic z_{i,j,k} to w_{i,j,k}
                Increase counters Ψ, Ω
            end
        while Accept the new segment topic assignments with
        probability φ_{w_{i,j,1},z_{i,j,1}} × ··· × φ_{w_{i,j,n},z_{i,j,n}}
    end
end
```

4 Experimental study

Models In our experiments, we compare the following topic models: (1) *copLDA$_{sen}$* that considers sentences as coherent segments, (2) *copLDA$_{np}$* that considers noun-phrases as coherent segments, (3) LDA as proposed in (Blei et al., 2003) using the collapsed Gibbs sampling inference of (Griffiths and Steyvers, 2004), and (4) *senLDA* described in (Balikas et al., 2016) using its public implementation. For *copLDA$_x$* models, we use the Frank copula which was reported to obtain the best performance in similar tasks (Amoualian et al., 2016) and was also found to achieve the best performance in our local validation settings compared to Gumbel and Clayton copulas. We have implemented the models using Python;[2] for sampling the Frank copulas we used the R `copula` package (Hofert et al., 2011) and `rPY`.[3] As mentioned in Section 3.2, λ is set to 2 for *copLDA$_{sen}$* and to 5 for *copLDA$_{np}$* (values which we found to perform well in every dataset we tried). Furthermore, the hyper-parameters α and β where set to $1/K$, where K is the number of topics, which was selected from $\{50, 100, 200, 300, 400\}$ for each dataset. For the shallow parsing step, required for *copLDA$_{np}$*, we used the Stanford Parser (Klein and Manning, 2003). The text pre-processing steps performed are: lower-casing, stemming using the Snowball Stemmer and removal of numeric strings.

[2]The models used in this paper are available for research purposes at `https://github.com/balikasg/topicModelling`.

[3]https://pypi.python.org/pypi/rpy2

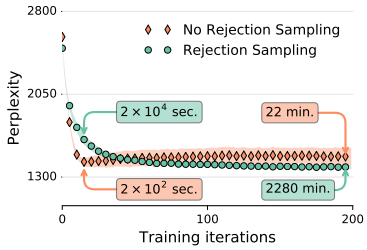


Figure 6: The effect of rejection sampling in efficiency and perplexity performance.

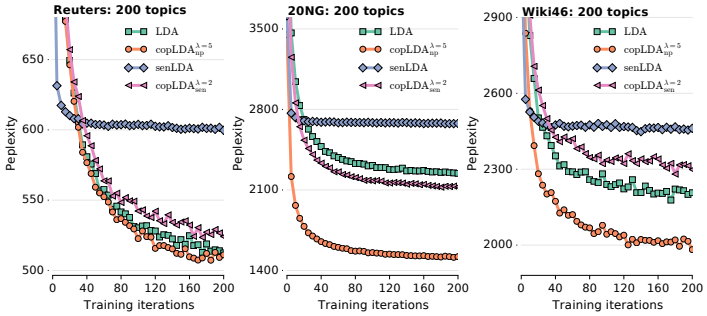


Figure 7: The perplexity curves of the investigated models for 200 Gibbs sampling iterations and different datasets.

	Basic Statistics				Perplexity Scores				Classification (MiF_1) scores			
	Docs.	$\|N\|$	$\|V\|$	Classes	senLDA	copLDA$_{sen}$	LDA	copLDA$_{np}$	senLDA	copLDA$_{sen}$	LDA	copLDA$_{np}$
20NG	19,056	1.7M	75.4K	20	2636	2083	2200	**1483**	0.5622	0.6328	0.6246	**0.6490**
TED	1,096	1.16M	30.4K	15	2099	1812	1805	**1775**	0.4612	0.4678	0.4633	**0.4764**
PubMed	5498	1.09M	28.7K	50	1601	1385	1384	**1085**	0.6666	**0.7525**	0.7406	0.7431
Reuters	10,788	875K	21.4K	90	579	512	501	**499**	0.7504	0.7692	**0.7893**	0.7851
Wiki15	1,198	162K	13.4K	15	2988	2766	2640	**2397**	0.6920	0.7230	0.74	**0.7403**
Wiki37	2,459	317K	19.7K	37	3103	2871	2711	**2395**	0.5717	0.6053	**0.6447**	0.6220
Wiki46	3,657	478K	23.4K	46	2220	2280	2135	**1978**	0.5326	0.6170	**0.6599**	0.6326
Austen	5,262	170K	6.3K	-	1110	898	**798**	805	-	-	-	-

Table 1: The basic statistics, the perplexity and the classification scores of the datasets used.

Datasets We have used the following publicly available data collections to test the performance of the topic models: (1) 20NG (20 news groups), which is a standard text dataset for such tasks as provided by (Bird et al., 2009), (2) Reuters (Reuters-21578, the "ModApte" version), also discussed in (Bird et al., 2009), (3) TED, that is transcriptions of TED talks released in the framework of the International Workshop on Spoken Language Translation 2013 evaluation campaign[4] (we have merged the train, development and test parts and we selected the transcriptions with at least one associated label among the 15 most common in the data[5]), (4) Wiki$_x$, with $x \in \{15, 37, 46\}$ and PubMed, both excerpts[6] from the Wikipedia dataset of (Partalas et al., 2015) and the PubMed dataset of (Tsatsaronis et al., 2015) used in (Balikas et al., 2016), and (5) "Austen", where we concatenated three books[7] written by Jane Austen, available from the Gutenberg project (each paragraph is considered as a document). Table 1 presents some basic statistics for these datasets.

Manual inspection of the topics We begin by comparing LDA and *copLDA*$_{\text{np}}$. For presentation purposes, we train the two topic models using the Wiki$_{47}$ dataset with 10 topics and we illustrate the top-10 words learned for each topic by the two models in Table 2. As one can note, since the two models have been trained on the same data with the same training parameters, the identified topics are very similar. This said, *copLDA*$_{\text{np}}$ manages to produce arguably better topics. This is for example the case for the topic "Birth"; although both models assign high probability to words like "born" and "american" due to the content of the dataset, copLDA$_{\text{np}}$ manages to identify several words corresponding to months which makes the topic more thematically consistent and easier to interpret compared to its LDA counterpart. In the same line, Table 3 visualizes the inferred topics for parts of the Wiki$_{47}$ dataset. Notice here that given the topic interpretations of Table 2, both models manage to identify intuitive topics. Note however how in most of the cases the text structure information used by *copLDA*$_{\text{np}}$ helps to obtain consistent topics to generate noun-phrases like "crime thriller film" and "raspy voice", a consistency that LDA is lacking.

Intrinsic evaluation: perplexity We present in Table 1 the perplexity scores achieved by the 4 models in

[4] http://workshop2013.iwslt.org/59.php

[5] Technology, Culture, Science, Global Issues, Design, Business, Entertainment, Arts, Politics, Education, Art, Creativity, Health, Biology and Music.

[6] https://github.com/balikasg/topicModelling/tree/master/data

[7] We used the books: Emma, Persuasion, Sense. We considered each paragraph as a document.

Profession	Science	Books	Art	Cinema	Places	Music	Birth	Elections	Inventions
profession	univers	book	art	film	state	record	born	elect	california
world	research	new	new	televis	unit	music	american	canadian	plant
footbal	scienc	work	work	role	us	band	known	parti	use
wrestl	professor	american	paint	appear	township	album	best	member	invent
play	work	publish	york	also	school	song	actress	liber	flower
born	institut	time	american	actor	univers	also	decemb	minist	compani
american	award	author	artist	born	serv	produc	june	hous	north
championship	prize	also	museum	play	war	releas	april	canada	patent
team	born	year	painter	seri	nation	new	juli	serv	inventor
first	receiv	york	studi	star	build	singer	januari	conserv	found
known	univers	book	art	film	township	record	play	elect	work
wrestl	research	new	new	born	state	music	footbal	canadian	first
born	scienc	american	york	televis	counti	band	born	serv	year
world	professor	author	paint	role	us	album	american	parti	photograph
profession	work	publish	american	actor	california	song	tour	member	design
american	institut	novel	work	appear	michigan	also	golf	liber	state
name	born	time	artist	also	plant	singer	year	hous	new
wrestler	prize	also	painter	seri	civil	releas	profession	minist	use
best	studi	writer	museum	actress	popul	produc	first	state	also
championship	award	magazin	born	american	flower	american	season	born	build

Table 2: The top-10 words of copLDA (upper half) and LDA (lower half) in the Wiki46 dataset.

Kiss of Death is *a 1995 crime thriller film* starring *David Caruso Samuel L. Jackson* and *Nicolas Cage*. *The film* is a *very loosely* based remake of *the 1947 film noir classic* of *the same name* that starred *Victor Mature*, *Brian Donlevy* and *Richard Widmark*.

Bertram Stern (born 3 *October 1929*) is *an American fashion and celebrity portrait photographer*.

Dana Hill (born *Dana Lynne Goetz* in *Los Angeles*, California; *May 6, 1964 - July 15, 1996*) was *an American actress and voice actor* with *a raspy voice and childlike appearance*, which allowed her to play *adolescent roles* well into *her 20s*.

Kiss of Death is *a 1995 crime thriller film* starring *David Caruso Samuel L. Jackson* and *Nicolas Cage*. *The film* is a *very loosely* based remake of *the 1947 film noir classic* of *the same name* that starred *Victor Mature*, *Brian Donlevy* and *Richard Widmark*.

Bertram Stern (born 3 *October 1929*) is *an American fashion and celebrity portrait photographer*.

Dana Hill (born *Dana Lynne Goetz* in *Los Angeles*, California; *May 6, 1964 - July 15, 1996*) was *an American actress and voice actor* with *a raspy voice and childlike appearance*, which allowed her to play *adolescent roles* well into *her 20s*.

Table 3: The discovered topics underlying the words of example documents for LDA (left) and copLDA (right). The parts of the documents in italics indicate the noun-phrases obtained by the Stanford Parser. The text colours refer to the topics described in Table 2.

each of the datasets we examined. We split each dataset in two parts with 80%/20% of the documents: we use the former for learning the model and the second for calculating the perplexity scores. First note that *copLDA*$_{\text{np}}$ achieves the lowest scores in most of the datasets. LDA is the second best performing model, whereas the third one is *copLDA*$_{\text{sen}}$. We believe that the difference between *copLDA*$_{\text{sen}}$ and *copLDA*$_{\text{np}}$ stems from the fact that perplexity is an evaluation measure that is calculated on the basis of words. Hence, considering sentences as coherent spans whose topics are bound results in less flexibility and this is reflected in higher perplexity scores. However, using copulas results in more flexibility than assigning the same topic in each term of the sentence which is illustrated in the performance difference between *copLDA*$_{\text{sen}}$ and *senLDA*. The former being more flexible, due to the copulas, performs better. In the same line, Figure 7 illustrates the perplexity curves of the hold-out documents for the four models on three of the datasets of Table 1 for 200 Gibbs sampling iterations. Note that *senLDA* is the model with the fastest convergence rate with respect to the number of Gibbs iterations. On the other hand, LDA, *copLDA*$_{\text{sen}}$ and *copLDA*$_{\text{np}}$ require the same number of iterations, which depends on the dataset. *copLDA*$_{\text{np}}$ manages to achieve the lowest perplexity scores: notice its steep curves in the first iterations.

Extrinsic evaluation: text classification To further highlight the merits of *copLDA*, we also present in Table 1 the classification results for the datasets used. The reported scores are the averages of 10-fold cross-validation. We use the per-document topic distributions as classification features fed to Support Vectors Machines (SVMs). We have used the implementation of (Pedregosa et al., 2011) with $C = 1$ for the SVM regularization parameter. For the multi-label datasets (TED and PubMed) we employed one-versus-rest: the SVMs return every category with a positive distance from the separating hyper-planes. As one can note, *copLDA*$_{\text{np}}$ and LDA achieve the highest MiF scores in most of the datasets, without a clear advantage to one vs the other. Binding the topics of sentence words with copulas improves over the

results of *senLDA*: *copLDA*$_{\text{sen}}$ performs only slightly worse than LDA and *copLDA*$_{\text{np}}$ on most datasets and outperforms them, only slightly again, on one dataset.

5 Conclusions

We proposed *copLDA* that extends LDA to incorporate the topical dependencies within sentences and noun-phrases using copulas. We have shown empirically the advantages of considering text structure and incorporating it in LDA with copulas. In our future work we plan to integrate procedures to learn the λ parameter of Frank copulas and to investigate ways to model not only dependencies within text segments like noun-phrases, but also dependencies between such segments with nested copulas.

Acknowledgements

The authors would like to thank the anonymous reviewers for their useful comments. This work is partially supported by the CIFRE N 28/2015.

References

H. Amoualian, M. Clausel, E. Gaussier, and M.R. Amini. 2016. Streaming-LDA: A Copula-based Approach to Modeling Topic Dependencies in Document Streams. In *Proceedings of the 22th International ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM.

G. Balikas, M.R. Amini, and M. Clausel. 2016. On a Topic Model for Sentences. *Journal of CoRR*, abs/1606.00253.

S. Bird, E. Klein, and E. Loper. 2009. *Natural Language Processing with Python*. " O'Reilly Media, Inc.".

D.M. Blei and J.D. McAuliffe. 2008. Supervised Topic Models. In *Advances in Neural Information Processing Systems 20 NIPS*, pages 121–128. Curran Associates, Inc.

D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning*, 3:993–1022, March.

J. Boyd-Graber and D.M. Blei. 2009. Syntactic Topic Models. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21 NIPS*, pages 185–192. Curran Associates, Inc.

L. Du, W. Buntine, and H. Jin. 2010. A Segmented Topic Model Based on the Two-parameter Poisson-Dirichlet Process. *Journal of Machine learning*, 81(1):5–19.

L. Du, W.L. Buntine, and M. Johnson. 2013. Topic Segmentation with a Structured Topic Model. In *Proceedings of HLT-NAACL*, pages 190–200.

G. Elidan. 2012. Copula Network Classifiers (CNCs). In *Proceedings of International Conference on Artificial Intelligence and Statistics*, pages 346–354.

G. Elidan. 2013. Copulas in Machine Learning. In *Advances in Copulae in mathematical and quantitative finance*, pages 39–60. Springer.

P. Embrechts, A. McNeil, and D. Straumann. 2002. Correlation and Dependence in Risk Management: Properties and Pitfalls. *Journal of Risk management: value at risk and beyond*, pages 176–223.

T.L. Griffiths and M. Steyvers. 2004. Finding Scientific Topics. *Journal of the National Academy of Sciences*, 101(suppl 1):5228–5235.

T.L. Griffiths, D.M. Steyvers, M.and Blei, and J.B. Tenenbaum. 2004. Integrating Topics and Syntax. In *Proceedings of Neural Information Processing Systems 17 NIPS*, volume 4, pages 537–544.

G. Heinrich. 2005. Parameter Estimation for Text Analysis. Technical report, Technical report.

M. Hofert, M. Mächler, et al. 2011. Nested Archimedean Copulas Meet R: The nacopula Package. *Journal of Statistical Software*, 39(9):1–20.

M. Hofert. 2011. Efficiently Sampling Nested Archimedean Copulas. *Journal of Computational Statistics & Data Analysis*, 55(1):57–70.

T. Hofmann. 1999. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, New York, NY, USA. ACM.

D. Klein and C.D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

H. Liu, J. Lafferty, and L. Wasserman. 2009. The Nonparanormal: Semiparametric Estimation of High Dimensional UndirectedGraphs. *Journal of Machine Learning Research*, 10:2295–2328.

A.W. Marshall and I. Olkin. 1988. Families of Multivariate Distributions. *Journal of the American Statistical Association*, 83(403):834–841.

A.J. McNeil. 2008. Sampling Nested Archimedean Copulas. *Journal of Statistical Computation and Simulation*, 78(6):567–581.

R.B. Nelsen. 2007. *An Introduction to Copulas*. Springer Science & Business Media.

I. Partalas, A. Kosmopoulos, N. Baskiotis, T. Artieres, G. Paliouras, E. Gaussier, I.Androutsopoulos, M.R. Amini, and P. Galinari. 2015. LSHTC: A Benchmark for Large-Scale Text Classification. *Journal of CoRR*, abs/1503.08581, march.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. 2011. Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

G. Salton, A. Wong, and C. Yang. 1975. A Vector Space Model for Automatic Indexing. *Journal of Communications of the ACM*, 18(11):613–620.

M. Shafiei and E. Milios. 2006. Latent Dirichlet Co-clustering. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 542–551, Washington, DC, USA. IEEE Computer Society.

M. Sklar. 1959. *Fonctions de Répartition à n Dimensions et Leurs Marges*. Université Paris 8.

D. Tran, D.M. Blei, and E.M. Airoldi. 2015. Copula Variational Inference. In *Proceedings of Neural Information Processing Systems 28 NIPS*, pages 3564–3572.

P.K. Trivedi and D.M. Zimmer. 2007. *Copula Modeling: An Introduction for Practitioners*. Now Publishers Inc.

G. Tsatsaronis, G. Balikas, P. Malakasiotis, I. Partalas, M. Zschunke, M.R. Alvers, D. Weissenborn, A. Krithara, S. Petridis, D. Polychronopoulos, et al. 2015. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *Journal of BMC bioinformatics*, 16(1):1.

Xuerui Wang, Andrew McCallum, and Xing Wei. 2007. Topical N-grams: Phrase and Topic Discovery, with an Application to Information Retrieval. In *Proceedings of ICDM 2007. Seventh IEEE International Conference on*, pages 697–702. IEEE.

D. Wang, S. Zhu, T. Li, and Y. Gong. 2009. Multi-Document Summarization using Sentence-based Topic Models. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 297–300. Association for Computational Linguistics.

Y. Wang, E. Agichtein, and M. Benzi. 2012. TM-LDA: Efficient Online Modeling of Latent Topic Transitions in Social Media. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 123–131, New York, NY, USA. ACM.

A.G. Wilson and Z. Ghahramani. 2010. Copula Processes. In *Advances in Neural Information Processing Systems 23 NIPS*, pages 2460–2468. Curran Associates, Inc.

X. Zhu, D.M. Blei, and J. Lafferty. 2006. Taglda: Bringing Document Structure Knowledge into Topic Models. Technical report, Technical Report TR-1553, University of Wisconsin.

Consensus Attention-based Neural Networks for Chinese Reading Comprehension

Yiming Cui[†*], **Ting Liu**[‡], **Zhipeng Chen**[†], **Shijin Wang**[†] and **Guoping Hu**[†]
[†]iFLYTEK Research, Beijing, China
[‡]Research Center for Social Computing and Information Retrieval,
Harbin Institute of Technology, Harbin, China
[†]{ymcui,zpchen,sjwang3,gphu}@iflytek.com
[‡]tliu@ir.hit.edu.cn

Abstract

Reading comprehension has embraced a booming in recent NLP research. Several institutes have released the Cloze-style reading comprehension data, and these have greatly accelerated the research of machine comprehension. In this work, we firstly present Chinese reading comprehension datasets, which consist of People Daily news dataset and Children's Fairy Tale (CFT) dataset. Also, we propose a consensus attention-based neural network architecture to tackle the Cloze-style reading comprehension problem, which aims to induce a consensus attention over every words in the query. Experimental results show that the proposed neural network significantly outperforms the state-of-the-art baselines in several public datasets. Furthermore, we setup a baseline for Chinese reading comprehension task, and hopefully this would speed up the process for future research.

1 Introduction

The ultimate goal of machine intelligence is to read and comprehend human languages. Among various machine comprehension tasks, in recent research, the Cloze-style reading comprehension task has attracted lots of researchers. The Cloze-style reading comprehension problem (Taylor, 1953) aims to comprehend the given context or document, and then answer the questions based on the nature of the document, while the answer is a single word in the document. Thus, the Cloze-style reading comprehension can be described as a triple:

$$\langle \mathcal{D}, \mathcal{Q}, \mathcal{A} \rangle$$

where $\mathcal{D}$ is the document, $\mathcal{Q}$ is the query and $\mathcal{A}$ is the answer to the query.

By adopting attention-based neural network approaches (Bahdanau et al., 2014), the machine is able to learn the relationships between document, query and answer. But, as is known to all, the neural network based approaches need large-scale training data to train a reliable model for predictions. Hermann et al. (2015) published the CNN/Daily Mail news corpus for Cloze-style reading comprehensions, where the content is formed by the news articles and its summarization. Also, Hill et al. (2015) released the Children's Book Test (CBT) corpus for further research, where the training samples are generated through automatic approaches. As we can see that, automatically generating large-scale training data for neural network training is essential for reading comprehension. Furthermore, more difficult problems, such as reasoning or summarization of context, need much more data to learn the higher-level interactions.

Though we have seen many improvements on these public datasets, some researchers suggested that these dataset requires less high-level inference than expected (Chen et al., 2016). Furthermore, the public datasets are all automatically generated, which indicate that the pattern in training and testing phase are nearly the same, and this will be easier for the machine to learn these patterns.

In this paper, we will release Chinese reading comprehension datasets, including People Daily news datasets and Children's Fairy Tale datasets. As a highlight in our datasets, there is a human evaluated

[*]This work was done by the Joint Laboratory of HIT and iFLYTEK (HFL).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1777–1786, Osaka, Japan, December 11-17 2016.

dataset for testing purpose. And this will be harder for the machine to answer these questions than the automatically generated questions, because the human evaluated dataset is further processed, and may not be accordance with the pattern of automatic questions. More detailed analysis will be given in the following sections. The main contributions of this paper are as follows:

- To our knowledge, this is the first released Chinese reading comprehension datasets and human evaluated test sets, which will benefit the research communities in reading comprehension.
- Also, we propose a refined neural network that aims to utilize full representations of query to deal with the Cloze-style reading comprehension task, and our model outperform various state-of-the-art baseline systems in public datasets.

The rest of the paper will be organized as follows. In Section 2, we will briefly introduce the existing Cloze-style datasets, and describe our Chinese reading comprehension datasets in detail. In Section 3, we will show our refined neural network architecture for Cloze-style reading comprehension. The experimental results on public datasets as well as our Chinese reading comprehension datasets will be given in Section 4. Related work will be described in Section 5, and we make a brief conclusion of our work at the end of this paper.

2 Chinese Reading Comprehension Datasets

We first begin with a brief introduction of the existing Cloze-style reading comprehension datasets, and then introduce our Chinese reading comprehension datasets: People Daily and Children's Fairy Tale.

2.1 Existing Cloze-style Datasets

Typically, there are two main genres of the Cloze-style datasets publicly available, which all stem from the English reading materials.

CNN/Daily Mail.[1] The news articles often come with a short summary of the whole report. In the spirit of this, Hermann et al. (2015) constructed a large dataset with web-crawled CNN and Daily Mail news data. Firstly, they regard the main body of the news article as the *Document*, and the *Query* is formed through the summary of the article, where one entity word is replaced by a placeholder to indicate the missing word. And finally, the replaced entity word will be the *Answer* of the *Query*. Also, they have proposed the *anonymize* the named entity tokens in the data, and re-shuffle the entity tokens for every sample in order to exploit general relationships between anonymized named entities, rather than the common knowledge. But as Chen et al. (2016)'s studies on these datasets showed that the anonymization is less useful than expected.

Children's Book Test. [2] There was also a dataset called the Children's Book Test (CBT) released by Hill et al. (2015), which is built from the children's book story. Different from the previously published CNN/Daily Mail datasets, they formed the *Document* with 20 consecutive sentences in the book, and regard the 21st sentence as the *Query*, where one word is blanked with a placeholder. The missing word are chosen from named entities (NE), common nouns (CN), verbs and prepositions. As the verbs and prepositions are less dependent with the document, most of the studies are focusing on the NE and CN datasets.

2.2 People Daily and Children's Fairy Tale Datasets

In this part, we will introduce our Chinese reading comprehension datasets in detail[3]. Though many solid works on previously described public datasets, there is no studies on Chinese reading comprehension datasets. What makes our datasets different from previous works are listed as below.

- As far as we know, the proposed dataset is the first Chinese Cloze-style reading comprehension datasets, which will add language diversity in the community.

[1]The pre-processed CNN and Daily Mail datasets are available at `http://cs.nyu.edu/~kcho/DMQA/`

[2]The CBT datasets are available at `http://www.thespermwhale.com/jaseweston/babi/CBTest.tgz`

[3]Our datasets are available at `http://hfl.iflytek.com/chinese-rc/`.

Document	1 ||| 人民网 1月 1日 讯 据 《 纽约 时报 》 报道， 美国 华尔街 股市 在 2013年的 最后 一 天 继续 上涨， 和 全球 股市 一样， 都 以 最 高 纪录 或 接近 最高 纪录 结束 本年的 交易 。 2 ||| 《 纽约 时报 》 报道 说， 标普 500 指数 今年 上升 29.6%， 为 1997年以来 的 最 大 涨幅 ； 3 ||| 道琼斯 工业 平均 指数 上升 26.5%， 为 1996年 以来 的 最 大 涨幅 ； 4 ||| 纳斯达克 上涨 38.3%。 5 ||| 就 12月 31日 来说， 由于 就业 前景 看好 和 经济 增长 明年 可 能 加速， 消费者 信心 上升 。 6 ||| 工商 协进会 报告， 12月 消费者 信心 上升 到 78.1， 明显 高 于 11月 的 72 。 7 ||| 另 据 《 华尔街 日报 》 报道， 2013年是 1995年 以来 美国 股市 表现 最 好 的 一 年 。 8 ||| 这 一 年 里， 投资 美国 股市 的 明智 做法 是 追 着 “ 傻钱 ” 跑 。 9 ||| 所谓 的 “ 傻钱 ” **XXXXX**， 其实 就 是 买 入 并 持有 美国 股票 这样 的 普通 组合 。 10 ||| 这个 策略 要 比 对冲 基金 和 其它 专业 投资者 使用 的 更为 复 杂 的 投资 方法 效果 好 得 多 。	1 ||| People Daily (Jan 1). According to report of “New York Times”, the Wall Street stock market continued to rise as the global stock market in the last day of 2013, ending with the highest record or near record of this year. 2 ||| “New York times” reported that the S&P 500 index rose 29.6% this year, which is the largest increase since 1997. 3 ||| Dow Jones industrial average index rose 26.5%, which is the largest increase since 1996. 4 ||| NASDAQ rose 38.3%. 5 ||| In terms of December 31, due to the prospects in employment and possible acceleration of economy next year, there is a rising confidence in consumers. 6 ||| As reported by Business Association report, consumer confidence rose to 78.1 in December, significantly higher than 72 in November. 7 ||| Also as “Wall Street journal” reported that 2013 is the best U.S. stock market since 1995. 8 ||| In this year, to chase the “silly money” is the most wise way to invest in U.S. stock. 9 ||| The so-called “silly money” is that, to buy and hold the common combination of U.S. stock. 10 ||| This strategy is better than other complex investment methods, such as hedge funds and the methods adopted by other professional investors.
Query	所谓 的 “ 傻钱 ” **XXXXX**， 其实 就 是 买 入 并 持有 美国 股票 这样 的 普通 组合 。	The so-called “silly money” **XXXXX** is that, to buy and hold the common combination of U.S. stock.
Answer	策略	strategy

Figure 1: Example training sample in People Daily datasets (the English translation is given in the right box). The "**XXXXX**" represents the missing word. In this example, the document consists of 10 sentences, and the 9th sentence is chosen as the query.

- We provide a large-scale Chinese reading comprehension data in news domain, as well as its validation and test data as the in-domain test.
- Further, we release two out-of-domain test sets, and it deserves to highlight that one of the test sets is made by the humans, which makes it harder to answer than the automatically generated test set.

People Daily. We roughly collected 60K news articles from the People Daily website[4]. Following Liu et al. (2016), we process the news articles into the triple form $\langle \mathcal{D}, \mathcal{Q}, \mathcal{A} \rangle$. The detailed procedures are as follows.

- Given a certain document $\mathcal{D}$, which is composed by a set of sentences $\mathcal{D} = \{s_1, s_2, ..., s_n\}$, we randomly choose an answer word $\mathcal{A}$ in the document. Note that, we restrict the answer word $\mathcal{A}$ to be a noun, as well as the answer word should appear at least twice in the document. The part-of-speech and sentence segmentation is identified using LTP Toolkit (Che et al., 2010). We do not distinguish the named entities and common nouns as Hill et al. (2015) did.
- Second, after the answer word $\mathcal{A}$ is chosen, the sentence that contains $\mathcal{A}$ is defined as the query $\mathcal{Q}$, in which the answer word $\mathcal{A}$ is replaced by a specific placeholder $\langle X \rangle$.
- Third, given the query $\mathcal{Q}$ and document $\mathcal{D}$, the target of the prediction is to recover the answer $\mathcal{A}$.

In this way, we can generate tremendous triples of $\langle \mathcal{D}, \mathcal{Q}, \mathcal{A} \rangle$ for training the proposed neural network, without any assumptions on the nature of the original corpus. Note that, unlike the previous work, using the method mentioned above, the document can be re-used for different queries, which makes it more general to generate large-scale training data for neural network training. Figure 1 shows an example of People Daily datasets.

Children's Fairy Tale. Except for the validation and test set of People Daily news data, we also present two out-of-domain test sets as well. The two out-of-domain test sets are made from the Children's Fairy Tale (CFT), which is fairly different from the news genre. The reason why we set out-of-domain test sets is that, the children's fairy tale mainly consists of the stories of animals or virtualized characters, and

[4] http://www.people.com.cn

	People Daily			Children's Fairy Tale	
	Train	Valid	Test	Test-auto	Test-human
# Query	870,710	3,000	3,000	1,646	1,953
Max # tokens in docs	618	536	634	318	414
Max # tokens in query	502	153	265	83	92
Avg # tokens in docs	379	425	410	122	153
Avg # tokens in query	38	38	41	20	20
Vocabulary	248,160			N/A	

Table 1: Statistics of People Daily datasets and Children's Fairy Tale datasets.

this prevents us from utilizing the gender information and world knowledge in the training data, which is important when solving several types of questions, such as coreference resolutions etc.

In CFT dataset, one test set is automatically generated using the algorithms described above, and the other one is made by the human, which suggest that the latter is harder than the former one. Because the automatically generated test sets are aware of the co-occurence or fixed collocation of words, and thus when the pattern around the query blank exactly appeared in the document, it is much easier for the machine to identify the correct answer. While in building human evaluation test set, we have eliminated these types of samples, which makes it harder for the machine to comprehend. Intuitively, the human evaluation test set is harder than any other previously published Cloze-style test sets.

The statistics of People Daily news datasets as well as Children's Fairy Tale datasets are listed in the Table 1.

3 Consensus Attention Sum Reader

In this section, we will introduce our attention-based neural network model for Cloze-style reading comprehension task, namely Consensus Attention Sum Reader (CAS Reader). Our model is primarily motivated by Kadlec et al. (2016), which aims to directly estimate the answer from the document, instead of making a prediction over the full vocabularies. But we have noticed that by just concatenating the final representations of the query RNN states are not enough for representing the whole information of query. So we propose to utilize every time slices of query, and make a *consensus* attention among different steps.

Formally, when given a set of training triple $\langle \mathcal{D}, \mathcal{Q}, \mathcal{A} \rangle$, we will construct our network in the following way. We first convert one-hot representation of the document $\mathcal{D}$ and query $\mathcal{Q}$ into continuous representations with a shared embedding matrix W_e. As the query is typically shorter than the document, by sharing the embedding weights, the query representation can be benefited from the embedding learning in the document side, which is better than separating embedding matrices individually.

Then we use two different bi-directional RNNs to get the contextual representations of document and query, which can capture the contextual information both in history and future. In our implementation, we use the bi-directional Gated Recurrent Unit (GRU) for modeling. (Cho et al., 2014)

$$e(x) = W_e * x, \ where \ \ x \in \mathcal{D}, \mathcal{Q} \tag{1}$$

$$\overrightarrow{h_s} = \overrightarrow{GRU}(e(x)) \tag{2}$$

$$\overleftarrow{h_s} = \overleftarrow{GRU}(e(x)) \tag{3}$$

$$h_s = [\overrightarrow{h_s}; \overleftarrow{h_s}] \tag{4}$$

We take h_{doc} and h_{query} to represent the contextual representations of document and query, both of which are in 3-dimension tensor shape. After that, we directly make a dot product of h_{doc} and $h_{query}(t)$ to get the "importance" of each document word, in respect to the query word at time t. And then,

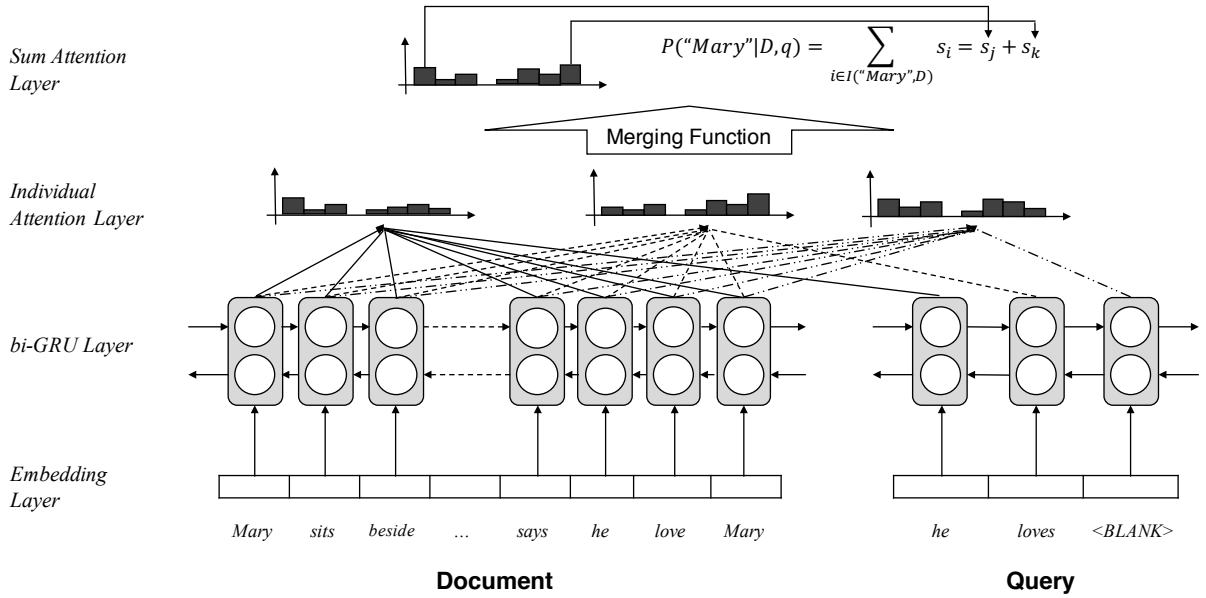


Figure 2: Architecture of the proposed Consensus Attention Sum Reader (CAS Reader).

we use the softmax function to get a probability distribution α over the document h_{doc}, also known as "attention".

$$\alpha(t) = softmax(h_{doc} \odot h_{query}(t)) \tag{5}$$

In this way, for every time step t in the query, we can get a probability distribution over the document, denoted as $\alpha(t)$, where $\alpha(t) = [\alpha(t)_1, \alpha(t)_1, ..., \alpha(t)_n]$, $\alpha(t)_i$ means the attention value of ith word in the document at time t, and n is the length of the document. To get a *consensus attention* over these individual attentions, we explicitly define a merging function f over $\alpha(1)...\alpha(m)$. We denote this as

$$s = f(\alpha(1), ..., \alpha(m)) \tag{6}$$

where s is the final attention over the document, m is the length of the query. In this paper, we define the merging function f as one of three heuristics, shown in equations below.

$$s \propto \begin{cases} softmax(\sum_{t=1}^{m} \alpha(t)), & \text{if } mode = sum; \\ softmax(\frac{1}{m} \sum_{t=1}^{m} \alpha(t)), & \text{if } mode = avg; \\ softmax(\max_{t=1...m} \alpha(t)), & \text{if } mode = max. \end{cases} \tag{7}$$

Finally, we map the attention result s to the vocabulary space V, and sum the attention value which occurs in different place of the document but shares the same word, as Kadlec et al. (2016) do.

$$P(w|\mathcal{D}, \mathcal{Q}) = \sum_{i \in I(w,\mathcal{D})} s_i, \;\; w \in V \tag{8}$$

where $I(w, \mathcal{D})$ indicate the position that word w appear in the document $\mathcal{D}$. Figure 2 shows the proposed neural network architecture.

4 Experiments

4.1 Experimental Setups

Training details of neural network models are illustrated as follows.

	Embed. # units	Hidden # units	Dropout
CNN News	384	256	None
CBTest NE	384	384	None
CBTest CN	384	384	None
People Daily & CFT	256	256	0.1

Table 2: Other neural network setups for each task. Note that, the dropout is only applied to the output of the GRUs.

	CNN News			CBT NE			CBT CN		
	Train	Valid	Test	Train	Valid	Test	Train	Valid	Test
# Query	380,298	3,924	3,198	108,719	2,000	2,500	120,769	2,000	2,500
Max # candidates	527	187	396	10	10	10	10	10	10
Avg # candidates	26	26	25	10	10	10	10	10	10
Avg # tokens	762	763	716	433	412	424	470	448	461
Vocabulary	118,497			53,063			53,185		

Table 3: Statistics of public Cloze-style reading comprehension datasets: CNN news data and CBTest NE(Named Entites) / CN(Common Nouns).

- Embedding Layer: We use randomly initialized embedding matrix with uniformed distribution in the interval $[-0.1, 0.1]$. Note that, no pre-trained word embeddings are used in our experiments.
- Hidden Layer: We initialized the GRU units with random orthogonal matrices (Saxe et al., 2013). As GRU still suffers from the gradient exploding problem, we set gradient clipping threshold to 10 in our experiments (Pascanu et al., 2013) .
- Vocabulary: For training efficiency and generalization, in People Daily and CFT datasets, we truncate the full vocabulary (about 200K) and set a shortlist of 100K. All unknown words are mapped to 10 different specific symbols using the method proposed by Liu et al. (2016). There is no vocabulary truncation in CNN and CBTest dataset.
- Optimization: We used the ADAM update rule (Kingma and Ba, 2014) with an initial learning rate $lr = 0.0005$, and used negative log-likelihood as the training objective function. The batch size is set to 32.

Other neural network setups, such as dimensions of embedding layer and hidden layer, and dropout (Srivastava et al., 2014) for each task, are listed in Table 2. We trained model for several epochs and choose the best model according to the performance of validation set. All models are trained on Tesla K40 GPU. Our model is implemented with Theano (Theano Development Team, 2016) and Keras (Chollet, 2015).

4.2 Results on Public Datasets

To verify the effectiveness of our proposed model, we first tested our model on public datasets. Our evaluation is carried out on CNN news datasets (Hermann et al., 2015) and CBTest NE/CN datasets (Hill et al., 2015), and the statistics of these datasets are listed in Table 3. No pre-processing is done with these datasets. The experimental results are given in Table 4. We evaluate the model in terms of its accuracy. Due to the time limitations, we did not evaluate our model in ensemble.

CNN News. The performance on CNN news datasets shows that our model is on par with the Attention Sum Reader, with 0.4% decrease in validation and 0.5% improvements in the test set. But we failed to outperform the Stanford AR model. While the Stanford AR utilized GloVe embeddings (Pennington et

	CNN News		CBTest NE		CBTest CN	
	Valid	Test	Valid	Test	Valid	Test
Deep LSTM Reader†	55.0	57.0	-	-	-	-
Attentive Reader†	61.6	63.0	-	-	-	-
Impatient Reader†	61.8	63.8	-	-	-	-
Human (context+query)‡	-	-	-	*81.6*	-	*81.6*
LSTMs (context+query)‡	-	-	51.2	41.8	62.6	56.0
MemNN (window + self-sup.)‡	63.4	66.8	70.4	66.6	64.2	63.0
Stanford AR♮	*72.4*	*72.4*	-	-	-	-
AS Reader♯	**68.6**	69.5	73.8	68.6	**68.8**	63.4
CAS Reader (mode: avg)	68.2	**70.0**	**74.2**	**69.2**	68.2	**65.7**

Table 4: Results on the CNN news, CBTest NE (named entity) and CN (common noun) datasets. Results marked with † are taken from (Hermann et al., 2015), and ‡ are taken from (Hill et al., 2015), and ♮ are taken from (Chen et al., 2016), and ♯ are taken from (Kadlec et al., 2016)

	People Daily		Children's Fairy Tale	
	Valid	Test	Test-auto	Test-human
AS Reader	64.1	67.2	40.9	33.1
CAS Reader (mode: avg)	**65.2**	**68.1**	41.3	**35.0**
CAS Reader (mode: sum)	64.7	66.8	**43.0**	34.7
CAS Reader (mode: max)	63.3	65.4	38.3	32.0

Table 5: Results on People Daily datasets and Children's Fairy Tale (CFT) datasets.

al., 2014), and only normalized the probabilities over the named entities in the document, rather than all the words, and this could make a difference in the results. But in our model, we do not optimize for a certain type of dataset, which make it more general.

CBTest NE/CN. In CBTest NE dataset, our model gives slight improvements over AS Reader, where 0.4% improvements in the validation set and 0.6% improvements in the test set. In CBTest CN, though there is a slight drop in the validation set with 0.6% declines, there is a boost in the test set with an absolute improvements 2.3%, which suggest our model is effective, and it is beneficial to consider every slices of the query when answering.

4.3 Results on Chinese Reading Comprehension Datasets

The results on Chinese reading comprehension datasets are listed in Table 5. As we can see that, the proposed CAS Reader significantly outperform the AS Reader in all types of test set, with a maximum improvements 2.1% on the CFT test-auto dataset. The results indicate that making a consensus attention over multiple time steps are better than just relying on single attention (as AS Reader did). This is similar to the use of "model ensemble", which is also a consensus voting result by different models.

We also evaluated different merging functions. From the results, we can see that the avg and sum methods significantly outperform the max heuristics, and the max heuristics failed to outperform the AS Reader. A possible reason can be explained that the max operation is very sensitive to the noise. If a non-answer word is given to a high probability in one time step of the query, the avg and sum could easily diminish this noise by averaging/summing over other time steps. But once there is a higher value given to a non-answer word in max situation, the noise can not be removed, and will preserve till the end of final attentions, which will influence the predictions a lot.

Also, we have noticed that, though we have achieved over 65% in accuracy among People Daily datasets, there is a significant drop in the two CFT test sets. Furthermore, the the human evaluated test set meets a sharp decline over 8% accuracy to the automatically generated test set. The analyses can be

concluded as

- As we regard the CFT datasets as the out-of-domain tests, there is a gap between the training data and CFT test data, which poses declines in these test sets. Such problems can be remedied by introducing the similar genre of training data.

- Regardless of the absolute accuracies in CFT datasets, the human test set is much harder for the machine to read and comprehend as we discussed before. Through these results, we can see that there is a big gap between the automatically generated queries and the human-selected questions.

Note that, in our human-evaluated test set, the query is also formulated from the *original* sentence in the document, which suggest that if we use more general form of queries, there should be another rise in the comprehension difficulties. For example, instead of asking "I went to the **XXXXX** this morning .", we change into a general question form of "Where did I go this morning ?", which makes it harder for the machine to comprehend, because there is a gap between the general question form and the training data.

5 Related Work

Many NN-based reading comprehension models have been proposed, and all of them are attention-based models, which indicate that attention mechanism is essential in machine comprehensions.

Hermann et al. (2015) have proposed a methodology for obtaining a large quantities of $\langle \mathcal{D}, \mathcal{Q}, \mathcal{A} \rangle$ triples. By using this method, a large number of training data can be obtained without much human intervention, and make it possible to train a reliable neural network to study the inner relationships inside of these triples. They used attention-based neural networks for this task. Evaluation on CNN/DailyMail datasets showed that their approach is effective than traditional baselines.

Hill et al. (2015) also proposed a similar approach for large scale training data collections for children's book reading comprehension task. By using window-based memory network and self-supervision heuristics, they have surpass all other methods in predicting named entities(NE) and common nouns(CN) on both the CBT and the CNN QA benchmark.

Our CAS Reader is closely related to the work by Kadlec et al. (2016). They proposed to use a simple model that using the attention result to directly pick the answer from the document, rather than computing the weighted sum representation of document using attention weights like the previous works. The proposed model is typically motivated by Pointer Network (Vinyals et al., 2015). This model aims to solve one particular task, where the answer is only a single word and should appear in the document at least once. Experimental results show that their model outperforms previously proposed models by a large margin in public datasets (both CBTest NE/CN and CNN/DailyMail datasets).

Liu et al. (2016) proposed an effective way to generate and exploit large-scale pseudo training data for zero pronoun resolution task. The main idea behind their approach is to automatically generate large-scale pseudo training data and then using the neural network model to resolve zero pronouns. They also propose a two-step training: a pre-training phase and an adaptation phase, and this can be also applied to other tasks as well. The experimental results on OntoNotes 5.0 corpus is encouraging and the proposed approach significantly outperforms the state-of-the-art methods.

In our work, we proposed an entirely new Chinese reading comprehension dataset, which add the diversity to the existing Cloze-style reading comprehension datasets. Moreover, we propose a refined neural network model, called Consensus Attention-based Sum Reader. Though many impressive progress has been made in these public datasets, we believe that the current machine comprehensions are still in the pre-mature stage. As we have discussed in the previous section, to answer a *pseudo query* to the document is not enough for machine comprehension. The general question form can be seen as a comprehensive processing of our human brains. Though our human-evaluated test set is still somewhat easy for machine to comprehend (but harder than the automatically generated test set), releasing such dataset will let us move a step forward to the real-world questions, and becomes a good bridge between automatic questions and real-world questions.

6 Conclusion

In this paper, we introduce the first Chinese reading comprehension datasets: People Daily and Children's Fairy Tale. Furthermore, we also propose a neural network model to handle the Cloze-style reading comprehension problems. Our model is able to take all question words into accounts, when computing the attentions over the document. Among many public datasets, our model could give significant improvements over various state-of-the-art baselines. And also we set up a baseline for our Chinese reading comprehension datasets, that we hopefully make it as a starter in future studies.

The future work will be carried out in the following aspects. First, we would like to work on another human-evaluated dataset, which will contain the real-world questions and is far more difficult than the existing datasets publicly available. Second, we are going to investigate hybrid reading comprehension models to tackle the problems that rely on comprehensive induction of several sentences.

Acknowledgements

We would like to thank the anonymous reviewers for their thorough reviewing and proposing thoughtful comments to improve our paper. This work was supported by the National 863 Leading Technology Research Project via grant 2015AA015407, Key Projects of National Natural Science Foundation of China via grant 61632011, and National Natural Science Youth Foundation of China via grant 61502120.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 13–16. Association for Computational Linguistics.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Association for Computational Linguistics (ACL)*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

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

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ting Liu, Yiming Cui, Qingyu Yin, Shijin Wang, Weinan Zhang, and Guoping Hu. 2016. Generating and exploiting large-scale pseudo training data for zero pronoun resolution. *arXiv preprint arXiv:1606.01603*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Wilson L Taylor. 1953. Cloze procedure: a new tool for measuring readability. *Journalism and Mass Communication Quarterly*, 30(4):415.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Semantic Annotation Aggregation with Conditional Crowdsourcing Models and Word Embeddings

Paul Felt
IBM Watson*
plfelt@us.ibm.com

Eric K. Ringger
Facebook*
eringger@fb.com

Kevin Seppi
Brigham Young University
kseppi@byu.edu

Abstract

In modern text annotation projects, crowdsourced annotations are often aggregated using item response models or by majority vote. Recently, item response models enhanced with generative data models have been shown to yield substantial benefits over those with conditional or no data models. However, suitable generative data models do not exist for many tasks, such as semantic labeling tasks. When no generative data model exists, we demonstrate that similar benefits may be derived by conditionally modeling documents that have been previously embedded in a semantic space using recent work in vector space models. We use this approach to show state-of-the-art results on a variety of semantic annotation aggregation tasks.

1 Introduction

Text annotation is a crucial part of natural language processing (NLP), enabling content analysis (Krippendorff, 2012) and providing training data for supervised and semi-supervised machine learning algorithms in NLP. Modern text annotation is often crowdsourced, meaning that the work is divided up and assigned to internet workers on micro-task marketplaces such as Amazon's Mechanical Turk[1] or CrowdFlower.[2] Although crowdsourced annotations tend to be error-prone, high quality labels may be obtained by aggregating multiple redundant low-quality annotations (Surowiecki, 2005). For many tasks, aggregated crowdsourced judgments have been shown to be more reliable than expert judgments (Snow et al., 2008; Cao et al., 2010; Jurgens, 2013).

Traditionally annotations were aggregated via majority vote. More sophisticated approaches jointly model annotator reliability and document labels. These models can down-weight the annotations of workers who often disagree with others and up-weight the annotations of workers who often agree with others. However, when annotation error is high or few annotations are available it can be difficult for these models to know which annotators to trust. Jin and Ghahramani (2002), Raykar et al. (2010), Liu et al. (2012), and Yan et al. (2014) show that crowdsourcing annotation models can be enhanced by conditioning the model on the document data (e.g., word content), improving the model by identifying annotators whose judgments tend to agree with word patterns found in the documents being annotated.

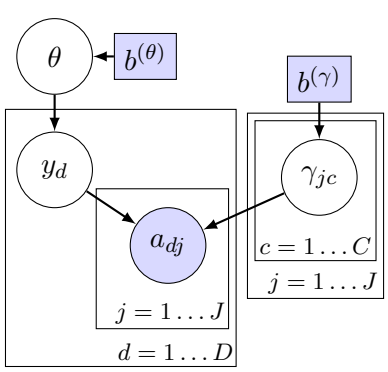
Recent work has shown that generative data models allow crowdsourcing models to converge to useful estimates even when few annotations are available, whereas by the time a conditional model has enough information (in the form of annotations) to be useful, the problem is often largely solved by majority vote (Felt et al., 2015b). However, although generative data modeling has been shown to be effective in categorizing text according to its topic, realistic generative data models are not always available. In this paper, we use advances in text representation to demonstrate that data-conditional annotation models can

* This work was completed while the first and second authors were at Brigham Young University.

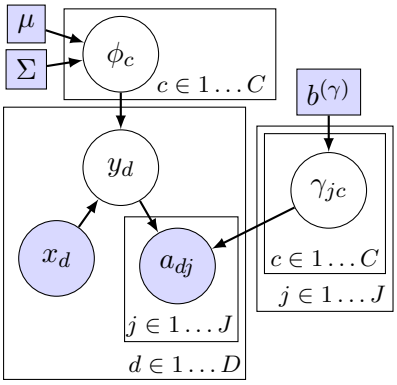
[1] http://mturk.com

[2] http://crowdflower.com

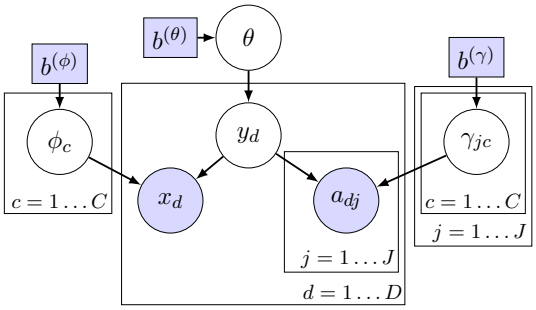
Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1787–1796, Osaka, Japan, December 11-17 2016.



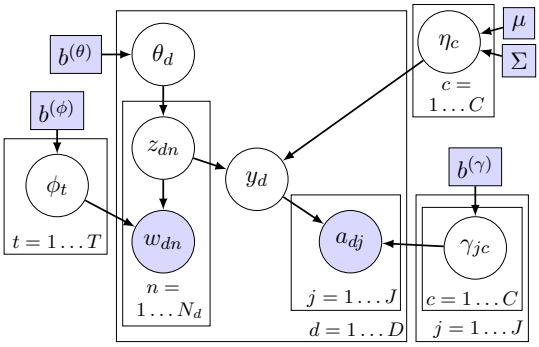
(a) **ITEMRESP** as a plate diagram.



(b) **LOGRESP** as a plate diagram.



(c) **MOMRESP** as a plate diagram.



(d) **CSLDA** as a plate diagram.

Figure 1: Round nodes are variables with distributions. Rectangular nodes are values without distributions. Shaded nodes are observed. D, J, C, and T are the number of documents, annotators, classes, and topics, respectively. N_d is the number of words in document d.

achieve gains similar to those of data-generative annotation models, including for tasks where generative data models are currently unavailable, such as paired text similarity and compatibility.

In Section 2 we briefly review annotation models with generative and conditional data components and also discuss representing words and documents via embeddings in a semantic vector space. In Section 3 we show that data-conditional annotation models succeed on a variety of text datasets and classification tasks. In Section 4 we conduct error analysis on an anomalous dataset, and in Sections 5 and 6 we list additional related work and summarize our conclusions.

2 Background

Most crowdsourcing models extend the item-response model of Dawid and Skene (1979). The Bayesian version of this model, referred to here as ITEMRESP, is illustrated by Figure 1a and defines the joint distribution $p(y, a, \theta, \gamma)$, where a is the annotation and y is an unobserved document label. In the generative story for this model, a confusion matrix γ_j is drawn for each human annotator j. Each row γ_{jc} of the confusion matrix γ_j is drawn from $Dir(b^{(\gamma)}_{jc})$, and encodes a probability distribution over label classes that annotator j is apt to choose when presented with a document whose true label is c. A general prior over label classes θ is drawn from $Dirichlet(b^{(\theta)})$, then for each document d an unobserved document label y_d is drawn from categorical distribution $Cat(\theta)$. Finally, annotations are generated as annotator j corrupts the true label y_d according to the multinomial distribution $Mult(\gamma_{jy_d})$.

2.1 Data-aware annotation models

Notice that the ITEMRESP model entirely ignores document data x (e.g., words). ITEMRESP extensions model the data x and related feature parameters ϕ either conditionally $p(y, a, \gamma, \phi|x)$ or else generatively $p(y, a, x, \theta, \gamma, \phi)$.

Conditional crowdsourcing models make few assumptions about the data and can use the same general log-linear structure as maximum entropy classifiers, which have enjoyed success in a large number of classification problems. Figure 1b shows a Bayesian formulation of a conditional crowdsourcing model $p(y, a, \gamma, \phi|x)$. For each class k, ϕ_k is drawn from a multivariate Gaussian distribution $Gauss(0, \Sigma)$. Then for each document, y_d is drawn from a log-linear distribution $p(y_d|\phi, x) \propto e^{\phi_{y_d}^T x}$. For this reason we refer to this model as LOGRESP. LOGRESP is representative of a popular class of conditional crowdsourcing models (Jin and Ghahramani, 2002; Raykar et al., 2010; Liu et al., 2012; Yan et al., 2014). In previous work we found that LOGRESP often provides only incremental gains over majority vote (Felt et al., 2015b). This partly because its ϕ estimates, like other conditional models, tend to converge relatively slowly with $O(N)$ labeled examples (Ng and Jordan, 2001). By the time LOGRESP's ϕ estimates become useful, there are often enough annotations available that majority vote is sufficient.

Generative data-aware crowdsourcing models have complementary strengths. Although they make strong assumptions about the data that they model, their parameters can converge quickly with only $O(logn)$ labeled examples (Ng and Jordan, 2001). This means that when data does not violate a generative model's assumptions too badly, the generative model can offer dramatic improvements over majority vote, especially when few annotations are available. Figures 1c and 1d depict two such generative models. In Figure 1c, each document d draws its data x_d from a class-conditional multinomial word distribution $Mult(\phi_{y_d})$. We call this model MOMRESP because it models data as a mixture of multinomials. MOMRESP represents a common class of generative crowdsourcing models (Bragg et al., 2013; Lam and Stork, 2005; Simpson and Roberts, 2015). Figure 1d shows CSLDA, a more sophisticated generative crowdsourcing model based on supervised topic modeling (Felt et al., 2015a).

For inference in the ITEMRESP, LOGRESP and MOMRESP crowdsourcing models, we use existing variational inference (Felt et al., 2015b). Note that variational inference for ITEMRESP is easily derived as a special case of MOMRESP inference where terms involving the data are dropped. Inference for CSLDA is stochastic expectation maximization.

2.2 Word and Document Representations

Documents have historically been represented in NLP algorithms by large, sparse word count vectors $x_d = \sum_{n=1}^{|x_d|} \mathbb{1}(x_{dn})$ where $\mathbb{1}(x_{dn})$ is a one-hot vector having length equal to the size of the vocabulary. However, word-count document representations have a number of drawbacks. They define a space that is often so high-dimensional and sparse that inter-document distances and other vector computations have little meaning. In word-count representations, features that strongly relate to one another (e.g., the words "horse" and "equine") are represented as entirely orthogonal dimensions, exploding the number of parameters needed by downstream learning algorithms.

Recently, methods have been developed to represent words as locations in low-dimensional vector spaces where distance and direction encode semantic and syntactic meaning (Mikolov et al., 2013a; Pennington et al., 2014). These embedding vectors have been shown to improve a variety of language tasks including named entity recognition, phrase chunking (Turian et al., 2010), relation extraction (Nguyen and Grishman, 2014), and part of speech induction (Lin et al., 2015). The hypothesis investigated by the current work is that semantic, vector-based text representations can help conditional annotation aggregation models achieve some of the same early performance advantage seen in their generative counterparts, as well as help them operate on datasets that make semantic distinctions. This hypothesis is plausible *a priori* because using data embeddings is akin to using semi-supervision to enable faster learning. The reason for this is that data embeddings are traditionally induced in an unsupervised manner on extremely large corpora before being applied to a downstream supervised task. In addition, operating on dense, low-dimensional vector data reduces the number of model parameters which can also reduce the number of instances required to learn effectively.

Although it might be possible to extend the CSLDA model to generatively model the embedding as described by Das et al. (2015), but it is unclear how the inference approach used there (Gibbs sampling based on Cholesky decompositions) would be efficiently applied in the context of the CSLDA model, thus we leave this possibility to future work and focus on using embeddings discriminatively. We use the

Dataset	Size	Unique Annotators	Annotations per Instance	Classes	Average Doc size	Gold Labels	Timestamps
Sentiment	1,000	83	5	2	12.8	1,000	No
Weather	1,000	102	20	5	13.6	724	Yes
Compatibility	17,977	411	10	2	2×1	15,157	No
Paraphrase	4,000	119	5	2	2×11.2	838	Yes

Table 1: Dataset statistics. Evaluation metrics are calculated only over the subset of each dataset for which gold labels are available. The timestamps column indicates whether or not it is known exactly when each annotation was generated. When available, timestamps determine the order of annotation in reported learning curves.

word2vec algorithm introduced by Mikolov et al. (2013a) to convert words to vectors for the purposes of this paper, understanding that other text embedding methods may be swapped in for additional improvements as they are developed. Word2vec operates on individual words. When sentences or documents must be vectorized, we do so by averaging the vectors of each word in the sentence or document without any word filtering or selection. While we briefly experimented with gensim's *doc2vec* implementation of Le and Mikolov (2014), we noticed little benefit for the twitter data explored in this paper, possibly because of the short, focused nature of tweets.

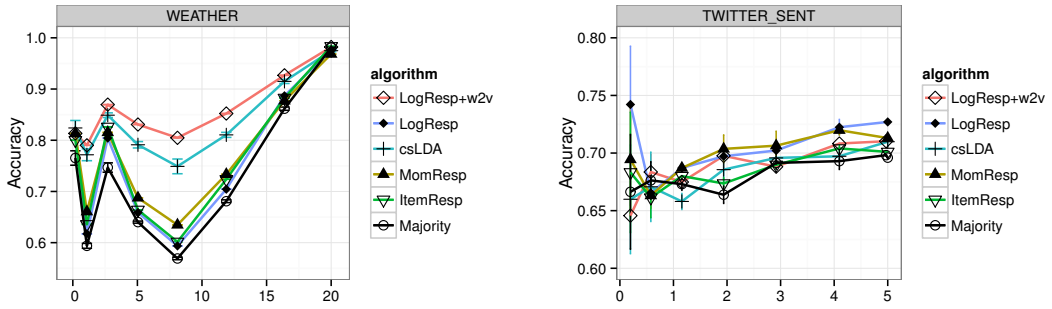
3 Experiments

In order to test the hypothesis that vector space document representations can improve conditional crowdsourcing model performance on semantic classification tasks, we plot and visually compare learning curves charting the accuracy of the labels inferred by various crowdsourcing models. All of the algorithms from Section 2 are trained on sparse one-hot vector representations of text; and an additional variant of LOGRESP is reported which is trained on low dimensional semantic vector representations (LOGRESP+w2v). Learning curves advance as annotations from multiple annotators are incrementally added to the set of annotations available to each model. When annotation timestamps are available, annotations are added in the empirical order in which they were created. When unavailable, annotations are added in randomized breadth-first order so that each document gets one annotation before any document receives a second. Accuracy is computed over the subset of gold labels having at least one annotation. This process illustrates model behavior both when few annotations per document are available (in the early stages of learning curves) and when many annotations per document are available (in the late stages of the learning curves).

For our word embedding model, we use the word2vec algorithm, implemented by the gensim document processing library (Řehůřek and Sojka, 2010) to train word embeddings on a June 2015 snapshot of the English Wikipedia pages and articles dump (approximately 2.1 billion words). The word2vec algorithm requires a number of parameters, which we report here for replicability. We train embeddings using a context window of 10 words, discarding words that occur fewer than 5 times. For training, we use hierarchical sampling with a skip-gram model and no negative sampling. Embeddings of size 300 are learned. All of these settings are rather standard for a large corpus like Wikipedia.

3.1 Datasets

In order to calculate the accuracy of inferred labels, we require datasets that have both crowdsourced annotations as well as gold standard labels for evaluation. We identify four suitable datasets, briefly describing both their annotation task as well as the way their gold standard labels are constructed. For all Twitter data, we use the Twitter text normalization dictionary of Han et al. (2012) to normalize tweets before embedding them. Note that for two of the datasets described below, **Compatibility** and **Weather**, the gold standard does not consist of the hand labels of an expert, but rather is constructed from the consensus vote of a reasonable number of crowd workers. A fundamental tenet of crowdsourcing is that



(a) The **Weather** Dataset (b) The **Sentiment** Dataset

Figure 2: Inferred label accuracy (y axis) learning curves of various crowdsourcing models. The x axis is the number of annotations $\times$ 1,000.

inexpert workers are, in aggregate, trustworthy. The purpose of automatic aggregation models such as those decribed in Section 2 is to arrive at the same judgment with a few annotations that a simpler scheme like majority vote would have arrived at given many annotations. Therefore, a crowd-constructed gold standard is appropriate for evaluation of such models.

Paraphrase. During an exploratory annotation phase, Xu et al. (2014) paid Amazon Mechanical Turk workers to annotate 4,000 tweet pairs with binary judgments indicating whether or not the tweet pair communicates the same information.[3] For example, the tweets "Star Wars Return of the Jedi is on" and "My favorite Star Wars movie is on" communicate mostly the same information and are labeled as paraphrases of one another, while the tweet "and of course because I drink and like Star Wars I know nothing about football" communicates different information, and is labeled as not a paraphrase of the other two tweets. Each tweet pair received 5 binary annotations. Gold standard labels were constructed for a subset of 838 tweet pairs by experts who rated each pair on a scale from 0-5. Following the original authors, expert ratings of 0-2 are labeled *no paraphrase*, and 4-5 are labeled *paraphrase*. Ratings of 3 are ignored for evaluation purposes.

Compatibility. Kruszewski and Baroni (2015) paid CrowdFlower workers to rate word pairs according to their semantic compatibility, meaning that the two words can be used to refer to the same real-world entity.[3] For example, the words "artist" and "teacher" are compatible with one another, whereas "bread" and "rattlesnake" are not. Each word pair was rated by 10 different annotators on a 7-point scale. Following the original authors, the gold standard is constructed by labeling items with a mean rating less than 1.6 as incompatible, and those with a mean rating greater than 3.7 as compatible. Ratings between 1.7 and 3.7 are ignored for evaluation purposes.

Sentiment. Mozafari et al. (2014) paid Mechanical Turk workers to annotate tweets with binary sentiment labels: *Positive* and *Negative*, and manually created gold standard labels using trusted (non-crowdsourced) labelers.[4]

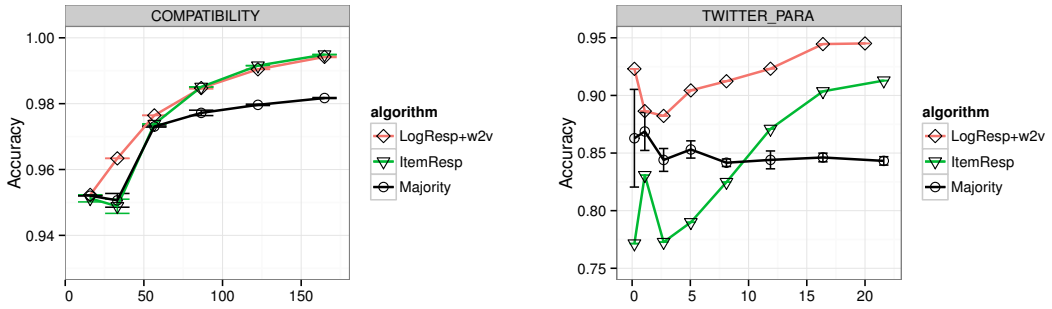
Weather. CrowdFlower has made a number of annotated datasets freely available.[5] In their "Weather sentiment" dataset, 20 annotators were paid to annotate weather-related tweets with sentiment labels: *Negative*, *Neutral*, *Positive*, *Unrelated to weather*, and *I can't tell*. A gold standard was constructed by running a separate evaluation task called "Weather sentiment evaluated" in which 10 additional annotators were paid to annotate the majority vote label from the previous task as correct or incorrect. We form a gold standard from those labels that are judged to be correct by at least 9/10 annotators.

Our focus in this paper is on challenging sentiment classification tasks which tend to have few classes. In preliminary experiments we observed that even on topical classification datasets such as 20 Newsgroups or the LDC-labeled Enron emails, LOGRESP is perceptibly improved by running on vector space features, although CSLDA remains dominant. Note that in the experiments described below, the **Weather** and **Paraphrase** annotations are applied in the order indicated in the dataset, however, the ac-

[3] Not publicly available at the time of writing.

[4] `http://web.eecs.umich.edu/~mozafari/datasets/crowdsourcing/index.html`

[5] `http://www.crowdflower.com/data-for-everyone`



(a) The **Compatibility** Dataset

(b) The **Paraphrase** Dataset

Figure 3: Inferred label accuracy (y axis) learning curves of vector space crowdsourcing models on tasks with paired-comparison data for which generative crowsdsourcing models are unsuitable. The x axis is the number of annotations $\times$ 1,000.

tual order of **Sentiment** and **Compatibility** annotations is not provided in the data set so they are applied in random order.

3.2 Comparison with generative methods

Two datasets, **Weather** and **Sentiment**, are traditional text classification tasks with instances consisting of one label per text document. We use these datasets to compare the performance of LOGRESP trained on vector space text features (LOGRESP+w2v) to the performance of alternatives using sparse word-count features, including LOGRESP as well as the generative models MOMRESP and CSLDA. The majority vote and ITEMRESP algorithms serve as baselines. In Figure 2a we see that on the **Weather** dataset, LOGRESP with embeddings (LOGRESP+w2v) performs far better than traditional LOGRESP, and even outperforms the previous state-of-the-art for this dataset, CSLDA. Although all algorithms eventually reach a high level of performance on the **Weather** dataset, we prefer algorithms like LOGRESP+w2v that reach high levels of accuracy using as few annotations as possible, potentially reducing annotation cost. The accuracy of all of the models is unstable until a reasonable number of annotations is obtained, 5,000 to 10,000 in this case. Models which make little or no use of the words themselves (especially Majority vote and ITEMRESP) are particularly susceptible to variability in the initial annotations. Data-sensitive models like CSLDA and LOGRESP+w2v are far less susceptible to these swings.

In Figure 2b we see that no algorithm improves much over majority vote on the **Sentiment** dataset. Also, the baseline accuracy levels at the end of the curves are extremely low for a binary classification task with 5 annotations per instance, meaning that annotator accuracy is unusually low. We include this dataset as a reminder that the "no free lunch" theorem applies to crowdsourcing models the same as to any other class of models. In Section 4 we explore in more detail what makes the **Sentiment** dataset particularly difficult for crowdsourcing models.

3.3 When generative methods are unavailable

The datasets **Compatibility** and **Paraphrase** both involve data pairs being compared for semantic content (see Section 3.1 for examples of these tasks). **Compatibility** compares the semantic compatibility of word pairs while **Paraphrase** compares the semantic similarity of tweets pairs. Generative crowdsourcing models such as MOMRESP and CSLDA do not natively accommodate such paired data since the data does not comport with these models' generative stories. To make them do so would require restructuring the models and their inference procedures. On the other hand, it is straightforward to combine the semantic vector representations of two documents v_1 and v_2. We do so by forming a new feature vector

$$\begin{aligned} v_{new} = &\langle cos(v_1, v_2), \\ &L1(v_1 - v_2), L2(v_1 - v_2), \\ &PC_{50}(v_1), PC_{50}(v_2), \\ &PC_{50}(v_1) - PC_{50}(v_2)\rangle \end{aligned}$$

		Alternative Gold Standard			
		Neg	Pos	None	Hard
Gold Standard	Neg	35	**6**	**5**	**1**
	Pos	**9**	29	**11**	**3**

Table 2: Disagreement between the original gold standard (rows) and an alternative gold standard (columns) on 100 arbitrarily selected tweets. The alternative gold standard employs a more flexible label set. Neg=*Negative*, Pos=*Positive*, None=*No sentiment*, Hard=*Can't decide*. Bold values reflect various kinds of disagreement between the labelings.

where $cos(\cdot)$ is cosine distance, $L1(\cdot)$ and $L2(\cdot)$ are the first two p-norms, and $PC_n(v)$ is a vector consisting of the top-n components of v, found via PCA on the set of embedded documents.

Figure 3a shows that LOGRESP with semantic embeddings outperforms majority vote and ITEMRESP baselines on the word **Compatibility** dataset when there are fewer than 3 annotations per instance. Later in the learning curve, when annotations become sufficiently abundant (up to 20 per instance), the data appears to no longer be helpful. Fortunately, incorporating data information using LOGRESP with semantic embeddings appears to never actually hurt compared with using just ITEMRESP.

On the other hand, Figure 3b shows that on the **Paraphrase** dataset, LOGRESP with semantic embeddings dramatically outperforms the baselines along the entire learning curve. This is partly because, unlike the **Compatibility** task, **Paraphrase** accuracy is low enough to permit the improved vector data representation to benefit LOGRESP.

3.4 Summary of experiments

Overall, with the exception of the somewhat anomalous **Sentiment** dataset, which we examine in more detail in Section 4, running LOGRESP on semantically embedded data is always an improvement over LOGRESP running on traditional document representations. The gains in Figures 3a and 3b strongly confirm the hypothesis that semantic embeddings can allow crowdsourcing models to see some of the same efficiency gains for challenging semantic labeling tasks as previously observed using generative data-aware crowdsourcing models on more straight-forward topical labeling tasks. Not only that, but the fact that semantic embeddings lend themselves to sensible vector-space operations allows data-aware crowdsourcing models to be applied to complex tasks like labeling paired text similarity and compatibility, which was not previously possible.

4 Sentiment dataset error analysis

An analysis of the **Sentiment** dataset (results in Figure 2b) helps explain why algorithms behave so differently on it than on the other datasets. Kilgarriff (1998) identifies three sources of annotation noise: inherent data ambiguity, poor task definition, and annotator error. The crowdsourcing models used here account only for annotator error. However, the **Sentiment** dataset task definition dictates that each tweet be labeled with a binary sentiment label, forcing annotators to make arbitrary decisions when tweets encode little or ambiguous sentiment. For example, the tweet "EBTM.com is BACK?!" is genuinely ambiguous, and the tweet "@comeagainjen if you dont, neither do i" contains little explicit sentiment. Kilgarriff (1998) suggests that an important step towards addressing data ambiguity is ensuring that tasks are defined so that annotators have the ability to explicitly identify ambiguous instances.

To assess the impact of inherent data ambiguity and task definition on the **Sentiment** dataset, we arbitrarily chose 100 instances with gold labels and compared them with an alternative gold standard labeled according to a more flexible annotation scheme. The latter labels were generated by a pair of graduate students working in tandem. We added a *No sentiment* label to address problems with task definition and a *Can't decide* label to capture inherent data ambiguity. Table 2 shows the confusion matrix between the two gold standard sets. A large percentage (16%) of tweets were assigned to *No sentiment* in the alternative gold standard. This indicates that task definition affects this dataset strongly.

Although this analysis does not make this dataset any less interesting (indeed, the problems associated with modeling and correcting the effects of task misspecification are highly interesting), it does warn us that it is less representative than the others of annotation projects where effort is made up-front to iteratively refine an annotation specification before paying for large number of annotations.

5 Additional Related Work

A sizable body of research is currently underway to improve vector word representations. Although most commonly word embeddings are trained in an unsupervised manner, they may be tuned to maximize performance on a particular target task (Le and Mikolov, 2014). They may also be supervised by multiple tasks simultaneously (Collobert et al., 2011). Others fit one embedding per word sense rather than per lexical type, improving model fit (Neelakantan et al., 2014). Srikumar and Manning (2014) embed not only word types, but also label types, modeling the fact that some labels are more similar than others.

Another line of work explores ways of embedding larger spans of text. Although words tend to compose surprisingly well simply via linear combination, many phrases are more than the sum of their parts (e.g., collocations like "White House"). These can be dealt with by using heuristics to identify and combine token phrases (Mikolov et al., 2013b). Other approaches incorporate composition functions as first-class constituents of the objective function itself. Mitchell and Lapata (2010) motivate a general composition framework and compare a number of simple instantiations, including additive, multiplicative, and tensor product combination. Socher et al. (2012) assign vectors representing semantic content and matrices representing semantic transformations to every node in a parse tree. Fyshe et al. (2015) focus on learning phrasal representations whose dimensions are easily interpretable by humans, similar to successful models whose topics are easy for humans to recognize and name because they align with a topic distinction known *a priori* to the human.

In this work we focus on using instance data to improve probabilistic crowdsourcing models. Passonneau and Carpenter (2014) argue that probabilistic crowdsourcing models are generally more effective and reliable than traditional chance-adjusted agreement heuristics such as Krippendorff's alpha for assessing corpus quality (Krippendorff, 2012). Other previous work in crowdsourcing ignores the data being annotated, focusing instead on modeling other aspects of the annotation process, such as item difficulty and noise (Whitehill et al., 2009; Welinder et al., 2010). Hovy et al. (2013) model the nonlinear nature of human reliability by adding binary variables to each annotator indicating whether they are a spammer or not. These extensions are orthogonal to the issue explored by this paper and could be incorporated into any of the models used here.

6 Conclusions and Future Work

Previous work indicates that generative crowdsourcing models enjoy significant learning advantages when aggregating topic-based document labels. Unfortunately, some text classification tasks make distinctions for which no good generative text models currently exist, such as labeling the similarity or compatibility of paired words and sentences. We have demonstrated that vector space text embeddings can be used to gain similar advantages using conditional models and for an even broader class of data. Using this approach, we have shown state-of-the-art annotation aggregation for several semantic annotation aggregation tasks. Future work includes experimenting with deep learning methods of jointly learning embeddings and hidden labels, rather than pipelining the two tasks.

Acknowledgments

This work was supported by the collaborative NSF Grant IIS-1409739 (BYU) and IIS-1409287 (UMD). Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

References

Jonathan Bragg, Mausam, and Daniel S. Weld. 2013. Crowdsourcing multi-label classification for taxonomy creation. In *Proc. Conference on Human Computation and Crowdsourcing (HCOMP)*.

Jing Cao, S Lynne Stokes, and Song Zhang. 2010. A Bayesian approach to ranking and rater evaluation an application to grant reviews. *Journal of Educational and Behavioral Statistics*, 35(2):194–214.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian lda for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 795–804, Beijing, China, July. Association for Computational Linguistics.

Alexander P. Dawid and Allan M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, pages 20–28.

Paul Felt, Eric K. Ringger, Jordan Boyd-Graber, and Kevin Seppi. 2015a. Making the most of crowdsourced document annotations: Confused supervised LDA. In *Proc. Conference on Computational Natural Language Learning (CoNLL)*.

Paul Felt, Eric K. Ringger, Kevin Seppi, and Robbie A. Haertel. 2015b. Early gains matter: A case for preferring generative over discriminative crowdsourcing models. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Alona Fyshe, Leila Wehbe, Partha P. Talukdar, Brian Murphy, and Tom M. Mitchell. 2015. A compositional and interpretable semantic space. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard H. Hovy. 2013. Learning whom to trust with MACE. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Rong Jin and Zoubin Ghahramani. 2002. Learning with multiple labels. In *Proc. Advances in Neural Information Processing Systems (NIPS)*.

David Jurgens. 2013. Embracing ambiguity: A comparison of annotation methodologies for crowdsourcing word sense labels. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Adam Kilgarriff. 1998. Gold standard datasets for evaluating word sense disambiguation programs. *Computer Speech & Language*, 12(4):453–472.

Klaus Krippendorff. 2012. *Content Analysis: An Introduction to its Methodology*. Sage.

Germán Kruszewski and Marco Baroni. 2015. So similar and yet incompatible: Toward the automated identification of semantically compatible words. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Chuck P. Lam and David G. Stork. 2005. Toward optimal labeling strategy under multiple unreliable labelers. In *Proc. AAAI Spring Symposium: Knowledge Collection from Volunteer Contributors*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proc. International Conference on Machine Learning (ICML)*.

Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. 2015. Unsupervised POS induction with word embeddings. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Qiang Liu, Jian Peng, and Alex T. Ihler. 2012. Variational inference for crowdsourcing. In *Proc. Advances in Neural Information Processing Systems (NIPS)*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proc. Workshops at International Conference on Learning Representations (ICLR)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*.

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

Barzan Mozafari, Purna Sarkar, Michael Franklin, Michael Jordan, and Samuel Madden. 2014. Scaling up crowd-sourcing to very large datasets: a case for active learning. In *Proc. Very Large Databases (VLDB) Conference*.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Andrew Y. Ng and Michael I. Jordan. 2001. On discriminative vs. generative classifiers: A comparison of logistic regression and Naive Bayes. In *Proc. Advances in Neural Information Processing Systems (NIPS)*.

Thien Huu Nguyen and Ralph Grishman. 2014. Employing word representations and regularization for domain adaptation of relation extraction. In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*.

Rebecca J. Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics*, 2:311–326.

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

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *The Journal of Machine Learning Research*, 11:1297–1322.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proc. LREC Workshop on New Challenges for NLP Frameworks*.

Edwin Simpson and Stephen Roberts. 2015. Bayesian methods for intelligent task assignment in crowdsourcing systems. In *Decision Making: Uncertainty, Imperfection, Deliberation and Scalability*, pages 1–32. Springer.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Vivek Srikumar and Christopher D. Manning. 2014. Learning distributed representations for structured output prediction. In *Advances in Neural Information Processing Systems 27*, pages 3266–3274.

James Surowiecki. 2005. *The Wisdom of Crowds*. Random House LLC.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*.

Peter Welinder, Steve Branson, Pietro Perona, and Serge J. Belongie. 2010. The multidimensional wisdom of crowds. In *Proc. Advances in Neural Information Processing Systems (NIPS)*.

Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L. Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Proc. Advances in Neural Information Processing Systems (NIPS)*.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B. Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from Twitter. *Transactions of the Association for Computational Linguistics*, 2:435–448.

Yan Yan, Rómer Rosales, Glenn Fung, Ramanathan Subramanian, and Jennifer Dy. 2014. Learning from multiple annotators with varying expertise. *Machine Learning*, 95(3):291–327.

Interactive-Predictive Machine Translation based on Syntactic Constraints of Prefix

Na Ye, Guiping Zhang and **Dongfeng Cai**
Human-Computer Intelligence Research Center
Shenyang Aerospace University, Shenyang 110136, China
yn.yena@hotmail.com, zgp@ge-soft.com, caidf@vip.163.com

Abstract

Interactive-predictive machine translation (IPMT) is a translation mode which combines machine translation technology and human behaviours. In the IPMT system, the utilization of the prefix greatly affects the interaction efficiency. However, state-of-the-art methods filter translation hypotheses mainly according to their matching results with the prefix on character level, and the advantage of the prefix is not fully developed. Focusing on this problem, this paper mines the deep constraints of prefix on syntactic level to improve the performance of IPMT systems. Two syntactic subtree matching rules based on phrase structure grammar are proposed to filter the translation hypotheses more strictly. Experimental results on LDC Chinese-English corpora show that the proposed method outperforms state-of-the-art phrase-based IPMT system while keeping comparable decoding speed.

1 Introduction

In recent years, the machine translation (MT) technology has achieved great progress. However, up till now the MT output still cannot meet the practical requirements on translation quality in many scenarios, and need to be post-edited by human translators before actually put to use. In order to help MT systems collaborate with human translators more effectively, researchers carried out the study on computer-assisted translation (CAT), in which the main goal of MT is supporting professional translators in improving their productivity. Therefore, the ability to interact with human becomes an important research topic in CAT.

The first interactive machine translation systems (Kay and Martins, 1973; Zajac, 1988; Yamron et al., 1993) focus on having human translators disambiguate the source texts through answering questions. However, this question-answering process remains a laborious one for human translators. Under such circumstances, the interactive-predictive machine translation (IPMT) method is proposed (Foster et al., 1997). In the IPMT mode, first the system generates one or more raw suggestions, and then the human translator validates the longest correct prefix in the suggestions and revises the first character in the corresponding suffix, next the new prefix is used to help the system predict the optimal suffix. This process is repeated until the correct translation is acquired. The IPMT technology enables human translators to avoid the burden of explaining the source text, and directly control the final translation generation, so it attracted widespread attention.

It can be seen that the essential difference between IPMT and MT is the introduction of a constraint, namely the target sentence must start with the human validated prefix. To achieve this goal, researchers attempted various MT models (Och et al., 2003; Civera et al., 2004; Tomás and Casacuberta, 2006; Barrachina et al., 2009; González-Rubio et al., 2013). In these methods, prefix is mainly used for performing character-level matching on translation hypotheses to reduce the search space. However, the hypotheses that only match the prefix on character level are perhaps not correct. Figure 1 gives an ex-

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1797–1806, Osaka, Japan, December 11-17 2016.

ample.

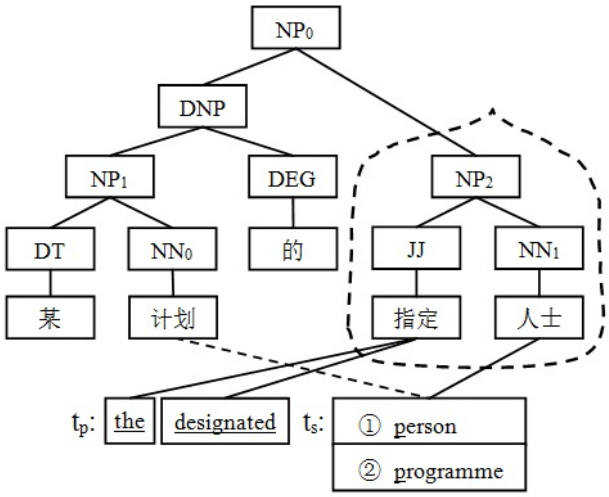


Figure 1: Example of translation hypothesis selection on character level.

In Figure 1, the validated prefix is "the designated p". Two hypotheses that start with "the designated person" and "the designated programme" both meet the requirement. If we use "person" to extend the prefix (see the alignment by solid line), then the hypothesis will form a complete subtree translation (circled by dashed line). But if we use "programme" to extend the prefix, then an error will occur. The circled subtree has only been partly translated before the hypothesis turns to translate another subtree (see the alignment by dashed line). In the future interactions, no matter how to extend the hypothesis, no reasonable syntactic alignment will be generated. Therefore, other than character-level matching results with the prefix, we should also take syntactic-level matching results into account for hypothesis selection. Only when a reasonable syntactic alignment is formed between the hypothesis and the source sentence, can we decide this is a good hypothesis.

In this paper we present a mathematical IPMT framework based on the syntactic alignment between the source sentence and the prefix. On the basis of the framework, we proposed two syntactic subtree constraints based on phrase structure grammar for selecting translation hypotheses. Experimental results on LDC corpora show that our method reduced the human-computer interaction times under comparable decoding speed.

2 Related Work

The task of IPMT (Barrachina et al., 2009) is to find the optimal suffix under the condition of a given source sentence s and a validated prefix t_p:

$$\hat{t}_s = \underset{t_s}{\operatorname{argmax}}\ P(t_s \mid s, t_p) = \underset{t_s}{\operatorname{argmax}}\ P(t_p, t_s \mid s) \quad (1)$$

where $(t_p, t_s)=t$, indicating that the prefix t_p and the predicted suffix t_s concatenate to form a complete translation t.

As previously discussed, when modelling $P(t_p, t_s \mid s)$, current methods mainly make use of the character-level matching results with the prefix. To enhance the guiding effect of the prefix, some work also considered other factors.

Sanchis-Trilles et al. (2008) integrates the user's mouse actions into the IPMT system. The method is based on an assumption that the first character of the predicted suffix must be different from the current suffix when the user clicks the mouse on the translation. In this way, the clues hidden in the prefix are further exploited. However, this method still restricted to character or word level matching.

Some researchers investigated the word alignment between the prefix and the source sentence. Nepveu et al. (2004) proposed a cache-based adaptive prediction model. For the predicted translation w of each active word a, once the user accepts w, a word pair (a, w) will be stored in the cache. Higher language model probability and translation probability will be assigned to w if a new source sentence

contains the word *a*. Ortiz-Martínez et al. (2009) performs word alignment between the prefix and the source sentence. The generation of suffix is limited to the translation of the unaligned parts in the source sentence. These methods deepened the prefix matching level, but still did not reach the syntactic level.

González-Rubio et al. (2013) adopts hierarchical phrase-based model (HPBM) to IPMT. Because HPBMs are on the basis of synchronous grammar, the prefix can guide decoding on a deeper level. However, the synchronous grammar in hierarchical phrases has no linguistic meaning, and cannot evaluate the reasonableness of the hypotheses from the view of syntactic structure.

Compared with the above research, our method made use of the prefix on syntactic level. The hypotheses for which it is impossible to generate reasonable syntactic structure alignment with the source sentence are identified and filtered.

3 IPMT Mathematical Model

In this paper, we examine the syntactic alignment between the source sentence s and t_p when we make use of the prefix. A hidden variable $T(s)$ is introduced to represent the parse tree of s. Consequently, $P(t_p, t_s \mid s)$ is converted to:

$$P(t_p, t_s \mid s) = \sum_{T(s)} P(t_p, t_s, T(s) \mid s) \tag{2}$$

The item on the right side is further deduced with:

$$P(t_p, t_s, T(s) \mid s) = P(T(s) \mid s) \times P(t_p \mid T(s), s) \times P(t_s \mid t_p, T(s), s) \tag{3}$$

The first factor on the right side is the syntactic parsing model of the source language, which is provided by the parser. The second factor corresponds to the transformation from the source sentence to the prefix, which is the machine translation model. So the two models need not be discussed. The third factor is the key model of this paper, which corresponds to the prediction of suffix t_s. We will emphasize on the modelling of this factor.

There are two ways to model $P(t_s \mid t_p, T(s), s)$. One is completely adopting the syntax-based MT framework and performing prefix matching during decoding. The other is adding syntactic information into the PBM-based SMT framework as rules. In comparison, the former way is more straightforward, but it is prone to be influenced by the performance of parsing and translation rule extraction algorithms. Incorrect parse tree of the source sentence and incorrect translation rules can both lead to the consequence that the prediction will never succeed. Although some researchers proposed forest-based translation approaches (Mi and Huang, 2008; Zhang et al., 2009) to avoid relying on 1-best parse tree, the computation costs of these approaches are too high for the IPMT systems which have strict speed requirements. The latter way allows the existence of non-syntactic phrases and has larger search space. To alleviate the negative effect of wrong parsing results, we can filter the hypotheses only when there is more than one candidate. In other words, the syntactic structure can play the role as soft constraints if we adopt relatively tolerant syntactic tree matching rules. Once the prefix of a hypothesis has the possibility of being correctly aligned to the source-language parse tree, the hypothesis will be kept. In this way, the system can be more error-tolerant. Therefore, we built the model in the latter way and adopted the phrase-based model (PBM) as in (Barrachina et al., 2009).

We introduce a hidden variable A to represent the phrase alignment between t_p and $T(s)$. The third factor of Equation 3 can be transformed to:

$$P(t_s \mid t_p, T(s), s) = \sum_{A} P(t_s, A \mid t_p, T(s), s) \tag{4}$$

This paper estimates the factor on the right side as follows:

$$P(t_s, A \mid t_p, T(s), s) = \begin{cases} P(t_s \mid t_p, s) & \text{if } \xi(t_p, A, T(s)) = \text{TRUE} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $\xi(t_p, A, T(s))$ is a function to judge whether the alignment A conforms to the syntactic tree matching rules. The goal is to decide whether there is possibly correct syntactic alignment between the prefix and the source sentence.

4 Hypothesis Selection

Generally, syntactic structure can be represented or labelled by two forms. One is the phrase structure grammar (PSG), and the other is the dependency structure grammar (DSG). In this paper we choose PSG for the extension of prefix. There are two reasons: first, PSG can clearly give the syntactic component borders (subtrees) with complete linguistic meaning; second, the PSG parsers mainly use probabilistic context-free grammar (PCFG), and the produced N-gram translation rules can well model the orders of the syntactic components. Such information can straightforwardly direct the extension of translation hypotheses. But the dependency grammar describes the binary head-modifier structure, so it is inefficient for the PBM model which takes multi-word phrases as the basic processing units.

In order to use the syntactic structure of the source sentence to guide the prefix extension, we analysed the nature of the prefix and propose two subtree matching rules (or constraints) which should be followed while selecting translation hypotheses.

4.1 Complete Subtree Constraint

This constraint requires that the selection of the phrase to extend the prefix needs to consider whether the current subtree[1] has been completely translated. Figure 2 gives an example.

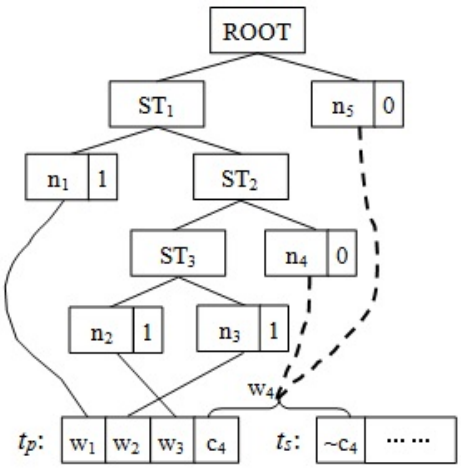


Figure 2: Translation hypothesis extension based on complete subtree constraint.

In Figure 2, the prefix t_p consists of complete words w_1, w_2, w_3 and an incomplete word c_4. The suffix t_s starts with an incomplete word $\sim c_4$. c_4 and $\sim c_4$ together form a complete word w_4. The data structure of node n_i records the information whether the word has already been translated (1 for *yes*, 0 for *no*). In this example, n_1, n_2 and n_3 are translated nodes (the word alignments are indicated by solid lines), n_4 and n_5 are untranslated nodes. If the translations of n_4 and n_5 can both be used to extend w_3 (indicated by dashed lines), then we need to examine the subtree (ST_1) to which n_2 belongs. Since there is still a node n_4 left untranslated in the subtree, n_4 should be chosen to perform extension.

The core of the complete subtree constraint is to judge whether the subtree to which the phrase to be extended belongs has been completely translated. Since a subtree may be nested in another subtree, it is possible that the same phrase is contained in multiple subtrees. In this paper we select the subtree with the minimum span to constrain the hypothesis extension.

After deciding the subtree ST to constrain hypothesis extension, the phrase p_1 which is to be extended and the phrase p_2 which is to extend p_1 are examined. Through checking the positions of the words covered by p_1 in the source sentence, the words not covered by subtree ST can be found. Because there

[1] In this paper, since it is meaningless to examine nodes under the complete tree S, the term "subtree" refers to proper subtree and does not include the complete tree.

are re-orderings in the translation, the positions of these words are perhaps not continuous. Therefore, the translation hypotheses should meet the following requirements: a) the positions of the source words covered by p_2 are fully contained in the span of the subtree; or b) the positions of the source words covered by p_2 are continuous and are located at the tail of the source words covered by the subtree, and the source words covered by p_1 fully cover the remained positions of the subtree.

4.2 Subtree Order Constraint

This constraint requires that the selection of the subtree to be extended (translated) needs to consider the overall structure of the syntactic tree and the re-ordering rules. Figure 3 gives an example.

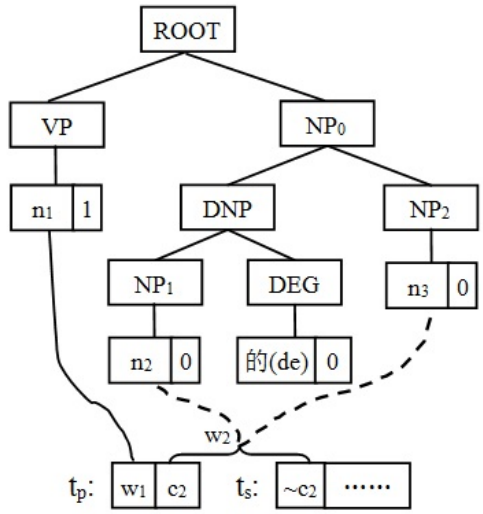


Figure 3: Translation hypothesis extension based on subtree order constraint.

In Figure 3, the prefix consists of a complete word w_1 and an incomplete word c_2. The suffix t_s starts with an incomplete word $\sim c_2$. If the translations of n_2 and n_3 can both be used to extend w_1, then we need to examine the subtrees to which n_2 and n_3 belongs. This example possesses an "NP_1 *de* NP_2" structure. According to the re-ordering rules, it should be translated into "NP_2 *of* NP_1". This means that we should choose the subtree to which n_3 belongs to perform extension.

We use the NiuTrans (Xiao et al., 2012) system to generate tree-to-string translation rules to evaluate whether the order of the subtrees are reasonable. First perform syntactic parsing on the source sentence and get the parse tree T; then identify all the subtrees ST_i in T and record the child nodes LC_i (left child) and RC_i (right child) of the root node of each subtree; next use these subtree structures to filter the candidate translation rule set RT that can be used by the current sentence. During decoding, the hypothesis with highest score that matches the translation rules is selected for extension.

The key of this constraint is judging whether the phrase p_1 to be extended and the phrase p_2 to extend p_1 are closely adjacent in the translation rule. To achieve this goal, we need to identify the minimum common subtree MCT of the source phrases corresponded to p_1 and p_2, and then use the translation rules that match this subtree to perform judging.

Since the translation rule base cannot fully cover all the syntactic structure instances, this paper generalized the subtree to acquire the rules that nearly match. If the needed rule is not in the rule base, then the subtrees in MCT will be generalized to their parent nodes from bottom to up. As long as any translation rule can match a generalized subtree, the new parent node will replace the child nodes for order judgement.

4.3 Balancing Strategies

During decoding, the two constraints go forward one by one. First judge whether the two phrases conform to the complete subtree constraint. If so, continue to judge whether they conform to the subtree order constraint. In any of the above two stages, once there is no hypothesis that matches the constraint, the algorithm will go back and accept the result of the previous stage.

However, there exist many mistakes in the parsing results. Therefore, strictly following the subtree constraints according to the parsing results will lead to the loss of some good hypotheses. To achieve balance between analysis depth and searching precision, we adopt the following strategies:

(1) If the hypotheses to extend the current translation contain the candidates that meet the constraints, then searching is limited within the scope of these candidates, otherwise we still search within all the hypotheses. This strategy can better take advantage of the non-syntactic phrases in the PBM-based SMT models.

(2) To reduce the complexity, we do not distinguish the border word of the prefix, but take phrases as the basic processing unit and examine whether the phrase to be extended and the phrase to extend it conform to the subtree constraints.

(3) We do not require that every hypothesis extension conforms to the two constraints. Considering the characteristics of IPMT task, we propose three strategies. a) only consider the subtree constraints when the hypothesis to be extended has not fully covered the prefix; b) only consider the subtree constraints when the hypothesis to be extended just covers the prefix; c) consider the subtree constraints under both the above two situations.

5 Experimental Results

5.1 Data Setup

This paper uses the Chinese-English Hong Kong Laws Parallel Text (LDC2000T47) as the corpora. 200,000 sentence pairs are taken as the training set. 1000 sentence pairs are randomly extracted from the remaining corpora as the development set, and 1558 sentence pairs as the testing set. Table 1 gives a detailed description of the corpora.

Corpus		Chinese	English
Training Set	Sentences	200K	200K
	Words	5.15M	5.11M
	Vocabulary	30K	31K
Development Set	Sentences	1000	1000
	Words	15K	15.7K
	Vocabulary	73.24	48.65
Testing Set	Sentences	1558	1558
	Words	20.6K	21.6K
	Perplexity	72.67	46.75

Table 1: Statistics of the Evaluation Corpora.

The Chinese portions of these data were pre-processed by ICTCLAS word segmenter[2], and the English portions were tokenized and lowercased. GIZA++ tool was used to perform the bi-directional word alignment of the training data, and the "grow-diag-final" strategy was used to merge the bi-directional results. A 3-gram language model was trained on the English portion of the training corpus with SRILM. For the building of PB SMT models, Moses (Koehn et al., 2007) was used, and the model includes 14 default features. For adjusting feature weights, the MERT (Och, 2003) method was applied, optimizing the BLEU-4 metric obtained on the development corpus. The parse trees were produced by the Berkeley parser, and 1-best tree was used for subtree extraction. During decoding, the size of hypothesis stack is set to 30, and the maximum number of translation options is set to 20 for each source phrase.

We used Key-stroke Ratio (Barrachina et al., 2009) to evaluate the performance of the IPMT systems. The lower the KSR score, the better the system performance. The baseline system is the state-of-the-art PBM-based IPMT approach using multi-stack-decoding algorithm as described in (Barrachina et al., 2009). We follow the user simulation approach for evaluation as previous works in the literature (Barrachina et al., 2009; González-Rubio et al., 2013). Statistical significance test is conducted using the resampling method proposed in (Koehn, 2004; Zhang et al., 2004).

[2] http://ictclas.nlpir.org/

5.2 Results and Analysis

Table 2 gives the comparative experimental results after using the complete subtree constraint (the distortion distance is limited to 10). ST_CM_BCP represents considering the rule when the hypothesis to be extended has not fully covered the prefix, ST_CM_ACP represents considering the rule when the hypothesis to be extended just covers the prefix, and ST_CM represents considering the rule under both the above two situations. We evaluated the systems on different *K*-best lists. The best results are displayed in bold fonts and have a statistically significant difference with respect to the baseline (95% confidence).

Method	1-best	5-best	10-best	20-best
baseline	48.66	47.93	47.76	47.55
ST_CM_BCP	48.05	47.41	47.18	46.97
ST_CM_ACP	48.44	47.75	47.48	47.21
ST_CM	**47.85**	**47.16**	**46.88**	**46.69**

Table 2: KSR scores after using the complete subtree constraint.

From Table 2 we can see that adding source-language syntactic structure constraint can effectively reduce the interaction times. And using the complete subtree constraint in the whole process of prefix generation (ST_CM) achieved more improvement than only using it in a certain stage (ST_CM_BCP and ST_CM_ACP). The reason is that the prefix is a fragment validated by the human, and it can guide the hypothesis extension at any stage. The following experiments in the paper are all based on this setting (consider the rules under both situations). With the increase of the number of translations, the complete subtree constraint also plays a positive role. On the testing corpus, the underling MT engine achieves a BLEU score of 0.2678.

Table 3 gives the KSR scores after adding the subtree order constraint. Experiments are conducted under different distortion distances. ST_CM represents only using the complete subtree constraint, ST_RD represents only using the subtree order constraint, and ST_IMT represents using both constraints.

Method	Distortion distance limitation = 10				Distortion distance limitation = 15			
	1-best	5-best	10-best	20-best	1-best	5-best	10-best	20-best
baseline	48.66	47.93	47.76	47.55	47.61	46.53	46.29	46.09
ST_CM	47.85	47.16	46.88	46.69	47.18	46.11	45.86	45.68
ST_RD	48.21	47.58	47.13	46.94	47.39	46.32	46.05	45.89
ST_IMT	**47.74**	**47.07**	**46.73**	**46.41**	**46.88**	**45.83**	**45.59**	**45.41**

Table 3: KSR scores after using the subtree order constraint.

Table 3 shows that after using the subtree order constraint, the interaction times decrease. But the improvement is not as obvious as the complete subtree constraint. The reason is that the translation rules consider the inner structure of the subtrees and are more fine-grained. So the matching difficulty increases. It also can be seen that using both constraints leads to larger improvement in performance, indicating that they are complementary constraints which guide the hypothesis selection from different aspects. These results verify that the user-validated prefixes are effective on multiple levels. In fact, when a user gives a prefix, he/she is making a comprehensive decision after analysing the overall structure and orders of the translation. And the proposed method exploits the syntactic structure information hidden in the prefix. In addition, although there are parsing mistakes, we can still get positive results; this shows that the method is error-tolerant.

Through comparing the IPMT process on specific sentences, we find that the decoding of many sentences will fail in the baseline system. In other words, there is no hypothesis that can match the prefix in the stack. However, after adding the subtree constraints, some hypotheses that do not have reasonable syntactic structures will be filtered during extension, thus some lower-ranked hypotheses have the opportunities to be pushed into the stack and finally match the prefix. In such cases, the efficiency of human-computer interaction greatly improves. In our experiment, 1.8% sentences which cannot find

the correct translation with the baseline method succeeded in finding the correct translation with the new method.

Table 4 shows the decoding speeds of different systems when using 1-best result for user reference (the distortion distance is limited to 10). The speed is the number of sentences decoded per second on the testing corpus. The hardware setting is 4G memory, 500G hard disk, Core i5 3.2GHz processor.

Method	Speed (sent/sec)
baseline	3.43
ST_CM_BCP	3.07
ST_CM_ACP	3.23
ST_CM	2.86
ST_RD	2.94
ST_IMT	2.78

Table 4: Decoding speed using different subtree constraints.

We can see that incorporating syntactic information did not lead to much decrease in the predicting/responding speed. This is because the subtree constraints help filtering some bad hypotheses and reduced the search space. This balanced the time cost in rule matching. We should note that in our experiments the testing corpus is parsed in advance. In practice, if the corpus cannot be acquired in advance, then each source sentence should be input to the IPMT procedure after parsing.

5.3 Comparison with Other Work

Some researchers proposed methods (Quirk et al., 2005; Marton and Resnik, 2008; Shen et al., 2008; Gao et al., 2011) that introduce the syntactic information to the phrase-based MT system on the basis of hierarchical phrase-based models. In these methods, hierarchical phrases rather than flat phrases are employed. However, the decoding is not in a left-to-right manner, and has difficulty in applying to the prefix matching process of IPMT systems.

The work more similar with ours are those in (Collins et al., 2005; Wang et al., 2007; Galley and Manning, 2008; Hunter and Resnik, 2010). Galley and Manning (2008) adopts a left-to-right shift-reduce method to build hierarchical structures for flat phrases. But this method is "formally syntactic-based" rather than "linguistically syntactic-based". Collins et al. (2005) and Wang et al. (2007) perform pre-ordering in the pre-processing stage instead of deciding the phrase orders in the decoding stage. This strategy may remove the translation hypotheses that match the prefix too early. Hunter and Resnik (2010) directly introduce the source-language syntactic constraints into the decoding of phrase-based MT system, which are almost the same as our work. However, this method builds an independent syntactic re-ordering model and scores the hypotheses through features. In fact, in the IPMT systems the user clearly gives the correct prefix, which can act as an explicit constraint that the hypotheses must match. So it is more suitable to use the syntactic constraints in the form of rules.

6 Conclusion

This paper deepened the constraints of prefix in state-of-the-art PBM-based IPMT system. We built the mathematical framework of IPMT model based on the syntactic alignment between the source sentence and the prefix. On the basis of the framework we proposed a method that introduces source-language syntactic information to hypothesis selection in the form of soft constraints, evaluating the hypotheses from the aspects of subtree completion and subtree order. We also proposed the strategies to avoid the matching rules from being too strict, making the system tolerant to the parsing mistakes. Experimental results proved that our method reduced the human-computer interaction times while the decoding speed does not decrease obviously.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 61402299). We would like to thank the anonymous reviewers for their insightful and constructive comments. We also want to thank Yapeng Zhang for help in the preparation of experimental systems in this paper.

References

Sergio Barrachina, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, and Juan-Miguel Vilar. 2009. Statistical Approaches to Computer-Assisted Translation. *Computational Linguistics*, 35(1):3–28.

Jorge Civera, Elsa Cubel, Antonio L. Lagarda, David Picó, Jorge González, Enrique Vidal, Francisco Casacuberta, Juan M. Vilar, and Sergio Barrachina. 2004. From Machine Translation to Computer Assisted Translation using Finite-State Models. In *Proceedings of the EMNLP*, pages 349–356.

Michael Collins, Philipp Koehn, and Ivona Kucerov. 2005. Clause Restructuring for Statistical Machine Translation. In *Proceedings of the ACL*, pages 531–540.

George Foster, Pierre Isabelle, and Pierre Plamondon. 1997. Target-text Mediated Interactive Machine Translation. *Machine Translation*, 12(1):175–194.

Michel Galley and Christopher D. Manning. 2008. A Simple and Effective Hierarchical Phrase Reordering Model. In *Proceedings of the EMNLP*, pages 848–856.

Yang Gao, Philipp Koehn, and Alexandra Birch. 2011. Soft Dependency Constraints for Reordering in Hierarchical Phrase-based Translation. In *Proceedings of the EMNLP*, pages 857–868.

Jesús González-Rubio, Daniel Ortiz-Martínez, José-Miguel Benedí, and Francisco Casacuberta. 2013. Interactive Machine Translation using Hierarchical Translation Models. In *Proceedings of the EMNLP*, pages 244–254.

Tim Hunter and Philip Resnik. 2010. Exploiting Syntactic Relationships in a Phrase-based Decoder: An Exploration. *Machine Translation*, 24(2):123–140.

Martin Kay and Gary R. Martins. 1973. The MIND System. *Natural Language Processing*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the ACL*, pages 177–180.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the EMNLP*, pages 388–395.

Yuval Marton and Philip Resnik. 2008. Soft Syntactic Constraints for Hierarchical Phrased-based Translation. In *Proceedings of the ACL*, pages 1003–1011.

Haitao Mi and Liang Huang. 2008. Forest-based Translation Rule Extraction. In *Proceedings of the EMNLP*, pages 206–214.

Laurent Nepveu, Guy Lapalme, Philippe Langlais, and George Foster. 2004. Adaptive Language and Translation Models for Interactive Machine Translation. In *Proceedings of the EMNLP*, pages 190–197.

Franz Josef Och, Richard Zens, and Hermann Ney. 2003. Efficient Search for Interactive Statistical Machine Translation. In *Proceedings of the EACL*, pages 287–293.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the ACL*, pages 160–167.

Daniel Ortiz-Martínez, Ismael García-Varea, and Francisco Casacuberta. 2009. Interactive Machine Translation based on Partial Statistical Phrase-based Alignments. In *Proceedings of the RANLP*, pages 330–336.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency Treelet Translation: Syntactically-informed Phrasal SMT. In *Proceedings of the ACL*, pages 271–279.

German Sanchis-Trilles, Daniel Ortiz-Martínez, and Jorge Civera. 2008. Improving Interactive Machine Translation via Mouse Actions. In *Proceedings of the EMNLP*, pages 485–494.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model. In *Proceedings of the ACL*, pages 577–585.

Jesús Tomás and Francisco Casacuberta. 2006. Statistical Phrase-based Models for Interactive Computer-Assisted Translation. In *Proceedings of the COLING/ACL*, pages 835–841.

Chao Wang, Michael Collins, and Philipp Koehn. 2007. Chinese Syntactic Reordering for Statistical Machine Translation. In *Proceedings of the EMNLP*, pages 737–745.

Tong Xiao, Jingbo Zhu, Hao Zhang, and Qiang Li. 2012. Niutrans: An Open Source Toolkit for Phrase-based and Syntax-based Machine Translation. In *Proceedings of the ACL*, pages 19–24.

Jonathan Yamron, James Baker, Paul Bamberg, Haakon Chevalier, Taiko Dietzel, John Elder, Frank Kampmann, Mark Mandel, Linda Manganaro, Todd Margolis, and Elizabeth Steele. 1993. Lingstat: An Interactive, Machine-aided Translation System. In *Proceedings of the workshop on Human Language Technology*, pages 191–195.

Remi Zajac. 1988. Interactive Translation: A New Approach. In *Proceedings of the COLING*, pages 785–790.

Ying Zhang, Stephan Vogel, and Alex Waibel. 2004. Interpreting BLEU/NIST Scores: How Much Improvement Do We Need to Have a Better System? In *Proceedings of the LREC*, pages 2051–2054.

Hui Zhang, Min Zhang, Haizhou Li, Aiti Aw, and Chew Lim Tan. 2009. Forest-based Tree Sequence to String Translation Model. In *Proceedings of the ACL*, pages 172–180.

Topic-Informed Neural Machine Translation

Jian Zhang, Liangyou Li, Andy Way, Qun Liu
ADAPT Centre
School of Computing
Dublin City University, Ireland
jian.zhang,liangyou.li,andy.way,qun.liu@adaptcentre.ie

Abstract

In recent years, neural machine translation (NMT) has demonstrated state-of-the-art machine translation (MT) performance. It is a new approach to MT, which tries to learn a set of parameters to maximize the conditional probability of target sentences given source sentences. In this paper, we present a novel approach to improve the translation performance in NMT by conveying topic knowledge during translation. The proposed *topic-informed* NMT can increase the likelihood of selecting words from the same topic and domain for translation. Experimentally, we demonstrate that topic-informed NMT can achieve a 1.15 (3.3% relative) and 1.67 (5.4% relative) absolute improvement in BLEU score on the Chinese-to-English language pair using NIST 2004 and 2005 test sets, respectively, compared to NMT without topic information.

1 Introduction

In statistical machine translation (SMT) (Och and Ney, 2000; Marcu and Wong, 2002; Koehn et al., 2007), several models are trained separately and then linearly integrated using the log-linear model (Och, 2003). Neural machine translation (NMT) (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015), being a new approach, employs an individual large neural network to maximize the translation probability directly.

One observation we obtain from machine translation (MT) training data is that some of the words within the same sentence often belong to the same or similar topic. Examine the example presented in Figure 1, where words *Commercial*, *market*, *prices* and *bank* have higher probabilities of being in the *Financial* domain. Intuitively, by allowing the topic information of the source input sentence and previous translated words to be provided to the decoder, we can maintain the same topic in the translations during the decoding phase, and consequently better translations can be produced. Specifically, when a source word has more than one translation option available (e.g. the word *bank* in Figure 1), it is clear that the translation in the *Financial* domain is more likely to be selected because many of the source words are from the same topic.

Topic models have been applied successfully in many SMT works. For example, similar "topic consistent" behaviour is also observed by Su (2015). In his work, a context-aware topic model is integrated into SMT for better lexical selection. Xiao et al. (2012) and Zhang et al. (2014) focus on document translations and propose a topic-similarity model and a topic-sensitivity model for hierarchical phrase-based SMT (Chiang, 2005) on the document level. The topic-similarity model is used to encourage or penalize topic-sensitive rules, and the topic-sensitivity model is applied to balance topic-insensitive rules. However, these approaches cannot be directly used in NMT.

In this work, we propose our novel topic-informed NMT model. In topic models, word-topic distributions can be viewed as a vector. In NMT, words are represented as vector-space representations. Thus, it is very natural to use them together. Word expressions in neural models are derived by its near context words while the topic vectors represent the document-level topic information. For this reason, we see

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1807–1817, Osaka, Japan, December 11-17 2016.

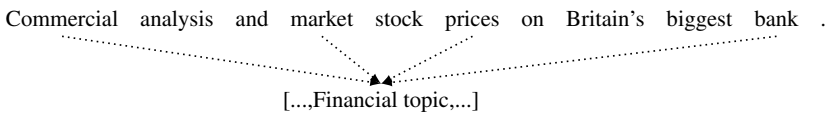


Figure 1: Some words within the same sentence belong to the same topic.

them as complementary to one another. Our intuition is that incorporating topic information will benefit NMT. Thus, we explore incorporating topic information into NMT in either/both the source side and target side.

The remainder of this paper is structured as follows. Section 2 presents related work. Section 3 gives review of the NMT architecture we use. Section 4 introduces our topic-informed NMT system, while Section 5 presents experimental results and some observations. Finally, the conclusions and future directions are provided in Section 6.

2 Related Work

Topic modelling has been applied successfully in many SMT works, especially in the domain adaptation literature. The motivation for introducing topic-model information in MT is that the translation performance decreases when there are dissimilarities between the training and the testing domains. A better approach is to make the use of the topic knowledge learned during training. Such knowledge can yield a better word or phrase choice in translation. Early work shows that the lexical translation table conditioned on the provenance of each domain can significantly improve translation quality (Chiang et al., 2011). Eidelman (2012) extends the provenance idea by including topic-dependent lexical weighting probabilities on the source side. Hasler (2012) successfully combines sparse word-pair and phrase-pair features with topic models. Later, Hasler (2014) also reports that translation performance is improved by using similarity features, which are computed from training phrase-pair and test-context vectors generated from the phrase-pair topic model. However, these ideas cannot be directly applied in NMT given the different training algorithms used in SMT and NMT.

Despite the fact that neural network training has been shown to be advantageous in many natural language processing tasks, little work has been proposed on using additional knowledge in NMT. Gulcehre et al. (2015) leverage monolingual corpora for NMT and propose two approaches to integrate a neural language model into the encoder-decoder architecture. He et al. (2016) integrate SMT features into NMT in order to solve the out-of-vocabulary problem. Furthermore, they use a n-gram language model trained on large monolingual data to enhance the local fluency of translation outputs. Linguistic information can also be used during NMT training (Sennrich and Haddow, 2016; García-Martínez et al., 2016). There are also works that include topic modelling in neural language model training. Mikolov and Zweig (2012) use a contextual real-valued input vector associated with each word in a recurrent neural network (RNN) language model.

3 Neural Machine Translation

3.1 Encoder-Decoder Architecture

In a nutshell, the fundamental job of the encoder-decoder architecture (Cho et al., 2014) in NMT is to probabilistically decode a target sequence given the encoded source sequence, where the two sequences can be of different lengths. Given a sentence pair (S, T), S is the foreign input sentence and T is the translation, where $S = (s_1, s_2, ..., s_{m-1}, s_m)$ and $T = (t_1, t_2, ..., t_{n-1}, t_n)$ are the words in the sentence pair. The encoder-decoder architecture needs to find the translation probabilities for each word in T, as in Equation (1):

$$p(T|S) = \sum_{j=1}^{n} p(t_j|t_{1:j-1}, S) \tag{1}$$

and the conditional probability is given by the decoder, which uses the $softmax$ function outputting the probability distribution on all words in the target, as in Equation (2):

$$p(t_j = t|t_{1:j-1}, S) = softmax(f(h_j)) \tag{2}$$

where f is a function that can transform the target context h_j into a vector, which has the same size with the size of target vocabulary. h_j is defined in Equation (3):

$$h_j = g(t_{j-1}, h_{j-1}, c) \tag{3}$$

where c is the source context vector computed by the encoder, t_{j-1} is the word vector representation of word $j-1$, h_{j-1} is the hidden state for time $j-1$ and g is a non-linear activation function. Thus, we use the source input sentence and previous translated words to predict the next word.

3.2 Attention Model

The encoder-decoder architecture uses a fixed-sized vector to represent the whole source input. Although RNNs are known to be better at capturing long range dependencies, experimental results (Bahdanau et al., 2015) show that translation quality decreases for long input sentences. Bahdanau (2015) use an attention mechanism to learn dynamic *soft-alignment* during the network training, as in Equation (4):

$$e_{ij} = a(h_{j-1}, h_i) \tag{4}$$

where e_{ij} is the alignment model to score the alignment at position i and j in S and T, respectively. h_{j-1} is the target hidden state as seen in Equation (3), and h_i is the source hidden state at time i.

Thus, a distinct context vector c_j can be computed for each word in T, and the source context vector c is rewritten as in (5):

$$c_j = \sum_{i=1}^{m} \alpha_{ij} h_i \tag{5}$$

where α_{ij} is a normalized weight for each hidden state in S, computed as in (6):

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{i=1}^{m} exp(e_{ij})} \tag{6}$$

With the attentional model, source information can be spread across the source context vector, and the decoder can selectively pay attention to different parts of the source context during decoding.

3.3 Bidirectional RNN

During the encoding phase, words can also be fed into the encoder in both directions, in which case we are using a bidirectional RNN. The intuition behind such a model is to include both *positive* and *negative* time stamps of the source input during encoding, which can shorten the distance between the decoding part with the relevant encoded part. Sutskever et al. (2014) claim that it is "extremely valuable" and can "greatly boost the performance" by using a bidirectional RNN in NMT.

In this paper, following Cho et al. (2014) and Bahdanau et al. (2015), we use the encoder-decoder architecture with a single layer bidirectional gated recurrent unit (GRU) (Chung et al., 2014) as the encoder, and a single layer GRU with attention model as the decoder in our NMT system.[1]

4 Topic-Informed Neural Machine Translation

In this section, we provide an explanation of how to include topic knowledge in NMT.

[1]More hidden layers usually result in much better quality. However, the training time increases. Thus, we use only a single layer in the encoder or the decoder.

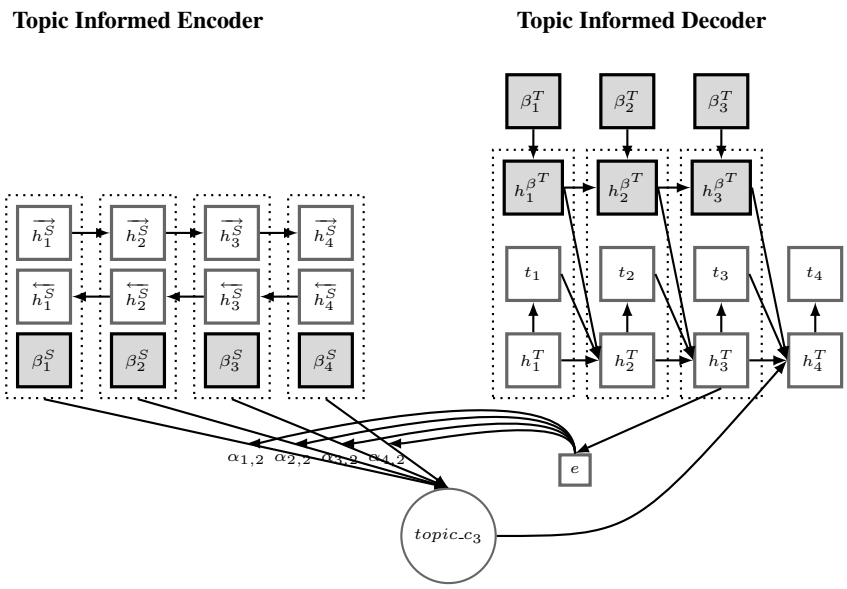


Figure 2: A graphical illustration of the proposed topic-informed NMT model, with 4 input words and 4 output words. The shaded units indicate the topic information used in the encoder and decoder. In order to distinguish the hidden states in the encoder-decoder architecture, we use h^S and h^T to represent the hidden states of the encoder and decoder, respectively, e.g. h_1^S indicates the encoder hidden state at time 1. Furthermore, in order to keep the notation consistent with Section 4.1, e.g. β_i^S denotes the word distributions over topics for the source word at position i, a similar notation is also used for the target words. For example, β_1^T indicates the word distributions over topics for the target word at position 1.

4.1 Topic-Informed Encoder

The encoder in standard NMT uses only word embedding to compute the source context vector. By using topic information on the source side, the decoder can have an overview of the source topics during decoding. Furthermore, the attention model can implicitly pay attention to the topic distributions of each source word. Topic information on the source side can be helpful to generate more accurate translations. Therefore, we first compute the topic distributions (word distributions over topics) for each source word of the input sentence. We then concatenate the topic distributions with each corresponding hidden state of the input source words. Finally, the hidden states with the topic information are used to compute the topic-informed source context vector $topic\_c_j$, as in Equation (7):

$$topic\_c_j = \sum_{i=1}^{m} \alpha_{ij}[h_i, \beta_i^S] \tag{7}$$

to obtain our topic-informed encoder, where β_i^S denotes the word distributions over topics for each source word in S, and $[h_i, \beta_i^S]$ denotes the concatenation operation on the corresponding h_i and β_i^S. Thus, Equation (3) is updated as in (8):

$$h_j = g(t_{j-1}, h_{j-1}, topic\_c_j) \tag{8}$$

in order to obtain the source topic-informed NMT model.

4.2 Topic-Informed Decoder

While the source topic-informed NMT model is useful for generating accurate translations, introducing topic information on the target side can help topic consistency between target words. A natural choice for maintaining this topic consistency is to use the same architecture as the decoder (i.e. GRU), to obtain the topic hidden state for the corresponding target word. We use $h_{j-1}^{\beta^T}$ to represent the hidden state of target topics for time $j-1$. We then can update Equation (3) as in (9):

$$h_j = g(t_{j-1}, h_{j-1}, c, h_{j-1}^{\beta^T}) \tag{9}$$

	Source	Target
Sentence Num.	1,501,652	1,501,652
Token Num.	38,388,118	44,901,788

Table 1: Training data statistics.

to obtain the target topic-informed NMT model. Consequently, the decoder can then use the topic knowledge of previous translated words to increase the likelihood of selecting words from the same topic.

4.3 Topic-Informed NMT

We now can combine the topic-informed encoder and the topic-informed decoder to obtain the overall topic-informed NMT system, as in Equation (10):

$$h_j = g(t_{j-1}, h_{j-1}, topic\_c_j, h_{j-1}^{\beta^T}) \tag{10}$$

Figure 2 provides a the graphical illustration of this novel topic-informed NMT model.

4.4 Topic Modeling

Instead of documents, sentences are often used to represent the mixture of topics in MT. For example, given the source or target training corpus, we can choose a fixed N topics to learn. The effectiveness of such an approach is that the topic representations can be learned directly using off-the-shelf algorithms. We take the same approach in this work, and use the translation training data directly to learn the topic representations. We use the Latent Dirichlet Allocation (LDA) implementation in the topic modeling toolkit (Řehůřek and Sojka, 2010) to train the topic model and compute the topic distributions of words.[2] We train the models with 200 iterations. Training takes approximately 40 hours on average for all the LDA models used in this work. For a detailed explanation of LDA, we refer the reader to Blei et al. (2003). Other options are to use Latent Semantic Analysis (LSA) (Deerwester et al., 1990) or Hidden Topic Markov Models (HTMM) (Gruber et al., 2007), which have not been studied in this work. We leave them as part of our future research.

5 Experiments

5.1 Data and Experiment Models

We report our experimental results on the NIST evaluation data set in the Chinese-to-English translation direction. Our MT training data are extracted from LDC corpora,[3] and NIST 2002 is used as our development set. Table 1 shows the details of the training data used. We use NIST 2004 and 2005 as our test sets. The English training data is tokenized and lowercased using scripts in Moses (Koehn et al., 2007). The Stanford Chinese word segmenter (Tseng et al., 2005) is used to segment the Chinese training data. In NMT, we limit our vocabularies to be the top 16,000 most frequent words, which covers 97.57% and 98.77% of the original words in the source and target training corpora, respectively. Outside the 16,000 threshold, all tokens are mapped to *UNK*.

We compare our approach against two baselines. The SMT baseline is trained using Moses, with a lexicalized reordering model (Koehn et al., 2005; Galley and Manning, 2008) and a 5-gram KenLM (Heafield, 2011) language model trained using the target side of the parallel training data. We use all the default parameters in Moses. Our second baseline is an NMT system. We use the encoder-decoder architecture with a single layer bidirectional GRU as the encoder, and a single layer GRU with attention model as the decoder. Each word in the training corpora is converted into a 512-dimensional vector during training. The hidden layers of the encoder and decoder each contain 1,024 hidden units. The bidirectional RNN described in Section 3.3 is also used. We use beam search during translation, with a beam size of 5. We use a minibatch (with batch size 32) stochastic gradient descent algorithm together with Adadelta (Zeiler, 2012) to train our models. All NMT models are trained up to 320,000 updates

[2] `https://radimrehurek.com/gensim/models/ldamulticore.html`

[3] LDC2002E18, LDC2003E07, LDC2003E14, LDC2004T07, the Hansards portion of LDC2004T08 and LDC2005T06

Systems	NIST 2002 (dev)	NIST 2004 (test)	NIST 2005 (test)
SMT	33.42	32.36	30.11
NMT	34.33	34.76	31.12
Source Topic-Informed NMT (40)	35.39	35.17†	31.95‡
Target Topic-Informed NMT (10)	36.31	35.43‡	32.50‡
Topic-Informed NMT (40,10)	34.86	35.91‡	32.79‡

Table 2: BLEU scores of the trained SMT and NMT models. We use ‡ and † to indicate significant (Koehn, 2004) improvements upon the baseline NMT using bootstrapping method at the level p = 0.01 and p = 0.05 level, respectively (with 1000 iterations).

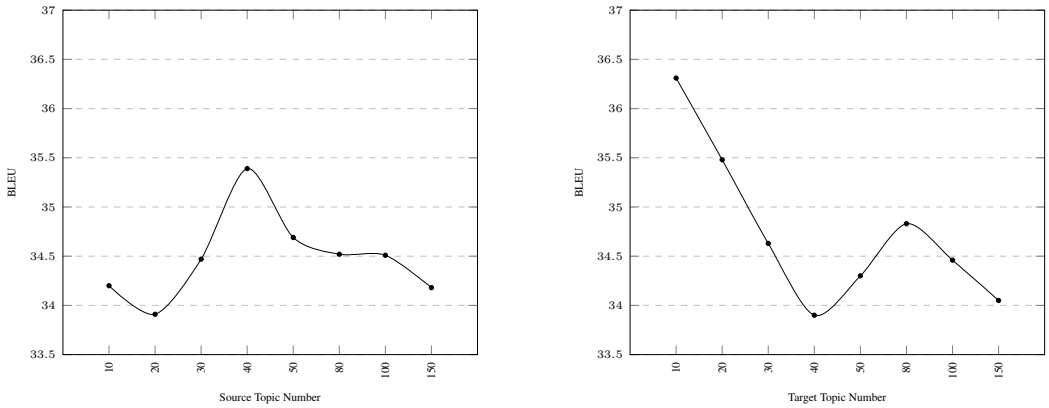


Figure 3: Topic numbers vs. translation BLEU scores on the NIST 2002 development dataset.

and the models are saved at each 1,000 updates. The training takes approximately 3 days on an NVIDIA GeForce GTX TITAN X GM200 GPU machine. We then choose the final model based on the BLEU[4] (Papineni et al., 2002) score on the development data.

5.2 Results

Table 2 presents the experiment results on the development and test data. In Table 2, the number next to each topic-informed NMT system indicates the number of topics used in the system, i.e. we use 40 source topics in the source topic-informed NMT system, and 10 target topics in the target topic-informed NMT model. The source and target topic numbers are experimentally chosen from $\{10, 20, 30, 40, 50, 80, 100, 150\}$ according to the development BLEU scores, as seen in Figure 3. We then leverage topic information on both source (with 40 topics) and target (with 10 topics) sides, which produces the topic-informed system (40,10) in Table 2. We think that the source topic number and the target topic number are not necessarily to be the same as the topic distributions are used differently in the proposed NMT model. The source topic distribution is appended to each word in the encoder, and the target distribution is passed via the RNN in the decoder.

We first compare the system performance on the development data. According to Table 2, using topic information can improve system performance over the two baseline systems. The source topic-informed NMT (40) system can gain absolute improvements of 1.97 (5.8% relative) and 1.06 (3.1% relative) BLEU scores compared to the SMT and NMT baseline systems, respectively. When using the topic information on the target side, we can observe absolute BLEU score improvements of 2.89 (8.6% relative) and 1.98 (5.8% relative) compared to the SMT and NMT baseline systems, respectively, which is the best-performed system on the development data. The topic-informed NMT (40, 10) system can only gain absolute 0.53 (1.5% relative) and 1.44 (4.3% relative) improvements compared to the SMT and NMT baseline systems, respectively.

In Table 2, significant improvements can be observed on the test data. There is a gain of 0.41 (absolute, 1.8% relative) and 0.83 (absolute, 2.7% relative) BLEU scores compared with the NMT baseline systems when the topic information is employed on the source side, on the NIST 2004 and NIST 2005 data sets,

[4] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

Source:	他/He 当即/immediately 被/sent 送往/to 医院/hospital 抢救/emergency treatment
Translation 1:	He was sent to hospital for **rescue**
Translation 2:	He was sent to hospital for **emergency treatment**
Source:	过半/Over half 英国人/British 不/disagree 赞同/disagree 政府/government 支持/support 美/US 打/attack 伊拉克/Iraq
Translation 1:	The British people **does not agree** to support United States to **fight** Iraq
Translation 2:	The British people **disagree** with the government 's support for US **war against** Iraq

Figure 4: The examples shows our observations that better word choices can be made in the topic informed NMT. Translation 1 is produced by the baseline NMT, and Translate 2 is the topic-informed NMT output.

Source:	印尼/Indonesia 国会/parliament **议长/speaker** 出庭/stands 受审/trial
Translation 1:	UNK of Indonesia 's congress in court
Translation 2:	Indonesian parliament **speaker** is trial
Source:	**卡伊达/Qaida** 组织/Qaida 领导层/leadership 也/also 开始/began to **活跃/active** 起来/be 。/.
Translation 1:	The leadership of the UNK city began to UNK .
Translation 2:	The leadership of the UNK organization has begun to be **active** .

Figure 5: The examples shows our observations that less number of *UNK* can be produced in the topic informed NMT. Translation 1 is produced by the baseline NMT, and Translate 2 is the topic-informed NMT output.

respectively. Furthermore, we find further absolute BLEU score improvements of 0.67 (absolute, 1.9% relative) and 1.38 (absolute, 4.4% relative) on the target topic-informed NMT (10) system, on the NIST 2004 and NIST 2005 data sets, respectively. The best-performing system on the test data is the topic-informed NMT (40,10) system, which achieves 35.91 and 32.79 BLEU scores on the NIST 2004 and NIST 2005 data sets, respectively. These results are 1.15 (absolute, 3.3% relative) and 1.67 (absolute, 5.4% relative) higher compared with the NMT baseline systems. Overall, the topic-informed NMT system significantly improves upon the baseline translation performance.

5.3 Observations

As we described earlier, by allowing topic information into the decoder, topic-consistent behaviour can be maintained, and consequently better translations can be produced. We find that the translations produced by the baseline NMT system is more fluent compared to SMT. However, two types of errors are still commonly made.

The first type of error produced in NMT translations is the lexical selection error. We find that when words can be translated by NMT, some of the translations are not appropriate in the context, i.e. words are not suitable for the domain. For example, in the baseline NMT translation examples in Figure 4, 抢救/"emergency treatment" is translated into *rescue* and 打/attack is translated into *fight*. In topic-informed NMT translations in Figure 4, the source input sentences and previous translations, e.g. *hospital* in the first example, and *British*, *government* and *US* in the second example, can give strong indications to the decoder about the current topics, namely "*Hospital*" and "*Politics*" topics in the two examples, respectively, that better word choices can thus be produced in the translations.

The second type of error that can be observed in the NMT translations is that the lexical coverage is low. Some of the source words are translated as *UNK* even though the correct translations can be found in the target vocabulary list. Comparing the *UNK* numbers produced by the baseline NMT and the topic-informed NMT, we observe that the number of *UNK* tokens has been reduced in the topic-informed NMT translations, as presented in Table 2. Interestingly, we also find that the topic-informed NMT system tends to produce more words in translations, namely 50,913 and 34,695 words in NIST 2004 and NIST 2005, respectively. In contrast, the NMT baseline produces 44,552 and 30,558 words in NIST 2004 and NIST 2005, respectively. In conclusion, topic-informed NMT can reduce the *UNK* number even when more words appear in the translation outputs. We think the reason for this the topic distribution of the *UNK* token favours to one particular topic. If the source inputs do not belong to the

	Baseline NMT	Topic-Informed NMT
NIST 2004	2.3%	1.9%
NIST 2005	2.7%	2.3%

Table 3: The percentage of *UNK* tokens produced in translation outputs by baseline NMT and topic-informed NMT systems.

same topic in which *UNK* appears, the *UNK* token will have less chance to be chosen as the translation output. Therefore, other word choices than *UNK* can be made and the overall *UNK* number is reduced. Inspecting the translation examples in Figure 5, 议长/speaker and 活跃/active fail to be translated by the baseline NMT system. However, topic-informed NMT is able to use the known topic information, either from the source sentences or previous translations, to produce correct translations. For example, the source words 议长/speaker and 活跃/active are translated into *speaker* and *active*, respectively. The source word 卡伊达/Qaida does not appear in the parallel training data, so both systems produce *UNK* in this case.

Figure 6 compares the word alignments produced by the two systems. In this example, we can find that the word alignments of topic-informed NMT can give more weights (α_{ij} in Equation (6)) to the correct aligned words, which consequently indicates to the decoder to pay more attentions on the correct source works to translate.

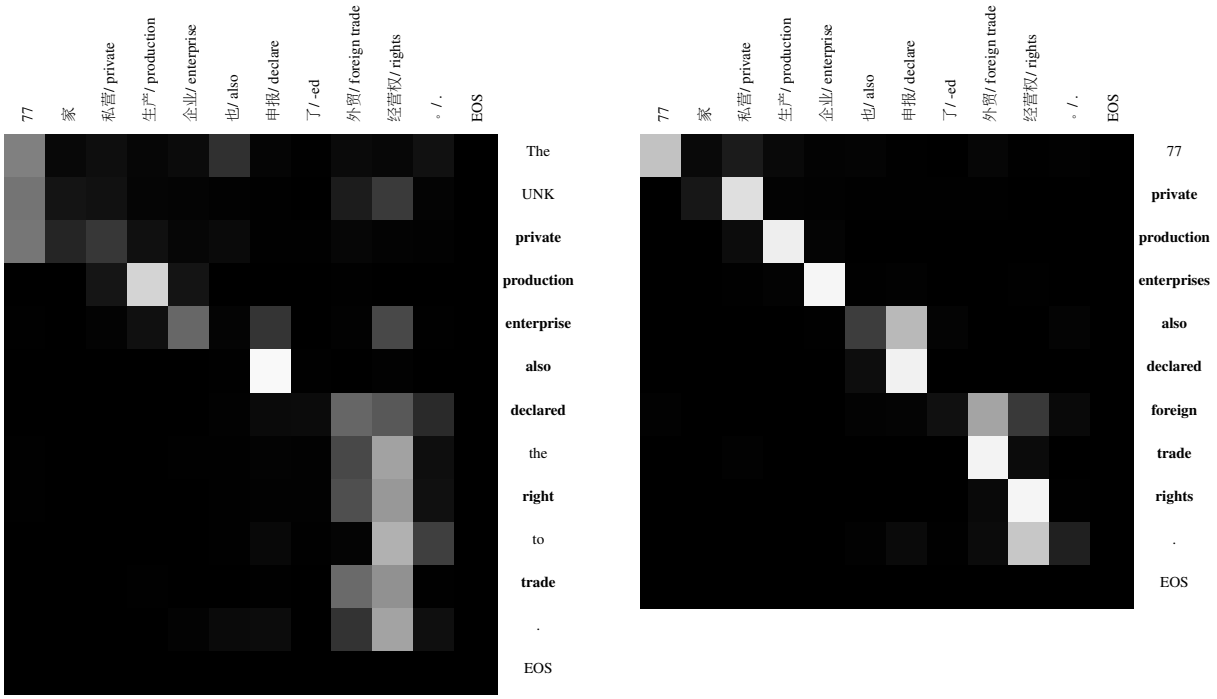


Figure 6: The comparison of alignments generated without (left) and with (right) topic knowledge. The example is chosen from the development data, where the x-axis sentence is the source training sentence (Chinese), and the y-axis sentence is the target training sentence (English). In the heatmap, we use grayscale colour schemes, where black means the word alignment probability is low, and white means the word alignment probability is high.

Our baseline NMT system contains 58,427,521 parameters to learn. Since we concatenate the source topic information and use a GRU network to store the target topic information, the topic-informed NMT system contains 817,984 more parameters to learn. During our experiments, we see that all of the trained NMT models (baseline NMT and topic-informed NMT systems listed in Table 2) produce their best-performing models between 280,000 and 320,000 in terms of update numbers. The baseline NMT and the topic-informed NMT systems require 284,000 and 289,000 updates to learn the best-performing models on the development data, respectively.

6 Conclusion

NMT has a lot of potential as a new approach to MT. In this paper, we present a novel approach of integrating topic knowledge into the existing NMT architecture. Through our experiments, we show that translation quality can be improved. We demonstrate that our topic-informed NMT can achieve 1.15 and 1.67 absolute improvements in BLEU score on two different test sets. The experimental results not only demonstrate the effectiveness of the proposed model, but also show an approach to enrich the representation of the context vector produced by the encoder and decoder. We show that introducing topic information in NMT can produce translations with better lexical selection, and a lower number of UNKs. We give concrete examples to support our observations. Furthermore, we argue that better word alignments can also be learned which consequently benefits translation quality.

In the future, we want to experiment on other topic learning approaches, e.g. LSA or HTMM, or jointly train neural topic model (Cao et al., 2015) and translation. We are also interested in further exploring the correlation between NMT performance and the quality of the topic modeling itself.

Acknowledgments

We would like to thank the three anonymous reviewers for their valuable and constructive comments and the Irish Center for High-End Computing (`www.ichec.ie`) for providing computational infrastructures. The ADAPT Centre for Digital Content Technology (`www.adaptcentre.ie`) at Dublin City University is funded under the Science Foundation Ireland Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the International Conference on Learning Representations 2015, ICLR 2015*, San Diego, California, USA.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A Novel Neural Topic Model and Its Supervised Extension. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015*, pages 2210–2216, Austin, Texas, USA.

David Chiang, Steve DeNeefe, and Michael Pust. 2011. Two Easy Improvements to Lexical Weighting. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, pages 455–460, Portland, Oregon, USA.

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 263–270, University of Michigan, USA.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734, Doha, Qatar.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *Computing Research Repository (CoRR)*, abs/1412.3555.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Vladimir Eidelman, Jordan L. Boyd-Graber, and Philip Resnik. 2012. Topic Models for Dynamic Translation Model Adaptation. In *The 50th Annual Meeting of the Association for Computational Linguistics*, pages 115–119, Jeju Island, Korea.

Michel Galley and Christopher D. Manning. 2008. A Simple and Effective Hierarchical Phrase Reordering Model. In *EMNLP 2008: 2008 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 848–856, Honolulu, Hawaii, USA.

Mercedes García-Martínez, Loïc Barrault, and Fethi Bougares. 2016. Factored Neural Machine Translation. *Computing Research Repository (CoRR)*, abs/1609.04621.

Amit Gruber, Yair Weiss, and Michal Rosen-zvi. 2007. Hidden Topic Markov Models. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS-07)*, volume 2, pages 163–170.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On Using Monolingual Corpora in Neural Machine Translation. *Computing Research Repository (CoRR)*, abs/1503.03535.

Eva Hasler, Barry Haddow, and Philipp Koehn. 2012. Sparse Lexicalised Features and Topic Adaptation for SMT. In *2012 International Workshop on Spoken Language Translation, IWSLT*, pages 268–275, Hong Kong.

Eva Hasler, Barry Haddow, and Philipp Koehn. 2014. Dynamic Topic Adaptation for SMT using Distributional Profiles. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 445–456, Baltimore, Maryland, USA.

Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. 2016. Improved Neural Machine Translation with SMT Features. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 151–157, Phoenix, Arizona, USA.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland.

Philipp Koehn, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *2005 International Workshop on Spoken Language Translation*, pages 68–75, Pittsburgh, PA, USA.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 388–395, Barcelona, Spain.

Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the Rare Word Problem in Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 11–19, Beijing, China.

Daniel Marcu and William Wong. 2002. A Phrase-Based, Joint Probability Model for Statistical Machine Translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 133–139, University of Pennsylvania, Philadelphia, PA, USA.

Tomas Mikolov and Geoffrey Zweig. 2012. Context Dependent Recurrent Neural Network Language Model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 234–239, Miami, Florida ,USA.

Franz Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In *38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, Hong Kong, China.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *41st Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 160–167, Sapporo Convention Center, Japan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, USA.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta.

Rico Sennrich and Barry Haddow. 2016. Linguistic Input Features Improve Neural Machine Translation. In *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL 2016*, pages 83–91, Berlin, Germany.

Jinsong Su, Deyi Xiong, Yang Liu, Xianpei Han, Hongyu Lin, Junfeng Yao, and Min Zhang. 2015. A Context-Aware Topic Model for Statistical Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 229–238, Beijing, China.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 3104–3112, Montreal, Quebec, Canada.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A Conditional Random Field Word Segmenter for Sighan Bakeoff 2005. In *In Fourth SIGHAN Workshop on Chinese Language Processing*.

Xinyan Xiao, Deyi Xiong, Min Zhang, Qun Liu, and Shouxun Lin. 2012. A Topic Similarity Model for Hierarchical Phrase-based Translation. In *The 50th Annual Meeting of the Association for Computational Linguistics*, pages 750–758, Jeju Island, Korea.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *Computing Research Repository (CoRR)*, Volume: abs/1212.5701.

Min Zhang, Xinyan Xiao, Deyi Xiong, and Qun Liu. 2014. Topic-based Dissimilarity and Sensitivity Models for Translation Rule Selection. *Journal of Artificial Intelligence Research*, 50(1):1–30.

A Distribution-based Model to Learn Bilingual Word Embeddings

Hailong Cao[1], **Tiejun Zhao**[1], **Shu Zhang**[2], **Yao Meng**[2]
[1]Harbin Institute of Technology, Harbin, China
[2]Fujitsu Research and Development Center, Beijing, China
{hailong,tjzhao}@mtlab.hit.edu.cn
{zhangshu,mengyao}@cn.fujitsu.com

Abstract

We introduce a distribution based model to learn bilingual word embeddings from monolingual data. It is simple, effective and does not require any parallel data or any seed lexicon. We take advantage of the fact that word embeddings are usually in form of dense real-valued low-dimensional vector and therefore the distribution of them can be accurately estimated. A novel cross-lingual learning objective is proposed which directly matches the distributions of word embeddings in one language with that in the other language. During the joint learning process, we dynamically estimate the distributions of word embeddings in two languages respectively and minimize the dissimilarity between them through standard back propagation algorithm. Our learned bilingual word embeddings allow to group each word and its translations together in the shared vector space. We demonstrate the utility of the learned embeddings on the task of finding word-to-word translations from monolingual corpora. Our model achieved encouraging performance on data in both related languages and substantially different languages.

1 Introduction

Learning word vector representations based on neural network is now a ubiquitous technique in natural language processing tasks and applications. Tremendous advances have been brought by distributed representations to the state-of-the-art methods (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013a; Pennington et al., 2014). In these models, words are represented by *dense real-valued low-dimensional vectors* referred to as word embeddings learned from raw text. Distributed representations have the property that similar words are represented by similar vectors and thus can achieve better generalization. Such representations are usually learned from monolingual data and therefore might not be generalized well across different languages.

In order to learn useful syntactic and semantic features that are invariant to languages, several models for learning cross-lingual representations have been proposed and achieved impressive effects by incorporating cross-lingual distributional information (Klementiev et al., 2012; Zou et al., 2013; Chandar et al., 2014; Faruqui and Dyer, 2014; Hermann and Blunsom, 2014; Gouws et al., 2015; Luong et al., 2015; Shi et al., 2015; Vulić and Moens, 2015; Upadhyay et al., 2016). In the cross-lingual settings, similar representations are desired for words denoting similar concepts in different languages (e.g., the embeddings of the English word *computer* and the French word *ordinateur* should be similar). Cross-lingual representations are especially useful for many natural language processing tasks such as machine translation (Zou et al., 2013; Zhang et al., 2014), computing cross-lingual word similarity (Zhang et al., 2016) and transferring knowledge from high-resource languages to low-resource languages (Guo et al., 2015), etc.

However, all these cross-lingual models require some form of cross-lingual supervision such as seed lexicon,word-level alignments, sentence-level alignments and document-level alignments. Reliance on supervision might limit the development and application of cross-lingual representations. In this paper, we proposed a distribution based model to learn bilingual word embeddings from monolingual data. The

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1818–1827, Osaka, Japan, December 11-17 2016.

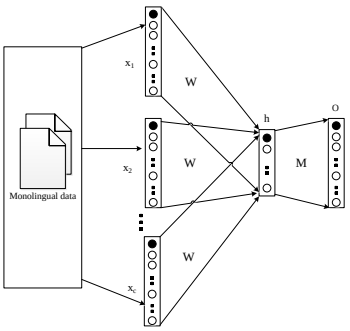


Figure 1: The CBOW architecture predicts the current word based on the context. It is a monolingual representations learning model.

proposed approach is complementary to the existing methods that rely on supervision. Our contributions are the following:

- We introduce a novel cross-lingual objective which is employed to match the distributions of monolingual embeddings as they are being trained in an online setting. Our model only requires monolingual data and therefore could be applied to any languages or domains that we are interested in.
- Our model can capture the common regularity shared by natural languages. The resulting bilingual embeddings allow to group a word and its translations together in vector space. We demonstrate the utility of our model on the task of finding word-to-word translations solely from monolingual corpora. Our model achieved encouraging performance on both related languages and substantially different languages.

2 Monolingual Word Embeddings Learning

Our framework is general enough to be built based on any monolingual embedding learning model. We adopt the popular continuous bag-of-Words (CBOW) model (Mikolov et al., 2013a) to demonstrate our approach. The CBOW model uses continuous distributed representation of the context where each word is mapped to a learned vector. The architecture is shown in the Figure 1. The training data D is a set of pairs in the form of (x, y), in which y is a word and $x = (x_1, x_2, ..., x_C)$ is a set containing C context words in which y appears. We have $y, x_i \in (1, 2, ..., V)$, where V is the vocabulary size. The training criterion is to seeking parameters minimizing the loss function which is the negative log probability of y given x:

$$\hat{W}, \hat{M} = \underset{W,M}{\operatorname{argmin}} \sum_{(x,y)\in D} L(W, M, x, y) = \underset{W,M}{\operatorname{argmin}} \sum_{(x,y)\in D} -log(P(y|x, W, M)) \tag{1}$$

We use the one-hot V-dimension column vector $\vec{x}_i$ to refer the context word x_i in which only the x_i^{th} unit is 1, and all other units are 0. W is a $K \times V$ matrix representing the weights between the input layer and the K-dimensional hidden layer. Each column of W is the K-dimensional vector representation of the associated word of the input layer. W transforms each context word x_i into a K-dimension real value vector. Each context word x_i is mapped into a real value K-dimensional vector by W. The CBOW model takes the average of these vectors as the value of hidden layer.

$$\vec{h} = \frac{1}{C} \sum_{i=1}^{C} W\vec{x}_i \tag{2}$$

where the matrix W is shared by all context words.

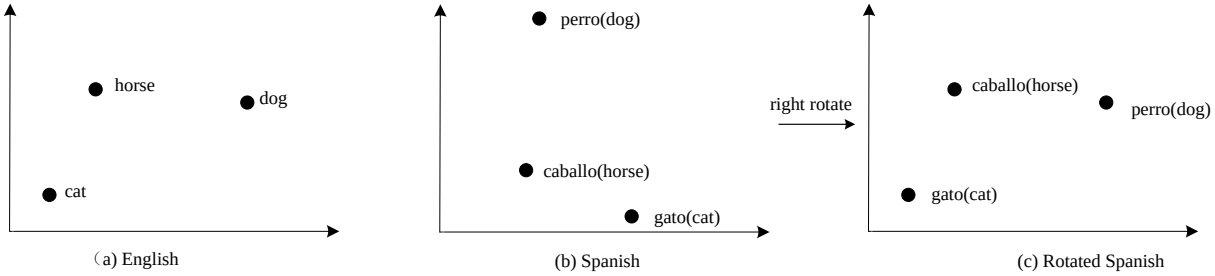


Figure 2: Monolingual embeddings have been shown to have similar geometric shape (a & b). It is desired to have more explicit similarity (a & c). (Mikolov et al., 2013b) achieved this by leaning a linear projection with a bilingual dictionary.

From the hidden layer to the output layer, there is a $V \times K$ weight matrix M by which the hidden vector is mapped into a V-dimensional output vector:

$$\vec{O} = M\vec{h} \tag{3}$$

which will be normalized by the soft-max function as a distribution over V candidate words. Finally, the probability of the word y given its context x is:

$$P(y|x, W, M) = \frac{exp(O_y)}{\sum_{i=1}^{V} exp(O_i)} \tag{4}$$

where O_i is the i^{th} unit of vector $\vec{O}$.

The parameters of W and M are tuned by the standard stochastic gradient descent (SGD) algorithm. There are very interesting properties in the learned word vectors. For example, similar words are nearby vectors in a vector space. And more importantly, if vectors learned for languages are manually rotated, Mikolov et al. (2013b) observed that languages share similar geometric arrangements in vector spaces (shown in Figure 2). The reason is that all common languages share universal structure of human lexical semantics(Youn et al., 2016). To capture the similarities, they use a bilingual dictionary to learn a linear projection between vectors learned independently from each language.

Our work is also motivated by the observation in (Mikolov et al., 2013b). Rather than relying on geometric transformation, we focus on word embeddings learning itself and explore an joint bilingual learning framework.

3 Learning Bilingual Word Embeddings by Distribution Matching

In the cross-lingual setup, we desire word embeddings to be generalized well across different languages. For example, given embedding of English context words {*the, cats, on, the, mat*}, cross-lingual models should not only be able to predict that the current word can be *sits* in English, but also be able to predict it can be *assis* in French. Similarly, given embeddings of French context words {*le, chat, est, sur, ma*}, cross-lingual models should be able to predict both *sits* and *assis*. Such desire bears a strong resemblance to the problem of domain adaptation which is well-studied in the field of natural language processing (Blitzer et al., 2006; Daume III, 2007). We apply the theory on domain adaptation which can learn cross-domain features to the task of learning cross-lingual features(word embeddings). Before we detail the proposed framework for unsupervised cross-lingual representation learning, we briefly introduce methods in domain adaptation.

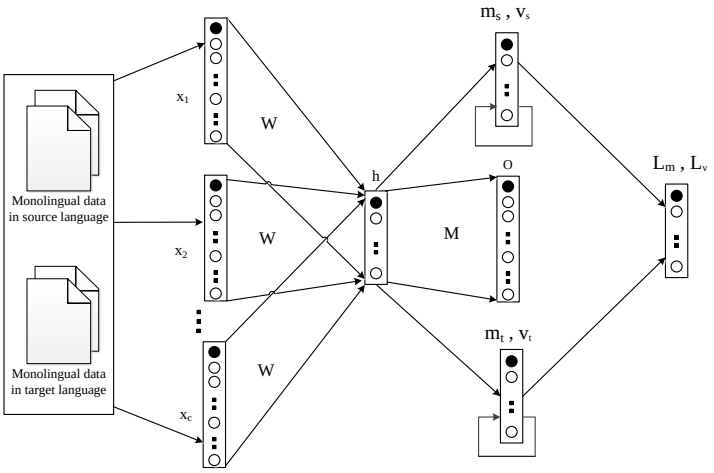


Figure 3: Bi-lingual representation learning is achieved by matching the distribution of the source and target languages. The dynamic statistic of hidden states of the source and target languages are calculated on line, and the dissimilarities between them are minimized through standard back propagation algorithm.

3.1 Domain Adaptation

This work is inspired by theory on domain adaptation (Ben-David et al., 2006; Ben-David et al., 2010) which suggest that, for effective domain transfer, predictions must be made based on data representations that cannot discriminate the source and target domains. Based on this theory, very simple and efficient domain adaptation approaches have been developed (Ajakan et al., 2014; Ganin and Lempitsky, 2015) for representation learning in neural networks. The main idea is matching feature space distributions of source and target domain. To this end, Ganin and Lempitsky (2015) added a domain classifier connected to the feature extractor. Feature distributions over the two domains are made similar (as indistinguishable as possible for the domain classifier), thus resulting in the domain-invariant features.

A straightforward way to learn cross-lingual representations is extending the ideas of Ganin and Lempitsky (2015) by replaceing their domain classifer with a language classifier. Alternatively, in this paper we explore a much more direct approach.

3.2 Model Architecture

We observed that, unlike features in image processing which are *high-dimensional*, language representations are usually in form of *dense real-valued low-dimensional* vector and therefore the distribution of them can be accurately estimated, given the freely available large scale raw text data. Based on this observation, we propose a novel learning objective which directly matches the distributions of word embeddings in one language with that in the other language. An overview of the architecture of bilingual word embeddings learning is given in Figure 3.

Same to that in the CBOW model, W and M are still $K \times V$ and $V \times K$ matrixes. But, now we have $V = V_s + V_t$ where V_s and V_t is the vocabulary size of source and target data respectively. All what we add to the CBOW model are the statistics of hidden states. We assume that the distribution of hidden states of each language is subject to a multi-dimensional normal distribution. So we have two statistics for each language, namely the mean $\vec{m}_s$ and the variance $\vec{v}_s$ on the source side, the mean $\vec{m}_t$ and the variance $\vec{v}_t$ on the target side. Like the hidden states, each statistic is also a K-dimensional vector. We use D_s and D_t to denote the nonparallel monolingual training data set from source language and target language respectively. Mathematically, on the source data, the mean is defined as:

$$\vec{m}_s = \frac{1}{|D_s|} \sum_{(x,y) \in D_s} \vec{h}(x) \tag{5}$$

where the hidden state h is defined in equation 2. And the variance is:

$$\vec{v}_s = \frac{1}{|D_s|} \sum_{(x,y) \in D_s} (\vec{h}(x) - \vec{m}_s)^2 \tag{6}$$

where the square operation is performed on each element of the K-dimensional vector respectively. On the target data, $\vec{m}_t$ and $\vec{v}_t$ are defined in the same way.

In order to encourage the source and the target data to have similar distributions in the shared space, one can directly minimize the dissimilarity between statistics of the source and the target data. More formally, the bilingual training criterion is to seeking parameters minimizing the standard monolingual objective of all data and the distribution dissimilarity:

$$\hat{W}, \hat{M} = \underset{W,M}{\operatorname{argmin}} (\sum_{(x,y) \in D_s \cup D_t} L(W, M, x, y) + \lambda_m L_m(\vec{m}_s, \vec{m}_t) + \lambda_v L_v(\vec{v}_s, \vec{v}_t)) \tag{7}$$

where L is defined in equation 1. L_m and L_v are cross-lingual objectives which are defined as the dissimilarities between statistics:

$$L_m(\vec{m}_s, \vec{m}_t) = \frac{1}{2} \sum_{i=1}^{K} ((\vec{m}_s)_i - (\vec{m}_t)_i)^2 \tag{8}$$

$$L_v(\vec{v}_s, \vec{v}_t) = \frac{1}{2} \sum_{i=1}^{K} ((\vec{v}_s)_i - (\vec{v}_t)_i)^2 \tag{9}$$

where i is index of each element in the vectors.

3.3 Dynamic Estimation

However, it is nontrivial to optimize the monolingual and cross-lingual objectives simultaneously in equation 7. The statistics defined in equation 5 and 6 can only be calculated when all word embeddings are given and fixed, while word embeddings have to be learned online by stochastic gradient descent algorithm. So it is impractical to use these statistics to guide the online learning of word embeddings. To deal with this chicken-egg problem, we propose to dynamically estimate the statistics. Initially, we have:

$$\vec{m}_s = 0 \tag{10}$$

$$\vec{v}_s = 0 \tag{11}$$

which will be iteratively updated by each incoming training instance (x_i,y_i):

$$\vec{m}_s = \frac{1}{scount + 1} (\vec{m}_s * scount + \vec{h}(x_i)) \tag{12}$$

$$\vec{v}_s \approx \frac{1}{scount + 1} (\vec{v}_s * scount + (\vec{h}(x_i) - \vec{m}_s)^2) \tag{13}$$

where $scount$ is the number of source training instances that have been used by the learning algorithm so far. The value of $scount$ is increased by one when a new training instance comes. To capture the latest trends and decay the effect the outdate data during training, we do not increase $scount$ anymore when it reaches 100 thousands.

On the target data, $\vec{m}_t$ and $\vec{v}_t$ are approximated in the same way.

3.4 Online Bilingual Training

Now we are ready to introduce the joint training procedure. In practice, we use multiple threads to train our model with source data and target data in parallel. On the source data, when a training instance (x_i, y_i) is accessed by the learning algorithm, we dynamically update the $\vec{m}_s$ and $\vec{v}_s$ with equation 12 and 13 as the output of the network. Then the golden references are the real time values of $\vec{m}_t$ and $\vec{v}_t$ which are being estimated in parallel on the target data. Based on loss function defined in equation 8 and 9, the gradient of distribution dissimilarities with respect to the hidden state is:

$$\frac{\partial(L_m + L_v)}{\partial \vec{h}(x_i)} = \frac{\lambda_m}{scount + 1}(\vec{m}_t - \vec{m}_s) + \frac{\lambda_v}{scount + 1}(\vec{v}_t - \vec{v}_s) \tag{14}$$

which will be added to the standard gradient of monolingual objective of the CBOW model:

$$\frac{\partial(L_m + L_v)}{\partial \vec{h}(x_i)} + \frac{\partial L(W, M, x, y)}{\partial \vec{h}(x_i)} \tag{15}$$

This gradient sum is utilized by the standard back propagation algorithm to make source data distribution similar to that of target data on one hand, and optimize the monolingual objective on the other hand.

In parallel, on the target data, the similar training procedure is being performed. Thus, the jointly learned source language word embeddings and target word embeddings will share similar distributions.

3.5 Bilingual Negative Sampling

The exact computation for probability shown in equation 4 for all words for every training instance is very expensive. Following the CBOW model, we adopted the negative sampling algorithm for high computational efficiency. As usual, for each source word y_s and its context x_s, we randomly sample a few words other than y_s from the *source* vocabulary. For example, given source word *sits* and its context {*the, cats, on, the, mat*}, we will sample a few words other than *sits*. Each selected word y_s^n is treated as a negative sample and the probability of predicting y_s^n given x_s is minimized.

Different from the CBOW model, we also randomly sample a few words from the *target* vocabulary and minimize the probability of predicting each selected target word y_t^n given x_s. Such bilingual negative sampling(BNS) procedure may introduce noises if the sampled target word y_t^n just happens to be the translation of the source word y_s. But, statistically such chance is quite small given the big vocabulary size of large scale text.

To further reduce such chance, we apply word frequency based diagonal beam (Nuhn et al., 2012) to constrain the BNS. Intuitively, the translation of a high frequency word should also be a frequent word, and vice-versa. Nuhn et al. (2012) use diagonal beam to select translation candidates, in this paper we apply it to filter out possible translations. We sort both source and target words by their frequency. Let $r(y_s)$ and $r(y_t)$ be the frequency rank of a source/target word. To avoid selecting y_t^n which is the translation of y_s, we require that the frequency rank of the sampled target word y_t^n should satisfy:

$$\left| r(y_t^n) - r(y_s)\frac{V_t}{V_s} \right| > BS \tag{16}$$

where BS is the beam size.

Similarly, we also apply the above BNS procedure when learning word embeddings for the target data in parallel.

4 Experiments

In this section we present experiments which evaluate the utility of the induced bilingual word embeddings. We implemented our model in C by building on the word2vec. The implementation launches a monolingual CBOW model by separate threads for each language. All threads access the shared embedding parameters and distribution means/variances. We evaluated the induced bilingual embeddings on the task of finding word-to-word translations from nonparallel corpora. This task is referred as decipherment which has drawn significant amounts of interest in the past few years (Nuhn et al., 2012;

	French	English
Training	29,608,749	27,355,418
Evaluation	60,474,279	54,478,614

Table 1: Size of data in tokens used in French to English decipherment experiment.

	5k	10k
MonoGiza without word embeedings	13.74	7.8
MonoGiza with word embeedings	17.98	10.56
Optimizing L	7.62	4.74
Optimizing L and L_m	22.24	17.05
Optimizing L, L_m and L_v	**23.54**	**17.82**

Table 2: French to English decipherment top-5 accuracy (%) of 5k and 10k most frequent word types.

Ravi, 2013; Dou et al., 2015). Decipherment views a foreign language as a cipher for English and finds a translation table that converts foreign texts into sensible English. It is a very challenging task since there is not any supervision.

4.1 Settings

We use MonoGiza[1] which implemented the state-of-the-art decipherment algorithms described in Dou and Knight (2012) and Dou et al. (2015) as baseline. In the preprocessing step, MonoGiza converts all words in data into integers and does not make any use of morphology similarity. For a fair comparison and to be general, we neither utilize that at all. All experiments are performed on plain raw text, we leave the use of syntactic relations for future work. The word embeddings used by MonoGiza are trained with word2vec. For all word embeddings in both word2vec and our model, the dimensionality is 50.

4.2 French to English Decipherment

Data The datasets in our English to French experiments are publicly-available Europarl data[2]. From the English-French Europarl parallel data, we select the first half of English sentences and the second half French sentences respectively as non-parallel corpora for decipherment experiments. To evaluate the decipherment, we use Giza++ (Och and Ney, 2003) to align the Europarl parallel data to build a dictionary. All texts are tokenized by scripts from www.statmt.org. Table 1 lists the sizes of monolingual and parallel data used in this experiment.

Decipherment In order to make all results comparable, results for all methods reported here were obtained using the same nonparallel corpora. λ_m and λ_v were set to 0.2 and 0.1 respectively. The number of negative samples from both source and target side for each word are 5. The beam size of BNS was set to 1000. We use default values for all other hyper parameters in word2vec and MonoGiza. Table 2 shows the experimental results. Bilingual word embeddings are induced by optimizing the objectives in equation 7. For each French word, we select its top-5 nearest neighbor words in English as translations based on the cosine similarity defined in the shared 50-dimensional space. We use the evaluation script included in the package of MonoGiza. Though the absolute accuracy is not very high, we believe it is encouraging given that there is not any supervised information. Only optimizing the monolingual training objective L can capture some similarities between languages, but the accuracy is pretty low. Simultaneously minimizing L and distribution mean dissimilarity L_m is effective. We achieved the best performance when three objectives L, L_m and L_v are optimized together. In such case, the distributions of source and target data share more similarities.

Table 3 shows a number of example translations from French to English. Though they are far from perfect, some translations are meaningful and are semantically related to the correct translation.

[1] http://www.isi.edu/natural-language/software/monogiza_release_v1.0.tar.gz
[2] http://www.statmt.org/europarl/

french word	English Translations	cosine similarity	Dictionary Entry
Du	the	-0.128120	the
	is	-0.131779	
	in	-0.136466	
	that	-0.137784	
	which	-0.139639	
sincèrement	Liberalisation	-0.007873	Frankly
	Essentially	-0.009341	
	Throughout	-0.016099	
	vodka	-0.023932	
	Frankly	-0.031336	
principaux	important	-0.018043	important
	good	-0.018260	
	able	-0.018508	
	much	-0.018643	
	come	-0.018987	

Table 3: Examples of translations of words from French to English. The five most likely translations are shown.

	Chinese	English
Training	315,800,768	403,215,310
Evaluation	41,888,921	49,822,055

Table 4: Size of data in tokens used in Chinese to English decipherment experiment.

4.3 Chinese to English Decipherment

We have demonstrated the effect of our model with experiments on French and English which are related languages. To further evaluate the ability our model, we experiment on data in Chinese and English which are substantially different.

Data We use large scale Chinese and English data released by LDC. The monolingual Chinese data is the Xinhua part of Chinese Gigaword. The monolingual English data is the LDC English Gigaword. Bilingual word embeddings are induced based the above non-parallel data. A golden dictionary is built by Giza++ based on parallel corpus LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06. All Chinese sentences are segmented by the Stanford Word Segmenter[3]. Table 4 lists the sizes of monolingual and parallel data used in this experiment. The settings are the same as the French-English case.

Decipherment For the calculation of accuracy, we discarded Chinese words if they are not covered by the gold dictionary. Table 5 shows the experimental results of Chinese to English decipherment. Though the two languages are substantially different, the results indicate that our model is still able to learn translation equivalences from the monolingual data by using only the learned bilingual word embeddings.

	5k	10k
MonoGiza without word embeedings	15.56	9.04
MonoGiza with word embeedings	17.5	10.57
Optimizing L, L_m and L_v	**24.76**	**18.45**

Table 5: Chinese to English decipherment top-5 accuracy (%) of 5k and 10k most frequent word types.

5 Conclusions and Future Work

We have proposed a novel model to learn bilingual word embeddings directly from monolingual raw text, without requiring any parallel data or dictionaries. A novel cross-lingual learning objective is proposed which directly matches the distributions of word embeddings in one language with that in the other language. We have demonstrated the utility of the learned word embeddings in the task of decipherment. Our model achieved encouraging performance on data from both related languages and substantially

[3]http://nlp.stanford.edu/software/segmenter.shtml

different languages. In the future, we would apply our method to more real applications such as cross-lingual dependency parsing, cross-lingual document classification and machine translation. Our model is complementary to the existing methods that rely on supervision, so we are also interested in combining it with supervised models to achieve much better cross-lingual word representations.

Acknowledgments

We thank anonymous reviewers for their insightful comments. The work of HIT is funded by the projects of National Natural Science Foundation of China(No.91520204, No.71531013, No. 61572154) and the project of National High Technology Research and Development Program of China(No. 2015AA015405)

References

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. 2014. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*.

Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. 2006. Analysis of representations for domain adaptation. In *Proceedings of the Conference on Advances in Neural Information Processing Systems(NIPS)*.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, Sydney, Australia, July. Association for Computational Linguistics.

A. P. Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M. Khapra, Balaraman Ravindran, Vikas C. Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Proceedings of the Conference on Advances in Neural Information Processing Systems(NIPS)*, pages 1853–1861.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of International Conference on Machine Learning*.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June. Association for Computational Linguistics.

Qing Dou and Kevin Knight. 2012. Large scale decipherment for out-of-domain machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 266–275, Jeju Island, Korea, July. Association for Computational Linguistics.

Qing Dou, Ashish Vaswani, Kevin Knight, and Chris Dyer. 2015. Unifying bayesian inference and vector space models for improved decipherment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 836–845, Beijing, China, July. Association for Computational Linguistics.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Advances in neural information processing systems the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden, April. Association for Computational Linguistics.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of The 32nd International Conference on Machine Learning*, Lille, France.

Stephan Gouws, Yoshua Bengio, and Greg Corado. 2015. Fast bilingual distributed representations without word alignments. In *Proceedings of The 32nd International Conference on Machine Learning*, Lille, France.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1234–1244, Beijing, China, July. Association for Computational Linguistics.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 58–68, Baltimore, Maryland, June. Association for Computational Linguistics.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*, pages 1459–1474, Mumbai, India, December. The COLING 2012 Organizing Committee.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the Workshop on Vector Space Modeling for NLP*.

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

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv:1309.4168*, abs/1309.4168.

Malte Nuhn, Arne Mauser, and Hermann Ney. 2012. Deciphering foreign language by combining language models and context vectors. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 156–164, Jeju Island, Korea, July. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

Sujith Ravi. 2013. Scalable decipherment for machine translation via hash sampling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 362–371, Sofia, Bulgaria, August. Association for Computational Linguistics.

Tianze Shi, Zhiyuan Liu, Yang Liu, and Maosong Sun. 2015. Learning cross-lingual word embeddings via matrix co-factorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 567–572, Beijing, China, July. Association for Computational Linguistics.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of ACL*.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 719–725, Beijing, China, July. Association for Computational Linguistics.

Hyejin Youn, Logan Sutton, Eric Smith, Cristopher Moore, Jon F. Wilkins, Ian Maddieson, William Croft, and Tanmoy Bhattacharya. 2016. On the universal structure of human lexical semantics. *In Proceedings of the National Academy of Sciences*.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014. Bilingually-constrained phrase embeddings for machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 111–121, Baltimore, Maryland, June. Association for Computational Linguistics.

Meng Zhang, Yang Liu, Huan-Bo Luan, Maosong Sun, Tatsuya Izuha, and Jie Hao. 2016. Building earth mover's distance on bilingual word embeddings for machine translation. In Dale Schuurmans and Michael P. Wellman, editors, *AAAI*, pages 2870–2876. AAAI Press.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398, Seattle, Washington, USA, October. Association for Computational Linguistics.

Pre-Translation for Neural Machine Translation

Jan Niehues, Eunah Cho, Thanh-Le Ha and Alex Waibel
Institute for Anthropomatics
Karlsruhe Institute of Technology, Germany
firstname.surname@kit.edu

Abstract

Recently, the development of neural machine translation (NMT) has significantly improved the translation quality of automatic machine translation. While most sentences are more accurate and fluent than translations by statistical machine translation (SMT)-based systems, in some cases, the NMT system produces translations that have a completely different meaning. This is especially the case when rare words occur.

When using statistical machine translation, it has already been shown that significant gains can be achieved by simplifying the input in a preprocessing step. A commonly used example is the pre-reordering approach.

In this work, we used phrase-based machine translation to pre-translate the input into the target language. Then a neural machine translation system generates the final hypothesis using the pre-translation. Thereby, we use either only the output of the phrase-based machine translation (PBMT) system or a combination of the PBMT output and the source sentence.

We evaluate the technique on the English to German translation task. Using this approach we are able to outperform the PBMT system as well as the baseline neural MT system by up to 2 BLEU points. We analyzed the influence of the quality of the initial system on the final result.

1 Introduction

In the last years, statistical machine translation (SMT) system generated state-of-the-art performance for most language pairs. Recently, systems using neural machine translation (NMT) were able to outperform SMT systems in several evaluations. These models are able to generate more fluent and accurate translation for most of sentences.

Neural machine translation systems provide the output with high fluency. A weakness of NMT systems, however, is that they sometimes lose the original meaning of the source words during translation. One example from the first conference on machine translation (WMT16) test set is the segment in Table 1.

The English word *goalie* is not translated to the correct German word *Torwart*, but to the German word *Gott*, which means *god*. One problem could be that we need to limit the vocabulary size in order to train the model efficiently. We used Byte Pair Encoding (BPE) (Sennrich et al., 2016) to represent the text using a fixed size vocabulary. In our case the word *goali* is splitted into three parts *go*, *al* and *ie*. Then it is more difficult to transport the meaning to the translation.

In contrast to this, in phrase-based machine translation (PBMT), we do not need to limit the vocabulary and are often able to translate words even if we have seen them only very rarely in the training. In the example mentioned before, for instance, the PBMT system had no problems translating the expression correctly.

On the other hand, official evaluation campaigns (Bojar et al., 2016) have shown that NMT system often create grammatically correct sentence and are able to model the morphologically agreement much better in German.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1828–1836, Osaka, Japan, December 11-17 2016.

Table 1: Example translation of NMT

English:	the goalie parried
NMT:	der Gott
NMT(gloss):	the god

The goal of this work is to combine the advantages of neural and phrase-based machine translation systems. Handling of rare words is an essential aspect to consider when it comes to real-world applications. The pre-translation framework provides a straightforward way to support such applications. In our approach, we will first translate the input using a PBMT system, which can handle the rare words well. In a second step, we will generate the final translation using an NMT system. This NMT system is able to generate a more fluent and grammatically correct translation. Since the rare words are already handled by the PBMT system, there should be less problems to generate the translation of these words. Using this approach naturally introduces a necessity to handle the potential errors by the PBMT systems.

The remaining of the paper is structured as follows: In the next section we will review the related work. In Section 3, we will briefly review the phrase-based and neural approach to machine translation. Section 4 will introduce the approach presented in this paper to pre-translate the input using a PBMT system. In the following section, we will evaluate the approach and analyze the errors. Finally, we will finish with a conclusion.

2 Related Work

The idea of linear combining of machine translation systems using different paradigms has already been used successfully for SMT and rule-based machine translation (RBMT) (Dugast et al., 2007; Simard et al., 2007). They build an SMT system that is post-editing the output of an RBMT system. Using the combination of SMT and RBMT, they could outperform both single systems.

Those experiments promote the area of automatic post-editing (Bojar et al., 2015). Recently, it was shown that models based on neural MT are very successful in this task (Junczys-Dowmunt and Grundkiewicz, 2016).

For PBMT, there has been several attempts to apply preprocessing in order to improve the performance of the translation system. A commonly used preprocessing step is morphological splitting, like compound splitting in German (Koehn and Knight, 2003). Another example would be to use pre-reordering in order to achieve more monotone translation (Rottmann and Vogel, 2007).

In addition, the usefulness of using the translations of the training data of a PBMT system has been shown. The translations have been used to re-train the translation model (Wuebker et al., 2010) or to train additional discriminative translation models (Niehues and Waibel, 2013).

In order to improve the translation of rare words in NMT, authors try to translate words that are not in the vocabulary in a post-processing step (Luong et al., 2015). In (Sennrich et al., 2016), a method to split words into sub-word units was presented to limit the vocabulary size. Also the integration of lexical probabilities into NMT was successfully investigated (Arthur et al., 2016).

3 Phrase-based and Neural Machine Translation

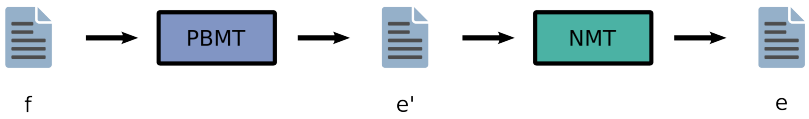
Starting with the initial work on word-based translation system (Brown et al., 1993), phrase-based machine translation (Koehn et al., 2003; Och and Ney, 2004) segments the sentence into continuous phrases that are used as basic translation units. This allows for many-to-many alignments.

Based on this segmentation, the probability of the translation is calculated using a log-linear combination of different features:

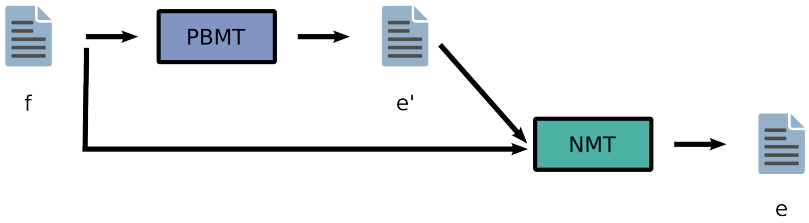
$$P(e^I, f^I) = \frac{exp(\sum_{n=1}^{N} \lambda_n h_n(e^I, f^I))}{\sum_{e'^I} exp(\sum_{n=1}^{N} \lambda_n h_n(e'^I, f^I))} \quad (1)$$

In the initial model, the features are based on language and translation model probabilities as well as a few count based features. In advanced PBMT systems, several additional features to better model the

Figure 1: Pre-translation methods



(a) Pipeline combination



(b) Mixed Input

translation process have been developed. Especially models using neural networks were able to increase the translation performance.

Recently, state-of-the art performance in machine translation was significantly improved by using neural machine translation. In this approach to machine translation, a recurrent neural network (RNN)-based encoder-decoder architecture is used to transform the source sentence into the target sentence.

In the encoder, an RNN is used to encode the source sentence into a fixed size continuous space representation by inserting the source sentence word-by-word into the network. In a second step, the decoder is initialized by the representation of the source sentence and is then generating the target sequence one word after the other using the last generated word as input for the RNN (Sutskever et al., 2014).

One main drawback of this approach is that the whole source sentence has to be stored in a fixed-size context vector. To overcome this problem, (Bahdanau et al., 2014) introduced the soft attention mechanism. Instead of only considering the last state of the encoder RNN, they use a weighted sum of all hidden states. Using these weights, the model is able to put attention on different parts of the source sentence depending on the current status of the decoder RNN. In addition, they extended the encoder RNN to a bi-directional one to be able to get information from the whole sentence at every position of the encoder RNN. A detailed description of the NMT framework can be found in (Bahdanau et al., 2014).

4 PBMT Pre-translation for NMT (PreMT)

In this work, we want to combine the advantages of PBMT and NMT. Using the combined system we should be able to generate a translation for all words that occur at least once in the training data, while maintaining high quality translations for most sentences from NMT. Motivated by several approaches to simplify the translation process for PBMT using preprocessing, we will translate the source as a preprocessing step using the phrase-base machine translation system.

The main translation task is done by the neural machine translation model, which can choose between using the output of the PBMT system or the original input when generate the translation.

4.1 Pipeline

In our first attempt, we combined the phrase-based MT and the neural MT in one pipeline as shown in Figure 1a. The input is first processed by the phrase-based machine translation system from the input language f to the target language e'. Since the machine translation system is not perfect, the output of the system may not be correct translation containing errors possibly. Therefore, we will call the output language of the PBMT system e'.

In a second step, we will train a neural monolingual translation system, that translates from the output of the PBMT system e' to a better target sentence e.

4.2 Mixed Input

One drawback of the pipelined approach is that the PBMT system might introduce some errors in the translation that the NMT can not recover from. For example, it is possible that some information from the source sentence gets lost, since the word is entirely deleted during the translation of the PBMT system.

We try to overcome this problem by building an NMT system that does not only take the output of the PBMT system, but also the original source sentence. One advantage of NMT system is that we can easily encode different input information. The architecture of our system is shown in Figure 1b.

The implementation of the mixed input for the NMT system is straight forward. Given the source input $f = f_1, \dots f_I$ and the output of the PBMT system $e' = e'_1, \dots e'_{J'}$, we generated the input for the NMT system. First, we ensured a non-overlapping vocabulary of f and e' by marking each token in f by a character and e' by different ones. Then both input sequences are concatenated to the input e^* of the NMT system.

Using this representation, the NMT can learn to focus on source word f_j and words $e'_{i'}$ when generating a word e'_j.

4.3 Training

In both cases, we can no longer train the NMT system on the source language and target language data, but on the output of the PBMT system and the target language data. Therefore, we need to generate translations of the whole parallel training data using the PBMT system.

Due to its ability to use very long phrases, a PBMT system normally performs significantly better on the training data than on unseen test data. This of course will harm the performance of our approach, because the NMT system will underestimate the number of improvements it has to perform on the test data.

In order to limit this effect, we did not use the whole phrase tables when translating the training data. If a phrase pair only occurs once, we cannot learn it from a different sentence pair. Following (Niehues and Waibel, 2013), we removed all phrase pairs that occur only once for the translation of the corpus.

5 Experiments

We analyze the approach on the English to German news translation task of the Conference on Statistical Machine Translation (WMT). First, we will describe the system and analyze the translation quality measured in BLEU. Afterwards, we will analyze the performance depending on the frequency of the words and finally show some example translations.

5.1 System description

For the pre-translation, we used a PBMT system. In order to analyze the influence of the quality of the PBMT system, we use two different systems, a baseline system and a system with advanced models. The systems were trained on all parallel data available for the WMT 2016[1]. The news commentary corpus, the European parliament proceedings and the common crawl corpus sum up to 3.7M sentences and around 90M words.

In the baseline system, we use three language models, a word-based, a bilingual (Niehues et al., 2011) and a cluster based language model, using 100 automatically generated clusters using MKCLS (Och, 1999).

The advanced system use pre-reodering (Herrmann et al., 2013) and lexicalized reordering. In addition, it uses a discriminative word lexicon (Niehues and Waibel, 2013) and a language model trained on the large monolingual data.

Both systems were optimized on the tst2014 using Minimum error rate training (Och, 2003). A detailed description of the systems can be found in (Ha et al., 2016).

The neural machine translation was trained using Nematus[2]. For the NMT system as well as for the PreMT system, we used the default configuration. In order to limit the vocabulary size, we use BPE as

[1] http://www.statmt.org/wmt16/translation-task.html

[2] https://github.com/rsennrich/nematus

described in (Sennrich et al., 2016) with 40K operations. We run the NMT system for 420K iterations and stored a model every 30K iterations. We selected the model that performed best on the development data. For the ensemble system we took the last four models. We did not perform an additional fine-tuning.

The PreMT system was trained on translations of the PBMT system of the corpus and the target side of the corpus. For this translation, we only used the baseline PBMT system.

5.2 English - German Machine Translation

The results of all systems are summarized in Table 2. It has to be noted, that the first set, tst2014, has been used as development data for the PBMT system and as validation set for the NMT-based systems.

System	Dev/Valid	Test	
	tst2014	tst2015	tst2016
NMT	20.79	23.34	27.65
NMT Ensemble	21.42	24.03	28.89
PBMT	19.76	21.80	26.42
Advanced PBMT	21.62	23.34	28.13
Pipeline	20.56	22.04	26.75
Pipeline Advanced	21.76	22.92	27.61
Mix	21.88	24.11	28.04
Mix Advanced	22.53	24.37	29.62
Mix Advanced Ensemble	23.16	25.35	30.67

Table 2: Experiments for English→German

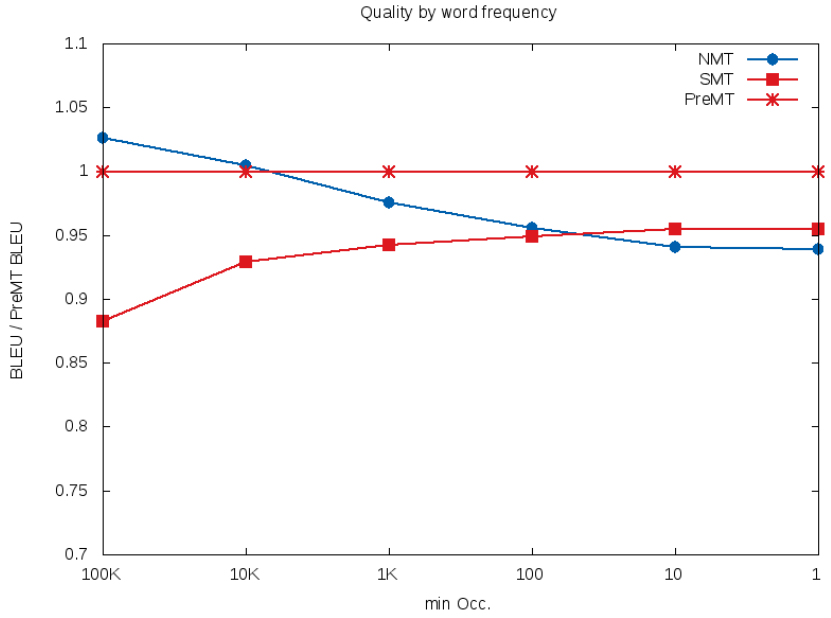
Using the neural MT system, we reach a BLEU score of 23.34 and 27.65 on tst2015 and tst2016. Using an ensemble system, we can improve the performance to 24.03 and 28.89 respectively. The baseline PBMT system performs 1.5 to 1.2 BLEU points worse than the single NMT system. Using the PBMT system with advanced models, we get the same performance on the tst2015 and 0.5 BLEU points better on tst2016 compared to the NMT system.

First, we build a PreMT system using the pipeline method as described in Section 4.1. The system reaches a BLEU score of 22.04 and 26.75 on both test sets. While the PreMT can improve of the baseline PBMT system, the performance is worse than the pure NMT system. So the first approach to combine neural and statistical machine translation is not able the combine the strength of both system. In contrast, the NMT system seems to be not able to recover from the errors done by the SMT-based system.

In a second experiment, we use the advanced PBMT system to generate the translation of the test data. We did not use it to generate a new training corpus, since the translation is computationally very expensive. So the PreMT system stays the same, being trained on the translation of the baseline PBMT. However, it is getting better quality translation in testing. This also leads to an improvement of 0.9 BLEU points on both test sets. Although it is smaller then the difference between the two initial phrase-based translation systems of around 1.5 BLUE points, we are able to improve the translation quality by using a better pre-translation system. It is interesting to see that we can improve the quality of the PreMT system, but improving one component (SMT Pre-Translation), even if we do it only in evaluation and not in training. But the system does not improve over the pure NMT system and even the post editing of the NMT system lowers the performance compared to the initial PBMT system used for pre-translation.

After evaluating the pipelined system, we performed experiments using the mixed input system. This leads to an improvement in translation quality. Using the baseline PBMT system for per-translation, we perform 0.8 BLEU points better than the purely NMT system on tst2015 and 0.4 BLEU point better on tst2016. It also showed better performance than both PBMT systems on tst2015 and comparable performance with the advanced PBMT on tst2016. So by looking at the original input and the pre-translation, the NMT system is able to recover some of the errors done by the PBMT system and also to prevent errors the NMT does if it is directly translating the source sentence.

Figure 2: Compare BLEU score by word frequency



Using the advanced PBMT system for input, we can get additional gains of 0.3 and 1.6 BLEU points The system even outperforms the ensemble system on tst2016. The experiments showed that deploying a pre-translation PBMT system with a better quality improves the NMT quality in the mixed input scheme, even when it is used only in testing, not in training.

By using an ensemble of four model, we improve the model by one BLEU point on both test sets, leading to the best results of 25.35 and 30.67 BLEU points. This is 1.3 and 1.8 BLEU points better than the pure NMT ensemble system.

5.3 System Comparison

After evaluating the approach, we further analyze the different techniques for machine translation. For this, we compared the single NMT system, the advanced PBMT system and the mixed system using the advanced PBMT system as input.

Out initial idea was that PBMT systems are better for translating rare words, while the NMT is generating more fluent translation. To confirm this assumption, we edited the output of all system. For all analyzed systems, we replaced all target words, which occur in the training data less than N times, by the *UNK* token. For large N, we have therefore only the most frequent words in the reference, while for lower N more and more words are used.

The results for $N \in \{1, 10, 100, 1K, 10K, 100K\}$ are shown in Figure 2. Of course, with lower N we will have fewer *UNK* tokens in the output. Therefore, we normalized the BLEU scores by the performance of the PreMT system.

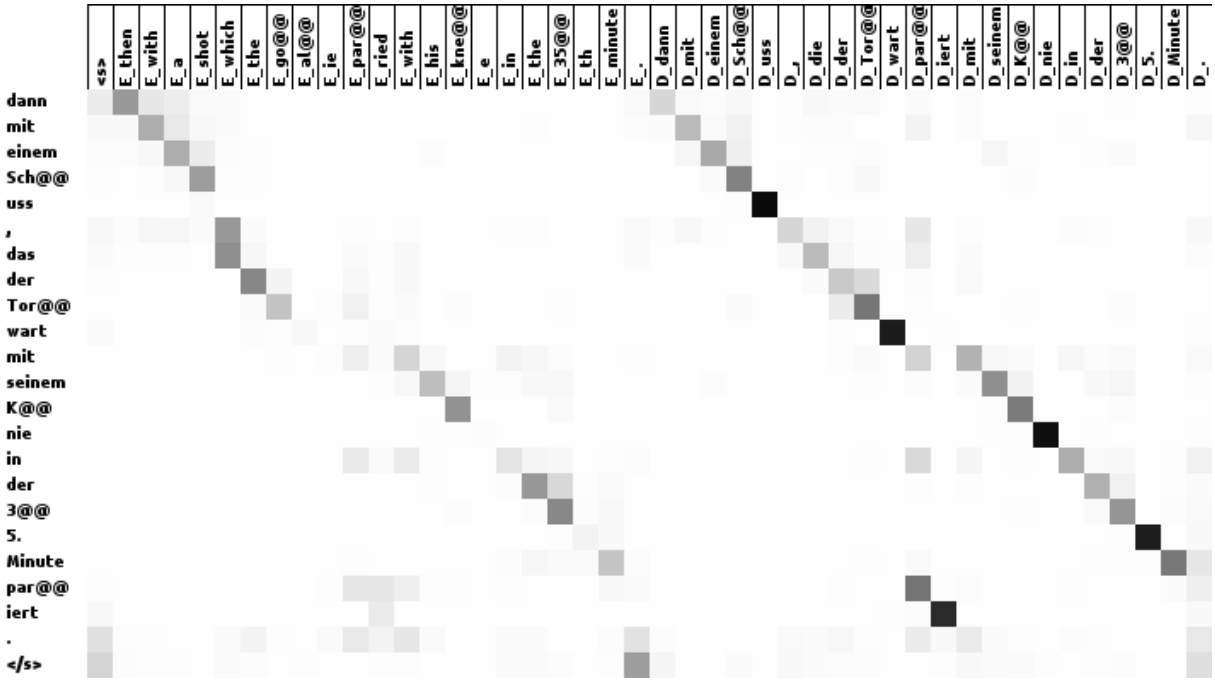
We can see in the figure, that when $N = 100K$, where only the common words are used, we perform best using the NMT system. The PreMT system performs similar and the PBMT system performs clearly worse. If we now decrease N, more and more less frequent words will be considered in the evaluation of the translation quality. Although the absolute BLEU scores raise for all systems, on these less frequent words the PBMT performs better than the NMT system and therefore, finally it even achieves a better performance.

In contrast to this, the PreMT is able to benefit from the pre-translation of the PBMT system and therefore stays better than the PBMT system.

Table 3: Example sentence translated by different techniques

English:	Then with a shot which the **goalie parried** with his knee in the 35th minute.
PBMT:	Dann mit einem Schuss, die der **Torwart pariert** mit seinem Knie in der 35. Minute.
NMT:	Dann mit einem Schuss, den der **Gott** mit seinem Knie in der 35. Minute.
Pre:	Dann mit einem Schuss, das der **Torwart** mit seinem Knie in der 35. Minute **pariert**.
Pre(gloss):	Then with a shoot, that the goali with his knee in the 35th minute parried.

Figure 3: Alignment generated by attention model



5.4 Examples

In Table 3 we show the output of the PBMT, NMT and PreMT system. First, for the PBMT system, we see a typical error when translating from and to German. The verb of the subclause *parried* is located at the second position in English, but in the German sentence it has to be located at the end of the sentence. The PBMT system is often not able to perform this long-range reordering.

For the NMT system, we see two other errors. Both, the words *goalie* and *parried* are quite rarely in the training data and therefore, they are splitted into several parts by the BPE algorithm. In this case, the NMT makes more errors. For the first word, the NMT system generates a complete wrong translation *Gott (engl. god)* instead of *Torwart*. The second word is just dropped and does not appear in the translation.

The example shows that the pre-translation system prevents both errors. It is generating the correct words *Torwart* and *pariert* and putting them at the correct position in the German sentence.

To better understand how the pre-translation system is able to generate this translation, we also generated the alignment matrix of the attention model as shown in Figure 3. The x-axis shows the input, where the words from the pre-translation are marked by $D\_$ and the words from the original source by $E\_$. The y-axis carries the translation. The symbol @@ marks subword units generated by the BPE algorithm. First, as indicated by the two diagonal lines the model is considering as both inputs, the original source and the pre-translation by the two diagonal lines.

Secondly, we see that the attention model is mainly focusing on the pre-translation for words that are not common and therefore got splitted into several parts by the BPE, such as *shoot*, *goalie* and *parried*.

A second example, which shows what happens with rare words occur in the source sentence, is shown in Table 4. In this case, the word *riot* is not translated but just passed to the target language. This

Table 4: Example of rare word translation

English:	... a riot in the stadium.
PBMT:	... einen Aufruhr im Stadion.
NMT:	... einen Riot im Stadion.
Pre:	... einen Aufruhr im Station.
Pre (gloss):	... a riot in_the stadium.

behaviour is helpful for rare words like named entities, but the NMT system is using it also for many words that are not named entities. Other examples for words that were just passed through and not translated are *crossbar* or *vigil*.

6 Conclusion

In this paper, we presented a technique to combine phrase-based and neural machine translation. Motivated by success in statistical machine translation, we used phrase-based machine translation to pre-translate the input and then we generate the final translation using neural machine translation.

While a simple serial combination of both models could not generate better translation than the neural machine translation system, we are able to improve over neural machine translation using a mixed input. By simple concatenation of the phrase-based translation and the original source as input for the neural machine translation, we can increase the machine translation quality measured in BLEU. The single pre-translated system could even outperform the ensemble NMT system. For the ensemble system, the PreMT system could outperform the NMT system by up to 1.8 BLEU points.

Using the combined approach, we can generate more fluent translation typical for the NMT system, but also translate rare words. These are often more easily translated by PBMT. Furthermore, we are able to improve the overall system performance by improving the individual components.

Acknowledgments

The project leading to this application has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n° 645452. This work was supported by the Carl-Zeiss-Stiftung.

References

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Austin, Texas, USA, November.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August. Association for Computational Linguistics.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June.

Loïc Dugast, Jean Senellart, and Philipp Koehn. 2007. Statistical post-editing on systran's rule-based translation system. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 220–223, Prague, Czech Republic, June. Association for Computational Linguistics.

Thanh-Le Ha, Eunah Cho, Jan Niehues, Mohammed Mediani, Matthias Sperber, Alexandre Allauzen, and Alexander Waibel. 2016. The karlsruhe institute of technology systems for the news translation task in wmt 2016. In *Proceedings of the First Conference on Machine Translation*, pages 303–310, Berlin, Germany, August. Association for Computational Linguistics.

Teresa Herrmann, Jan Niehues, and Alex Waibel. 2013. Combining Word Reordering Methods on different Linguistic Abstraction Levels for Statistical Machine Translation. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, Altanta, Georgia, USA.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. In *Proceedings of the First Conference on Machine Translation*, pages 751–758, Berlin, Germany, August. Association for Computational Linguistics.

Philipp Koehn and Kevin Knight. 2003. Empirical Methods for Compound Splitting. In *EACL*, Budapest, Hungary.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 11–19.

Jan Niehues and Alex Waibel. 2013. An MT Error-Driven Discriminative Word Lexicon using Sentence Structure Features. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, Sofia, Bulgaria.

Jan Niehues, Teresa Herrmann, Stephan Vogel, and Alex Waibel. 2011. Wider Context by Using Bilingual Language Models in Machine Translation. In *Sixth Workshop on Statistical Machine Translation (WMT 2011)*, Edinburgh, Scotland, United Kingdom.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Comput. Linguist.*, 30(4):417–449, December.

Franz Josef Och. 1999. An Efficient Method for Determining Bilingual Word Classes. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL 1999)*, Bergen, Norway.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *41st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sapporo, Japan.

Kay Rottmann and Stephan Vogel. 2007. Word Reordering in Statistical Machine Translation with a POS-Based Distortion Model. In *Proceedings of the 11th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI 2007)*, Skövde, Sweden.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Michel Simard, Cyril Goutte, and Pierre Isabelle. 2007. Statistical phrase-based post-editing. In *In Proceedings of NAACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

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

Direct vs. indirect evaluation of distributional thesauri

Vincent Claveau
IRISA - CNRS
Campus de Beaulieu
35042 Rennes, France
vincent.claveau@irisa.fr

Ewa Kijak
IRISA - Univ. of Rennes 1
Campus de Beaulieu
35042 Rennes, France
ewa.kijak@irisa.fr

Abstract

With the success of word embedding methods in various Natural Language Processing tasks, all the fields of distributional semantics have experienced a renewed interest. Beside the famous word2vec, recent studies have presented efficient techniques to build distributional thesaurus; in particular, Claveau et al. (2014) have already shown that Information Retrieval (IR) tools and concepts can be successfully used to build a thesaurus. In this paper, we address the problem of the evaluation of such thesauri or embedding models. Several evaluation scenarii are considered: direct evaluation through reference lexicons and specially crafted datasets, and indirect evaluation through a third party tasks, namely lexical subsitution and Information Retrieval. For this latter task, we adopt the query expansion framework proposed by Claveau and Kijak (2016). Through several experiments, we first show that the recent techniques for building distributional thesaurus outperform the word2vec approach, whatever the evaluation scenario. We also highlight the differences between the evaluation scenarii, which may lead to very different conclusions when comparing distributional models. Last, we study the effect of some parameters of the distributional models on these various evaluation scenarii.

1 Introduction

For years, distributional semantic has aimed at building thesauri (or lexicons) automatically from text corpora. For a given input (ie. a given word), these thesauri identify semantically similar words based on the assumption that they share a distribution similar to the input word's one. In practice, this distributional assumption is set such that two words would be considered close if their occurrences share similar contexts. These contexts are typically co-occurring words in a limited window around the considered words, or words syntactically linked. Recently, many studies have explored new techniques to represent the word (or phrase or document) through embeddings: most often, words are thus represented in a vector space, such that two words with close meanings are close in this space. One of the most popular approach is the famous word2vec technique (Mikolov et al., 2013). Of course, such embedding techniques rely on the same assumption than "traditional" distributional semantics (although many studies do not acknowledge it clearly).

Evaluating these thesauri or embeddings remains a crucial point to assess the quality of the construction methods and parameters used. In this article, we propose to examine this evaluation problem with different evaluation protocols. Indeed, a commonly used approach is to compare the generated thesauri to one or several reference lexicons. This evaluation procedure, called 'intrinsic', has the advantage of being straightforward and simple as it aims at estimating the quality and completeness of the generated thesaurus. However, it is based on reference lexicons whose own completeness, quality, or simply their availability for the considered domain/language/genre are not always granted. Here, we propose to examine the impact of these problems by also using other evaluation protocols, based on other types of reference datasets, and by third-party tasks. In particular, following the work of Claveau and Kijak (2016), we rely on information retrieval (IR) as a realistic evaluation use case. The comparison of the results obtained with these different protocols/datasets should help us to judge the relevance of these assessment scenarios.

After a review of related work (next section), the article addresses the aforementioned subjects: the aspects related to the construction of thesauri are presented in Section 3, while those about the evaluation

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1837–1848, Osaka, Japan, December 11-17 2016.

with specially crafted resources or by IR are respectively discussed in Section 4 and Section 5. Finally, we present some conclusions and perspectives about this work in the last section.

2 Related work

2.1 Building distributional thesauri

Distributional thesauri rely on the famous formula of Firth (Firth, 1957): *"You should know a word by the company it keeps"*. In such thesauri, each word is semantically characterized by all the contexts in which it appears. The semantic neighbors of an entry word are then words whose contexts are similar to that of the entry. Since the pioneering work of Grefenstette (1994) and Lin (1998), many studies have examined distributional thesaurus building. The semantic link considered between an entry word and its neighbors is varied: synonyms, hyperonymy, hyponymy or another (Budanitsky and Hirst, 2006; Adam et al., 2013, for a discussion). Despite their diversity, these links are interesting for many applications related to Natural Language Processing. The various aspects of the thesaurus building is therefore a research field still very active.

One first step concerns the definition of the distributional context of a given word. The graphical contexts simply consider the words appearing around the occurrences of the target word, while syntactic contexts are formed with the syntactic predicates and arguments of the occurrences of the target word. The latter, if it is considered more accurate, requires a prior parsing step which (i) is not always available, (ii) may be misleading.

There are many relationships between distributional semantics and IR. For example, vectorial representations of the contexts are often used (Turney and Pantel, 2010), but unrelated with weighting schemes and relevance functions used in IR (with the exception of Vechtomova and Robertson (2012) in the slightly different context of computing similarities between named entities). However, the weighting of contexts provides more relevant neighbors. Broda et al. (2009) thereby proposed to consider the ranks rather than directly the weights of contexts to overcome the influence of weighting functions. Some bootstrap methods were also proposed to modify the weight of the contexts of a word, by taking into account its semantic neighbors (Zhitomirsky-Geffet and Dagan, 2009; Yamamoto and Asakura, 2010). Another example of relationship between distributional semantics and IR is the use of search engines to collect co-occurrence information or contexts on the web (Turney, 2001; Bollegala et al., 2007; Sahami and Heilman, 2006; Ruiz-Casado et al., 2005). Finally, given that the "traditional" distributional representation of contexts is sparse and redundant (Hagiwara et al., 2006), several methods for dimension reduction were tested: from Latent Semantic Analysis (Landauer and Dumais, 1997b; Padó and Lapata, 2007; Van de Cruys et al., 2011) to *Random Indexing* (Sahlgren, 2001), through factorization by non-negative matrices (Van de Cruys, 2010).

The problem of the construction of distributional thesaurus may be expressed in a simple way as a conventional IR problem (Claveau et al., 2014). All contexts of a target word are then represented as a document (or query), and distributional neighbors of the target word are the sets of similar contexts. This formulation of distributional neighbors search process offers interesting research tracks and easily accessible tools.

2.2 Tested models

In this paper, several distributional models are tested. A first group of models, that may be called traditional distributional models are considered. In the following sub-section we report results obtained by a state-of-the art approach, hereafter denoted *base*, that uses a cosine similarity and weighting by mutual information (Ferret, 2013), an improved version (*rerank*) which uses machine learning technique to rerank neighbors (Ferret, 2013), and another version (*synt*) based on syntactic contexts (Ferret, 2014) rather than graphic ones are also tested.

As explained in Claveau et al. (2014), the problem of building a distributional thesaurus can be translated into a problem of searching similar documents and can therefore be carried out with IR techniques. In this context, all the contexts of a given word in a corpus are collected; this set of contexts forms what is considered as a document. Building an entry in the thesaurus, ie. finding the closest words (in a dis-

tributional sense) of a word w_i, is thus equivalent to finding documents (contexts) close to the document representing the contexts of w_i (seen as a query in the IR system). This has led to new distributional models, that differ from the previous ones by the way the similarity between two words (or their contexts) is computed. We also report the results of systems based on this IR approach. Several variants, each using a different way to compute the similarity between the sets of context, are considered: namely TF-IDF/cosine and Okapi-BM-25 (Robertson et al., 1998). In previous work, we also proposed an adjusted versions of them *adjusted-TF-IDF, adjusted-Okapi BM25*, in which the influence of the document size is reinforced in order to give more importance to the most discriminating context words (Claveau et al., 2014). Other IR systems are tested; they are based on probabilistic language modeling (denoted LM), with both Dirichlet smoothing (varying the values of the parameter μ) and Hiemstra smoothing (smoothing with the probabilities of occurrence of words throughout the collection; with different values of λ) (Claveau and Kijak, 2016). All these very classical IR models are not detailed further here; the interested reader will find the concepts and useful details (such as the role of the parameters) in the cited references or IR surveys (Manning et al., 2008, for example).

As a last group of models, we consider approaches based on dimensionality reduction. In such models, the data is represented in dense vector space, usually of small dimensionality (for instance, $\mathbb{R}^{500}$). We report results yielded by models based on usual dimension reduction techniques (LSI, LDA, Random projections (RP)), with different numbers of dimensions. In this group, we also consider the very popular embedding approaches, namely Word2Vec (Mikolov et al., 2013). Indeed, they have been shown to be equivalent to standard distributional models with an additional dimensionality reduction step (Levy and Goldberg, 2014). For comparison purpose, we also indicate the results of a word2vec model pre-trained on the Google News corpus (which is significantly larger than AQUAINT-2); it is freely available at `https://code.google.com/p/word2vec/`.

2.3 Evaluating distributional thesauri

As already mentioned, the evaluation of a thesaurus can be done in two ways: (i) the intrinsic evaluation consists in comparing the produced thesaurus with a reference resource; (ii) the extrinsic evaluation assesses the thesaurus through its use in a given task.

The intrinsic evaluation requires to have reference lexicons. Usual lexicons used as references are WordSim 353 (Gabrilovich and Markovitch, 2007), WordNet 3.0 (Miller, 1990) or Moby (Ward, 1996). The two latter exploit larger resources (as synonyms) and are those used in this work for the intrinsic evaluation, as in Ferret (2013). Other data sets, as the set of synonyms from the TOEFL test (Landauer and Dumais, 1997a) or the semantic relationships in BLESS (Baroni and Lenci, 2011), are not directly lexicons, but can also be used for direct evaluation. Given a reference lexicon, it is then easy to compute recall, accuracy or any other measure of quality.

If the intrinsic evaluation is simple, its relevance depends on the adequacy of the lexicons used as references. For this reason, several studies have suggested extrinsic evaluation through a task, such as the lexical substitution task proposed at SemEval 2007 (McCarthy and Navigli, 2009). The goal is to replace a word in a sentence by a neighbor (given by the evaluated thesaurus) and verify that it did not change the meaning of the sentence, by comparing the obtained results to the substitutions proposed by humans. In such a task, the exact synonyms are favored over other types of semantic relationships.

Several studies use distributional information within an IR framework (Besançon et al., 1999; Billhardt et al., 2002), like recent lexical representations such as word2vec (Huang et al., 2012; Mikolov et al., 2013). The aim is to improve the representation of documents and/or the Relevance Status Value function (RSV, ie. the function used in IR systems to rank the answers to a query according to their supposed relevance), by exploiting the similarities between word contexts. Nevertheless, the process of creating the distributional thesaurus is not dissociated from the IR process in these studies, which makes the evaluation of the only distributional information contribution impossible. Recently, we have proposed to specifically evaluate the distributional thesauri in IR by using semantic neighbors to expand the queries (Claveau and Kijak, 2016). In this paper, we adopt the same framework in Section 5.

3 Intrinsic Evaluation of distributional models

3.1 Principles and material

For the sake of comparison with published results, the data used for our experiments are those used in several studies. The corpus used to collect the contexts is AQUAINT-2 (Vorhees and Graff, 2008); it is composed of articles in English containing a total of 380 millions of words. The words considered for our thesaurus entries are common nouns occurring at least 10 times in the corpus, that is 25 000 different nouns. The contexts of all occurrences of these words are collected; in the experiments reported below, contexts are formed by the two words at the right and two words at the left of the target noun, along with their position. For example, in the sentence "... all forms of restriction on freedom of expression, threats ..." the words restriction-2, on-1, of+1, expression+2 are added to the set of contexts of freedom.

As we mentioned earlier, we use WordNet (WN) and Moby for intrinsic assessment of generated thesauri. These two resources have different, additional characteristics: WN identifies strong semantic links (synonyms or quasi-synonyms) while Moby identifies a greater variety of links (hypernyms, meronyms, co-hyponymy...). WN offers on average 3 neighbors for 10 473 nouns of AQUAINT-2, and Moby contains on average 50 neighbors of 9 216 nouns. Together, these resources cover 12 243 nouns of the corpus with 38 neighbors on average. These resources are used as reference for the evaluation; more details about the semantic links considered by these resources and their use for distributional thesaurus evaluation can be found in the literature (Ferret, 2013; Claveau et al., 2014). The number of nouns and the variety of semantic relations that they contain make these references a comprehensive evaluation data set, compared with other existing benchmarks (such as WordSim 353 (Gabrilovich and Markovitch, 2007) for instance).

3.2 Intrinsic evaluation results

Figure 1 presents the results obtained by different thesaurus building systems presented in Section 2.2, applied to the AQUAINT-2 corpus. The performance measures used to compare the generated thesauri with the reference (WordNet + Moby, denoted by WN+M) are those typically used for this task: precision at different levels (on the top 5, 10, 50, 100 neighbors), MAP (Mean Average Precision) and R-precision, expressed as a percentage, averaged on the 12 243 nouns in the WN+M reference. For all these distributional models, among all the parameters tested, we only report some of the best performing ones in terms of MAP (number of dimensions, size of the context window...).

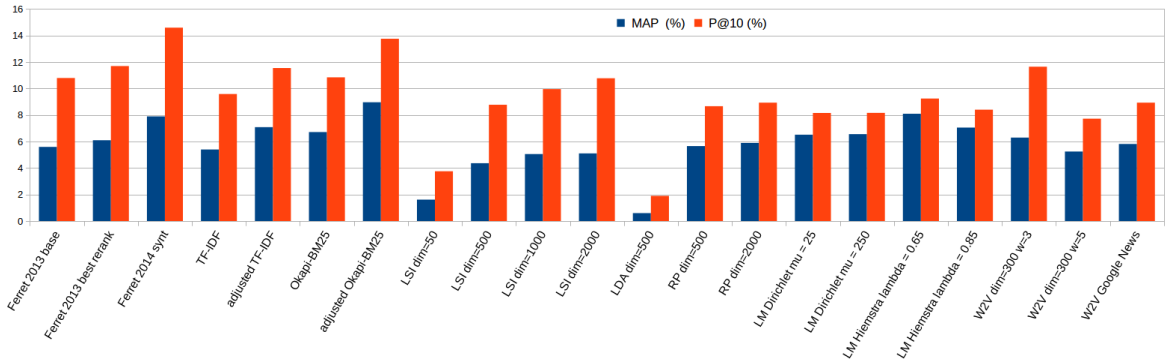


Figure 1: Performance of various distributional and embedding models for building distributional thesauri over the WN+M reference

As already mentioned in the literature, this kind of evaluation with reference lexicons leads to very severe conclusions about the quality of the evaluated distributional thesauri. For instance, the P@10 score states that, in average, 9 out the 10 first neighbors of an entry word are errors, whatever the model. Nonetheless, it is worth noting that some traditional models, and in particular recent IR-based ones such as adjusted Okapi-BM25, obtain better results than the popular word2vec ones. Overall, dimension reduction techniques yields low results: The lower the number of dimensions considered, the worse the

Method	Pearson's r	Spearman's ρ	Kendal's τ
adjusted Okapi	-0.0009 (p=0.9723)	0.3148 (p=5.0e-32)	0.2093 (p=2.5e-30)
W2V dim=300 w=3	0.0027 (p=0.9314)	0.2913 (p=1.4e-21)	0.1944 (p=9.7e-21)

Table 1: Correlation coefficients (with their p-values) between the Average Precision (AP) of each word on the WN+M dataset and its frequency

Method	Pearson's r	Spearman's ρ	Kendal's τ
adjusted Okapi	-0.0923 (p=0.0007)	0.0604 (p=0.0274)	0.0465 (p=0.0111)
W2V dim=300 w=3	-0.1218 (p=9.01e-05)	0.0657 (p=0.0352)	0.0479 (p=0.0213)
W2V GoogleNews	-0.1609 (p=5.08e-09)	-0.1848 (p=1.77e-11)	-0.1374 (p=1.07e-13)

Table 2: Correlation coefficients (with their p-values) between the Average Precision (AP) of each word on the WN+M dataset and its polysemy

results. This negative result is in line with some conclusions of previous work (Van de Cruys, 2010). The occurrence of certain very specific contextual words is indeed a strong indicator of the semantic proximity of words. Aggregation of different words into a single dimension is then detrimental to distinguish the semantic neighbors. This is also confirmed by the fact that within a model family, the parameter settings leading to the best results are those which give more weight to discriminating words: squared IDF for Okapi, very few smoothing for language modeling (ie. low values of μ and λ).

3.3 Influence of data characteristics

It is interesting to examine how some characteristics of the data may influence the results. For instance, it has already been noted (Ferret, 2013) that the frequency of words for which we try to find the neighbors has a great influence on the final quality. This is easily explained by the fact that the more frequent the nouns are, the more contexts they have to describe them; and finally, the better the results are. In order to verify what is the actual effect on our results, given our data and methods, we estimate the correlation between the entry quality in the distributional thesaurus, measured by the Average Precision on the WN+M dataset, and the its frequency in the AQUAINT-2 corpus; results are given in Table 1.

As expected, these results confirm the effect of the frequency on the entry quality. Yet, it should be noted that this correlation is not perfect; other characteristics may interfere on the results.

Another data property that may influence the results is polysemy. Indeed, all the methods evaluated in Section 3.2 consider that all the occurrences of a word (or lemma) should result in one thesaurus entry, that is, no disambiguation is performed. Thus, one could suspect that the list of distributional neighbors of a polysemic word would be impacted. As we have done before, to verify this supposition, we estimate the correlation between the entry quality (AP on the WN+M reference) and the number of senses for that entry as encoded in WordNet. Results are given in Table 2. For comparison purpose, we also report the results obtained with the word2vec GoogleNews model.

The effect of polysemy on the thesaurus results is not obvious. On the one hand, one can observe that there is no correlation between the results and the number of word senses for the models trained on our corpus. On the other hand, the word2vec GoogleNews model shows a statistically significant negative correlation; it means that the less polysemic words tends to yield the best results.

4 Evaluating with specially crafted resources

The evaluation through reference lexicons, as presented above, has several shortcomings already mentioned in Section 2. In addition to these, it is worth noting that resources such as WordNet or Moby were not initially designed for evaluation, and choices made for their building are not necessarily adequate for our evaluation task. For instance, there is no graduation: for a given semantic relation two words are either related or not. In order to provide more relevant evaluation resources, other direct evaluation of distributional thesauri were proposed. In this section, we consider SimLex999 (Hill et al., 2014) that was specially developed to evaluate distributional models. One of its most interesting particularities is that,

according to the authors: "it explicitly quantifies similarity rather than association or relatedness, so that pairs of entities that are associated but not actually similar have a low rating".

4.1 Experimental setting

SimLex999 is a resource in which pairs of words (nouns, verbs, adjectives) are given a score according to their association strength. This score was given by a group of annotators with light instructions on what is to be considered as "association" (Hill et al., 2014, Sec. 3.3). This is intended to reflect the light formalization of the semantic links captured by distributional models.

The evaluation based on this resource aims at comparing the word pair list sorted by association strength and the word pair sorted according to the distributional score. In practice, the Spearman's ρ rank correlation is used. A review of the most recent results obtained with this dataset can be found at `http://www.cl.cam.ac.uk/~fh295/simlex.html`.

4.2 Results

Table 3 shows the SimLex999 results of the best traditional model (adjusted Okapi-BM25) and the best word2vec model. For comparison purpose, we also report the results obtained with the freely available word2vec model trained on the GoogleNews corpus. One can observe a very important difference between the Okapi model and the word2vec trained on the AQUAINT-2 corpus. The GoogleNews model also yields a good score, but one has to keep in my mind that it was trained on a larger corpus than ours.

Method	Spearman's ρ
adjusted Okapi	0.4511
W2V dim=300 w=3	0.3691
W2V GoogleNews	0.4419

Table 3: Correlation coefficients (with their p-values) of distributional models on the SimLex999 dataset

4.3 Influence of data characteristics

As done for the previous evaluation scenario, it is interesting to examine how the SimLex999 results are impacted by the data characteristics. In Figure 2, we report the evolution of Spearman's ρ according to the frequencies of the words considered; ρ is computed on subsets of the SimLex999 pairs whose frequencies are both under a given threshold. The model used here is word2vec trained on the AQUAINT-2 corpus with dim=300 and w=3. One can observe that there are some variations when considering the lowest frequencies; they are due to the small number of SimLex pairs which make the correlation computation very sensitive. Beside that, overall, the frequency does not seem to play an active role in the performance, which seems to contradict what was observed in Section 3.3. Yet, it is worth noting that the less frequent SimLex999 words are not rare in our corpus (more than 2,000 occurrences for the rarest one). SimLex999 contains only frequent or very frequent words and thus does not adequately evaluate the capacity of the model to handle rarer words.

Figure 3 adopts the same setting and presents the impact of polysemy on the SimLex999 performance: ρ is computed on subsets of the SimLex999 pairs whose sum of senses, as encoded in WordNet, is under a given threshold. Here, the effect of polysemy appears very clearly: polysemic words have a negative impact on the capacity of the model to encode the similarity between words as measured by the SimLex999 dataset. This result is not surprising, but it did not appeared in the previous lexicon-based evaluation scenario due to the protocol used.

Finally, these two direct evaluation scenarii, using reference lexicon or specially crated datasets, give dissimilar results, and are not impacted by the same data characteristics.

5 Indirect evaluation through query expansion

Following our previous work (Claveau and Kijak, 2016), in this section we use an IR task as a way to evaluate distributional models. More precisely, we use distributional thesauri to expand queries: for each

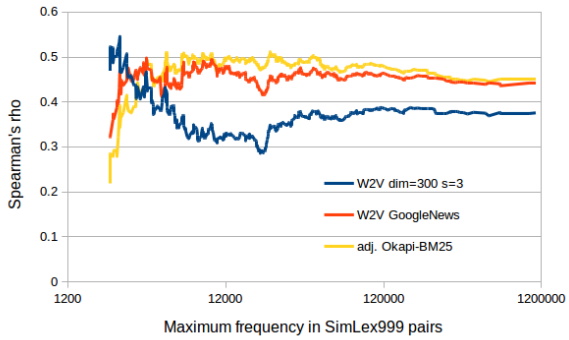


Figure 2: Performance on the SimLex999 dataset according to the frequency of the words considered; log-scale

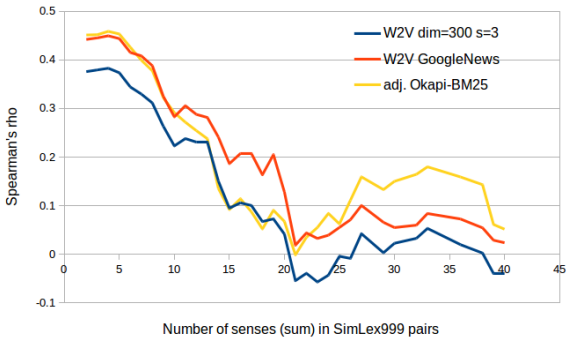


Figure 3: Performance on the SimLex999 dataset according to the sum of the sense of the words in the considered pairs

query noun, its neighbors found in the considered thesaurus are added to the query. The experimental framework and the results obtained are successively presented below.

5.1 Experimental setting

The IR collection used in the experiments comes from the Tipster project and was used as part of TREC. It contains more than 170 000 documents and 50 queries in English (the queries are structured: the query itself, a narrative field detailing the criteria of relevance; in the experiments reported below, we only use the query field). This collection is particularly suited since its documents come from the *Wall Street Journal* and are similar to those of AQUAINT-2.

The IR system used is Indri (Metzler and Croft, 2004; Strohman et al., 2005). This probabilistic system implements a combination of language modeling (Ponte and Croft, 1998) (as the ones used in Sect. 3) and inference networks (Turtle and Croft, 1991); it is known to provide state-of-the-art results. In the experiments reported below, we use its standard settings, ie. Dirichlet smoothing (with $\mu = 2500$ as recommended). In our case, Indri offers an additional advantage: it has a complex query language that allows us to include the words of the distributional thesaurus by making best use of the inference network model; in practice, the dedicated operator `'#syn'` is used to aggregate the counts of the words indicated as synonyms (see Indri documentation for details). To remove the effects of inflection on the results, the plural and singular forms of nouns of the queries are added, either in the non-extended, original queries or those extended with the semantic neighbors. As example, we give below a sample query, with its non-expanded form and its expanded form (adjusted Okapi top 5) using the inference network operators of Indri:

– query : `coping with overcrowded prisons`
– normal form : `#combine( coping with overcrowded #syn( prisons prison ) )`
– expanded form : `#combine( coping with overcrowded #syn( prisons prison inmate inmates jail jails detention detentions prisoner prisoners detainee detainees ) )`

The performance for this IR task is typically measured by precision at different thresholds (P@x), R-precision, and MAP. Therefore, to evaluate the thesaurus, we measure the gains in terms of precision, MAP, etc. between the results without and with expansion. We also indicate the average of the AP (Average Precision) gain by query, noted AvgGainAP (not be confused with the gain of MAP, which is the gain calculated from the AP averages over the query). Non statistically significant results (Wilcoxon and t-test with $p < 0.05$) are in italics.

5.2 Expansion results

Table 4 presents the performance gains achieved by expanding the queries with the words collected in the thesaurus (the ones used in the previous experiments). Here we only report results when expanding

with the top 10 nearest neighbors (other settings lead to similar conclusions ; see (Claveau and Kijak, 2016)). We also show the results obtained by expanding the queries with the reference lexicons (WN alone and WN+M).

Expansion	MAP	AvgGainAP	R-Prec	P@5	P@10	P@50	P@100
without	21.78	-	30.93	92.80	89.40	79.60	70.48
with WN	+12.44	+36.3	+7.01	+4.31	+7.16	+7.60	+10.87
with WN+M	+11.00	+28.33	+7.78	*+3.02*	+5.37	+6.53	+9.17
with adjusted Okapi top 10	+13.80	+24.36	+9.58	*+2.16*	+4.03	+5.58	+8.26
with W2V dim=300 s=3 top 10	+5.20	+17.83	+4.75	*+1.29*	*+2.68*	+4.32	+5.16
with W2V GoogleNews top 10	+13.70	+30.52	+9.52	*+3.02*	*+3.58*	+8.14	+10.19

Table 4: Relative gain of performance (%) when expanding queries with different thesauri

First, we note that for any thesaurus used, the query expansion brings a significant gain in performance. By the way, it contradicts the conclusions of (Voorhees, 1994) about the alleged lack of interest in using WN to expand queries. The most notable fact here is the excellent results obtained with the thesaurus built automatically (traditional or word2vec), that even exceed those of the reference lexicons. While its precision on the first 10 neighbors was evaluated under 14% in Section 3, the adjusted Okapi-BM25 and word2vec thesauri generate expansions yielding the better MAP gains. The average AP gain (AvgGainAP) also provides interesting information: it is maximum with WN, which therefore provides a stable improvement (gain for most queries). This is due to the fact that the words added to the query by WN are synonyms, which presents a low risk. This stability is lower with other thesauri; as the MAP gain remains generally good, it indicates that only certain queries benefit from significant absolute gains.

6 Comparing evaluation results

6.1 Overview

The results of the previous experiences raise questions about the consistency between the lexicon based, similarity-based and task-based evaluations. We want to know, for example, if the P@10 on the WN-based evaluation between two thesaurus construction methods, even if stated as statistically significant, is sensible when now using WN+M as a reference dataset, or in IR, or in the SimLex999 evaluation. We also add the results obtained on the lexical substitution framework proposed in SemEval 2007 (McCarthy and Navigli, 2009). In this task, the distributional thesauri are evaluated on their ability to provide words that are judged as similar in a specific context. In the experiments reported below, we use a very simple strategy: the 10 closest neighbors are proposed, whatever the context. The results are evaluated in terms of precision (referred as the out-of-ten precision in SemEval 2007).

Let us consider Figure 4. The performance on the different evaluation protocols is reported for four distributional models used before: the adjusted Okapi-BM25, word2vec (dim=300, w=3), word2vec (dim=300, w=9) and the GoogleNews word2vec model. Several things are worth noting from this figure. Some models (adjusted Okapi-BM25 for instance) outperform others (for instance, word2vec) on every evaluation protocol. Yet, large difference for one evaluation protocol does not lead to large margin in another evaluation (consider for instance P@10 on the IR task versus the SimLex999 correlation scores). When considering models with more similar results, one can even see that one model can be better at a task and worse at another, sometimes with significant margins (for instance, consider the two word2vec models with s=3 and s=9).

This latter point raises questions on the validity of certain evaluation protocols. In previous work (Claveau and Kijak, 2016), we have shown that the precision of the thesaurus, as measured by a comparison with exiting lexicons, is largely under-estimated when considering the IR-based evaluation. It is mainly caused by the incompleteness of the lexicons used in the intrinsic evaluation. Yet, in this previous work, there was a correlation between these two evaluation protocols (models obtaining the best results for IR yielded the best results for the lexicon-based evaluation, etc.). As seen in Figure 4 with the adjusted Okapi and GoogleNews models, this is not the case in general.

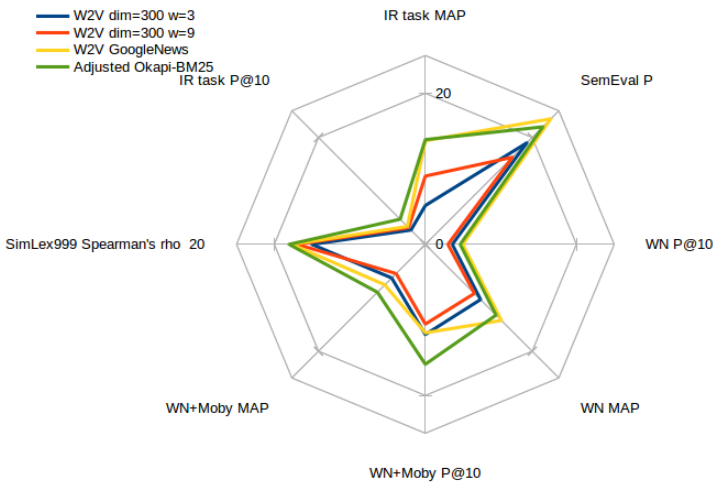


Figure 4: Performance of the adjusted Okapi and word2vec models trained on the AQUAINT-2 corpus; for comparison purpose, the results of the GoogleNews word2vec model are also reported.

6.2 Model parameters

One could argue that the previous results are due to the different approaches (or training corpus in the case of the GoogleNews model) used by these models. In order to confirm or infirm that, we study the evolution of all our evaluation scores according to one specific parameter for a given model (here, word2vec). Figure 5 shows the performance evolution on the different evaluation scenarii according to the dimensionality (respectively the size of the context window). Several points are worth noting. First, this figure highlights here again the fact that the best model (or set of parameters) for one evaluation scenario is not necessarily the best for another. A small context window yields the best results for SimLex999, while a maximum is reached at s=3 when evaluating with WN (strict synonymy) or WN+M (large range of semantic relations). Concerning the IR task, the window size has few effect on P@10 but has a great impact on the MAP. This is due to the fact that the terms used as expansion barely modify the 10 first documents retrieved by the search engines, but have a greater impact on the whole list of retrieved documents (expansion helps finding documents that share no common words with the original queries). Large windows help identifying words appearing in the same documents than the query words (much like LSI would do) and thus benefits to the MAP.
Using a small dimension is detrimental to every task; The resulting vector space cannot represent adequately the word semantics. Conversely, a large number of dimensions does not allow the necessary generalization needed for retrieving new documents (IR MAP) or the large variety of relations used in WN+M. Thus, for these evaluation scenarii, a dimension between 300 and 500 appears as a good compromise. But for the SimLex999 and the WN-based evaluation, a large number of dimension will allow a more precise description of the word: such setting is more suited to detect synonymy or close semantic links.

7 Conclusion

In this article, following the work of Claveau and Kijak (2016), we compared different scenarii to evaluate distributional and embedding models. Beside the intrinsic evaluation through reference lexicons (here, WordNet and WordNet+Moby) and specialy crafted data (SimLex999), we also rely on an extrinsic evaluation with an IR task, as initially proposed by Claveau and Kijak (2016).

In this work, several conclusions are worth noting. The main one is certainly that direct, or intrinsic, evaluation (be it with reference lexicons or specially crafted data) should be avoided if possible. Indeed, the thesaurus characteristics they evaluate are unclear and may be very different from one's specific need. This is particularly obvious when comparing the IR task results with those of the WN+Moby evaluation (Section 5 and see also (Claveau and Kijak, 2016) for further discussion). Indeed, the very weak results

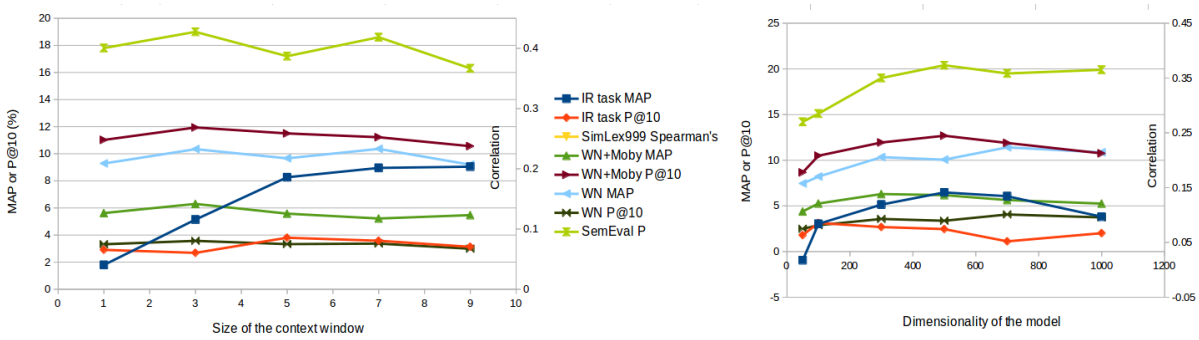


Figure 5: Influence of context size and dimensionality in word2vec models on the results of different evaluation protocols. On the left,the size of the context window is set to 3; on the right, the dimensionality is set to 300.

of the generated thesaurus at the lexicon-based evaluations are not confirmed in the third-party evaluation framework (in our case, query expansion for IR).

Beside that, the evaluation resources (WordNet, Moby or SimLex999) are not complete enough to provide reliable results. For instance, by showing that the thesaurus generated with our models obtains extrinsic results at least as good as the reference lexicons (WN and Moby) used for the intrinsic evaluation, we question previous conclusions of many studies only based on intrinsic evaluation. This incompleteness problem also exists with SimLex999 since it focuses on frequent words (cf. Section 4.3), which are known to be the easiest to model with distributional approaches, but are not necessarily the most interesting for one's needs. Indirect, or task-based, evaluation (lexical substitution or IR) seems conceptually more grounded, but the datasets used may suffer from the same problem of incompleteness or non-representativity. Another related conclusion remark, is that the model parameters have very different effects depending on the considered evaluation scenario (cf. Section 6.2). It is thus important to fine tune with respect to the final task rather than on unrelated datasets.

As a side results, all the experiments presented here confirm the interest of using the IR approaches for building distributional thesaurus (Claveau et al., 2014). In particular, the adjusted Okapi-BM25 model offers significant gains of performance over word2vec for most of the evaluation scenarii considered.

Acknowledgements

This work was partly funded by French government supports granted to the FUI project NexGenTV and to the CominLabs excellence laboratory and managed by the National Research Agency in the "Investing for the Future" program under reference ANR-10-LABX-07-01.

References

Clémentine Adam, Cécile Fabre, and Philippe Muller. 2013. Évaluer et améliorer une ressource distributionnelle : protocole d'annotation de liens sémantiques en contexte. *TAL*, 54(1):71–97.

M. Baroni and A. Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10.

Romaric Besançon, Martin Rajman, and Jean-Cédric Chappelier. 1999. Textual similarities based on a distributional approach. In *in Proceedings of the Tenth International Workshop on Database and Expert Systems Applications (DEXA'99)*, pages 180–184.

Holger Billhardt, Daniel Borrajo, and Victor Maojo. 2002. A context vector model for information retrieval. *J. Am. Soc. Inf. Sci. Technol.*, 53(3):236–249, February.

D. Bollegala, Y. Matsuo, and M. Ishizuka. 2007. Measuring semantic similarity between words using web search engines. In *Proceedings of WWW'2007*.

Bartosz Broda, Maciej Piasecki, and Stan Szpakowicz. 2009. Rank-Based Transformation in Measuring Semantic Relatedness. In *22nd Canadian Conference on Artificial Intelligence*, pages 187–190.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating WordNet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

Vincent Claveau and Ewa Kijak. 2016. Distributional thesauri for information retrieval and vice versa. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Vincent Claveau, Ewa Kijak, and Olivier Ferret. 2014. Improving distributional thesauri by exploring the graph of neighbors. In *International Conference on Computational Linguistics, COLING 2014*, Dublin, Irlande, August.

Olivier Ferret. 2013. Identifying bad semantic neighbors for improving distributional thesauri. In *51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 561–571, Sofia, Bulgaria.

Olivier Ferret. 2014. Typing relations in distributional thesauri. In N. Gala, R. Rapp, and G. Bel, editors, *Advances in Language Production, Cognition and the Lexicon*. Springer.

John R. Firth, 1957. *Studies in Linguistic Analysis*, chapter A synopsis of linguistic theory 1930-1955, pages 1–32. Blackwell, Oxford.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 6–12.

Gregory Grefenstette. 1994. *Explorations in automatic thesaurus discovery*. Kluwer Academic Publishers.

Masato Hagiwara, Yasuhiro Ogawa, and Katsuhiko Toyama. 2006. Selection of effective contextual information for automatic synonym acquisition. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006)*, pages 353–360, Sydney, Australia.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Evaluating semantic models with (genuine) similarity estimation.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *50th Annual Meeting of the Association for Computational Linguistics (ACL'12)*, pages 873–882.

Thomas Landauer and Susan Dumais. 1997a. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

Thomas K. Landauer and Susan T. Dumais. 1997b. A solution to Plato's problem: the latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211–240.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics (ACL-COLING'98)*, pages 768–774, Montréal, Canada.

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

Diana McCarthy and Roberto Navigli. 2009. The English lexical substitution task. *Language Resources and Evaluation*, 43(2):139–159.

D. Metzler and W.B. Croft. 2004. Combining the language model and inference network approaches to retrieval. *Information Processing and Management Special Issue on Bayesian Networks and Information Retrieval*, 40(5):735–750.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2013)*, pages 746–751, Atlanta, Georgia.

George A. Miller. 1990. WordNet: An On-Line Lexical Database. *International Journal of Lexicography*, 3(4).

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

J. M. Ponte and W. B. Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual international ACM SIGIR Conference on Research and Development in information Retrieval (SIGIR '98)*, pages 275–281.

Stephen E. Robertson, Steve Walker, and Micheline Hancock-Beaulieu. 1998. Okapi at TREC-7: Automatic Ad Hoc, Filtering, VLC and Interactive. In *Proc. of the 7th Text Retrieval Conference, TREC-7*, pages 199–210.

M Ruiz-Casado, E. Alfonseca, and P. Castells. 2005. Using context-window overlapping in synonym discovery and ontology extension. In *Proceedings of RANLP-2005*, Borovets, Bulgarie.

M. Sahami and T.D. Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of WWW'2006*.

Magnus Sahlgren. 2001. Vector-based semantic analysis: Representing word meanings based on random labels. In *ESSLLI 2001 Workshop on Semantic Knowledge Acquisition and Categorisation*, Helsinki, Finland.

T. Strohman, D. Metzler, H. Turtle, and W.B. Croft. 2005. Indri: A language-model based search engine for complex queries (extended version). Technical report, CIIR.

P. Turney and P. Pantel. 2010. From frequency to meaning : Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

P.D. Turney. 2001. Mining the web for synonyms: Pmiir versus lsa on toefl. *Lecture Notes in Computer Science*, 2167:491–502.

H. Turtle and W.B. Croft. 1991. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information System*, 9(3):187–222.

T. Van de Cruys, T. Poibeau, and A. Korhonen. 2011. Latent vector weighting for word meaning in context. In Association for Computational Linguistics, editor, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1012–1022.

Tim Van de Cruys. 2010. *Mining for Meaning. The Extraction of Lexico-semantic Knowledge from Text*. Ph.D. thesis, University of Groningen, The Netherlands.

Olga Vechtomova and Stephen E. Robertson. 2012. A domain-independent approach to finding related entities. *Information Processing and Management*, 48(4):654–670.

Ellen M. Voorhees. 1994. Query expansion using lexical-semantic relations. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 61–69, New York, NY, USA. Springer-Verlag New York, Inc.

Ellen Vorhees and David Graff. 2008. Aquaint-2 information-retrieval text research collection.

Grady Ward. 1996. Moby thesaurus. Moby Project.

Kazuhide Yamamoto and Takeshi Asakura. 2010. Even unassociated features can improve lexical distributional similarity. In *Second Workshop on NLP Challenges in the Information Explosion Era (NLPIX 2010)*, pages 32–39, Beijing, China.

Maayan Zhitomirsky-Geffet and Ido Dagan. 2009. Bootstrapping Distributional Feature Vector Quality. *Computational Linguistics*, 35(3):435–461.

D-GloVe: A Feasible Least Squares Model for Estimating Word Embedding Densities*

Shoaib Jameel and Steven Schockaert
School of Computer Science and Informatics,
Cardiff University.
{JameelS1, S.Schockaert}@cardiff.ac.uk

Abstract

We propose a new word embedding model, inspired by GloVe, which is formulated as a feasible least squares optimization problem. In contrast to existing models, we explicitly represent the uncertainty about the exact definition of each word vector. To this end, we estimate the error that results from using noisy co-occurrence counts in the formulation of the model, and we model the imprecision that results from including uninformative context words. Our experimental results demonstrate that this model compares favourably with existing word embedding models.

1 Introduction

Several vector space models for word meaning have already been proposed (Lund and Burgess, 1996; Landauer and Dumais, 1997; Turney and Pantel, 2010; Mikolov et al., 2013; Pennington et al., 2014). While there are considerable differences in how these vector space models are learned, most approaches represent words as vectors. However, a few authors have proposed models that represent words as regions or densities in a vector space (Erk, 2009; Vilnis and McCallum, 2015), motivated by the view that region or density based representations are better suited to model the diversity of word meaning, and can thus capture e.g. hyponymy in a natural way. In this paper, we also use densities in a low-dimensional vector space to represent word meaning. In contrast to previous work, however, we use densities for modelling our lack of knowledge about the precise meaning of a word. This allows us to use a more cautious representation for rare words, and leads to better confidence estimates in downstream tasks. Note that this use of densities is indeed fundamentally different from its use in previous work. For example, increasing the corpus size in our case will lead to more precise estimates (i.e. distributions with lower variance) while the models from (Erk, 2009; Vilnis and McCallum, 2015) may arrive at distributions with higher variance, reflecting the broader set of context windows that may be found.

Our approach is based on the GloVe model for word embedding (Pennington et al., 2014). In particular, we also associate two vectors with each word i: the vector w_i, which intuitively represents the meaning of word i, and the vector $\tilde{w}_j$, which intuitively represents how the occurrence of i in the context of another word j affects the meaning of that word. Moreover, we also use a least squares formulation to constrain these vectors such that $w_i \cdot \tilde{w}_j$ reflects the co-occurrence statistics of words i and j. In contrast to GloVe, however, we explicitly model two factors that contribute to the residual error of this model: (i) the fact that corpus statistics for rare terms are not reliable and (ii) the fact that not all words are equally informative. This has two key advantages. First, it allows us to formulate the underlying optimization problem as a feasible generalized least squares problem. As we show in our experiments, this leads to word embeddings (as vectors) that substantially outperform those obtained from the GloVe model, and other baselines, in standard word similarity and analogy tasks. Second, it allows us to explicitly represent our uncertainty about the precise definitions of the word vectors. Rather than using w_i for modelling the meaning of word i, we then consider a density which is defined by the residual error model.

Specifically, the residual error model allows us to naturally associate a univariate density with each context vector $\tilde{w}_j$, given a target word i. A natural geometric interpretation can be obtained by fixing the

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1849–1860, Osaka, Japan, December 11-17 2016.

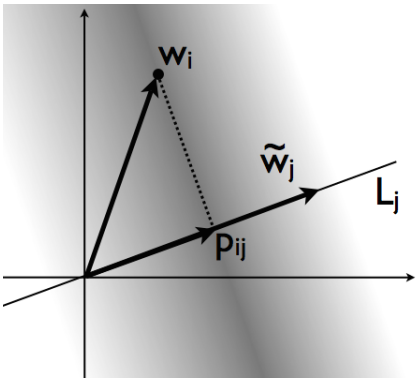


Figure 1: Density modelling the possible values of w_i, induced by a single context word $\tilde{w}_j$.

context vectors. The density associated with a given context word can then be viewed as a soft constraint on the possible values of the vector w_i, as illustrated in Figure 1. The variance of this density depends on the size of the corpus (larger corpora lead to more precise estimates) and on the informativeness of the context word, where densities associated with uninformative context words should have a high variance, reflecting the fact that they should not have a strong impact on the word embedding.

The remainder of this paper is structured as follows. In the next section, we give an overview of related work. Subsequently, in Section 3 we introduce our probabilistic model for word embedding and discuss its relationship with the GloVe model. Finally, we present our experimental results and conclusions.

2 Related work

Word embedding models construct vector space models of word meaning by relying on the distributional hypothesis, which states that similar words tend to appear in similar linguistic contexts (Harris, 1954). One class of methods relies on constructing a term-document (Landauer and Dumais, 1997) or, more commonly, a term-term (Lund and Burgess, 1996) co-occurrence matrix. Intuitively, we can think of the row vectors in these matrices as representing the contexts in which a given word occurs. Given the sparse and high-dimensional nature of these vectors, most approaches use some form of dimensionality reduction based on matrix factorization, such as singular value decomposition (SVD). An important factor in the performance of such methods is how co-occurrences are weighted, with Positive Pointwise Mutual Information (PPMI) generally considered to be a suitable choice (Bullinaria and Levy, 2007).

In the last few years, a number of neural network inspired methods have been proposed that formulate the problem of learning word embeddings as an optimization problem. The well-known skip-gram (SG) method (Mikolov et al., 2013), for example, aims to construct vectors, such that the log-probability that word c appears in the context of word w is proportional to $w \cdot c$. The related Continuous Bag of Words (CBOW) model uses a similar idea, but instead focuses on predicting the probability of a given target word, given its context. The GloVe model (Pennington et al., 2014), which our approach is based on, learns two word vectors w_i and $\tilde{w}_j$ and a bias b_i for each word i, using the following least squares regression formulation:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} f(x_{ij})(w_i \cdot \tilde{w}_j + b_i + \tilde{b}_j - \log x_{ij})^2$$

where x_{ij} is the number of times words w_i and $\tilde{w}_j$ co-occur, $\tilde{b}_j$ is the bias for the context word j, and n is the number of different words in the considered corpus. The function f weights the terms of the model to limit the effect of small co-occurrence counts, as these are deemed to be noisy. It is defined as $f(x_{ij}) = \left(\frac{x_{ij}}{x_{max}}\right)^{\alpha}$ if $x_{ij} < x_{\max}$ and $f(x_{ij}) = 1$ otherwise. The purpose of x_{max} is to prevent common words from dominating the objective function too much.

An interesting property of the representations learned by SG, CBOW and GloVe is that they capture similarity as well as analogies and related linear relationships. As a result, both word similarity and word analogy tasks are now commonly used to evaluate the quality of word embeddings. Finally, note

that while methods such as SG might seem like a radical departure from matrix factorization based methods, it was shown in (Levy and Goldberg, 2014) that SG implicitly finds a factorization of a shifted-PMI weighted term-term co-occurrence matrix. It has been observed that, compared to the factorization model underlying SG, SVD remains a useful choice for modeling word similarity, but it is less suited for discovering analogies (Levy and Goldberg, 2014).

The standard word embedding models have recently been improved in various ways. For example, some authors have proposed so-called multi-prototype representations, where the idea is to deal with ambiguity by learning several vectors for each word (Reisinger and Mooney, 2010; Huang et al., 2012; Liu et al., 2015; Neelakantan et al., 2015), intuitively by clustering the contexts in which the word appears, as a proxy for word senses, and learning one vector for each context. Other authors have shown how word embeddings can be improved by taking into account existing structured knowledge, e.g. from lexical resources such as WordNet or from knowledge graphs such as Freebase (Yu and Dredze, 2014; Xu et al., 2014; Faruqui et al., 2015).

While most word embedding models represent words as vectors, a few authors have explored the usefulness of region and density based representations. For example, in (Erk, 2009), two models are proposed to induce regions from context vectors. Among others, it is shown that these regions can be used to encode hyponym relations. In (Vilnis and McCallum, 2015), a model is proposed which represents words as Gaussian densities, and the usefulness of this model for discovering word entailment is demonstrated. As already mentioned in the introduction, while our model also represents words as densities, our densities model the uncertainty about the true location of a word vector, rather than modelling the diversity of the underlying concept. As a result, for instance, Kullback-Leibler divergence is meaningful for modelling word similarity in the approach from (Vilnis and McCallum, 2015), but would not be appropriate in our model. To the best of our knowledge, the model presented in this paper is the first that explicitly models the uncertainty associated with the word vectors. Note that while several probabilistic models have been proposed for word embedding (Maas and Ng, 2010; Li et al., 2015), these works model the probability that a document has been generated, rather than the probability that a given vector is the correct representation of a given word.

3 Our model

Similar to the GloVe model, we propose to learn word embeddings by solving the following weighted least squares problem:

$$\sum_{i=1}^{n} \sum_{j \in J_i} \frac{1}{\sigma_{ij}^2} (w_i \cdot \tilde{w}_j + \tilde{b}_j - s_{ij})^2 \tag{1}$$

where n is the number of words in the vocabulary, $J_i \subseteq \{1, ..., n\}$, and $\tilde{b}_j$ and s_{ij} are constants. In particular, the GloVe model can be recovered by choosing $J_i = \{j \,|\, x_{ij} > 0\}$, $\sigma_{ij}^2 = \frac{1}{f(x_{ij})}$ and $s_{ij} = \log x_{ij} - b_i$. However, as we explain below, these choices are sub-optimal. First, in Section 3.1 we propose to use a Dirichlet-Multinomial language modeling approach and choose s_{ij} as the expectation of $\log P(j|i)$ in this model. Section 3.2 then explains how suitable estimates for σ_{ij}^2 can be obtained. The importance of these estimates is twofold: they should improve the quality of the word vectors that we obtain by solving (1) and they will enable us to precisely model our uncertainty about the exact location of each word vector. This latter point is discussed in more detail in Section 3.3, which explains how we can evaluate the likelihood that a vector is the correct representation of a given word.

3.1 Dealing with imperfect corpus statistics

Let us write $s_{ij}^{glove} = \log x_{ij} - b_i$. The idea behind the derivation of the GloVe model in (Pennington et al., 2014) is that s_{ij}^{glove} is an estimation of $\log P(j|i)$, with $P(j|i)$ the probability of seeing word j in the context of word i. Rather than fixing $b_i = \log \sum_l x_{il}$, in line with this view, it is assumed that $\log \sum_l x_{il}$ is absorbed in the bias term b_i. One of the main advantages of this choice is that it makes the model

symmetric w.r.t. the role of the target vectors w_i and context vectors $\tilde{w}_j$; e.g. in some experiments it was observed that using the average of w_i and $\tilde{w}_j$ can lead to a small increase in performance.

In our model, s_{ij} will be chosen as an estimation of $\log P(j|i)$. This will enable a more elegant modeling of the residual errors, and offer a more principled way of dealing with sparse frequency counts. It also leads to a clearer geometric interpretation. In particular, let us write p_{ij} for the orthogonal projection of w_i on the line $L_j = \{p \,|\, p = \lambda \cdot \tilde{w}_j, \lambda \in \mathbb{R}\}$ (see Figure 1), then $w_i \cdot \tilde{w}_j = \|\tilde{w}_j\| \cdot \|p_{ij}\|$. This allows us to write the residual error as $e_{ij} = \|\tilde{w}_j\| \cdot \|p_{ij}\| - s_{ij} + \tilde{b}_j$. We can think of $\|p_{ij}\|$ as the coordinate of word i in a one-dimensional word embedding, which is constrained by the model to correspond to a linear function of s_{ij}. The relation between this one-dimensional embedding and the full embedding is determined by $\|\tilde{w}_j\|$ and $\tilde{b}_j$, which only depend on the context word j. Another way to look at this geometric interpretation is that each context word j acts as a soft constraint on the possible choices of w_i, which is illustrated by the shaded area in Figure 1.

Clearly $\frac{x_{ij}}{\sum_l x_{il}}$ only gives us a reliable estimate of $P(j|i)$ if the number of occurrences of i is sufficiently large. This problem is well known and can be alleviated by various smoothing techniques (Zhai and Lafferty, 2004). In this paper, we will adopt Bayesian smoothing. In addition to smoothing the frequency counts, this will give us a way to estimate variance. In particular, we assume that for each target word i there is a multinomial distribution from which all words that appear in the context of i are drawn. A standard approach is to assume that the parameters of that multinomial distribution are drawn from a Dirichlet distribution. Specifically, let x_{ij} be the number of times word j appears in the context of word i in the considered corpus, as before, and let $x_i = \sum_l x_{il}$. Note that x_i is the total number of tokens that occur in the context of i. The probability that among these there are y_1 occurrences of word 1, y_2 occurrences of word 2, etc. is given by

$$P(\mathbf{y} \,|\, \alpha) = \frac{x_i B(\sum_j \alpha_j, x_i)}{\prod_{y_j > 0} y_j B(\alpha_j, y_j)}$$

where $\mathbf{y} = (y_1, ..., y_n)$, $\alpha = (\alpha_1 + x_{i1}, ..., \alpha_n + x_{in})$ and B is the Beta function. In the experiments, we will use the overall corpus statistics to set the parameters of the Dirichlet prior, i.e. we will choose $\alpha_i = \lambda \cdot \frac{n_i}{\sum_j n_j}$, where n_i is the total number of occurrences of word i in the corpus and $\lambda > 0$ is a parameter that will be chosen based on tuning data.

Using this Dirichlet-Multinomial model, we can set s_{ij} as the expectation of $\log \frac{Y_{ij}}{x_i}$, where the random variable Y_{ij} represents the number of occurrences of word j in the context of word i. We estimate this expectation using a Taylor expansion:

$$s_{ij} = E[\log Y_{ij}] - \log x_i \approx \log E[Y_{ij}] - \frac{\mathit{Var}[Y_{ij}]}{(2 \cdot E[Y_{ij}]^2)} - \log x_i$$

where

$$E[Y_{ij}] = \frac{x_i \alpha_j^*}{\sum_l \alpha_l^*} \qquad \mathit{Var}[Y_{ij}] = \Big(\frac{x_i \alpha_j^*}{\sum_l \alpha_l^*}\Big)\Big(1 - \frac{\alpha_j^*}{\sum_l \alpha_l^*}\Big)\Big(\frac{x_i + \sum_l \alpha_l^*}{1 + \sum_l \alpha_l^*}\Big)$$

with $\alpha_j^* = \alpha_j + x_{ij}$. This choice of s_{ij} has two advantages over s_{ij}^{glove}. First, by smoothing the raw frequency counts, we obtain more reliable estimates for rare terms. Second, context words j for which $x_{ij} = 0$ are completely ignored in the GloVe model. From the point of view of the proposed geometric interpretation, this means that valuable information is ignored. In particular, if we want $\|p_{ij}\|$ to reflect how strongly context word j is related to word i, we should require that it is small when $x_{ij} = 0$. On the other hand, evaluating (1) for every pair (i, j) is not feasible as it would make the complexity of the model quadratic. Therefore, we let J_i contain all indices j for which $x_{ij} > 0$, as well as a random sample of indices for which $x_{ij} = 0$. In our experiments, we choose the sample size such that the number of indices for which $x_{ij} > 0$ is equal to the number of indices for which $x_{ij} = 0$.

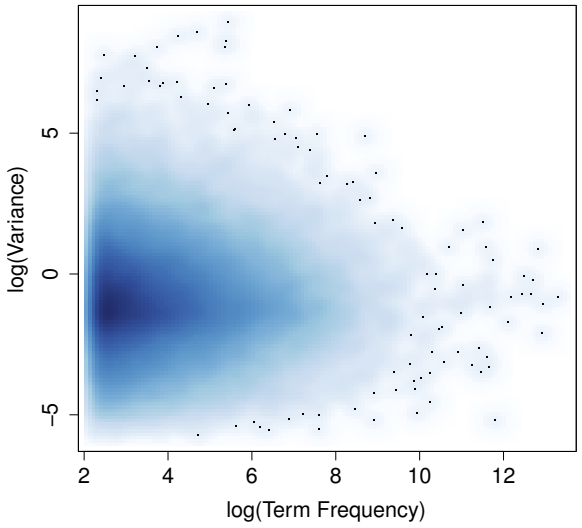


Figure 2: Scatter plot comparing term frequency with $\mathit{Var}[e_j^{\mathit{info}}]$.

3.2 Estimating the variance of residual errors

The choice of $f(x_{ij})$ as the weight for the term corresponding to target word i and context word j reflects the implicit assumption that $\frac{1}{f(x_{ij})}$ is a reasonable estimate of the variance σ_{ij}^2 of the residual error $e_{ij}^{\mathit{glove}} = w_i \cdot \tilde{w}_j + b_i + \tilde{b}_j - \log x_{ij}$. As we will see, this assumption is rather questionable.

A standard technique for selecting the weights in weighted least squares problems, called feasible generalized least squares, consists in estimating the variance of the residual errors in an initial solution (e.g. obtained using standard least squares). This allows us to reformulate the objective function by deriving appropriate weights from the estimated variances σ_{ij}^2. Solving the resulting optimization problem in turn allows us to obtain better estimates of the variances. This process is repeated for a fixed number of times, or until the estimated variances converge.

In our model, we will follow this strategy to estimate the variances σ_{ij}^2 from the observed residual errors e_{ij}. This requires us to make assumptions about which factors affect these variances, as we can clearly not estimate σ_{ij}^2 from e_{ij} alone. We will assume that e_{ij} is the sum of two independent errors, viz. $e_{ij} = e_{ij}^{\mathit{count}} + e_j^{\mathit{info}}$. Intuitively e_{ij}^{count} is the error that results from using unreliable co-occurrence statistics and e_j^{info} is the error that results when the target word j is uninformative. Again using a Taylor expansion, we can estimate $\mathit{Var}[e_{ij}^{\mathit{count}}]$ as follows:

$$\mathit{Var}[e_{ij}^{\mathit{count}}] = \mathit{Var}[\log Y_{ij}] \approx \frac{\mathit{Var}[Y_{ij}]}{E[Y_{ij}]^2}$$

where $E[Y_{ij}]$ and $\mathit{Var}[Y_{ij}]$ are evaluated as before. Furthermore, we can estimate $\mathit{Var}[e_j^{\mathit{info}}]$ from the observed residual errors, as follows:

$$\frac{\sum\{e_{ij}^2 : j \in J_i\} - \sum\{\mathit{Var}[e_{ij}^{\mathit{count}}] : j \in J_i\}}{|\{e_{ij}^2 : j \in J_i\}|} \tag{2}$$

This allows us to estimate σ_{ij}^2 as $\mathit{Var}[e_{ij}^{\mathit{count}}] + \mathit{Var}[e_j^{\mathit{info}}]$.

Figure 2 shows the relationship between $\mathit{Var}[e_j^{\mathit{info}}]$ and the number of occurrences of the context word j in the text collection (for a subset of Wikipedia[1]). As can be seen from the figure, the correlation

[1] http://mattmahoney.net/dc/text8.zip

Table 1: Examples illustrating the weak correlation between term frequency and informativeness, measured in terms of the variance $Var[e_j^{info}]$.

Frequent and informative

	Term Frequency	$Var[e_j^{info}]$
one	411764	1.39
time	21412	0.720
states	14916	0.259
united	14494	0.282
city	12275	0.221
university	10195	0.632
french	8736	0.270
two	192644	0.815
american	20477	1.26
government	11323	1.54

Frequent and uninformative

	Term Frequency	$Var[e_j^{info}]$
in	372201	58.08
was	112807	43.16
or	68945	57.10
his	62603	47.05
also	44358	44.87
their	31523	84.35
used	22737	31.80
these	19864	25.96
e	11426	45.75
without	5661	30.38

Infrequent and uninformative

	Term Frequency	$Var[e_j^{info}]$
wendell	40	29.38
actuality	42	29.75
ebne	54	30.17
christology	45	31.04
mico	45	33.38
utilised	30	21.52
reopened	54	21.32
generalizes	24	19.07
flashing	49	19.83
eitc	27	20.77

Infrequent and informative

	Term Frequency	$Var[e_j^{info}]$
psycho	56	0.05
quantization	56	0.25
residue	56	0.02
inert	54	0.98
imap	54	0.19
batsman	52	0.68
bilinear	52	0.18
crucified	50	0.08
germanium	50	0.11
lactose	50	0.45

between these two quantities is very weak, e.g. high-frequency words can be very informative. For example, the words 'family' and 'service' are frequent in Wikipedia but were still found to be highly informative context words (i.e. $Var[e_j^{info}]$ is low for these words), while stop words such as 'were' and 'is' are found to be uninformative (i.e. $Var[e_j^{info}]$ is high for these words). Similarly, there are low-frequency words which are found to be uninformative, such as 'ga', 'scoula' and 'niggle' while other low-frequency words were found to be highly informative, such as 'compactness' and 'nasdaq'. Table 1 shows a number of additional examples of words with high/low frequency and high/low variance.

3.3 Evaluating likelihood

Explicitly modelling the residual error allows us to associate a density with each word. For example, the density shown in Figure 1 intuitively captures the evidence about the embedding of the word i that is provided by the context word j. In this section, we will assume that each target word is associated with a random vector. Note that the residual error e_{ij} then is a random variable. We will evaluate the likelihood that e_{ij} takes a given value by evaluating the likelihood that e_{ij}^{count} takes a given value s and that e_{ij}^{info} takes the value $r - s$. Let $S_{ij} \subseteq \mathbb{R}$ be the set of possible values that e_{ij}^{count} can take, i.e.:

$$S_{ij} = \{E[\log Y_{ij}] - \log P(Y_{ij} = k) : 0 \leq k \leq x_i\}$$

With each target word i and context word j we can associate the density f_{ij} defined for $r \geq 0$ as:

$$f_{ij}(r) = \frac{1}{|S_{ij}|} \sum_{s \in S_{ij}} P(Y_{ij} = k) \cdot f_{ij}^{info}(r - s) \tag{3}$$

Here $f_{ij}(r)$ is the likelihood that the residual error e_{ij} takes the value r, while $f_{ij}^{info}(s)$ is the likelihood that e_{ij}^{info} takes the value s. The variance σ_{ij}^{info} of f_{ij}^{info} is given by (2). If we furthermore assume that e_{ij}^{info} is normally distributed, we obtain:

$$f_{ij}^{info}(r - s) = \mathcal{N}(r - s, 0, \sigma_{ij}^{info})$$

If we treat each context word as an independent source of evidence, we obtain the following density g_i, modelling our knowledge about the possible choices of a word vector for word i ($w_i \in \mathbb{R}^s$):

$$g_i(w_i) = \prod_{j \in J_i} f_{ij}(w_i \cdot \tilde{w}_j - s_{ij} + \tilde{b}_j) \tag{4}$$

Note that we assume that the context vectors $\tilde{w}_j$ are given.

4 Evaluation

In this section we compare our method with existing word embedding models on a range of standard benchmark tasks.

4.1 Methodology

Corpora We have used the following text collections: Wikipedia[2] (1,335,766,618 tokens), the English Gigaword corpus[3] (1,094,733,691 tokens), a concatenation of the Wikipedia and Gigaword corpora (2,430,500,309 tokens), UMBC[4] (2,714,554,484 tokens) and ClueWeb-2012 Category-B[5] (6,030,992,452 tokens). Note that the first three text collections have also been used in (Pennington et al., 2014). We adopted a straightforward text preprocessing strategy. In particular, following (Pennington et al., 2014), we have removed punctuations, lower-cased the tokens, removed HTML/XML tags, and conducted sentence segmentation. For the ClueWeb collection, we used the preprocessing implementation of the reVerb tool[6], which was specifically designed to process ClueWeb, and only considered terms which occur at least 100 times in the collection, to offset the larger size of this collection. This led to a vocabulary size of 283,701 words. For the other collections, we used our own code, which is available along with the rest of our implementation[7], and used the NLTK library[8] for sentence segmentation. As these collections are smaller than the ClueWeb collection, we considered all words that appear at least 10 times. The resulting vocabulary sizes are 1,252,101 words for Wikipedia, 469,052 words for the Gigaword corpus, 1,524,043 words for Wikipedia+Gigaword and 541,236 words for UMBC. When counting word co-occurrence statistics, we do not cross sentence boundaries. Similar to GloVe, words in the context windows in our model were weighted using the harmonic function. For the baseline models, we used the context word weighting scheme from their original implementations.

Baseline methods and variants We consider the following state-of-the-art word embedding baselines: the Skip-Gram (SG) and Continuous-Bag-of-Words (CBOW) models from (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and the Gaussian word embedding model (Gauss) from (Vilnis and McCallum, 2015). In all cases, we have used existing implementations of these models[9][10][11]. Furthermore, we have considered several variants of our model to better understand what components are responsible for the improvements over GloVe. In the standard version, we estimate the similarity between words w_i and w_j by evaluating the likelihood $g_i(w_j)$, as defined in (4). In variant DG-ZC we instead use cosine similarity, as in the GloVe model. In variant DG-C we also use the cosine similarity and in addition set J_i as in the GloVe model (i.e. we disregard pairs (i, j) for which $x_{ij} = 0$). Variant DG-UfL differs from the standard model by not considering the error term e_j^{info}.

Evaluation tasks We have evaluated the models on traditional word analogy and word similarity tasks (Levy et al., 2015). In particular, we have used an existing Google Word analogy dataset which we obtained from the GloVe project[12]. In addition, we have used the Microsoft Word analogy dataset[13] as well as twelve existing word similarity datasets[14]. The aim of these evaluation tasks has been explained in detail in (Levy et al., 2015). A new evaluation task for word embedding has recently been proposed

[2] We used the dump from November 2nd, 2015.
[3] https://catalog.ldc.upenn.edu/LDC2011T07
[4] http://ebiquity.umbc.edu/resource/html/id/351
[5] http://lemurproject.org/clueweb12/
[6] http://reverb.cs.washington.edu/
[7] https://github.com/bashthebuilder/pGlove
[8] http://www.nltk.org/
[9] https://code.google.com/archive/p/word2vec/
[10] http://nlp.stanford.edu/projects/glove/
[11] https://github.com/seomoz/word2gauss
[12] http://nlp.stanford.edu/projects/glove/
[13] https://bitbucket.org/omerlevy/hyperwords/src
[14] https://github.com/mfaruqui/retrofitting

Table 2: Comparison with baseline methods on standard word embedding evaluation tasks.

	Gsem	Gsyn	MSR	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Outlier	
	Acc			Spearman's ρ												Acc	OPP
	Wikipedia																
SG	71.6	**64.2**	**68.6**	0.658	0.773	0.784	**0.645**	**0.708**	**0.456**	0.500	**0.415**	**0.435**	0.773	0.655	0.731	70.3	93.8
CBOW	74.2	62.4	66.2	0.644	0.768	0.740	0.532	0.622	0.419	0.341	0.361	0.343	0.707	0.597	0.693	73.4	95.3
Gauss	61.3	53.3	43.8	0.593	0.632	0.681	0.409	0.506	0.256	0.392	0.337	0.416	0.649	0.601	0.644	04.6	40.0
GloVe	80.2	58.0	50.3	0.595	0.755	0.746	0.515	0.577	0.318	0.533	0.382	0.354	0.690	0.652	0.724	58.8	92.6
D-GloVe	**81.4**	59.1	59.6	**0.670**	**0.789**	**0.789**	0.560	0.658	0.401	**0.540**	0.413	0.391	**0.780**	**0.656**	**0.749**	**73.5**	**96.1**
	Gigaword																
SG	61.5	**63.2**	**67.5**	0.676	0.628	0.594	**0.550**	**0.614**	**0.446**	0.408	**0.422**	**0.408**	**0.691**	0.621	**0.696**	74.9	84.1
CBOW	50.2	58.1	64.8	0.615	0.568	0.600	0.416	0.518	0.405	0.259	0.347	0.343	0.625	0.520	0.610	74.1	84.0
Gauss	38.2	45.1	40.1	0.600	0.474	0.548	0.413	0.507	0.326	0.223	0.307	0.204	0.504	0.473	0.567	56.0	66.2
GloVe	64.4	59.6	55.8	0.600	0.669	**0.599**	0.511	0.535	0.336	0.486	0.327	0.255	0.593	0.606	0.668	74.2	83.2
D-GloVe	**65.5**	61.5	58.9	**0.697**	**0.673**	**0.599**	0.521	0.555	0.394	**0.499**	0.387	0.289	0.663	**0.622**	**0.696**	**75.2**	**86.0**
	Gigaword+Wikipedia																
SG	74.4	**69.6**	**69.3**	0.678	0.712	0.794	**0.659**	**0.719**	**0.459**	0.511	**0.518**	**0.437**	0.731	0.631	0.733	82.8	91.6
CBOW	72.2	61.2	66.2	0.633	0.699	0.681	0.528	0.592	0.419	0.321	0.405	0.359	0.688	0.561	0.662	80.1	90.1
Gauss	56.1	53.2	51.9	0.601	0.583	0.619	0.421	0.518	0.311	0.318	0.332	0.319	0.581	0.557	0.617	40.1	55.9
GloVe	78.8	66.9	58.6	0.608	0.741	0.735	0.598	0.581	0.388	0.578	0.399	0.357	0.711	0.616	0.719	81.8	89.6
D-GloVe	**86.8**	67.2	66.3	**0.689**	**0.749**	**0.799**	0.606	0.589	**0.459**	**0.589**	0.482	0.401	**0.743**	**0.640**	**0.742**	**85.2**	**92.5**
	UMBC																
SG	62.0	65.8	**68.7**	0.619	0.778	0.753	**0.594**	**0.620**	**0.355**	0.572	0.390	0.367	**0.684**	0.664	0.735	76.2	86.2
CBOW	73.5	**67.4**	65.3	0.619	0.768	0.733	0.586	0.616	0.345	0.577	0.347	0.352	**0.684**	0.658	0.723	76.1	85.3
Gauss	56.7	64.4	55.2	0.614	0.764	0.742	0.583	0.608	0.342	0.571	0.362	0.344	0.674	0.652	0.717	45.9	59.2
GloVe	65.9	66.5	65.2	0.618	0.770	0.731	0.587	0.613	0.344	0.572	0.374	0.363	0.679	0.660	0.723	75.9	85.2
D-GloVe	**77.5**	66.7	65.3	**0.620**	**0.796**	**0.756**	0.591	0.618	**0.355**	**0.592**	**0.394**	**0.368**	**0.684**	**0.667**	**0.736**	**77.1**	**87.1**
	ClueWeb12-B																
SG	37.3	58.9	**87.5**	0.674	0.725	0.713	**0.632**	0.680	0.463	**0.384**	0.389	0.388	0.730	0.643	0.718	86.7	98.1
CBOW	50.0	**61.7**	**87.5**	0.636	0.702	0.704	0.514	0.612	0.422	0.329	0.362	0.367	0.691	0.612	0.668	86.4	97.9
Gauss	39.5	49.0	72.1	0.611	0.664	0.670	0.416	0.520	0.261	0.314	0.339	0.312	0.669	0.599	0.647	75.8	81.7
GloVe	48.9	51.7	85.2	0.651	0.724	0.720	0.621	0.681	0.421	0.321	0.356	0.361	0.700	0.619	0.678	79.8	97.1
D-GloVe	**56.7**	60.4	87.0	**0.675**	**0.744**	**0.736**	0.629	**0.683**	**0.533**	0.383	**0.390**	**0.389**	**0.731**	**0.653**	**0.724**	**86.8**	**98.2**

Table 3: Results for different variants of our model on the Wikipedia collection.

	Gsem	Gsyn	MSR	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Outlier	
	Acc			Spearman's ρ												Acc	OPP
D-GloVe	81.4	59.1	59.6	0.670	0.789	0.789	0.560	0.658	0.401	0.540	0.413	0.391	0.780	0.656	0.749	73.5	96.1
DG-ZC	80.9	58.8	51.8	0.659	0.781	0.786	0.521	0.589	0.320	0.533	0.383	0.370	0.779	0.661	0.747	66.1	95.0
DG-C	80.8	58.3	51.5	0.659	0.781	0.784	0.518	0.581	0.321	0.533	0.382	0.361	0.778	0.661	0.740	62.8	94.9
DG-UfL	79.9	56.2	50.1	0.615	0.763	0.758	0.491	0.568	0.311	0.509	0.376	0.349	0.709	0.645	0.704	61.8	93.7

in (Camacho-Collados and Navigli, 2016), which we have also considered, using the evaluation script provided by the authors[15]. The aim of this task is to find the outlier in a given set of words. We refer to (Camacho-Collados and Navigli, 2016) for a detailed explanation of the task and the considered evaluation metrics.

Parameter tuning We select the parameters for each of the methods using a 25% validation set and report results on the remaining 75% of each evaluation set. The parameters were tuned separately for each of the evaluation tasks. For CBOW and SG, we chose the number of negative samples from a pool of {1, 5, 10, 15}. For GloVe, we selected the x_{max} value from {10, 50, 100} and α from {0.1, 0.25, 0.5, 0.75, 1}. For the Gaussian word embedding approach, we used the spherical Gaussian with KL-divergence model. For our model, we selected the Dirichlet prior constant λ from {0.0001, 0.001, 0.01, 0.1, 1000, 2000, 5000, 8000}. For all models, the number of dimensions was chosen from {100, 300}, the size of the context windows was chosen from {2, 5, 10}, and the number of iterations was fixed as 50. In our model, we re-estimate the variances σ_{ij}^2 every five iterations.

[15] http://lcl.uniroma1.it/outlier-detection/

Table 4: Results for high-frequency and low-frequency words for Wikipedia (left) and UMBC (right).

Most frequent	S4	S5	S6	S8	S9
SG	0.504	0.452	0.598	0.711	0.596
D-GloVe	**0.530**	**0.560**	**0.650**	**0.773**	**0.724**
Least frequent	S4	S5	S6	S8	S9
SG	**0.623**	**0.445**	**0.400**	**0.560**	**0.559**
D-GloVe	0.579	0.328	0.200	0.182	0.245

Most frequent	S4	S5	S6	S10
SG	0.511	0.256	0.591	0.287
D-GloVe	**0.575**	**0.354**	**0.626**	**0.333**
Least frequent	S4	S5	S6	S10
SG	**0.661**	**0.372**	**0.100**	**0.331**
D-GloVe	0.605	0.312	0.091	0.313

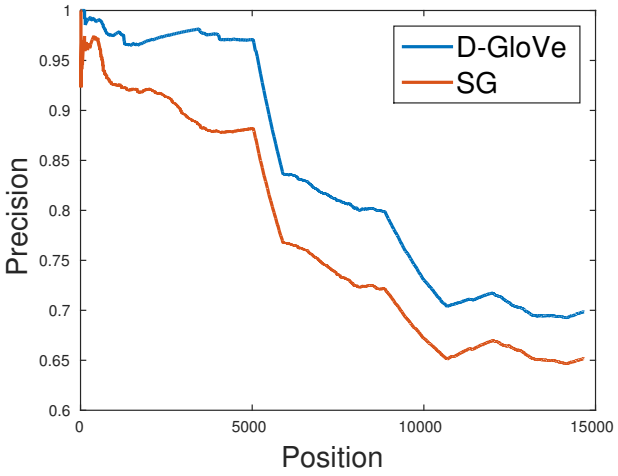


Figure 3: Confidence ranking plot for the Google word analogy test set.

4.2 Results

We present our main results in Table 2. The word similarity datasets are indexed as: S1: EN-MTurk-287, S2: EN-RG-65, S3: EN-MC-30, S4: EN-WS-353-REL, S5: EN-WS-353-ALL, S6: EN-RW-STANFORD, S7-EN-YP-130, S8-EN-SIMLEX-999, S9-EN-VERB-143, S10-EN-WS-353-SIM, S11: EN-MTurk-771, and S12: EN-MEN-TR-3k. We can see from the results that our model consistently outperforms GloVe. To further understand what components are responsible for this improvement, Table 3 shows the result for some variants of our model, showing that each of the proposed adaptations of the GloVe model contributes to the overall result. Compared to the other baselines, the results in Table 2 show that our model performs substantially better for the semantic instances of the Google analogy datasets (Gsem), while it is outperformed by SG (and in some cases CBOW) for the syntactic instances (Gsyn) and for the Microsoft dataset (MSR), which contains only syntactic instances. Our model also outperforms the baselines for the outlier detection task. For the similarity test instances, the performance is mixed. What is noticeable is that our model performs comparatively better for large corpora (e.g. ClueWeb) and worse for smaller corpora (e.g. Gigaword).

In Table 4 we present a more detailed analysis of the similarity test sets for which our model performs worse than SG. In particular, the table shows the results of a modified test set that only considers the 30% most frequent terms and a modified test set that only considers the 30% least frequent terms. These results clearly show that our model outperforms SG for high-frequency terms and that it is outperformed by SG for low-frequency terms. Dirichlet-Multinomial model are indeed known to struggle with low-frequency terms (Sridhar, 2015), which can e.g. be addressed by the use of asymmetric Dirichlet priors (Wallach et al., 2009). Note that this observation also explains why our model performs comparatively better for larger corpora and why it performs worse for syntactic analogy instances (given that such instances tend to contain low-frequency terms). While this can be seen as a limitation of our model, the fact that our model treats low-frequency terms in a cautious way may actually be advantageous in downstream applications.

An advantage of our approach is that the likelihood based formulation naturally allows us to estimate the confidence that a given prediction is correct. In Figure 4.2 we show the accuracy for the k analogy

instances of the Google dataset about which our model was most confident, for varying values of k. Similarly, the figure also shows the accuracy of the predictions made by SG, ranked in terms of cosine similarity. As can be seen, our model is better able to identify those instances that it can answer correctly (e.g. the accuracy of the top 5000 instances remains close to 1).

5 Conclusions

We have proposed a new word embedding model in which each word is represented as a density, obtained by associating with each word i and each context word j a univariate density. These univariate densities are in turn obtained by explicitly modelling the residual error of the considered least squares optimization function. Our experiments reveal that the model consistently outperforms the GloVe model, on which it is based. The proposed model also outperforms skip-gram, and other baselines, for high-frequency terms. For low-frequency terms, our model takes a rather cautious approach, which means that it is often outperformed by skip-gram in standard evaluation settings.

6 Acknowledgments

This work was supported by ERC Starting Grant 637277. This work was performed using the computational facilities of the Advanced Research Computing@Cardiff (ARCCA) Division, Cardiff University.

References

John A Bullinaria and Joseph P Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526.

Jose Camacho-Collados and Roberto Navigli. 2016. Find the word that does not belong: A framework for an intrinsic evaluation of word vector representations. In *Proceedings of the ACL Workshop on Evaluating Vector Space Representations for NLP*.

Katrin Erk. 2009. Representing words as regions in vector space. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 57–65.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615.

Zellig Harris. 1954. Distributional structure. *Word*, 10:146–162.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 873–882.

Thomas K Landauer and Susan T Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104:211–240.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Shaohua Li, Jun Zhu, and Chunyan Miao. 2015. A generative word embedding model and its low rank positive semidefinite solution. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1599–1609.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2418–2424.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28:203–208.

Andrew L Maas and Andrew Y Ng. 2010. A probabilistic model for semantic word vectors. In *Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pages 3111–3119.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv:1504.06654*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117.

Vivek Kumar Rangarajan Sridhar. 2015. Unsupervised topic modeling for short texts using distributed representations of words. In *Proceedings of the 2015 Conference of theNorth American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 192–200.

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

Luke Vilnis and Andrew McCallum. 2015. Word representations via Gaussian embedding. In *Proceedings of the International Conference on Learning Representations*.

Hanna M Wallach, David M Mimno, and Andrew McCallum. 2009. Rethinking LDA: Why priors matter. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*, pages 1973–1981.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. RC-NET: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, pages 1219–1228.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 545–550.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22:179–214.

Predicting human similarity judgments with distributional models: The value of word associations

Simon De Deyne and Amy Perfors
Computational Cognitive Science Lab
School of Psychology
University of Adelaide
simon.dedeyne@adelaide.edu.au
amy.perfors@adelaide.edu.au

Daniel J Navarro
School of Psychology
University of New South Wales
dan.navarro@unsw.edu.au

Abstract

Most distributional lexico-semantic models derive their representations based on *external* language resources such as text corpora. In this study, we propose that *internal* language models, that are more closely aligned to the mental representations of words could provide important insights into cognitive science, including linguistics. Doing so allows us to reflect upon theoretical questions regarding the structure of the mental lexicon, and also puts into perspective a number of assumptions underlying recently proposed distributional text-based models. In particular, we focus on word-embedding models which have been proposed to learn aspects of word meaning in a manner similar to humans. These are contrasted with internal language models derived from a new extensive data set of word associations. Using relatedness and similarity judgments we evaluate these models and find that the word-association-based internal language models consistently outperform current state-of-the art text-based external language models, often with a large margin. These results are not just a performance improvement; they also have implications for our understanding of how distributional knowledge is used by people.

1 Introduction

How is semantic information encoded? How is similarity represented in the brain? And how can we capture this information computationally? One answer to this question involves distributional lexico-semantic models, which quantify the semantic similarity between lexical items based on their distributional properties in large samples of data. Recent models like `word2vec` (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), which rely on external corpora as the source of data, increasingly appear to capture word meaning in ways that ever-more-closely resemble human representations. For instance, these models show systematic improvements over previous work in key benchmarks such as human similarity judgments of word pairs (Baroni et al., 2014). The strong performance of these models has also suggested to cognitive scientists that the learning mechanisms they embody might resemble how humans learn the meaning of some words (Mandera et al., in press).

In this study we show that using word-association data instead of corpus data improves performance substantially above the current state-of-the-art. We suggest that this is because data-intensive distributional models like `word2vec`, formidable though they are, may not capture word representations the way the average adult language speaker does. Their enormous, high-quality input data enables them to mimic human behavior, but they do relatively poorly compared to performance based on data that more accurately captures people's true representations of meaning.

The distinction between using text corpora or word association data maps onto the distinction made by Taylor (2012) between External language models (E-language) and Internal language models (I-language). An E-language model, like `word2vec`, treats language as an "external" object consisting of the all utterances made in a speech-community. An I-language model sees language as the body of knowledge residing in the brains of its speakers. Largely due to the easy availability of high-quality external corpora – for instance, there are over one trillion words in the Google n-gram corpus (Michel

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1861–1870, Osaka, Japan, December 11-17 2016.

et al., 2011) – computational linguists have traditionally focused on E-language models (Bullinaria and Levy, 2007; Baroni et al., 2014; Levy et al., 2015). Whether a similar distributional approach based on I-language might also be useful has received relatively less attention. One explanation could be purely on the basis of practical arguments, as it's not clear whether appropriate I-language resources are available. This paper fills that gap, by introducing an approximation of I-language using a new database of word associations considerably larger than previous ones and conducting a direct comparison of how both kinds of approaches predict human similarity judgments. It is valuable not just in demonstrating that models based on I-language greatly improve their performance. It also suggests that when people judge similarity, they may be relying more on networks of semantic associations than on statistics calculated from the distributional patterns of the words they hear.

Why should we expect to see (and why *do* we see) such improvements when the models use word associations data rather than high-quality large-scale text corpora data? After all, word association models generally incorporate far less data. Moreover, one might presume that word associations are themselves simply derived from the distribution of words in the external language: in that case, one would expect them to be an inferior and noisy measure.

However, several strands of research support the idea that word associations capture representations that cannot be fully reduced to the distributional properties of the E-language environment. Previous attempts to predict word associations from E-language have had limited success (Griffiths et al., 2007; Michelbacher et al., 2007; Wettler et al., 2005). E-language typically only predicts the strongest associate in the minority of cases and does even worse in predicting non-primary responses. Why is this? At least part of it is that E-language has the structure it does because people are using it to communicate to each other; it is not simply a reflection of their mental representations. For instance, the word "yellow" is a very strong associate of "banana", but the two words co-occur relatively infrequently since most bananas are yellow. As a result, modifying the word *banana* with *yellow* is uninformative, so most people leave it out when talking. Many of the divergences between the distributions of words in external language and the strength of internal associations may occur because so much of E-language is shaped by pragmatic and communicative considerations such as these. There is also evidence that meaning representations in the brain, as reflected in word associations, are shaped by far more than the distributional properties of the E-language. For instance, fMRI measures reveal that imagery-related areas like the precuneus are activated during word association tasks (Simmons et al., 2008).

The structure of this paper is as follows. In Part 2 we describe the origin and nature of the data we are using as the E-language source (text corpora) and I-language source (word association data). Part 3 describes the distributional models which we will apply to each data source, while Part 4 describes the multiple human similarity and relatedness judgments that each model and data source will be used to predict. Part 5, the results, demonstrates that models based on I-language consistently perform substantially better than the same model based on E-language.

2 Data sources

The central comparison in this paper is between model performance on E-language vs I-language data. The increasing number of online resources from which text can be extracted means that obtaining a representative E-language has become more straightforward. Furthermore, better balanced corpora that easily surpass the knowledge of the average human are readily available. We derived our E-language data based on four different kinds of existing text-based corpora, as described below.

Free word association data are used as the I-language data. Although there are certainly other possibilities, word association data are advantageous as they appear to tap directly into mental representations (Deese, 1965; McRae et al., 2012; Szalay and Deese, 1978). Moreover, we shall show that a new procedure and larger data sets address important shortcomings from previous work that relied on a relatively small number of cues words for which only a single association response was asked from the participant (Kiss et al., 1973; Nelson et al., 2004).

2.1 A text corpus to train E-language models

Our aim was to combine corpora that would provide us with a fairly balanced set of texts that is representative of the sort of language a person experiences during a lifetime – including both formal and informal language as well as spoken and written language. Four different corpora were combined.

1. Subtitles for English movies between 1970 and 2016 extracted from the OpenSubtitle corpus as described in Tiedemann (2012). Subtitle corpora have been frequently used in cognitive science because they capture daily language better than extremely large written corpora like the Google n-gram corpus (Brysbaert et al., 2011).

2. The Corpus of Contemporary English (COCA), as described in Davies (1990 present). It consists of a balanced set of formal and informal language including fiction, newspaper articles and spoken texts. We excluded the sub-corpus for academic texts.

3. The Global Web-Based English corpus (GlowBE), as described in Davies (2013). We included the sub-corpora of British, American, Canadian and Australian texts.

4. SimpleWiki, which presents knowledge that is likely available to the average person (18 million tokens in comparison to the 2.9 billion words in the full English Wikipedia).

Altogether, in compiling these corpora we aimed to be generous in terms of the quality and quantity of items so that models incorporating it would perform similarly to the existing state-of-the-art. For similar reasons, we used word-forms rather than lemmas: this matches previous work and provides the best possible match with the stimuli in the human benchmarks. The resulting corpus consisted of 2.16 billion tokens and 4.17 million types. Each sentence was uncased and stop words were removed. We further excluded words that did not occur at least 300 times, retaining 65,632 unique word types. This cut-off is larger than previous approaches using count models and word embedding models but allowed us to reduce the memory requirements for the count model we introduce later and to make sure that words in the evaluation sets were at least as frequent as the words in the association study for which we collected 300 responses. Moreover, we piloted different cut-offs and found it didn't affect our findings.

2.2 A novel word association dataset for I-language models

One of the shortcomings with previous word association studies of considerable size like the Edinburgh Association Thesaurus (Kiss et al., 1973) or the University of South Florida norms (Nelson et al., 2004) is that they only include the strongest associations (Aitchison, 2012) because only a single response is generated for each cue word. For example, in the case of *umbrella*, most participants would respond *rain*, which prevents the inclusion of weaker links. A better way to include weaker associates as well is by using a continued procedure where multiple responses for each cue word were collected (Szalay and Deese, 1978). Extending the response set to include weaker responses and including enough cue words to capture most words used in daily languages motivated us to set up a new large-scale study. The current data are collected as part of the *Small World of Words* project, an ongoing effort to map the mental lexicon in various languages [1]. Each participant was given a short list of cue words (between 15 and 20 words) and asked to generate three different responses to each cue. To avoid chaining responses, the instructions stressed to only give a response to the cue word. If a word was unknown or no secondary or tertiary response could be given, the participants were able to indicate this. Additional details on the procedure are available in (De Deyne et al., 2013).

The results reported here are based on 10,021 cue words for which at least 300 responses have been collected (100 primary, 100 secondary and 100 tertiary) for every cue. The study was presented as an online crowd sourced project in which fluent English speakers volunteered to participate. The responses were based on over 85,496 participants of which 82% were native speakers. Responses indicated as

[1] The word association task and details of the project can be accessed at `https://smallworldofwords.org/`. A paper describing an extension of these norms including over 12,000 cues is currently in preparation and the data will be made available on the same website.

unknown (1.17% of cue words) or missing, because a participant could not think of any secondary or tertiary associations (4.15% of responses), were excluded. In line with previous work, we constructed a semantic graph from these data. This graph closely resembles the bag-of-words count models but represented as a graph makes it possible to consider the spreading activation discussed in the next section. A graph $\mathbf{G}$ was constructed by only including responses that also occurred as a cue word. This converted the bimodal cue $\times$ response graph to a unimodal cue $\times$ response graph. In this weighted graph $\mathbf{G}$, g_{ij} counts the number of times that word j is given as an associate of word i. We extracted the largest strongly connected component by only keeping those cues that were also given at least once as a response. This way all words can be reached by both in- and out-going links. The resulting graph consists of 10,014 nodes, which retains 84% of the original data consisting of all responses. The average number of tokens per word is 267 and the number of different word types each word is connected to (i.e., its out-degree of) is 92, ranging from 12 (for the word *done*) to 169 (for *control*). As expected, the graph is also very sparse: only 0.92% of words are connected (i.e., $\mathbf{G}$ has 0.92% non-zero entries). Throughout the text we will refer to this graph as $\mathbf{G_{123}}$, since it incorporates all three responses given by participants.

3 Models

We consider four different models in this paper, two E-language models estimated from the text corpora, and two I-language models that use word association data. In both cases, one model is a simple count based model and the other aims to exploit the structure of the input data.

3.1 Count based model for text corpora

Count models of text corpus data use a simple representation: they track how many times a pair of words co-occur in a document or sentence. For our analyses, we applied a sliding window at the sentence level similar to Pennington et al. (2014). Specifically, we used a symmetric dynamic window that linearly weighted words as a function of the distance between them (Pennington et al., 2014). The resulting co-occurrence frequencies were transformed using the positive point-wise mutual information (PMI^+), given the evidence that this measure performs well in count models (Bullinaria and Levy, 2007; Levy et al., 2015). In particular, we follow Levy et al. (2015) in applying a discount factor in order to prevent very rare words from biasing the results (see their Equation 3), and unless otherwise stated we used the same discount factor (0.75) that they did.

3.2 Predicting structure from text corpora using word embeddings

An alternative approach to representing text corpora is to apply a lexico-semantic model that aims to extract the latent semantic structure embedded in the text corpus by learning to predict words from context. We focused on the word embeddings derived from the neural network approach in *word2vec* (Mikolov et al., 2013; Levy et al., 2015), using a continuous bag of words (CBOW) architecture in which the model is given the surrounding context for a word (i.e., the other words in a sliding window) and is trained to predict that word.

For our analyses, we used the *gensim* implementation of *word2vec* (Řehůřek and Sojka, 2010). Based on previous work (Baroni et al., 2014; Mandera et al., in press) the following settings were used: a negative sampling value of 10, and a down-sampling rate of very frequent terms of 1e-5. Additionally, the following hyper-parameters were manipulated, using the values reported by previous work (Levy et al., 2015) as a starting point. We considered window sizes between 2 to 10, and fitted models with between 100 and 500 dimensions with steps of 100. We will focus on the best fitting hyper-parameter values, but for the purposes of robustness we will also examine the performance of models using previously published semantic vectors.

3.3 Count based model for word associations

In an E-language model, the goal is to characterize the linguistic contents of a text corpus, whereas an I-language model aims to capture the mental representation that a human speaker might employ. The difference between these two goals motivates a difference in the kinds of data that one might use

(e.g., text corpora versus word associations) but there are commonalities between the two approaches. For example, there is evidence that the relationship between (observed) word association frequency and (latent) associative strength is nonlinear (Deese, 1965), an observation that suggests the PMI$^+$ measure might be reasonably successful as a simple count model for association strength. With that in mind our first model is a simple PMI$^+$ measure using the word association frequency as the input.[2]

3.4 A spreading activation approach to semantic structure

While the PMI$^+$ model captures the semantic information in the raw word association data, it does not attempt to capture any deeper semantic structure that these data encode. Inspired by classic work in human semantic memory by Collins and Loftus (1975), we use word association data to construct a network that connects associated words, and model semantic similarity using denser distributions derived from a *random walk* defined over this network, similar to the Katz index (Katz, 1953). The intuitive idea is that when a word is presented it activates the corresponding node in the graph, and starts a random walk (or many such walks) through the graph, activating nodes that the walk passes through. If there are many short paths that connect two nodes, then it is easy for a random walk through the graph to start at one node and end at the other, and the words are deemed to be more similar as a consequence.

To implement this idea we first normalize the word association matrix such that each row sums to 1, thus converting it to a transition matrix $\mathbf{P}$. Then, in order to construct an explicit model to derive new direct paths between words, we consider the following iterative procedure (Newman, 2010). First consider a walk of a maximum length r where $\mathbf{I}$ is the identity matrix and a "damping parameter" $\alpha < 1$ governs the extent to which new paths are dominated by short paths or by longer paths. During each iteration, indirect links reflecting paths of length r are added to the graphs, producing this sequence of "augmented" graphs:

$$\begin{aligned} \mathbf{G_{rw}}^{(r=0)} &= \mathbf{I} \\ \mathbf{G_{rw}}^{(r=1)} &= \alpha\mathbf{P} + \mathbf{I} \\ \mathbf{G_{rw}}^{(r=2)} &= \alpha^2\mathbf{P}^2 + \alpha\mathbf{P} + \mathbf{I} \end{aligned} \tag{1}$$

In these expressions, longer paths receive lower weights due to operation of the α parameter. The probability of an associative chain surviving across r links is thus α^r. The smaller the value of α, the larger the contribution made by very short paths. This "decay" parameter serves an important role to limit the spread of activation and avoid the entire network to become quickly activated. In the limit, where we consider paths of arbitrarily long length (and accordingly, arbitrarily low weight) we obtain the following expression:

$$\mathbf{G_{rw}} = \textstyle\sum_{r=0}^{\infty}(\alpha\mathbf{P})^r = \mathbf{I} - \alpha\mathbf{P}^{-1} \tag{2}$$

At this point, the "random walk graph" $\mathbf{G}_{\text{rw}}$ combines paths of various lengths obtained from the random walk. However, these paths do not precisely match the associative strength measure proposed earlier, and to address this we apply the exact same procedure that we have used for the other models, namely the PMI$^+$ transformation. Applying the PMI weighing function to $\mathbf{G_{rw}}$ reduces the frequency bias introduced by this type of walk (Newman, 2010) and also keeps the graph sparse.

To see how this spreading activation mechanism can be very powerful, consider the word *tiger*. Before applying spreading activation its meaning vector consists of 92 different association responses. When we apply the spreading activation measure we uncover nearly 559 new associations which ordered by their weights included *zebra*, *cheetah*, *claws*, *cougar* and *carnivore*, all of which seem meaningfully related to *tiger* but were not among the responses when *tiger* was presented as a cue word.

4 Comparing model predictions to human judgments

To assess how well each of these four models captures human semantic knowledge, we evaluate them using several standard data sets that measure human judgments of similarity and relatedness, and addi-

[2]We did not apply a discount factor for the word association data due to the different characteristics of text corpora and word associations: with smaller data sets the problem of rare words is less pronounced in word associations.

tionally introduce a new data set based on the "remote triad task". We used a variety of different data sets in order to provide insight into *why* some models perform well on some kinds of task and not on others.

4.1 Similarity and relatedness judgments

The data sets used to evaluate the models broadly fall into one of two classes. Two of the studies asked participants to judge the similarity between words, namely the WordSim-353 similarity data set (Agirre et al., 2009)[3] and the SimLex-999 data (Hill et al., 2016). In the remaining studies people were asked to judge relatedness. These include the WordSim-353 relatedness data set (Agirre et al., 2009), the MEN data (Bruni et al., 2012), the Radinsky2011 Amazon Mechanical Turk data (Radinsky et al., 2011), the popular Rubenstein and Goodenough (RG1965) data (Rubenstein and Goodenough, 1965) and the MTURK-771 data (Halawi et al., 2012).

4.2 Remote triads task

In addition to these data sets, we include data from a relatedness judgment task based on triadic comparisons using a procedure introduced in De Deyne et al. (2016). In this task, participants are asked to select the most related pair out of a set of three English nouns. An advantage of this task is that the third word acts as a context, which makes judgments less ambiguous. Critically, the triads were constructed by choosing words largely at random from the English word association data set. The only constraints were that the words in a triad had to be roughly matched on judged concreteness and word frequency. This was done to avoid simple heuristics such as grouping abstract or common words together. The consequence of this procedure is that the triads tended to consist of words that are only weakly related to each other, such as BRANCH - ROCKET - SHEET or CLOUD - TENNIS - SURGEON, and it is for this reason it is referred to as the "remote triads task". A total set of 100 triads was constructed this way and judgments were collected for 40 native English speakers [4].

4.3 Additional details

All four models represent word meanings as a semantic vector, and we used the cosine similarity measure in all cases. Only word pairs that were present in the text corpus and the word association data were included. As shown in Table 1 (columns 2 and 3), most words were retained. For the triads task model predictions were obtained by normalizing the similarities between the three words in each triad and correlating them with the frequencies of the choice preferences.

5 Results

The best performing parameters were a window size of 3 for the corpus count model, and a window size of 7 and 400 dimensions for *word2vec*, although the findings for other window sizes and dimensions were quite similar. The word association count model is based on $\mathbf{G_{123}}$ and has no free parameters, whereas for the random walk model we used a parameter value of $\alpha = 0.75$, similar to previous studies (De Deyne et al., 2016). Table 1 shows the performance of all models, and it is clear that the I-language models substantially outperform the E-language models in almost every case. It is also clear that extracting structure helps: *word2vec* generally outperformed the corpus count model, and the random walk model outperformed the word association count model. For the E-language models the magnitude of this effect was slightly smaller than reported elsewhere (Baroni et al., 2014; Mandera et al., in press), and the count model outperformed *word2vec* on the remote triads data.

5.1 Other versions of the I-language models

Given the superiority of the I-language models over E-language models in predicting human responses, it is natural to ask why this occurs. Our word association data arise from a task that elicited multiple judgments from each person. To estimate the effect of the multiple judgment procedure, we restricted the training data to the first associate only (using $\mathbf{G_1}$ rather than $\mathbf{G_{123}}$). After doing so the average

[3] Note that the items were determined by post-hoc raters who split the original WordSim-353 data set in related and similar items. As such, these judgments might not consist of "pure" similarity judgments.

[4] The data are available at `http://simondedeyne.me/data`

Table 1: Spearman rank order correlations between human relatedness and similarity judgments, and the predictions from all four models described earlier. Word association results presented here are based on $\mathbf{G_{123}}$. Further details for $\mathbf{G_1}$ are available in the text.

			Text Corpus		Word Associations	
Data set	*n*	*n(overlap)*	Count	*word2vec*	Count	Random Walk
WordSim-353 Related	252	207	.67	.70	.77	.82
WordSim-353 Similarity	203	175	.74	.79	.84	.87
MTURK-771	771	6788	.67	.71	.81	.83
SimLex-999	998	927	.37	.43	.70	.68
Radinsky2011	287	137	.75	.78	.74	.79
RG1965	65	52	.78	.83	.93	.95
MEN	3000	2611	.75	.79	.85	.87
Remote Triads	300	300	.65	.52	.62	.74
mean			.67	.69	.78	.82

correlation for the count model fell from .78 to .67, with a much smaller decline (from .82 to .78) for the random walk model. The difference is illuminating: $\mathbf{G_1}$ is much sparser than $\mathbf{G_{123}}$, allowing indirect paths to have a larger impact, which is why the random walk model is more robust than the count model.

A different question to ask is whether our new data set encoded in $\mathbf{G_{123}}$ produces better results than previous ones. There appears to be some modest evidence for this: when using the Edinburgh Association Thesaurus (EAT) consisting of 8,400 words (Kiss et al., 1973), the count model produced an average correlation of .65 to the test data, and the random walk model correlated at .74, both of which are smaller than the values obtained (.78 and .81) using a matched $\mathbf{G_{123}}$ that contains the same words as the EAT. A similar exercise using the USF association data (Nelson et al., 2004) produced correlations of .65 and .77 for the count and random walk model compared to .78 and .82 for the matched $\mathbf{G_{123}}$.

5.2 Other versions of the E-language models

Previous papers have discussed the performance of the E-language models (Levy et al., 2015), but a few additional comments are worth making. Analogous to the effect that the training data have on the I-language models, one might wonder if the poorer performance of *word2vec* was due to the text corpus we used to extract semantic vectors. To test this, we relied on recently published semantic vectors from Mandera et al. (in press)[5], Levy et al. (2015)[6] and Pennington et al. (2014)[7] including items part of $\mathbf{G_{123}}$. For the GloVe vectors based on 6 billion tokens from Pennington et al. (2014), the best result was found for 300 dimensions; the average correlation was .64. Using the GloVe vectors for a 64 billion tokens corpus improved the correlation to .67 and using the GloVe vectors for an enormous corpus of 840 billion tokens to .70. Compared with the published results from Levy et al. (2015), the average correlation was .65. Using the best-fitting vector spaces from Mandera et al. (in press), the correlation was .69 for the published vectors derived from English subtitles using 300 dimensions. Given this, it does not seem likely that the problem was our specific choice of corpus or hyperparameters.

6 Discussion

The goal of this study was to compare two kinds of semantic models: "I-language" models that encode mental representations, and "E-language" models that encode lexical contingencies. In one respect the superior performance of the I-language models is unsurprising: the training data directly reflect human mental representations, and as such *should* be more strongly linked to human semantic judgments. On the other hand, the I-language models were trained on a *much* smaller data set than the E-language models,

[5] http://zipf.ugent.be/snaut-downloads/spaces/english/predict/

[6] https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/

[7] http://nlp.stanford.edu/projects/glove/

with an average of 260 words contributing to the distributional representation of each word. Given this, it is worth considering the broader implications of the findings.

6.1 Cognitive plausibility of E-language models

Previous work has argued that the *word2vec* model is more cognitively plausible than count models due to its similarity to models of classical conditioning (Mandera et al., in press). This is contrasted with more statistical approaches such as Latent Semantic Analysis (Landauer and Dumais, 1997) and topic models (Griffiths et al., 2007). However, it is not clear that this holds up in light of the fact that we find very little difference in performance between count models and *word2vec*, or previous work arguing that word embedding models perform an implicit matrix factorization (Levy and Goldberg, 2014).

Perhaps more importantly, there is something strange about the claim that E-language models are cognitively plausible when the data sets upon which they are trained are as large as they are. If purely text based models are intended to stand as models for how humans acquire semantic structure, then they should be trained on a corpus small enough that it plausibly represents the language exposure of the young adults who participated in the benchmark tasks. If billions of tokens are required to produce adequate predictions while still being unable to match the performance of simple I-language models, it is not clear what claims can be made about human language acquisition.

6.1.1 Relation between relatedness and similarity

A final remaining question is how we can interpret the lower results for similarity judgments in one of the tasks (SimLex-999) across all models. Does this indicate a fundamental shortcoming in the models?

In this case, the answer depends. Similarity might be important in a NLP setting, for example in constructing thesauri, but the role of similarity in human semantic cognition is mostly an empirical matter. If anything, a variety of studies support a prominent role for relatedness. This includes semantic priming effects when processing a word proceeded by a related one (Hutchison, 2003), event-related potentials in EEG that are triggered by related but not similar words (Kutas and Hillyard, 1984) and fMRI studies that map the structure of the mental lexicon in a more thematic rather than taxonomic way based on similarity (Huth et al., 2016). Add to this that similarity can only be derived for certain concept combinations, and similarity ratings tend to be less reliable than relatedness ratings (Hill et al., 2016), suggests that relatedness judgments have broader use in studies of human semantic cognition.

6.2 Future directions

One obvious way to improve E-language models is by including non-linguistic information as well. As mentioned in the introduction, access to imagery contributes to the responses people give in a word association task and this might explain the fact that I-language models perform very well on the basis of a small number of words. While recent studies have shown some promising results by enhancing language models with visual representations, there's still room for improvement. For example, Bruni et al. (2012) reported findings of a multimodal model that uses word embeddings combined with features extracted from images. An evaluation using the MEN dataset resulted in a correlation of .78 for the best performing model, which is considerably lower than current results for the E-language and especially the I-language models.[8] On the upside, if I-language models do considerably better because they have a privileged access to imagery compared to E-language model, this would also suggest that further improvements by constructing more elaborate multimodal representations are possible. More generally, because the both I-language and E-language models use the same kind of symbolic language-based representations, determining what kind of features (perceptual or other) make the I-language models so successful with very little data might also provide us with valuable pointers towards further refining existing E-language models and NLP applications build from them.

Going forward will also require us to increase the discriminatory power of existing benchmarks, by including judgments that focus on finer perceptual distinctions, or by capitalizing on how humans infer additional structure beyond what's available in E-language. In this study, we have shown that the remote

[8] Unfortunately we did not have access to the semantic vectors from Bruni et al. (2012) so we could not verify whether this is due to the fact that some items were excluded in our experiments.

triad task might be ideally suited to test how well a model can generalize beyond the input and provide a benchmark capable of differentiating competing models. Apart from more sophisticated and more realistic approximating of the E-language environment, there's also a need for better I-language models. While the new word association data addresses some issues from previous studies, future work will aim to include at least 20,000 different cues. Furthermore, improvements might be achieved by looking at native speakers only, applying differential weights to the primary, secondary and tertiary responses, or designing more elaborate spreading activating mechanisms. As such, further gains for association based I-language models would not be a surprise and might help us bridge the external and internal language world.

Acknowledgments

Special thanks to Gert Storms and Marc Brysbaert for supporting the English association data collection. Salary support for this research was provided to Simon De Deyne from ARC grant DE140101749, to Amy Perfors from ARC grant DE120102378, and to Daniel J. Navarro from ARC grant FT110100431.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Jean Aitchison. 2012. *Words in the mind: An introduction to the mental lexicon.* Wiley-Blackwell.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *The 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 238–247.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Marc Brysbaert, Emmanuel Keuleers, and Boris New. 2011. Assessing the usefulness of google books word frequencies for psycholinguistic research on word processing. *Frontiers in Psychology*, 2.

John A Bullinaria and Joseph P Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526.

Allan M. Collins and Elizabeth F. Loftus. 1975. A spreading-activation theory of semantic processing. *Psychological Review*, 82:407–428.

Mark Davies. 1990–present. The Corpus of Contemporary American English: 520 million words.

Mark Davies. 2013. Corpus of Global Web-Based English: 1.9 billion words from speakers in 20 countries.

Simon De Deyne, Daniel J Navarro, and Gert Storms. 2013. Better explanations of lexical and semantic cognition using networks derived from continued rather than single word associations. *Behavior Research Methods*, 45:480–498.

Simon De Deyne, Daniel J Navarro, Amy Perfors, and Gert Storms. 2016. Structure at every scale: A semantic network account of the similarities between unrelated concepts. *Journal of Experimental Psychology: General*, 145:1228–1254.

James Deese. 1965. *The structure of associations in language and thought*. Johns Hopkins Press.

Thomas L Griffiths, Mark Steyvers, and Joshua B Tenenbaum. 2007. Topics in semantic representation. *Psychological review*, 114(2):211.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414. ACM.

Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41:665–695.

Keith A. Hutchison. 2003. Is semantic priming due to association strength or feature overlap? *Psychonomic Bulletin and Review*, 10:785–813.

Alexander G Huth, Wendy A de Heer, Thomas L Griffiths, Frédéric E Theunissen, and Jack L Gallant. 2016. Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature*, 532(7600):453–458.

Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika*, 18:39–43.

George Kiss, Christine Armstrong, R. Milroy, and J. Piper. 1973. The computer and literacy studies. chapter An associative thesaurus of English and its computer analysis, pages 153–165. Edinburgh University Press, Edinburgh.

Marta Kutas and Steven A Hillyard. 1984. Brain potentials during reading reflect word expectancy and semantic association. *Nature*, 307:161–163.

Tom K. Landauer and Susan T. Dumais. 1997. A solution to Plato's Problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104:211–240.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Paweł Mandera, Emmanuel Keuleers, and Marc Brysbaert. in press. Explaining human performance in psycholinguistic tasks with models of semantic similarity based on prediction and counting: A review and empirical validation. *Journal of Memory and Language*, 92:57–78.

Ken McRae, Saman Khalkhali, and Mary Hare. 2012. Semantic and associative relations in adolescents and young adults: Examining a tenuous dichotomy. In *The adolescent brain: Learning, reasoning, and decision making*, pages 39–66. American Psychological Association.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, et al. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331:176–182.

Lukas Michelbacher, Stefan Evert, and Hinrich Schütze. 2007. Asymmetric association measures. *Proceedings of the Recent Advances in Natural Language Processing (RANLP 2007)*.

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

Douglas L Nelson, Cathy L McEvoy, and Thomas A Schreiber. 2004. The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, and Computers*, 36:402–407.

Mark E J Newman. 2010. *Networks: An Introduction*. Oxford University Press.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8:627–633.

William K Simmons, Stephan B Hamann, Carla N Harenski, Xiaoping P Hu, and Lawrence W. Barsalou. 2008. fMRI evidence for word association and situated simulation in conceptual processing. *Journal of Physiology - Paris*, 102:106–119.

Lorand B Szalay and James Deese. 1978. *Subjective meaning and culture: An assessment through word associations*. Lawrence Erlbaum Hillsdale, NJ.

John R Taylor. 2012. *The mental corpus: How language is represented in the mind*. Oxford University Press.

Jorg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Manfred Wettler, Reinhard Rapp, and Peter Sedlmeier. 2005. Free word associations correspond to contiguities between words in texts*. *Journal of Quantitative Linguistics*, 12(2-3):111–122.

Distributional Hypernym Generation by Jointly Learning Clusters and Projections

Josuke Yamane[†] **Tomoya Takatani**[‡] **Hitoshi Yamada**[‡]
Makoto Miwa[†] **Yutaka Sasaki**[†]

[†] Toyota Technological Institute [‡] Toyota Motor Corporation

[†] {sd16432,makoto-miwa,yutaka.sasaki}@toyota-ti.ac.jp
[‡] {tomoya_takatani,hitoshi_yamada_aa}@mail.toyota.co.jp

Abstract

We propose a novel word embedding-based *hypernym generation* model that jointly learns clusters of hyponym-hypernym relations, i.e., hypernymy, and projections from hyponym to hypernym embeddings. Most of the recent hypernym detection models focus on a *hypernymy classification* problem that determines whether a pair of words is in hypernymy or not. These models do not directly deal with a hypernym generation problem in that a model generates hypernyms for a given word. Differently from previous studies, our model jointly learns the clusters and projections with adjusting the number of clusters so that the number of clusters can be determined depending on the learned projections and vice versa. Our model also boosts the performance by incorporating inner product-based similarity measures and negative examples, i.e., sampled non-hypernyms, into our objectives in learning. We evaluated our joint learning models on the task of Japanese and English hypernym generation and showed a significant improvement over an existing pipeline model. Our model also compared favorably to existing distributed hypernym detection models on the English hypernym classification task.

1 Introduction

Hypernym-hyponym relations, a.k.a. hypernymy, are important information for several NLP tasks such as question answering and ontology construction. Some manually-constructed semantic resources like WordNet contain hypernymy; however, they have limited coverage. Plenty of studies have been conducted to automatically detect hypernymy, e.g., (Hearst, 1992; Roller et al., 2014; Fu et al., 2015).

Hypernymy detection was traditionally often tackled with unsupervised methods using Hearst-style patterns (Hearst, 1992) or distributional inclusion hypothesis (Geffet and Dagan, 2005). These methods treat hypernymy pairs individually. Recent progress in the word representation allows to represent words in a shared low-dimensional vector space, and several models using the distributional word representation or word embeddings have been proposed for hypernymy detection (Roller et al., 2014; Weeds et al., 2014; Turney and Mohammad, 2015; Levy et al., 2015). Such models employ supervised learning. Most of the models focus on a *hypernymy classification* problem, i.e., whether a given word pair is in hypernymy or not, and they ignore a more practical *hypernym generation* problem to generate hypernyms for a given word.

Few studies have examined hypernym generation using word embeddings (Fu et al., 2015; Tan et al., 2015). Fu et al. (2015) proposed a two-step, pipeline model that partitions hypernymy pairs into several clusters and learns a projection matrix between words in a pair for each cluster separately. The projection matrix projects the embedding vector of a hyponym close to that of its hypernym. The incorporation of clustering allows representing several types of hypernymy and shows a higher performance than other traditional models in constructing semantic hierarchies. The clusters, however, may not be appropriate for hypernym generation since clustering is independent of hypernym generation.

This paper presents a novel hypernym generation model that jointly learns clusters of hypernymy and the projections from hyponyms to their hypernyms in the clusters . Unlike most previous supervised

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1871–1879, Osaka, Japan, December 11-17 2016.

models using word embeddings, we target the hypernym generation problem, not hypernymy classification. This paper has newly incorporated the following points into hypernym generation. First, our model performs the clustering of the relations with learning projection matrices jointly, to obtain appropriate clusters for hypernym generation. Second, motivated by DP-means (Kulis and Jordan, 2012), our model adjusts the number of clusters during training so that the model can fit the number of clusters to the training data. Third, the similarity measure in learning projection matrices is selected to be consistent with one in training word embeddings. Finally, we use sampled non-hypernym instances in learning projection matrices so that our model can distinguish hypernymy from non-hypernymy. We evaluated our model on hypernym generation tasks in Japanese and English, as well as a well-studied hypernymy classification task in English. Our joint learning model shows a significant improvement over a pipeline learning model (Fu et al., 2015) on the hypernym generation tasks. As for the hypernymy classification, our model showed a comparable performance to the state-of-the-art hypernymy classification model (Levy et al., 2015).

2 Related Work

In the task of detecting hypernym, traditional approaches focused on Hearst-style lexical patterns that indicate hypernymy (Hearst, 1992; Snow et al., 2005; Kozareva and Hovy, 2010). For instance, from a sentence ... *works by such authors as Shakespeare* ..., an "is-a" pair between *Shakespeare* and *author* can be detected by using a pattern that a word *A* is a hypernym of another word *B* when *A* and *B* are linked by *such A as B*. These methods typically show high precision but suffer from low recall because many hypernymy pairs are not explicitly mentioned in texts as such patterns.

To overcome the problems of the lack of explicit hypernymy mentions, several other traditional unsupervised methods are proposed based on distributional inclusion hypothesis. This hypothesis states that a term can only be used in contexts where its hypernyms can be used and that a term might be used in any contexts where its hyponyms are used (Geffet and Dagan, 2005; Kotlerman et al., 2010). These methods have shown limited performance because the hypothesis is not always correct and it cannot distinguish co-hyponymy and meronymy from hypernymy.

Recently, distributed word representation or word embeddings such as skip-gram (Mikolov et al., 2013) and ivLBL (Mnih and Kavukcuoglu, 2013) has been often employed for the task of detecting hypernymy, since such representation are shared among words. Most models using word embeddings focus on a hypernymy classification task where a model needs to predict whether a given word pair is in hypernymy or not. They take distributional vectors for a pair of a word x and its candidate *hypernym* y as input, calculate features from these vectors, and predict whether the pair is in hypernymy or not using a classifier like support vector machines (SVMs). For example, the *concat* model (Baroni et al., 2012) uses the concatenation of hypernymy pair $\langle \mathbf{x}, \mathbf{y} \rangle$, while the *diff* model (Roller et al., 2014; Weeds et al., 2014; Fu et al., 2015) uses $\langle \mathbf{y} - \mathbf{x} \rangle$. Here, $\mathbf{x}$ and $\mathbf{y}$ represent word embeddings of x and y, respectively. These classification-based models, however, have not been evaluated on hypernym generation, where a model generates hypernyms for a given word.

Less work has been done on hypernym generation using word embeddings (Fu et al., 2015; Tan et al., 2015). Fu et al. (2015) proposed a model of using the projection matrices, each of which projects x to y. They proposed a two-step, pipeline algorithm to detect hypernym. The first step is clustering. They partitioned training pairs into clusters using k-means so that pairs in each cluster are close in the sense of the differences (*offsets*) between x and y, i.e., $\langle \mathbf{y} - \mathbf{x} \rangle$. The second step is projection. They trained a projection matrix Φ_c for each cluster c. Φ_c is obtained by using the least squares objective:

$$\Phi_c^* = \arg\min_{\Phi_c} \frac{1}{N_c} \sum_{(x,y) \in P_c} \|\Phi_c \mathbf{x} - \mathbf{y}\|^2, \quad (1)$$

where P_c is a set of word pairs in c-th cluster and N_c is the number of word pairs in P_c. This model performs clustering and projection separately, so the performance of clustering depends on the *offsets* and their underlying embeddings and the clustering results also may not be appropriate for learning hypernymy. The similarity scores between instances in different clusters may not be comparable since

Algorithm 1: Jointly Learning Clusters and Projections of Hypernymy

input : $(x_1, y_1), \ldots, (x_n, y_n)$: input data, λ: threshold
output: $\Phi_1, \ldots, \Phi_k$, $b_1, \ldots, b_k$: cluster parameters, k: number of clusters

1. Initialize k to 1, and set z_i to 1 for $i = 1, \ldots, n$;
2. Initialize Φ_1 with a randomized matrix and b_1 with 0;
3. Repeat until convergence or reaching the max number of epochs;
foreach (x_i, y_i) **do**
  Compute $sim_c(x_i, y_i)$ for $c = 1, \ldots, k$;
  if $\max_c sim_c(x_i, y_i) < \lambda$ **then**
    $k = k + 1, \;\; z_i = k$;
    Initialize Φ_k with a randomized matrix and b_k with 0;
  else
    $z_i = \arg\max_c sim_c(x_i, y_i)$;
  end
  Update Φ_{z_i} and b_{z_i} according to Equation 3;
end
4. Return $\Phi_1, \ldots, \Phi_k$, $b_1, \ldots, b_k$, and k;

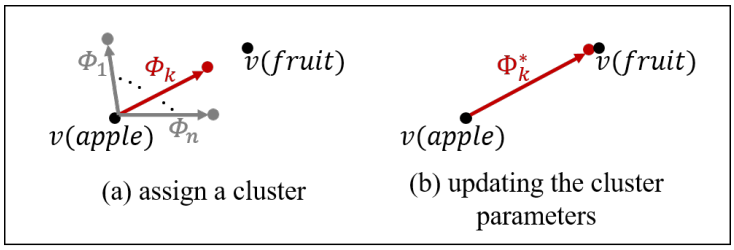


Figure 1: Overview of our proposed joint learning model

the projection matrices are trained independently. Further, this model does not consider non-hypernymy pairs during training, so it is not clear how the model performs on non-hypernymy pairs.

3 Jointly Learning Clusters and Projections of Hypernymy

This paper proposes a novel model that learns projections from words to their hypernyms and clusters of hypernymy jointly. Our model automatically estimates the number of clusters, motivated by DP-means clustering (Kulis and Jordan, 2012) that automatically estimates the number of clusters during clustering unlike k-means.

In learning the projections and clusters, our model receives d-dimensional word embeddings and training word pairs as its input, clusters the pairs with updating the cluster parameters ($d \times d$ projection matrices and bias terms in a similarity measure), and produces cluster parameters as its output. After the parameters are learned, our model generates hypernyms for a given word by projecting a word by using the projection matrices and selecting words that have the highest similarities with the projected word.

We will detail the learning and hypernym generation processes in the rest of this section.

3.1 Joint learning

Our joint learning model starts learning with a single cluster that includes all the training pairs and learns projection matrices with automatically increasing the number of clusters during learning. The entire algorithm is shown in Algorithm 1.

Our model updates the parameters of the cluster each training pair belongs to. The training pair consists of word x_i and its hypernym y_i. For each update, the model first finds (or generates) an appropriate

cluster for the target pair and then updates the parameters of the cluster as in Figure 1.

To find an appropriate cluster for a training pair, our model first calculates the similarity between the words in the pair for all the clusters and adds the pair to a cluster that achieves the highest similarity. Our similarity measure is based on an inner product, which is used in learning word embeddings. The similarity is defined as:

$$sim_c(x_i, y_i) = \sigma\left(\Phi_c \mathbf{x}_i \cdot \mathbf{y}_i + b_c\right). \tag{2}$$

Here, Φ_c denotes a $d \times d$ projection matrix for c-th cluster, b_c denotes bias of c-th cluster, σ denotes a logistic sigmoid function, and $\mathbf{x}_i$ and $\mathbf{y}_i$ are normalized word embeddings of x_i and y_i. This similarity measure calculates how close the matrix Φ_c projects x_i to y_i. Using the similarities, our model categorizes the pair into a cluster with the highest similarity if the similarity is beyond a threshold λ. If all the similarities are below the threshold and no matrix projects $\mathbf{x}$ close enough to $\mathbf{y}$, our model instead generates a new cluster and assign the cluster to the pair.

After a cluster is assigned to a pair, our model updates the cluster parameters (Φ and b) of the cluster for projection matrix and similarity measure. During the update of the parameters, motivated by negative sampling (Mikolov et al., 2013), our model generates pseudo negative non-hypernyms and penalizes the parameters so that the model do not generate the non-hypernyms in prediction. Our model maximizes the following objective function:

$$J = \sum_{c=1}^{k} \sum_{(x,y) \in P_c} \left(\log \mathrm{sim}_c\left(x, y\right) + \sum_{i=1}^{m} \log\left(1 - \mathrm{sim}_c\left(x, y_i'\right)\right) \right). \tag{3}$$

Here, k is the current number of clusters, $y_i' (\neq y)$ denotes a negative sample, and m denotes the number of negative samples. For the negative samples, our model selects the most confusing words that show the highest similarities with x because this selection produced better results than uniform sampling in our preliminary experiment.

Although our model employs clustering and projection matrices, it is substantially different from Fu et al. (2015). Our model newly incorporates the following points into hypernym generation:

- our model jointly learns clusters of hypernymy and the cluster parameters, i.e., projection matrices Φs and biases bs, so that the clusters and their parameters are well tuned to hypernymy generation and the cluster parameters are tuned so that similarity measures are consistent among all the clusters.
- our model automatically determines the number of clusters based on the similarity threshold λ in order to determine the appropriate number of clusters for the target data set.
- our model employs the same similarity measure as that used in training word embeddings, in order to keep consistency of word similarity in our model and word embedding training.
- our model uses negative non-hypernymy instances during training, which allows the model not only to keep the projection of hyponym far from wrong hypernyms but also to deal with non-hypernymy instances in prediction.

3.2 Hypernym generation

Since we do not know which cluster a pair of x and w belongs to, we check all the clusters and select the most appropriate cluster that produces the highest similarity for the pair. When generating hypernyms for a word x, we calculate the similarity score for each word w in our vocabulary as follows:

$$score_{gen}(x, w) = \max_c sim_c(x, w). \tag{4}$$

After obtaining the scores for all the words in our vocabulary, the words with the highest scores are selected as the hypernyms of x.

	Japanese	English
model	ivLBL	word2vec
data	Yahoo Answers	Google News
dimension	300	300
window size	5	5
#vocabulary	1 million	3 million
#words	$\sim$10 billion	$\sim$100 billion

Table 1: Data and parameters in learning word embeddings

	Japanese (ALAGIN)	English (Baroni)
#train	14,814	970 (2,011)
#validation	1,752	69 (223)
#test	4,598	346 (536)
#total	21,164	1,385 (2,770)
#hypernyms / #hyponym	≈ 1	≈ 1.22

Table 2: The number of hypernymy relations and the average number of hypernyms per hyponym in the hypernymy data sets. The numbers in parentheses show the total number of positive and negative pairs in the English data set.

4 Evaluation Settings

We evaluate our model on Japanese and English hypernymy data sets. We will first explain resources for these two language data sets, and then explain the task settings.

4.1 Data sets

Our model requires word embeddings and training data of hypernymy pairs.

We obtained word embeddings for Japanese by applying ivLBL (Mnih and Kavukcuoglu, 2013) to texts crawled from Japanese Yahoo Answers (Yahoo Chiebukuro)[1]. As for English, we used a pre-trained Google News word embeddings[2] (Mikolov et al., 2013), which has shown high performance in several word similarity tasks. Table 1 summarizes the settings in learning word embeddings. We use the words in the word embedding as our vocabularies.

To obtain supervision for training hypernymy, we used ALAGIN typed hierarchies for Japanese[3], and the data by Baroni et al. (2012) for English. The statistics of the data are shown in Table 2. Word pairs in ALAGIN were made by automatically extracting word hierarchy from Wikipedia (Sumida et al., 2008), selecting the top-level pairs, and manually cleaning the selected pairs. Since this data deal with not only words but phrases, we selected hypernymy pairs that include only words and split the pairs into training, validation, and test pairs. Word pairs of Baroni's data were extracted from WordNet. While the data originally include 1,385 positives and 1,385 negatives that are from a random permutation of positive pairs, we used only positive pairs for training because our model generates negative pairs during training.

4.2 Task and evaluation settings

We optimized our model parameters with Adam (Kingma and Ba, 2015) with recommended parameters in their paper. We set the number of negative samples m to 1, according to the preliminary experiment. We tuned other hyperparameters, e.g., λ in our model and k in k-means, with the validation data. As a result of several preliminary experiments, we decided other hyperparameters as well. We consider only direct hypernymy in our experiment, differently from Fu et al. (2015). For instance, we consider only

[1] http://chiebukuro.yahoo.co.jp/

[2] https://code.google.com/archive/p/word2vec/

[3] https://alaginrc.nict.go.jp/resources/nict-resource/li-info/li-outline.html#A-4

	Japanese (ALAGIN)		English (Baroni)	
	pipeline	joint	pipeline	joint
#clusters	5	19	1	2
MRR	0.193	**0.349**	0.280	**0.339**

Table 3: MRR of our joint model and the pipeline model on the test parts of the Japanese ALAGIN data set and English Baroni's data set.

(*apple*, *fruit*) and (*fruit*, *plant*) pairs for the hypernymy hierarchy *apple* $\underset{H}{\longrightarrow}$ *fruit* $\underset{H}{\longrightarrow}$ *plant*, and do not consider (*apple*, *plant*). Here, *A* $\underset{H}{\longrightarrow}$ *B* means that *B* is a hypernym of *A*.

We evaluated our model on the task of *hypernym generation* using both the Japanese ALAGIN data set and the Baroni's data set. We generate hypernyms from our vocabulary in word embeddings.

To compare our model with the existing projection matrix-based pipeline model of Fu et al. (2015), we implemented their pipeline model, i.e., k-means clustering and projection matrix learning with the least square objective as explained in Section 2, but we did not include indirect hypernymy and employed Adam instead of stochastic gradient descent to optimize the model parameters (projection matrices). In generating hypernyms for a word x, we assigned a cluster c to the pair (x, w) by k-means for each word w in our vocabulary. We then selected words that had the lowest least square distances as the hypernyms of x. Here, the least square distances are defined as follows:

$$distance_c(x, w) = \|\Phi_c \mathbf{x} - \mathbf{w}\|^2, \tag{5}$$

where $\mathbf{w}$ denotes the word embedding of w. We call this model as a *pipeline* model, in contrast to our *joint* model.

We use Mean Reciprocal Rank (MRR) for the evaluation of hypernymy generation.

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i}, \tag{6}$$

where N is the amount of test data, $rank_i$ is the rank which correct hypernym is generated. MRR ranges from 0 to 1, and higher MRR indicates better performance of the model.

We also used the Baroni's data set to evaluate the performance of our joint model on a well-studied *hypernymy classification* task. We compared our model with the state-of-the-art supervised distributional model by Levy et al. (2015). Since our model is not classification-based, we performed the following simple threshold-based classification in order to judge whether a given word pair is in hypernymy or not:

$$score_{gen}(x, w) \begin{cases} \geq 0.5 & \Rightarrow \text{ positive} \\ otherwise & \Rightarrow \text{ negative.} \end{cases} \tag{7}$$

5 Evaluation

We present the results of hypernym generation evaluation in section 5.1 and those of hypernymy classification evaluation in section 5.2.

5.1 Hypernym generation

We compared MRRs of our joint model with those of the pipeline model on Japanese and English test data sets. The results, along with the numbers of clusters, are summarized in Table 3. These numbers of clusters are tuned using the validation data sets. In both Japanese and English test data sets, our joint model outperformed the pipeline model. Further, in our model, the number of clusters tends to be larger than that of the pipeline model. This may be partly because our joint model can capture more differences in hypernymy relations than the *pipeline* model, which used offsets, i.e., $\langle \mathbf{y} - \mathbf{x} \rangle$, in the clustering. Note that the small numbers of clusters in English data were considered to stem from the size of the data set;

Japanese (ALAGIN)		English (Baroni)	
pipeline (least squares)	joint (inner products)	pipeline (least squares)	joint (inner products)
0.215	**0.279**	0.321	**0.343**

Table 4: MRR of our model with inner product-based similarity measure and the pipeline model with least squares-based similarity measure without clustering on the validation parts of the Japanese ALAGIN data set and English Baroni's data set.

λ	0	0.1	0.15	0.2
#clusters	1	5	19	44
MRR	0.279	0.327	**0.368**	0.353

Table 5: MRR and the number of clusters on the validation part of the Japanese ALAGIN data set.

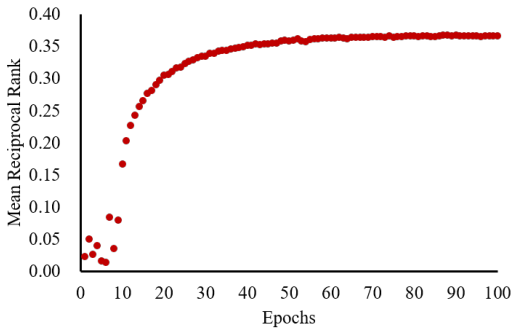


Figure 2: Learning curve of our joint model on the Japanese ALAGIN validation data set.

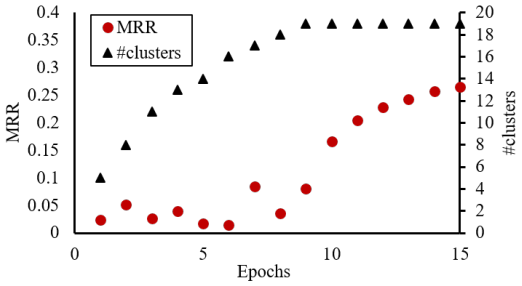


Figure 3: MRR on the validation set and the number of clusters of our joint model in the early learning stage on Japanese ALAGIN data set.

the English data set was about 6.5% of the Japanese data set in the number of positive instances as in Table 2.

We compare our model and the pipeline model without clustering on the Japanese validation data set in Table 4. Our model uses the inner product-based similarity in Equation 2 and negative sampling in Equation 3, while the pipeline model used the least square distance as in Equation 5. The results show that our model produces better results than the least square-based model.

The threshold of the clustering is a crucial parameter to decide the number of the clusters in our model. Table 5 shows the number of clusters and MRR on the Japanese validation data set when we varied the clustering threshold λ.

Figure 2 shows the learning curve of our joint model with the best hyperparameter on Japanese validation data set. This learning curve is stable except for the initial stage of learning (#epochs ≤ 9). To understand the behavior of the learning in the initial stage, we show MRR and the number of clusters in the first 15 epochs of learning our models in Figure 3. The number of clusters of our model increases in the initial learning stage (#epochs ≤ 9), and it stops to increase after several iterations. Due to this increase of clusters, MRR is unstable in the initial stage of learning. MRR starts to improve constancy after the number of clusters is fixed.

Our joint model and the pipeline model are different from other existing hypernymy classification models in that the former models can generate hypernyms from a given word. We present two examples of estimated hypernyms for given words in Tables 6 and 7.

5.2 Hypernym classification

We evaluated our model on the hypernym classification task using the Baroni's English data set. Our model produces 0.766 in F_1 score, while the model of Levy et al. (2015) compared several similarities

	joint		pipeline	
rank	estimated hypernym	$score_{gen}$	estimated hypernym	$\|\Phi\mathbf{x} - \mathbf{y}\|^2$
1	企業 “firm”	0.974	企業 “firm”	0.617
2	会社 “company”	0.752	有名企業 “famous firm”	0.696
3	グループ “group”	0.748	関連会社 “associated company”	0.726
4	法人 “corporation body”	0.358	自動車メーカー “auto manufacturer”	0.736
5	メーカー “manufacturer”	0.283	会社 “company”	0.737

Table 6: Examples of estimated hypernyms for a word トヨタ自動車 “Toyota Motor Co.”.

	joint		pipeline	
rank	estimated hypernym	$score_{gen}$	estimated hypernym	$\|\Phi\mathbf{x} - \mathbf{y}\|^2$
1	学 “study”	0.987	学問 “science”	0.645
2	理論 “theory”	0.730	方法論 “methodology”	0.710
3	システム “system”	0.692	用語 “technical term”	0.737
4	手法 “method”	0.678	手法 “method”	0.743
5	技術 “technique”	0.504	アルゴリズム “algorithm”	0.768

Table 7: Examples of estimated hypernyms for a word 機械学習 “machine learning”.

measures and reported the state-of-the-art F_1 score of 0.802. In this comparison, we did not tune our hypernym generation model for the classification task. Instead, we directly applied our best hypernymy generation model in Table 3 to the task by using the hypernymy generation scores as classification scores (see. Equation 7). According to the results, our joint model compares favorably to the state-of-the-art model.

6 Conclusions

This paper proposed a novel word embedding-based hypernym generation model that clusters hyponym-hypernym relations and learns the parameters for the projection from hyponyms to their hypernyms in each cluster. Unlike many existing hypernym classification models, our proposed model can generate hypernyms from a given word. This generation will help to directly apply our model to several NLP tasks. Furthermore, the number of clusters is automatically determined according to the word embeddings and the training pairs.

We employ the inner product-based similarity measures so that word similarity can be consistent in training word embeddings and in our model. We also employ negative sampling so that the generated hypernyms can not be close to wrong hypernyms. We evaluate our model over Japanese and English data sets. Our proposed model significantly outperformed the other hypernym generation model on both Japanese and English data sets. We also evaluated our model on the hypernymy classification in English, and, without task-specific tuning, the result showed the F_1 score comparable to the state-of-the-art model.

In future work, we aim to extend our model for indirect hypernyms or hyponymy generation. This extension can improve the model performance and also widen the application areas of our model.

References

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32, Avignon, France, April. Association for Computational Linguistics.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2015. Learning semantic hierarchies: A continuous vector space approach. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 23(3):461–471.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceed-*

ings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), pages 107–114, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations*.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(04):359–389.

Zornitsa Kozareva and Eduard Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1110–1118, MIT, Massachusetts, USA, October. Association for Computational Linguistics.

Brian Kulis and Michael I Jordan. 2012. Revisiting k-means: New algorithms via bayesian nonparametrics. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 513–520.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976, Denver, Colorado, May–June. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2265–2273. Curran Associates, Inc.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1025–1036, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems*, pages 1297–1304.

Asuka Sumida, Naoki Yoshinaga, and Kentaro Torisawa. 2008. Boosting precision and recall of hyponymy relation acquisition from hierarchical layouts in wikipedia. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may. European Language Resources Association (ELRA).

Liling Tan, Rohit Gupta, and Josef van Genabith. 2015. Usaar-wlv: Hypernym generation with deep neural nets. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 932–937, Denver, Colorado, June. Association for Computational Linguistics.

Peter D Turney and Saif M Mohammad. 2015. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering*, 21(03):437–476.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2249–2259, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Incremental Fine-grained Information Status Classification Using Attention-based LSTMs

Yufang Hou
IBM Research Ireland, Dublin, Ireland
yhou@ie.ibm.com

Abstract

Information status plays an important role in discourse processing. According to the hearer's common sense knowledge and his comprehension of the preceding text, a discourse entity could be `old`, `mediated` or `new`. In this paper, we propose an attention-based LSTM model to address the problem of fine-grained information status classification in an incremental manner. Our approach resembles how human beings process the task, i.e., decide the information status of the current discourse entity based on its preceding context. Experimental results on the ISNotes corpus (Markert et al., 2012) reveal that (1) despite its moderate result, our model with only word embedding features captures the necessary semantic knowledge needed for the task by a large extent; and (2) when incorporating with additional several simple features, our model achieves the competitive results compared to the state-of-the-art approach (Hou et al., 2013) which heavily depends on lots of hand-crafted semantic features.

1 Introduction

Information status (IS) (Halliday, 1967; Prince, 1981; Nissim et al., 2004) accounts for the familiarity of a discourse entity according to its accessibility to the hearer at a given point in the text, e.g., `old` mentions[1] are known to the hearer and have been referred to previously; `mediated` mentions have not been mentioned before but are accessible to the hearer by reference to another `old` mention or to prior world knowledge; `new` mentions are "not being recoverable from the preceding discourse" (Halliday, 1967).

Information status has attracted a large amount of interests in theoretical linguistics under the framework of *information structure* (Halliday, 1967; Prince, 1981; Prince, 1992; Gundel et al., 1993; Lambrecht, 1994; Birner and Ward, 1998; Kruijff-Korbayová and Steedman, 2003). Many NLP tasks can benefit from knowing information status of discourse entities. Cahill and Riester (2009) improve the performance of generation ranking in German by incorporating features modeling IS. Rahman and Ng (2011) show that a coreference system can profit from IS classification. Baumann and Riester (2013) conduct an empirical study of information status in spoken German and demonstrate that IS can influence prosody in read speech. Hou et al. (2013) model bridging anaphora recognition as a subtask of learning fine-grained information status.

In this paper, we focus on classifying IS on written text because many applications which can benefit from IS concentrate on written texts. We follow the IS scheme for written text proposed by Markert et al. (2012). It adopts the three major IS categories (`old`, `new` and `mediated`) from Nissim et al. (2004) and distinguishes six subcategories for `mediated`. Section 2 provides a brief description of the scheme.

We address the task of fine-grained IS classification via an attention-based LSTM model in an incremental manner. The model resembles human beings' cognitive process of determining IS for discourse entities, i.e., during the process of reading a text from left to right, assign IS for each discourse entity according to its own property and its preceding context.

[1] A mention is a noun phrase which refers to a discourse entity and carries information status.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1880–1890, Osaka, Japan, December 11-17 2016.

Previous approaches on fine-grained IS classification (Markert et al., 2012; Hou et al., 2013) explore world knowledge by integrating hand-crafted semantic features extracted from manually and automatically constructed knowledge bases. One goal of this paper is to investigate in which extent word embeddings learned from large corpora can replace such hand-crafted semantic features. Experimental results on the ISNotes corpus (Markert et al., 2012) show that our model with only word embedding features achieves reasonable results for several IS categories. We further demonstrate that when incorporating with several additional simple features, our model achieves competitive results compared to the state-of-the-art approach (Hou et al., 2013) which heavily depends on lots of hand-crafted semantic features.

2 An Overview of Information Status in ISNotes

ISNotes (Markert et al., 2012) contains 10,980 mentions annotated for information status in 50 texts taken from the Wall Street Journal portion of the OntoNotes corpus (Weischedel et al., 2011). Below we briefly illustrate the definitions of eight IS categories with examples.

A mention is `old` if it is either coreferent with an already introduced entity, or if it is a generic or deictic pronoun.

`Mediated` mentions have not been mentioned before but are not autonomous, i.e., they can only be correctly interpreted by reference to another mention or to prior world knowledge. ISNotes distinguishes six subcategories of `mediated` mentions:

- `mediated/worldKnowledge` mentions are generally known to the hearer. This category includes many proper names, such as *Germany*.
- `mediated/syntactic` mentions are syntactically linked via a possessive relation, a proper name premodification or a PP (prepositional phrase) postmodification to other `old` or `mediated` mentions, such as:

  [[their]$_{old}$ liquor store]$_{mediated/syntactic}$,

  [the [Federal Reserve]$_{mediated}$ boss]$_{mediated/syntactic}$, and

  [the main artery into [San Francisco]$_{mediated}$]$_{mediated/syntactic}$.
- `mediated/bridging` mentions are inferable because a related entity or event (antecedent) has been previously introduced in the discourse, such as **the streets** in Example 1.
- `mediated/comparative` mentions usually include a premodifier that makes clear that this entity is compared to a previous one (antecedent), such as **others** in Example 2.
- `mediated/aggregate` mentions are coordinated mentions where at least one element in the conjunction is `old` or `mediated`, such as *[Not only [George Bush]$_{mediated}$ but also [Barack Obama]$_{mediated}$]$_{mediated/aggregate}$.*
- `mediated/function` mentions refer to a value of a previously explicitly mentioned function (e.g., **3 points** in Example 3). The function needs to be able to rise and fall.

(1) *Oranjemund, the mine headquarters*, is a lonely corporate oasis of 9,000 residents. Jackals roam **the streets** at night ...

(2) As the death toll from last week's temblor climbed to 61, the condition of *freeway survivor Buch Helm*, who spent four days trapped under rubble, improved, hospital officials said. Rescue crews, however, gave up hope that **others** would be found.

(3) IBM shares were down$_{function}$ **3 points**.

`New` mentions are entities that have not yet been introduced in the discourse and that the hearer cannot infer from either previously mentioned entities/events or general world knowledge.

Table 1 shows the IS distribution in ISNotes which contain 1,726 sentences in total.

Mentions	10,980	
old	3237	29.5%
mediated	3,708	33.8%
syntactic	1,592	14.5%
world knowledge	924	8.4%
bridging	663	6.0%
comparative	253	2.3%
aggregate	211	1.9%
func	65	0.6%
new	4,035	36.7%

Table 1: IS distribution in ISNotes. The last column indicates the percentage of each IS category relative to the total number of mentions.

3 The Attention-based LSTM Model

In this section we first briefly describe LSTMs in Section 3.1. We then detail our attention-based LSTM model for fine-grained IS classification in Section 3.2.

3.1 LSTMs

Recently, recurrent neural networks (RNNs) with long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997) have been empirically shown to perform well in a range of NLP tasks, such as machine translation (Sutskever et al., 2014), parsing (Vinyals et al., 2015), and sentence compression (Filippova et al., 2015). LSTMs contain special units called memory blocks in the recurrent hidden layer. These memory blocks are designed to avoid vanishing gradients and to remember some long-distance dependencies from the input sequence. The vanilla LSTM model with hidden size k is defined as follows: given a sequence of input $(x_1, ..., x_T)$, LSTMs compute the *h*-sequence and the *m*-sequence using the following equations iteratively from t=1 to T:

$$i_t = sigm(W_1 x_t + W_2 h_{t-1}) \quad (1) \qquad f_t = sigm(W_2 x_t + W_3 h_{t-1}) \quad (2)$$

$$o_t = sigm(W_5 x_t + W_6 h_{t-1}) \quad (3) \qquad i_t\prime = tanh(W_7 x_t + W_8 h_{t-1}) \quad (4)$$

$$m_t = m_{t-1} \odot f_t + i_t \odot i_t\prime \quad (5) \qquad h_t = o_t \odot tanh(m_t) \quad (6)$$

The operator $\odot$ denotes element-wise multiplication, and $sigm$ and $tanh$ are computed element-wise. The matrices $W_1, \ldots, W_8$ and the vector h_0 are the parameters of the model.

3.2 Incremental IS Classification with Attention-based LSTMs

Model. In practice LSTMs still have difficulties to handle long-range dependencies because the model tries to encode the full input sequence into a fixed-length vector. To alleviate this problem, attention-based LSTMs allow the decoder to "attend" the different part of the input sequence when making the prediction. In an IS classification scenario, for each document, the attention-based LSTM model reads the mentions from left to right, and predicts each mention's IS output according to (1) the current mention's state cell and (2) weighted representation of the preceding mentions.

Figure 1 shows the high-level structure of our model. More precisely, for a mention m_i and its preceding t mentions, let two LSTMs read the mentions from left to right. The first LSTM uses the sum of word embeddings as mention representations, whereas the second LSTM uses one-hot vectors as mention representations[2]. Let k_1 and k_2 be the hyper-parameters denoting the size of mention representations and

[2]Another choice is to concatenate the sum of word embeddings with one-hot vectors and use only one LSTM. In practice, we find that encoding them with two LSTMs works better.

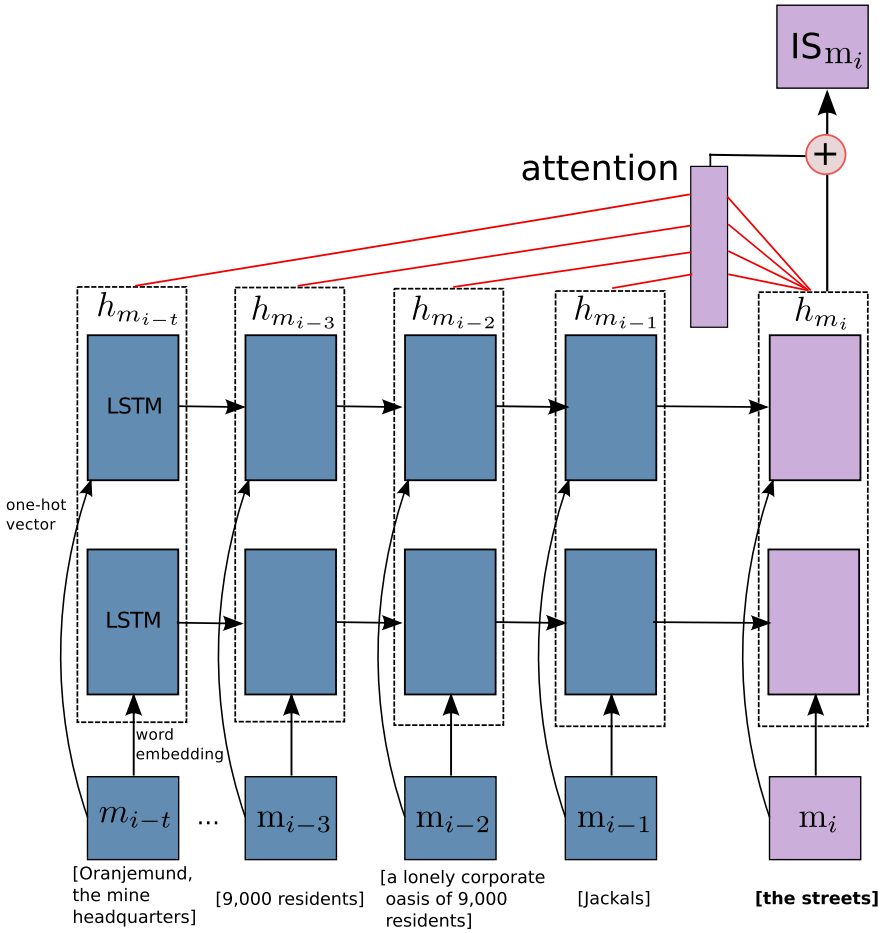


Figure 1: Fine-grained IS classification using attention-based LSTMs.

hidden layers in the two LSTMs respectively[3], and $H_1 \in \mathbb{R}^{k_1 \times (t+1)}$ and $H_2 \in \mathbb{R}^{k_2 \times (t+1)}$ to denote the output vectors from the first LSTM and the second LSTM. We then stack the first t output vectors from the two LSTMs:

$$H = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}, H \in \mathbb{R}^{(k_1+k_2) \times t} \tag{7}$$

Let $k = k_1 + k_2$, we define an attention vector α over the preceding t mentions and their weighted representation r as follows:

$$M = tanh(W_H H + [(W_{m_i} h_{m_i})_{\times t}]), M \in \mathbb{R}^{k \times t} \tag{8}$$

$$\alpha = softmax(W^T M), \alpha \in \mathbb{R}^t \tag{9}$$

$$r = H\alpha^T, r \in \mathbb{R}^k \tag{10}$$

where h_{m_i} is the stacked output vector of mention m_i from the two LSTMs, the matrices $W_H, W_{m_i} \in \mathbb{R}^{k \times k}$ and the vector $W \in \mathbb{R}^k$ (W^T denotes its transpose) are learned parameters of the model. Note that we repeat the linear transformation of the state cell of mention m_i (i.e., $W_{m_i} h_{m_i}$) t times. As a result, each column in M is the attention representation for each preceding mention $m_j (i - t \leq j < i)$ by combining the output vector h_{m_i} of mention m_i and the output vector of mention m_j (j's column vector in H).

[3]The size of the hidden layer in each LSTM is equal to its mention representation size.

We obtain the final representation of m_i using:

$$\hat{h}_{m_i} = tanh(W_1 r + W_2 h_{m_i}), \hat{h}_{m_i} \in \mathbb{R}^k \quad (11)$$

where the matrices $W_1, W_2 \in \mathbb{R}^{k \times k}$ are the model's parameters. Finally, we use a softmax layer to project $\hat{h}_{m_i}$ into the target space of eight IS classes.

Training instances. For all mentions in a document, we first add a dummy mention with the span of $[-1, -1]$ in the beginning of the document. We then order all mentions according to their end positions in ascending order; if two mentions have the same end position, we order them according to their start positions in descending order. This rule ensures that for embedded mentions, the inside mention is ordered before its parent. Such arrangement of embedded mentions is important because for `mediated/syntactic` and `mediated/aggregate`, a mention's IS label is dependent on the IS labels of its (syntactic) children. Table 2 shows several examples of how embedded mentions are ordered under this rule.

embedded mentions	result of ordering
(1) [[their] liquor store]	[their] – [their liquor store]
(2) [the [Federal Reserve] boss]	[Federal Reserve] – [the Federal Reserve boss]
(3) [the main artery into [San Francisco]]	[San Francisco] – [the main artery into San Francisco]
(4) [[he] and [[his] skilled team]]	[he] – [his] – [his skilled team]–[he and his skilled team]

Table 2: Results of ordering for embedded mentions.

After ordering, we create a training instance for each mention using its preceding mentions as the context. The training instances are created in an incremental manner. For instance, given a document containing the sentence shown in Example 1 (Section 2), the training instances will be (m_0 is the dummy mention):

- m_0 || **[the mine headquarters]**
- m_0*–[the mine headquarters]* || **[Oranjemund, the mine headquarters]**
- m_0*–[the mine headquarters]–[Oranjemund, the mine headquarters]* || **[9,000 residents]**
- m_0*–[the mine headquarters]–[Oranjemund, the mine headquarters]–[9,000 residents]* || **[a lonely corporate oasis of 9,000 residents]**
- m_0*–[the mine headquarters]–[Oranjemund, the mine headquarters]–[9,000 residents]–[a lonely corporate oasis of 9,000 residents]* || **[Jackals]**
- m_0*–[the mine headquarters]–[Oranjemund, the mine headquarters]–[9,000 residents]–[a lonely corporate oasis of 9,000 residents]–[Jackals]* || **[the streets]**
- m_0*–[the mine headquarters]–[Oranjemund, the mine headquarters]–[9,000 residents]–[a lonely corporate oasis of 9,000 residents]–[Jackals]–[the streets]* || **[night]**

Decoding. In the testing stage, given a document and its ordered mentions based on the rule described above, we predict IS classes for these mentions incrementally from left to right. Because a mention's IS could depends on the IS labels of its context mentions, we also encode the IS class as one-hot representation. The gold standard labels and the predicted IS labels of the context mentions are used for training and decoding respectively[4].

[4]The IS class one-hot representation for the target mention is a zero vector during training and decoding.

Network parameters. We use Adam (Kingma and Ba, 2015) for optimization with the learning rate of 0.01. We train all models with 10 epochs using cross-entropy loss. To avoid over-fitting, we apply dropout before and after the LSTM layer with the probability of 0.1. For each mention, we set the maximum number of its context mentions as 50[5]. Therefore we unfold the network 51 times and apply masking for the instances which have less than 50 context mentions.

We use GloVe vectors (Pennington et al., 2014) with 100 dimensions trained on Wikipedia and Gigaword as word embeddings, which we do not optimize during training. Out-of-vocabulary words in the training set and the testing set are set to fixed random vectors. We approximate mention representations fed into the first LSTM by summing embeddings of all words from a mention as Yu and Dredze (2015) show that sum of word embeddings achieves reasonable result to induce phrase embeddings. In ISNotes, around 30% of mentions contain only one word and around 70% of mentions contain less than four words. Mention representations fed into the second LSTM are one-hot vectors of the mentions' features and their IS classes.

4 Experiments

4.1 Experimental Setup

We perform experiments on the ISNotes corpus (Markert et al., 2012). Following Hou et al. (2013), all experiments are performed via 10-fold cross-validation on documents. We use gold standard mentions and the OntoNotes syntactic annotation layer for feature extraction. We report overall accuracy as well as precision, recall and F-measure per IS category. In the following, we describe the baseline and our model with different feature settings.

Baseline. Hou et al. (2013) report the state-of-the-art performance for fine-grained IS classification on ISNotes using collective classification. They explore a wide range of features (34 in total), including a large number of lexico-semantic features as well as a couple of surface features and syntactic features. Hou et al. (2013) observe that bridging anaphors are rarely marked by surface features. Therefore they carefully design discourse structure, lexico-semantic and genericity detection features to capture the phenomenon. The semantic features are extracted from manually or automatically constructed knowledge bases, such as WordNet (Fellbaum, 1998) and the General Inquirer lexicon (Stone et al., 1966).

LSTM. To test how well we can predict IS for mentions without using any hand-crafted features, we only use word embeddings and IS labels in our attention-based LSTM model described in Section 3.2. Specifically, we use mention embeddings (100 dimensions) as the input of the first LSTM. Mention embeddings are obtained by summing word embeddings of all words from a mention, where word embeddings are from GloVe vectors trained on Wikipedia and Gigaword. We use one-hot vectors (8 dimensions) to encode IS labels and use them as the input of the second LSTM.

LSTM+PAR. Hou et al. (2013) show that coordination parent-child relations and other syntactic parent-child relations among mentions are highly effective for `mediated/coordination` and `mediated/syntactic` classes. We use one-hot vectors (2 dimensions) to integrate such parent-child information into the LSTM model described above. Table 3 demonstrates two examples of one-hot representation for parent-child relations.

[[their] liquor store]	$[\text{their}]_{[1,0]} \parallel \textbf{[their liquor store]}_{[1,0]}$
[[he] and [[his] skilled team]]	$[\text{he}]_{[0,1]} - [\text{his}]_{[0,0]} - [\text{his skilled team}]_{[0,1]} \parallel \textbf{[he and his skilled team]}_{[0,1]}$

Table 3: one-hot representations for parent-child relations.

LSTM+PAR+FEAT. We hypothesize that knowledge about a mention's surface and syntactic properties can be useful to decide its information status. Therefore, we add a small feature set (see Table 4) from

[5]In practice, we find that in our model, the results are similar with the maximum number of context mentions as 10, 20, or 50.

Hou et al. (2013) into our attention-based LSTM model (*LSTM+PAR*) using one-hot representations. $f1$-$f5$ are surface and syntactic features, and $f6$-$f8$ are three simple lexical-semantic features. $f6$-$f8$ provide additional semantic knowledge for three mediated classes (i.e., `mediated/comparative`, `mediated/worldKnowledge` and `mediated/function`) that our current mention embedding representations do not capture well.

Feature	**Value**
$f1$ FullPrevMention	{yes, no, NA}
$f2$ PartialPreMention	{yes, no, NA}
$f3$ Determiner	{bare, def, dem, indef, poss, NA}
$f4$ NPtype	{pronoun, common, proper, other}
$f5$ GrammaticalRole	{subject, subjectPassive, pp, other}
$f6$ PreModByCompMarker	{yes, no}
$f7$ IsFrequentProperName	{yes, no}
$f8$ DependOnChangeVerb	{yes, no}

Table 4: A small feature set from Hou et al. (2013).

4.2 Results and Discussion

Results. Table 5 shows the results of our models compared to the baseline. Our model with word embeddings and only a couple of simple features (*LSTM+PAR+FEAT*) performs as good as the state-of-the-art approach (Hou et al., 2013) which explores a wide range of semantic features based on various knowledge resources. It is worth noting that the model with only word embeddings (*LSTM*) achieves an accuracy of 66.8. Also *LSTM* performs similar as the baseline for bridging anaphora recognition under the multi-class classification setting. This indicates that word embeddings in our model do capture certain semantics needed for the task. The improvement in *LSTM+PAR* over *LSTM* confirms the effectiveness of the two parent-child relations for `mediated/syntactic` and `mediated/aggregate` categories.

Comparing the results of *LSTM+PAR+FEAT* to *LSTM+PAR* and *LSTM+PAR+FEAT−wordEmb*, it seems that word embeddings and the small set of simple features (most of them are capturing the surface and syntactic properties of mentions) are complementary. Specifically, `mediated/function` and `mediated/bridging` benefit most from word embeddings which provide useful semantic knowledge to capture these two categories. On the contrary, the feature set (*FEAT*) provides better generalization capability for `old`, `mediated/worldKnowledge` and `mediated/comparative`.

	baseline Hou et al.(2013)			*LSTM*			*LSTM+PAR*			*LSTM+PAR +FEAT*			*LSTM+PAR +FEAT−wordEmb*		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
old	84.4	86.0	**85.2**	75.7	75.6	75.6	77.9	71.2	74.4	85.4	84.9	**85.2**	83.3	85.7	84.5
m/worldKnow.	67.4	77.3	**72.0**	45.6	52.6	48.8	39.8	53.1	45.5	67.1	74.5	70.6	60.4	65.1	62.7
m/syntactic	82.2	81.9	**82.0**	63.6	63.8	63.7	80.4	73.4	76.7	80.8	81.9	81.4	76.4	79.8	78.1
m/aggregate	64.5	79.5	71.2	11.8	35.2	17.7	50.7	65.2	57.1	67.8	84.6	**75.3**	65.9	86.9	74.9
m/function	67.7	72.1	69.8	46.2	57.7	51.3	26.2	53.1	35.1	64.6	76.4	**70.0**	12.3	88.9	21.6
m/comparative	81.8	82.1	**82.0**	15.0	34.9	21.0	14.2	38.7	20.8	77.9	83.1	80.4	78.3	80.8	79.5
m/bridging	19.3	39.0	**25.8**	16.3	36.9	22.6	18.7	34.0	24.1	15.7	32.3	21.1	0.0	0.0	NaN
new	86.5	76.1	**81.0**	80.5	67.3	73.3	76.2	70.8	73.4	87.2	74.8	80.5	85.0	68.2	75.7
acc		**78.9**			66.8			68.6			78.6			75.1	

Table 5: Experimental results of the attention-based LSTM models compared to the baseline. Bolded scores indicate the best performance for each IS class. There is no significant difference between *LSTM+PAR+FEAT* and the baseline at the level of $p < 0.01$ (Statistical significance is measured using McNemar's χ^2 test (McNemar, 1947)).

The incremental prediction mechanism in our model utilizes the (predicted) IS class information of previous mentions when predicting the IS class for the current mention. To gain a better understanding of such mechanism, we conduct an experiment by removing IS label information from our best model (thus *LSTM+PAR+FEAT - ISLabels*). This leads to a mild decrease in the overall accuracy (from

78.6 to 77.7). When looking at the results, we found that the decrease is centered on the categories of `mediated/syntactic`, `mediated/aggregate` and `new`. This confirms that our incremental decoding strategy helps the model to capture the IS label dependencies among mentions better.

Analysis of attention mechanism. We further analyze the attention mechanism in our model *LSTM+PAR* by manually checking some testing examples from one fold. We choose this setting because we want to investigate whether the model can capture long distance relations between mentions without being informed by the features which indicate whether a mention is fully or partially mentioned before.

Figure 2 shows heat maps of several examples that our model predicts correctly[6]. Note that we must take into account that the model only partially relies on representations obtained from attention, i.e., in Equation 11, the final prediction depends on the combination of attention representation as well as the long range contextual representation obtained from LSTM encoders.

It is interesting to see that in the first example, the model attends to several reasonable antecedents for the pronoun "[it]" when predicting its information status. In the second example, when predicting information status for "[the kingdom]", the model focuses on its antecedent "[Saudi Arabia]". In the third and the fourth examples, the model focuses on the syntactic children when predicting information status for "[its percentage share of OPEC production]" and "[Motorola and other companies]".

We also notice that for `old` mentions, when their antecedents do not appear in the preceding context mentions, the weights of attention are more uniformly distributed. Furthermore, for correctly predicted `mediated/comparative` and `mediated/bridging` mentions, we do not observe clear patterns in their attention weights. We assume this is because we have less training data for these two categories. In addition, only a few of them have antecedents occurring in the preceding ten mentions. Therefore, the model seems mainly uses the last output vector (h_{m_i} in Equation 11) for prediction.

5 Related Work

Automatic IS classification. Markert et al. (2012) applied joint inference for IS classification on the ISNotes corpus. Built on this work, Hou et al. (2013) proposed a cascading collective classification algorithm for bridging anaphora recognition with various semantic features. They report the state-of-the-art result for fine-grained IS classification using collective classification.

Rahman and Ng (2012) studied the fine-grained IS classification problem on the Switchboard dialogue corpus (Nissim et al., 2004). They first designed a rule-based system to assign IS classes to mentions. The rule-based system heavily depends on knowledge resources such as FrameNet (Baker et al., 1998), WordNet (Fellbaum, 1998), and ReVerb (Fader et al., 2011). They then applied an SVM^{multiclass} algorithm for this task by combining the prediction from the rule-based system, the ordering of the rules as well as two lexical features.

Another work on IS classification was carried out by Cahill and Riester (2012). They assumed that the distribution of IS classes within sentences tends to have certain linear patterns, e.g., *old* > *mediated* > *new*. Under this assumption, they trained a CRF model with syntactic and surface features for fine-grained IS classification on the German DIRNDL radio news corpus (Riester et al., 2010).

Our work differs from the above mentioned work in that we explore a new model which resembles human beings' cognitive process for the task and we replace hand-crafted semantic features with word embeddings which were learned from large corpora in an unsupervised manner.

Attention-based RNNs in NLP. Recently, RNNs with attention mechanisms have demonstrated success in various NLP tasks, such as machine translation (Bahdanau et al., 2015), parsing (Vinyals et al., 2015), image captioning (Xu et al., 2015), and textual entailment (Rocktäschel et al., 2016). Attention-based RNNs allow the model to access its internal memory when making predictions. This property is intuitive for our task because in order to decide a discourse entity's information status, we need to access its context and choose one (or none) discourse entity to attend.

[6]Due to the space limitation, we only show plots for maximum number of context mentions at ten. The patterns we observe here are similar for maximum number of context mentions at 20 or 50.

Figure 2: Attention heat maps.

6 Conclusions

We develop an attention-based LSTM model which draws on the recent advances in research on RNNs for fine-grained IS classification. The system imitates how human beings reason information status of a discourse entity based on its preceding context. The results indicate that our model with only pre-trained word embeddings captures semantic knowledge needed for the task by a large extent. Extending the model with several simple features improves the ability of the system, resulting in competitive results on the ISNotes corpus compared to the state-of-the-art approach which explores a broad variety of semantic features.

The model presented here is intuitive for understanding discourse entities. In the future, it would be worthwhile exploring how to extend the system to model other related discourse processing tasks, such as coreference resolution and bridging resolution.

Acknowledgements

The author would like to thank Charles Joachim for fruitful discussions and the anonymous reviewers for their valuable feedback.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR 2015), San Diego, 2015*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics,* Montréal, Québec, Canada, 10–14 August 1998, pages 86–90.

Stefan Baumann and Arndt Riester. 2013. Coreference, lexical givenness and prosody in German. *Lingua*. Accepted.

Betty J. Birner and Gregory Ward. 1998. *Information Status and Noncanonical Word Order in English*. John Benjamins, Amsterdam, The Netherlands.

Aoife Cahill and Arndt Riester. 2009. Incorporating information status into generation ranking. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing,* Singapore, 2–7 August 2009, pages 817–825.

Aoife Cahill and Arndt Riester. 2012. Automatically acquiring fine-grained information status distinctions in German. In *Proceedings of the SIGdial 2012 Conference: The 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue,* Seoul, Korea, 5–6 July 2012, pages 232–236.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing,* Edinburgh, Scotland, U.K., 27–29 July 2011.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass.

Katja Filippova, Enrique Alfonseca, Carlos Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* Lisbon, Portugal, 17–21 September 2015, pages 360–368.

Jeanette K. Gundel, Nancy Hedberg, and Ron Zacharski. 1993. Cognitive status and the form of referring expressions in discourse. *Language*, 69:274–307.

M. A. K. Halliday. 1967. Notes on transitivity and theme in English, Part 2. *Journal of Linguistics*, 3:199–244.

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

Yufang Hou, Katja Markert, and Michael Strube. 2013. Cascading collective classification for bridging anaphora recognition using a rich linguistic feature set. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing,* Seattle, Wash., 18–21 October 2013, pages 814–820.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR 2015), San Diego, 2015*.

Ivana Kruijff-Korbayová and Mark Steedman. 2003. Discourse and information structure. *Journal of Logic, Language and Information. Special Issue on Discource and Information Structure*, 12(3):149–259.

Knud Lambrecht. 1994. *Information Structure and Sentence Form*. Cambridge, U.K.: Cambridge University Press.

Katja Markert, Yufang Hou, and Michael Strube. 2012. Collective classification for fine-grained information status. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics,* Jeju Island, Korea, 8–14 July 2012, pages 795–804.

Quinn McNemar. 1947. Note on the sampling errors of the difference between correlated proportions or percentages. *Phychometrika*, 12(2):153–157.

Malvina Nissim, Shipara Dingare, Jean Carletta, and Mark Steedman. 2004. An annotation scheme for information status in dialogue. In *Proceedings of the 4th International Conference on Language Resources and Evaluation,* Lisbon, Portugal, 26–28 May 2004, pages 1023–1026.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 1532–1543.

Ellen F. Prince. 1981. Towards a taxonomy of given-new information. In P. Cole, editor, *Radical Pragmatics*, pages 223–255. Academic Press, New York, N.Y.

Ellen F. Prince. 1992. The ZPG letter: Subjects, definiteness, and information-status. In W.C. Mann and S.A. Thompson, editors, *Discourse Description. Diverse Linguistic Analyses of a Fund-Raising Text*, pages 295–325. John Benjamins, Amsterdam.

Altaf Rahman and Vincent Ng. 2011. Learning the information status of noun phrases in spoken dialogues. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing,* Edinburgh, Scotland, U.K., 27–29 July 2011, pages 1069–1080.

Altaf Rahman and Vincent Ng. 2012. Learning the fine-grained information status of discourse entities. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics,* Avignon, France, 23–27 April 2012, pages 798–807.

Arndt Riester, David Lorenz, and Nina Seemann. 2010. A recursive annotation scheme for referential information status. In *Proceedings of the 7th International Conference on Language Resources and Evaluation,* La Valetta, Malta, 17–23 May 2010, pages 717–722.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phi Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of the 4th International Conference on Learning representations (ICLR 2016),* Caribe Hilton, San Juan, Puerto Rico, 2–4 May 2016.

Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, Daniel M. Ogilvie, and Cambridge Computer Associates. 1966. *General Inquirer: A Computer Approach to Content Analysis*. MIT Press, Cambridge, Mass.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 3104–3112. Curran Associates, Inc.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 2773–2781. Curran Associates, Inc.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2011. OntoNotes release 4.0. LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32th International Conference on Machine Learning,* Lille, France, 6–11 July 2015, pages 2048–2057.

Mo Yu and Mark Dredze. 2015. Learning composition models for phrase embeddings. *TACL*, 3:227–242.

Detection, Disambiguation and Argument Identification of Discourse Connectives in Chinese Discourse Parsing

Yong-Siang Shih and Hsin-Hsi Chen
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
{r02922036,hhchen}@ntu.edu.tw

Abstract

In this paper, we investigate four important issues together for explicit discourse relation labelling in Chinese texts: (1) discourse connective extraction, (2) linking ambiguity resolution, (3) relation type disambiguation, and (4) argument boundary identification. In a pipelined Chinese discourse parser, we identify potential connective candidates by string matching, eliminate non-discourse usages from them with a binary classifier, resolve linking ambiguities among connective components by ranking, disambiguate relation types by a multiway classifier, and determine the argument boundaries by conditional random fields. The experiments on Chinese Discourse Treebank show that the F1 scores of 0.7506, 0.7693, 0.7458, and 0.3134 are achieved for discourse usage disambiguation, linking disambiguation, relation type disambiguation, and argument boundary identification, respectively, in a pipelined Chinese discourse parser.

1 Introduction

Discourse relations represent how discourse units logically connect with each other. A discourse connective explicitly signals the presence of a discourse relation, and therefore it is an important clue for discourse analysis. There are several challenges in Chinese discourse parsing.

Firstly, there are more discourse connectives in Chinese than in English and their parts of speech have more varieties (Huang et al., 2014). Therefore, it is likely to encounter words that have the same surface forms as real connectives but do not function as discourse connectives.

Secondly, many Chinese connectives are parallel connectives that have multiple discontinuous components (Zhou and Xue, 2012). Each connective can be composed of one or more *connective components*. For example, "雖然-但" (although-but) consists of two connective components: "雖然" (although) and "但" (but). When multiple connectives are present in a paragraph, their components often link with each other in multiple possible ways. (S1) is an example. There are five possible connective candidates, including (1) "除了...還" (in addition to ... also), (2) "還...也" (also ... also), (3) "除了" (in addition to), (4) "還" (also), and (5) "也" (also). Figure 1 illustrates the ambiguous linking. Only candidates (1) and (5) are correct. Moreover, when spurious component candidates exist in a paragraph, they form many more spurious connective candidates by linking together in different ways. Finding the correct linking between correct components is useful for discourse analysis because they provide clues to determine the positions of the relations.

(S1) 除了投資環境優越，還在於這些企業所具有的產品優勢。(...) 對上海的產業優化也有很大的帶動作用。(In addition to superior investment environment, it's also due to the product advantages possessed by the enterprise. (...) It also has great effect in promoting the optimization of industries in Shanghai.)

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1891–1902, Osaka, Japan, December 11-17 2016.

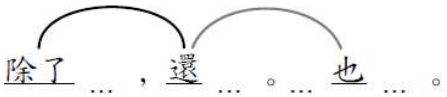


Figure 1: Ambiguous linking between connective components.

Thirdly, the sentence structures in Chinese texts are not clearly defined. Thus it is more challenging to detect the arguments for a given relation. Here, arguments of a discourse relation are the discourse units it involves. Since the relations form a hierarchical discourse structure, the arguments for relations higher in the hierarchy or lower in the hierarchy could span over ranges of various lengths.

In this paper, we aim at investigating these unique challenges at the same time. The goal of this research is to build an end-to-end system to analyse the explicit discourse relations in Chinese texts. In particular, we deal with four tasks together: (1) extraction of explicit discourse connectives, (2) linking resolution between the component candidates, (3) classification of the relation type for each discourse connective, and (4) extraction of the discourse arguments of a connective.

This paper is organized as follows. Section 2 surveys the related work. Section 3 describes the datasets used in this study. Section 4 presents our Chinese discourse parser. Section 5 shows and discusses the experimental results. Section 6 concludes the remarks.

2 Related Work

Rhetorical Structure Theory Discourse Treebank (RST-DT) (Carlson et al., 2001) and the Penn Discourse Treebank 2.0 (PDTB2) (Prasad et al., 2008) are two popular English discourse corpora for discourse analysis. Many groups have investigated different subtasks of English discourse parsing on PDTB2, including discourse connective identification, relation type disambiguation (Pitler and Nenkova, 2009; Wellner, 2009; Faiz and Mercer, 2013), and argument extraction (Wellner and Pustejovsky, 2007; Elwell and Baldridge, 2008; Ghosh et al., 2011; Ghosh et al., 2012; Kong et al., 2014). Lin et al. (2014) build an end-to-end discourse parser. As RST-DT provides hierarchical discourse structure annotations, there are also many attempts to construct the discourse structures automatically for sentences (Sporleder and Lapata, 2005; Fisher and Roark, 2007; Joty et al., 2012) and documents (Hernault et al., 2010; Feng and Hirst, 2012; Joty et al., 2013; Li, Li et al., 2014; Ji and Eisenstein, 2014).

Comparatively, there have been few large-scale Chinese discourse corpora until recently (Zhou and Xue, 2012; Zhou and Xue, 2015; Zhang et al., 2014; Li, Feng et al., 2014). Due to limited resource, early studies often used self-constructed corpora that made it difficult to compare between different works. T'sou et al. (1999), T'sou et al. (2000) and Chan et al. (2000) investigated connective detection in Chinese texts as a part of a tagging system. Hu et al. (2009) developed an automatic system to extract connective components from sentences. They used a rule-based method and found that the performance was sensitive to the connective lexicon. They improved their performance by removing words commonly used in non-discourse contexts. Zhou et al. (2012) and Li, Carpuat et al. (2014) employed cross-lingual information to deal with discourse usage ambiguity. Li, Carpuat et al. (2014) used 5-way classification to classify a connective between four relation types and non-discourse usage. Li et al. (2015) used CDTB to investigate detection and classification of connective components. They used maximum entropy and decision tree algorithms with various syntactic features. Chen et al. (2016) investigated fine-grained Chinese discourse relation labelling. Hu et al. (2011) dealt with linking ambiguity. However, they only focused on intra-sentential relations in sentences that have multiple clauses and assumed all connective components in the sentence have already been correctly identified.

As researchers start to focus on higher level problems for linguistic analysis, interest in discourse parsing also grows. The CoNLL-2015 Shared Task (Xue et al., 2015) and the CoNLL-2016 Shared Task (Xue et al., 2016) both focus on shallow discourse parsing. In particular, the CoNLL-2016 Shared Task features Chinese discourse parsing with Chinese Discourse Treebank 0.5 (Zhou and Xue, 2015), a PDTB-style annotated corpus.

3 Datasets

In this section, we briefly introduce the datasets used in this study and provide some statistics.

3.1 Chinese Discourse Treebank (CDTB)

In Chinese Discourse Treebank (CDTB) (Li, Feng et al., 2014), 500 documents selected from the Chinese Treebank (CTB) (Xue et al., 2005) were annotated. Totally, CDTB contains 2,342 paragraphs. Each paragraph was segmented into elementary discourse units (EDUs), and each paragraph is represented as a discourse tree as shown in Figure 2. Each relation is represented by an internal node, while each EDU is represented by a leaf node. The explicit and implicit relations between different spans of EDUs were annotated. In addition, discourse connectives for each relation were annotated. The exact positions of the arguments for a relation is heavily influenced by the complete discourse tree, and can range over multiple sentences in a paragraph.

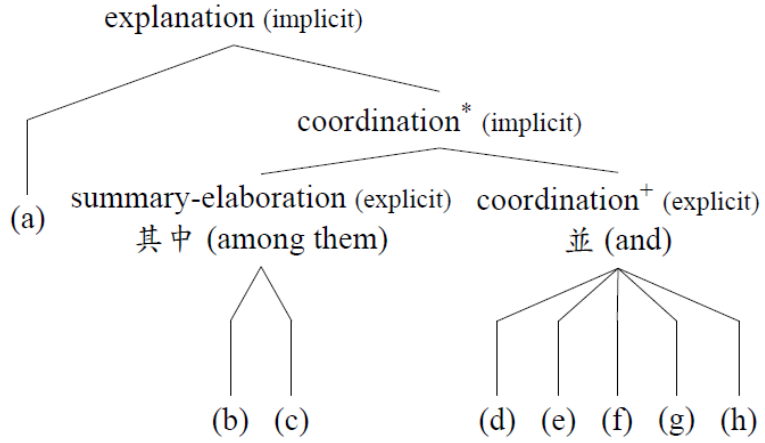


Figure 2: A discourse tree.

In CDTB, total 7,310 relations are annotated, and 1,814 of them are explicit. The set of discourse relation types is organized in a three-level hierarchy. In this paper, we only focus on the four top-level relation types, i.e., causality, coordination, transition, and explanation.

Some errors including duplicate annotations and erroneous positions are found in the corpus. After manual correction for explicit relation annotations, there are 1,813 explicit relations, each of which consists of exactly one connective instance. These 1,813 connectives are composed of 2,131 connective component instances.

The length distribution for the annotated connectives is shown in Table 1. The distribution is imbalance. There are totally 274 classes of connectives, but 147 of them appear only once.[1] Since some connectives share the same components, there are only 227 classes of connective components. Most of the connective classes only have one unique top-level relation type.

#Components	1	2	3	4	6	7
#Connective Classes	143	108	15	6	1	1
#Instances	1,544	235	24	8	1	1

Table 1: Lengths of connectives.

Table 2 shows the number of arguments for explicit relations. Most relations only have two arguments, but there are relations that have as many as 7 arguments. The number of arguments does not always match with the number of connective components. For example, single connective "并" (and) can have as many as 5 arguments because it connects multiple parallel segments together.

#Arguments	2	3	4	5	6	7
#Instances	1,688	85	33	4	2	1

Table 2: Number of arguments for each explicit relation.

Totally there are 3,802 arguments for explicit relations. Depending on the position the relation resides in the discourse tree, the length of an argument could vary greatly, but most of the arguments for

[1] These numbers are computed by their surface forms, i.e., instances of the same connective class may have different relation types.

explicit relation are composed of only one EDU. On average, each argument is composed of 1.6 EDUs, and the longest argument is composed of 20 EDUs.

3.2 NTU PN-Gram Corpus

Recently, efficient methods to learn word embeddings have been developed. In this paper, we investigate whether such word vectors are useful for dealing with discourse issues. NTU PN-Gram Corpus released by Yu et al., (2012), which was constructed by POS-tagging the Chinese texts extracted from the ClueWeb09 dataset (Callan et al., 2009). It has 21,217,147 unique sentences, containing 326,996,602 tokens. We used this corpus to create 400-dimensional embeddings using GloVe tool (Pennington et al., 2014) and word2vec tool (Mikolov et al., 2013a; Mikolov et al., 2013b) with skip-gram and continuous bag-of-words models.

3.3 Connective Component Dictionary

Discourse connectives serve as linking elements that connect discourse units. There are three kinds of linking directions (Li and Thompson, 1989): (1) forward-linking, (2) backward-linking, and (3) couple-linking. Such linking directions could be useful for identifying the positions of arguments for a given connective. We used the connective linking dictionary collected by Huang et al. (2014) as features for argument boundary identification. Totally, it contains 301 distinct connective components that are annotated with linking directions.

4 Chinese Discourse Parser

There are five modules in the proposed pipelined system. Each paragraph in CDTB is processed by the following modules: (1) identify connective candidates, (2) eliminate non-discourse usages from connective candidates, (3) resolve linking ambiguities, (4) disambiguate relation types, and (5) extract arguments.

4.1 Connective Candidate Extraction

We use string matching with the connective lexicon collected from CDTB to extract all possible instances. Directly matching with raw text would yield 12,498 candidate components[2] because many characters used for connectives appear in other unrelated words. Therefore, Stanford Chinese segmenter (Chang et al., 2008) is employed to segment paragraphs into tokens. Only the components composed of complete tokens are extracted.

Total 7,649 component candidates which recover 2,068 of 2,131 annotated components are extracted. These candidates form 7,976 connective candidates which recover 1,755 of 1,813 annotated connectives. While some correct instances are not extracted due to segmentation errors, it reduces the number of spurious candidates substantially while maintaining high recall.

4.2 Discourse Usage Disambiguation

A logistic regression classifier is trained to eliminate spurious connective candidates. The features are listed below:

P&N: We used a subset of the features selected from Pitler and Nenkova (2009). It includes four binary features for each connective component: (1) the highest category that dominates exactly the component itself, which is called self-category, (2) the parent, (3) the left-sibling, and (4) the right-sibling of the self-category. Null features are set when no such nodes exist. For example, in Figure 3, there is no node that dominates exactly the component "卻是" (but). For multi-component connectives, the union of the features is used. We have also experimented with the full feature set, but the performance does not increase.

CONNECTIVE: The string of the connective.

[2] Candidate components that could not form complete connectives are already eliminated. If we simply match with component lexicon without checking whether they could form connectives, 24,539 candidate components would be detected.

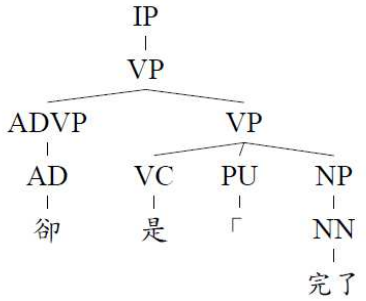


Figure 3: A sub-parsing tree for 卻是.

POS: The feature set contains: (1) POS tags for all tokens that constitute the connective component, (2) POS tag of the token to the left of the component, and (3) POS tag of the token to the right of the component. For multi-component connectives, the union of the features is used.

NUM: The feature set contains a one-hot encoded feature–the number of components. In addition, there are seven numerical features: (1) the number of overlapped connective candidates, (2) the number of connective candidates that enclose any components of the current connective, (3) the distance between the leftmost and the rightmost tokens of the connective measured by tokens, (4) the geometric mean of distances between all neighbouring connective components for the current connective candidate, (5) the distances from the leftmost component to a separating element including "!?:;, ◦ " or the paragraph boundary on the left, (6) the distance from the rightmost component to a separating element on the right, and (7) the minimum distance from any separating element to any connective component. We normalize the numerical features by scaling each to zero mean and unit variance.

VECTOR: This feature set is built using word embeddings. The vectors are used to construct three features for each connective component: (1) the mean of the vectors representing each token that constitutes the connective component, (2) the vector for the token to the left, and (3) the vector for the token to the right. Zero-valued vector is used when the vector does not exist. In total, it is a 1,200-dimensional vector for each component when the 400-dimensional embeddings are used. For multi-component connectives, we averaged the vectors.

4.3 Connective Linking Disambiguation

If we can classify discourse usage perfectly, we would have already solved the linking ambiguities because only the correct connectives remain. Due to imperfect classification, there may still exist some overlapped candidates. Here, we propose a greedy algorithm to resolve linking ambiguities among a set of connective candidates as outlined in Algorithm 1. The algorithm filters the candidate set C and produces a result set A that contains only non-overlapped connective candidates. All connective candidates are ranked under some criteria and the one with the highest priority is greedily accepted. We will use different ranking criteria to evaluate our models, including (1) Score: the probability obtained by logistic regression as described in Section 4.2, (2) Length: the number of components each connective candidate has, the larger the better, and (3) Position. Position is used mainly as a tiebreaker. In particular, we accept the left-most candidate first.

Algorithm 1. Linking Resolution Algorithm

Input C: A set of connective candidates
Output A: A set of accepted connectives

1. $A \leftarrow \{\}$
2. Rank all connective candidates in C
3. **while** C is not empty **do**
4. let c_i be the connective candidate that has the highest priority
5. $C \leftarrow C - \{c_i\}$
6. $A \leftarrow A \cup \{c_i\}$
7. Remove all connective candidates $c_j \in C$ that overlap with c_i
8. **end while**
9. **return** A

4.4 Discourse Relation Type Disambiguation

We use a logistic regression classifier to investigate whether the features we discussed in Section 4.2 are useful for relation type disambiguation.

4.5 Connective Argument Extraction

We formulate argument extraction as a sequence labelling problem. As we know the arguments span over a continuous interval, we use four labels for the EDUs: *before*, *start*, *inside*, and *after*. Each argument is represented by a *start* followed by zero or more *inside*s. Figure 2 is a discourse tree, where the explanation relation has two arguments (a) and (b-h), while the coordination$^+$ relation has five arguments (d), (e), (f), (g), and (h). Figure 4 shows the arguments and the corresponding labels for summary elaboration relation shown in Figure 2.

As our goal is to extract the arguments, only the argument boundaries must be determined. Although correct EDU segmentation is unavailable to our system because we attempt to extract arguments from raw texts, the EDU boundaries in CDTB only occur with certain punctuation symbols that separate phrases and sentences. Thus, we segment a paragraph by these symbols, and solve the sequence labelling problem on these segments instead of EDUs.

We use Conditional Random Fields (CRFs) to deal with the sequence labelling problem. CRFsuite (Okazaki, 2007) is adopted along with its default parameters. When training, each explicit relation with its corresponding labelling is used as a training case. When testing, CRFs are used to label the segments for each connective we extracted. Therefore, the same segments for a paragraph are labelled independently for each explicit relation inside the paragraph. The resulting argument boundaries are used to extract the connective's arguments.

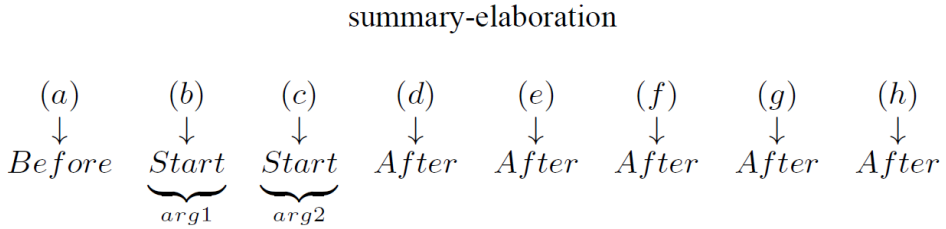


Figure 4: Sequence labelling for relation arguments identification.

The features, which are determined by the current connective being considered and the segment, are shown as follows.

CONTEXT: The concatenation of the self-category and the categories of the parent, the left-sibling, and the right-sibling is used as a binary feature as done by Kong et al. (2014).

PATH: The feature set is similar to the CON-NT-Path features of Kong et al. (2014). We use the path from the self-category of each connective component to the self-category of the segment as the feature.

POS: The POS tags for all tokens in the segment.

SUBJ: The SUBJ feature is set if a segment contains a subject.

ENDCHAR: The last token in the segment, i.e., the symbol that separates the current segment from the next one.

COMPONENT: The feature set contains the information about connective components: (1) whether the segment has a connective component, (2) the string of the component if it exists, (3) whether there exists a component at the beginning of the segment, (4) whether there exists a component at the end of the segment, (5) whether the segment contains only a component and the separating symbol, (6) whether the segment is before all connective components, (7) the distance to the first segment that contains a component as a binary feature, (8) whether the segment is after all connective components, and (9) the distance to the last segment that contains a connective component as a binary feature.

LINK: The feature set contains the linking directions a connective component could be used if it exists in the given segment. A connective component dictionary is consulted.

CONNECTIVE: This feature set contains connective related information, including the string of the connective and the number of connective components it has.

5 Results and Discussions

In the experimental setups, we first evaluate the performance of each individual disambiguation task and then examine the propagation effects in the pipelined system.

5.1 Discourse Usage Disambiguation

We evaluate our models using 10-fold cross-validation. The 2,342 paragraphs are divided into 10 splits while keeping the distribution for the number of explicit relations in each paragraph roughly equal. The averaged precision, recall, and F1 scores for the positive connective instances are reported. As there are many spurious connectives, we balance the training set by oversampling the correct instances for three times, but keep the original distribution when evaluating on test set.

For statistical significance, we use Wilcoxon signed-ranks test (Wilcoxon, 1945) as suggested by Demšar (2006) at confidence level 95%. For each experiment, we select the best model (denoted by bold), and * is used to denote the scores that are significantly different.

We firstly investigate the performance between different word embeddings as shown in Table 3. The vectors constructed by skip-gram model are the most useful. We will use them in the remaining experiments. Table 4 shows the results for all features. The best results are obtained with ALL-SKIP-GRAPM in discourse usage disambiguation. We also experiment with different learning models including Naive Bayes, SVM, decision trees, and random forest. Logistic Regression performs the best. For all models, we use default parameters provided by scikit-learn without tuning, i.e., C=1.0, penalty=l2 for Logistic Regression.

Features	Precision	Recall	F1 Score
CBOW	0.5808	0.7625	0.6593*
SKIP-GRAM	0.6013	0.8068	**0.6887**
GLOVE	0.5980	0.7996	0.6840
ALL	0.5837	0.7439	0.6539*

Table 3: Performance of discourse usage disambiguation using different word embeddings.

Features	Precision	Recall	F1 Score
P&N	0.4239	0.8409	0.5634*
CONNECTIVE	0.5205	0.8620	0.6487*
POS	0.5426	0.7805	0.6399*
NUM	0.4298	0.8456	0.5696*
SKIP-GRAM	0.6013	0.8068	0.6887*
ALL-P&N	0.6547	0.8186	0.7273*
ALL-POS	0.6576	0.8222	0.7305*
ALL-NUM	0.6357	0.8160	0.7144*
ALL-SKIP-GRAM	0.6503	0.8882	**0.7506**
ALL	0.6682	0.8203	0.7363*

Table 4: Performance of discourse usage disambiguation using different features.

5.2 Connective Linking Disambiguation

To evaluate linking disambiguation individually, we first assume all correct connective components are already known. We use the 10-fold for paragraphs specified in Section 5.1 to evaluate our model using each connective as an instance. The results are reported in Table 5. We evaluate different ranking criteria and the combination. A baseline model that simply ranks the candidates by their positions is also reported. We find that the ambiguity among the components is low. The baseline model already achieves an F1 score of 0.8797. In fact, only 472 out of 2,131 components are involved in more than one connective candidate. Length is relatively weaker than Score reported by the logistic regression model. Moreover, we also evaluate linking disambiguation within the pipelined system. The results are shown in Table 6. Integrating both Score and Length criteria performs the best. The comparison shows that most of the linking ambiguity is caused by spurious linking with spurious connective component candidates.

Ranking Criteria	Precision	Recall	F1 Score
Baseline	0.8528	0.9084	0.8797*
Score	0.9770	0.9796	**0.9783**
Length	0.9760	0.9604	0.9681*
Length+Score	0.9793	0.9636	0.9714*

Table 5: Performance of linking disambiguation with known connective components.

Ranking Criteria	Precision	Recall	F1 Score
Baseline	0.6696	0.7919	0.7254*
Score	0.7024	0.8222	0.7573*
Length	0.7099	0.8238	0.7624*
Length+Score	0.7165	0.8310	**0.7693**

Table 6: Performance of linking disambiguation in the pipelined system.

Methods	Precision	Recall	F1 Score
w/o Linking resolution	0.7399	0.8680	0.7985
Length+Score	0.7493	0.8585	**0.7999**
Li et al. (2015) ME	0.7880	0.6180	0.6920
Li et al. (2015) DT	0.5680	0.4960	0.5230

Table 7: Performance of discourse connective disambiguation on the component level.

In the above evaluation, we consider a connective instance as an evaluation unit. To compare with the related work we also take a connective component as an evaluation unit. Table 7 summarizes the results of discourse connective disambiguation on this level. The first model is for discourse usage disambiguation without linking disambiguation. The second model eliminates additional candidates by resolving linking ambiguity. The experimental results of Li et al. (2015) are listed for reference because they use the same dataset as us. The best results for Maximum Entropy (ME) and Decision Tree (DT) classifiers with automatic parsing tree features are selected. Although linking disambiguation has small effect for identifying individual components, it effectively improves the performance of connective extraction as shown in Table 6. The result is also important for argument extraction, because the positions of connective components provide clues for the positions of the arguments of an explicit relation.

5.3 Discourse Relation Type Disambiguation

At first, we evaluate the relation type disambiguation by assuming connectives are known. We use 10-fold cross-validation with the 1,813 explicit connectives to evaluate our model. We keep the distribution for the relation types roughly equal for each fold. While the NUM features have some discriminative power for discourse usage disambiguation, it does not help for relation disambiguation. When used independently, the performance is the same as always predicting the major category. On the other hand, the string of the connective provides strong clues for the relation type. When used individually, it already achieves a micro average F1 of 0.9308. In addition, the SKIP-GRAM feature is also useful for this task, achieving a micro average F1 of 0.9473. In Table 8, we show the performance for different relation types using ALL-NUM as features. We can find that the number of instances affects the performance of the learning model. The lesser the instances, the worse the performance. To compare with Li et al. (2015), we also evaluate the results on component level. Table 9 shows that we also achieve better performance on relation type classification.

Relation Type	Precision	Recall	F1 Score	#instances
causality	0.9634	0.9504	0.9561	465
coordination	0.9575	0.9723	0.9645	974
transition	0.9372	0.9131	0.9234	173
explanation	0.9754	0.9450	0.9588	201
macro average	0.9584	0.9452	0.9507	1,813

Table 8: Performance of relation type disambiguation when connectives are known.

	Our Model			Li et al. (2015)		
Relation Type	P	R	F1	P	R	F1
causality	0.9584	0.9420	0.9490	0.8380	0.6840	0.7510
coordination	0.9566	0.9734	0.9645	0.8250	0.9360	0.8770
transition	0.9504	0.9178	0.9318	0.7850	0.5960	0.6700
explanation	0.9754	0.9407	0.9563	0.8970	0.8280	0.8590

Table 9: Performance of relation type disambiguation on component level.

We further evaluate the relation type classification on the pipelined system. We predict the relation type with features ALL-NUM for each connective candidate extracted by using the Length+Score approach. The 10-fold for paragraphs specified in Section 5.1 is used. We obtain micro-averaged F1 score of 0.7458. Compared with the F1 score of 0.7693 shown in Table 6, the performance only decreases a little when one more module is integrated into the pipelined system. That shows the effectiveness of our model for relation type disambiguation.

5.4 Connective Argument Extraction

To evaluate argument extraction individually, we first assume all connectives are already correctly identified. The 10-fold for paragraphs specified in Section 5.1 is used for cross-validation. The precision, recall, and F1 scores for the argument boundaries are computed. In addition, accuracy scores evaluated on 1,813 connective instances are computed. Each instance is counted as correct only when all boundaries are all correctly identified. The averaged results over all folds are reported in Table 10. While the best F1 for argument boundaries is 0.7848, the accuracy for the connectives as evaluation units is only 0.4074. It means that for each connective, we are able to recover most of its argument boundaries, but it is challenging to recover all of them at the same time.

An analysis on the errors reveals that our model can handle the relations that have exact two arguments. For more arguments, the error rates are almost close to 1. In addition, out of the 1,074 error cases, there are only 164 cases that both sides of the interval the arguments span over are incorrect. The reason behind this is probably due to the fact that the existence of a connective often gives strong hint on at least one side of the interval. On the contrary, there is often no explicit indication on the boundary of the other side of the interval.

Finally, we evaluate the performance of the pipelined Chinese discourse parser. Here, the Length+Score approach is used for connective extraction, the ALL-NUM features are used to disambiguate the 4 top-level relation types, and the CRF models with ALL features are used for argument boundary detection. Each explicit relation is counted as true positive only when the three tasks are all correctly done; otherwise, it is counted as false positive. Under the rigorous evaluation, precision, recall, and F1 score are 0.2917, 0.3389 and 0.3134, respectively. We also evaluate on a relaxed partial match for argument extraction. An F1 score is computed for each argument tokenwise, and an instance is treated as correct when the number of arguments is correct and the averaged tokenwise F1 score is above a threshold.[3] The F1 scores computed with 0.3, 0.5, 0.7, and 0.9 as thresholds are 0.6013, 0.5782, 0.5063 and 0.3455, respectively.

Features	Precision	Recall	F1 Score	Accuracy
CONTEXT	0.5225	0.3030	0.3835*	0.0189
PATH	0.7660	0.5645	0.6497*	0.1471
POS	0.5576	0.3856	0.4556*	0.0734
SUBJ	0.8800	0.0788	0.1440*	0.0000
ENDCHAR	0.4606	0.2974	0.3614*	0.0000
LINK	0.7690	0.4001	0.5261*	0.0183
CONNECTIVE	0.4695	0.3046	0.3695*	0.0049
COMPONENT	0.6884	0.6698	0.6789*	0.2190
ALL	0.8024	0.7680	**0.7848**	**0.4074**

Table 10: Performance of argument boundary detection using different features.

[3] The tokenwise averaged F1 is adopted from partial scoring for CoNLL 2016 Shared Task: http://conll16st.blogspot.tw/2016/04/partial-scoring-and-other-evaluation.html.

6 Conclusion

In this paper, we investigate four issues regarding Chinese discourse analysis at the same time. We propose four types of features for discourse usage disambiguation. A greedy algorithm is also developed to resolve linking ambiguities. Besides, we also investigate relation type disambiguation and argument extraction. These modules are integrated into a pipelined system that extracts explicit discourse relations and their arguments from the raw text. The pipelined system achieves an overall F1 score of 0.3134.

There still exist some issues that need to be further investigated. Firstly, a closer integration between discourse usage disambiguation and linking disambiguation may be valuable. In our work, these two stages are pipelined. Although some linking information is used as features, the greedy algorithm still ranks each candidate individually. We expect that utilizing global relationship between conflicting candidates may improve the performance for both tasks. Secondly, the arguments for a relation may be useful for relation type recognition. However, the accuracy for argument extraction must be improved before the extracted arguments are used as features. Finally, implicit relations must be dealt with to construct the full discourse structure. Resolving these issues will be helpful to construct a complete Chinese discourse parser.

7 Acknowledgments

This research was partially supported by Ministry of Science and Technology, Taiwan, under grants MOST-102-2221-E-002-103-MY3 and MOST-105-2221-E-002-154-MY3, and National Taiwan University under grant NTU-ERP-104R890858. We are also very thankful to Professor Zhou Guodong for providing us Chinese Discourse TreeBank, and the three anonymous reviewers for their helpful comments to revise this paper.

References

Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. 2009. Clueweb09 data set.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a Discourse-Tagged Corpus in the Framework of Rhetorical Structure Theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*, pages 1--10, Stroudsburg, PA, USA.

Samuel W. K. Chan, Tom B. Y. Lai, Weijun Gao, and Benjamin K. T'sou. 2000. Mining Discourse Markers for Chinese Textual Summarization. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic Summarization*, pages 11--20, Stroudsburg, PA, USA.

Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. 2008. Optimizing Chinese Word Segmentation for Machine Translation Performance. In P*roceedings of the Third Workshop on Statistical Machine Translation*, pages 224--232, Stroudsburg, PA, USA.

Huan-Yuan Chen, Wan-Shan Liao, Hen-Hsen Huang and Hsin-Hsi Chen. 2016. Fine-Grained Chinese Discourse Relation Labelling. In *Proceedings of 10th Language Resources and Evaluation Conference*, pages 1034--1038, Portorož, Slovenia.

Janez Demšar. 2006. Statistical Comparisons of Classifiers Over Multiple Data Sets. *The Journal of Machine Learning Research*, 7:1--30.

Robert Elwell and Jason Baldridge. 2008. Discourse Connective Argument Identification with Connective Specific Rankers. In *Proceedings of 2008 IEEE International Conference on Semantic Computing*, pages 198--205.

Syeed Ibn Faiz and Robert E. Mercer. 2013. Identifying Explicit Discourse Connectives in Text. *Lecture Notes in Computer Science*, volume 7884, pages 64--76. Springer Berlin Heidelberg.

Vanessa Wei Feng and Graeme Hirst. 2012. Text Level Discourse Parsing with Rich Linguistic Features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 60--68.

Seeger Fisher and Brian Roark. 2007. The Utility of Parse-derived Features for Automatic Discourse Segmentation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 488--495, Prague, Czech Republic.

Sucheta Ghosh, Richard Johansson, Giuseppe Riccardi, and Sara Tonelli. 2011. Shallow Discourse Parsing with Conditional Random Fields. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1071—1079, Chiang Mai, Thailand.

Sucheta Ghosh, Giuseppe Riccardi, and Richard Johansson. 2012. Global Features for Shallow Discourse Parsing. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '12, pages 150--159, Stroudsburg, PA, USA.

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

Jin-zhu Hu, Jiang-bo Shu, Shuang-yun Yao, Xing Zhou, Feng-wen Wu, and Sheng Xiao. 2009. Research on the Extraction of Relation Markers in Compound Sentences Oriented to Chinese Information Processing. *Computer Engineering and Science*, 31(10):90--93.

Jin-zhu Hu, Li-li Lei, Jin-cai Yang, Jiang-bo Shu, and Chen Jiang-man. 2011. Research on a Solving Model of the Collocations between the Relation Markers in Multiple Compound Sentences. *Computer Engineering and Science*, 33(11):177--182.

Hen-Hsen Huang, Tai-Wei Chang, Huan-Yuan Chen, and Hsin-Hsi Chen. 2014. Interpretation of Chinese Discourse Connectives for Explicit Discourse Relation Recognition. In *Proceedings of 25th International Conference on Computational Linguistics*, pages 632—643, Dublin, Ireland.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation Learning for Text-level Discourse Parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Shafiq Joty, Giuseppe Carenini, and Raymond T. Ng. 2012. A Novel Discriminative Framework for Sentence-level Discourse Analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 904--915, Stroudsburg, PA.

Shafiq R Joty, Giuseppe Carenini, Raymond T Ng, and Yashar Mehdad. 2013. Combining Intra-and Multi- Sentential Rhetorical Parsing for Document-level Discourse Analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 486--496.

Fang Kong, Hwee Tou Ng, and Guodong Zhou. 2014. A Constituent-based Approach to Argument Labelling with Joint Inference in Discourse Parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 68--77, Doha, Qatar.

Charles N Li and Sandra A Thompson. 1989. Mandarin Chinese: A Functional Reference Grammar. University of California Press.

Yancui Li, Jing Sun, and Guodong Zhou. 2015. Automatic Recognition and Classification on Chinese Discourse Connective. Acta Scientiarum Naturalium Universitatis Pekinensis, 2:016.

Jiwei Li, Rumeng Li, and Eduard Hovy. 2014. Recursive Deep Models for Discourse Parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 2061--2069, Doha, Qatar.

Junyi Jessy Li, Marine Carpuat, and Ani Nenkova. 2014. Cross-lingual Discourse Relation Analysis: A Corpus Study and A Semi-supervised Classification System. In *Proceedings of the 25th International Conference on Computational Linguistics*: Technical Papers, pages 577--587.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A PDTB-styled End-to-End Discourse Parser. *Natural Language Engineering*, 20:151--184.

Yancui Li, Wenhe Feng, Jing Sun, Fang Kong, and Guodong Zhou. 2014. Building Chinese Discourse Corpus with Connective-Driven Dependency Tree Structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 2105--2114.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in Neural Information Processing Systems* 26, pages 3111--3119.

Naoaki Okazaki. 2007. CRFsuite: a Fast Implementation of Conditional Random Fields (CRFs).

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532--1543.

Emily Pitler and Ani Nenkova. 2009. Using Syntax to Disambiguate Explicit Discourse Connectives in Text. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 13--16, Stroudsburg, PA, USA.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse Treebank 2.0. In *Proceedings of LREC*.

Caroline Sporleder and Mirella Lapata. 2005. Discourse Chunking and Its application to Sentence Compression. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 257--264, Stroudsburg, PA, USA.

Benjamin K. T'sou, Weijun Gao, Tom B. Y. Lai, and Samuel W. K. Chan. 1999. Applying Machine Learning to identify Chinese Discourse Markers. In *Proceedings of International Conference on Information Intelligence and Systems*, pages 548--553.

Benjamin K. T'sou, Tom B. Y. Lai, Samuel W. K. Chan, Weijun Gao, and Xuegang Zhan. 2000. Enhancement of a Chinese Discourse Marker Tagger with C4.5. In *Proceedings of the Second Workshop on Chinese Language Processing*, pages 38--45, Stroudsburg, PA, USA.

Ben Wellner and James Pustejovsky. 2007. Automatically Identifying the Arguments of Discourse Connectives. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 92--101, Prague, Czech Republic.

Ben Wellner. 2009. Sequence Models and Ranking Methods for Discourse Parsing. Ph.D. thesis, Waltham, MA, USA..

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. Biometrics bulletin, pages 80--83.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi Prasad, Christopher Bryant, and Attapol Rutherford. 2015. The CoNLL-2015 Shared Task on Shallow Discourse Parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning: Shared Task*, Beijing, China.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Bonnie Webber, Attapol Rutherford, Chuan Wang, and Hongmin Wang. 2016. The CoNLL-2016 Shared Task on Multilingual Shallow Discourse Parsing. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task*, Berlin, Germany, August. Association for Computational Linguistics.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The Penn Chinese Treebank: Phrase Structure Annotation of a Large Corpus. Nat. Lang. Eng., 11(2):207--238.

Chi-Hsin Yu, Yi-jie Tang, and Hsin-Hsi Chen. 2012. Development of a Web-scale Chinese Word N-gram Corpus with Parts of Speech Information. In *Proceedings of LREC*, pages 320--324.

Muyu Zhang, Bing Qin, and Ting Liu. 2014. Chinese Discourse Relation Semantic Taxonomy and Annotation. *Journal of Chinese Information Processing*, 28(2):28.

Yuping Zhou and Nianwen Xue. 2012. PDTB-style Discourse Annotation of Chinese Text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 69--77, Stroudsburg, PA, USA.

Yuping Zhou and Nianwen Xue. 2015. The Chinese Discourse Treebank: a Chinese Corpus Annotated with Discourse Relations. *Language Resources and Evaluation*, 49(2):397--431.

Lanjun Zhou, Wei Gao, Binyang Li, Zhongyu Wei, and Kam-Fai Wong. 2012. Cross-lingual Identification of Ambiguous Discourse Connectives for Resource Poor Language. In Proceedings of COLING 2012: Posters, pages 1409--1418.

Multi-view and multi-task training of RST discourse parsers

Chloé Braud
CoAStaL
Dep. of Computer Science
University of Copenhagen
braud@di.ku.dk

Barbara Plank
Computational Linguistics
CLCG
University of Groningen
b.plank@rug.nl

Anders Søgaard
CoAStaL
Dep. of Computer Science
University of Copenhagen
soegaard@di.ku.dk

Abstract

We experiment with different ways of training LSTM networks to predict RST discourse trees. The main challenge for RST discourse parsing is the limited amounts of training data. We combat this by regularizing our models using task supervision from related tasks as well as alternative views on discourse structures. We show that a simple LSTM sequential discourse parser takes advantage of this multi-view and multi-task framework with 12-15% error reductions over our baseline (depending on the metric) and results that rival more complex state-of-the-art parsers.

1 Introduction

Documents are not just an arbitrary collection of text spans, but rather an ordered list of structures forming a discourse. Discourse structures describe the organization of documents in terms of discourse or rhetorical relations. For instance, the discourse relation CONDITION holds between the two discourse units (marked with square brackets) in example (1a) and a relation MANNER-MEANS holds between the segments in example (1b).[1]

(1) a. [The gain on the sale couldn't be estimated] [until the "tax treatment has been determined."]

b. [On Friday, Datuk Daim added spice to an otherwise unremarkable address on Malaysia's proposed budget for 1990] [by ordering the Kuala Lumpur Stock Exchange "to take appropriate action immediately" to cut its links with the Stock Exchange of Singapore.]

Different theories of discourse structure exist. For instance, Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) analyzes texts as constituency trees covering entire documents. This theory has led to the RST Discourse Treebank (RST-DT) (Carlson et al., 2001) for English and the development of text-level discourse parsers (Hernault et al., 2010; Joty et al., 2012; Feng and Hirst, 2014; Ji and Eisenstein, 2014b). Such parsers have proven to be useful for several downstream applications (Taboada and Mann, 2006; Daumé III and Marcu, 2009; Thione et al., 2004; Sporleder and Lapata, 2005; Louis et al., 2010; Bhatia et al., 2015; Burstein et al., 2003; Higgins et al., 2004). Another corpus has been annotated for discourse phenomena in English, the Penn Discourse Treebank (Prasad et al., 2008) (PDTB). In contrast to RST-DT, PDTB does not encode discourse as tree structures and documents are not fully covered. In this study we focus on the RST-DT, but among other things, we consider the question of whether the information in PDTB can be used to improve RST discourse parsers.

Discourse parsing is known to be a hard task (Stede, 2011). It involves several complex and interacting factors, touching upon all layers of linguistic analysis, from syntax, semantics up to pragmatics. Consequently, also annotation is complex and time consuming, and hence available annotated corpora are sparse and limited in size. The aim of this paper is to address this training data sparsity by proposing to leverage different views of the same data as well as information from related auxiliary tasks. We aim at investigating which source of information are relevant for the discourse parsing task.

[1] The examples are taken from the RST Discourse Treebank, documents 1179 and 0613, respectively.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1903–1913, Osaka, Japan, December 11-17 2016.

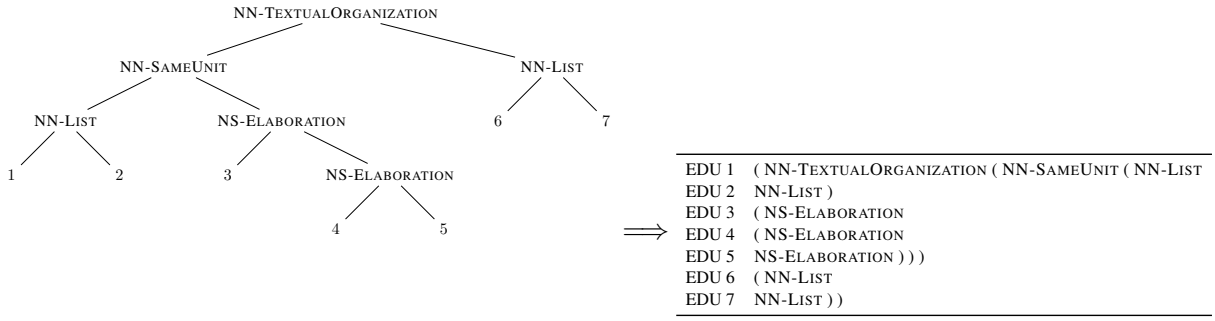


Figure 1: From RST-DT discourse trees to constituency sequence labels.

Specifically, we draw upon the recent success of deep learning methods and present a novel multi-view multi-task hierarchical deep learning model for discourse parsing. Our model, a bidirectional Long Short-Term Memory (bi-LSTM) model, learns joint text segment representations for predicting RST-DT discourse trees and learns from several auxiliary tasks. We encode RST-DT trees as sequences of bracket and label n-grams, and exploit multiple views of the data (such as RST-DT structures as dependencies) as well as multi-task learning through auxiliary tasks (such as modality information from TimeBank, or discourse relations as annotated in the PDTB). Our multi-view learning is different from the standard notion of multi-view learning. Jin et al. (2013), for example, combine multi-view and multi-task learning, but here, multiple views refer to multiple, independent feature sets describing the datapoints. We are, to the best of our knowledge, the first to use multiple views *on the output structures* to effectively regularize learning.

Contributions We present a hierarchical multi-task bi-LSTM architecture for multi-task learning, enabling better learning of discourse parsers with other views of the data and related tasks. Our approach achieves competitive performance compared to previous state-of-the-art models by making use of auxiliary tasks. We make the code and preprocessing scripts available for download at `http://bitbucket.org/chloebt/discourse`.

2 Baseline RST parser

Discourse parsing is a prediction problem where the input is a document, i.e., a sequence of elementary discourse units (EDUs) consisting of text fragments. The output of the task is a binary tree[2] with EDUs at the leaf nodes. The non-terminal nodes are labeled with two sets of information: (a) discourse relations and (b) an indication of whether the daughters are nucleus or satellite. A nucleus is being considered as the most important part of the text whereas a satellite presents secondary information. A discourse relation may involve a nucleus and a satellite (mononuclear relation) or two nuclei (multinuclear relation).

2.1 From sequences to trees

Our approach is to learn sequential models with transfer from models from related tasks, but the output structures are trees. For this purpose, we encode trees as sequences in a very simple way that preserves all the information from the original trees: Every EDU is labeled with its local surrounding discourse structure. More precisely, the first EDU is labeled by the entire path of the root of the tree to itself. Then, the following EDUs are labeled as beginning a new relation (using the opening bracket and the relation name) or ending one or more relations (using the relation name and closing brackets). For example, the EDU 1 in the tree in Figure 1 will be labeled with: "NS-TEXTUALORGANIZATION (NS-SAMEUNIT (NN-LIST".[3] Then, the EDU 2 ends a LIST relation, and the EDU 3 begins an ELABORATION relation. See Figure 1 for a complete conversion.

[2] As in all the previous studies on the RST-DT, we binarize the trees using right-branching.

[3] 'N' means nucleus and 'S' satellite, a relation is thus labeled 'NS' if its first argument is the nucleus of the relation and its second argument, the satellite.

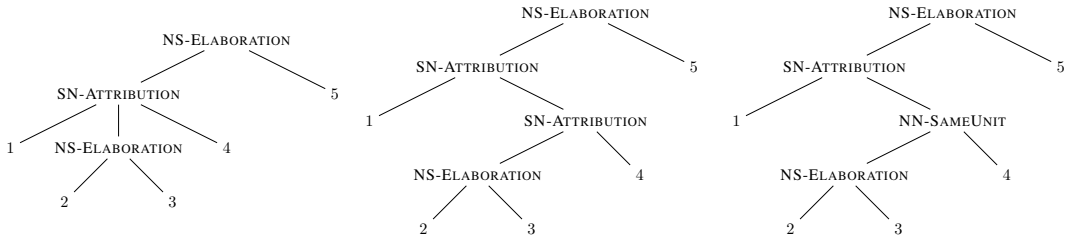


Figure 2: RST predicted, corrected and gold trees for document 1129 from the RST-DT.

Our output will be labeled trees, but below, in our multi-task learning models, we will consider unlabeled parsing and labeled parsing with only nuclearity or relations as auxiliary tasks. Note, however, that in a sequence prediction model we have no guarantee that our output structures form well-formed discourse trees. Therefore we use the following heuristics to guarantee well-formed output structures:

Heuristics The first three heuristics are enough to guarantee well-formed trees in practice: 1) If the first predicted label only contains closing parenthesis, we replace them by opening ones. 2) We remove any right hand side bracket that ends the tree too early, i.e., leads to a well-formed tree only covering a left subsequence of the sequence of EDUs. 3) We add right hand side brackets at the end if there are unclosed brackets after processing the sequence of labels.

However, we not only need to produce well-formed trees, we need to produce well-formed binary trees. Hence, we add the following two heuristics: 4) We first transform them to Chomsky Normal Form.[4] 5) We then remove unary nodes as follows:

- If the unary node is the root, an internal node whose child is a relation node, or a pre-terminal node (its child is a leaf and an EDU node), we replace it by its child.
- If the unary node is an internal node and its child an EDU node (but not a leaf node), then the EDU node becomes its left daughter and the daughter of the EDU node becomes its right daughter.

For example, the document 1129 in the RST-DT is predicted by our baseline model as the sequence in (2a), corrected first using the steps from 1) to 3). Here, we only need to remove a closing parenthesis after EDU 4. We obtain the sequence (2b) corresponding to the first tree in Figure 2. This tree only needs to be binarized, we thus end with the second tree in Figure 2 that can then be compared to the gold tree (third tree in Figure 2).

(2) a. (NS-ELABORATION (SN-ATTRIBUTION (1) (NS-ELABORATION (2)(3)) (4))) (5))

b. (NS-ELABORATION (SN-ATTRIBUTION (1) (NS-ELABORATION (2)(3)) (4)) (5))

3 Auxiliary tasks

We consider two types of auxiliary tasks: first, tasks derived from the RST-DT (multi-view), that is dependency encoding of the trees and additional auxiliary tasks derived from the main one; second, we consider tasks derived from additional data, namely, the Penn Discourse Treebank (Prasad et al., 2008), Timebank (Pustejovsky et al., 2003; Pustejovsky et al., 2005), Factbank (Saurí and Pustejovsky, 2009), Ontonotes (Hovy et al., 2006) and the Santa Barbara corpus of spoken American English (Du Bois, 2000).

All the auxiliary tasks are, as the main one, document-level sequence prediction tasks. In Table 1 we report the number of documents and single labels for each task. We hypothesize that such auxiliary information is useful to address data sparsity for RST discourse parsing.

[4] Using the implementation available in NLTK.

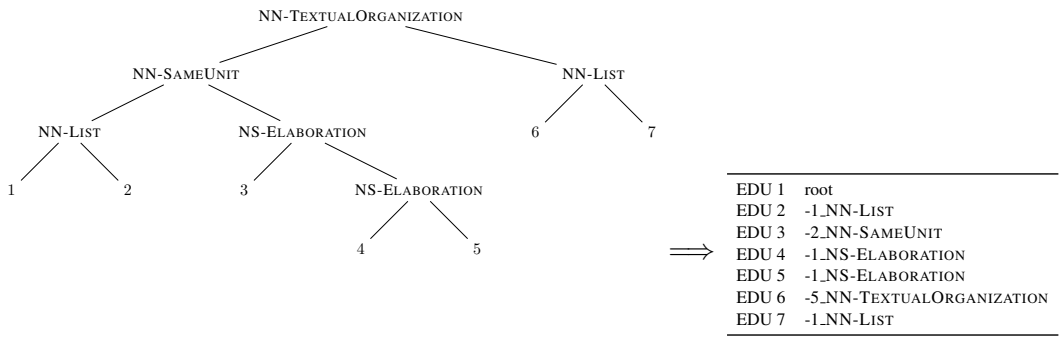


Figure 3: From RST-DT discourse trees to dependency sequence labels. The numbers indicate the position of the head of the EDU, e.g. EDU 2 and EDU 3 have the root EDU 1 as head.

3.1 Building other views of the RST-DT trees

Binary dependencies We first use a representation of the RST-DT trees as binary dependencies (RST-Dep). We roughly do the same transformation as (Muller et al., 2012; Li et al., 2014) but contrary to the latter, we choose as root the nucleus of the root node of the tree rather than the first EDU of the document. More precisely, we associate each node with its saliency set as defined in (Marcu, 1997): The nucleus is the salient EDU of a relation, and the nuclei can go up in the tree with possibly several nuclei in the saliency set of a node. Like Li et al. (2014), we replace all multi-nuclear relations (NN) by mono-nuclear ones choosing the left DU as the nucleus (NS). We thus have only one nucleus in each saliency set. Figure 3 illustrates the conversion of an RST tree into dependency sequence labels.

Nuclearity and relations We further add two alternative views that simply correspond to the main task with one label information removed, keeping either only nuclearity labels (Nuc) or discourse relations (Lab). The idea here is to break up the labeling task, since with the set of 18 discourse relations traditionally used, adding the nuclearity information leads to a large number of 41 labels.

Fine-grained labels Finally, we also use the main task with the original 78 fine-grained relations as an auxiliary task, the idea being of helping the model to learn finer distinctions between the relations.

3.2 Using additional annotations

As we already discussed, discourse relation identification is a hard task that requires access to high-level information. Previous work has shown that and indication about the *events* involved in the discourse units aids identification or constrains the set of inferable relations (Asher and Lascarides, 2003; Danlos and Rambow, 2011; Taboada and Das, 2013). Consider our examples given in the introduction. For instance, modals can indicate conditional relations as in example (1a). Similarly, in example (1b) two asynchronous successive events can be an indication for a causal relation, and besides marking temporal relations, the presence of a present participle may trigger a causal or a manner relation.

In this work we consider time and factuality auxiliary tasks in a multi-task setup. We use two resources for this, Factbank and Timebank, described next. We also include information concerning co-reference using Ontonotes annotations, and use the Santa Barbara corpus that contains conversations split into speaking turns. Finally, we incorporate some annotations from the PDTB, another corpus for discourse that however follows a different annotation scheme than the RST-DT. We describe below the different resources used, as well as how we convert the annotations into sequence labeling tasks in order to use them into the multi-task framework. See Table 1 for dataset characteristics.

Factbank and Timebank FactBank (Saurí and Pustejovsky, 2009) is a corpus of news reports that links events to their degree of factuality. The factuality corresponds to four modality values ('certain', 'probable', 'possible', 'unknown') combined to a polarity value ('positive', 'negative', 'unknown'). Factbank has been annotated on top of TimeBank and a part of AQUAINT TimeML, corpora that provide an annotation of the events according to the TimeML specifications (Pustejovsky et al., 2005). Each event is

annotated with several types of information, among which, of particular interest for discourse, are tense ('infinitive', 'pastpart', 'past', 'future', 'prespart', 'present', 'none'), aspect ('perfective', 'progressive', 'perfective_prog', 'none'), polarity ('positive', 'negative', 'none') and modality (e.g. 'have_to', 'would have to', 'should have to', 'possible', 'must', 'could', ...).

In order to build a sequence prediction task upon FactBank and TimeBank annotations, we choose to use sentences as minimal units. We then simply label each sentence in a document with its most frequent tag for each dimension (tense, aspect, modality and factuality). A more fine grained approach would be to retrieve the clause for each event.

Ontonotes OntoNotes (Hovy et al., 2006) contains, among other layers, the annotation of coreference links between entities in documents. Coreference and rhetorical relations are linked, as shown in (Ji and Eisenstein, 2014a). We only keep the English texts. We use sentences as minimal units. The first sentence of the document is annotated as root. We then label each sentence as coreferent to the immediately previous one, to one preceding sentence or as no coreferent.

Santa Barbara corpus We use the Santa Barbara corpus of spoken American English (Du Bois, 2000) to get a sequence labeling task corresponding to turns in a conversation, the idea being that rhetorical structure could share similarities with the structure of conversations. Specifically, we segment the dialogues by pauses and label the first turn-taking utterance as beginning a new turn. All other utterances are labeled as inside the turn of the current speaker. We randomly split the data into documents containing 100 turns.

Penn Discourse Treebank The PDTB (Prasad et al., 2008) is another corpus annotated at the discourse level for English. Contrary to the RST-DT, the annotation is theory neutral: the spans of text are not necessarily all connected, there is no specific structure representing a document. However, the PDTB contains much more data than the RST-DT, with more than two thousands documents annotated against around four hundreds in the RST-DT, making it interesting to try to take advantage of this relatively large amount of discourse annotated data. Since the PDTB and the RST-DT follow different annotation guidelines (i.e. different definitions of the minimal discourse units, of the relations, of the structures involved), multi-task learning is a relevant framework to try to combine them.

In PDTB, EDUs are the arguments of connectives and adjacent sentences inside paragraphs. The EDUs are mainly clauses, but the annotators are free to choose a span not covering an entire clause, or covering more than one sentence. In this paper, we use sentences as EDUs rather than the manually identified segments: if a relation links more than two sentences, we keep the relation between the last sentence of the first argument and the first sentence of the second argument; if a relation links two fragments belonging to two different sentences, we expand the text of each argument to cover the entire sentences. We ignore intra-sentential explicit relations.[5]

We use a BIO annotation scheme for relations between adjacent sentences. More precisely, a sentence is labeled with a BIO label and a discourse relation R_i among the 16 corresponding to the second level in the PDTB hierarchy of sense[6] and the pseudo relation EntRel corresponding to a link between entities. A sentence labeled with "B-R_a" is the first argument of a relation R_a whose second argument is the following sentence. If this following sentence is also the first argument of a relation R_b, it is labeled as "B-R_b", else, it is labeled as ending the current relation, thus "I-R_a". A sentence that is not linked to the previous or following sentence is labeled with "O".

4 Hierarchical bi-LSTMs and baselines

Our main technical contribution is a hierarchical bi-LSTM that composes embeddings for a sequence of words from lower-level word bi-LSTMs, and uses these to predict sequences of labels, encoding especially discourse tree structures.

[5]Preliminary experiments including non overlapping intra-sentential relations did not show improvements against only keeping inter-sentential ones. However, including intra-sentential instances requires more pre-processing and it makes necessary to decide which intra-sentential relations to keep to avoid overlaps.

[6]We only keep the first relation annotated for a pair of arguments.

Task	# Doc	# Labels
Constituent	322	1955
Nuclearity	322	284
Relation	322	1159
Dependency	322	708
Fine grained	322	2,700
Aspect	208	4
Factuality	208	7
Modality	208	10
Polarity	208	3
Tense	208	7
Coreference	2,361	4
PDTB	2,065	35
Speech	446	2

Table 1: Number of documents (# Doc) and labels (# Labels) per task (training data). The main task corresponds to the first line (Constituent).

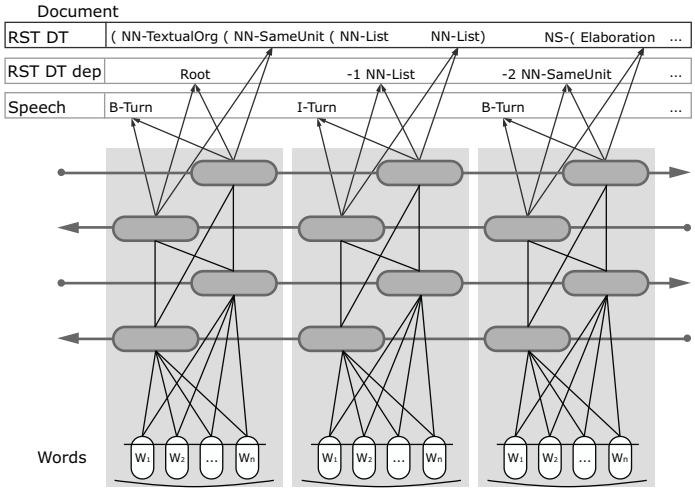


Figure 4: Multi-task learning, hierarchical bi-LSTM network architecture (with 2 layers).

In regular bi-directional recurrent neural networks (bi-RNNs), sequences are read in both regular and reversed order, enabling conditioning predictions on both left and right context. Below, in the forward pass, we run the input data through an embedding layer and compute the predictions of the forward and backward states, which are connected in one or more feed-forward layers, from which we compute the softmax predictions for the sequence based on a linear transformation. We then calculate the objective function derivative for the sequence using cross-entropy (logistic loss) and use backpropagation to calculate gradients and update the weights accordingly. LSTMs (Hochreiter and Schmidhuber, 1997) replace the cells of RNNs with LSTM cells, in which multiplicative gate units learn to open and close access to the error signal.

The overall architecture is shown in Figure 4: each input sequence in the document (i.e. a discourse unit, a speaking turn, a sentence, depending on the task) goes through the hierarchical bi-LSTM that outputs a sequence of labels for the entire document. In particular, an input sequence is represented as a sequence of word embeddings. This sequence goes first through the bi-directional LSTM at the lower level, and the final states (forward, backward) of the bi-LSTMs is taken as input representation for the document-level bi-LSTM at the upper level, which consists of two stacked layers.

For multi-task learning, each task is associated with a specific output layer, whereas the inner layers – the stacked LSTMs – are shared across the tasks. At training time, we randomly sample data points from target or auxiliary tasks and do forward predictions. In the backward pass, we modify the weights of the shared layers and the task-specific outer layer. Except for the outer layer, the target task model is thus regularized by the induction of auxiliary models.

Bi-LSTMs have already been used for syntactic chunking (Huang et al., 2015) and semantic role labeling (Zhou and Xu, 2015), as well as other tasks. Our model differs from most of these models in being a hierarchical model, composing word embeddings into sentence embeddings that are the inputs of a bigger bi-LSTM model. This means our model can also be initialized by pre-trained word embeddings. We implemented our recurrent network in CNN/pycnn,[7] fixing the random seed. We use standard SGD for learning our model parameters.

5 Experiments

Data The RST-DT contains 385 Wall Street Journal articles from the Penn Treebank (Marcus et al., 1993), with 347 documents for training and 38 for testing in the split used in previous studies. We

[7] https://github.com/yoavg/cnn/

follow previous works in using gold standard segmentation (Joty et al., 2012; Ji and Eisenstein, 2014b). Discourse segmentation on the RST-DT can be performed with performance above 95% in accuracy (Xuan Bach et al., 2012). The RST-DT contains newswire articles from the Wall Street Journal.

Baseline As baseline, we train a standard bi-LSTMs on the RST-DT corpus without any auxiliary task information.

Systems As our system, we use hierarchical bi-LSTMs with task supervision from other related tasks. We experiment with using pre-trained embeddings in both baselines and systems.

Competitive systems We compare our approach with the state-of-the-art text-level discourse parser DPLP (Ji and Eisenstein, 2014b). In our comparison, we reproduced the best results reported, including both proposed approaches for DPLP – DPLP concat (*concatenation form* for the projection matrix) and DPLP general (*general form*).

Parameter tuning We used a development set of 25 documents randomly chosen among the training set. We optimized the number of passes p over the data ($p \in [10, 60]$), the value of the Gaussian noise ($\sigma \in \{0.0, 0.2\}$), the number of hidden dimensions ($d \in \{200, 400\}$), the number of stacked layers ($h \in \{1, 2, 3, 4, 5\}$), and the auxiliary tasks to be included and combined. In the end, we report results using 2 feed-forward layers with 128 dimensions, a Gaussian noise with sigma of 0.2, 200 hidden dimensions, 20 passes over the data, 2 layers and Polyglot embeddings (Al-Rfou et al., 2013)[8].

Metrics Following (Marcu, 2000b) and most subsequent work, output trees are evaluated against gold trees in terms of how similar they bracket the EDUs (Span), how often they agree about nuclei when predicting a true bracket (Nuclearity), and in terms of the relation label, i.e., the overlap between the shared brackets between predicted and gold trees (Relation).[9] These scores are analogous to labeled and unlabeled syntactic parser evaluation metrics. The exact definitions of the three metrics are:

- Span: This metric is the unlabeled F_1 over gold and predicted trees, and identical to the PARSEVAL metric in syntactic parsing. This metric reflects a correct bracketing and ignores nuclearity and relation labels.
- Nuclearity: This metric is the labeled F_1 over gold and predicted discourse trees, disregarding the discourse relations.
- Relation: This metric is the labeled F_1 over gold and predicted discourse trees, disregarding the nuclearity information.

6 Results

Our results are summarized in Table 2. We note that the bi-LSTM baseline that only receives task supervision from RST-DT discourse trees achieves scores comparable to the state-of-the-art for the unlabeled structure (Span), but lower scores for the other metrics.

More importantly, multi-task learning, i.e., combining different representations of the data, leads to substantial improvements over our baseline for 8 out of the 11 tasks tested. We found that it is much more beneficial to have multiple views, thus, interestingly using different views on the data, with all the tasks derived from the main one leading to improvements (RSTFin, RSTDep, Nuc+Lab). Especially, the model takes advantage of using the data from the main task but with fine grained relations, with 82.88% in unlabelled F_1 (Span), 67.46% in labelled F_1 considering nuclearity (Nuclearity), and 53.25% in labelled F_1 considering relations (Relation). This auxiliary view helps the model to discriminate between the relations.

Most of the tasks derived from additional annotations also lead to improvements. Especially, we found that the speech data (Speech) leads to good results: this confirms our assumption that the turns

[8] `https://sites.google.com/site/rmyeid/projects/polyglot`

[9] We use the evaluation script provided at `https://github.com/jiyfeng/DPLP`.

System	RSTFin	Fact	Speech	Asp	RSTDep	Nuc+lab	Mod	Pol	PDTB	Coref	Ten	Span	Nuclearity	Relation
Prior work														
DPLP concat	-	-	-	-	-	-	-	-	-	-	-	82.08	**71.13**	61.63
DPLP general	-	-	-	-	-	-	-	-	-	-	-	81.60	70.95	**61.75**
Our work														
Hier-LSTM	-	-	-	-	-	-	-	-	-	-	-	81.39	64.54	49.15
MTL-Hier-LSTM	✓	-	-	-	-	-	-	-	-	-	-	82.88	67.46	53.25
MTL-Hier-LSTM	-	✓	-	-	-	-	-	-	-	-	-	83.40	67.16	52.10
MTL-Hier-LSTM	-	-	✓	-	-	-	-	-	-	-	-	83.26	67.51	51.75
MTL-Hier-LSTM	-	-	-	✓	-	-	-	-	-	-	-	83.69	66.25	51.25
MTL-Hier-LSTM	-	-	-	-	✓	-	-	-	-	-	-	81.25	65.34	51.24
MTL-Hier-LSTM	-	-	-	-	-	✓	-	-	-	-	-	82.09	65.68	51.12
MTL-Hier-LSTM	-	-	-	-	-	-	✓	-	-	-	-	81.66	65.31	50.58
MTL-Hier-LSTM	-	-	-	-	-	-	-	✓	-	-	-	82.01	65.29	50.11
MTL-Hier-LSTM	-	-	-	-	-	-	-	-	✓	-	-	81.61	63.10	48.89
MTL-Hier-LSTM	-	-	-	-	-	-	-	-	-	✓	-	80.26	63.35	47.70
MTL-Hier-LSTM	-	-	-	-	-	-	-	-	-	-	✓	81.33	62.34	47.57
Best combination	-	-	-	-	✓	✓	✓	-	✓	-	-	**83.62**	69.77	55.11
Human annotation	-	-	-	-	-	-	-	-	-	-	-	88.70	77.72	65.75

Table 2: Parsing results of different models on the RST-DT test data. Prior work results are reprinted (DPLP) (Ji and Eisenstein, 2014b). The auxiliary tasks are: RST-DT sequences from trees but keeping only the relations (Lab) or the nuclearity information (Nuc), RST-DT dependency parsing (RSTDep), sequence labels from Factbank using modality information (Mod), and inter-sentential relation from the PDTB (PDTB).

of speech and the structures involved share some similarities with the rhetorical units and structures. Moreover, factuality (Fact), aspect (Asp), modality (Mod) and polarity (Pol) information prove to be useful for discourse parsing. On the other hand, the tasks derived from tense (Ten) and coreference (Coref) annotations do not lead to improvements. These information, crucial for the task, would probably benefit from a finer grained encoding at the sentence level. The task derived from the PDTB, taken alone, lowers slightly the results.

Finally, we experiment with task combinations. Our best system only uses the views based on nuclearity and label (Nuc+lab), the encoding of the tree as dependency (RSTDep), the modality information (Mod) and the task derived from the PDTB data. This combination leads to substantial improvements, with 83.62% in unlabelled F_1 (Span), 69.77% in labelled F_1 considering nuclearity (Nuclearity), and 55.11% in labelled F_1 considering relations (Relation). This closes 60,7% of the gap to human performance on unlabelled discourse parsing. It is slightly better than state-of-the-art in discourse parsing for Span. Feng and Hirst (2014) proposed a system with better scores for these metrics, but the comparison to their system is not entirely fair, since they add common-sense constraints that are not clearly explained and post-editing. Besides, there is no single approach that does best for all metrics.

Our results indicate that our architecture learns useful representations capturing some of the syntactic and contextual information needed for the task.

7 Related work

Some of the first text-level discourse parsers were based on hand-crafted rules and heuristics, making mainly use of the connectives as indication of the relations and using constraints to build the entire RST trees (Marcu, 2000a; Le Thanh et al., 2004).

More recent works proposed learning based approaches inspired by syntactic parsing. Hernault et al. (2010) (HILDA) proposed a greedy approach with SVM classifiers performing attachment and relation classification at each step of the tree building. Joty et al. (2012) (TSP) built a two-stage parsing system, training separate sequential models (CRF) for the intra and the inter-sentential levels. These models jointly learn the relation and the structure, and a CKY-like algorithm is used to find the optimal tree. Feng and Hirst (2014) noticed the inefficiency of TSP and proposed a greedy approach inspired by

HILDA but using CRF as local models for the inter- and intra-sententials levels, allowing to take into account sequential dependencies.

Last studies also focused on the issue of building a good representation of the data. Feng and Hirst (2012) introduced linguistic features, mostly syntactic and contextual ones. Ji and Eisenstein (2014b) (DPLP) proposed to learn jointly the representation and the task, more precisely a projection matrix that maps the bag-of-words representation of the discourse units into a new vector space. This idea is promising, but a drawback could be the limited amount of data available in the RST-DT, an issue even more crucial for other languages.

Discourse parsing has proven useful for many applications (Taboada and Mann, 2006), ranging from summarization (Daumé III and Marcu, 2009; Thione et al., 2004; Sporleder and Lapata, 2005; Louis et al., 2010), sentiment analysis (Bhatia et al., 2015) or essay scoring (Burstein et al., 2003; Higgins et al., 2004). However, the range of applications and the improvement allowed are for now limited by the low performance of the existing discourse parsers.

We are not aware of other studies trying to combine various encodings of the RST-DT trees or to leverage relevant information through multi-task learning to improve discourse parsing. To the best of our knowledge, multi-task learning has only been used for discourse relation classification (Lan et al., 2013) on the Penn Discourse Treebank to combine implicit and explicit data.

We are not the first to propose using bi-LSTMs for tree structure prediction problems. Zhou and Xu (2015), for example, use bi-LSTMs to produce semantic role labelling structures. Zhang et al. (2015) did the same for relation extraction. None of them considered multi-task learning architectures, however. Multi-task learning in deep networks was first introduced by Caruana (1993), who did multi-task learning by doing parameter sharing across several deep networks, letting them share hidden layers. The same technique was used by Collobert et al. (2011) for various NLP tasks, and for sentence compression in (Klerke et al., 2016). Hierarchical multi-task bi-LSTMs have been previously used for part-of-speech tagging (Plank et al., 2016).

8 Conclusion and future work

We presented the first experiments exploiting different views of the data and related tasks to improve text-level discourse parsing. We presented a hierarchical bi-LSTM model allowing to leverage information from various sequence prediction tasks (multi-task learning) that achieves a new state-of-the-art performance on unlabeled text-level discourse parsing, and competitive performance in predicting nuclearity and discourse relations.

For relation prediction, future work includes adding additional information at the sentence level, such as syntactic information used in most of the studies identifying discourse relation on the PDTB (Pitler et al., 2009; Lin et al., 2009; Rutherford and Xue, 2014), or better representation of the combination between the arguments (Ji and Eisenstein, 2014b).

Acknowledgements

We thank the three anonymous reviewers for their comments. Chloé Braud and Anders Søgaard were funded by the ERC Starting Grant LOWLANDS No. 313695.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of Conll*.

Nicholas Asher and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from rst discourse parsing. In *Proceedings of EMNLP*.

Jill Burstein, Daniel Marcu, and Kevin Knight. 2003. Finding the write stuff: automatic identification of discourse structure in student essays. *IEEE Intelligent Systems: Special Issue on Advances in Natural Language Processing*, 18.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*.

Rich Caruana. 1993. Multitask learning: a knowledge-based source of inductive bias. In *ICML*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Laurence Danlos and Owen Rambow. 2011. Discourse Relations and Propositional Attitudes. In *CID 2011 - Fourth International workshop on Constraints in Discourse*.

Hal Daumé III and Daniel Marcu. 2009. A noisy-channel model for document compression. In *Proceedings of ACL*.

John W Du Bois. 2000. *Santa Barbara Corpus of Spoken American English*. University of California, Santa Barbara Center for the Study of Discourse.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of ACL*.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of ACL*.

Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010. Hilda: A discourse parser using support vector machine classification. *Dialogue and Discourse*, 1:1–33.

Derrick Higgins, Jill Burstein, Daniel Marcu, and Claudia Gentile. 2004. Evaluating multiple aspects of coherence in student essays. In *Proceedings of HLT-NAACL*.

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

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of HLT-NAACL*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv:1508.01991*.

Yangfeng Ji and Jacob Eisenstein. 2014a. One vector is not enough: Entity-augmented distributional semantics for discourse relations. *TACL*.

Yangfeng Ji and Jacob Eisenstein. 2014b. Representation learning for text-level discourse parsing. In *Proceedings of ACL*.

Xin Jin, Fuzhen Zhuang, Shuhui Wang, Qing He, and Zhongzhi Shi. 2013. Shared structure learning for multiple tasks with multiple views. In *Machine Learning and Knowledge Discovery in Databases*, pages 353–368. Springer.

Shafiq R. Joty, Giuseppe Carenini, and Raymond T. Ng. 2012. A novel discriminative framework for sentence-level discourse analysis. In *Proceedings of EMNLP*.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *Proceedings of NAACL*.

Man Lan, Yu Xu, and Zhengyu Niu. 2013. Leveraging synthetic discourse data via multi-task learning for implicit discourse relation recognition. In *Proceedings of ACL*.

Huong Le Thanh, Geetha Abeysinghe, and Christian Huyck. 2004. Generating discourse structures for written text. In *Proceedings of COLING*.

Sujian Li, Liang Wang, Ziqiang Cao, and Wenjie Li. 2014. Text-level discourse dependency parsing. In *Proceedings of ACL*.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of EMNLP*.

Annie Louis, Aravind Joshi, and Ani Nenkova. 2010. Discourse indicators for content selection in summarization. In *Proceedings of SIGDIAL*.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory : Toward a functional theory of text organization. *Text*, 8:243–281.

Daniel Marcu. 1997. *The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts*. Ph.D. thesis, University of Toronto.

Daniel Marcu. 2000a. The rhetorical parsing of unrestricted texts: A surface-based approach. *Computational Linguistics*.

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

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Philippe Muller, Stergos Afantenos, Pascal Denis, and Nicholas Asher. 2012. Constrained decoding for text-level discourse parsing. In *Proceedings of COLING*.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of ACL-IJCNLP*.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of ACL*.

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

James Pustejovsky, Patrick Hanks, Roser Saurí, Andrew See, Rob Gaizauskas, Andrea Setzer, Dragomir R. Radev, Beth Sundheim, David S. Day, Lisa Ferro, and Marcia Lazo. 2003. The TIMEBANK corpus. In *Proceedings of Corpus Linguistics*.

James Pustejovsky, Robert Knippen, Jessica Littman, and Roser Saurí. 2005. Temporal and event information in natural language text. *Language Resources and Evaluation*, 39(2):123–164.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proceedings of EACL*.

Roser Saurí and James Pustejovsky. 2009. Factbank: a corpus annotated with event factuality. In *Proceedings of LREC*, volume 43, pages 227–268.

Caroline Sporleder and Mirella Lapata. 2005. Discourse chunking and its application to sentence compression. In *Proceedings of HLT/EMNLP*.

Manfred Stede. 2011. *Discourse Processing*. Morgan & Claypool.

Maite Taboada and Debopam Das. 2013. Annotation upon annotation: Adding signalling information to a corpus of discourse relations. *Dialogue and Discourse*, 4(2):249–281.

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

Gian Lorenzo Thione, Martin Van den Berg, Livia Polanyi, and Chris Culy. 2004. Hybrid text summarization: Combining external relevance measures with structural analysis. In *Proceedings of the ACL Workshop Text Summarization Branches Out*.

Ngo Xuan Bach, Nguyen Le Minh, and Akira Shimazu. 2012. A reranking model for discourse segmentation using subtree features. In *Proceedings of Sigdial*.

Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. 2015. Bidirectional long short-term memory networks for relation classification. In *Proceedings of PACLIC*.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of ACL*.

Implicit Discourse Relation Recognition with Context-aware Character-enhanced Embeddings

Lianhui Qin[1,2], **Zhisong Zhang**[1,2], **Hai Zhao**[1,2,*]
[1]Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, 200240, China
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
{qinlianhui, zzs2011}@sjtu.edu.cn,zhaohai@cs.sjtu.edu.cn

Abstract

For the task of implicit discourse relation recognition, traditional models utilizing manual features can suffer from data sparsity problem. Neural models provide a solution with distributed representations, which could encode the latent semantic information, and are suitable for recognizing semantic relations between argument pairs. However, conventional vector representations usually adopt embeddings at the word level and cannot well handle the rare word problem without carefully considering morphological information at character level. Moreover, embeddings are assigned to individual words independently, which lacks of the crucial contextual information. This paper proposes a neural model utilizing context-aware character-enhanced embeddings to alleviate the drawbacks of the current word level representation. Our experiments show that the enhanced embeddings work well and the proposed model obtains state-of-the-art results.

1 Introduction

It is widely agreed that in a formal text, units including clauses and sentences are not isolated but instead connected logically, semantically, and syntactically. Discourse parsing is a fundamental task in natural language processing (NLP) that analyzes the latent relation structure and discovers those connections across text units. It could benefit various downstream NLP applications such as question answering (Chai and Jin, 2004; Verberne et al., 2007), machine translation (Hardmeier, 2012; Guzmán et al., 2014), sentiment analysis (Bhatia et al., 2015; Hu et al., 2016b), and automatic summarization (Maskey and Hirschberg, 2005; Murray et al., 2006).

For discourse parsing, Penn Discourse Treebank (PDTB) (Prasad et al., 2008) provides the lexically-grounded annotations of discourse relations. Each discourse relation consists of two abstract object arguments and the corresponding sense annotations, which can be roughly characterized according to whether explicit connectives could be drawn from the texts. In *Explicit* relations, explicit connectives can be found in the texts; when such indicators are not given directly, an inferred connective expression could be inserted, forming *Implicit* relations. The following two examples describes these two kinds of discourse relations: the former has an explicit connective *"so"* which reveals the *Explicit* relation, while in the latter case, an inferred *Implicit* connective *"that is"* has to be inserted to express the relation.

(1) **Arg1:** We're standing in gasoline.
Arg2: So don't smoke. (Contingency.Cause.Result - wsj_0596)

(2) **Arg1:** The ploy worked.
Arg2: Implicit=that is The defense won. (Contingency.Cause - wsj_1267)

It has been shown that discourse connective is crucial for high-accuracy relation recognition (Pitler et al., 2009; Lin et al., 2014). Compared to explicit discourse relations in which senses between adjacent

*Corresponding author. This paper was partially supported by Cai Yuanpei Program (CSC No. 201304490199 and No. 201304490171), National Natural Science Foundation of China (No. 61170114, No. 61672343 and No. 61272248), National Basic Research Program of China (No. 2013CB329401), Major Basic Research Program of Shanghai Science and Technology Committee (No. 15JC1400103), Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04), and Key Project of National Society Science Foundation of China (No. 15-ZDA041).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,
pages 1914–1924, Osaka, Japan, December 11-17 2016.

clauses are effectively indicated by explicit connectives like *"but"* and *"so"*, implicit discourse relation recognition is much more difficult. Without effective indicators, the relations could only be inferred from indirect plain texts, which makes implicit discourse relation recognition the bottleneck of the entire discourse parsing system (Qin et al., 2016a; Li et al., 2016; Chen et al., 2015). This paper attempts to deal with this challenging task.

The challenge stems from the fact that, without connective cues, recognizing implicit relation has to rely solely on two textual arguments, and it must capture latent semantic and logical relationship between two arguments in discourse-level. First, given limited amount of annotated corpus, both the traditional indicator feature based methods and the recent embedding based neural methods (Wang et al., 2015) can suffer from insufficient data. The training is especially difficult for rare words, which appear rarely in the corpus but generally take up a large share of the dictionary, making it hard to effectively learn their representations, resulting in high perplexities for discourse relation recognition. Moreover, implicit relation recognition calls for semantic understanding, which needs to encode the word meaning in the context and the sentence-level understanding for the argument pairs. Considering the complexity of natural language, the task is quite nontrivial and requires more effective encoding of the arguments.

Conventional methods for implicit discourse relation recognition are based on manually specified indicator features, such as bag-of-words, production rules, and other linguistically-informed features (Zhou et al., 2010; Park and Cardie, 2012; Biran and McKeown, 2013; Rutherford and Xue, 2014). Recently, embedding based neural models have been proved effective to address the data sparsity problem that is not well solved in traditional methods. The key techniques include real-valued dense embeddings for feature representations and non-linear neural models for feature combinations and transformations. However, most of the neural models take words as the smallest processing units, which can suffer a lot from insufficient training on rare words. Discourse parsing, as the highest level language processing at present, covers word and sentence levels for feature representation. This work extends the current word-level representation onto more fine-grained character-level which is helpful for encoding morphology information and alleviating the rare word problem.

In summary, this paper presents a neural model with context-aware character-enhanced embeddings to address implicit discourse relation recognition task. Recently, character-aware models have been popular for English and other morphologically rich languages (Kim et al., 2016; Zhang et al., 2015b; Ling et al., 2015). The proposed model enhanced the word embeddings with character-aware representations learned from stacked convolutional and recurrent neural models. Utilizing these enhanced embeddings, the model covers information of three levels from character, word, to sentence. Through extensive experiments on standard discourse corpus, we analyze several models and show the superiority of the proposed method.

The remaining of the paper is organized as follows: Section 2 discusses related work; Section 3 describes the proposed model; Section 4 provides the details of experiments and model analysis; and Section 5 concludes the paper.

2 Related work

Implicit discourse relation recognition is the subcomponent of the end-to-end discourse parsing system, which is also used as the share-task in CoNLL 2015 and CoNLL 2016 (Xue et al., 2015; Xue et al., 2016). In the share-task, the classification task concerns other Non-Explicit types including *EntRel* and *AltLex*, in addition to the *Implicit* relations.

Early work for implicit discourse relation recognition focuses on typical machine learning solutions with sparse indicator features and linear models. Pitler et al. (2009) use several linguistically informed features, including polarity tags, Levin verb classes and length of verb phrases. Zhou et al (2010) improve the performance through predicting connective words as extra features. Park and Cardie (2012) propose a method using a locally-optimal feature set. Biran and McKeown (2013) collect word pairs from arguments of explicit examples to help the learning. Rutherford and Xue (2014) employ Brown cluster pairs to represent discourse relation and incorporate coreference patterns to identify the meaning in text. Li and Nenkova (2014) introduce a syntactic representation to reduce sparsity. Rutherford and

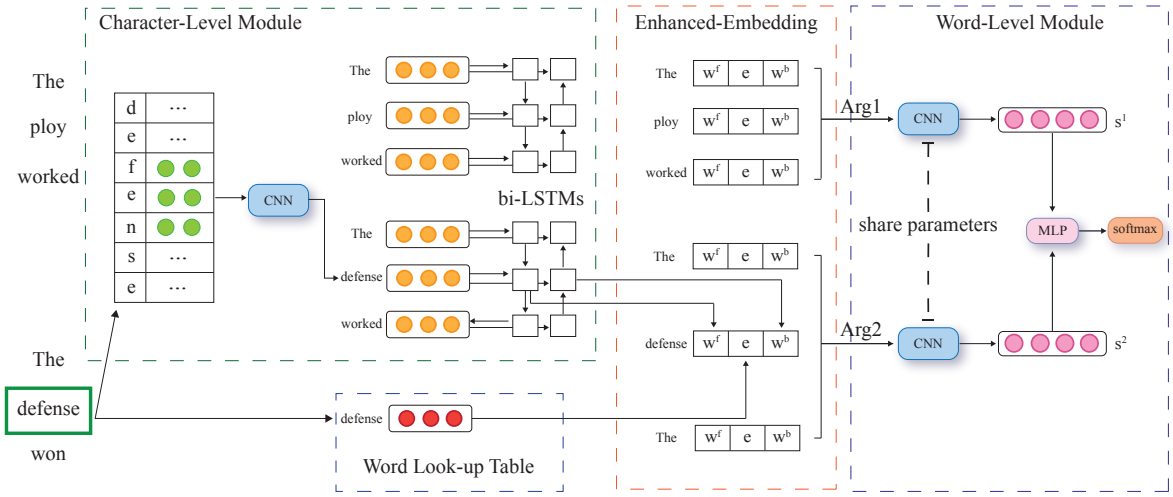


Figure 1: Architecture of the proposed model.

Xue (2015) and Ji et al. (2015) add automatically-labeled instances to expand data. Fisher and Simmons (2015) incorporate a mixture of labeled and unlabeled data to reduce the need for annotated data.

More recently, neural network models have been proved effective for NLP tasks (Wang et al., 2016; Zhang et al., 2016; Cai and Zhao, 2016; Hu et al., 2016a) and also utilized for implicit discourse relation recognition. Ji and Eisenstein (2015) adopt recursive neural network and incorporated with entity-augmented distributed semantics. Zhang et al. (2015a) propose a simplified neural network which contains only one hidden layer and use three different pooling operations (max, min, average). Chen et al. (2016) adopt a deep gated neural model to capture the semantic interactions between argument pairs. Ji et al. (2016) propose a latent variable recurrent neural network architecture for jointly modeling sequences of words. (Qin et al., 2016b) propose a stacking neural network model to solve the classification problem. In their model, convolutional neural network is utilized for sentence modeling and a collaborative gated neural network is proposed for feature transformation.

3 Model

3.1 Architecture

The architecture of our model is shown in Figure 1. The model is a hybrid neural network including a character-level module and a word-level module. The character-level module receives inputs of character-level embeddings followed by stacked CNN and bidirectional LSTMs layers. First, the convolutional and max-pooling operations perform local information encoding and feature selection, obtaining a fixed-dimensional representation of the character-based word representation sequence. Then via bidirectional LSTMs, the sequence is transformed to a new sequence which encodes the rich contextual information. This new sequence is the output of the character-level module which models context-aware character-level information, and will be utilized in later layers. In the word-level module, the character-based word representations will be concatenated to ordinary word embeddings, forming enhanced embeddings which integrate both character-level and word-level information. Later CNN will be utilized again to obtain sentence-level representations for the two arguments, followed by conventional hidden layers and a softmax layer for the final classification.

3.2 Character-Level Module

This module aims to get the most out of the character sequence and obtain the character-based word representations, utilizing an architecture of stacked CNN and LSTM. Modeling from character level could alleviate rare words problem and useful capture morphological information, like the prefixes and suffixes of words.

Character Embedding The character representation will still be in the form of embeddings. In this task, we define an alphabet of characters which contains uppercase and lowercase letter as well as numbers and punctuation. The input word will be decomposed into a character sequence. Through a character look-up table, a word will be projected to a sequence of character vectors: $[\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n]$, where $\mathbf{c}_i \in \mathbb{R}^{d_c}$ is the vector for the i-th character in the word with dimension d_c and n is word length. For the convenience of notation, the character vectors for a word can be regarded as a character matrix $\mathbf{C}$:

$$\mathbf{C} = [\mathbf{c}_1; \mathbf{c}_2; \ldots; \mathbf{c}_n]$$

Convolutional Neural Network A convolutional operation followed by a max-pooling operation will be applied to the character matrix $\mathbf{C}$ of each word. The convolutional layer is used to extract and combine local features from adjacent characters and the following max-pooling layer forms the representations for the current word. For the convolutional operation, k groups of filter matrices [$\mathbf{F}_1$, $\mathbf{F}_2$, ..., $\mathbf{F}_k$] with variable sizes $[l_1, l_2, \ldots, l_k]$ and biases [$\mathbf{b}_1$, $\mathbf{b}_2$, ..., $\mathbf{b}_k$] are utilized. Each of them transforms the character matrix $\mathbf{C}$ to another sequence. The transformed sequences $\mathbf{C}'_j (j \in [1, k])$ will be obtained as follows:

$$\mathbf{C}'_j = [\ldots; \tanh(\mathbf{F}_j \cdot \mathbf{C}_{[i:i+l_j-1]} + \mathbf{b}_j); \ldots]$$

Here, i indexes the convolutional window. Next, a one-max-pooling operation is adopted and the representation $\mathbf{w}$ for a word is obtained through concatenating all the mappings after pooling as follows:

$$\mathbf{w}'_j = \mathbf{max}(\mathbf{C}'_j)$$
$$\mathbf{w} = [\mathbf{w}'_1 \oplus \mathbf{w}'_2 \oplus \ldots \mathbf{w}'_k]$$

Bidirectional LSTM The character-based word representation obtained through CNN can be directly utilized in the word-level module, however, each word vector from the CNN is individually obtained and lacks of the encoding of contextual information. In a sentence, word can never be understood independently without context. Nearby words can offer important cues to the current word as suggested by N-gram language model and context-aware sentence modeling. Motivated by this, we propose to utilize bidirectional LSTMs to encode the character-based word vectors.

Given the character-based word representations $[\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n]$ as the input sequence, an LSTM computes the state sequence $[\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n]$ by applying the following formulation for each time step:

$$\mathbf{i}_t = \sigma(\mathbf{W}^i_w \mathbf{w}_t + \mathbf{W}^i_h \mathbf{h}_{t-1} + \mathbf{W}^i_c \mathbf{w}_{t-1} + \mathbf{b}_i)$$
$$\mathbf{f}_t = \sigma(\mathbf{W}^f_w \mathbf{w}_t + \mathbf{W}^f_h \mathbf{h}_{t-1} + \mathbf{W}^f_c \mathbf{w}_{t-1} + \mathbf{b}_f)$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot tanh(\mathbf{W}^c_w \mathbf{w}_t + \mathbf{W}^c_h \mathbf{h}_{t-1} + \mathbf{b}_c)$$
$$\mathbf{o}_t = \sigma(\mathbf{W}^o_w \mathbf{w}_t + \mathbf{W}^o_h \mathbf{h}_{t-1} + \mathbf{b}_o)$$
$$\mathbf{h}_t = \tanh(\mathbf{c}_t) \odot \mathbf{o}_t$$

Here the σ denotes the sigmoid function and the $\odot$ denotes element-wise multiplication. $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{c}_t$, $\mathbf{o}_t$ and $\mathbf{h}_t$ stand for input gate, forget gate, memory cells, output gate and the current state, respectively. Finally, the state sequence $[\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n]$ will be utilized as the context-aware word representations. Recent works (Graves et al., 2013; Graves et al., 2005) show that backward LSTM can also effectively encode the context by modeling the word sequence backward, combined with the ordinary forward LSTM, the so-called Bidirectional LSTM could effectively capture the information from both past and future words, and we will utilize it in this module. We will denote the output state sequence (originally noted as $\mathbf{h}$) of forward LSTM as $[\mathbf{w}^f_1, \mathbf{w}^f_2, \ldots, \mathbf{w}^f_n]$ and the one of the backward LSTM as $[\mathbf{w}^b_1, \mathbf{w}^b_2, \ldots, \mathbf{w}^b_n]$.

3.3 Word-Level Module

In recent neural models, words are represented as real-valued dense vectors. With the prevalence of deep learning methods in NLP, continuous space word vectors have been found an effective means for word

representations. Unlike the traditional model in which words usually represent as one-hot vectors and are independent with each other, vector space models reveal the relationship and capture the intuition among words which different or similar to others along a variety of dimensions (Mikolov et al., 2013). However, embeddings considering only at word level is usually not good for rare words as discussed above and we introduce character-level embedding to enhance the current word embedding.

Enhanced Word Embedding For obtaining word representations, we enhance ordinary word vectors with the character-based vectors obtained from the character-level module by concatenating all the representations on words. Thus a sequence of enhanced word embeddings could be obtained, which could cover contextual information from character-level to word-level, and the character-based embedding could alleviate rare word problems in some way. Formally speaking, an argument could be represented as a sequence as follows:

$$\mathbf{M} = [\mathbf{w}_1^f \oplus \mathbf{e}_1 \oplus \mathbf{w}_1^b; \mathbf{w}_2^f \oplus \mathbf{e}_2 \oplus \mathbf{w}_2^b; \ldots; \mathbf{w}_n^f \oplus \mathbf{e}_n \oplus \mathbf{w}_n^b]$$

where $\oplus$ is the concatenation operator. $\mathbf{w}^f, \mathbf{e}, \mathbf{w}^b$ stand for the state of forward LSTM, word embedding and the state of backward LSTM, respectively.

Convolutional Neural Network In the word-level module, CNN is utilized again to extract local context features. Like the convolutional layer in character-level module, several groups of filter matrices with various filter window sizes are utilized to extract features from different ranges. This procedure is quite similar to the one in character-level module and we will leave out the formulas. Unlike the character-level CNN, here the convolutional operation is applied on the arguments and the following max-pooling layer will produce the sentence vectors. Via parameter sharing, this feature extraction procedure become same for both arguments. We will note the sentence vectors for the two arguments as $\mathbf{s}^1$ and $\mathbf{s}^2$.

Softmax Getting the sentence-level representations, we can concatenate the sentence vectors and feed them to the conventional softmax layer for the final classification.

$$\mathbf{v} = \mathbf{s}^1 \oplus \mathbf{s}^2$$

$$\Pr(y_i) = \frac{\exp \mathbf{w}^i \times \mathbf{v}}{\sum_j^l \exp \mathbf{w}^j \times \mathbf{v}}$$

Here, $\Pr(y_i)$ means the probability of assigning the instance to label i, $\mathbf{w}$ indicates the parameters in the final softmax layer. Additionally, multilayer perceptron (MLP) hidden layers could be added between sentence vectors and the final softmax layer, and we will leave out the descriptions for brevity.

3.4 Training

For training, the object is the cross-entropy error with *L2* regularization:

$$E(\hat{y}, y) = -\sum_j^l y_j \times \log(\Pr(\hat{y}_j))$$

$$J(\theta) = \frac{1}{m} \sum_k^m E(\hat{y}^{(k)}, y^{(k)}) + \frac{\lambda}{2} \|\theta\|^2$$

where $y^{(k)}$ is the gold labels and $\hat{y}^{(k)}$ is the predicted ones. For the optimization process, we apply the diagonal variant of AdaGrad (Duchi et al., 2011) with mini-batches.

4 Experiment

PDTB 2.0[1], which is one of the largest manually annotated corpus of discourse relation, is utilized for the experiments. Annotated on Wall Street Journal corpus with one million words, the data contain

[1] http://www.seas.upenn.edu/ pdtb/

16,224 implicit relations. It provides three hierarchies of relations: Level 1 *Class*, Level 2 *Type*, and Level 3 *Subtypes*. The first level consists of four major relation *Class*: COMPARISON, CONTINGENCY, EXPANSION and TEMPORAL. There are 16 Level 2 relation types of implicit relations. The third level of *Subtypes* is types that are only available for specific types.

For the evaluation of implicit relation classification, there are two settings in previous works: one is multi-class classification for second-level discourse relations (Lin et al., 2009); the other is the "One-Versus-Others" setting which employs binary classification only for Level 1 $Class$, which is first used by Pitler et at. (2009). Note that the results for the latter setting can be also derived from the specific statistics over the results of the former setting. In this paper, we will focus on the more practical multi-class classification, which is a necessary component for building a complete discourse parser such as that for the shared tasks of CoNLL-2015 and 2016 (Xue et al., 2015; Xue et al., 2016). For the model analysis, we perform the experiments with the multi-classification setting. In order to compare with previous results, we will also evaluate our system on the binary relation classification task.

4.1 Multi-class classification

Following (Lin et al., 2009), we adopt the standard PDTB splittings as follows: Sections 2-21 as training set, Section 22 as development set and Section 23 as test set. We will denote this dataset as the PDTB Standard setting *PDTB-STD*. In order to be in consistence with previous setting, we also remove 5 *Types* which are too few in the corpus: CONDITION, PRAGMATIC CONCESSION, PRAGMATIC CONCESSION, PRAGMATIC CONTRAST and EXCEPTION. Thus, we use the remaining 11 Level 2 *Types* in our experiments. In addition, for nearly 2% of the implicit relations have more than one type during annotating in PDTB, we consider these relations as two relation types with the same argument pairs when training. During testing, the predictions which match one of the gold types will be considered as correct. To compare with the state-of-the-art system (Ji and Eisenstein, 2015), which uses a slightly different setting: Sections 2-20 as training set, 0-1 as development set, and 21-22 as testing set. We also run experiments on this setting (noted as the PDTB Alternative setting *PDTB-ALT*) and show the comparisons.

4.1.1 Hyper-Parameters

For the hyper-parameters of the model and training process, we fix the lengths of both arguments (number of words) to be 80 and the lengths of the words (number of characters) to be 20, and apply truncating or zero-padding when necessary. The dimensions for character embeddings and word embeddings are 30 and 300 respectively. The word embeddings are initialized with pre-trained word vectors using *word2vec*[2] (Mikolov et al., 2013) and other parameters are randomly initialized by sampling from uniform distribution in [-0.5, 0.5] including character embeddings. The learning rate is set as 0.002.

In the character-level module, the CNN part uses three groups of 128 filters, with filter window sizes of (2, 3, 4); while the output dimensions of bidirectional LSTMs is set to 50. In the word-level module, the CNN part also contains three groups of filters. For we need more parameters to accurately model the sentence level information, each group has 1024 filters and their filter window sizes are (2, 4, 8). We also add another hidden layer above the concatenated sentence vectors and its dimension is set to 100.

4.1.2 Models

In this sub-section, we will describe the models in the comparisons of our main experiments, which show the effectiveness of the proposed neural model with enhanced embeddings. Our experiments mainly concerns four group of models: Baseline Models, Word-level Only Neural Models, Character-level Only Neural Models and Combined Models. The proposed model, namely **Char+Word-Enhanced**, falls into the last group and many other models can be considered as partial models of it.

Baseline Models These include simplified models or previous traditional model.

- **Majority Baseline** The most commom *Type* class is CAUSE, which accounts for 26.1% of the implicit relations in the PDTB test set.

[2]http://www.code.google.com/p/word2vec

Model	Accuracy
Majority Baseline	26.10
Word Representation	34.07
Lin et al. (2009)	40.20
Word-BiLSTMs	33.42
Word-CNN	41.12
Char-CNN	30.15
Char-[CNN+BiLSTMs]	34.86
Char+Word-Concat	42.55
Char+Word-Enhanced	**43.81**

Table 1: Comparisons on test set of *PDTB-STD* for multi-class classification.

Model	Accuracy
Majority Baseline	26.03
Word Representation	36.86
Lin et al. (2009)	-
+Brown clusters	40.66
Ji and Eisenstein (2015)	36.98
+Entity semantics	37.63
+Surface features	43.75
+both	44.59
Char+Word-Enhanced	**45.04**

Table 2: Comparisons on test set of *PDTB-ALT* for multi-class classification.

- **Word Representation** This model just utilizes sum of word vector as sentence vectors, for showing how the model with only word vector embeddings can work.
- **Lin et al. (2009)** Traditional linear model with manually specified features, including production rules, dependency rules, word pairs and context features.

Word-level Models These models only utilize conventional word vectors through a word-level embedding table looking-up process and does not use the character-level module.

- **Word-level CNN** This model adopts CNN with conventional word embeddings, which does not utilize the character-level module.
- **Word-level BiLSTMs** This model replaces CNN to Bidirectional LSTMs, also with conventional word embeddings. The sentence vectors will be the last state vector of the LSTMs.

Character-level Models These models only make use of the word representations learned from the Character-level module. For the word-level module (from word-level representations to sentence-level ones), CNN will be used.

- **Char-level CNN** This model utilizes only the embeddings from character-level module (without concatenating the word-level embeddings), and in the character-level part BiLSTMs are not utilized and the word representations are directly from CNN.
- **Char-level CNN+BiLSTMs** This model integrates BiLSTMs in the character-level module, which could encode the context information in the character-level embeddings.

Combined Models These combine the Char-level and Word-level modules (**Char-level CNN+BiLSTMs** and **Word-level CNN**) through concatenation on different levels.

- **Char+Word-Concat** This model combines the two modules at the sentence representation level, by concatenating the sentence vectors.
- **Char+Word-Enhanced** This is the proposed model, which combines the modules at the word embedding level, forming enhanced embeddings.

4.1.3 Model Analysis

The analysis of the models will be based on the results of Table 1 and we will discuss them in groups. First, the traditional linear model performs well, but it needs manually specified features. Simply adding word vectors is not a very good idea, because it ignores the crucial information of word order. In the second group, we could see that CNN performs well, for it provides the capacity of modeling local word sequences (via convolutional operations) and capturing sentence-level features (via max-pooling operations). Somewhat surprisingly, the model of BiLSTMs seems not good for this sentence-pair modeling task, the reason might be that using the last states of LSTMs ignores too much information of the

Competitive System	COMP.	CONT.	EXP.+	TEMP.	AVG.
Pitler et al. (2009)	21.96	47.13	76.42	16.76	40.57
Zhou et al. (2010)	31.79	47.16	70.11	20.30	40.32
Park and Cardie (2012)	31.32	49.82	79.22	26.57	46.73
McKeown and Biran (2013)	25.40	46.94	75.87	20.23	42.11
R&Xue (2014)	39.70	54.42	80.44	28.69	50.81
Ji and Eisenstein (2015)	35.93	52.78	80.02	27.63	49.09
Braud (2015)	36.36	**55.76**	61.76	29.30	45.80
Zhang et al.(2015a)	33.22	52.04	-	30.54	-
Chen et al. (2016)	**40.17**	54.76	80.62	31.32	51.72
Char+Word-Enhanced	38.67	54.91	**80.66**	**32.76**	**51.75**

Table 3: Comparisons of F_1 scores (%) for binary classification. (symbol **+** means EXP. with *Entrel*)

previous words of the sequence. Thus, for the rest models, CNN will be selected to compute sentence vectors. In the third group, we will explore how the character-based embeddings will perform without conventional word-level embeddings. The character-based embeddings learned from **Char-CNN** model are individually calculated and lacks of the information of surrounding words, thus stacking BiLSTMs improves the accuracies because the recurrent layer could effectively capture rich context characteristics. Not surprisingly, utilizing only character-level embedding performs not that good, even the simple adding-word-vectors method gives better accuracies. This suggests that conventional word-level embeddings should not be abandoned because a word is only meaningful at the word-level. However, the character forming of a word could be also helpful, especially when we are dealing with rare words. Thus in the fourth group, we will explore the combination of character-level and word-level representations. The **Char+Word-Concat** model that concatenates the sentence vectors (from different CNNs) indeed improves the performance. The proposed model, **Char+Word-Enhanced**, combines the two modules at the word representation level, this is different from the **Concat** model because the influence of character-level representations are directly integrated into the final word representations before fed to CNN. The proposed model outperforms all the others, which shows the character-based representations do make extra helps.

4.2 Binary Classification

In order to compare with some previous work, we run our model on the binary implicit relation classification task. The dataset is also from PDTB and conventional splitting for binary classification is followed: Section 2-20 for training, 0-1 for development and Section 21-22 for testing. For the training set, since the number of negative examples is much greater than the number of positive examples, extra negative examples are extracted randomly to provide balanced training set. All examples in sections 21 and 22 are included for testing. Following previous work, the evaluation metric for binary classification will be Macro-F1 score. The hyper-parameters of our model are roughly the same as in multi-class classification expect that learning rate is set to 0.0002 for the binary classification task.

4.3 Results

As shown in Table 1, 2 and 3, the proposed model **Char+Word-Enhanced** outperforms most of the previous models, both for multi-class and binary classification task. This shows the effectiveness of context-aware character-enhanced embeddings and that these enhanced embeddings corporate well with sentence-level neural models.

5 Conclusion

In this paper, we propose a character-level neural module to obtain context-aware character-based embeddings for implicit discourse relation recognition. Utilizing the combined character-enhanced embeddings, our model performs well, which shows that the character-level information captured by the

proposed model may effectively improve this semantic understanding task.

References

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from RST discourse parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2212–2218, Lisbon, Portugal, November.

Or Biran and Kathleen McKeown. 2013. Aggregated word pair features for implicit discourse relation disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 69–73, Sofia, Bulgaria, August.

Chloé Braud and Pascal Denis. 2015. Comparing word representations for implicit discourse relation classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2201–2211, Lisbon, Portugal, September.

Deng Cai and Hai Zhao. 2016. Neural Word Segmentation Learning for Chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 409–420, Berlin, Germany, August.

Joyce Y Chai and Rong Jin. 2004. Discourse structure for context question answering. In *Proceedings of the Workshop on Pragmatics of Question Answering at North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL)*, volume 2004, pages 23–30, San Diego, USA.

Changge Chen, Peilu Wang, and Hai Zhao. 2015. Shallow discourse parsing using constituent parsing tree. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task (CONLL)*, pages 37–41, Beijing, China, July.

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Implicit discourse relation detection via a deep architecture with gated relevance network. In *Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany, August.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Robert Fisher and Reid Simmons. 2015. Spectral semi-supervised discourse relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 89–93, Beijing, China, July.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks (ICANN)*, pages 799–804, Warsaw, Poland.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278, Olomouc, Czech Republic.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2014. Using discourse structure improves machine translation evaluation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 687–698, Baltimore, Maryland, June.

Christian Hardmeier. 2012. Discourse in statistical machine translation: A survey and a case study. *Discours-Revue de linguistique, psycholinguistique et informatique*, 11.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric P Xing. 2016a. Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2410–2420, Berlin, Germany, August.

Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric P Xing. 2016b. Deep neural networks with massive learned knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Austin, USA, November.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *Transactions of the Association for Computational Linguistics (TACL)*, 3:329–344.

Yangfeng Ji, Gongbo Zhang, and Jacob Eisenstein. 2015. Closing the gap: Domain adaptation from explicit to implicit discourse relations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2219–2224, Lisbon, Portugal, September.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse-driven language models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 332–342, San Diego, California, June.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*, pages 2741–2749, Phoenix, USA.

Junyi Jessy Li and Ani Nenkova. 2014. Reducing sparsity improves the recognition of implicit discourse relations. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 199–207, Philadelphia, USA.

Zhongyi Li, Hai Zhao, Chenxi Pang, Lili Wang, and Huan Wang. 2016. A constituent syntactic parse tree based discourse parser. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task (CONLL)*, pages 60–64, Berlin, Germany, August.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 343–351, Suntec,Singapore.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(02):151–184.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1520–1530, Lisbon, Portugal, September.

Sameer Maskey and Julia Hirschberg. 2005. Comparing lexical, acoustic/prosodic, structural and discourse features for speech summarization. In *the 9th biennial conference of the International Speech Communication Association (ISCA) and the 6th in the annual series of INTERSPEECH events (INTERSPEECH 2005-EUROSPEECH)*, pages 621–624, Lisbon, Portugal.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems (3)*, pages 3111–3119, South Lake Tahoe, Nevada, USA, December.

Gabriel Murray, Steve Renals, Jean Carletta, and Johanna Moore. 2006. Incorporating speaker and discourse features into speech summarization. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (NAACL)*, pages 367–374, New York, USA.

Joonsuk Park and Claire Cardie. 2012. Improving implicit discourse relation recognition through feature set optimization. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 108–112, Seoul, South Korea, July.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of he Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 683–691, Suntec, Singapore, August.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The Penn Discourse TreeBank 2.0. In *The 6th edition of the Language Resources and Evaluation Conference (LREC)*, pages 2961–2968, Marrakech, Morocco.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016a. Shallow discourse parsing using convolutional neural network. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task (CONLL)*, pages 70–77, Berlin, Germany, August.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016b. A stacking gated neural architecture for implicit discourse relation classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Austin, USA, November.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 645–654, Gothenburg, Sweden, April.

Attapol Rutherford and Nianwen Xue. 2015. Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL): Human Language Technologies*, pages 799–808, Denver, Colorado, May–June.

Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2007. Evaluating discourse-based answer extraction for why-question answering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 735–736, Amsterdam, Holland, July.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. Word embedding for recurrent neural network based TTS synthesis. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4879–4883, Brisbane, Australia.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2016. Learning distributed word representations for bidirectional LSTM recurrent neural network. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 527–533, San Diego, California, June.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi Prasad, Christopher Bryant, and Attapol Rutherford. 2015. The CoNLL-2015 shared task on shallow discourse parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task (CONLL)*, pages 1–16, Beijing, China, July.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Bonnie Webber, Attapol Rutherford, Chuan Wang, and Hongmin Wang. 2016. The CoNLL-2016 shared task on shallow discourse parsing. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning - Shared Task (CONLL)*, pages 1–19, Berlin, Germany, August.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015a. Shallow convolutional neural network for implicit discourse relation recognition. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2230–2235, Lisbon, Portugal, September.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015b. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NIPS)*, pages 649–657, Montral, Quebec, Canada.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1382–1392, Berlin, Germany, August.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pages 1507–1514, Beijing, China, August.

Measuring Non-cooperation in Dialogue

Brian Plüss
Knowledge Media Institute
The Open University
Milton Keynes, UK
brian.pluss@open.ac.uk

Paul Piwek
School of Computing and Communications
The Open University
Milton Keynes, UK
paul.piwek@open.ac.uk

Abstract

This paper introduces a novel method for measuring non-cooperation in dialogue. The key idea is that linguistic non-cooperation can be measured in terms of the extent to which dialogue participants deviate from conventions regarding the proper introduction and discharging of conversational obligations (e.g., the obligation to respond to a question). Previous work on non-cooperation has focused mainly on non-linguistic task-related non-cooperation or modelled non-cooperation in terms of special rules describing non-cooperative behaviours. In contrast, we start from rules for normal/correct dialogue behaviour – i.e., a dialogue game – which in principle can be derived from a corpus of cooperative dialogues, and provide a quantitative measure for the degree to which participants comply with these rules. We evaluated the model on a corpus of political interviews, with encouraging results. The model predicts accurately the degree of cooperation for one of the two dialogue game roles (interviewer) and also the relative cooperation for both roles (i.e., which interlocutor in the conversation was most cooperative). Being able to measure cooperation has applications in many areas from the analysis – manual, semi and fully automatic – of natural language interactions to human-like virtual personal assistants, tutoring agents, sophisticated dialogue systems, and role-playing virtual humans.

1 Introduction

This paper describes a general method for measuring the degree of cooperation of dialogue participants' behaviour. Central to the method is the idea, following Traum (1994) and Matheson et al. (2000), that in dialogue obligations are continually created and resolved. Our contribution is a proposal for measuring non-cooperation in terms of the degree to which dialogue participants deviate from the obligations that they acquire during the course of the dialogue. We focus on an application of the proposed method to political interviews in order to evaluate its validity. We developed this method to extend the state-of-the-art of computational dialogue modelling to cases in which the conversational flow is compromised to some extent but without reaching complete breakdown. Shedding light on the nature of linguistic non-cooperation in dialogue promises to yield a better understanding of conversation. The method can be used for the analysis – manual, semi and fully automatic – of natural language interactions and for applications such as human-like virtual personal assistants, tutoring agents, sophisticated dialogue systems, and role-playing virtual humans.

In the remainder of this paper, we proceed as follows. In Section 2, we look at recent research in computational modelling of non-cooperative dialogue. We highlight the similarities and differences with the approach proposed in this paper. The next two sections then describe the two principal steps of our method. In Section 3, we introduce the first step. This step consists of segmentation of dialogue transcripts and coding of the speakers' contributions. We describe the segmentation and annotation schemes, and report on their reliability. In this step, the individual annotations are neutral with regards to cooperation. Section 4 introduces a fully automated method for combining the annotations from the first step with a model of the dialogue game (specific to the dialogue genre in question). The result of

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1925–1936, Osaka, Japan, December 11-17 2016.

this automatic analysis is a dialogue marked up with cooperative and non-cooperative features. These features lead to a score for each speaker that indicates the extent to which they behaved according to the obligations associated with their role in the dialogue, which we interpret as the degree of cooperation of the participant with respect to the conversational setting. The dialogue game model, in this case for political interviews, is extracted from descriptive accounts in the linguistics literature of the dialogue genre. Next, in Section 5, the validity of the method is assessed by analysing the correlation between the resulting scores and human judgement on the same set of political interview transcripts. Finally, Section 6 presents our conclusions and some suggestions for further work.

2 Related Work on Computation and Annotation of Non-cooperation in Dialogue

Possibly the earliest computational model of non-cooperation is presented by Jameson (1989). It includes an extensive study for modelling bias, individual goals, projected image and belief ascription in conversation. Jameson implemented some of these ideas, in the context of used car sales, by means of a dialogue system that can assume different roles (Jameson et al., 1994). These contributions show that user-model approaches to dialogue modelling are flexible enough to account for situations of an arbitrary degree of intricacy. However, as noted, e.g., by Taylor et al. (1996) the level of detail required in the characterisation of the user and the complexity of mechanism for reasoning about user models can lead to problems like infinite regress in nested beliefs (speaker's beliefs about the hearer's beliefs about the speaker's beliefs...).

More recently, Traum (2008) brought attention to the need for computational accounts of dialogue situations in which a broader notion of cooperation is not assumed. Traum's work on non-cooperative dialogue is mainly aimed at creating virtual humans – or embodied conversational agents (Cassell, 2001) – with abilities to engage in adversarial dialogue. Traum et al. (2005; 2008) present a model of conversation strategies for negotiation, implemented as a virtual human that can be used for teaching negotiation skills. A recent version of the system (Plüss et al., 2011; Traum, 2012) supports cooperative, neutral and deceptive behaviour, and also is able to reason in terms of secrecy in order to avoid volunteering certain pieces of information. However, their model the adversarial scenarios by means of a set of rules that the interlocutors follow. Our approach contrasts with this in that it models non-cooperation in terms of systematic deviation from the rules of the dialogue game.

Along lines similar to Traum et al., the work of Kreutel and Matheson (2001; 2003) accounts for non-cooperative behaviour at the level of the task, what the authors call *strategic acting*. At the conversational level, however, their models – as well as those of Traum and Allen (1994) and Matheson et al. (2000) – always discharge a speaker's obligations before considering their private goals. This also holds for the recent work on learning non-cooperative dialogue behaviours using statistical methods (Efstathiou and Lemon, 2014): conversational or linguistic cooperation is assumed (i.e., dialogue participants honour their discourse obligations), whereas non-linguistically, participants fail to cooperate. The method we describe in this paper is complementary to this work in that we aim to characterise, analyse and measure *conversational/linguistic* non-cooperation.

Previous research on dialogue annotation for non-cooperation is scarce. The only instances of complete research we know of are those of Davies (1997; 2006) and Cavicchio (2010) – see also Cavicchio and Poesio (2012).[1] Both are in the context of task-oriented dialogues, and more specifically the HCRC Map Task domain (Anderson et al., 1991; Carletta et al., 1997).

Davies (1994; 1997; 2006) proposes a direct approach to annotating cooperation in order to analyse its relation with effort and task success. Her annotation approach shares some characteristics with ours, but cooperation is judged directly by the annotators, as "positive codings (i.e., finding an instance of the behaviour in an utterance), and negative codings (i.e., finding an instance where we believe a particular behaviour should have been used)" (Davies, 2006, p. 43). In her doctoral thesis, Cavicchio (2010) applies Davies's coding scheme to a multi-modal corpus of the Map Task domain and studies the relation

[1] Additionally, two short papers by Asher et al. (2012) and Afantenos et al. (2012) report on ongoing data collection and preliminary annotation of negotiation dialogues surrounding a board game, following a theory of strategic conversation proposed by Asher and Lascarides (2013).

between (non-)cooperation and emotions. Her focus is not however on how to assess cooperation in dialogue, but on to what extent psychophysiological indicators of emotion (e.g., heartrate and facial expressions) correlate with cooperative behaviour.

The key difference between Davies's and our approach is that the former already includes the normative notion of dialogue game we use later in the assessment of cooperation. This reduces the flexibility of the coding scheme, as the assessment of cooperation is part of the annotation process. By detaching these steps, the method proposed here allows for assessment of cooperation of the same annotated data using different dialogue games, e.g., to explore how the same behaviour would be perceived by audiences with different cultural backgrounds.

3 Corpus Annotation

The degree of cooperation of dialogue participants is determined in two steps. The input for the process is a dialogue transcript. In the first step, this transcript is manually segmented and annotated. In this manual step, the annotators are *not* required to make any judgements about the cooperation of the interlocutors. The actual determination of the extent of cooperation takes places in the second fully automated step.

In this section, we describe the first step by briefly introducing the annotation schemes and providing our results on their reliability. The complete annotation guidelines, tool, and fully annotated corpus are available online.[2]

3.1 The Corpus

In order to test our approach, we applied it to a corpus of six political interviews with a total of 88 turns (3556 words). The number of turns and words in each fragment is shown in Table 1.

Table 1: Political interview fragments in the corpus annotation study

Interview	Turns	Words
1. Brodie and Blair	16	734
2. Green and Miliband	9	526
3. O'Reilly and Hartman	19	360
4. Paxman and Osborne	16	272
5. Pym and Osborne	10	595
6. Shaw and Thatcher	18	1069
Total	88	3556

The fragments were selected from a larger set of 15 interviews collected from publicly available sources (BBC News, CNN, Youtube, etc.). We selected this particular set with the aim of including behaviours at different levels of cooperation for both interviewer and interviewee role. At the same time, we avoided extreme cases in which the exchange broke down or turned into a dialogue of an entirely different type (e.g., confrontation or debate). A second criterion was to ensure coverage of the annotation scheme, with special attention to the dialogue act taxonomy.

3.2 Segmentation and Dialogue Act Annotation

We followed the recommendations put forward in the ISO standard proposal by Bunt et al. (2009; 2010; 2012), simplifying the terminology and some aspects of the scheme when needed. For this we drew on work by Carletta et al. (1997), Allen and Core's (1997) DAMSL, Traum and Hinkelman's (1992) Conversation Acts theory – following Poesio and Traum (1997; 1998) and proposed as a standard by the Discourse Resource Initiative (Initiative, 1997) –, and Stoyanchev and Piwek (2010a; 2010b). We consider two main classes of functions for dialogue acts: Initiating and Responsive. Initiating dialogue acts are primarily meant to provoke a response by the other speaker as opposed to being themselves responses to previous dialogue acts. Responsive dialogue acts are mainly reactions of the speaker to a previous (initiating or responsive) action of the other party. These are distinguished by the prefixes **Init** and **Resp** in Table 3. Initiating dialogue acts are further divided into information giving and information

[2] At `http://mcs.open.ac.uk/nlg/non-cooperation/`.

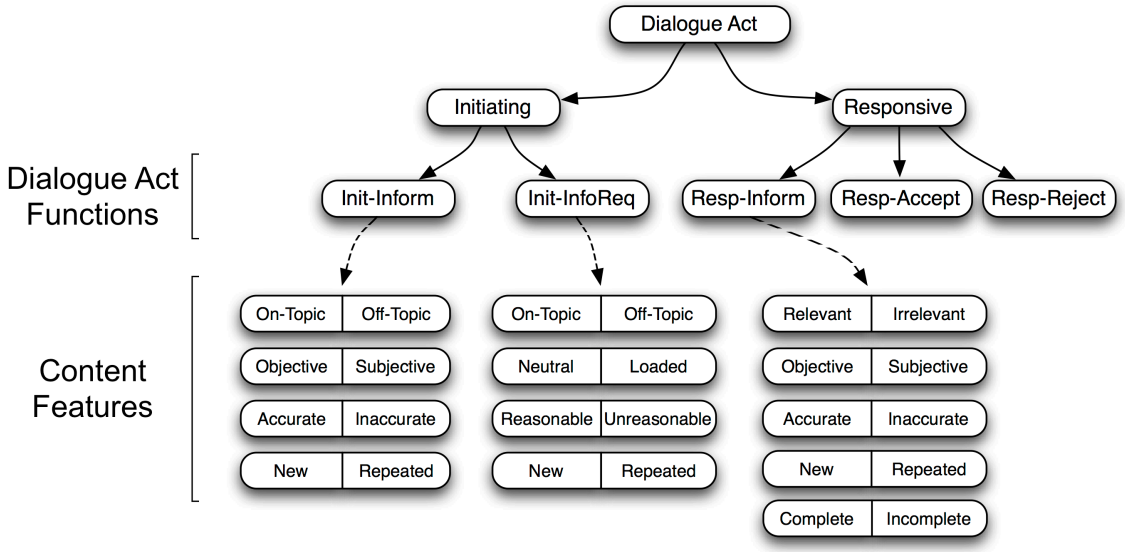


Figure 1: Annotation scheme for dialogue act functions and content features

requesting dialogue acts (**Init-Inform** and **Init-InfoReq**, respectively). Responsive dialogue acts are further divided into information giving, accepting and rejecting dialogue acts (**Resp-Inform**, **Resp-Accept**, and **Resp-Reject**, respectively). The entire annotation scheme, including dialogue act functions and content features, is shown in Figure 1.

For the segmentation and dialogue act annotation stage, four annotators (one of the authors and three native English-speaking researchers with previous experience in dialogue annotation) received transcripts of the corpus and were asked to segment the turns in each dialogue and to annotate each segment with dialogue act functions and, when applicable, with referent segments (i.e., a segment in a previous turn of the other speaker to which the current segment responds). A *segment* is defined as a stretch of a turn that can be labelled with a single dialogue act function. Stretches of a turn can belong to only one segment - i.e., segments do no overlap - and some stretches can remain unannotated. The instructions for segmenting and dialogue act functions for each turn in a dialogue are summarised as follows:

1. Segment the turn by selecting the stretches of speech that have a clear dialogue act function.

2. Assign a dialogue act function to each segment, identifying whether the dialogue act is initiating an exchange (i.e., requesting information, giving information as context for an upcoming question, etc.), or responding to a previous dialogue act (i.e., accepting a question or an answer, answering a question, rejecting a premise, providing additional information, etc.).

3. For each responsive segment, select the segment that caused the response.

Furthermore, when choosing the stretches of a turn that constitute separate segments two criteria are followed: (*a*) the stretch has to be of a length such that it can be assigned one of the available dialogue act functions, and (*b*) its contents have to request for or convey a clearly identifiable, ideally unique piece of information, or several pieces of information on the same topic.

We measured inter-annotator agreement for segmentation using Krippendor's α_U coefficient (Krippendorff, 1995), which was adapted for segmentation of transcribed dialogue. In general, agreement for segmentation, see Table 2, is high, i.e., "substantial", in terms of Landis and Koch (1977). Consistent with intuition, disagreement is greater in dialogues with longer turns.

Annotators independently segmented the turns and selected dialogue act functions for these segments in the same annotation step. This means that the units for annotation identified by one coder can differ from those identified by another coder. These differences make it possible to analyse the reliability of

Table 2: Inter-annotator agreement for segmentation (Krippendorff's α_U)

Interview	α_U	D_o	D_e
1. Brodie and Blair	0.802	3.217	16.251
2. Green and Miliband	0.618	3.276	8.565
3. O'Reilly and Hartman	0.773	4.138	18.219
4. Paxman and Osborne	0.92	0.993	12.468
5. Pym and Osborne	0.672	4.0	12.184
6. Shaw and Thatcher	0.653	7.951	22.890
Overall	0.74	23.574	90.577

Table 3: Inter-annotator agreement for dialogue act functions (Krippendorff's α)

Label	α	D_o	D_e
Init-Inform	0.409	0.040	0.068
Init-InfoReq	0.893	0.009	0.089
Resp-Inform	0.645	0.038	0.107
Resp-Accept	0.606	0.011	0.029
Resp-Reject	0.635	0.018	0.050
Overall	0.657	0.059	0.171

the original annotation data only in terms of Krippendorff's α,[3] which supports missing annotations for some of the items. The value of this coefficient for each label (i.e., regarding the rest of the categories as **Other**) and for entire dialogue act taxonomy is given in Table 3. Agreement ranges from "moderate" to "perfect", with overall agreement being "substantial".

Finally, for responsive dialogue acts, we also asked annotators to indicate which dialogue segment they were a response to. Inter-annotator agreement for referent segment annotations is "substantial" at $\alpha = 0.732$ and $(D_o, D_e) = (0.038, 0.141)$.

3.3 Content Feature Selection

For the second stage, we identified a set of dimensions on which the content of a contribution is judged (see Figure 1). These are based, in part, on Bull and Mayer's (1993) and Bull's (1994; 2003) extensive work on the micro-analysis of equivocation in political discourse.

Annotations from the previous stage were automatically aggregated to produce a single segmented and partially annotated version of each dialogue. These were used in the second stage of the study in which seven annotators (the four coders that took part in the first stage, plus another linguistic expert, with near native English, and two native English speakers with no background in linguistics or experience in dialogue analysis) were asked to select content features.

When judging the content of a segment, annotators had to consider – to the best of their knowledge – several elements of the context of the conversation (e.g., topical, political, historical), as well as common sense, world knowledge, etc. They also had to take into account previous contributions of both participants, and in some cases contributions made later on in the dialogue. Every time annotators made a judgement, they were instructed to ask themselves the following question: 'Do I have any evidence to make this choice?' If the answer was 'Yes', they could go ahead with their choice. Otherwise, they had to be *charitable*. This means that, for instance, if it is not possible to determine whether the information provided in a segment was accurate or not, the first option was chosen. Similarly, if whether a question is reasonable or not cannot be decided, then it is considered reasonable.

Table 4 shows the values of agreement for Krippendorff's α, observed and expected disagreement, observed (or average) agreement A_o, and multi-rater versions of Cohen's κ and Scott's π (or Siegel and Castellan's K) with their respective expected agreements A_e – observed agreement is the same for both coefficients and as given under A_o.[4] We report on agreement for the content features individually, aggregated for each dialogue act function, and overall for the entire corpus. Overall agreement is moderate ($\alpha = 0.454$).

[3]Krippendorff's α is a family of reliability coefficients (Krippendorff, 2003, Chapter 11) defined in terms of the ratio between the disagreement observed among the coders and the disagreement expected by chance: $\alpha = 1 - \frac{D_o}{D_e}$, where D_o and D_e are, respectively, the observed and expected disagreements.

[4]In addition to Krippendorff's α, we report reliability of the annotation of content features using multi-rater versions of Cohen's κ (Cohen, 1960; Davies and Fleiss, 1982) and Scott's π (Scott, 1955; Fleiss, 1971) – called K by Siegel and Castellan (1988). This is because these measures are often found in the literature when discussing the results of dialogue annotation exercises. The general form for both coefficients is: $\pi, \kappa = \frac{A_o - A_e}{1 - A_e}$, where A_o and A_e are, respectively, the observed – or average – agreement and the agreement expected by chance. The observed agreement A_o is the same for both coefficients and equal to the ratio between the number of instances in which any two annotators agreed in the classification of an item and the total number of pairs of annotations of each item. See discussions by Artstein and Poesio (2008) and Plüss (2014, Chapter 4) on the applications of these coefficients to studies in computational linguistics.

Table 4: Inter-annotator agreement for content features

Content Feature	$\alpha\ (D_o, D_e)$	A_o	$\kappa\ (A_e)$	$\pi \mid K\ (A_e)$
Init-Inform	0.398 (0.137, 0.227)	0.863	0.402 (0.772)	0.393 (0.775)
On-Topic | Off-Topic	0.079 (0.100, 0.109)	0.900	0.083 (0.891)	0.072 (0.892)
Objective | Subjective	0.370 (0.305, 0.483)	0.695	0.377 (0.510)	0.365 (0.520)
Accurate | Inaccurate	0.467 (0.090, 0.170)	0.910	0.467 (0.830)	0.463 (0.832)
New | Repeated	0.641 (0.052, 0.146)	0.948	0.640 (0.855)	0.638 (0.855)
Init-InfoReq	0.563 (0.081, 0.185)	0.919	0.564 (0.814)	0.560 (0.816)
On-Topic | Off-Topic	0.104 (0.022, 0.025)	0.978	0.105 (0.975)	0.100 (0.975)
Neutral | Loaded	0.481 (0.213, 0.410)	0.787	0.486 (0.586)	0.478 (0.592)
Reasonable | Unreasonable	0.514 (0.050, 0.104)	0.950	0.512 (0.897)	0.512 (0.897)
New | Repeated	0.806 (0.039, 0.202)	0.961	0.805 (0.799)	0.805 (0.799)
Resp-Inform	0.438 (0.198, 0.352)	0.802	0.443 (0.645)	0.436 (0.649)
Relevant | Irrelevant	0.407 (0.228, 0.385)	0.772	0.411 (0.613)	0.405 (0.616)
Objective | Subjective	0.316 (0.338, 0.494)	0.662	0.333 (0.493)	0.314 (0.507)
Accurate | Inaccurate	−0.014 (0.032, 0.032)	0.968	−0.014 (0.968)	−0.016 (0.968)
New | Repeated	0.763 (0.083, 0.348)	0.917	0.762 (0.652)	0.762 (0.653)
Complete | Incomplete	0.383 (0.309, 0.501)	0.691	0.385 (0.498)	0.382 (0.500)
Overall	0.454 (0.143, 0.262)	0.857	0.458 (0.736)	0.452 (0.739)

4 Computing Cooperation

4.1 From Annotations to Actions Labels

As a first step, the dialogue act functions and content features in the annotations are mapped to *action labels*. The rules of a dialogue game are formulated in terms of the actions that participants perform during a conversation. These actions are represented as labels that capture those aspects of the speakers' contributions that are necessary for applying the rules.

The mapping, see Table 5, is carried out automatically, based on rules that are tailored to a specific dialogue game and coding scheme pair. This approach allows for a separation between the prescriptive nature of the dialogue game and the descriptive character of the coding scheme. Such independence facilitates, for instance, changing the rules of the dialogue game so that it better relates to the social norms, conventions and expectations of different cultural backgrounds, while keeping the coding scheme unchanged and using the same annotated data. It is worth noting that this mapping is independent of the set of interviews in the corpus and, like the dialogue game, was devised based on the linguistics literature for political interviews (Bull and Mayer, 1993; Bull, 1994; Heritage, 1998; Clayman and Heritage, 2002; Heritage, 2005). Also, given the formalisation of the dialogue game (see Figure 2 below), the application of the rules for mapping annotated dialogue into action labels is straightforward.

Table 5: Mapping annotations to action labels in political interviews

Annotation Scheme: Dialogue Act	Content Features	Dialogue Game: Action Label
Init-Inform	**On-Topic** *and* **Objective** *and* **Accurate** *and* **New**	`valid-statement`
Init-Inform	**Off-Topic** *or* **Subjective** *or* **Inaccurate** *or* **Repeated**	`invalid-statement`
Init-InfoReq	**On-Topic** *and* **Neutral** *and* **Reasonable**	`valid-question`
Init-InfoReq	**Off-Topic** *or* **Loaded** *or* **Unreasonable**	`invalid-question`
Resp-Inform	*Any*	`invalid-reply`
Resp-Accept		`acceptance`
Resp-Reject		`rejection`

(a) Interviewer segments

Annotation Scheme: Dialogue Act	Content Features	Dialogue Game: Action Label
Init-Inform	*Any*	`invalid-statement`[a]
Init-Inform	**On-Topic** *and* **Accurate** *and* **New**	`valid-statement`[b]
Init-Inform	**Off-Topic** *or* **Inaccurate** *or* **Repeated**	`invalid-statement`[b]
Init-InfoReq	*Any*	`invalid-question`
Resp-Inform	**Relevant** *and* **Accurate** *and* **New**	`valid-reply`
Resp-Inform	**Irrelevant** *or* **Inaccurate** *or* **Repeated**	`invalid-reply`
Resp-Accept		`acceptance`
Resp-Reject		`rejection`

(b) Interviewee segments

[a] If the interview starts with a question by the interviewer.
[b] In the first turn of an interview that starts with a statement by the interviewee.

4.2 Cooperative and Non-Cooperative Feature Computation

Linguistic cooperation of a dialogue participant with respect to a conversational setting equates to the participant following the rules of the dialogue game for that conversational setting. Figure 2 shows the dialogue game of political interviews that we used for the current study, derived from the descriptive accounts in the linguistics literature (Heritage, 1998; Clayman and Heritage, 2002; Heritage, 2005). Each turn in a dialogue is associated with an amount of cooperation and an amount of non-cooperation. These are given by the number of dialogue rules that the turn, respectively, conformes with and violates. The instances in which rules are conformed with are called *cooperative features* and those in which rules are broken are called *non-cooperative features*.

Participants can break the rules of the game in two ways: (*a*) by performing a conversational action that is not allowed for their role and (*b*) by failing to perform an action they were obliged to perform. Instances of (*a*) are violations of static obligations, which we call *static non-cooperative features*. Instances of (*b*) are violations of dynamic obligations, which we call *dynamic non-cooperative features*. An analogous distinction is made for cooperative features, called, respectively, *static cooperative features* and *dynamic cooperative features*. The *degree of cooperation* of each dialogue participant is thus the ratio between the number of cooperative features – static and dynamic – and the total number of features of that participant. In general, this value can be obtained for the entire conversation and for any continuous fragments. The complete algorithms for computing these features, given an annotated transcript and dialogue game, is available online.[5]

In each turn, we check whether the actions performed by the speaker are allowed for his or her role as specified in the dialogue game. If an action is in the the speaker's set of allowed actions, then it constitutes a static cooperative feature, otherwise it becomes a static non-cooperative feature.

In each turn, we look at the speaker's obligations pending after and discharged in that turn. If an obligation on the speaker has been discharged within the turn, then it constitutes a dynamic cooperative feature, otherwise it becomes a dynamic non-cooperative feature.

Once we have computed the static and dynamic features for each turn, we can regard the proportion of these that are cooperative as an indicator of the extent to which each participant acted within the rules of the game. This is the *degree of cooperation* of a dialogue participant with respect to a dialogue game. Formally, for speaker s and dialogue $D = \langle t_1; \ldots; t_n \rangle$ this is:

$$dc_{D,s} = \frac{cf_{D,s}}{cf_{D,s} + ncf_{D,s}}$$

where $cf_{D,s}$ is the number of cooperative features – both static and dynamic – of participant s and $ncf_{D,s}$ is the analogous for non-cooperative features. This is[6]:

$$cf_{D,s} = \sum_{\substack{i=1 \\ [s_i=s]}}^{n} |sf_i(2)| + |df_i(2)| \qquad ncf_{D,s} = \sum_{\substack{i=1 \\ [s_i=s]}}^{n} |sf_i(3)| + |df_i(3)|$$

Note that, although these definitions are here expressed for the complete dialogue, the same applies to any contiguous subsequences of turns.

The *degree of non-cooperation* of a dialogue participant s in dialogue D is: $dnc_{D,s} = 1 - dc_{D,s}$.

5 Evaluation

We obtained judgements on the behaviour of participants in the political interviews in the corpus by means of an online survey constructed using SurveyMonkey.[7] Observers were shown transcripts of the dialogues and asked to rate the performance of the participants on a 5-point scale (from Incorrect to Correct), based on their intuitions on how interviewers and politicians *ought* to behave.[8]

[5] See `http://mcs.open.ac.uk/nlg/non-cooperation/`.

[6] The elements in the sequences of both static and dynamic features $SF_D = \langle sf_1; \ldots; sf_n \rangle$ and $DF_D = \langle df_1; \ldots; df_n \rangle$ are triples (s_i, C_i, NC_i), where s_i is the speaker in turn t_i, and C_i and NC_i are the associated sequences of, respectively, cooperative and non-cooperative features.

[7] `http://www.surveymonkey.com`

[8] The complete survey is available online at `http://mcs.open.ac.uk/nlg/non-cooperation/`.

$$G_{PI} = (Allow_{PI}, Introduce_{PI}, Discharge_{PI})$$

where

$$\begin{aligned}
Allow_{PI} = \{&[\texttt{ir} : \{\texttt{valid-statement}, \texttt{valid-question}, \texttt{acceptance}, \texttt{rejection}\}], && (1)\\
&[\texttt{ie} : \{\texttt{valid-statement}, \texttt{valid-reply}, \texttt{acceptance}, \texttt{rejection}\}]\} && (2)\\
Introduce_{PI} = \{&[(\texttt{ir}, (\texttt{s}) : \texttt{valid-statement}) \rightsquigarrow (\texttt{ie}, \texttt{acceptance@(s)})], && (3)\\
&[(\texttt{ir}, (\texttt{q}) : \texttt{valid-question}\ \mathbf{N}) \rightsquigarrow (\texttt{ie}, \texttt{acceptance@(q)})], && (4)\\
&[(\texttt{ie}, \texttt{acceptance@(q)}) \rightsquigarrow (\texttt{ie}, \texttt{valid-reply@(q)}\ \mathbf{C})], && (5)\\
&[(\texttt{ie}, (\texttt{s}) : \texttt{valid-statement}) \rightsquigarrow (\texttt{ir}, \texttt{acceptance@(s)})], && (6)\\
&[(\texttt{ie}, (\texttt{r}) : \texttt{valid-reply@(q)}) \rightsquigarrow (\texttt{ir}, \texttt{acceptance@(r)})], && (7)\\
&[(\texttt{ir}, \texttt{acceptance}) \rightsquigarrow (\texttt{ir}, \texttt{valid-question}\ \mathbf{N})], && (8)\\
&[(\texttt{ir}, (\texttt{s}) : \texttt{invalid-statement}) \rightsquigarrow (\texttt{ie}, \texttt{rejection@(s)})], && (9)\\
&[(\texttt{ir}, (\texttt{q}) : \texttt{invalid-question}) \rightsquigarrow (\texttt{ie}, \texttt{rejection@(q)})], && (10)\\
&[(\texttt{ir}, (\texttt{r}) : \texttt{invalid-reply}) \rightsquigarrow (\texttt{ie}, \texttt{rejection@(r)})], && (11)\\
&[(\texttt{ie}, (\texttt{s}) : \texttt{invalid-statement}) \rightsquigarrow (\texttt{ir}, \texttt{rejection@(s)})], && (12)\\
&[(\texttt{ie}, (\texttt{q}) : \texttt{invalid-question}) \rightsquigarrow (\texttt{ir}, \texttt{rejection@(q)})], && (13)\\
&[(\texttt{ie}, (\texttt{r}) : \texttt{invalid-reply}) \rightsquigarrow (\texttt{ir}, \texttt{rejection@(r)})]\} && (14)\\
Discharge_{PI} = \{&[*\text{-}\texttt{question}\ \mathbf{R} \succ \texttt{rejection}], && (15)\\
&[*\text{-}\texttt{statement} \succ \texttt{acceptance}], && (16)\\
&[*\text{-}\texttt{question}\ \mathbf{N} \succ \texttt{acceptance}], && (17)\\
&[*\text{-}\texttt{reply} \succ \texttt{acceptance}]\} && (18)
\end{aligned}$$

Figure 2: Dialogue game G_{PI} for political interviews, consisting of (*i*) $Allow_{PI}$, which specifies the actions allowed for the interviewer (ir) and interviewee (ie), respectively; (*ii*) $Introduce_{PI}$, which stipulates which actions by a specific participant give rise to obligations – e.g., (4) says that a new (**N**) valid question by the interviewer obliges the interviewee to accept that question and (5) says that after accepting a question an interviewee is obliged to provide a complete (**C**) valid reply; (*iii*) $Discharge_{PI}$ specifies how certain actions can count as other actions for (implicitly) discharging obligations – e.g., (15) says that repetition (**R**) of any questions counts as discharging the obligation for a rejection dialogue act and (16) says that a (valid or invalid) statement counts as discharging the obligation for an acceptance dialogue act; so, for instance, an obligation for acceptance by ie that has been created through application of rule (3), can be discharged by ie by producing a statement, in accordance with rule (16).

We used the six interviews in the corpus described above in Section 3.1. Judges (54 respondents in total) were shown the same context and transcript as the annotators.

We studied the relation between human judgement resulting from the survey and the degree of cooperation obtained from the method described above by means of a correlation analysis (see Figure 3).We carried out the correlation analysis separating interviewers from interviewees. The rationale for this step is that some of the rules of the dialogue game are role-specific, making the method strictly different for each participant in an interview. A similar argument applies to the way human observers are expected to judge the behaviour of interviewers and politicians. The two sets of six points are shown in Figure 4, with separate regression lines and values for Pearson's r. The results show that correlation is significantly better for interviewers ($r = 0.753$) than for interviewees ($r = 0.271$). Statistical significance is also stronger for interviewers ($p = 0.084$) indicating a trend towards positive correlation between the results of our method and human judgement. For the interviewees the correlation is not statistically significant ($p = 0.603$). With a sample of this size, correlation analysis is fairly sensitive to outliers, which could explain such a high p-value for the interviewees. Take, for instance, the interviewee in Interview 3 (O'Reilly and Hartman) which corresponds to the point furthest up from the regression line for interviewees (blue) in Figure 4. Coincidentally, Interview 3 has been described by one of the annotators as more like a debate than a political interview which could explain the unexpected value given by the method.

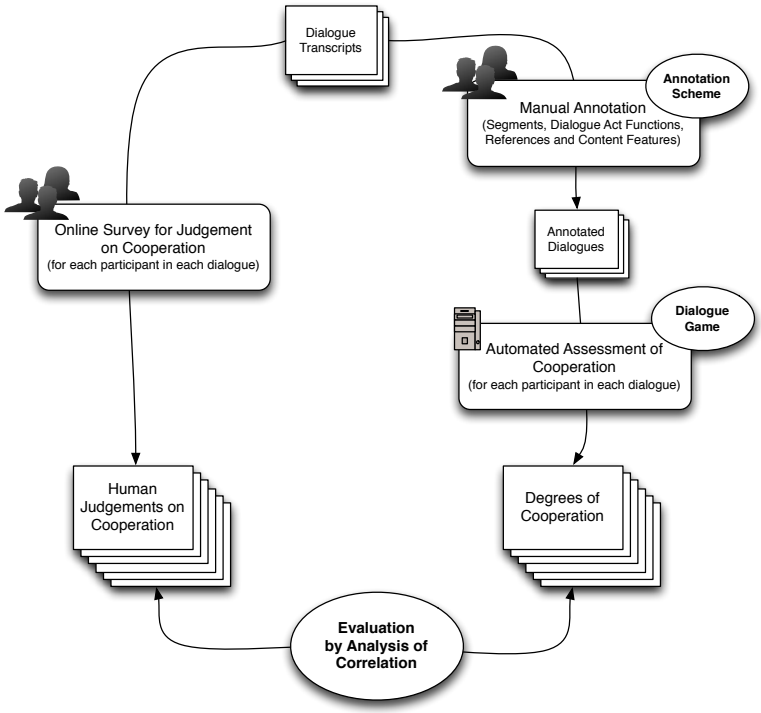


Figure 3: Evaluating the semi-automatic measure of cooperation via correlation with human judgement

6 Conclusions and Further Work

The method presented above is, to date and to our best knowledge, the most elaborate attempt at annotating and analysing naturally occurring dialogue in the light of linguistic cooperation. Also novel is the application of such an approach to a corpus of real political interviews, especially in that both speakers received the same amount of attention and that the method was subject to an extensive evaluation.

The results of the evaluation for reliability are encouraging and indicate that the method is suitable for the systematic analysis of non-cooperation. They also expose some of its weaknesses, such as the difficulties with applying some of the criteria in the manual annotation, a degree of vagueness in the definition of a few of the concepts and the inherent subjectivity of many of the judgements involved in properly characterising non-cooperation.

The evaluation of validity produced fairly good results, especially considering how little information was given to observers in the survey as to what was meant by linguistic cooperation and the total absence of a reference to the specific dialogue game adopted as part of the semi-automatic measure.

It is worth pointing out that the method, in its current form, was able to predict accurately in the six interviews of the corpus which of the participants behaved better with respect to their interlocutors. Beyond the correlation of the precise scores, the ability to determine this binary judgement without mistakes in all cases is of great interest and an indication of the adequacy of the approach.

It is unfortunate that the size of the sample in the corpus prevented from obtaining statistically significant results for each speaker role, particularly the interviewee. A larger sample, including more interview fragments would help in setting this right. Given the relative ease in collecting human judgements, the inclusion of new fragments should start with one or more surveys similar to the one described above. This would allow a decision on the choice of subset of interviews that offers the best coverage of the range of possible behaviours.

6.1 Further work

The method proposed in this paper can be extended to include further aspects of dialogue, like prosody, gestures and other multi-modal aspects of dialogue interaction, as well as sub-utterance elements such as interruptions, incomplete and overlapped speech, etc.

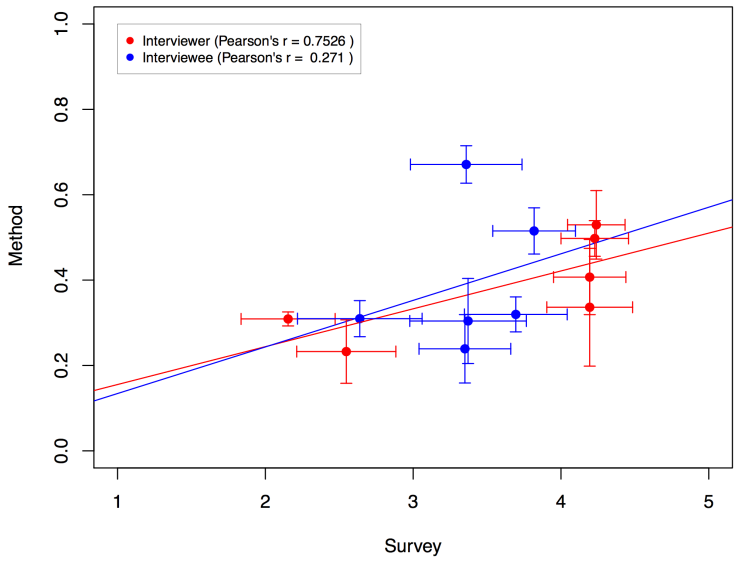


Figure 4: Survey results and the degree of cooperation for interviewers and interviewees (means with error bars, regression line and Pearson's r correlation coefficient)

A further line of work is towards full automation. Data-driven techniques using machine learning can be used to automatically annotate the dialogues with the labels needed to assess the degree of cooperation. Further, we speculate that the rules of the dialogue game could be learned from a sufficiently large corpus of interviews that are deemed conventional.

Our decoupling of the dialogue game from the annotations allows for further evaluation of the approach with participants from cultures with different conventions for political interviews (using the current corpus or a translation of it). Similarly, although the method has been described and evaluated in detail for political interviews, the approach is generally domain-independent. Applications to other conversational domains in which it is possible to identify a set of rules of expected interaction would allow further assessment of the approach. Such domains include courtroom interrogations, tutoring sessions, doctor-patient discussions, customer services, and many more.

References

Stergos Afantenos, Nicholas Asher, Farah Benamara, Anais Cadilhac, Cedric Dégremont, Pascal Denis, Markus Guhe, Simon Keizer, Alex Lascarides, Oliver Lemon, Philippe Muller, Soumya Paul, Vladimir Popescu, Verena Rieser, and Laure Vieu. 2012. Modelling Strategic Conversation: model, annotation design and corpus. In *Proceedings of SemDial 2012 (SeineDial), 16th Workshop on the Semantics and Pragmatics of Dialogue*, Paris, France, September.

James Allen and Mark Core. 1997. Draft of DAMSL: Dialog Act Markup in Several Layers. Technical report, University of Rochester.

Anne H. Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline C. Kowtko, Jan McAllister, Jim Miller, Catherine Sotillo, and Henry S. Thompson. 1991. The HCRC Map Task Corpus. *Language and speech*, 34(4):351–366.

Ron Artstein and Massimo Poesio. 2008. Inter-coder Agreement for Computational Liguistics. *Computational Linguistics*, 34(4):555–596.

Nicholas Asher and Alex Lascarides. 2013. Strategic conversation. *Semantics and Pragmatics*, 6(2):1–62, August.

Nicholas Asher, Alex Lascarides, Oliver Lemon, Markus Guhe, Verena Rieser, Philippe Muller, Stergos Afantenos, Farah Benamara, Laure Vieu, Pascal Denis, Soumya Paul, Simon Keizer, and Cedric Dégremont. 2012. Modelling Strategic Conversation: the STAC project. In *Proceedings of SemDial 2012 (SeineDial), 16th Workshop on the Semantics and Pragmatics of Dialogue*, Paris, France, September.

Peter Bull and K. Mayer. 1993. How not to answer questions in political interviews. *Political Psychology*, pages 651–666.

Peter Bull. 1994. On identifying questions, replies, and non-replies in political interviews. *Journal of Language and Social Psychology*, 13(2):115.

Peter Bull. 2003. *The Microanalysis of Political Communication: Claptrap and ambiguity*. Routledge.

Harry Bunt, Jan Alexandersson, Jean Carletta, Jae-Woong Choe, Alex Chengyu Fang, Koiti Hasida, Kiyong Lee, Volha Petukhova, Andrei Popescu-Belis, Laurent Romary, Claudia Soria, and David Traum. 2010. Towards an ISO Standard for Dialogue Act Annotation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA).

Harry Bunt, Jan Alexandersson, Jae-Woong Choe, Alex Chengyu Fang, Koiti Hasida, Volha Petukhova, Andrei Belis-Popescu, and David Traum. 2012. ISO 24617-2: A semantically-based standard for dialogue annotation. In *LREC 2012*, Istanbul, Turkey.

Harry Bunt. 2009. The DIT++ taxonomy for functional dialogue markup. In *AAMAS 2009 Workshop, Towards a Standard Markup Language for Embodied Dialogue Acts*, pages 13–24.

Jean Carletta, Stephen Isard, G. Doherty-Sneddon, Amy Isard, J. C. Kowtko, and A. H. Anderson. 1997. The reliability of a dialogue structure coding scheme. *Computational Linguistics*, 23(1):13–31.

Justine Cassell. 2001. Embodied Conversational Agents: Representation and Intelligence in User Interfaces. *AI Magazine*, 22(4):67–84.

F. Cavicchio and M. Poesio. 2012. (non)cooperative dialogues: the role of emotions. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 54:546–559.

Federica Cavicchio. 2010. *Computational Modeling of (un)Cooperation: The Role of Emotions*. Ph.D. thesis, University of Trento. CIMeC.

Steven Clayman and John Heritage. 2002. *The News Interview: Journalists and Public Figures on the Air*. Cambridge University Press.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurment*, 20(1):37–46.

Mark Davies and Joseph L. Fleiss. 1982. Measuring agreement for multinomial data. *Biometrics*, pages 1047–1051.

Bethan L. Davies. 1994. To cooperate or not to cooperate - is that the question? In *Proceedings of the Edinburgh Linguistics Department Conference*, pages 17–32. Citeseer.

Bethan L. Davies. 1997. *An Empirical Examination of Cooperation, Effort and Risk in Task-oriented Dialogue*. Ph.D. thesis, University of Edinburgh.

Bethan L. Davies. 2006. Testing dialogue principles in task-oriented dialogues: An exploration of cooperation, collaboration, effort and risk. *Leeds Working Papers in Linguistics and Phonetics*, 11:30–64.

Ioannis Efstathiou and Oliver Lemon. 2014. Learning non-cooperative dialogue behaviours. In *Proceedings of the SIGDIAL 2014 Conference*, pages 60–68, Philadelphia, U.S.A, 18–20 June. Association for Computational Linguistics.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378.

John Heritage. 1998. Conversation analysis and institutional talk. Analyzing distinctive turn-taking systems. In *Proceedings of the 6th International Congress of IADA (International Association for Dialog Analysis), Tubingen, Niemeyer*.

John Heritage. 2005. Conversation analysis and institutional talk. *Handbook of language and social interaction*, pages 103–147.

Discourse Resource Initiative. 1997. Standards for dialogue coding in natural language processing. In *Technical-Report167, Dagstuhl-Seminar*.

A. Jameson, B. Kipper, A. Ndiaye, R. Schaefer, J. Simons, T. Weis, and D. Zimmermann. 1994. Cooperating to be Noncooperative: The Dialog System PRACMA. *Lecture Notes in Computer Science*, pages 106–106.

A. Jameson. 1989. But what will the listener think? Belief ascription and image maintenance in dialog. *User Models in Dialog Systems. Springer-Verlag*, pages 255–312.

Jörn Kreutel and Colin Matheson. 2001. Cooperation and strategic acting in discussion scenarios. In *Proceedings of the Workshop on Coordination and Action*. Citeseer.

Jörn Kreutel and Colin Matheson. 2003. Incremental information state updates in an obligation-driven dialogue model. *Logic Journal of IGPL*, 11(4):485–511.

Klaus Krippendorff. 1995. *On the reliability of unitizing continuous data*. Sociological Methodology.

Klaus Krippendorff. 2003. *Content Analysis: An Introduction to Its Methodology*. Sage Publications, Incorporated, second edition edition, December.

J.R. Landis and G.G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174.

Colin Matheson, Massimo Poesio, and David Traum. 2000. Modelling grounding and discourse obligations using update rules. In *Proceedings of the 1st conference on North American chapter of the Association for Computational Linguistics*, pages 1–8, Morgan Kaufmann Publishers Inc. Morgan Kaufmann Publishers Inc.

Brian Plüss, David DeVault, and David Traum. 2011. Toward Rapid Development of Multi-Party Virtual Human Negotiation Scenarios. In *SemDial 2011: Proceedings of the 15th Workshop on the Semantics and Pragmatics of Dialogue*, November.

Brian Plüss. 2014. *A Computational Model of Non-Cooperation in Natural Language Dialogue*. Ph.D. Thesis. Department of Computing and Communications, The Open University.

Massimo Poesio and David Traum. 1997. Conversational actions and discourse situations. *Computational intelligence*, 13(3):309–347.

Massimo Poesio and David Traum. 1998. Towards an Axiomatization of Dialogue Acts. In *Proceedings of the Twente Workshop on the Formal Semantics and Pragmatics of Dialogues*, pages 207–222.

William A. Scott. 1955. Reliability of content analysis: The case of nominal scale coding. *Public opinion quarterly*, 19(3):321–325.

Sidney Siegel and N. John Castellan. 1988. *Nonparametric statistics for the behavioral sciences*. McGraw-Hill, New York, 2 edition, January.

Svetlana Stoyanchev and Paul Piwek. 2010a. Annotation Scheme for Authored Dialogues. Version 1.1. Technical Report 2010/15, Centre for Research in Computing, The Open University, July.

Svetlana Stoyanchev and Paul Piwek. 2010b. Constructing the CODA corpus: A parallel corpus of monologues and expository dialogues. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, Malta, May.

J. A. Taylor, Jean Carletta, and C. Mellish. 1996. Requirements for belief models in cooperative dialogue. *User Modeling and User-Adapted Interaction*, 6(1):23–68.

David Traum and James Allen. 1994. Discourse obligations in dialogue processing. In *Proceedings of the 32nd annual meeting of ACL*, pages 1–8. Association for Computational Linguistics Morristown, NJ, USA.

David Traum and Elizabeth A Hinkelman. 1992. Conversation acts in task-oriented spoken dialogue. *Computational intelligence*, 8(3):575–599.

David Traum, W. Swartout, S. Marsella, and J. Gratch. 2005. Fight, Flight, or Negotiate: Believable Strategies for Conversing Under Crisis. *Lecture Notes in Computer Science*, 3661:52.

David Traum, W. Swartout, J. Gratch, and S. Marsella. 2008. A virtual human dialogue model for non-team interaction. *Recent Trends in Discourse and Dialogue. Springer*.

David Traum. 2008. Extended Abstract: Computational Models of Non-cooperative dialogue. In Jonathan Ginzburg, Patrick Healey, and Yo Sato, editors, *Proceedings of LONDIAL 2008, the 12th Workshop on the Semantics and Pragmatics of Dialogue*, pages 11–14, London, UK.

David Traum. 2012. Non-cooperative and Deceptive Virtual Agents. *IEEE Intelligent Systems: Trends and Controversies: Computational Deception and Noncooperation*, 27(6):66–69.

Representation and Learning of Temporal Relations

Leon Derczynski
Department of Computer Science
University of Sheffield
S1 4DP, UK
leon.d@shef.ac.uk

Abstract

Determining the relative order of events and times described in text is an important problem in natural language processing. It is also a difficult one: general state-of-the-art performance has been stuck at a relatively low ceiling for years. We investigate the representation of temporal relations, and empirically evaluate the effect that various temporal relation representations have on machine learning performance. While machine learning performance decreases with increased representational expressiveness, not all representation simplifications have equal impact.

1 Introduction

Textual accounts often contain descriptions of events, times, and how they relate to one another temporally. To connect events and times to each other, we need to know the kind of temporal ordering between them. This ordering can be modeled with **temporal relations** that hold between pairs of entities, each of which may be an event or time. Extracting these relations is critical to understanding the text: for example, given an almanac of presidents of the USA, there are likely to be many indications of different people being president – but only one will be factual at any given time. In order to reason about events and the applicability of information in a document, linguistic expressions of time need to be converted to a formal representation. This task is temporal relation annotation.

To annotate temporal relations for reasoning or information extraction, one must to select a way of representing temporal relations. Such representations typically comprise a **set of temporal relations**, with each member describing a different temporal ordering. Building such representations is a key artificial intelligence task in reasoning and planning, and proposed solutions have amounted to major work in the field; e.g. Allen (1984), Freksa (1992).

Temporal relation annotation has two key parts. One must decide which entity pairs to relate, and then determine the nature of their relation. These are referred to as temporal relation **identification** and temporal relation **typing**.

These may be approached as a joint task, especially when it is possible for the type of one link to influence the type of another. For example, if event A is before event B and that event B is before event C, due to transitivity, the choice of relations between A and C may be constrained. Choosing how to represent the types of temporal relations is the focus of this paper.

Machine learning of temporal relations is hard. Mani et al. (2007) detail experiments with features annotated in TimeML (Pustejovsky et al., 2004), and reach around 75% accuracy at overall relation typing,[1] using gold-standard relation identification.[2] This pattern repeats in subsequent literature. Many others reach a similar performance, or perhaps even exceed it by 1-2% (Mirroshandel et al., 2011; Do et al., 2012); some do well by focusing on specific sub-parts of the problem (e.g. relations that are expressed by tense shifts (Derczynski and Gaizauskas, 2013b); relations between events in the same verb clause construction (Bethard et al., 2007)), or in contrast by taking a holistic whole-graph approach (Chambers

[1] Calculated by weighing the event-event and event-timex relation scores in their paper according to the distribution in the dataset used.

[2] This is typically made up of around 70% for event-event relations and 80% for event-time relations.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1937–1948, Osaka, Japan, December 11-17 2016.

Relation	Symbol	Explanation of A-relation-B
before	<	A finishes before B starts
after	>	A starts after B ends
equals	=	A and B happen at the same time
meets	m	A happens immediately before B
is met by	mi	A happens immediately after B
overlaps	o	A is an interval during which B starts but does not finish
is overlapped by	oi	A starts within B but finishes after
during	d	A happens between B's start and finish
contains	di	A starts before, and finishes after, B
starts	s	A and B start at the same time, but B continues past A's end
is started by	si	A starts at the same time as B, but then goes on for longer
finishes	f	A starts after B, but they finish at the same time
is finished by	fi	A starts before B, and they finish at the same time

Table 1: Temporal interval relations, using Allen's symbols

et al., 2014); or relations where a temporal conjunction is present (Derczynski and Gaizauskas, 2013a)); but no general breakthrough appears to be on the horizon.

Indeed, the top systems in each of the TempEval challenges had tightly-clustered scores showing some 5-10% error reduction over the most-common-class baseline, despite the work being spread over years of active research (Verhagen et al., 2009; UzZaman et al., 2013; Bethard et al., 2016).

When one considers how human annotators cope with the task as it is often cast, it is unsurprising that machine performance is not high. Choosing from a set of abstract temporal relation types to fit an arbitrary pair of event mentions in a given document is difficult. For the largest TimeML corpus, inter-annotator agreement on relation types was just 0.71 (kappa) between a set of experts familiar with temporal annotation.[3] In this case, they had to assign one of a set of fourteen different temporal ordering types to the pair of events.

Problems may lie in the schema of temporal relation types. Assigning types requires annotators to perform abstract reasoning often with incomplete information; systems must do the same. Thus, the choice of representation for temporal relation types is critical. The more relation types to choose from, the more reasoning is required, and so the potential for error increases.

The bulk of work has concentrated on describing the order of two events by using relations from Allen's interval algebra. However, no work has directly investigated the effect of temporal relation representation design on performance of machine learning systems.

This paper investigates temporal relation type representations, in the framework of TimeML when possible. It compares a range of existing relation sets and discuss two dimensions for relation representation design. Following this is a general comparison of the impact that relation set choice can have on machine learning performance. Next follows discussion of techniques commonly used to improve training sets, including relation type mapping, and examine their impact. Finally, the paper investigates how discriminable often-conflated relations are, providing insight into the difficulty added by using finer-grained relation sets, and reports on research on human aspects of temporal relation representation.

2 Temporal Representation Schemes

This section introduces fundamental temporal relation type sets for linguistic annotation. Standards for representing temporal relations are then described in the context of these fundamental relation sets.

[3] See http://timeml.org/site/timebank/documentation-1.2.html

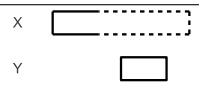
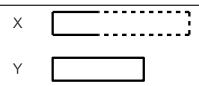
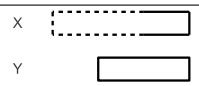
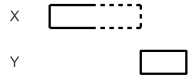
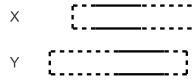
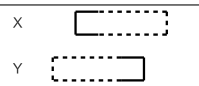
Relation	Illustration	TimeML relation type disjunction
X is *older* than Y Y is *younger* than X		X [BEFORE, IBEFORE, ENDED_BY, INCLUDES, DURING] Y
X is *head to head* with Y		X [BEGINS, SIMULTANEOUS, IDENTITY, BEGUN_BY] Y
X *survives* Y Y is *survived by* X		X [INCLUDES, BEGUN_BY, IAFTER, AFTER] Y
X is *tail to tail* with Y		X [ENDED_BY, SIMULTANEOUS, IDENTITY, ENDS] Y
X *precedes* Y Y *succeeds* X		X [BEFORE, IBEFORE, ENDED_BY, INCLUDES, DURING_INV] Y
X is a *contemporary* of Y		X [INCLUDES, IS_INCLUDED, BEGUN_BY, BEGINS, DURING, DURING_INV, SIMULTANEOUS, IDENTITY, ENDS, ENDED_BY] Y
X is *born before death* of Y Y *dies after birth* of X		X [IS_INCLUDED, ENDS, DURING_INV, BEFORE, IBEFORE, INCLUDES, DURING, ENDED_BY] Y

Table 2: Semi-interval relations. Adapted from Freksa (1992). The superset of relations is omitted here, but related in that work.

2.1 A Very Simple Relation Set

A Very Simple set of relations for representing temporal relations could have just three members: BEFORE, OVERLAP, and AFTER. These are mutually exclusive; before means wholly before, and after means wholly after, precluding any ambiguity once a relation type has been assigned. No specific model of events is required here.

However, this representation is too simplistic to describe many of the temporal relations that are often explicitly conveyed in language. For example, there is no way to represent the distinction between *after* and *just after*, or between *I opened the door yesterday* and *I held the door open all day yesterday*.

2.2 Interval Relations

Allen (1983) introduces a richer relation set which models the events (or times) that are related as **temporal intervals**. Each interval consists of a start and end point, which correspond to when the item in question began and ended. Based on this, thirteen possible interval arrangements are identified and given names. The relations are shown in Table 1. This model can be used to model relations between events and times, both those observed and those described in natural language (Allen, 1984).

2.3 Semi-Interval Relations

Freksa (1992) describes a relation set based on semi-intervals, with temporal reasoning based on natural language as one of its intended uses. In this semi-interval relation set and algebra, events and times are still modelled as temporal intervals, but one is not required to fully specify the bounds of both related intervals in order to describe the relation between them. Rather, different relation types apply depending on both the order and the degree of specification. This allows for a more expressive representation. A subset of these underspecified interval relations are demonstrated in Table 2.

Its relation types (or "conceptual neighbourhoods") can be thought of as disjunctions of Allen interval relation types. For example, Freksa's TAIL-TO-TAIL relation corresponds to any situation where both

intervals end simultaneously (in Allen's terms, $\texttt{fi} \vee = \vee \texttt{f}$). The CONTEMPORARY relation is analogous to our Very Simple relation set's OVERLAP type (Section 2.1). If A is the set of Allen interval relation types and F the set of Freksa semi-interval relation types, then $F \subset \mathcal{P}(A)$. This semi-interval relation set is flexible, but this comes at a cost. Annotators and relation typing systems have a greater selection of relation types to choose from: there are total 31 types.

These three representations provide most of the concepts required to understand the other major relation type sets.

2.4 Annotation Standards

TimeML TimeML uses the Allen relations (although slightly renamed), and adds a IDENTITY relation to indicate co-reference. This extra relation is temporally equivalent to SIMULTANEOUS. Interpretation of TimeML's DURING and DURING_INV relations has been ambiguous; this work adopts the TimeML-strict definitions (Derczynski et al., 2013), which map directly to Allen's overlap and overlap-inverse.

As with Allen's temporal interval relations, TimeML requires full specification of both intervals and established versions only permit assignment of a single relation type. Links in newer prototype versions of ISO-TimeML (Pustejovsky et al., 2010) permit selection of disjunctions of interval relation types.

TempEval TempEval-1 and TempEval-2 (Verhagen et al., 2009; Verhagen et al., 2010) used the same three types as the Very Simple scheme, plus two less-specific types – BEFORE-OR-OVERLAP and AFTER-OR-OVERLAP – and the relaxed VAGUE. TempEval-3 used TimeML relations.

STAG The STAG annotation scheme (Setzer and Gaizauskas, 2000) – a precursor to TimeML – specifies three basic relations: SIMULTANEOUS, BEFORE and INCLUDES. Representation of "after" and "is-included-by" relations can be achieved by swapping the order of arguments (i.e. A BEFORE B vs. B BEFORE A). This intuitive, simple scheme cannot express Allen-style overlap, and the STAG INCLUDES relation is vague in that it subsumes BEGINS and STARTS Allen relation types.

Narrative Containers Pustejovsky and Stubbs (2011) attempt to reduce the scope of temporal relation typing task by defining narrative containers over sets of event mentions in documents. These containers represent a time interval during which groups of events occur. The containing time interval may either be explicit and reified with a narrative time, or implicit, with the exact bounds determined by context and text type. Containers reduce the amount of uncertainty in relation typing by grouping related events into local contiguous temporal scopes. This reduces annotator load and provides a basis for the temporal structure of a given document, without requiring specification of the type of every temporal relation. The narrative container approach is designed to support event-to-time relation typing, and succeeds in doing so directly for 50% of events with more informative relation descriptions. Empirical work (Miller et al., 2013) indicates that these containers can be annotated by humans sufficiently well to learn an initial automated approach. However, in the absence of a large general-purpose corpus annotated with them, this paper does not include narrative containers.

For a general overview and history of temporal annotation standards, see (Strötgen and Gertz, 2016).

3 Analysing temporal relation sets

This section investigates selected sets of temporal relation types. The sets are chosen to demonstrate key differences, though the list is not exhaustive. A graph-based comparison of three sets can also be found in (Denis and Muller, 2010).

3.1 Expressiveness and Specificity

We qualitatively describe temporal relation sets in two dimensions: The first dimension, *expressiveness vs. simplicity* details the range of different combinations of event orderings they can capture. The second, *specificity vs. laxness* details how much constraint the relation set's types imply, and how much one needs to know before typing a relation.

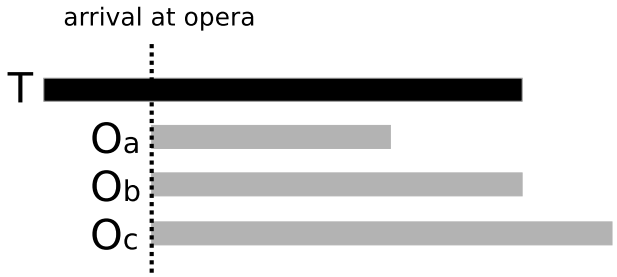


Figure 1: The Opera problem: Linguistic ambiguity leads to inability to choose a single interval relation. Here, we know the time of arrival at the opera but not of the departure, making it hard to relate intervals O and T.

For example, the Very Simple relation set is not very expressive – it has only three relations. The interval set is specific – one must describe relations using both endpoints of both intervals. Compared to the interval set, semi-interval relations are more expressive and less specific, because less information is required to choose a relation type.

3.2 Precision and Annotation

When describing the order of events and times, it is important to get the right degree of precision. Time can be thought of as a unidirectional, continuous dimension. This continuous aspect of time permits relatively small differences in the alignments of events, which may be at best useless and at worst counter-productive when reasoning.

However, lacking an inherent temporal quantisation or discretisation framework (such as chronons (Lévi, 1927)), we have no objective ground for ignoring small discrepancies in event timing. Instead, cues come from the temporal scale at which events take place (Gaizauskas et al., 2012).

A pragmatic approach suits this situation. Annotations over a text should match the information expressed. So, in *"I smiled when she entered the room"*, whether the smiling and the entrance were delayed by a few fractions of a second is unlikely to be salient. Rather, the language indicates that these events happened at the same time, and so this is the best thing to annotate. Texts regarding quantum events or star formation would both use different temporal scales.

3.3 The Opera Problem

The interval-based relation set is a richer representation than the Very Simple one, but has the disadvantage that the order of both related intervals' start and end (i.e. four points in total) must be known before a single relation type can be chosen. It is desirable to choose a single relation type when annotating linguistic data in order to keep down the number of choices that annotators need to make, which in turn affects the quality of output. The linguistic ambiguity in natural language texts can make it hard to choose a single interval relation type. This can be illustrated using the Opera problem, as put forward in Derczynski (2017)

Given the statement *Irene went to the opera today*, assume interval O is the visit to the opera and interval T is today.[4] Without knowing when Irene left, one cannot choose a single interval relation out of (a) O is included in T, (b) O and T end at the same time, or (c) O overlaps T – see Figure 1.

Underspecifying the endpoints of intervals allows ambiguity to be captured in a single relation type. The semi-interval relation set offers a solution the Opera problem using a relation where the start of O is specified but its end is not. In this representation it is O YOUNGER-CONTEMPORARY-OF T, which corresponds to the disjunction of Allen relations $\texttt{f} \vee \texttt{d} \vee \texttt{oi}$.

Similarly, the Very Simple set can give a lossy representation for this problem with its OVERLAP relation. It is lossy because the text describes more than can be represented by this relation type.

Confusions like the one the Opera problem presents occur frequently in TimeML annotation when linking events to the document time stamp, which is typically a one-day interval. For example, news

[4]This treats O as an eventuality which represents the state of Irene being at the opera; (Hobbs and Pustejovsky, 2003) examines the semantics of events in the context of temporal interval representations.

stories – the bulk of the TimeML annotated data in existence – refer to the day in which they are reported, but as one cannot see into the future and determine whether or not assertions and events in the news will persist past the end of the day, it is not always possible to accurately assign a single relation type with certainty.

4 Machine Learning and Temporal Relation Types

There are specific problems when applying machine learning to the temporal relation typing task. When approached as a classification task, the assumption made by most tools is that classes are independent. However, this is not correct in this case, for two reasons.

Firstly, relation types are not all equally different. TimeML's "immediately after" relation, IAFTER, is more like AFTER than it is INCLUDES. In fact, this non-orthogonality is critical to the construction and behaviour of Freksa's conceptual neighbourhood relations. The connected nature of relation types offers nuance in training, classification and evaluation of machine learning approaches to temporal relation typing.

Secondly, relation typing is a highly structured task, with local and global dependencies. Not only should local relation constraints (e.g. through transitivity and commutativity) be taken into account, as noted by Hovy et al. (2012); also, the annotation of a well-formed document should be globally temporally consistent, otherwise it will detail an impossible sequence of events.[5]

This interdependent aspect of temporal relations is hard to model, partially due to the underlying computational complexity (Vilain and Kautz, 1986). Nevertheless, including features for local dependencies give modest accuracy improvements (Chambers and Jurafsky, 2008; Yoshikawa et al., 2009).

5 Evaluating Learning of Relation Sets

Using a variety of temporal relation sets, machine learning of temporal relations is evaluated over the same data. This is achieved by mapping TimeML gold-standard relations to other relation sets thus:

Simple – before: BEFORE, IBEFORE; after: AFTER, IAFTER; overlap: everything else.

STAG – add inverses of the before and includes types, to make five relations. before: BEFORE, IBEFORE; after: AFTER, IAFTER; simultaneous: SIMULTANEOUS, DURING, DURING_INV, IDENTITY; is-included: IS-INCLUDED; includes: everything else.

Allen – One change: TimeML IDENTITY becomes Allen's "=" (equal, i.e. simultaneous).

TimeBank 1.2 (Pustejovsky et al., 2003) is the corpus, which has 6 418 temporal relations (TimeML TLINKs) in 183 documents. This is merged with the AQUAINT TimeML corpus, which has 2 340 TLINKs in 73 documents.[6] The AQUAINT corpus is structured in four themes, each of which contains multiple documents tracking the same story over time.

A 75%/25% split training and evaluation split is used. To avoid information that could be extracted through inference leaking between documents, splits are made at document and not TLINK level.

When evaluating learning of temporal information, one must also be wary of **timeline pollution** (Bergmeir and Benítez, 2012) – that is, introducing later knowledge into the training set that might give clues as to the contents of an earlier document in the test set. This pollution concern is offset by avoiding duplicating text across splits. These factors work against each other: avoiding timeline pollution, by only including later-dated documents in the test split, increases the risk of these documents re-using fragments of earlier articles on the same topic. Avoiding re-use is probably more important from a basic NLP point of view, so splits are arranged to reduce this. The solution adopted in the AQUAINT corpus is to split at theme-level, as a lot of similar text is repeated across each theme.

The feature set is the same as that in Mani et al. (2007). These are the attributes and values of the TimeML annotations, plus the strings they annotate, and in addition: the string of the conjunction coordinating the relation, if present; and two flags indicating whether, in a relation between two verb events, the verbs have the same tense or same aspect. Following previous work (Mani et al., 2007), experiments use a maximum entropy classifier (MegaM (Daumé III, 2004)), which has consistently provided stable

[5] This requirement makes the assumption that the author has written a temporally consistent story.

[6] From http://timeml.org/site/timebank/timebank.html

Relation set	MCC baseline	MaxEnt	ID3
Simple	44.79	73.96	57.65
STAG	38.00	60.36	51.13
Allen	26.08	58.58	46.01
TimeML	26.08	57.08	45.51

Table 3: Relation typing precision using a variety of temporal relation sets, compared to a most-common-class baseline.

Relation set	MCC baseline	MaxEnt	ID3
TempEval-2	55.61	59.25	61.73
TimeML	33.87	50.43	33.03

Table 4: Relation typing precision on a smaller set of links, using the TempEval-2 relations.

results in this task, and a decision tree classifier, ID3 (Quinlan, 1986), for its different inductive bias – i.e., away from the independence assumption. The goal here is to highlight relative performance using various relation representations. Results are given in Table 3.

Some relation sets could not be directly mapped from TimeBank. The TempEval-2 data uses a subset of TimeBank, but due to its relation set's multiple levels of specification, cannot be directly mapped. In a second set of experiments, the TempEval-2 dataset is compared with the corresponding TimeML relations. Performance using TimeML is included for comparison. Results are given in Table 4.

Generally, results suggest automatic relation typing is easier on smaller and simpler relation type sets. Greater expressiveness leads to both lower baselines and lower ML performance. Note the performance difference between Allen and TimeML relations sets – these are identical except for the addition of a co-reference relation in TimeML, for which relations are slightly harder to learn. Our Very Simple relation set of just three coarse-grained temporal arrangements was the easiest to assign using the existing data.This anti-monotone relation between increased expressiveness and lower machine learning performance held over all representations and algorithms tested.

6 Adapting Training Data

This section examines techniques for improving machine learning of temporal relations: simplifying the problem, and increasing available training data. Taking transitive closures is omitted, as it does not help (Mani et al., 2007).

6.1 Folding

Many relation types used in both TimeML and Allen's relation sets have a corresponding inverse relation. Mapping between these can be achieved by inverting the relation type and the argument order. For example, BEFORE*(Monday, Tuesday)* is equivalent to AFTER*(Tuesday, Monday)*. This relation **folding** simplifies classification by reducing the number of target classes, and is common in temporal relation classification, e.g. Mani et al. (2007); Mirroshandel et al. (2011).

6.2 Doubling

Doubling the reverse of folding: instead of using a relation type mapping to reduce the number of classes, use the mappings to double the number of examples. In controlled evaluation, it is possible to reverse the order of arguments in the evaluation set so the set only contains relation types the classifier has seen in folded training data. This is not possible where the relation type is never known, as one does not have control over argument order. E.g. if all AFTER relations are removed from the training data by swapping their arguments and changing them to BEFORE, when faced with the previously-unseen relation C AFTER D, a classifier trained on folded data will fail.

To solve this, instead of reducing the set of available relations, one can increase training data size by using the folding mappings to create new training instances instead of altering existing ones. That is, given an example A AFTER B one automatically adds an extra B BEFORE A. This is relation doubling.

Data set	MCC baseline	MaxEnt	ID3
Unfolded TimeML	26.08	57.08	45.51
Doubled TimeML	26.08	28.68	18.58
Folded TimeML	42.12	70.48	59.93

Table 5: Relation typing precision on a smaller set of links, using folding and doubling.

6.3 Closure

Many relation sets have types that exhibit properties of transitivity or commutativity. While a useful technique for increasing the volume of available data, the results of adding training data generated through closure can be unpredictable, and it is not always a suitable technique to apply; therefore, we omit effects of temporal graph closure here.

For example, in the interval relations, if A BEFORE B and B BEFORE C, then by transitive closure A BEFORE C. These properties allow the automatic addition of temporal relations to an existing set through inference. The set of links resulting from inferring as many possible relations from a source document is called that document's temporal closure.

While a useful technique for increasing the volume of available data, the results of adding training data generated through closure can be unpredictable, and it is not always a suitable technique to apply. For further reference, results on the use of closed data are presented by Mani et al.; extended discussion can be found in Verhagen (Verhagen, 2004).

6.4 Evaluation

Machine learning performance on folded training data using the TimeML relation set is evaluated as above. Results are in Table 5. Folding offers a big improvement over other methods, in terms of both error reduction above most-common-class baseline and also absolute accuracy. However, it is not applicable to real-world data. While doubling overcomes this critical deficiency, it overall reduces performance.

7 Relation Distinctions

As relation sets become more complex, categories of temporal relation are subdivided. For example, TimeML distinguishes between immediately before, IBEFORE, and before with an interceding gap, BEFORE, whereas neither our simple set, the STAG set or the TempEval-2 set do. It may be that some of these distinctions are easier to learn than others. To investigate, subgroups of relations that may be (or have been) sources of potential confusion are identified, and classifiers trained just on relations of the types in the group. These were then evaluated on the relations in the group from the test examples. As these subdivisions result in small amounts of training data, folded relations are used to simplify the problem and improve the examples to classes ratio.

The confusion between TimeML DURING and INCLUDES relations is also investigated. These were identified by annotators as confusing during TempEval-3 and cursory examination during TempEval-3 revealed many questionably annotated examples in TimeBank.

Results are shown in Table 6. Many of the finer distinctions can be successfully made in these situations, though there is significant variation. Unsurprisingly, accuracy on each relation type is higher the more examples are seen, with most having accuracies above 80% and some in the high 90s. The *overlap* group was hardest, at an accuracy of 55.23%. It also has one of the least skewed class distributions. It is interesting to note that relations in the STAG *includes* group, while sharing much with the *overlap* group, was generally easier to distinguish from one another. The co-reference relation added to the Allen relations by TimeML is also relatively easy to distinguish from non co-referent simultaneity, with the classifier reaching 30% – 50% error reduction above baseline. Indeed, (Glavaš and Šnajder, 2013) achieve F1 scores as high as 0.95 using graph kernels on this problem. With regard to the TimeML INCLUDES/DURING confusion, binary classification accuracy is lower than the most-common-class baseline; an indication of a difficult problem, or of poor-quality data – which uncertain annotators would generate.

Coarse relation	TimeML rel'n	Data %	Acc.
Before (3 300 instances)	BEFORE	97.17	99.65
	IBEFORE	2.83	31.34
	Overall	-	97.72
After (1 590 instances)	AFTER	95.34	99.12
	IAFTER	4.66	20.00
	Overall	-	95.43
Overlap (2 792 instances)	IDENTITY	35.52	64.98
	INCLUDES	31.09	61.29
	SIMUL.	23.89	50.82
	ENDS	4.91	47.45
	BEGINS	4.12	11.30
	DURING	3.47	3.09
	Overall	-	55.23
STAG includes (1 120 instances)	INCLUDES	77.50	96.20
	ENDS	12.23	52.55
	BEGINS	10.27	19.13
	Overall	-	82.95
Coreference distinction (1 646 instances)	IDENTITY	57.65	78.06
	SIMUL.	42.35	60.87
	Overall	-	72.76
TimeML "during" (4 284 instances)	INCLUDES	89.95	97.70
	DURING	10.05	11.34
	Overall	-	89.02

Table 6: Folded relation typing precision when training on and classifying subgroups of similar TimeML relation types.

8 Human Factors in Temporal Annotation

When it comes to designing new representations, human factors can have bearing on design choices. Scheuermann et al. (2013) provide valuable insights into annotator preferences in temporal annotation. They report that experts prefer more expressive representations, at the cost of simplicity, whereas non-experts lean towards perceived user-friendliness at the cost of expressivity. This matches the anti-monotone interaction between machine learning performance and representational expressiveness: the simpler the representation, the easier it is to annotate and to learn.[7] Large variation was seen across all annotators. Schaeken and Johnson-Laird (2000) show that agreement and success of humans annotating timelines can be perturbed by even slight changes in text phrasing, which perhaps explains the variations seen in Scheuermann et al.'s work.

These findings are worth considering in the construction of relation representations and human temporal annotation tasks. One caveat, as Allen (1991) points out, is that the presence of parallels between natural language and a given representation is not always indicative of its suitability for automated reasoning. Freksa's relation set may be a better target for human annotation: it is intended to better reflect the usage of temporal relation expression in natural language, and only requires judgment over pairs of points rather than intervals, for construction. For example, one may decompose interval relations into point relations, and ask "Does A begin before B ends?". We already know that simpler questions yield better results from annotators (Sabou et al., 2014), and so a decomposed, less constrained approach to annotation may be fruitful. Certainly, the 13-way Allen (or 14-way TimeML) task has been tough, and it is reasonable that some disagreements come from annotators having to chose a particular relation type when in fact many (or none) seem appropriate to them.

9 Conclusion

This paper has investigated choices in representation of temporal relations. Human annotation agreement is often low in relation typing, and the task is also difficult for automatic systems. The expressivity of a representation has an inverse correlation with automatic temporal relation typing performance.

However, more expressive representations are required in order to accurately capture temporal structure, or even to reduce annotator confusion. It was shown that decisions over fine distinctions between

[7] The effect agreement has on training data is not measured here, but is stable across sets at around kappa 0.7.

temporal relation types in more expressive systems are of varying difficulty. This finding fits the observation that not all temporal relation types are equally different from each other.

In summary, expressivity is lacking in popular temporal relation representations, but large relation types sets are harder to annotate. It is important to get specificity right in a given relation set, and to select carefully the groupings made when creating coarser relation sets. Therefore, being aware of the expressiveness of different representations is critical to choosing how to represent temporal relations effectively for a given scenario.

Acknowledgements

Thanks to all who reviewed this paper, for their extensive, constructive feedback. Thanks also to Rob Gaizauskas for feedback on precursors to this work, and to Aarhus University for their kind facility provision. This research has received support from the EU from the H2020 program under grant agreement No. 687847, COMRADES.

References

J. F. Allen. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

J. F. Allen. 1984. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154.

J. F. Allen. 1991. Time and time again: The many ways to represent time. *International Journal of Intelligent Systems*, 6(4):341–355.

C. Bergmeir and J. M. Benítez. 2012. On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192–213.

S. Bethard, J. Martin, and S. Klingenstein. 2007. Timelines from Text: Identification of Syntactic Temporal Relations. In *Proceedings of the International Conference on Semantic Computing (ICSC)*, pages 11–18.

S. Bethard, G. Savova, W.-T. Chen, L. Derczynski, J. Pustejovsky, and M. Verhagen. 2016. Semeval-2016 task 12: Clinical TempEval. *Proceedings of SemEval*, pages 1052–1062.

N. Chambers and D. Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 698–706. ACL.

N. Chambers, T. Cassidy, B. McDowell, and S. Bethard. 2014. Dense Event Ordering with a Multi-Pass Architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

H. Daumé III. 2004. Notes on CG and LM-BFGS Optimization of Logistic Regression. Paper available at `http://pub.hal3.name`, implementation available at `http://hal3.name/megam/`, August.

P. Denis and P. Muller. 2010. Comparison of different algebras for inducing the temporal structure of texts. In *Proceedings of Coling 2010*, pages 250–258, Beijing.

L. Derczynski and R. Gaizauskas. 2013a. Temporal Signals Help Label Temporal Relations. In *Proceedings of the annual meeting of the Association for Computational Linguistics (ACL)*, pages 645–650. ACL.

L. Derczynski and R. Gaizauskas. 2013b. Empirical Validation of Reichenbach's Tense Framework. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS)*, pages 71–82.

L. Derczynski, H. Llorens, and N. UzZaman. 2013. TimeML-strict: clarifying temporal annotation. *arXiv preprint arXiv:1304.7289*.

L. Derczynski. 2017. *Automatically Ordering Events and Times in Text*. Springer.

Q. X. Do, W. Lu, and D. Roth. 2012. Joint inference for event timeline construction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 677–687. ACL.

C. Freksa. 1992. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1):199–227.

R. Gaizauskas, E. Barker, C.-L. Chang, L. Derczynski, M. Phiri, and C. Peng. 2012. Applying ISO-Space to Healthcare Facility Design Evaluation Reports. In *Seventh Workshop on Interoperable Semantic Annotation (ISA), Eighth International Conference on Language Resources and Evaluation*, pages 13–20.

G. Glavaš and J. Šnajder. 2013. Recognizing Identical Events with Graph Kernels. In *Proceedings of the annual meeting of the Association for Computational Linguistics (ACL)*, pages 797–803. ACL.

J. Hobbs and J. Pustejovsky. 2003. Annotating and reasoning about time and events. In *Proceedings of AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, SS-03-05, pages 1–9.

D. Hovy, J. Fan, A. Gliozzo, S. Patwardhan, and C. Welty. 2012. When Did that Happen? - Linking Events and Relations to Timestamps. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 185–193. ACL.

R. Lévi. 1927. Théorie de l'action universelle et discontinue. *J. Phys. Radium*, 8(4):182–198.

I. Mani, B. Wellner, M. Verhagen, and J. Pustejovsky. 2007. Three approaches to learning TLINKS in TimeML. Technical Report CS-07-268, Brandeis University, Waltham, MA, USA.

T. Miller, S. Bethard, D. Dligach, S. Pradhan, C. Lin, and G. Savova. 2013. Discovering Temporal Narrative Containers in Clinical Text. In *Proceedings of the Workshop on Biomedical Natural Language Processing*, pages 18–26. ACL.

S. Mirroshandel, G. Ghassem-Sani, and A. Nasr. 2011. Active Learning Strategies for Support Vector Machines, Application to Temporal Relation Classification. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, pages 56–64.

J. Pustejovsky and A. Stubbs. 2011. Increasing informativeness in temporal annotation. In *Proceedings of the Linguistic Annotation Workshop*, pages 152–160. ACL.

J. Pustejovsky, P. Hanks, R. Sauri, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro, et al. 2003. The Timebank Corpus. In *Proceedings of the Corpus Linguistics Conference*, pages 647–656.

J. Pustejovsky, B. Ingria, R. Sauri, J. Castano, J. Littman, and R. Gaizauskas. 2004. The Specification Language TimeML. In *The Language of Time: A Reader*, pages 545–557. Oxford University Press.

J. Pustejovsky, K. Lee, H. Bunt, and L. Romary. 2010. ISO-TimeML: An International Standard for Semantic Annotation. In *Proceedings of the International conference on Language Resources and Evaluation (LREC)*, pages 394–397. European Language Resources Association.

J. R. Quinlan. 1986. Induction of decision trees. *Machine learning*, 1(1):81–106.

M. Sabou, K. Bontcheva, L. Derczynski, and A. Scharl. 2014. Corpus Annotation through Crowdsourcing: Towards Best Practice Guidelines. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, pages 859–866. ELRA.

W. Schaeken and P. N. Johnson-Laird. 2000. Strategies in temporal reasoning. *Thinking & Reasoning*, 6(3):193–219.

A. Scheuermann, E. Motta, P. Mulholland, A. Gangemi, and V. Presutti. 2013. An empirical perspective on representing time. In *Proceedings of the international conference on Knowledge Capture*, pages 89–96. ACM.

A. Setzer and R. Gaizauskas. 2000. Annotating events and temporal information in newswire texts. In *Proceedings of the Second International Conference On Language Resources And Evaluation (LREC)*, 321, pages 1–7. European Language Resources Association.

J. Strötgen and M. Gertz. 2016. *Domain-Sensitive Temporal Tagging*. Morgan Claypool.

N. UzZaman, H. Llorens, L. Derczynski, M. Verhagen, J. F. Allen, and J. Pustejovsky. 2013. SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations. In *Proceedings of SemEval*, pages 1–9. ACL.

M. Verhagen, R. Gaizauskas, F. Schilder, M. Hepple, J. Moszkowicz, and J. Pustejovsky. 2009. The TempEval challenge: identifying temporal relations in text. *Language Resources and Evaluation*, 43(2):161–179.

M. Verhagen, R. Saurí, T. Caselli, and J. Pustejovsky. 2010. SemEval-2010 task 13: TempEval-2. In *Proceedings of SemEval*, pages 57–62. ACL.

M. Verhagen. 2004. *Times Between The Lines*. Ph.D. thesis, Brandeis University.

M. B. Vilain and H. A. Kautz. 1986. Constraint Propagation Algorithms for Temporal Reasoning. In *Proceedings of the AAAI Conference*, volume 86, pages 377–382.

K. Yoshikawa, S. Riedel, M. Asahara, and Y. Matsumoto. 2009. Jointly identifying temporal relations with Markov logic. In *Proceedings of the annual meeting of the Association for Computational Linguistics (ACL)*, pages 405–413. ACL.

Revisiting the Evaluation for Cross Document Event Coreference

Shyam Upadhyay Nitish Gupta Christos Christodoulopoulos Dan Roth
{upadhya3,ngupta19,christod,danr}@illinois.edu
University of Illinois at Urbana-Champaign, IL, USA

Abstract

Cross document event coreference (CDEC) is an important task that aims at aggregating event-related information across multiple documents. We revisit the evaluation for CDEC, and discover that past works have adopted different, often inconsistent, evaluation settings, which either overlook certain mistakes in coreference decisions, or make assumptions that simplify the coreference task considerably. We suggest a new evaluation methodology which overcomes these limitations, and allows for an accurate assessment of CDEC systems. Our new evaluation setting better reflects the corpus-wide information aggregation ability of CDEC systems by separating event-coreference decisions made across documents from those made within a document. In addition, we suggest a better baseline for the task and semi-automatically identify several inconsistent annotations in the evaluation dataset.

1 Introduction

Understanding events is crucial to natural language understanding and has applications ranging from question answering (Berant et al., 2014; Narayanan and Harabagiu, 2004), to causal reasoning (Do et al., 2011; Chambers and Jurafsky, 2008) to headline generation (Sun et al., 2015). The task of Cross Document Event Coreference (CDEC) determines if two event mentions (which belong to different documents) refer to the same event (Bagga and Baldwin, 1999; Bejan and Harabagiu, 2014). Figure 1 shows an example of two events whose mentions co-refer across 3 documents,

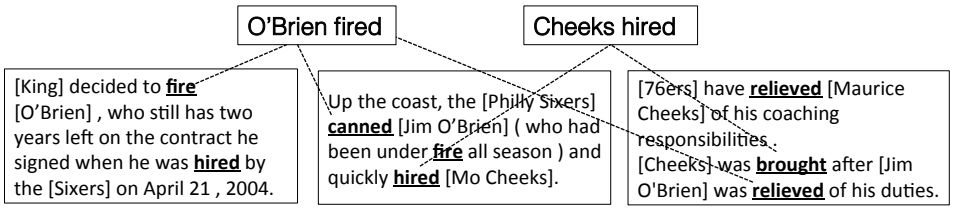


Figure 1: Event coreference across 3 documents. Event mentions are shown in **bold** and its participants are enclosed by brackets, []. Note that there are multiple **firing** events, but only a few co-refer.

An efficient CDEC system enables corpus-level aggregation of event attributes, which can prove valuable for tasks such as information extraction and aggregation (Humphreys et al., 1997; Zhang et al., 2015), topic detection and tracking (Allan et al., 1998), multi-document summarization (Daniel et al., 2003) and knowledge discovery (Mayfield et al., 2009).

In this work, we analyze whether existing CDEC evaluations reflect a system's ability to predict cross document links. Past works have adopted different evaluation methodologies which either make simplifying assumptions about the coreference task, such as ignoring singletons, or overlook certain coreference mistakes made by a CDEC system (as discussed in §4). Furthermore, under existing evaluations, a system which only predicts *within* document coreference links can score higher than a system which

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1949–1958, Osaka, Japan, December 11-17 2016.

predicts both within and cross document links. However, the latter system is clearly useful at aggregating information at corpus-level by finding across document links.

We address these issues by lifting the aforementioned assumptions, and proposing a new setting which enables accurate evaluation of the cross document coreference performance (§ 4.1). Lifting these assumptions also allowed us to semi-automatically identify several inconsistent annotations in the dataset. As the current evaluation dataset is the only available dataset with cross document coreference annotations, it is prudent to improve its annotation quality. We describe these annotation errors in § 7.3.

2 Related Work

Work on event coreference deals primarily with coreference within document, mostly building on insights gained from the entity coreference literature (Ng and Cardie, 2002; Bengtson and Roth, 2008; Lee et al., 2012; Peng et al., 2015). Recent approaches (Liu et al., 2014; Araki et al., 2014; Peng et al., 2016) have shown improvements in within document event coreference by exploiting event specific sub-structure (viz. sub-event or information propagation to arguments) and new event representations.

In comparison, cross document event coreference has been a less well-studied problem. Early work on cross document event coreference was done by Bagga and Baldwin (1999), who showed preliminary results on small exploratory datasets. To encourage research in this direction, Bejan and Harabagiu (2010) created the Event Coreference Bank (ECB), the first dataset with both within and across document event coreference annotations. ECB contained 482 documents, obtained from the Google News archive. Bejan and Harabagiu (2010) also showed encouraging results on ECB with several unsupervised Bayesian approaches. Later, ECB was augmented by Lee et al. (2012), to include entity level coreference annotations as well. On the new dataset, which they named EECB, they showed how entity and within document event coreference can benefit from making joint coreference decisions. Using EECB, Wolfe et al. (2015) formulated event coreference as an predicate alignment problem. Cybulska and Vossen (2014) noted that ECB and EECB did not have enough lexical diversity, thus oversimplifying the cross document coreference task. To get around this, they augmented the ECB corpus with 502 documents and released a larger corpus with event coreference annotations, named ECB+.

While there have been other works which use multi-modal supervision signals (Zhang et al., 2015) for CDEC, at present, ECB+ is the sole public dataset with cross document coreference annotations for events, leading to its popularity (Bejan and Harabagiu, 2014; Cybulska and Vossen, 2015; Yang et al., 2015). The most recent work using the ECB+ corpus is that of Yang et al. (2015), who developed a novel Bayesian clustering framework, which clusters event within and across documents, by modeling the clustering process as a Hierarchical Distance Dependent Chinese Restaurant Process (HDDCRP).

3 Cross Document Event Coreference

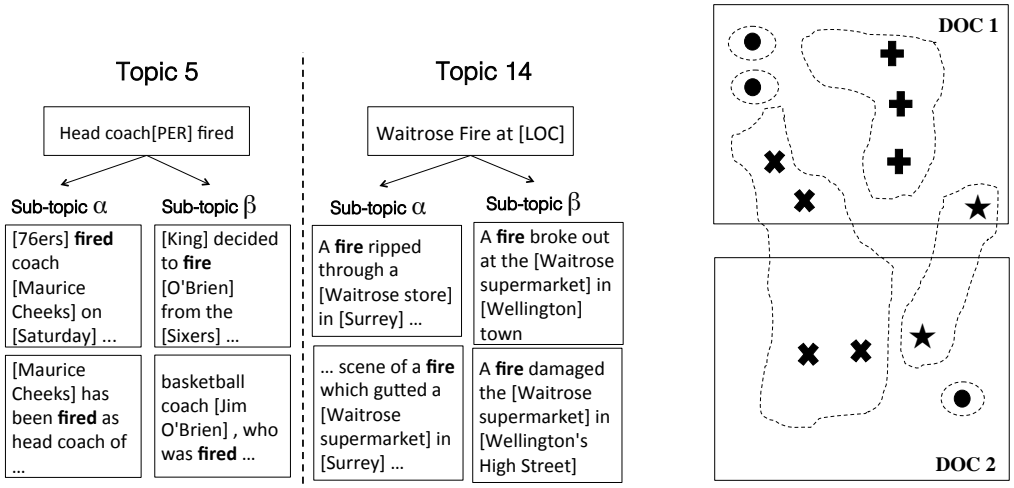
We view CDEC as a clustering task aimed at event-specific information aggregation. The clustering generated by a CDEC system allows one to examine all appearances of an event over a large corpus, shedding light on how the same event gets described in different documents.

We first describe our notation and the layout of the ECB+ dataset, and use Figures 2a and 2b to illustrate our definitions. Some statistics for the splits are shown in Table 1. We use the train, dev and test splits of Yang et al. (2015).

Event Mention A phrase which describes the action associated with an event. eg, in Figure 2a, *fired* is the event mention of the event "76ers fired coach Maurice Cheeks". We denote event mentions in **bold**.

Event Argument An event can have several arguments associated with it, describing its participants, location, or time of occurrence. eg., in Figure 2a, for the event "76ers fired coach Maurice Cheeks", "76ers" and "Maurice Cheek" are participant arguments and "Saturday" is the time argument. We collectively refer to these as the event arguments associated with the event mention (shown in [brackets]).

Event An event mention together with its arguments constitute an event, which describes an action and its arguments (location, time, participant etc.). An events in ECB+ can be in one of 3 categories:



(a) Layout of the ECB+ corpus.

(b) Event Types

Figure 2: (a) Documents in ECB+ are divided into topics and sub-topics, and coreference links can exist across documents in the same sub-topic. However, a coreference system is not aware of the topic (or sub-topic) partition at test time. (b) Coreference clusters are enclosed by a dotted line. Solid circles denote singleton mentions, solid crosses and stars denote cross document mentions, and solid plus denotes within document mentions.

(a) *Singleton Event* - an event with only one mention in the entire corpus. Shown as solid circles in Figure 2b, **(b)** *Within-Document (WD) Event* - an event with multiple mentions all of which appear in *one* document *only*. Shown as solid plus signs in Figure 2b, **(c)** *Cross-Document (CD) Event* - an event with multiple mentions spanning several documents. Shown as solid crosses and stars in Figure 2b.

Two events are *coreferent* if they represent the same situation. In particular, the text representation of the event should involve the same action and (some of the) arguments (Yang et al., 2015).

3.1 The ECB+ Corpus

In this section we discuss the specifics and layout of the ECB+ corpus.

Topics The documents in ECB+ are partitioned into several topics $\{T_1, T_2, ..., T_k\}$ each of which contains documents describing events of the same *event type*. For example, topic T_5 contains documents describing the firing of the head coach of a sports team. Figure 2a shows topics T_5 and T_{14} and the event types they described ("Head Coach [PER] fired" and "Waitrose Fire at [LOC]").

Sub-Topics A collection of documents that describe the same event within a topic, constitute a sub-topic. For example, the α sub-topic in topic 5 in Figure 2a, contains all documents describing the "firing of Maurice Cheeks", while those in the β sub-topic describe the "firing of O'Brien". Every topic T_i contains two sub-topics.[1] The documents in the β sub-topic were added by Cybulska and Vossen (2014) to increase the difficulty of the task, as a naive cross document event coreference system may incorrectly link a mention of the "O'Brien fire" event with a mention of the "Cheek fire" event.

3.2 Problem Formulation

Input: A collection of documents, each of which containing several event mentions.
Output: The system outputs an assignment of a cluster-id to each event mention, such that event mentions belonging to different documents but sharing the same cluster-id are coreferent.

Note that at test time, the CDEC system is not aware of the layout of the dataset so that it does not exploit the partitioning of documents into topics and sub-topics when making coreference decisions. As a result, a system can (incorrectly) link two event mentions across topics (or sub-topics), for which it should be appropriately penalized in evaluation. However, we will see in the next section that current evaluations do not meet this requirement.

[1] In principle, we can have more than 2 sub-topics per topic, but this does not occur in ECB+.

	Train	Dev	Test	Total
docs	462	73	447	982
topics	20	3	20	43
sub-topics	40	6	40	86
ev. mentions	5443	608	8951	14874
singleton	1866	167	5572	7605
WD	23	0	89	112
CD	3554	441	3290	7285
WD chains	2502	316	2138	4956
CD chains	691	47	479	1217

Table 1: Statistics of the ECB+ dataset. WD, CD refer to within document and cross document mentions respectively. WD (CD) Chains count the number of mention clusters that refer to the same event within (across) document(s). See text for details.

4 Evaluation Settings

All evaluation settings transform the cross document coreference problem to a within document coreference problem, by merging documents to create meta-document(s), such that cross document event coreference chains correspond to within document chains in the meta-document(s). We first revisit the evaluation settings that have been used in previous work. All these rely on the layout of the corpus, which affects their strictness.

Bejan and Harabagiu (2014) (B&H) In this case, the documents belonging to each topic are merged together to form a single meta-document $M(T_i)$. In this way, we have k meta-documents, one for each topic. Each meta-document is then evaluated separately for within document coreference, and the scores are aggregated by taking the micro-average. This evaluation is also followed by other works (Cybulska and Vossen, 2014; Cybulska and Vossen, 2015).

In this setting each topic's meta-document is evaluated in isolation, assuming that there were no coreference links made by a system across topics. However, a system can potentially link mentions across topics, as the input to the system does not describe the topic (or sub-topic) layout. As a result, this evaluation is oblivious to incorrect coreference made across topics. For example, it will ignore a (incorrect) link between the fire event mentions in topic T_5 and topic T_{14} in Figure 2a.

Yang et al. (2015) (YCF) In their case, all documents which belong to the same sub-topic (say α) of topic T_i are merged to create a meta-document $M(T_i, \alpha)$ for each sub-topic. Each such meta-document is then evaluated separately for within document coreference. In addition, all singleton event mentions are removed from the keys of all documents. [2]

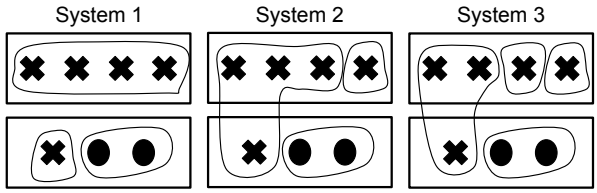
This setting implicitly makes two simplifying assumptions. First, it ignores singletons (referred to as I.S.) during evaluation, this making the coreference task considerably simpler (singletons constitute over 60% of the test mentions) as the system does not get penalized for making incorrect links to mentions of singleton events. Furthermore, by creating separate meta-document per sub-topic, this setting disregards the inclusion of documents from the β sub-topic into the corpus (see § 3), since incorrect coreference links across the sub-topics will not be penalized. This limitation (referred in Table 2 as S.T.) is similar, but more lenient, to the one in the B&H setting. (see § 7).

4.1 Our Proposals

We now discuss our proposed evaluation methodologies which address the limitations discussed above.

Proposal 1 (SIMPLE-CDEC) In this case, we merge all documents, across topics, into a single meta-document M. Unlike (Yang et al., 2015), we consider all event mentions (including singletons) in the corpus, and do not create a separate meta-document for each sub-topic, making our evaluation closer to what Cybulska and Vossen (2014) envisioned.

[2] We confirmed the YCF setting through personal communication with the authors and also examining the key and response files they provided.



(a) Enclosing rectangles denote documents, solid crosses (and circles) denote coreferent mentions of the same event (gold clustering). Polygon regions denote system response.

	MUC	B^3	$CEAF_e$	BLANC
System 1	88.8	87.0	75.5	80.5
System 2	88.8	87.0	75.5	80.5
System 3	74.9	74.9	58.3	63.7

(b) Performance of different systems as evaluated under SIMPLE-CDEC. Systems performing WD coreference are awarded equally or more than system performing CD coreference.

Figure 3: (a) Outputs of three different systems. System 1 only performs within-document (WD) coreference. System 2 and 3 performs cross-document(CD) coreference and correctly identify the cross document link of the solid cross event, but make mistake in the WD coreference. (b) SIMPLE-CDEC evaluation awards System 1 and System 2 same scores, and System 3 lower scores. Note that this issue is not limited to SIMPLE-CDEC, but also B&H and YCF.

SIMPLE-CDEC fulfills the requirement that the evaluation of a CDEC system should not be aware of the corpus layout into topics (as in ECB+) and all output links should be appropriately evaluated, unlike what is currently done in B&H and YCF. [3]

However, the SIMPLE-CDEC setting (just like B&H and YCF) does not distinguish the cross document performance from the within document performance. For example, in Figure 3a, a system which performs only within document coreference can get the same score as a system that performs both within and across document coreference (see Table 3b). However, it is desirable to have an evaluation setting that facilitates this distinction, and focuses on the ability of a CDEC system to aggregate information *across* documents in a large corpus, that is, evaluate on purely cross document coreference links. A CDEC system which performs within document event coreference only is not fully exploiting the large corpus and does not discover links between documents.

Proposal 2 (PURE-CDEC) In this setting, we first preprocess each document so that all within document mentions of the same event are reduced to a single meta-mention in that document. In this way, we discount the within document coreference links and focus on the cross document coreference links. Then, we follow the SIMPLE-CDEC setting to evaluate a single meta-document M generated from the preprocessed system response and the gold key documents.

5 Evaluation Metrics

We briefly describe the coreference evaluation metrics that are used in our experiments.

MUC (Vilain et al., 1995) A link-level metric, MUC counts the minimum number of edge-insertions or edge-deletions required to obtain the gold clustering (key) from the predicted clustering (response). A known limitation of MUC is that it does not reward a system for correctly identifying singletons.

B^3 (Bagga and Baldwin, 1998) A mention-level metric, B^3 calculates precision and recall for each mention by measuring the proportion of overlap between the predicted and gold coreference chains. The final score is an average over the scores for all mentions. Although it overcomes the limitations of MUC, it uses mentions of the same entity (coreference chain) more than once.

$CEAF_e$ (Luo, 2005) An entity-level metric which first finds an optimal alignment between entities in the key to the entities in the response by maximizing an entity similarity objective. This alignment is then used to calculate the CEAF precision and recall.

Blanc (Luo et al., 2014) First described in (Recasens and Hovy, 2011) and later extended in (Luo et al., 2014). Blanc is based on the Rand Index (Rand, 1971), and computes two F-scores, one evaluating the quality of coreference decisions (Pairwise) and another evaluating the quality of the non-coreference

[3] While evaluation should not rely on the corpus layout, we do not in any way suggest that this is the ideal approach for performing coreference. Indeed, considering coreference decisions over the entire corpus will be prohibitively expensive.

decisions (Pairwise-Negative). The Pairwise (PW) and the Pairwise-Negative (PWN) F-scores are then averaged to compute the final Blanc F-score.

Each of the above metrics have some drawbacks. B^3 and CEAF scores rapidly approach 100 if many singletons are present, and the same can drive the Blanc-PWN score to dominate the Blanc-PW score (Recasens and Hovy, 2011). This is why the entity coreference literature reports the CoNLL average, which is average of MUC, B^3 and $CEAF_e$ F-scores (Pradhan et al., 2011). Following previous work, we report CoNLL F1 and also the F-score averaged across all metrics.

Coreference Scorers For running evaluations under the YCF and B&H setting we use the standard within-document coreference scorer of Pradhan et al. (2014). However, in the SIMPLE-CDEC and PURE-CDEC setting, creating a single meta-document for the entire test split leads to a document with over 8000 mentions, which causes runtime issues for metrics like $CEAF_e$ and Blanc with Pradhan et al. (2014)'s scorer.[4] We use Hachey et al. (2014)'s scorer instead, which provides more efficient implementations.[5]

6 Baselines

We evaluate the following event coreference baseline models under the different evaluation settings,

Lemma In this model two events are coreferent if the head lemmas of their event mentions match.

Lemma-WD In this model we consider two event mentions coreferent if their head lemmas match and they belong to the same document. This is the within document variant of the Lemma model.

Lemma-δ In this model two event mentions are considered coreferent only if their head lemmas match and the *tf-idf* document similarity of the documents containing them exceeds a threshold (we use $\delta = 0.3$ after tuning on dev). For $\delta = 0$ this reduces to the Lemma baseline.

Supervised Agglomerative Clustering (SAC) Performs greedy agglomerative clustering, using a learned pairwise classifier to score if two cluster of mentions are coreferent. Inter-cluster similarity score is computed as the average of the similarity score of their mentions. Like Lemma-δ, two clusters are considered for linking if the document similarity exceeds a threshold (we use 0.3 as above).

We re-implemented the pairwise classifier of Yang et al. (2015) using the same feature set. Mention heads were found using dependency parses obtained by Stanford CoreNLP (Manning et al., 2014). We identify the Framenet (Baker et al., 1998) frame evoked by an event mention, we use the SEMAFOR (Das and Smith, 2011). For identifying the arguments for an event mention we use Illinois-SRL (Punyakanok et al., 2008). The pairwise classifier was trained using the Illinois-SL package (Chang et al., 2015).

7 Experiments

We first show the limitations of using current evaluation settings by showing how they can produce a wide range of results for the same baseline system. Next, we show the value of isolating the performance on predicting cross document coreference links. Finally, we assess the quality of the dataset and describe how we semi-automatically detected different types of annotation errors. Since we focus here on evaluation and not on developing an end-to-end CDEC system, we use the gold event mentions provided.

7.1 Comparing Evaluation Settings

We evaluate the lemma baseline under B&H, YCF, and SIMPLE-CDEC as described in §4. We aim to show that the same baseline approach can achieve highly variable results depending on the evaluation setting employed. We also isolate the effect of the two assumptions made by the YCF setting – first by creating a meta-document per sub-topic (S.T. in Table 2), and then by ignoring singletons (I.S. in Table 2) in SIMPLE-CDEC setting.

The results are shown in Table 2. Compared to SIMPLE-CDEC, all other settings assign the lemma baseline unrealistically high scores. B&H does not penalize links predicted across topics and therefore the F-score of the lemma baseline across all metrics increases. In addition to above, S.T. does not

[4]The $CEAF_e$ and Blanc evaluation did not finish after > 30 hours.

[5]Available at `github.com/wikilinks/neleval`.

Eval. Setting	MUC			B^3			$CEAF_e$			Blanc			CoNLL Avg of 4	
	P	R	F	P	R	F	P	R	F	PW	PWN	F	avg.	avg.
Settings that include singletons.														
SIMPLE-CDEC	30.5	75.7	43.4	28.4	81.9	42.2	65.6	18.6	29.0	11.1	**98.4**	54.8	38.2	42.3
B&H	37.4	75.3	50.0	49.4	81.6	61.5	39.9	**70.8**	51.0	30.5	93.4	62.0	54.2	56.1
S.T.	**40.9**	**75.9**	**53.2**	**58.8**	**82.7**	**68.7**	**71.3**	45.7	**55.7**	**37.0**	95.4	**66.2**	**59.2**	**61.0**
Settings that ignore singletons.														
I.S.	79.5	**76.1**	77.8	50.7	**54.0**	52.3	**39.9**	46.7	43.1	31.0	98.4	64.7	57.7	59.5
YCF (=IS+ST)	**94.5**	75.8	**84.2**	**92.0**	53.6	**67.8**	36.2	**75.2**	**48.9**	**53.3**	**98.7**	**76.0**	**67.0**	**69.2**

Table 2: **Evaluating the lemma baseline in different settings**. S.T. creates a separate meta-document for each sub-topic, while I.S. ignores singleton event mentions during evaluation. Note that the evaluation becomes more lenient from top to bottom, with SIMPLE-CDEC being the most strict. Best values in each column under different settings are shown in **bold**.

Baseline	MUC			B^3			$CEAF_e$			Blanc			CoNLL Avg of 4	
	P	R	F	P	R	F	P	R	F	PW	PWN	F	avg.	avg.
SIMPLE-CDEC														
Lemma-WD	38.0	20.4	26.5	**88.7**	68.4	**77.2**	67.5	**80.8**	**73.6**	5.3	**98.5**	51.9	59.1	57.3
Lemma	30.5	**75.7**	43.4	28.4	**81.9**	42.2	65.6	18.6	29.0	11.1	98.4	54.8	38.2	42.3
Lemma-δ	40.9	72.5	**52.3**	59.0	81.1	68.3	**73.6**	45.5	56.2	**32.8**	**98.5**	**65.6**	58.9	60.6
SAC	**44.2**	52.9	48.2	75.2	76.0	75.6	70.5	62.6	66.3	28.6	**98.5**	63.5	**63.4**	**63.4**
PURE-CDEC														
Lemma-WD	0.0	0.0	0.0	**90.1**	65.6	**75.9**	67.0	**80.2**	**73.0**	0.0	76.2	38.1	49.6	46.8
Lemma	18.7	**62.7**	28.8	27.2	**74.8**	39.9	65.7	18.6	29.0	7.2	76.2	41.7	32.6	34.9
Lemma-δ	29.4	42.4	34.7	68.6	71.6	70.1	70.6	56.8	62.9	**26.5**	76.2	51.3	53.0	52.6
SAC	**30.5**	42.0	**35.3**	70.6	74.5	72.5	**71.2**	63.2	67.0	25.3	**79.0**	**52.2**	**58.3**	**56.7**

Table 3: **Comparing the baseline approaches under SIMPLE-CDEC and PURE-CDEC**. Best values in each column under different settings are shown in **bold**.

penalize cross sub-topic predictions as well, which boosts the performance further. This confirms that settings like B&H do not penalize certain incorrect coreference links by exploiting the corpus layout.

Next, we compare the settings that ignore singleton event mentions (which constitute over 60% of the test data) during evaluation. When ignoring singletons (I.S.), the F-score for all metrics improves compared to SIMPLE-CDEC. Under I.S., incorrect coreference links to singleton events are discounted, resulting in high scores. Finally, with YCF, which combines the assumptions of I.S. and S.T., the scores again improve dramatically.

This experiment clearly shows that the assumptions made by B&H and YCF lead to lenient evaluations. SIMPLE-CDEC is a more appropriate evaluation setting since it remains unaware of the corpus layout and correctly penalizes all incorrect coreference link predictions when evaluating a CDEC system.

7.2 Understanding Cross Document Coreference Performance

In this experiment we evaluate all the baselines using SIMPLE-CDEC and PURE-CDEC. The aim is to show that if we do not evaluate the cross document coreference separately, a system can exploit the evaluation by only focusing on correctly predicting within document coreference links.

The results are shown in Table 3. Lemma-WD predicts lots of singletons as it only links mention in the same document. This results in high B^3 and CEAF scores in both SIMPLE-CDEC and PURE-CDEC since over 60% event mentions in the test data are singletons.

For SIMPLE-CDEC, the lemma-WD baseline performs surprisingly well, second to the SAC model. All metrics reward Lemma-WD for identifying the within document links, which leads to the high CoNLL score. Surprisingly, its CoNLL score is slightly higher than Lemma-δ, which is a model which does both within and across linking. This is because the latter may incorrectly link some singletons across documents. Note that this behavior is not exclusive to the SIMPLE-CDEC – a system which exclusively

predicts within document links can achieve such high scores under the B&H and YCF settings.

However, we can get a clearer assessment of cross document coreference performance by evaluating the models under the PURE-CDEC setting. As this evaluation only rewards finding correct cross document links, Lemma, Lemma-δ and SAC model are the only models which get non-zero MUC and Blanc-PW scores. As a result, the Lemma-WD average F-score drops considerably (by almost 10%) in comparison to the Lemma-δ and SAC model (around 6%), revealing that these models indeed do better cross document coreference. It is evident that to accurately assess the cross document coreference performance, we should report *both* PURE-CDEC and SIMPLE-CDEC results.

It should be noted that Lemma performs worse than Lemma-WD because it makes incorrect cross document links, due to the naive nature of the lemma match. On the other hand, Lemma-WD does not make *any* across document links, avoiding incurring these penalties. It gets rewarded for "identifying" singletons as described earlier in SIMPLE-CDEC. This conservative nature of Lemma-WD gets better average scores than Lemma, even in the PURE-CDEC setting. Overall, SAC and Lemma-δ are the two best models in Table 3.

Choice of Baseline Besides the aforementioned insights, Table 3 offers a better choice of baselines. Bejan and Harabagiu (2014) and Yang et al. (2015) claimed that Lemma is a strong baseline for CDEC. We believe that this claim held in these works only due to the lenient evaluation settings of B&H and YCF, which did not appropriately penalize the incorrect across topic (and sub-topic) links made by the Lemma baseline. However, the SIMPLE-CDEC and PURE-CDEC evaluations show that Lemma-δ is a stronger baseline. For future comparisons, using Lemma-δ as a baseline is more appropriate.

7.3 The Annotation Quality of ECB+

Evaluating with singletons also helped in discovering annotation errors in the dataset. In addition to identifying annotation errors, as described below, we found that several documents were partially annotated,[6] which is consistent with a similar observation made in (Liu et al., 2014).

We used the approach of Goldberg and Elhadad (2007) to semi-automatically detect annotation errors, by training an anchored SVM. First, for each pair of mention (m_i, m_j) in the training data, we added a unique anchor feature a_{ij}, thus making the data linearly separable. Next, we trained a SVM classifier on all of the data with a high penalty parameter C. The classifier uses the anchor features to memorize the hard to classify examples, which are either genuine hard coreference pairs, or incorrect annotations. By thresholding the features weights for the anchor features $|a_{ij}| > \delta$ (we use $\delta = 0.95$), we generated a short-list of these hard cases, which we then examined by a annotator for mistakes. The errors we found can be categorized as one of:

Missing Singleton-to-Singleton Link Two gold event mentions which should have been marked as coreferent, but were marked as singletons. For example:

Aceh was hit extremely hard by the massive Boxing Day **earthquake** and **tsunami** in 2004, killing 170,000 people
A massive **quake** struck off Aceh in 2004 , sparking a **tsunami** that killed 170,000 people ...

Both mention pairs (*earthquake*,*quake*) and (*tsunami*,*tsunami*) refer to the same event, but their coreference links were missing in the annotation. Both the enclosing documents belong to the same topic.

Missing Singleton-to-Cluster Link A singleton event mention which should have been linked to an existing cluster of mentions describing the same event.

LaRue , a Mississippi oil heir who became the first person found guilty of participating in the **Watergate** coverup
... strategy for capturing Southern votes and then a significant participant in the **Watergate scandal** .

The *Watergate scandal* mention is marked as singleton. However, the watergate scandal event appears in several other documents to which the *Watergate* mention is marked as being coreferent. Again, both the mentions belong to documents in the same topic.

We found over 300 such annotation errors which were incorrectly not linking singleton mentions. The list of errors detected is available at `http://cogcomp.cs.illinois.edu/page/publication_view/801`.

[6]For example, Only 5 sentences out of 40 in a document were marked with events.

8 Conclusion

Accurate evaluation and high-quality annotations are crucial to our ability to measure progress in any task. We showed that past work for CDEC have resorted to widely different evaluation approaches, making several implicit assumptions, which simplify the coreference task and lead to overlooking coreference mistakes. In particular, as we showed, excluding singleton mentions from the evaluation does not seem justified. Furthermore, current evaluation methods heavily rely on the corpus being organized into topics and sub-topics, but this may not always be available, especially for evaluation corpora. To accurately measure CDEC performance, it is necessary to drop these assumptions. We recommend that future evaluations report results using both SIMPLE-CDEC and PURE-CDEC settings.

Beyond these assumptions, the annotations in the current dataset are incomplete in several respects. Indeed, we showed that over 300 annotation errors in the dataset can be detected semi-automatically, and also noted that many documents were partially annotated. As this dataset is presently the only dataset with cross document coreference annotations for events, such annotation errors make evaluation difficult. We believe that to correctly evaluate this task and make progress, efforts must be made to create thoroughly annotated datasets with high quality annotations.

Acknowledgements

Thanks to Bishan Yang, Piek Vossen and Joel Nothman for answering questions and sharing system outputs. This work was supported by Contract HR0011-15-2-0025 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. 1998. Topic detection and tracking pilot study final report.

Jun Araki, Zhengzhong Liu, Eduard H Hovy, and Teruko Mitamura. 2014. Detecting subevent structure for event coreference resolution. In *LREC*.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*.

Amit Bagga and Breck Baldwin. 1999. Cross-document event coreference: Annotations, experiments, and observations. In *the Workshop on Coreference and its Applications*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *ACL-COLING*.

Cosmin Bejan and Sanda Harabagiu. 2010. Unsupervised event coreference resolution with rich linguistic features. In *ACL*.

Cosmin Adrian Bejan and Sanda Harabagiu. 2014. Unsupervised event coreference resolution. *Computational Linguistics*.

E. Bengtson and D. Roth. 2008. Understanding the value of features for coreference resolution. In *EMNLP*.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *EMNLP*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL*.

Kai-Wei Chang, Shyam Upadhyay, Ming-Wei Chang, Vivek Srikumar, and Dan Roth. 2015. IllinoisSL: A JAVA library for structured prediction. In *Arxiv Preprint*, volume abs/1509.07179.

Agata Cybulska and Piek Vossen. 2014. Using a sledgehammer to crack a nut? lexical diversity and event coreference resolution. In *LREC*.

Agata Cybulska and Piek Vossen. 2015. Translating granularity of event slots into features for event coreference resolution. In *Workshop on Events*.

Naomi Daniel, Dragomir Radev, and Timothy Allison. 2003. Sub-event based multi-document summarization. In *the HLT-NAACL 03 on Text summarization workshop-Volume 5*.

Dipanjan Das and Noah Smith. 2011. Semi-supervised frame-semantic parsing for unknown predicates. In *ACL*.

Q. Do, Y. Chan, and D. Roth. 2011. Minimally supervised event causality identification. In *EMNLP*.

Yoav Goldberg and Michael Elhadad. 2007. SVM model tampering and anchored learning: A case study in hebrew NP chunking. In *ACL*.

Ben Hachey, Joel Nothman, and Will Radford. 2014. Cheap and easy entity evaluation. In *ACL*.

Kevin Humphreys, Robert Gaizauskas, and Saliha Azzam. 1997. Event coreference for information extraction. In *a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *EMNLP-CoNLL*.

Zhengzhong Liu, Jun Araki, Eduard H Hovy, and Teruko Mitamura. 2014. Supervised within-document event coreference using information propagation. In *LREC*.

Xiaoqiang Luo, Sameer Pradhan, Marta Recasens, and Eduard Hovy. 2014. An extension of BLANC to system mentions.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *EMNLP*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL: System Demonstrations*.

James Mayfield, David Alexander, Bonnie J Dorr, Jason Eisner, Tamer Elsayed, Tim Finin, Clayton Fink, Marjorie Freedman, Nikesh Garera, Paul McNamee, et al. 2009. Cross-document coreference resolution: A key technology for learning by reading. In *AAAI Spring Symposium: Learning by Reading and Learning to Read*.

Srini Narayanan and Sanda Harabagiu. 2004. Question answering based on semantic structures. In *COLING*.

V. Ng and C. Cardie. 2002. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *COLING*.

Haoruo Peng, Kai-Wei Chang, and Dan Roth. 2015. A joint framework for coreference resolution and mention head detection. In *CoNLL*.

Haoruo Peng, Yangqiu Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *EMNLP*, page 11. ACL.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *CoNLL*.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *ACL*.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2).

William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*.

Marta Recasens and Eduard Hovy. 2011. BLANC: Implementing the rand index for coreference evaluation. *Natural Language Engineering*.

Rui Sun, Yue Zhang, Meishan Zhang, and Donghong Ji. 2015. Event-driven headline generation. In *ACL*.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *the 6th conference on Message understanding*.

Travis Wolfe, Mark Dredze, and Benjamin Van Durme. 2015. Predicate argument alignment using a global coherence model. In *NAACL*.

Bishan Yang, Claire Cardie, and Peter Frazier. 2015. A hierarchical distance-dependent bayesian model for event coreference resolution. *TACL*.

Tongtao Zhang, Hongzhi Li, Heng Ji, and Shih-Fu Chang. 2015. Cross-document event coreference resolution based on cross-media features. In *EMNLP*.

Modeling Discourse Segments in Lyrics Using Repeated Patterns

Kento Watanabe[1], **Yuichiroh Matsubayashi**[1], **Naho Orita**[1], **Naoaki Okazaki**[1],
Kentaro Inui[1], **Satoru Fukayama**[2], **Tomoyasu Nakano**[2],
Jordan B. L. Smith[2], **Masataka Goto**[2]
[1]Graduate School of Information Sciences, Tohoku University
[2]National Institute of Advanced Industrial Science and Technology (AIST)
{kento.w,y-matsu,naho,okazaki,inui}@ecei.tohoku.ac.jp,
{s.fukayama,t.nakano,jordan.smith,m.goto}@aist.go.jp

Abstract

This study proposes a computational model of the discourse segments in lyrics to understand and to model the structure of lyrics. To test our hypothesis that discourse segmentations in lyrics strongly correlate with repeated patterns, we conduct the first large-scale corpus study on discourse segments in lyrics. Next, we propose the task to automatically identify segment boundaries in lyrics and train a logistic regression model for the task with the repeated pattern and textual features. The results of our empirical experiments illustrate the significance of capturing repeated patterns in predicting the boundaries of discourse segments in lyrics.

1 Introduction

Lyrics are an important element of popular music. They provide an effective means to express the message and emotion of music. Similar to prose text, lyrics are a discourse: i.e., they are, typically, a sequence of related lines, rather than an unconnected stream of lines of arbitrary order. Thus, like texts, lyrics also have a discourse structure consisting discourse segments. Each discourse segment exhibits an individual topic in discourse, and the transition of topics over successive discourse segments constitutes a flow (Austin et al., 2010; Watanabe et al., 2014). Unlike prose text, lyrics have their own peculiar properties, such as frequent repetition of identical or similar phrases and extensive use of rhyme and refrain (Austin et al., 2010), as illustrated in Figure 1. Analogous to prose text, the lyrics in Figure 1 can be viewed as a sequence of discourse segments, wherein each segment is depicted by a colored box. Segments ① and ③ appear repeatedly (e.g., segments ④ and ⑧ are identical to segment ①), which is not typically observed in prose text.

Our goal is to reveal the discourse structure of lyrics in popular music by quantitatively analyzing a large-scale lyrics corpus. The motivations behind the goal are as follows. First, there are hardly any studies that have focused on the data-oriented research of the discourse structure of lyrics; however, multiple studies have focused on the structure of music audio (Paulus et al., 2010, for a review). Second, a better understanding of the nature of lyrics structure combined with the existing theories of music audio could lead to a more comprehensive theory of the overall nature of popular music. Third, understanding the nature of lyrics structure will allow us to devise a variety of useful computer systems, such as systems that automatically generate lyrics, assist human lyrics writers, or evaluate the quality of lyrics.

As a first but crucial step toward achieving this goal, this study explores the nature of the discourse structure of lyrics with a focus on the rbyof repeated patterns as an indicator of segment boundaries. We choose discourse segments as our primary focus because exploring discourse segments is a necessary step toward a comprehensive understanding of lyrics structure. To address this issue, we consider the task of computationally predicting the boundaries of discourse segments in lyrics under the assumption that a better prediction model would allow us to better understand the nature of the discourse structure of lyrics.

By conducting a large-scale data analysis, we examine our primary hypothesis that *discourse segments in lyrics strongly correlate with repeated patterns*. For example, if a sequence of lines in lyrics has a

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1959–1969, Osaka, Japan, December 11-17 2016.

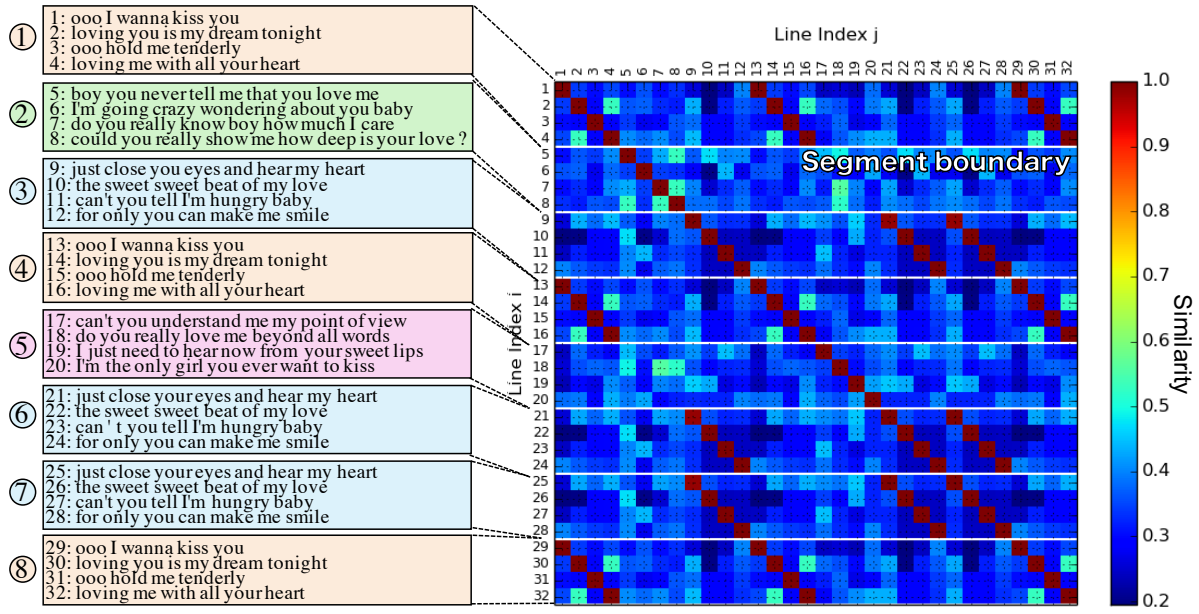


Figure 1: An example of lyrics and corresponding self-similarity matrix (title of lyrics: *How Deep Is Your Love?* (RWC-MDB-P-2001 No. 81 from RWC Music Database (Goto et al., 2002)))

repetition, such as *abcdefabc* (each letter represents a line, with repeated letters being repeated lines), we expect the boundaries of the discourse segments tend to agree with the boundaries of the repeated parts as in |*abc*|*def*|*abc*|, where "|" indicates a boundary.

To examine the extent to which these repeated patterns capture the segment structure of lyrics, we use a large-scale corpus of popular music lyrics that contains more than 140,000 songs. This is, to the best of our knowledge, the first study that takes a data-driven approach to exploring the discourse structure of lyrics in relation to repeated patterns.

One issue to be addressed before conducting the corpus study is that no existing corpus has annotated the discourse structure of lyrics. In this study, we preliminarily assume that discourse segment boundaries are indicated by empty lines inserted by lyrics writers. We admit that empty lines may not be "true" discourse segment boundaries and discourse segments may exhibit a hierarchical structure (e.g., verse–bridge–chorus structure). These issues could be better addressed by combining the analysis of the discourse structure of lyrics with the structural analysis of music. We believe this direction of research will open an intriguing new field for future exploration.

The remainder of this study is organized as follows. Section 2 reviews related work into the discourse structure of lyrics, with particular focus on the segmentation using repeated patterns. Section 3 presents the first quantitative analysis of the distribution of repeated lines and segments in lyrics and suggests cues that could help to identify the segment boundaries. Section 4 describes our computational model, which predicts the boundaries of discourse segments in lyrics using repeated patterns. Section 5 presents experimental results that show the importance of repeated patterns in predicting the boundaries of discourse segments in lyrics. Section 6 concludes and discusses future work. Note that throughout this study, we use the term *segment* to refer to a discourse segment in lyrics.

2 Related work

This section reviews related work into the discourse structure of lyrics, with particular focus on the segmentation of lyrics using repeated patterns.

Text segmentation is a classic text retrieval problem, and there exists a rich body of research into text segmentation in natural language processing. Various linguistic cues have been suggested to identify text boundaries such as expressions that frequently appear at the end of segments (Beeferman et al., 1999), contextual/topical changes (Choi, 2000; Malioutov and Barzilay, 2006; Riedl and Biemann, 2012),

and word/entity repetition (Kan et al., 1998; Reynar, 1999).

Although we share the same motivation as these studies, these text segmentation methods do not consider repeated patterns of phrasal segments because this type of repetition is nearly always absent in prose text. On the other hand, segments in lyrics often have repetitions (Austin et al., 2010) as shown in Section 3.1. We aim to capture the segment structure of lyrics using repeated patterns.

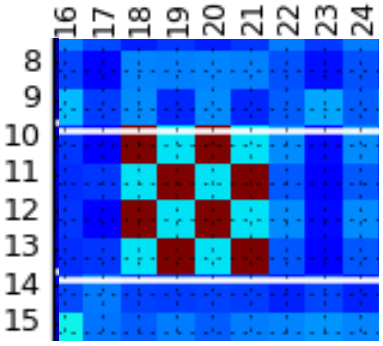


Figure 2: Negative example against pattern (1)

Previous computational work into lyrics segmentation has focused on identifying the segment labels of lyrics that are already segmented. For example, the structure of lyrics can be represented using labels A–B–C–A–B in which each letter refers to a group of lines; e.g., A might represent a chorus that appears twice. Barate et al. (2013) proposed a rule-based method to estimating such structure labels of *segmented* lyrics. Our task differs from this task in that we aim to estimate the segment boundaries of *unsegmented* lyrics using machine learning techniques.

In contrast to the segmentation of lyrics, much previous work has analyzed and estimated the segment structure of music audio signals using repeated musical parts such as verse, bridge, and chorus (Foote, 2000; Lu et al., 2004; Goto, 2006; Paulus and Klapuri, 2006; McFee and Ellis, 2014). To automatically identify these repeated musical parts in music audio signals, a self-similarity matrix (SSM) as shown in Figure 1 is often used. Repeated segments lead to high-valued lines in the off-diagonals of the matrix, and these patterns are used to identify the structure. To capture segments in lyrics using repeated patterns, we apply the SSM to lyrics. Lyrical repetition is known to be an important property of lyrics (Austin et al., 2010), and we expect that repetition patterns would also appear in lyrics as they do in audio signals.

In summary, no previous computational work has exactly focused on the segmentation of lyrics using repeated patterns. Section 3 presents the first quantitative analysis of the distribution of repeated lines and segments in lyrics, and suggests cues that help identify segment boundaries.

3 Statistics of repeated patterns and segment boundaries

As an initial step toward modeling the discourse structure of lyrics, we examine the distribution of segments in lyrics by focusing on repeated patterns. We first show the basic distributions of lyrics and suggest potential cues to indicate segment boundaries in lyrics. To examine the distribution of repeated patterns in lyrics and their relation to segment boundaries, we use a large scale lyrics database that contains 144,891 songs[1].

3.1 The basic distribution of lyrics

Among the 144,891 songs in the lyrics database, there are 5,666,696 lines and 969,176 segments in total, with segment breaks inferred from empty lines. Per song, there are 39.11 lines and 6.69 segments on average. Most songs have at least one repeated line (84.79% using an exact criterion; 90.34% using a lenient matching criterion of normalized edit distance ≥ 0.8, explained in the next section). A fair number of songs also have at least one repeated segment (exact match: 37.73%, lenient match: 54.57%). Per song, 13.73 lines and 0.52 segments (both exact match) are repeated at least once on average. These distributions show that repetition of lines and segments occurs frequently in lyrics, in line with our expectations. Next, we suggest potential repeated patterns to help in identifying segment boundaries.

3.2 Correlation between repeated patterns and segment boundaries

To examine what kinds of repeated patterns would help identify segments in lyrics, we use the SSM, similar to previous work into the segmentation of music audio signals (Section 2). Figure 1 shows an example SSM. Throughout this study, we represent the i^{th} line in lyrics as l_i ($1 \leq i \leq L$), where L is the number of lines of the lyrics. A degree of similarity between l_i and l_j, i.e., $\text{sim}(l_i, l_j)$, is represented

[1]Music Lyrics Database. http://www.odditysoftware.com/page-datasales1.htm

Repeated pattern	Prior and conditional probabilities	Value
Pattern (1)	P(Boundary appears)	0.1455 (824286/5666696)
	P(Boundary appears | at the starting/ending of a diagonal line)	**0.2218** (339020/1530824)
Pattern (2)	P(Boundary does not appear)	0.8545 (4842410/5666696)
	P(Boundary does not appear | within a diagonal line)	**0.9273** (751195/810098)
Pattern (3)	P(Adjacent lines appear within a segment)	0.8507 (4697520/5521806)
	P(Adjacent lines appear within a segment | adjacent lines are similar)	**0.9439** (218524/231518)
Pattern (4)	P(Boundary appears after l_i)	0.1455 (824286/5666696)
	P(Boundary appears after l_i | l_i is similar to the last line of lyrics)	**0.4230** (125659/297069)
	P(Boundary appears after l_i | l_{i+1} is similar to the first line of lyrics)	**0.4189** (46531/111079)

Table 1: Correlation between each repeated pattern and segment boundary

as an intensity at a cell where the i^{th} row and j^{th} column overlap. Using the normalized edit distance $\text{NED}(l_i, l_j)$, we compute the degree of similarity (Yujian and Bo, 2007): $\text{sim}(l_i, l_j) = 1 - \text{NED}(l_i, l_j)$. The red diagonal lines in Figure 1 are the result of exact line repetitions [2]. The white horizontal lines in Figure 1 indicate the true segment boundaries.

After manually examining more than 1,000 lyrics and their SSMs, we suggest the following four types of repeated patterns as indicators of segment boundaries.

(1) The Start and end points of a diagonal line are segment boundaries. Some repeated segments correspond to the red diagonal lines. For example, in Figure 1, segment ① is repeated twice (segments ④ and ⑧), and each repetition ,starting at l_{13} and l_{29}, can be observed as a diagonal line from l_1 to l_4. This suggests that some segments could be divided at the start and end points of such a diagonal line.

(2) A segment boundary does not appear within a diagonal line. This is related to (1). A segment boundary does not normally appear within a diagonal line because each diagonal line often corresponds to a segment.

(3) Similar adjacent lines appear within a segment. Line-level repetitions that are adjacent, such as rhymes and refrains, tend to occur within a segment. For example, line l_7 rhymes internally with l_8 where these lines appear within a segment because $\text{sim}(l_7, l_8)$ indicates moderate similarity.

(4) A line similar to the first or last line of a song is an indicator of a segment boundary. Lines similar to the first or last line of lyrics tend to be repeated at segment boundaries. For example, in Figure 1, the first and last lines of the song, i.e., l_1 and l_{32}, are exactly the same as the first and last lines of segments ④ and ⑧. This is because the first and last lines of lyrics tend to be part of a chorus section that is often repeated throughout the lyrics.

To examine the extent to which these four patterns correlate with segment boundaries, we compute the prior and conditional probabilities of each pattern using the full lyrics database. Table 1 shows that all conditional probabilities are greater than their corresponding prior probabilities. These results suggest that the above repeated patterns reasonably capture segment boundaries, supporting the use of repeated patterns for modeling the segment structure of lyrics.

Note that pattern (1) does not hold for many cases. Figure 2 illustrates a typical negative example against pattern (1). This figure includes three diagonal lines, but the shorter lines do not agree with a segment boundary at either end. Similar cases are abundant partly because even a repetition of a single line is identified as a diagonal line in this experiment. This problem implies that a single occurrence of our local repeated pattern is not a sufficient clue for identifying a segment boundary. The conflict between patterns (1) and (2) is also shown in Figure 1, where a segment boundary implied by pattern (1) bisects a diagonal line from (l_{25}, l_9) to (l_{32}, l_{16}), which goes against pattern (2). This motivates us to build a machine learning-based model to capture combinations of multiple clues. The subsequent section describes how we represent these repeated patterns as features for predicting segment boundaries.

[2]The diagonal line of $\text{sim}(l_i, l_i)$ is ignored for analysis because it conveys no information.

4 Computational modeling of segment patterns in lyrics

To confirm the validity of our four repeated patterns for segment structures, we address the novel task of detecting segment boundaries in lyrics. Given the lyrics of a song where all segments are concatenated (no empty lines), the task is to identify the segment boundaries of the lyrics reproducing the empty lines. We formalize this task as a binary classification problem to predict the end ($y = 1$) or continuation ($y = 0$) of a segment between lines l_i and l_{i+1}. We model the conditional probability $p(y|i)$ using logistic regression with two different types of features: (1) *repeated patterns in lyrics* and (2) *textual expressions appearing at the line boundaries*.

4.1 Repeated patterns

We propose four subtypes of repeated pattern features (RPF1, RPF2, RPF3, and RPF4) corresponding to the four hypotheses presented in Section 3.2. Here, matrix M denotes the SSM of the lyrics. Each element $m_{i,j}$ represents the similarity between lines l_i and l_j, i.e., $m_{i,j} = \mathrm{sim}(l_i, l_j)$.

RPF1 The first repeated pattern (*the beginning or end point of a diagonal line in an SSM is a clue for a segment boundary*) is formalized as follows. Given two lines i and j, we expect that there exists a boundary after both of these lines if the lines are similar/dissimilar, but $i + 1$ and $j + 1$ are opposite (dissimilar/similar). For a given line i, Equation 1 enumerates a set of lines j ($1 \leq j \leq L$) where there may be boundaries after line i and every line j:

$$g_\lambda(i) = \{j \mid (m_{i,j} - \lambda)(m_{i+1,j+1} - \lambda) < 0\} \tag{1}$$

Here, λ is a threshold for detecting similarity and dissimilarity. The left side of Figure 3 illustrates four likely boundaries for line $i = 24$ with the threshold $\lambda = 0.6$: $g_{0.6}(24) = \{8, 12, 20, 28\}$.

Using the function $g_\lambda(i)$, we define feature functions $f_\lambda^{(\mathrm{RPF1\#})}(i)$ and $f_\lambda^{(\mathrm{RPF1v})}(i)$ that assess how likely it is that line i is located at the beginning or end points of diagonal lines in the SSM:

$$f_\lambda^{(\mathrm{RPF1\#})}(i) = |g_\lambda(i)| \tag{2}$$

$$f_\lambda^{(\mathrm{RPF1v})}(i) = \frac{1}{|g_\lambda(i)|} \sum_{j \in g_\lambda(i)} |m_{i,j} - m_{i+1,j+1}| \tag{3}$$

To sum up, $f_\lambda^{(\mathrm{RPF1\#})}(i)$ counts the number of likely boundaries after line i and other lines j, and $f_\lambda^{(\mathrm{RPF1v})}(i)$ computes the mean of the similarity differences at likely boundaries after line i and other lines j. We define multiple features with different threshold values λ.

RPF2 The second repeated pattern (*a segment boundary does not appear inside of a diagonal line of an SSM*) is formalized analogously to RPF1. Given two lines i and j, we expect that lines i and j are *points of continuity* if lines i and j are similar and $i + 1$ and $j + 1$ are also similar. For a given line i, Equation 4 enumerates a set of lines j ($1 \leq j \leq L$) where i and j are points of continuity:

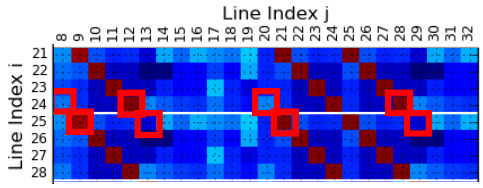
$$c_\lambda(i) = \{j \mid m_{i,j} \geq \lambda \wedge m_{i+1,j+1} \geq \lambda\} \tag{4}$$

The middle of Figure 3 shows an example of continuous points (here, $c_{0.6}(10) = \{22, 26\}$ in this example).

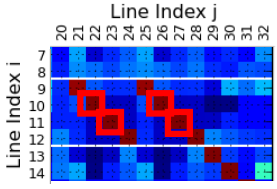
Similar to RPF1, Equations 5 and 6 count the number of continuous points and the mean of the similarity differences at continuous points, respectively.

$$f_\lambda^{(\mathrm{RPF2\#})}(i) = |c_\lambda(i)| \tag{5}$$

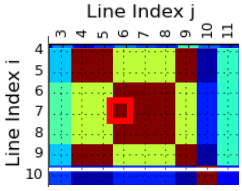
$$f_\lambda^{(\mathrm{RPF2v})}(i) = \frac{1}{|c_\lambda(i)|} \sum_{j \in c_\lambda(i)} |m_{i,j} - m_{i+1,j+1}| \tag{6}$$



Some diagonal lines end at line 24 and begin at line 25. A segment boundary is likely to appear between lines 24 and 25.



Two diagonal lines span over lines 9 to 12. Lines 9 to 12 are not likely to divide.



Lines 6 and 7 are similar. A segment boundary is unlikely to appear between lines 6 and 7.

Figure 3: Repeated pattern features: RPF1, RPF2, and RPF3

Position	-3	-2	-1	0 (Line Break)	1	2	3
Word	oh	oh	!!		I	love	you
POS tag	UH	UH	SYM		PRP	VBP	PRP

6 words between i^{th} line and $i+1^{th}$ line

TF1_Uni-gram(-3) = "oh"	TF1_Bi-gram(-2) = "oh_oh"	TF1_Tri-gram(-2) = "oh_oh_!!"
TF1_Uni-gram(-2) = "oh"	TF1_Bi-gram(-1) = "oh_!!"	TF1_Tri-gram(-1) = "oh_!!_I"
TF1_Uni-gram(-1) = "!!"	TF1_Bi-gram(0) = "!!_I"	TF1_Tri-gram(1) = "!!_I_love"
TF1_Uni-gram(1) = "I"	TF1_Bi-gram(1) = "I_love"	TF1_Tri-gram(2) = "I_love_you"
TF1_Uni-gram(2) = "love"	TF1_Bi-gram(2) = "love_you"	
TF1_Uni-gram(3) = "you"		

Textual Feature 1 (TF1): 15 word N-grams

TF2_Uni-gram(-3) = "UH"	TF2_Bi-gram(-2) = "UH_UH"	TF2_Tri-gram(-2) = "UH_UH_SYM"
TF2_Uni-gram(-2) = "UH"	TF2_Bi-gram(-1) = "UH_SYM"	TF2_Tri-gram(-1) = "UH_SYM_PRP"
TF2_Uni-gram(-1) = "SYM"	TF2_Bi-gram(0) = "SYM_PRP"	TF2_Tri-gram(1) = "SYM_PRP_VBP"
TF2_Uni-gram(1) = "PRP"	TF2_Bi-gram(1) = "PRP_VBP"	TF2_Tri-gram(2) = "PRP_VBP_PRP"
TF2_Uni-gram(2) = "VBP"	TF2_Bi-gram(2) = "VBP_PRP"	
TF2_Uni-gram(3) = "PRP"		

Textual Feature 2 (TF2): 15 Part of speech N-grams

Figure 4: Textual features: TF1 and TF2

RPF3 (similarity with a subsequent line) RPF3 encodes the third repeated pattern, i.e., similar adjacent lines belong to the same segment. For a given line index i, this is quantified by the similarity $\mathrm{sim}(l_i, l_{i+1})$:

$$f^{(\mathrm{RPF3})}(i) = m_{i,i+1} \tag{7}$$

The right of Figure 3 shows an example where RPF3 indicates a continuation between lines 6 and 7.

RPF4 (similarity with the first and last lines) The fourth repeated pattern (i.e., *a line similar to the first line of the lyrics is likely to be the first line of a segment*, and *a line similar to the last line of the lyrics is likely to be the last line of a segment*) is encoded by two feature functions $f^{(\mathrm{RPF4b})}(i)$ and $f^{(\mathrm{RPF4e})}(i)$:

$$f^{(\mathrm{RPF4b})}(i) = m_{i,1} \tag{8}$$

$$f^{(\mathrm{RPF4e})}(i) = m_{i,n} \tag{9}$$

4.2 Textual expressions

Some textual expressions appear selectively at the beginning or end of a segment. For example, the phrase "So I" often appears at the beginning of a line but rarely appears at the beginning of a segment. To exploit such indications of the beginnings/ends of lines, we propose two textual features (TF1 and TF2).

TF1 (word n-grams at a line boundary) A phrase like "oh oh !!" tends to appear at the end of a segment. In contrast, a phrase like "I'm sorry" may appear at the beginning of a segment. Previous work on sentence boundary estimation has often used n-grams to detect segment boundaries (Beeferman et al.,

Method	P_k (%)	WD (%)	Precision (%)	Recall (%)	F-measure (%)
Random	49.35	53.67	14.29	12.50	13.33
TF$*$	40.51	44.65	34.95	31.66	33.22
RPF$*$	**27.00**	**32.16**	56.05	59.42	57.68
Proposed (ALL)	27.22	32.22	56.58	60.65	**58.55**
Ablation test					
−RPF1	31.38	35.89	51.95	51.32	51.63
−RPF2	30.62	36.73	49.22	57.64	53.10
−RPF3	27.46	32.71	55.59	59.40	57.43
−RPF4	27.64	32.68	55.73	59.94	57.76
−TF1_Uni_gram	27.00	31.95	56.90	60.73	58.75
−TF1_Bi_gram	26.84	31.88	56.96	61.41	59.10
−TF1_Tri_gram	27.32	32.53	55.88	61.42	58.52
−TF2_Uni_gram	28.24	31.84	59.40	51.91	55.40
−TF2_Bi_gram	26.89	31.25	58.86	58.12	58.49
−TF2_Tri_gram	26.67	31.40	57.91	60.23	59.05
−TF1_{Uni,Bi}_gram, TF2_Tri_gram (Best Performance)	**26.58**	31.55	57.40	61.21	**59.24**

Table 2: Results of ablation tests

1999). Thus, we define word n-gram features (for $n = 1, 2, 3$) around a line boundary. More specifically, we define 15 n-gram features at different positions, listed and illustrated with an example in Figure 4.

TF2 (part of speech n-grams around a line boundary) Parts of speech (POS), such as *particles* or *determiners* do not tend to appear at the end of a sentence, and *conjunctions* do not appear at the beginning of a sentence. We exploit these tendencies by defining features for POS. Similar to TF1, we define POS n-gram features (for $n = 1, 2, 3$) around a line boundary. Specifically, we define 15 POS n-gram features at different positions, as shown in Figure 4.

5 Experiment

We sampled 105,833 English songs from the Music Lyrics Database v.1.2.7 so that each song contains at least five segments. The resulting dataset includes 2,788,079 candidate boundaries and 517,234 actual boundaries. We then split these songs into training (60%), development (20%), and test (20%) sets. For feature extraction, we used the Stanford POS Tagger (Toutanova et al., 2003). To train the segment boundary classifiers, we used the Classias implementation (Okazaki, 2009) of L2-regularized logistic regression. By employing multiple threshold values of λ from 0.1 to 0.9 with a step size of 0.1, we used them all together.

5.1 Performance evaluation metrics

We used two sets of metrics to evaluate the performance of each model for the task. One was standardly used in audio music segmentation, i.e., the precision, recall, and F-measure of identifying segment boundaries. Precision is the ratio of correctly predicted boundaries over all predicted boundaries, recall is the ratio of correctly predicted boundaries over all true boundaries, and F-measure is the harmonic mean of precision and recall. The other set was standardly used in text segmentation literature: P_k (Beeferman et al., 1999) and WindowDiff (*WD*) (Pevzner and Hearst, 2002). P_k is the probability of segmentation error that evaluates whether two lines l_i and l_j in lyrics fewer than k lines apart are incorrectly concatenated or divided by a segmentation model. P_k is considered a more suitable measure than F-measure in text segmentation because it assigns partial credit to nearly correct estimations. *WD* is a variant of P_k that resolves a problem of P_k by penalizing false positives. We set the window size k of P_k and *WD* to

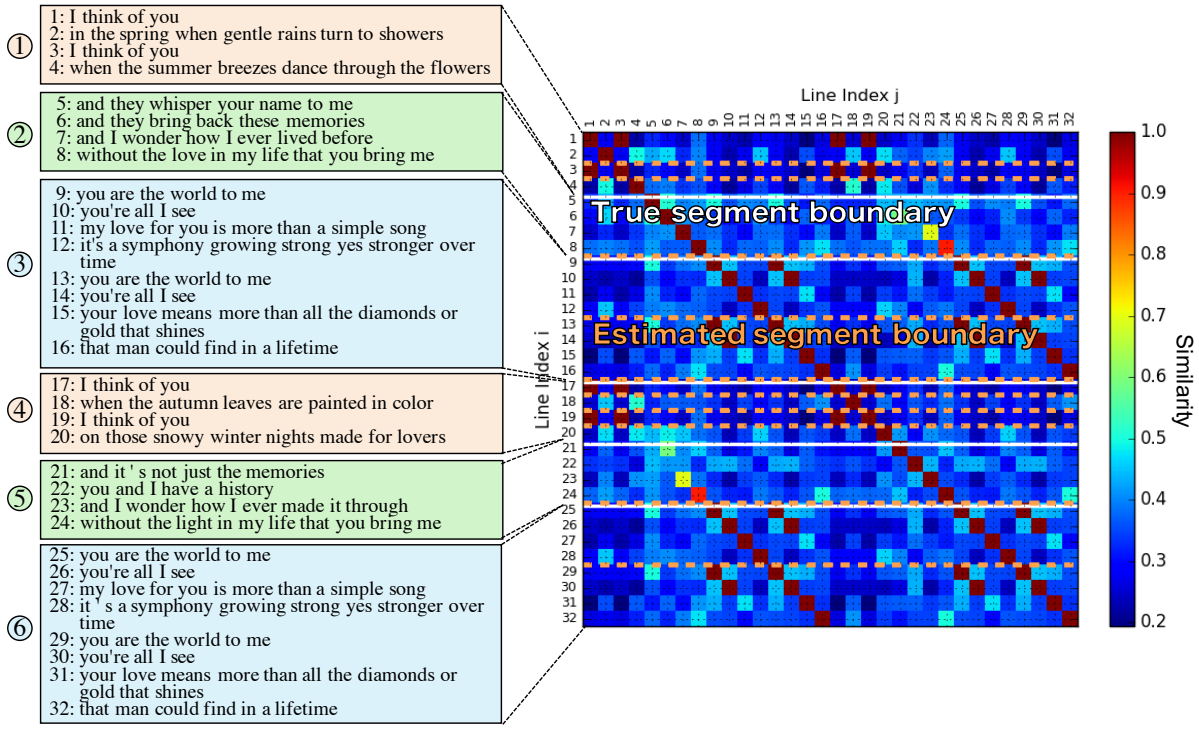


Figure 5: Examples of false positives (title of lyrics: *I think of you* (RWC-MDB-P-2001 No.87 from RWC Music Database (Goto et al., 2002))). White horizontal lines indicate true segment boundaries. Orange horizontal dashed lines indicate predicted boundaries.

one-half the average line length of the correct segments for each song in the test set.

5.2 Contributions of different features

We investigated the contribution of each feature set by conducting ablation tests over different combinations of feature sets. The results are shown in Table 2. **Random** denotes our baseline, a model selecting boundaries with uniform probability $P = 0.186$, the true frequency of boundaries ($P = 517,234/2,788,079$). RPF$*$ and TF$*$ denote the models with all repeated pattern features and all textual features, respectively. **Proposed** indicates the performance of the model with all proposed features. At the top of Table 2, the F-measure of the proposed method was 58.44, or 45 points higher than that of the random baseline.

The results of the ablation tests are shown in the bottom of Table 2. For example, "$-$RPF1" indicates that we ablated the feature RPF1 from the proposed method, which uses all of the features. Our best-performing model achieved an F-measure of 59.24 by excluding the TF1 unigram and bigram and TF2 trigram features.

The table shows that each type of our RPF features contributes to performance. Note that these four types are not redundant, and each of our hypotheses yielded positive results. Note that removing RPF1 and RPF2, which are intended to capture long-range repeated patterns, decreased the F-measure by 6.92 and 5.45 points, respectively. This result supports the hypothesis that sequences of repeated lines (diagonal lines in the SSM) are important clues for modeling lyrics segmentation.

In contrast to results reported in text segmentation literature (Beeferman et al., 1999), TF features turned out to be ineffective for lyrics segment boundary estimation, except for TF2 unigram features. One possible reason is that there is a larger variety of expressions used at the beginning or end of a segment in lyrics compared with prose texts. Still, the inclusion of some textual features did lift the performance of the RPF* model by nearly 2 points. Further investigation of TF features is left for our future work.

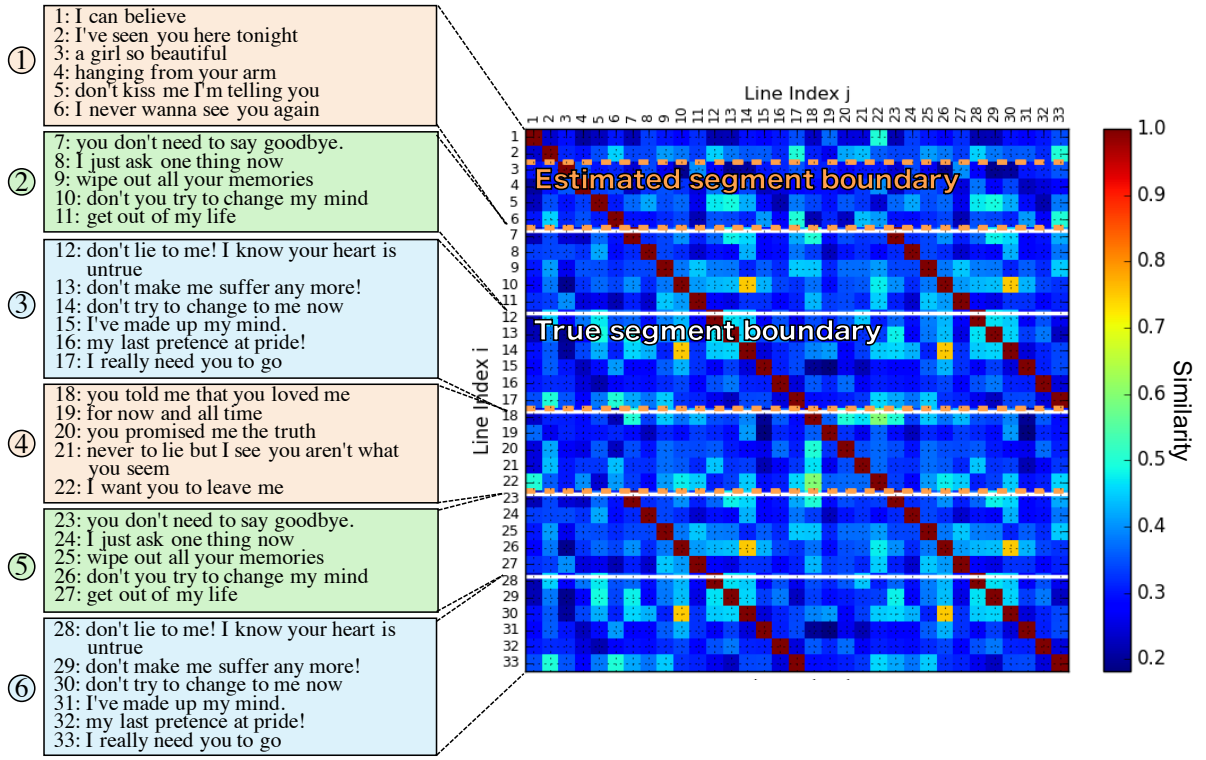


Figure 6: Examples of false negatives (title of lyrics: *Don't lie to me* (RWC-MDB-P-2001 No.97 from RWC Music Database (Goto et al., 2002))). White horizontal lines indicate true segment boundaries. Orange horizontal dashed lines indicate predicted boundaries.

5.3 Error analysis

Figures 5 and 6 give two examples of lyrics and SSMs that illustrate typical errors of our best model. Horizontal dashed lines depict predicted boundaries. As shown in Figure 5, the model sometimes overly divides a true segment into segments as small as single lines, false positives that appear to be due to occurrences of repeated single lines (here, lines 1, 3, 17 and 19). This is not a trivial problem because repetitions of single lines sometimes serve as an important clue. In fact, when restricting diagonal lines to be of the length of two or more lines, we considerably lose recall while gaining precision. More investigation is needed for further improvement.

In contrast to the case of Figure 5, Figure 6 shows a typical example of false negatives. We missed a boundary between, for example, lines 11 and 12. For this boundary, we cannot find any clear repeated pattern indicator. Such cases suggest a limitation of repeated pattern features and the need for further refinement of the model. One direction is to incorporate semantics-oriented state-of-the-art techniques for prose text segmentation such as topic tiling (Riedl and Biemann, 2012).

6 Conclusion and future work

This study has addressed the issue of modeling discourse segments in lyrics in order to understand and model the discourse-related nature of lyrics. We first conducted a large-scale corpus study into the discourse segments of lyrics, in which we examined our primary hypothesis that discourse segmentations strongly correlate with repeated patterns. To the best of our knowledge, this is the first study that takes a data-driven approach to explore the discourse structure of lyrics in relation to repeated patterns. We then proposed a task to automatically identify segment boundaries in lyrics and explored machine learning-based models for the task with repeated pattern features and textual features. The results of our empirical experiments show the importance of capturing repeated patterns in predicting the boundaries of discourse segments in lyrics. In future, we plan to refine the model further by incorporating topic/semantic

information, to extend the modeling of lyric discourse by combining it with audio musical structure, and to embed a resulting model into application systems, such as lyrics generation systems and lyrics composition support systems.

Acknowledgments

This study utilized the RWC Music Database (Popular Music). This work was partially supported by a Grant-in-Aid for JSPS Research Fellow Grant Number JP16J05945, JSPS KAKENHI Grant Numbers JP15K16045 and JP15H01702, and CREST, JST.

References

Dave Austin, Jim Peterik, and Cathy Lynn Austin. 2010. *Songwriting for Dummies*. Wileys.

Adriano Barate, Luca Andrea Ludovico, and Enrica Santucci. 2013. A semantics-driven approach to lyrics segmentation. In *Proceedings of the 8th International Workshop on Semantic and Social Media Adaptation and Personalization*, pages 73–79.

Doug Beeferman, Adam Berger, and John Lafferty. 1999. Statistical models for text segmentation. *Journal of Machine learning*, 34(1):177–210.

Freddy Y.Y. Choi. 2000. Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 26–33.

Jonathan Foote. 2000. Automatic audio segmentation using a measure of audio novelty. In *Proceedings of International Conference on Multimedia and Expo 2000*, pages 452–455.

Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. 2002. RWC music database: Popular, classical and jazz music databases. In *Proceedings of the 3rd of International Society for Music Information Retrieval*, volume 2, pages 287–288.

Masataka Goto. 2006. A chorus section detection method for musical audio signals and its application to a music listening station. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1783–1794.

Min-Yen Kan, Judith L. Klavans, and Kathleen R. McKeown. 1998. Linear segmentation and segment significance. In *Proceedings of the 6th International Workshop of Very Large Corpora (WVLC-6)*, pages 197–205.

Lie Lu, Muyuan Wang, and Hong-Jiang Zhang. 2004. Repeating pattern discovery and structure analysis from acoustic music data. In *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 275–282.

Igor Malioutov and Regina Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 25–32.

Brian McFee and Daniel PW Ellis. 2014. Learning to segment songs with ordinal linear discriminant analysis. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing 2014*, pages 5197–5201.

Naoaki Okazaki. 2009. Classias: A collection of machine-learning algorithms for classification. `http://www.chokkan.org/software/classias/`.

Jouni Paulus and Anssi Klapuri. 2006. Music structure analysis by finding repeated parts. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 59–68.

Jouni Paulus, Meinard Müller, and Anssi Klapuri. 2010. Audio-based music structure analysis. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 625–636.

Lev Pevzner and Marti A. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

Jeffrey C. Reynar. 1999. Statistical models for topic segmentation. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 357–364. Association for Computational Linguistics.

Martin Riedl and Chris Biemann. 2012. Topictiling: a text segmentation algorithm based on lda. In *Proceedings of Association for Computational Linguistics 2012 Student Research Workshop*, pages 37–42. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180.

Kento Watanabe, Yuichiroh Matsubayashi, Kentaro Inui, and Masataka Goto. 2014. Modeling structural topic transitions for automatic lyrics generation. In *Proceedings of The 28th Pacific Asia Conference on Language, Information and Computation*, pages 422–431.

Li Yujian and Liu Bo. 2007. A normalized Levenshtein distance metric. *Journal of IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1091–1095.

Multi-level Gated Recurrent Neural Network for Dialog Act Classification

Wei Li
Key Laboratory of Computational Linguistics, Peking University
Bejing, China
liweitj47@pku.edu.cn

Yunfang Wu
Key Laboratory of Computational Linguistics, Peking University
Bejing, China
wuyf@pku.edu.cn

Abstract

In this paper we focus on the problem of dialog act (DA) labelling. This problem has recently attracted a lot of attention as it is an important sub-part of an automatic dialog model, which is currently in great demand. Traditional methods tend to see this problem as a sequence labelling task and deal with it by applying classifiers with rich features. Most of the current neural network models still omit the sequential information in the conversation. Henceforth, we apply a novel multi-level gated recurrent neural network (GRNN) with non-textual information to predict the DA tag. Our model not only utilizes textual information, but also makes use of non-textual and contextual information. In comparison, our model has shown significant improvement over previous works on the Switchboard Dialog Act (SWDA) data by over 6%.

1 Introduction

Dialog act labelling is one of the ways to find the shallow discourse structures of natural language conversations. It represents the meaning or intention of each short sentence within a conversation by giving a tag to each sentence (Austin and Urmson, 1962; Searle, 1969). DA can be of help to many tasks, for example, the DA of the current sentence provides very important information for answer generation in an automatic question answering system. This converts a complex system into a classification problem, enabling many existing systems to fit in the problem.

Traditional methods apply classifiers with rich human-crafted features to tag the sentences. One can view each sentence in the dialog as a separate one and label it accordingly, such as the work of (Silva et al., 2011), but this results in the loss of sequential information in the conversation context. Stolcke et al. (2000) used a segmented version of switchboard dialog act (SWDA) (Godfrey et al., 1992) with 43 tags based on the DAMSL labelling system , and proposed to use a hidden Markov model with rich features to predict the DA of each sentence. Although their model produces relatively good results, the feature construction and tuning consume too much human effort, and also make the adaptation between tasks difficult.

Using the deep learning framework, researchers have developed various systems to deal with DA and related problems like sentiment analysis and sentence classification. One can build a simple CNN architecture like Kim (2014) to do the labelling work. However, the sentences in a conversation are highly variant in length, some of which can be as short as one to two words or may even include nothing but some telephone script symbols. For example, a lot of sentences consist of nothing but "<*laughter*>." and "*Okay*". To be specific, in the SWDA data, 3,253 sentences consist of a single word and the length of 41.4% sentences are under 5 words. Figure 1 shows the distribution of sentence lengths in detail. As is shown in the figure, most of the sentences (61%) are under 10 words, which implies that a significant portion of the overall accuracy can be attributed to short sentences.

Most of previous models tend to do poorly on these extremely short sentences because of the lack of information. To deal with short texts, one must uncover more information, such as context sentences,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1970–1979, Osaka, Japan, December 11-17 2016.

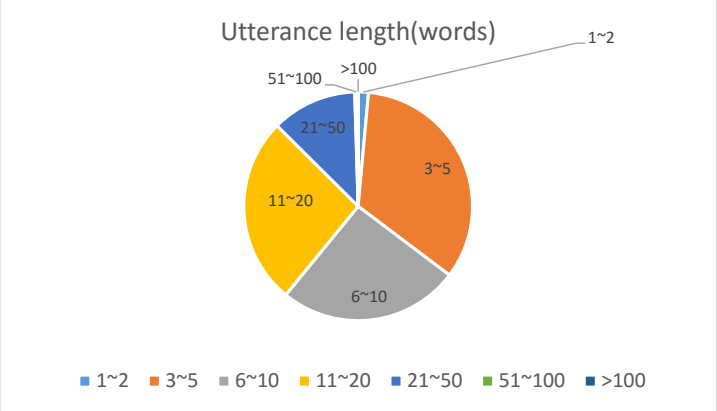


Figure 1: Sentence length distribution in the SWDA corpus

to facilitate the labelling process. In fact, the most important character of DA labelling that is different from simple sentence classification is that utterances appear sequentially in a conversation. Lee and Dernoncourt (2016) tried to make use of historical information by feeding previous sentences in a fixed window together with the current one to a feed forward neural network. This makes a good attempt in applying contextual information. However, this approach loses long distance dependency, thus giving very little improvement when compared with the CNN baseline. Zhou et al. (2015) tried to capture sequential information with the conditional random field (CRF) on the basis of a heterogeneous neural network. While their model works very well, we must also be keen to note that the RNN family models surpass CRF in sequence prediction tasks, as pointed out by Irsoy and Cardie (2014) and Yao et al. (2014).

Apart from textual and contextual information, non-textual information can also be considered. Hu et al. (2013) applied a restricted Boltzmann machine to combine textual and non-textual features in a community question answering problem. Their work makes good use of the non-textual features by combining them with textual features in an unsupervised manner.

To deal with the limitations of previous works, we propose a multi-level GRNN with non-textual features to predict the DAs. Our contributions can be highlighted in the following aspects:

- We apply a two-level GRNN to predict the DA. The low level GRNN is designed for modelling textual information of each sentence, and the top level GRNN is designed to make use of historical information in a conversation. This method produces an obvious improvement over the previous works as it automatically selects what information in the context to remember and forget.

- We use a feed forward neural network to capture the non-textual information. Then we feed the hidden layer as sentence level non-textual information to the top level GRNN.

- We conduct extensive experiments for DA labelling on the open SWDA corpus by exploiting different neural network models. With the new framework applied, our model achieves a significant improvement over previous works in SWDA task by over 6% from 73.1 to 79.37.

2 Related Work

2.1 Traditional methods on dialog act labelling

Dialog acts are to represent the intention of each sentence within a conversation. Allen and Core (1997) proposed the Dialog Act Markup in Several Layers (DAMSL) scheme to provide a top level structure for

anotating dialogs, which was applied by many dialog annotation systems (Jurafsky et al., 1997; Dhillon et al., 2004). Bunt et al. (2012) gave a detailed summary over the standard of dialog acts annotation in semantic annotation framework.

Dialog act labelling was traditionally viewed as a sequence labelling or sentence modelling problem. Most of the previous works try to predict the DA by calculating the probability of each label. Reithinger and Klesen (1997) used a language model to predict the probability of a certain DA. However, the effort to predict probability using a language model results in a severe loss of information, thereby leading to a poor result. Louwerse and Crossley (2006) introduced n-gram features to predict the DA, which is widely used in NLP tasks. This model uncovers more information from the text, but it fails to capture long-distance dependency. Surendran and Levow (2006) used SVM on individual sentences then viterbi decoding to make use of contextual information in a HMM style. This model builds a rather good framework for sequential labelling, as it not only feeds each sentence to a strong classifier SVM, but also makes use of context information in a probability graph. (Kim et al., 2010) further proposed to use CRF to deal with the problem, using both traditional bag of words features and new features such as dialog structures and dependencies between utterances. The common weakness of these methods is that they depend heavily on the features selected, and the feature construction process consumes much human effort.

2.2 Deep Learning models

As deep learning becomes increasingly popular, researchers have been trying to apply deep learning frameworks to deal with natural language processing and understanding tasks, including sentence modelling, DA labelling and many other tasks. Collobert and Weston (2007), Collobert and Weston (2008) and Collobert et al. (2011) constructed deep neural network structures for natural language processing tasks, which project one-hot word representations into distributed representations with a look-up table (or a projection layer) and build either feed forward or convolutional neural network upon them. This type of models seek to free researchers from laborious feature engineering, and allow the systems to easily adapt to different tasks.

Kalchbrenner et al. (2014) proposed a dynamic convolution neural network with multiple layers of convolution and k-max pooling to model a sentence. As imagined, this model is computationally expensive due to the many layers. Conversely, the CNN model proposed by Kim (2014) takes just one convolution and pooling layer with multi-channel word embeddings, followed by a softmax classifier. This model succeeded in many NLP tasks, such as sentence classification, sentiment analysis and so on.

Apart from CNN like architectures, researchers also applied recurrent neural network (RNN) and its variants to model sentences. Originally proposed by Elman (1990), RNN is expected to propagate information through time, which means one can make use of past information as latent variables. Mikolov et al. (2010) applied RNN to language modelling and got some very interesting results for word embedding. However, this vanila RNN suffers from the same problem as other deep neural networks, the problem of vanishing gradient. More specifically, gradients can either explode or vanish through time (Bengio et al., 1994). To tackle this problem, Hochreiter and Schmidhuber (1997) proposed long short term memory (LSTM), which uses a cell with input, forget and output gates to prevent the vanishing gradient problem. This makes RNN family networks much more powerful by memorizing information from long distance.

Recently, inspired by the gating idea, Cho et al. (2014) proposed another variant of RNN named gated recurrent neural network, which only uses a reset gate and a update gate to encode and decode sentences in a translation system. As reported in Chung et al. (2014), GRNN can achieve better results than LSTM in most tasks.

Palangi et al. (2015) proposed to sequentially take each word in a sentence, extract its information, and embed it into a semantic vector. This way, one can access the sentence level vector and use it to deal with other tasks such as information retrieval. Shen and Lee (2016) introduced one type of attention mechanism to sentence modelling based on LSTM, they also tested their model on SWDA task, which we will reference as a baseline. Their model performed better on longer sentences by highlighting the important parts of the sentence. But, as aforementioned, the most important part of this problem is

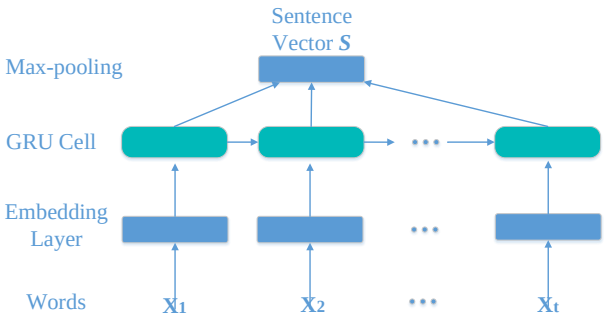


Figure 2: Gated recurrent neural network for sentence representation based on textual information

not about long sentences, but the short ones, which take the majority share of the corpus. Lee and Dernoncourt (2016) regarded this problem as a sequential short text classification problem, which is a good direction. However, although they tried to capture the historical information, they failed to seize long distant information in a conversation, because they only feed a fixed window to the neural network and the capability of the feed forward neural network is very limited.

3 Our Approach

In this paper, we propose to utilize a multi-level GRNN architecture to mine the information from both within the sentence and between the sentences. Gated recurrent neural network is a variant of the recurrent neural network. The GRNN allows information to flow over time without the problem of vanishing gradient, and is expected to memorize long distance dependency.

Equations 1 to 2 show the method to calculate the output h_t at time stamp t, with the input x_t and history information h_{t-1}, which is the output at time stamp $t-1$. In each gated recurrent unit, the *reset gate* (Equation 1) and the *update gate* (Equation 2) are designed to decide which latent information is to be discarded and which is to be held. Equation 3 calculates the candidate unit similar to vanilla RNN unit, except that it uses a *reset gate* to filter history information, and Equation 4 uses the *update gate* and the candidate unit to get the final output unit.

In our model, we first use the low level GRNN on the scale of words to learn sentence level vector, then we use GRNN to propagate the information between sentences over time within the same conversation. To discover more information on the sentence level, we also apply a feed forward neural network to capture the non-textual information such as the length of the sentence, the index of the utterance and so on.

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \quad (1)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \quad (2)$$

$$\tilde{h}_t = tanh(W x_t + U(r_t \odot h_{t-1})) \quad (3)$$

$$h_t = (1 - z_t) h_{t-1} + z_t \tilde{h}_t \quad (4)$$

3.1 Textual information

Textual information is the basis of our end-to-end labelling system. We use a GRNN with max-pooling to encode the sentence into a vector.

As is shown in Figure 2, we treat each word as a separate unit. We first look up the corresponding embedding in a lookup table, which gives a matrix of $D * L$, D is the dimension of word embedding and L is the sentence length. Then we feed each word in the sentence into the low level GRNN, one word per time step, and then perform max pooling on the output of the GRU cells over the whole sentence.

caller	utterance index	sub-utterance index	act tag	text
A	5	2	qy	{F Um, } {F uh, } do you live right in the city itself? /
B	6	1	nn	No, /
B	6	2	sd	I'm more out in the suburbs, /
B	6	3	sd	{C but } I certainly work near a city. /
A	7	1	bk	Okay, /
A	7	2	qy	{C so } [ca-, +

Table 1: Utterance examples in SWDA corpus

3.2 Non-textual information

Although the aforementioned low level GRNN can capture the textual information within the sentence itself, it fails to make use of information from a higher level. For instance, in our DA labelling , the length of sentence plays an important role in identifying the tag of sentence, because the distribution of sentence length varies between different DAs. For sentences under the label of *acknowledge*, most sentences are below 10 words; whereas for sentences under the label of *statement non-opinion*, the sentences have more varied length distribution. As a matter of fact, it is shown in our experiment that this sentence length feature alone gives a much better prediction than random guesses.

Feed forward neural network (FFNN) is one of the simplest form of deep neural networks, and does a good job in many tasks. In this part of the neural network, we feed four shallow non-textual features to a FFNN. We use the hidden layer as the vector representing the non-textual information of the sentence. The four features we used are listed below. To better understand the features, Table 1 shows some examples from the original scripts.

- **Utterance index**: A conversation consists of multiple natural utterances, which are further split into lines of sentences for the convenience of tagging. Utterance index is the index of utterances, which can span multiple sentences. For example in Table 1, caller B says three sentences, and these three sentences share the same utterance index (6), but have different sub-utterance index. This feature may help when different acts take place in different parts of the conversation, for instance, conversations tend to begin with greetings.

- **Sub-utterance index**: Utterances can be broken across lines, sub-utterance index gives the internal position of the current sentence in the utterance. For example, in Table 1, the 6th utterance has three sentences or sub-utterances indexing from 1 to 3. This feature helps when different acts appear in different parts of an utterance. For example, questions tend to appear at the end of each utterance.

- **Same speaker**: This feature is a boolean feature of 0 or 1, indicating whether the identity of the speaker changes. Unlike the features above, this feature is deduced from the sub-utterance index. If the sub-utterance index is 1, then this feature is set to 1, otherwise 0.

- **Sentence length**: As explained earlier, the length of sentence plays an important role in predicting the label. As sentence lengths vary a lot, we normalize the lengths using Equation 5, where l is the word-wise sentence length.

$$l_{norm} = \frac{l - range(l)/2}{std(l)} \tag{5}$$

After we have the vector for textual and non-textual information aforementioned, we concatenate them together to get a combined vector for the sentence, as shown in the lower part of Figure 3.

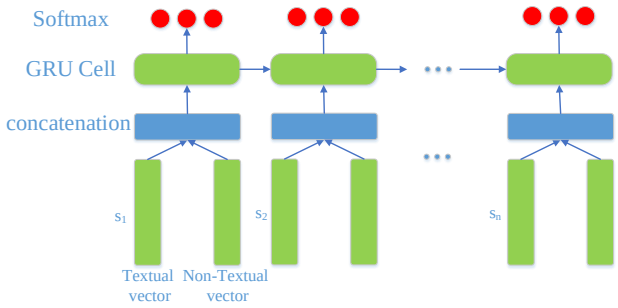


Figure 3: Gated recurrent neural network on sentence feature

3.3 Context information

GRNN is designed to remember valuable information while discarding useless information. In the DA labelling problem, the segmentation of sentences is not very strict. Many sentences are very short, which makes it very difficult to classify a sentence based on only little textual information and sentence level non-textual information. Therefore, GRNN can fit this problem very well.

In our model, we try to use GRNN to capture the structure between sentences, as shown in Figure 3. This enables our model to utilize information from longer distances, unlike the structure proposed by (Lee and Dernoncourt, 2016), which uses a fixed window to capture history information. Learning distant information is crucial for the fact that the dialog turn changes with no pattern, whereas some utterances consist of a single sentence while others consist of multiple sentences, which makes it impossible to learn the words from both speakers within a fixed window, as words of one speaker in the current sentence can be distant from the last words from the other speaker.

4 Experiment

4.1 Settings

We conducted experiments on the switchboard dialog act corpus, which extends the Switchboard-1 Telephone Speech Corpus with turn/utterance-level dialog-act tags. The tags summarize syntactic, semantic, and pragmatic information about the associated turn. There are over 200 tags in the corpus. Jurafsky et al. (1997) defines a system for collapsing them down to 44 tags.

In our experiments, we use the same data version as Stolcke et al. (2000), where there are 1,115 conversations (1.4M words, 198K utterances) in the training set, and 19 conversations (29K words, 4K utterances) in the test set. We use the same valid set as Lee and Dernoncourt (2016), which consists of 19 randomly chosen conversations. [1]

In our experiment, we build our model upon tensorflow by Abadi et al. (2015) [2], which is a popular package developed by Google for deep learning.

We use all the tokens of the utterances including texts and other telephone related symbols to train word embeddings with word2vec [3] (Mikolov et al., 2013), and set the dimension of word embeddings to 300. We use the Adam stohastic optimization method (Kingma and Ba, 2014) to minimize the negative log-likelihood cost with fine-tuning on the word embeddings. To try to avoid the over-fitting problem, we run each experiment for 10 epochs, and use the hyper-parameters from the epoch with the highest validation accuracy. We use rectified linear unit (relu) as the activation function.

4.2 Baselines

We conduct extensive experiments on the SWDA corpus by utilizing various neural network models.

[1] The train/validation/test splits were found at https://github.com/Franck-Dernoncourt/naacl2016
[2] available in https://www.tensorflow.org
[3] available in https://code.google.com/archive/p/word2vec/

Method	Accuracy
Sequential short-text classification (Lee and Dernoncourt, 2016)	73.1
Neural attention (Shen and Lee, 2016)	72.6
Our model	79.37

Table 2: Experimental results compared with previous state-of-the-art methods

- **CNN**: We implemented a convolutional neural network following the framework of (Kim, 2014). We use filters of length 2,3 and 4, and for each window length there are 100 feature maps. So each sentence has a vector of 300 real numbers. After the convolution and max-pooling layer, there is a softmax layer to predict the DA of each sentence.

- **non-textual**: We feed the four non-textual features to a typical three-layer feed forward neural network as described in Section 3.2. We set the unit number of the hidden layer to 300 and use the output of the softmax layer to predict the label.

- **CNN+non-textual**: This model is a combination of CNN and non-textual. We concatenate the pooled feature maps of CNN and the hidden layer of non-textual FFNN, and feed this new combined vector to a softmax layer to predict the label.

- **single-level GRNN**: This model follows the description in section 3.1. We feed the word embedding to the GRU cells, each word per cell. After we get the output of the GRU cells from each time step, we perform a max-pooling over them and get the sentence vector. Then we feed the sentence vector to a softmax layer to predict the tag.

- **single-level GRNN + non-textual**: This model combines the max-pooled sentence vector from single-level GRNN and the hidden layer of non-textual FFNN in the same way as CNN+non-textual. Then the concatenated vector is fed to a softmax layer to predict the tag.

- **non-textual+GRNN**: We feed the hidden layer of the non-textual FFNN to a GRNN. Then we feed the output of each GRU cell to the softmax layer to predict the labels.

- **CNN+GRNN**: We feed the sentence vector from CNN to GRNN. Then we feed the output of each GRU cell to the softmax layer to predict the labels.

- **multi-level GRNN**: We feed the sentence vector from lower level GRNN to the upper level GRNN. Then we feed the output of each GRU cell to the softmax layer to predict the labels.

- **CNN+non-textual+GRNN** We feed the combination of sentence vector from CNN and hidden layer from non-textual FFNN to a GRNN. Then we feed the output of each GRU cell to the softmax layer to predict the labels.

- **multi-level GRNN+non-textual**: This is our model in this paper. In this model, we feed the combination of sentence vector from lower level GRNN and hidden layer from non-textual FFNN to the upper level GRNN. Then we feed the output of each GRU cell to the softmax layer to predict the labels.

4.3 Comparison with previous models

Table 2 shows our result compared with other state-of-the-art results. By utilizing information from previous time stamp with GRNN, we achieve significant improvement over the previous works. As seen in Table 2, we improve the performance by 6.27% over Lee and Dernoncourt (2016) and 6.77% over Shen and Lee (2016), as we better capture both the sentence level knowledge and contextual information in a conversation.

Method	Accuracy
CNN	68.25
single-level GRNN	69.75
non-textual	43.60
CNN+non-textual	70.86
single-level GRNN + non-textual	71.90
non-textual+GRNN	48.09
CNN+GRNN	77.14
multi-level GRNN	77.65
CNN+non-textual+GRNN	78.40
multi-level GRNN+non-textual	79.37

Table 3: Results of different neural networks in our experiment

Text	standard	single-level GRNN	final model
{F Um, } {F uh, } do you live right in the city itself? /	qy	qy	qy
No, /	nn	nn	nn
I'm more out in the suburbs, /	sd	sd	sd
{C but } I certainly work near a city. /	sd	sd	sd
Okay, /	bk	**fo_o_fw_by_bc**	bk
{C so } [ca-, +	qy	**sd**	qy

Table 4: The tagging DA results using two different neural network models, where "standard" means the golden standard tag in the data.

4.4 Comparison with baseline models

The experimental results in Table 3 show that both CNN and single-level GRNN with textual information can give relatively good results (68.25 & 69.75) for the DA labelling task.

Non-textual information can further improve the accuracy as they provide information about the whole sentence, instead of just individual words. This is verified by the fact that CNN+non-textual improves 2.61% over CNN and single-level GRNN+non-textual improves 2.15% over single-level GRNN. In fact, non-textual itself gives a surprisingly good result compared with random guess.

It is the GRNN which captures long distance dependency from context that produces the most significant improvement to the problem. As a matter of fact, the role of GRNN is so important that GRNN based on the weak classifier non-textual FFNN improves the result by almost 5% over the non-textual FFNN alone, and GRNN on the basis of CNN improves the result by almost 10% over the raw CNN. Altogether, our model of "multi-level GRNN+non-textual" surpasses the CNN baseline significantly by over 11%.

4.5 Analysis

In Table 4 we show the tagging results of the examples listed in Table 1. These results are from the single-level GRNN (one of our baselines) and our final model, respectively. The sentences are selected from the first conversation in the test set.

From the examples, we can observe that sentences with obvious features can be easily recognized by both models, for instance the first sentence "*do you*" is correctly tagged as "*qy*" (*Yes-No-Question*). However, when the sentence is short and ambiguous or can appear in multiple circumstances, such as "*Okay*", the simpler model mistakes the "*bk*" (*Response Acknowledgement*) for "*fo_o_fw_by_bc*" (*other*), while our final model which utilizes contextual information succeeds in predicting the right tag.

5 Conclusion

In this paper, we propose a multi-level GRNN model combined with non-textual features to deal with the dialog act labelling problem. We manage to mine multi-level information out of the conversation. Our model does a very good job on predicting short sentences in the SWDA corpus. Our results surpass previous state-of-the-art results significantly without much feature engineering, which makes our model easier to adapt to similar tasks. In the future, we hope to introduce the attention mechanism into our model and make better use of contextual information.

Acknowledgement. This work is supported by National Natural Science Foundation of China (61371129), National High Technology Research and Development Program of China (2015AA015403) and Key Program of Social Science foundation of China (12&ZD227). The corresponding author of this paper is Yunfang Wu.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

James Allen and Mark Core. 1997. Draft of damsl: Dialog act markup in several layers. *Unpublished manuscript*, 2.

John Langshaw Austin and JO Urmson. 1962. *How to Do Things with Words. The William James Lectures Delivered at Harvard University in 1955.[Edited by James O. Urmson.].* Clarendon Press.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Harry Bunt, Jan Alexandersson, Jae-Woong Choe, Alex Chengyu Fang, Koiti Hasida, Volha Petukhova, Andrei Popescu-Belis, and David R Traum. 2012. Iso 24617-2: A semantically-based standard for dialogue annotation. In *LREC*, pages 430–437. Citeseer.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Ronan Collobert and Jason Weston. 2007. Fast semantic extraction using a novel neural network architecture. In *Annual meeting-association for computational linguistics*, volume 45, page 560.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Rajdip Dhillon, Sonali Bhagat, Hannah Carvey, and Elizabeth Shriberg. 2004. Meeting recorder project: Dialog act labeling guide. Technical report, DTIC Document.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 517–520. IEEE.

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

Haifeng Hu, Bingquan Liu, Baoxun Wang, Ming Liu, and Xiaolong Wang. 2013. Multimodal dbn for predicting high-quality answers in cqa portals.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *EMNLP*, pages 720–728.

Dan Jurafsky, Elizabeth Shriberg, and Debra Biasca. 1997. Switchboard swbd-damsl shallow-discourse-function annotation coders manual. *Institute of Cognitive Science Technical Report*, pages 97–102.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Su Nam Kim, Lawrence Cavedon, and Timothy Baldwin. 2010. Classifying dialogue acts in one-on-one live chats. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 862–871. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. *arXiv preprint arXiv:1603.03827*.

Max M Louwerse and Scott A Crossley. 2006. Dialog act classification using n-gram algorithms. In *FLAIRS Conference*, pages 758–763.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, volume 2, page 3.

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

Hamid Palangi, Li Deng, Yelong Shen, and Jianfeng Gao. 2015. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio Speech & Language Processing*, 24(4):694–707.

Norbert Reithinger and Martin Klesen. 1997. Dialogue act classification using language models. In *EuroSpeech*. Citeseer.

John R Searle. 1969. *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press.

Sheng-syun Shen and Hung-yi Lee. 2016. Neural attention models for sequence classification: Analysis and application to key term extraction and dialogue act detection. *arXiv preprint arXiv:1604.00077*.

Joao Silva, Luísa Coheur, Ana Cristina Mendes, and Andreas Wichert. 2011. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154.

Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373.

Dinoj Surendran and Gina-Anne Levow. 2006. Dialog act tagging with support vector machines and hidden markov models. In *INTERSPEECH*.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 189–194. IEEE.

Yucan Zhou, Qinghua Hu, Jie Liu, and Yuan Jia. 2015. Combining heterogeneous deep neural networks with conditional random fields for chinese dialogue act recognition. *Neurocomputing*, 168:408–417.

Multimodal Mood Classification - A Case Study of Differences in Hindi and Western Songs

Braja Gopal Patra, Dipankar Das, and Sivaji Bandyopadhyay
Department of Computer Science and Engineering,
Jadavpur University, Kolkata, India
{brajagopal.cse,dipankar.dipnil2005,sivaji.cse.ju}@gmail.com

Abstract

Music information retrieval has emerged as a mainstream research area in the past two decades. Experiments on music mood classification have been performed mainly on Western music based on audio, lyrics and a combination of both. Unfortunately, due to the scarcity of digitalized resources, Indian music fares poorly in music mood retrieval research. In this paper, we identified the mood taxonomy and prepared multimodal mood annotated datasets for Hindi and Western songs. We identified important audio and lyric features using correlation based feature selection technique. Finally, we developed mood classification systems using Support Vector Machines and Feed Forward Neural Networks based on the features collected from audio, lyrics, and a combination of both. The best performing multimodal systems achieved F-measures of 75.1 and 83.5 for classifying the moods of the Hindi and Western songs respectively using Feed Forward Neural Networks. A comparative analysis indicates that the selected features work well for mood classification of the Western songs and produces better results as compared to the mood classification systems for Hindi songs.

1 Introduction

Global digitization has led to music being available in the form of CDs, DVDs or other portable formats. With the rapid growth in Internet connectivity over the last decade, the audio or video music files are easily available and accessible over the World Wide Web. The number of music compositions created worldwide already exceeds a few millions and continues to grow. Similarly, the popularity of downloading and purchasing of music from online music shops has also been increased at the same pace. Thus, the organization and management of the music files are the important issues to be tackled carefully. Recently, studies on music information retrieval (MIR) have shown that moods are desirable access keys to music repositories and collections (Hu and Downie, 2010a).

In order to find out such access keys, most of the experiments on music mood classification of Western music have been performed based on the audio (Lu et al., 2006; Hu et al., 2008), lyrics (Zaanen and Kanters, 2010) and combination of both (Laurier et al., 2008; Hu and Downie, 2010b; Hu and Downie, 2010a). In case of the Indian music, few tasks have been performed on the Hindi music mood classification based on the audio (Ujlambkar and Attar, 2012; Patra et al., 2013a; Patra et al., 2013b; Patra et al., 2016b), lyrics (Patra et al., 2015c) and combination of both (Patra et al., 2016a). The maximum F-measure achieved for the multimodal system for Hindi songs was 68.6% in Patra et al. (2016a).

Indian music can be divided into two broad categories namely, "classical" and "popular" (Ujlambkar and Attar, 2012). Further, classical music tradition of India has two main variants; namely Hindustani and Carnatic. The prevalence of Hindustani classical music is found largely in north and central parts of India whereas Carnatic classical music dominates largely in the southern parts of India. Hindi or Bollywood music, also known as popular music, is mostly present in Hindi cinemas or Bollywood movies. Hindi is one of the official languages of India and is fourth most widely spoken language in the World[1]. Hindi songs make approximately 72% of the total music sales in India (Ujlambkar and Attar, 2012).

[1] https://www.cia.gov/library/publications/the-world-factbook/fields/2098.html

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1980–1989, Osaka, Japan, December 11-17 2016.

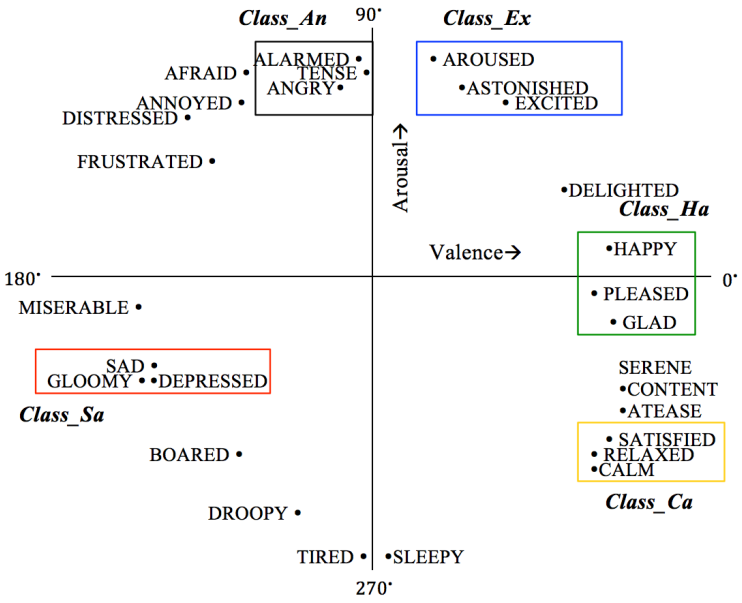


Figure 1: Russell's circumplex model of 28 affect words (Russell, 1980)

In order to deal with the above mentioned issues, the contributions of the authors are given below.

1. We employed our earlier proposed mood taxonomy for music mood classification in Hindi and Western songs (Patra et al., 2015c; Patra et al., 2016a; Patra et al., 2016b).

2. We annotated the audio and lyrics of the Hindi and Western songs using the above mood taxonomy.

3. We observed difference in mood while annotating the mood at the time of listening to the music and reading its corresponding lyric in case of the Hindi songs.

4. We identified important features using correlation based feature selection technique.

5. The Feed Forward Neural Networks (FFNNs) is implemented for mood classification purpose.

6. We have developed a multimodal system based on the audio and lyrics of the songs.

This paper is organized as follows: Section 2 introduces the mood taxonomy and describes the process of dataset preparation. Section 3 describes the audio and lyrics features. The FFNNs and the developed systems with comparison are described in the section 4. Finally, the conclusion is drawn in Section 5.

2 Mood Taxonomy and Dataset

2.1 Mood Taxonomy

We chose the *Russell's circumplex model* (Russell, 1980) to build our own mood taxonomy. The *circumplex model* and a subset of this dimensional model have been used earlier for several mood classification studies (Ujlambkar and Attar, 2012; Patra et al., 2013a; Patra et al., 2013b; Patra et al., 2015c; Patra et al., 2016a; Patra et al., 2016b). The *circumplex model* is based on *valence* and *arousal*, which is widely accepted by the research community. *Valence* indicates the positivity and negativity of emotions whereas *arousal* indicates the emotional intensity. The mood taxonomy is prepared by clustering the similar affect words of the *circumplex model* into a single class and each class contains three affect words of the *circumplex model* as shown in Figure 1. Each of our mood classes has distinct positions in terms of *arousal* and *valence*. We considered the five coarse mood classes, namely "*Class_An*", "*Class_Ca*", "*Class_Ex*", "*Class_Ha*", and "*Class_Sa*" for our experiments.

2.2 Dataset

All audio files of Hindi and Western song were collected from CDs bought from registered stores. The lyrics of the corresponding songs were collected from the web. The lyrics of the Hindi songs were written in *Romanized English* characters while essential resources like Hindi sentiment lexicons and list of stop words are available in *utf-8* character encoding. Thus, we transliterated the *Romanized* Hindi lyrics to *utf-8* characters using the transliteration tool available in the English to Indian Language Machine Translation (EILMT) project[2]. We observed several errors in the transliteration process. For example, words like 'oooohhhhooo' 'aaahhaa' were not transliterated due to the presence of repeated characters. Again, the words like 'par' and 'paar', 'jan' and 'jaan' were transliterated into different words 'पर' and 'पार', 'जन' and 'जान', but, the above pairs are the same words 'पर' and 'जान'. Hence, these mistakes were corrected manually.

Related research suggests that the state-of-the-art experiments on music mood classification have been performed on audio clips of 30 seconds (Hu et al., 2008; Ujlambkar and Attar, 2012; Patra et al., 2013b). In case of the Hindi songs, it is difficult to annotate mood of 30 second song clips since the annotators get confused in between adjacent mood classes while annotating short duration audio files. Thus, we sliced each of the audio files into 60 second clips. Each of the audio clips and lyric files of the Hindi and Western songs were annotated by three different annotators. The undergraduate students and research scholars belonging to the age group of 18-35 served as annotators.

From the annotation, we observed that different mood classes were chosen by the annotators during listening to the audio and reading the corresponding lyrics. The difference between listener's and reader's perspectives for the same song motivated us to investigate the root cause of such discrepancy. The authors believe that the subjective influence of music modulates the perception of lyric of a song in the listeners. For example, a song "Bhaag D.K.Bose Aandhi Aayi"[3] has mostly sad words like "dekha to katora jaka to kuaa (*the problem was much bigger than it seemed at first*)" in the lyric. This song was annotated as "*Class_Sa*" while reading the lyric, whereas it was annotated as "*Class_An*" while listening to the corresponding audio as it contains mostly rock music and arousal is also high. Similarly a song "Dil Duba"[4] was annotated as "*Class_Sa*" and "*Class_Ha*" while reading the lyric and listening to the corresponding audio, respectively. This song portrays negative emotions by using sad or negative words like "tere liye hi mar jaunga (*I would die for you*)", however, this song contains high valence. The above observations emphasize that the combined effect of lyric and audio plays a pivotal role in indicating the final mood inducing characteristics of a music piece. Moreover, the intensity of the emotion felt during listening to a song is much more than the intensity of the emotion while reading a lyric. The main reason behind this may be that the music with the voice induces the emotion.

We did not notice such differences in mood for the Western music. However, the intensity of the emotion was less while reading a lyric as compared to listening to the corresponding music in case of both Hindi and Western music. The confusion matrix of the Hindi song mood annotation is shown in Table 1. Detailed statistics of the annotated Hindi and Western songs are given in Table 2. We considered only those Hindi songs for our experiments, which were annotated with the same class both after listening to it and reading the corresponding lyric.

We calculated pairwise inter-annotator agreements on the dataset by computing Cohen's κ coefficient (Cohen, 1960). The inter-annotator agreements were calculated separately for audio clip annotation and lyric annotation. The overall inter-annotator agreement scores with five mood classes were found to be 0.94 and 0.84 for Hindi audio and lyrics, respectively. In case of the Western songs, the inter-annotator agreement scores with five mood classes were 0.91 and 0.87 for audio and lyrics, respectively. These correlation coefficients can be interpreted as almost perfect agreements.

[2]http://tdil-dc.in/index.php?option=com vexrtical&parentid=72

[3]http://www.lyricsmint.com/2011/05/bhaag-dk-bose-aandhi-aayi-delhi-belly.html

[4]http://www.hindilyrics.net/lyrics/of-Dil%20Duba%20Dil%20Duba.html

Table 1: Confusion matrix of the annotated songs with respect to the five mood classes [after listening to the audio (L_{Audio}) and reading the lyrics (R_{Lyrics})].

		R_{Lyrics}					
		a	**b**	**c**	**d**	**e**	**Total**
L_{Audio}	**Class_An = a**	**48**	0	15	2	10	75
	Class_Ca = b	2	**65**	3	17	13	100
	Class_Ex = c	13	3	**62**	16	6	100
	Class_Ha = d	4	11	15	**66**	4	100
	Class_Sa = e	8	17	7	9	**78**	125

Table 2: Statistics of the Hindi and Western songs

	Hindi Songs		**Western Songs**	
	Clips	**Total Songs**	**Clips**	**Total Songs**
Class_An	203	48	230	60
Class_Ca	252	65	247	72
Class_Ex	258	62	236	63
Class_Ha	232	66	218	58
Class_Sa	285	78	180	45
Total	1230	319	1111	298

3 Feature Extraction

This section describes the process of extracting features from both audio and lyrics. Feature extraction and selection play an important role in machine-learning frameworks. The important features from audio and lyrics were identified using correlation based feature selection technique. We considered different audio and textual features for mood classification in Hindi and Western songs.

3.1 Audio Features

We considered the key audio features like *intensity*, *rhythm* and *timbre* for the mood classification task. These features had been used by researchers for music mood classification in Indian languages (Ujlambkar and Attar, 2012; Patra et al., 2015b). These features were extracted using the jAudio (McKay et al., 2005) toolkit. In addition to these features, *chroma* and *harmonics* features were extracted from the audio files using the openSMILE (Eyben et al., 2010) toolkit.

3.2 Lyric Features

We adopted a wide range of textual features such as sentiment words, stylistic features and N-gram based features which are discussed in the following subsections.

Preprocessing: First we cleaned the lyrics dataset by removing the junk characters and HTML tags. Subsequently we removed the duplicate lines as it was observed that the starting stanza is usually repeated in the song at least a few times. Therefore, we removed these duplicate sentences to remove the biasness in the lyric.

3.2.1 Sentiment Lexicons (SL)

The emotion or sentiment words are one of the most important features for mood classification from lyrics. These words in the Hindi lyrics were identified using three lexicons - Hindi Subjective Lexicon (HSL) (Bakliwal et al., 2012), Hindi SentiWordnet (HSW) (Joshi et al., 2010) and Hindi Wordnet Affect (HWA) (Das et al., 2012). Similarly, we used two lexicons - SentiWordNet (Baccianellaet al., 2010) (SWN) and WordNetAffect (Strapparava and Valitutti, 2004) (WA) for identifying the sentiment words from the lyrics of the Western songs. It was observed that the number of sentiment words found in the lyrics of Hindi songs was less than the number of sentiment words found in the lyrics of Western

songs. The main reason was that the the performances of the POS tagger and stemmer/lemmatizer for Hindi language were not up to the mark. The CRF based *Shallow Parser*[5] is available for POS tagging and lemmatization, but it also did not perform well on the lyrics data because of the free word order nature of Hindi language. Most of the inflected sentiment words in Hindi lyrics were not matched with the sentiment words available in the Hindi sentiment or emotion lexicons. Thus, the number of words matched with sentiment or emotion lexicons are considerably less. In case of the Western songs, we used the RitaWordNet [6] to get the stemmed words and the parts-of-speech (POS) tags for the lyric words. The statistics of the sentiment or emotion words identified by these lexicons are given in Table 3.

Table 3: Statistics of unique sentiment and emotion words present in the lyrics of Hindi and Western songs

Classes	HWA	WA	Classes	HSL	HSW	SWN
Angry	248	312	**Positive**	1185	872	7853
Disgust	17	32				
Fear	20	52				
Happy	352	412	**Negative**	963	735	5271
Sad	110	231				
Surprise	39	81				

3.2.2 Text Stylistic Features (TSF)

Text stylistic features are widely used in text stylometric analysis such as authorship identification, author identification, etc. These features were also been used for mood classification from Hindi lyrics (Patra et al., 2015c) and Western music lyrics (Hu and Downie, 2010b). It was observed that these features reduce the performance of the system. The TSF such as the number of unique words, number of repeated words, number of lines, etc. were considered for our experiments.

3.2.3 N-Grams (NG)

It was noted by researchers that N-gram based features work well for mood classification using lyrics as compared to the stylistic or sentiment features (Zaanen and Kanters, 2010; Hu and Downie, 2010b; Patra et al., 2015c). The term frequency and document frequency (TF-IDF) scores of unigram, bigram and trigram were considered for the present study. The higher order N-grams tend to reduce the performance of the system. The N-grams having document frequencies more than one were considered to reduce the sparsity of the document vectors. We also removed the stopwords while considering the N-grams as it was observed that the stopwords do not contain any information related to the classification.

3.3 Feature Selection

Feature level correlation (Hall, 1999) was used to identify the most important features as well as to reduce the feature dimension in (Patra et al., 2015a). Thus, we used the correlation based supervised feature selection technique implemented in Weka toolkit [7] to find out the important contributory feature set for audio and lyrics.

A total of 445 audio features were extracted from Hindi and Western music audio files using jAudio and openSMILE. We also collected 12 sentiment features, 12 textual stylistic features and 6832 N-gram features from Hindi lyrics, whereas 8 sentiment features, 12 textual features and 8461 N-gram features were collected from the Western music lyrics. The feature selection technique implemented using Weka yields 154 important audio features for both Hindi and Western songs. 12 sentiment, 8 stylistic, and 1601 N-gram features from Hindi lyrics and 8 sentiment, 8 stylistic and 3175 N-gram features from Western music lyrics were extracted using feature selection technique. We subsequently used these features for the classification purpose.

[5] http://ltrc.iiit.ac.in/analyzer/hindi/

[6] http://www.rednoise.org/rita/

[7] http://www.cs.waikato.ac.nz/ml/weka/

4 Classification Framework

We used the FFNNs for the mood classification purpose. It was observed that the FFNNs give better accuracy as compared to other machine learning algorithms like Support Vector Machines (SVMs) and Decision Trees (Patra et al., 2015b). Patra et al., (2015a) achieved low root mean square value for *arousal* and *valence* calculation using FFNNs. Moreover, they (Patra et al., 2015b) also reported higher F-measure for Hindi music mood classification based on audio.

4.1 Feed Forward Neural Networks (FFNNs)

Feed Forward Neural Networks refer to a special topology of neural networks in which each neuron belonging to a layer is connected to all the other neurons in the next layer. Neural networks are widely used for several classification and regression problems for its structural simplicity. The network is divided into multiple layers namely *input layer*, *hidden layer* and *output layer*. The *input layer* consists of inputs to the network. Then, the network follows a *hidden layer* which may consist of any number of *neurons* placed in parallel. Each neuron performs a weighted summation of the inputs which is then passed on to a nonlinear *activation function* (σ), also called the *neuron* function. Mathematically, the functionality of a hidden *neuron* is described as: $\sigma = \sum_{j=1}^{n}(w_j x_j + b_j)$, where the weights $\{w_j, b_j\}$ are symbolized with the arrows feeding into the neurons. The network output is formed by another weighted summation of the outputs of the neurons in the hidden layer (Mathematica Neural Networks- Train and Analyze Neural Networks to Fit Your Data, 2005). This summation on the output is called the output layer. The gradient descent learning principle is used to update the weights as the errors are back-propagated through each layer by the well-known back-propagation algorithm (Rumelhart et al., 1986). The updation rule can be stated as $\theta = \theta - \partial E/\partial\theta$.

4.2 Results Analysis and Discussion

We used FFNNs and LibSVM, a variant of the support vector machines (SVMs) implemented in Weka for the classification purpose. Several systems were developed using the audio features, lyric features and a combination of both. All experiments were conducted on the features selected by the correlation based feature selection technique. To obtain reliable accuracy, a 10-fold cross validation was performed for each of the classifiers.

4.2.1 Mood classification based on Audio

Initially, we performed experiments using only the timbre features and then added the other features incrementally. First we developed LibSVM based mood classification systems for Hindi and Western music. For Hindi music, the audio features based system achieved F-measure of 59.0, whereas for the Western music, the system achieved F-measure of 70.5 using all the audio features. The audio feature based Western music mood classification system achieved better F-measure (11.5 points absolute, 19.5% relative) than the Hindi-one. However, there were less number of instances present in the Western music as compared to the Hindi music. Therefore, we developed another pair of mood classification systems using the same number of instances for both the Hindi and Western music. In this case, the maximum F-measure of 56.8 and 64.5 were achieved for Hindi and Western music mood classification respectively using only audio features. The F-measure of the Western music mood classification system was 7.7 points absolute higher than the one for the Hindi music mood classification system developed on the same number of instances.

In the next phase, we developed audio based mood classification systems using FFNNs on the same set of 154 features. The maximum F-measure of 65.2 and 75.7 were achieved for the Hindi and Western music mood classification systems. The performance of the audio based systems are given in Table 4.

4.2.2 Mood classification based on Lyrics

We performed experiments using sentiment features initially and sequentially added other features incrementally. First, we used LibSVM for the classification purpose for feature ablation study. Subsequently, then we used the FFNNs using all the features together. We observed that the text stylistic features reduced the performance of the system. Thus, we removed text stylistic features from all other systems.

The lyric features based mood classification systems achieved the maximum F-measure of 55.3 and 68.2 for Hindi and Western song lyrics, respectively. The Western song mood classification achieves better F-measure of around 13.0 points absolute than the Hindi song mood classification system based only on lyric features. It was observed that the N-gram features yield good F-measure alone in case of the mood classification systems for Hindi and Western songs. The relative improvement of F-measure in case of Hindi song mood classification was much more than the mood classification system for Western songs. The main reason may be that the Hindi is free word order language and the Hindi lyrics are also more free in word order than the Hindi language itself.

We also developed lyric based systems for both song categories using the FFNNs. The corresponding mood classification systems achieved the maximum F-measure of 57.1 and 69.2 for Hindi and Western songs respectively. The performance of the lyric based systems are reported in Table 4.

4.2.3 Multimodal Music Mood classification

We developed multimodal music mood classification systems using LibSVM and FFNNs. The multimodal music mood classification system based on both audio and lyric features achieved F-measures of 68.9 and 80.4 for Hindi and Western songs using LibSVM. The multimodal music mood classification system for Western songs performs 11.5 points absolute better than the one for Hindi songs in terms of F-measure.

The FFNNs based multimodal music mood classification systems achieved the maximum F-measures of 75.1 and 83.5 for Hindi and Western songs, respectively. The performance of the multimodal systems are shown in Table 4 and the confusion matrices for these multimodal music mood classification systems are given in Table 5.

From the confusion matrix, it is observed that multimodal mood classification system for Hindi songs performs better in case of "*Class_Sa*" and performs poorly in case of "*Class_Ha*". This is obvious since the number of instances are more in case of "*Class_Sa*". The maximum number of instances from "*Class_Ha*" are classified as other classes because of the similar audio and lyric features. We also observed that the systems for Hindi songs are quite biased towards the "*Class_Sa*". In case of the Western songs, the "*Class_Ca*" contains the maximum number of instances and thus maximum number of instances are classified correctly for "*Class_Ca*". The system performs better in case of the "*Class_An*" and performs poorly in case of the "*Class_Ex*". It was also observed that some of the instances from each of the classes have tendency to go towards its neighboring classes. The main reason may be the similar features in between the neighbor classes.

4.2.4 Comparison with other systems

The proposed mood classification system for Hindi songs performs poorly as compared to the system of (Ujlambkar and Attar, 2012) which achieved F-measures of 75 to 81 using only audio based features. They used different mood taxonomy and they sliced the songs into 30 second clips. Unfortunately, their dataset is not freely available for research purpose. The features used for the experiments in (Ujlambkar and Attar, 2012) are a subset of our features. The audio based Hindi music mood classification system performs poor as compared to the system developed in (Patra et al., 2015b). Patra et al., (2015b) used more number of instances, but used a subset of our featureset. The audio based mood classification system outperformed other audio based systems reported in (Patra et al., 2013a; Patra et al., 2013b), but they developed their systems using smaller dataset and less number of features.

Our lyrics based mood classification system for Hindi songs outperformed the system reported in (Patra et al., 2015c) by 18.6 points absolute in terms of F-measure. We used similar features, but the number of instances were more in case of the present system. The significant difference in experimental setup is that their dataset was annotated with mood classes after listening to the corresponding audio files, whereas our lyrics dataset was annotated after reading the lexical content of lyrics. To the best of our knowledge, currently there is no other lyrics based mood classification system for Hindi music available in the literature. Patra et al., (2016a) developed multimodal mood classification system for Hindi songs using LibSVM and achieved F-measure of 68.6, which is 0.3 point absolute less than our multimodal mood classification system for Hindi songs using the same LibSVM. The main reason may be that we

Table 4: Performance of the mood classification systems with respect to different features using LibSVM and FFNNs

		Hindi Music			Western Music		
Systems	**Features**	P	R	F	P	R	F
Audio Features using LibSVM	**Timbre**	55.2	54.5	54.8	63.7	63.2	63.4
	Timbre+Intensity	55.7	55.3	55.5	66.9	66.6	66.8
	Timbre+Intensity+ Rhythm	58.8	57.8	58.2	70.3	70.0	70.2
	All audio features	58.9	59.1	59.0	70.5	70.5	70.5
Audio Features using FFNNs	**All audio features**	65.3	65.1	**65.2**	75.8	75.7	**75.7**
Lyrics Features using LibSVM	**SL**	41.3	39.4	40.4	60.0	59.6	59.8
	SL+TSF	38.6	38.7	38.6	59.7	59.9	59.8
	NG	46.3	46.7	46.5	60.2	60.3	60.2
	SL+TSF+NG	55.3	52.8	54.1	68.2	68.3	68.2
	SL+NG	55.9	54.7	55.3	68.2	68.3	68.2
Lyrics Features using FFNNs	**SL+NG**	57.2	57.0	**57.1**	69.3	69.1	**69.2**
Multimodal using LibSVM	**Audio+ Lyrics(Excluding TSF)**	69.2	68.6	68.9	80.3	80.5	80.4
Multimodal using FFNNs	**Audio+ Lyrics(Excluding TSF)**	76.8	73.5	**75.1**	84.8	82.2	**83.5**

used more number of instances for the present system. Till date, to the best of the author's knowledge, no other multimodal system has also been developed for Hindi songs based on audio and lyric features.

Table 5: Confusion matrix for multimodal systems using FFNNs

	Hindi Songs					Western Songs				
Classified as –>	**a**	**b**	**c**	**d**	**e**	**a**	**b**	**c**	**d**	**e**
Class_An = a	**153**	11	18	7	14	**195**	2	23	3	7
Class_Ca = b	1	**185**	3	35	28	0	**208**	2	12	25
Class_Ex = c	37	10	**192**	12	7	25	2	**192**	12	5
Class_Ha = d	5	35	10	**170**	12	3	9	16	**182**	8
Class_Sa = e	14	37	2	8	**224**	9	12	2	6	**151**

For Western music, it is very difficult to compare our mood classification system with other systems available in the literature, as our mood taxonomy is totally different from the mood taxonomy proposed by the existing multimodal mood classification systems (Hu and Downie, 2010a; Hu and Downie, 2010b). The number of mood classes present in their taxonomy is much higher (eighteen) than ours. Taking this into consideration, our present Western music mood classification system performed better than those systems. We used almost similar audio and lyric features as compared to the above mentioned systems. They used sentiment lexicons like General Inquirer, ANEW and WordNet-Affect, whereas we used SentiWordNet and WordNet-Affect for identifying the sentiment words.

4.2.5 Observations

Each mood classification system for Western songs outperforms the corresponding mood classification system for Hindi songs developed with the same classifier. The reasons for such results are listed below.

1. The moods experienced during listening to audio and reading the corresponding lyric are different in case of Hindi songs.

2. The mood is not very clear in the first 60 seconds clip of a Hindi song. Starting 20-30 seconds of the first clip of a song is mostly calm.

3. Western songs are usually much more rhythmic than Hindi songs.

4. The intensity of the mood felt in case of reading a lyric is less than the intensity of the mood felt at the time of listening to the audio in both song types.

5. We need more sophisticated features for audio to identify the mood in case of the Hindi music.

5 Conclusions

We developed mood annotated multimodal (lyrics and audio) datasets for Hindi and Western songs. Based on these multimodal datasets, we developed automatic multimodal music mood classification systems using LibSVM and FFNNs. The best performing systems developed using FFNNs achieved the maximum F-measures of 75.1 and 83.5 for Hindi and Western songs, respectively. It was observed that the different moods were perceived by the annotators while listening to audio and reading the corresponding song lyric in case of the Hindi songs. The main reason for such difference may be that the audio and lyrics were annotated by different annotators. Another reason may be that the mood is not transparent in lyrics as compared to the mood present in the audio of the corresponding song. In future, we intend to perform deeper analysis of the listener's and reader's perspectives of mood aroused from songs. We would also like to collect more instances for mood annotated datasets. We are also planning to use bagging and voting approach for the classification purpose.

Acknowledgments

The work reported in this paper is supported by a grant from the "Visvesvaraya Ph.D. Scheme for Electronics and IT" funded by Media Lab Asia of Ministry of Electronics and Information Technology (MeitY), Government of India.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. *LREC*, 2200–2204.

Akshat Bakliwal, Piyush Arora, and Vasudeva Varma. 2012. Hindi subjective lexicon: A lexical resource for hindi polarity classification. *LREC*.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales *Educational and Psychological Measurement* 20(1):37–46.

Dipankar Das, Soujanya Poria, and Sivaji Bandyopadhyay. 2012. A classifier based approach to emotion lexicon construction. *Natural language processing and information systems*, 320–326.

Florian Eyben, Martin Wöllmer, and Björn Schuller. 2010. Opensmile: the munich versatile and fast open-source audio feature extractor. *Proceedings of the 18th ACM international conference on Multimedia*, 1459–1462, ACM.

Mark A. Hall. 1999. Correlation-based feature selection for machine learning. *PhD dissertation*, The University of Waikato.

Xiao Hu, J. Stephen Downie, Cyril Laurier, Mert Bay, and Andreas F. Ehmann. 2008. The 2007 MIREX audio mood classification task: Lessons learned. *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, 462–467.

Xiao Hu and J. Stephen Downie. 2010a. When Lyrics Outperform Audio for Music Mood Classification: A Feature Analysis. *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, 619–624.

Xiao Hu and J. Stephen Downie. 2010b. Improving mood classification in music digital libraries by combining lyrics and audio. *Proceedings of the 10th Annual Joint Conference on Digital Libraries*, 159–168.

Aditya Joshi, A. R. Balamurali, and Pushpak Bhattacharyya. 2010. A fall-back strategy for sentiment analysis in Hindi: a case study. *Proceedings of the 8ht International Conference on Natural Language Processing (ICON -2010)*.

Cyril Laurier, Jens Grivolla, and Perfecto Herrera. 2008. Multimodal music mood classification using audio and lyrics. *Proceedings of the 7th International Conference on Machine Learning and Applications (ICMLA'08)*, 688–693, IEEE.

Lie Lu, Dan Liu, and Hong-Jiang Zhang. 2006. Automatic mood detection and tracking of music audio signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 5–18.

Cory McKay, Ichiro Fujinaga, and Philippe Depalle 2005 jAudio: A feature extraction library. *Proceedings of the International Conference on Music Information Retrieval*, 600–603.

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2013. Automatic Music Mood Classification of Hindi Songs. *Proceedings of the 3rd Workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2013)*, 24–28.

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2013. Unsupervised approach to Hindi music mood classification. *Mining Intelligence and Knowledge Exploration*, 62–69.

Braja G. Patra, Promita Maitra, Dipankar Das, and Sivaji Bandyopadhyay. 2015. MediaEval 2015: Feed-Forward Neural Network based Music Emotion Recognition. *Proceedings of MediaEval 2015 Workshop*.

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2015. Music Emotion Recognition System. *Proceedings of the International Symposium Frontiers of Research Speech and Music (FRSM-2015)*, 114–119.

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2015. Mood Classification of Hindi Songs based on Lyrics. *Proceedings of the 12th International Conference on Natural Language Processing (ICON-2015)*.

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016. Multimodal Mood Classification Framework for Hindi Songs. *Computación y Sistemas*, 20(3):515-526.

Braja G. Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016. Labeling Data and Developing Supervised Framework for Hindi Music Mood Analysis. *Journal of Intelligent Information Systems*.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323:533–536.

James A. Russell. 1980. A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161-1178.

Carlo Strapparava and Alessandro Valitutti. 2004. WordNet Affect: an Affective Extension of WordNet. *LREC*, 4:1083–1086.

Aniruddha M. Ujlambkar and Vahida Z. Attar. 2012. Mood classification of Indian popular music. *Proceedings of the CUBE International Information Technology Conference*, 278–283.

Menno Van Zaanen and Pieter Kanters. 2010. Automatic Mood Classification Using TF*IDF Based on Lyrics. *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, 75–80.

Mathematica Neural Networks- Train and Analyze Neural Networks to Fit Your Data. 2005. *Wolfram Research Inc., First Edition*, Champaign, Illinois, USA.

Detecting Context Dependent Messages in a Conversational Environment

Chaozhuo Li[†], **Yu Wu**[†], **Wei Wu**[‡], **Chen Xing**[◇], **Zhoujun Li**[†], **Ming Zhou**[‡]
[†]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[‡] Microsoft Research, Beijing, China
[◇] Nankai University, Tianjin, China
{lichaozhuo,wuyu,lizj}@buaa.edu.cn {wuwei,v-chxing,mingzhou}@microsoft.com

Abstract

While automatic response generation for building chatbot systems has drawn a lot of attention recently, there is limited understanding on when we need to consider the linguistic context of an input text in the generation process. The task is challenging, as messages in a conversational environment are short and informal, and evidence that can indicate a message is context dependent is scarce. After a study of social conversation data crawled from the web, we observed that some characteristics estimated from the responses of messages are discriminative for identifying context dependent messages. With the characteristics as weak supervision, we propose using a Long Short Term Memory (LSTM) network to learn a classifier. Our method carries out text representation and classifier learning in a unified framework. Experimental results show that the proposed method can significantly outperform baseline methods on accuracy of classification.

1 Introduction

Together with the rapid growth of social media such as Twitter and Weibo, the amount of conversation data on the web has tremendously increased. This makes building open domain chatbot systems with data-driven approaches possible. To carry on reasonable conversations with humans, a chatbot system needs to generate proper response with regard to users' messages. Recently, with the large amount of conversation data available, learning a response generator from data has drawn a lot of attention (Ritter et al., 2011; Shang et al., 2015; Vinyals and Le, 2015).

A key step to coherent response generation is determining when to consider linguistic context of messages. Existing work on response generation, however, has overlooked this step. They either totally ignores linguistic context (Ritter et al., 2011; Shang et al., 2015; Vinyals and Le, 2015) or simply considers context for every message (Sordoni et al., 2015b; Serban et al., 2015). The former case is easy to lead to irrelevant responses when users' input messages rely on the context information in previous conversation turns, while the latter case is costly (e.g., on memory and responding time) for building a real chatbot system and has the risk of bringing in noise to response generation especially when users want to end the current conversation topic and start a new one. According to our observation, there are two types of messages in a conversational environment. The first type is context dependent message, which means to reply to the message, one must consider previous utterances in the dialogue[1], while the second type is context independent message, which means even without the previous utterances, the message itself can still lead to a reasonable response. Table 1 compares the two types of messages using examples. In Case 1, "why do you think so" is a context dependent message. In order to reply to the message, one cannot ignore its linguistic context "I think it will rain tomorrow". On the other hand, in Case 2, "Well, what time is it now" is a context independent message, as one can give a reasonable response without looking at the previous turns. Distinguishing context dependent messages from context independent messages is important for building a good response generator. Missing linguistic context for context dependent

[1]Broadly speaking, context may not be limited to linguistic context. For example, a user's interest could also be a kind of context. As the first step, in this work, we only focus on "linguistic context".

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1990–1999, Osaka, Japan, December 11-17 2016.

Table 1: Two types of messages

Case 1 : a context dependent message	Case 2 : a context independent message
User : What will the weather be like tomorrow?	User : What are you doing?
Chatbot : I think it will rain tomorrow.	Chatbot : I am waiting for you to watch NBA.
User : ***Why do you think so?***	User : ***Well, what time is it now?***

messages will lead to nonsense response. For example, "because I love you" could also be a response for the message "why do you think so" if we only look at the message itself, but it is nonsense appearing in the dialogue of Case 1. Incorporating context information into context independent messages will increase the workload of a generation system and has the risk of bringing in noise to the generation process. For example, if we consider the context "NBA" for the message "Well, what time is it now", the chatbot will probably say something about "NBA" rather than answer the question with a time answer. Although detecting context dependent messages is crucial for building chatbot systems, there is limited understanding about it.

In this paper, we study this important but less explored problem. Instead of answering how to incorporate context information, we try to understand when we need the information. Therefore, our effort is complementary to the existing work on response generation. It can keep the existing generation algorithms context-aware and improve their efficiency and robustness to noise. The task is challenging, as messages in a conversational environment are usually short and informal, and evidence that can indicate a message is context dependent is scarce. For example, on 3 million post-response pairs crawled from Weibo, the average length of messages is 4.65. On such short texts, classic NLP tools such as POS Tagger and Parser suffer from bad performance (Derczynski et al., 2013; Foster et al., 2011) and it is difficult to explicitly extract features that are discriminative on the two types of messages. More seriously, there are no large scale annotations available for building a supervised learning procedure.

We consider leveraging the large amount of human-human conversation data available on the web to learn a message classifier. Our intuition is that a context dependent message has different linguistic context in different conversation sessions, therefore its responses could be more diverse on content than responses of a context independent message. To verify this idea, we study the distributions of responses of messages using conversation data crawled from social media and find that the length distribution of responses and the word distribution of responses are quite discriminative on the two types of messages. Based on this observation, for each message in the crawled data, we estimate the average length of responses, the entropy of the word distribution of responses, and the maximum mass of the word distribution of responses, and take these characteristics as weak supervision signals to learn a classifier. The classifier takes a message as input and can make prediction for any messages in a real conversation environment, even though the messages do not appear in the crawled data and characteristics like entropy are not available for them. We propose using a Long Short Term Memory (LSTM) architecture to learn the classifier. Our model represents message texts in a continuous vector space using a one-layer LSTM network. The text vectors are then provided as input to a two-layer feed-forward neural network to perform classification. The neural network architecture carries out feature learning and model learning in a unified framework, and thus can avoid explicit feature extraction which is difficult on short conversational messages. Our method leverages large scale weak supervision signals extracted from responses in social conversation data and can reach a satisfactory accuracy with only a few human annotations.

We conduct experiments on large scale English and Chinese conversation data mined from Twitter and Weibo respectively, and test the performance of our method on thousands of messages annotated by human labelers. Experimental results show that our method can significantly outperform baseline methods on accuracy of message classification on both of the two data sets.

We make the following contributions in this paper: 1) proposal of detecting context dependent messages in a conversational environment; 2) proposal of learning weak supervision signals from responses of messages using large scale conversation data; 3) proposal of using an LSTM architecture to learn a message classifier; 4) empirical verification of the proposed method on human annotated data.

2 Related Work

Our work lies in the path of building chatbot systems with data-driven approaches. Differing from traditional dialogue systems (cf., (Young et al., 2013)) which rely on hand-crafted features and rules to generate reply sentences for specific applications such as voice dialling (Williams, 2008) and appointment scheduling (Janarthanam et al., 2011) etc., recent effort focuses on exploiting an end-to-end approach to learn a response generator from social conversation data for open domain dialogue (Koshinda et al., 2015; Higashinaka et al., 2016). For example, Ritter et al. (Ritter et al., 2011) employed a phrase-based machine translation model for response generation. In (Shang et al., 2015; Vinyals and Le, 2015), neural network architectures were proposed to learning response generators from one-round conversation data. Based on these work, Sordoni et al. (Sordoni et al., 2015b) incorporated linguistic context into the learning of response generator. Serban et al. (Serban et al., 2015) proposed a hierarchical neural network architecture to building context-aware response generation. In this paper, instead of studying how to incorporate context into response generation, we consider the problem that when we need context in the process. Our work can keep the existing generation algorithms context-aware and at the same time improve their efficiency and robustness.

We employ a Recurrent Neural Network (RNN) architecture to learn a message classifier. RNN models (Elman, 1990), due to their capability of modeling sequences with arbitrary length, have been widely used in many natural language processing tasks such as language modeling (Mikolov et al., 2010) and tagging (Xu et al., 2015) etc. Recently, it is reported that Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014) as two special RNN models which can capture long term dependencies in sequences outperform state of the art methods on tasks like machine translation (Sutskever et al., 2014) and response generation (Shang et al., 2015). In this paper, we apply the LSTM architecture to the task of context dependent message detection. We append LSTM with a two-layer feed-forward neural network, thus feature learning and model learning can be carried out simultaneously.

Our work belongs to the scope of short text classification (Song et al., 2014). Existing applications of short text classification include query classification (Kang and Kim, 2003), tweet classification (Sriram et al., 2010), and question classification (Zhang and Lee, 2003). We study a new problem in short text classification: distinguishing context dependent messages from context independent messages in a conversational environment. The task is important for building open domain chatbot systems and has its unique challenges (e.g., new data structure). We tackle the challenges by leveraging the responses of messages and utilizing an LSTM network to conduct feature learning and model learning simultaneously.

3 Learning to Detect Context Dependent Messages

Suppose that we have a data set $\mathcal{D} = \{(m_i, y_i)\}_{i=1}^{N}$ where m_i is a message composed of a sequence of words $(w_{m_i,1}, \ldots, w_{m_i,n_i})$ and y_i is an indicator whose value reflects whether m_i is context dependent or not. Our goal is to learn a function $g(\cdot) \in \{-1, 1\}$ using $\mathcal{D}$, thus for any new message m, $g(\cdot)$ predicts m a context dependent message if $g(m) = 1$. To this end, we need to answer two questions: 1) how to construct $\mathcal{D}$; 2) how to perform learning using $\mathcal{D}$.

For the first question, we can crawl conversation data from social media like Twitter and ask human labelers to annotate the messages in the data. The problem is that human annotation is expensive and time consuming and therefore we cannot obtain a large scale data set for learning. To solve the problem, we automatically learn some weak supervision signals using responses of messages in social conversation data, and take the signals as $\{y_i\}$ in $\mathcal{D}$. For the second question, one straightforward way is first extracting shallow features such as bag-of-words and syntax from messages and then employing off-the-shelf machine learning tools to learn a model. The problem is that shallow features are not effective enough on representing semantics in short conversation messages, which will be seen in our experiments. We propose using a Long Short Term Memory (LSTM) architecture to learn a model from $\mathcal{D}$. The advantage of our approach is that it can avoid explicit feature extraction and large scale human annotations, and carry out feature learning and model learning in a unified framework.

3.1 Learning Weak Supervision Using Responses

Instead of requiring human annotations, we consider creating signals that are discriminative on the two types of messages from large scale social conversation data available on the web. Our intuition is that a context dependent message has different linguistic context in different conversation sessions, therefore, its responses could be more diverse on content than responses of a context independent message (one message may appear multiple times, and therefore it may correspond to multiple responses). Table 2 illustrates our idea with some examples from Twitter. The last column of the table represents the frequency of the message or the frequency of the response under the message. For each message, we show the top 5 most frequent responses. From the examples, we can see that a context dependent message tends to have divergent and uniformly distributed responses corresponding to different linguistic context, while the responses of a context independent message share relatively similar content and some content dominates the distribution.

Table 2: Responses of the two types of messages

Context dependent message : why	2196	Context independent message : Good night	644
Response 1 : I am kidding	7	Response 1 : Good night	47
Response 2 : He can be like mcdaniels for sixer	5	Response 2 : Goodnight	44
Response 3 : Because I say no	5	Response 3 : Night	23
Response 4 : I am tired	5	Response 4 : Sleep well	10
Response 5 : U will become dependent on them	5	Response 5 : Thank you	9

The examples inspire us to investigate some statistical characteristics that can reflect the diversity of responses. These characteristics could be good indicators of context dependent messages, and we can construct $\{y_i\}$ in $\mathcal{D}$ using the characteristics. We estimate the following statistical characteristics for each message using its responses, and examine how the characteristics are discriminative on the two types of messages using 1000 labeled messages from Twitter and Weibo respectively. The details of the labeled data will be described in our experiments.

Entropy: the first characteristic we investigate is the entropy of the word distribution of responses, which is a common measure for diversity. Given a word distribution $P = (p_1, p_2, \ldots, p_n)$, the entropy of the distribution is defined as

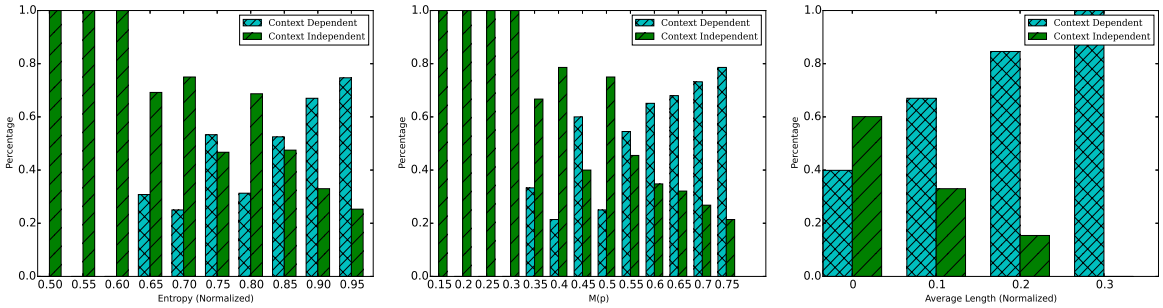
$$E(P) = \sum_{i=1}^{n} -p_i \log_2(p_i). \tag{1}$$

The maximum of the entropy is $\log_2(n)$ which is reached when the distribution is uniform. Then, a large entropy means a word distribution covers many words (i.e., n is big) and is close to a uniform distribution. Therefore, a context dependent message should have a larger entropy on responses than a context independent message (see the comparison in Table 2). We normalize the entropy to $[0, 1]$ by $\frac{E(P)-\min(E)}{\max(E)-\min(E)}$, where $\max(E)$ and $\min(E)$ represent the maximum entropy and the minimum entropy in the data set. Figure 1(a) shows the comparison of the two types of messages on normalized entropy using the Twitter labeled data. In the figure, each value on the x-axis represents an interval with a fixed length 0.05. For example, 0.50 means an interval $[0.5, 0.55)$. Each value on the y-axis represents the percentage of messages in a specific interval. For example, among messages falling in the interval $[0.95, 1)$, nearly 80% are labeled as context dependent and only about 20% are labeled as context independent. From the figure, we can see that entropy is discriminative on the two types of messages: context dependent messages distributes on large entropy areas, while context independent messages tend to have smaller entropy.

M(P): in addition to entropy, another characteristic that might reflect the diversity of responses could be the maximum mass of the word distribution of responses, as in diverse responses, words should be uniformly distributed (stopwords are removed), while in less diverse responses, there may exit dominant words (e.g., "night" in Table 2). Given a word distribution $P = (p_1, p_2, \ldots, p_n)$, we define a characteristic as

$$M(P) = 1 - \max_{1 \leqslant i \leqslant n} p_i \tag{2}$$

Figure 1(b) compares the two types of messages on $M(P)$ using the Twitter labeled data, in which values



(a) Comparison on entropy

(b) Comparison on $M(P)$

(c) Comparison on average length of responses

Figure 1: Comparison of the two types of messages on three characteristics.

on the x-axis and y-axis have the same meaning as those in Figure 1(a). From the figure, we can see that similar to entropy, $M(P)$ is useful on distinguishing the two types of messages. Context dependent messages have larger $M(P)$ than context independent messages.

Average length of responses: finally, we consider the length distribution of responses. Since responses of context dependent messages are more diverse on content, they might be longer than responses of context independent messages. We calculate the average length of responses for each message and normalize it to $[0, 1]$ in the same way as entropy. Figure 1(c) compares the two types of messages on average length of responses using the Twitter labeled data, where values on the x-axis represent intervals with a length 0.1. The result supports our claim and clearly indicates that average length is discriminative on the two types of messages.

We combine the three characteristics using a linear SVM classifier learned with the 1000 labeled messages and take the output of the SVM (a real value) as $\{y_i\}$ in $\mathcal{D}$. By this means, we can create a large scale training data set with only a little human labeling effort. Here, as a reference, we also report the classification accuracy of the three characteristics and the SVM classifier on the 1000 labeled data. Each characteristic corresponds to a threshold tuned on the 1000 labeled data with 5-fold cross validation. If a value of a characteristic of a message is larger than the threshold, then the message will be predicted as context dependent. Table 3 shows the classification accuracy of 5-fold cross validation (average of 5 results), where SVM (com) refers to the SVM classifier. Details of experiment setting will be described in Section 4. From Table 3, we can see that the numbers are consistent with Figure 1(a), 1(b), and 1(c).

Table 3: Classification accuracy on 1000 labeled data

	Weibo	Twitter
Entropy	72.6 %	70.5 %
$M(P)$	72.6 %	69.8 %
Average length of responses	72.8 %	68.5 %
SVM (com)	73.8 %	71.2 %

3.2 Model Learning

We head for learning $g(\cdot)$ using $\mathcal{D}$ constructed in Section 3.1. Note that $g(\cdot)$ only takes a message m as input, and thus can make prediction for any messages in a real chatbot system even though the messages are not in $\mathcal{D}$ and their entropy, M(P), and average length of responses are not available. Our idea is that we first learn a regression model by fitting $\{y_i\}$ in $\mathcal{D}$ through minimizing the sum of squared residuals and then construct $g(\cdot)$ by comparing the output of the regression model with a threshold. We can obtain the threshold by tuning it on a few labeled data (e.g., the 1000 labeled data). The key is how to learn the regression model. We propose using a Recurrent Neural Network (RNN) architecture to embed messages into a continuous vector space and learning a regression model with the embedding of messages using a feed-forward neural network. The RNN model, which is capable of embedding sequences with arbitrary

length, can encode the order of words and the semantics of a message into a vector representation which has been recently proven effective on capturing similarity of short texts (Sordoni et al., 2015a). We take the output vector given by RNN as a feature representation of a message and feed it to a feed-forward neural work. By this means, we can conduct feature learning and model learning in a unified framework and jointly optimize the two components.

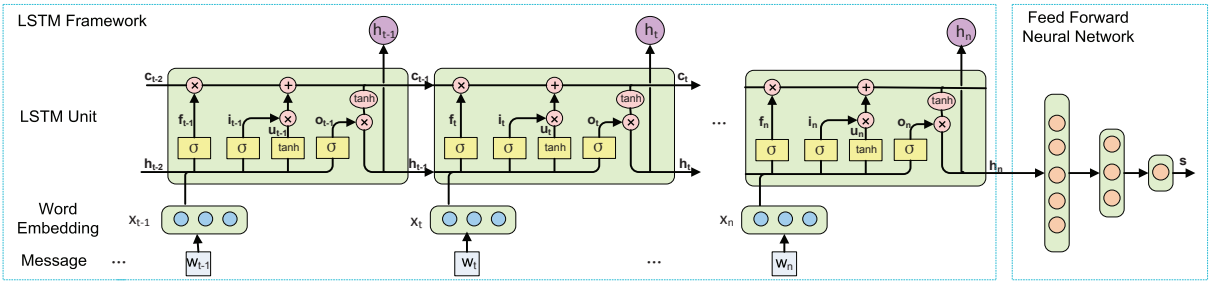


Figure 2: The architecture of our method

Given a message m which consists of n words, the RNN model reads the words one by one, and updates a recurrent state h_t for the t-th word w_t by

$$h_t = f(h_{t-1}, x_t),\ h_0 = 0, \tag{3}$$

where $h_t \in \mathbb{R}^{d_h}$, $x_t \in \mathbb{R}^{d_w}$ is the vector representation of w_t, and f is non-linear transformation. h_t acts as an encoding of the semantics of the word sequence up to position t, and the final output h_n is a representation of message m. Both x_t and h_t are learned in the optimization of the RNN model. We select the Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) as f, since it can model long term dependencies in sequences with affordable complexity. LSTM controls the learning of the representation of a sequence by gates. Specifically, at position t, LSTM controls the information that should be kept from previous states by an input gate i_t, and the information that should be forgotten by a forget gate f_t. After memorizing and forgetting, the information is stored in a memory cell c_t. c_t generates the recurrent state h_t through an output gate o_t. The specific parameterization of LSTM is given by

$$\begin{aligned}
i_t &= \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}) \\
f_t &= \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) \\
o_t &= \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}) \\
u_t &= \tanh(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}) \\
c_t &= i_t \otimes u_t + f_t \otimes c_{(t-1)} \\
h_t &= o_t \otimes \tanh(c_t),
\end{aligned}$$

where $\sigma(\cdot)$ is a sigmoid function and $\tanh(\cdot)$ is a hyperbolic tangent function. $W^{(i)}$, $W^{(f)}$, $W^{(o)}$, $W^{(u)}$ $\in \mathbb{R}^{d_h \times d_w}$, $U^{(i)}$, $U^{(f)}$, $U^{(o)}$, $U^{(u)} \in \mathbb{R}^{d_h \times d_h}$, and $b^{(i)}$, $b^{(f)}$, $b^{(o)}$, $b^{(u)} \in \mathbb{R}^{d_h \times 1}$ are parameters. $\otimes$ means element-wise multiplication. After we get the final state h_n, we feed it to a two-layer feed-forward neural network to get an output s which is defined by

$$s = b_2 + W_2\left(\tanh(b_1 + W_1 h_n)\right), \tag{4}$$

where $b_1 \in \mathbb{R}^{d_s \times 1}$, $W_1 \in \mathbb{R}^{d_s \times d_h}$, $W_2 \in \mathbb{R}^{1 \times d_s}$, and $b_2 \in \mathbb{R}$ are parameters. Figure 2 illustrates the architecture of our method.

For each m_i in $\mathcal{D}$, we calculate an s_i using Equation (4) as an estimation of y_i. We then learn the parameters of the LSTM network and the feed-forward network by minimizing the sum of the squared residuals. Formally, our learning approach can be formulated as

$$\arg\min_{s} \sum_{i=1}^{N} (y_i - s_i)^2. \tag{5}$$

After we obtain the parameters, we can calculate an s_m for any message m using Equation (4). We then tune a threshold T with a few labeled messages. The classifier $g(\cdot)$ is given by

$$g(m) = \begin{cases} 1 & \text{if } s_m > T \\ -1 & \text{otherwise} \end{cases} \quad (6)$$

The gradients of the objective function (5) are computed using the back-propagation through time (BPTT) algorithm (Williams and Peng, 1990). We share the code for model learning at `https://github.com/whatsname1991/coling2016`.

4 Experiments

4.1 Experiment Setup

We constructed the conversation data for experiments from Weibo and Twitter. In each of the two social media, two persons can communicate by replying to each other under a post. We crawled sequences of reply with posts and extracted triples like "(context, message, response)" as experimental data. In a triple, "message" is a reply, "context" is the sentence in the previous turn of the message (a reply or a post), and "response" is the sentence in the next turn (reply to the message). Note that in this work, we restrict the context of a message to a single sentence. This is a simplification of context in conversation. In real conversation, context could be more complicated and we leave the discussion of it as future work.

We crawled 5.9 million English triples from Twitter, and 3.1 million Chinese triples from Weibo. The numbers of distinct messages in the Twitter data and in the Weibo data are $92,755$ and $112,175$ respectively. On average, each Twitter message has 63.26 responses (some messages like "hello" can have many different responses) and each Weibo message has 27.52 responses. The average word length of Twitter message is 3.39 and the word average length of Weibo message is 4.65. English sentences were stemmed and stop words were removed, and Chinese sentences were segmented.

We constructed $\mathcal{D} = \{(m_i, y_i)\}_{i=1}^N$ in Section 3.1 in the following way: we first calculated entropy, $M(P)$, and average length of responses for each message using the 5.9 million English triples and 3.1 million Chinese triples. Then from these data, we randomly sampled 1000 English triples and 1000 Chinese triples as validation sets. For each triple in the validation data, we hid the response and recruited human judges to label if the message is context dependent or not. Note that we hid responses when labeling messages because this is more close to the real case. In a real chatbot system, one has to determine if a message is context dependent or not before generating a response. Each judge labeled a message with 1 if it is context dependent, otherwise the judge labeled the message with -1. Each message got three labels and the majority of the labels was taken as the final decision for the message. In the Weibo data, there are 412 positive examples and 588 negative examples. In the Twitter data, the two numbers are 440 and 560, respectively. With the two validation data sets, we learned two SVM classifiers in order to combine the three characteristics as described in Section 3.1. Parameters of SVMs were tuned by 5-fold cross validation. Finally, we assigned a y_i to each m_i in the $112,175$ Twitter messages and $92,755$ Weibo messages by the output of the SVM classifiers, and formed $\mathcal{D}$ for both English data and Chinese data. We trained LSTM models using $\mathcal{D}$.

To evaluate the performance of different models, we crawled another 3000 Chinese context-message pairs and 1000 English context-message pairs from Weibo and Twitter respectively, and followed the same way as the validation data to judge if the messages are context dependent or not. We used these data to simulate real context-message pairs in chatbot systems. In the Weibo data, there are 2715 unique messages and 1983 messages are not in $\mathcal{D}$. The numbers of positive examples and negative examples are 1472 and 1528 respectively. In the Twitter data, the number of unique messages is 875 and 366 messages are not included by $\mathcal{D}$. The numbers of positive and negative examples are 464 and 536 respectively. Note that for messages that are not included by $\mathcal{D}$, their characteristics (i.e., entropy, $M(P)$, and average length of responses) are not available, and we can only use classifiers whose features are extracted from messages (like our LSTM models) to make prediction. This is close to a real situation in chatbots, and we took the two data sets as test sets.

We considered the following methods as baselines:

Length: intuitively, short messages tend to be context dependent (e.g., "why" in Table 2). Therefore, we employed length of a message as a baseline. A message shorter than a threshold will be predicted as a context dependent message.

MDF: given a word, we estimated the number of messages that contain the word and named it "document frequency" (DF). We constructed a list of words associated with DF using $\mathcal{D}$. For a new message, we calculated the minimal DF of words in the message using the list. A context dependent message like "why do you think so" may consist of common words, and thus correspond to a high minimal DF. We considered minimal DF as a baseline. A message with a minimal DF larger than a threshold will be predicted as a context dependent message.

SVM (Length+MDF): we linearly combined Length and MDF by learning an SVM classifier on the validation data.

SVM (classification): we extracted unigrams, bigrams, and frequencies of POS tags as features from a message, and learned a linear SVM classifier on the validation data with these features. POS tags for Chinese data were obtained using Stanford Parser (`http://nlp.stanford.edu/software/lex-parser.shtml`) and POS tags for English data were obtained using TweetNLP (`http://www.cs.cmu.edu/~ark/TweetNLP/`).

SVM (regression): instead of learning a classifier from annotations in the validation data, we fitted $\{y_i\}$ in $\mathcal{D}$ by learning an SVM regression model using the same features as SVM (classification) and made predictions on new messages by a threshold.

All SVM models were learned using SVM-Light (`http://svmlight.joachims.org/`). We employed classification accuracy as an evaluation metric.

4.2 Parameter Tuning

For Length and MDF, the only parameter is a threshold. We tuned the thresholds on the validation data. For all SVM models, we selected the trade-off parameter in SVM from $\{0.01, 0.1, 1, 10, 100\}$ by 5-fold cross validation on the validation data. SVM (regression) also needs a threshold. We tuned it on the validation data. The parameters of LSTM include the dimension of word vectors d_w, the dimension of hidden states d_h, and the dimension of the first layer of the feed-forward network d_s. We set $d_w = d_h = 256$, and $d_s = 100$. Besides these parameters, we also set a dropout rate 0.1 in the learning of the feed-forward network as regularization.

Table 4: Accuracy on two test sets

	Weibo	Twitter
Length	62.6 %	61.3 %
MDF	62.1 %	58.6 %
SVM (Length+MDF)	63.0 %	62.2 %
SVM (classification)	66.8 %	65.4 %
SVM (regression)	64.3 %	68.3 %
LSTM	**75.6 %**	**73.4 %**

Table 5: Comparison between LSTM, SVM (classification), and SVM (regression)

Example	context : Have you heard Taylor Swift's new song? *message: Yep, I have heard it on Saturday night.*
Label	context dependent
SVM (regression)	context independent
SVM (classification)	context independent
LSTM	context dependent

4.3 Quantitative Evaluation

Table 4 reports quantitative evaluation results on the test data. From the results, we can see that our methods outperform baseline methods. The improvement over the best performing baseline methods (i.e., SVM (classification) on Webio and SVM(regression) on Twitter) is statistically significant (sign test, p-value < 0.01).

Length and MDF are characteristics of messages. The results tell us that these characteristics are not so discriminative on the two types of messages. The reason is easy to understand: we may think that context dependent messages tend to be short and consist of common words, but the fact is that short messages composed of common words could be context independent (e.g., "Good night" in Table 2) while long messages like "Yep, I have heard it on Saturday night" (see the example in Table 5) could be context

dependent. Both SVM (classification) and SVM (regression) perform worse than our LSTM model, indicating that shallow features are not effective enough to represent the semantics in short conversation messages. Our method outperforms the baseline methods on both data sets. The results verified our idea on leveraging responses for context dependent message detection, and demonstrates the power of big data and the advantage of LSTM on capturing semantics in short messages.

4.4 Qualitative Evaluation

We use an example to further explain why our method is effective on distinguishing the two types of messages. Table 5 compares LSTM with SVM (classification) and SVM (regression). Both SVM (classification) and SVM (regression) rely on shallow features such as bag of words and pos tags to perform learning. These features, however, are not effective on representing the semantics of short messages. The representation is easily to be biased by some specific words like "Saturday night" in the example. Therefore, both SVM (classification) and SVM (regression) failed on this case. On the other hand, LSTM models term dependencies in sequences with a memorizing-forgetting mechanism. It can capture the semantics in the message "Yep, I have heard it on Saturday night." and identify that it is similar to messages like "Yes, I did" and "Yes, I have". For example, the cosine of the vector of "Yep, I have heard it on Saturday night." and the vector of "Yes, I have" given by LSTM is 0.63. Since messages like "Yes, I did" and "Yes, I have" are common context dependent messages, LSTM can successfully recognize that the message in the example is also context dependent.

5 Conclusion

We propose learning a LSTM network with weak supervision signals estimated from responses of messages to detecting context dependent messages in a conversational environment. Evaluation results show that the proposed method can significantly outperform baseline methods on distinguishing the two types of messages.

Acknowledgement

This work was supported by Beijing Advanced Innovation Center for Imaging Technology (No.BAICIT-2016001), the National Natural Science Foundation of China (Grand Nos. 61370126, 61672081), National High Technology Research and Development Program of China (No.2015AA016004),the Fund of the State Key Laboratory of Software Development Environment (No.SKLSDE-2015ZX-16).

References

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP*.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *RANLP*, pages 198–206.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Jennifer Foster, Özlem Çetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef Van Genabith. 2011. # hardtoparse: Pos tagging and parsing the twitterverse. In *AAAI 2011 Workshop on Analyzing Microtext*, pages 20–25.

Ryuichiro Higashinaka, Nozomi Kobayashi, Toru Hirano, Chiaki Miyazaki, Toyomi Meguro, Toshiro Makino, and Yoshihiro Matsuo. 2016. Syntactic filtering and content-based retrieval of twitter sentences for the generation of system utterances in dialogue systems. In *Situated Dialog in Speech-Based Human-Computer Interaction*, pages 15–26. Springer.

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

Srinivasan Janarthanam, Helen Hastie, Oliver Lemon, and Xingkun Liu. 2011. The day after the day after tomorrow?: a machine learning approach to adaptive temporal expression generation: training and evaluation with real users. In *Proceedings of the SIGDIAL 2011 Conference*, pages 142–151. Association for Computational Linguistics.

In-Ho Kang and GilChang Kim. 2003. Query type classification for web document retrieval. In *SIGIR*, pages 64–71. ACM.

Makoto Koshinda, Michimasa Inaba, and Kenichi Takahashi. 2015. Machine-learned ranking based non-task-oriented dialogue agent using twitter data. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 3, pages 5–8. IEEE.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *EMNLP*, pages 583–593. Association for Computational Linguistics.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Building end-to-end dialogue systems using generative hierarchical neural network models. *arXiv preprint arXiv:1507.04808*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Ge Song, Yunming Ye, Xiaolin Du, Xiaohui Huang, and Shifu Bie. 2014. Short text classification: A survey. *Journal of Multimedia*, 9(5):635–643.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015a. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015b. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.

Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. 2010. Short text classification in twitter to improve information filtering. In *SIGIR*, pages 841–842. ACM.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Ronald J Williams and Jing Peng. 1990. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501.

Jason Williams. 2008. Demonstration of a pomdp voice dialer. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Demo Session*, pages 1–4. Association for Computational Linguistics.

Wenduan Xu, Michael Auli, and Stephen Clark. 2015. Ccg supertagging with a recurrent neural network. In *ACL'15*, volume 2, pages 250–255.

Stephanie Young, Milica Gasic, Blaise Thomson, and John D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *SIGIR*, pages 26–32. ACM.

Joint Inference for Mode Identification in Tutorial Dialogues

Deepak Venugopal
Department of Computer Science
University of Memphis
Memphis, TN 38152
dvngopal@memphis.edu

Vasile Rus
Department of Computer Science
Institute for Intelligent Systems (IIS)
University of Memphis
Memphis, TN 38152
vrus@memphis.edu

Abstract

Identifying *dialogue acts* and *dialogue modes* during tutorial interactions is an extremely crucial sub-step in understanding patterns of effective tutor-tutee interactions. In this work, we develop a novel joint inference method that labels each utterance in a tutoring dialogue session with a dialogue act and a specific mode from a set of pre-defined dialogue acts and modes, respectively. Specifically, we develop our joint model using Markov Logic Networks (MLNs), a framework that combines first-order logic with probabilities, and is thus capable of representing complex, uncertain knowledge. We define first-order formulas in our MLN that encode the inter-dependencies between dialogue modes and more fine-grained dialogue actions. We then use a joint inference to jointly label the modes as well as the dialogue acts in an utterance. We compare our system against a pipeline system based on SVMs on a real-world dataset with tutoring sessions of over 500 students. Our results show that the joint inference system is far more effective than the pipeline system in mode detection, and improves over the performance of the pipeline system by about 6 points in F1 score. The joint inference system also performs much better than the pipeline system in the context of labeling modes that highlight important pedagogical steps in tutoring.

1 Introduction

One-on-one instruction, i.e. tutoring, is one of the most effective forms of instruction. Intelligent Tutoring Systems (ITS) (Rus et al., 2013) have the potential to make effective and affordable "instruction-for-all" a reality since they do not suffer from traditional constraints such as lack of trained and expensive human tutors, physical teaching facilities, etc. However, in order to build effective automated tutoring systems, i.e. tutoring systems the induce student learning gains, we first need to understand what effective human tutors do. Specifically, we would like to identify specific pedagogical steps that promote effective tutoring. For instance, a good tutor may start by building a rapport with the students, followed by helping the student identify the domain of the problem, and so on. The sequence of steps taken by expert human tutors can in turn be used to improve the performance of ITS by re-enacting such effective tutorial strategies that are likely to promote better learning.

Understanding what good tutors do to help students learn has been the subject of much theoretical and empirical research (Chi et al., 2001; Eugenio et al., 2006; Cade et al., 2008; Jeong et al., 2008; Boyer et al., 2010; Lehman et al., 2012). A standard approach to understanding effective tutoring is to characterize tutor-tutee interactions based on the actions tutors and tutees take and then identify patterns of such actions that are associated with effective tutoring. For instance Cade et al. (Cade et al., 2008) used dialogue acts, which are constructs used to describe the intentions behind speakers' utterances, to model tutor-learner dialogue-based interactions. Boyer et al. (Boyer et al., 2010) modeled interactions as a combination of both task actions, which specify fine-grained steps taken by a user such as opening a file, and dialogue acts. However, dialogue acts only identify individual, isolated acts, e.g. asking a question, associated with a particular utterance lacking to characterize the meaning of a sequence of coherent actions, e.g. by the tutor, that might reveal high level constructs such as pedagogical strategies,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2000–2011, Osaka, Japan, December 11-17 2016.

Speaker	Utterance	Act	Subact	Mode
Tutor	Welcome	Expressive	Greeting	Opening
Student	Hi	Expressive	Greeting	Opening
Tutor	How can i help you today?	Prompt	Question	Rapport Building
Tutor	you can just write the problem on board	Assertion	Process	Process Negotiation
Student	okay	Expressive	Neutral	Process Negotiation
Tutor	so we need to find the slope-intercept form	Assertion	Identification	Problem Identification
Student	are we asked for the graph or just the equation	Question	Neutral	Problem Identification
Tutor	We have slope given as m =0	Assertion	Calculation	Scaffolding
Student	I graphed the intercept (0, -27) correctly	Assertion	Calculation	Scaffolding
Tutor	and the y-intercept is (0, -27)	Assertion	Calculation	Scaffolding
Student	the slope is 0	Assertion	Calculation	Scaffolding

Table 1: Example for modes, acts and subacts in a dialogue.

e.g. scaffolding. In this work, we present a novel approach to identify higher-level tutorial constructs called *modes* in tutor-learner dialogue-based interactions. Specifically, dialogue modes are sequences of dialogue acts that map to pedagogical goals such as scaffolding (and sometimes to general dialogue goals such as opening a conversation). An example of the hierarchy of modes, dialogue acts and subacts in utterances is shown in Table 1.

As is often the case in several NLP tasks, a *pipeline* architecture can be naturally adopted to identify dialogue modes. Specifically, we first label acts in each utterance of the dialogue, then, using the labeled acts, we label subacts, and using both the labeled acts and subacts, we finally label the higher level modes. However, as is the case in general with pipeline based architectures, such a system is bound to have a fair amount of *error propagation*, where errors in labeling the acts or subacts affect the performance of mode labeling. Therefore, we propose a novel *joint inference* method for this task where we label modes jointly with dialogue acts and subacts, thereby taking advantage of the inter-dependencies between them. Prior approaches in the ITS research community have largely focused on dialogue act classification (Marineau et al., 2000; Serafin and Di Eugenio, 2004; Moldovan et al., 2011) or on mode labeling given labeled dialogue acts (Cade et al., 2008; Boyer et al., 2010; Rus et al., 2015). To the best of our knowledge, our work is the first joint inference method for this task.

We develop our joint inference system using a modeling language called Markov Logic Networks (MLNs) (Domingos and Lowd, 2009). MLNs are a powerful representation, where uncertain domain-knowledge is encoded as first-order formulas with weights attached to each formula. The weights in an MLN model indicate the uncertainty associated with the formulas. The larger the weight, the more confidence we have in the formula being true. Over the last few years, MLNs have been routinely used for several joint inference tasks in entity resolution (Poon and Domingos, 2008), event extraction (Poon and Vanderwende, 2010; Venugopal et al., 2014) and question answering (Khot et al., 2015). The main advantage of MLNs is that it can represent a large, complex probabilistic model through a highly compact, lifted representation specified through first-order formulas. However, at the same time, the compact representation makes scaling up probabilistic inference and learning a huge challenge in MLNs (Domingos and Lowd, 2009; Poon and Domingos, 2007). More specifically, in our task, the Markov network underlying the MLN turns out to be extremely large with millions of nodes and edges. By systematically exploiting the structure of our MLN model, we scale up MLN learning and inference methods for our task.

We evaluate our joint inference model on a dataset of human annotated dialogue transcripts of 500 students with around 32,000 dialogue utterances. To compare against our approach, we build a baseline, pipeline system using Support Vector Machines where we treat utterances as independent instances and sequentially label dialogue acts, subacts and modes in this dataset. We then compare our joint inference model with the baseline and obtain nearly a 6 point increase in F1-score for mode labeling with both higher recall and higher precision, clearly showing the promise of our joint inference approach.

The rest of this paper is organized as follows. We first present related work and give a brief overview of MLNs. We then present our joint inference model using MLNs and finally conclude with our evaluation.

2 Related Work

Speech-act theory that was developed in in the 1960's (Austin, 1962; Searle, 1969) has been typically used to model speakers' intentions. According to speech-act theory, when we say something, we do something. There are three levels of speech: the locutionary level which is the actual utterance, the illocutionary level which is the intention behind the utterance and perlocutionary level which is the effect of the utterance. Speech acts model the illocutionary level and denote speech acts such as greeting (*Hello*), questioning (*how is the weather?*), etc.

A speech act could be described as the sum of the illocutionary forces carried by an utterance (Moldovan et al., 2011). It is worth mentioning that within one utterance, speech acts can be hierarchical, hence the existence of a division between direct and indirect speech acts, the latter being those by which one says more than what is literally said, in other words, the deeper level of intentional meaning. In the phrase, *Would you mind passing me the salt?*, the direct speech act is the request best described by *Are you willing to do that for me?* while the indirect speech act is the request *I need you to give me the salt*. In a similar way, in the phrase, *Bill and Wendy lost a lot of weight with a diet and daily exercise*. the direct speech act is the actual statement of what happened, i.e., *They achieved "this" by doing "that"*, while the indirect speech act could be the encouraging, *If you do the same, you could lose a lot of weight too*. The present study assumes there is one direct speech act per utterance.

The task of classifying direct speech acts has been well-studied in the general context (Reithinger, 1995; Stolcke et al., 2000; Reithinger and Maier, 1995; Ries, 1999; Moldovan et al., 2011) as well as in the specific context of ITS (Marineau et al., 2000; Serafin and Di Eugenio, 2004; Samei et al., 2014). A related problem of generating the next speech act in a dialogue has also been investigated to some extent (Reithinger, 1995; Bangalore and Stent, 2009). Also, there is work on automatically discovering dialogue acts using data-driven approaches (Moldovan et al., 2011) but it is beyond the scope of this paper to automatically discover the dialogue acts in our tutoring sessions. In the automated speech act classification literature, typically researchers have considered rich feature sets extracted from the utterances such as the actual words (possibly lemmatized or stemmed) and ngrams (sequences of consecutive words) to characterize the type of speech act.

Dialogue modes in tutorial dialogues are sequences of dialogue acts that correspond to general conversational segments of a dialogue, e.g. an Opening mode corresponds to the first phase of the dialogue when the conversational partners greet each other, or to segments associated with pedagogical goals, e.g. a Scaffolding mode would correspond to the tutorial dialogue segment when the student works on something and the tutor scaffolds the learners activity. Compared to speech act classification, mode identification has been far less studied. Based on a manual analysis, Cade et al. (Cade et al., 2008) defined a set of eight mutually exclusive tutorial modes: introduction, lecture, highlighting, modeling, scaffolding, fading, off-topic, and conclusion. An interesting aspect of their analysis is the granularity at which they defined the pedagogically important modes. In their approach, the modes correspond to either the tutor or the student or both focusing on solving a full problem. In our approach, we used a different definition of modes proposed by Morrison et al. (Morrison et al., 2014). In this approach, a tutor or student could switch between proposed modes while working on a particular problem. That is, a particular mode is not associated with one problem solving task but rather with parts of such a problem solving task. Finally, Boyer et al. (Boyer et al., 2010) used acts in conjunction with more specific task actions, e.g., opening a specific file, etc., to discover hidden modes using a HMM. In contrast, we assume a pre-defined set of modes (see next section) that generalize across tutors and identify modes and acts jointly. This is similar to the Conditional Random Fields (CRF) approach proposed by Rus and colleagues (Rus et al., 2015) who used a expert-defined set of modes. It should be noted that Rus and colleagues report a best dialogue mode labeling performance of accuracy=57.18% when they used gold, i.e. human-labeled, dialogue acts as input. The accuracy dropped to 28.77% when automatically labeled dialogue acts were provided as input to the CRF-based dialogue mode labeling system. It should be noted that our results reported here are not exactly comparable to the ones reported by Rus and colleagues as they used a different, albeit related, human-labeled dataset to train and test their system.

3 Background

In this section, we give a brief overview of MLNs and describe the dataset used in this paper.

3.1 Markov Logic Networks

Markov logic networks (MLNs) unify first-order logic with Markov networks (undirected probabilistic graphical models abbreviated as PGMs). Formally, an MLN consists of a set of weighted first-order formulas, $\{(f_i; w_i)\}_{i=1}^{K}$, where f_i is a first-order formula and w_i is a real-valued weight attached to f_i. The weight w_i quantifies the uncertainty in f_i. Higher the weight of a formula, the more belief we have that the formula is true. If the weight w_i is ∞, then it acts as a hard constraint that f_i should always be true, while a weight $-\infty$ specifies the hard constraint that f_i should always be false. MLNs assume Herbrand semantics, i.e., there is a finite number of objects that can be substituted for the variables in the first-order formulas. This set of real-world objects is referred to as the domain. Throughout this paper, we specify constants with capital letters (e.g., A, B, etc.) and variables in the formulas with small letters (e.g., x, y, etc.)

A *ground atom* in the MLN is a first-order predicate where all variables are grounded with constants from the domain. Similarly, a ground formula is an instantiation of a first-order formula, where all variables have been grounded with constants from the domain. Given a domain of interest, MLNs specify a Markov network where a ground atom (a first-order predicate where all variables are grounded with constants) is a binary variable in the network and each ground formula (a first-order formula grounded with constants) is a function over the variables specific to that formula. For example, assume that the domain for the MLN, $\texttt{Smokes}(x) \Rightarrow \texttt{Cancer}(x)$; w, is equal to $\{A, B\}$. Then, $\texttt{Smokes}(A)$ is a ground atom in the MLN which represents a binary random variable in the Markov network. Similarly, $\texttt{Smokes}(A) \Rightarrow \texttt{Cancer}(A)$ represents a function in the Markov network defined over the binary variables corresponding to $\texttt{Smokes}(A)$ and $\texttt{Cancer}(A)$. An assignment (either 0 or 1) to all possible ground atoms in the MLN, $\texttt{Smokes}(A)$, $\texttt{Smokes}(B)$, $\texttt{Cancer}(A)$, $\texttt{Cancer}(B)$, is called a *world*. The MLN describes a *log-linear* model where the probability distribution is defined over the set of possible worlds. Specifically, the probability distribution represented by the MLN is given by,

$$\Pr(\omega) = \frac{1}{Z} \exp\left(\sum_i w_i N_i(\omega)\right) \tag{1}$$

where $N_i(\omega)$ is the number of groundings of the first order formula f_i that evaluate to True given a world ω.

Since MLNs are simply a compact representation of PGMs, all inference tasks in PGMs are also applicable to MLNs. Specifically, the two main inference tasks for MLNs are, 1) Marginal inference, and 2) MAP inference. In marginal inference, given evidence atoms, i.e., ground atoms whose truth value is known/observed, the task is to compute marginal probabilities over other query atoms. For example, say we are given evidence atoms $\texttt{Smokes}(A)$ and $\texttt{Smokes}(B)$, the task is to compute probabilities such as $P(\texttt{Cancer}(A)|\texttt{Smokes}(A), \texttt{Smokes}(B))$. In MAP inference, given evidence, we compute the assignment to the non-evidence atoms such that the probability of that assignment is maximized. For instance, given evidence $\texttt{Smokes}(A)$ and $\texttt{Smokes}(B)$, we need to compute the assignment to $\texttt{Cancer}(A)$, $\texttt{Cancer}(B)$ for which the probability is maximum in the joint distribution. Both marginal inference and MAP inference are computationally intractable and therefore typically approximate algorithms are used for both these tasks.

3.2 Dataset

Our dataset consists of dialogue transcripts of 500 tutoring sessions collected from 500 students working on elementary algebra and physics problems. In all, there are 32,368 individual utterances in these tutorial sessions, where we define an utterance as a single dialogue turn by either the student or the tutor. We selected this data from a sample of sessions obtained from an online, commercial tutoring service. These

sessions are about problem solving in the context of various Algebra and Physics topics. These are student-initiated sessions, mostly in the context of homework help.

We label each utterance with a predefined set of dialogue acts. The dialogue act taxonomy was developed with the assistance of subject matter experts, all experienced tutors and tutor mentors working for an online commercial tutoring service, resulting in a fine-grained 2-level hierarchical taxonomy that includes 17 main act categories. Each main dialog act category consists, in turn, of different subcategories, which we refer to as *subacts*, resulting in an overall taxonomy of 196 distinct dialog act-subact combinations. The size of the dialogue-act and -subact taxonomy is at least one order of magnitude larger than taxonomies proposed and used by others such as Boyer and colleagues (Boyer et al., 2010). It should be noted that the dialog acts were defined and refined to minimize overlap between categories and maximize the coverage of distinct acts.

There were a set of 17 dialogue modes defined by the experts and each utterance was annotated with the act, subact and the dialogue mode for the utterance by humans. The data was manually annotated by a group of tutoring experts who were trained on both the dialogue act taxonomy and set of dialogue modes. When annotating independently, the inter-annotator agreement was 80.91% and kappa statistic was 0.77 for dialogue acts and 64.90% and kappa of 0.63 for dialogue acts and subacts together. These values correspond to very good agreement among the annotators. For modes, the agreement was lower at 55.03% and kappa of 0.47. The list of modes and the number of times they occur in our labeled data set is shown below.

Opening(667), Problem Identification (3177), Assessment (338), Method Identification (126), Method Roadmap (1056), Rapport Building (1006), Process Negotiation (3281), MetaCognition (533), Sensemaking (2889), Fading (2466), Scaffolding (4574), Modeling (1159), Telling (1806), Session (8), ITSupport (1251), WrapUp/Close (871), and Off-topic (4).

4 MLN Model

Here, we describe our joint inference model for identifying dialogue modes based on MLNs. We first describe the set of first-order formulas of the joint model. We then discuss how we perform joint inference and learning scalably in our model.

4.1 MLN Formulas

The four main predicates in our MLN are: `Act`, `Subact`, `Mode` and `ModeSwitch`. We next describe each of these predicates.

`Act`(s, t, a!) is a predicate that asserts that the dialogue act in the tutorial session corresponding to student s, at time step t, is equal to a. When defining our MLN, we refer to "time" as the utterance number in a dialogue session between the tutor and student. The "!" mark is a special symbol in the MLN language that specifies a hard constraint that for every grounding of s and t, there is exactly one act label. That is, every utterance corresponds to one and only one dialogue act. Similarly, `Subact`(s, t, u!) asserts that the dialogue subact in the tutorial session for student s at time t is equal to u. `Mode`(s, t, m!) asserts that the dialogue mode in the tutorial session for student s at time t is equal to m. Finally, `ModeSwitch`(s, t) asserts that there was a switch in dialogue mode at time t for the tutoring session associated with student s. That is, the mode in the previous time step was different from the mode in the current time step. Since our interest in this task is mode identification, `Mode` is called as a *query* predicate and the inference task is to collectively set a 0/1 truth assignment to all groundings of this predicate. `Act`, `Subact` and `ModeSwitch` are called *hidden* predicates since the truth assignments of their groundings are unknown.

Using the above predicates, we define the following formulas. Unless specified, all variables in the below described formulas are assumed to be universally quantified.

1. The first set of hard formulas specify that each act maps to a specific subset of subacts. This formula encodes the two-level hierarchy that we define in our taxonomy. We specify this by implication formulas of the form,

$$\texttt{Act}(s, t, A) \Leftrightarrow \texttt{Subact}(s, t, U_1) \vee \ldots \texttt{Subact}(s, t, U_k)$$

where $U_1 \ldots U_k$ are possible subacts corresponding to act A.

2. Next, we define hard formulas that encode the rule of mode switching. That is, we specify that mode-switching causes a shift in the dialogue mode using two implications.

$$\texttt{ModeSwitch}(s,t) \wedge \texttt{Mode}(s,t-1,m) \Rightarrow \neg\texttt{Mode}(s,t,m)$$
$$\neg\texttt{ModeSwitch}(s,t) \wedge \texttt{Mode}(s,t-1,m) \Rightarrow \texttt{Mode}(s,t,m)$$

3. The first and last modes of a dialogue are always fixed. We specify this with a conjunctive hard formula,

$$\texttt{Mode}(s,T_0,Opening) \wedge \texttt{Mode}(s,T_k,Closing)$$

where T_0 is the first utterance in the dialogue and T_k is the last utterance in the dialogue.

4. We encode the inter-dependency between modes, acts and subacts with a set of soft formulas. Specifically, we model this interaction by encoding a formula that connects two successive time-steps of a dialogue. The resulting formulation is similar to encoding Hidden Markov Models using MLNs, where we assert that the dialogue mode at time-step t is influenced by the acts, modes and subacts at the previous time-step. Clearly, this is a formula which would not hold true for all possible instantiations. Therefore, we specify a soft formula of the form,

$$\texttt{Mode}(s,t-1,+m_1) \wedge \texttt{Act}(s,t-1,+a) \wedge \texttt{Subact}(s,t-1,+u) \Rightarrow \texttt{Mode}(s,t,+m_2)$$

An important aspect to note about the above soft formula is the "+" sign for variables in the formula. The "+" sign is a special symbol in MLNs that allows us to define multiple weights for a single formula. Recall that in MLNs, generally, all the groundings of a first-order formula share the exact same weight. However, in several practical cases, we need to decrease the bias of the model by introducing more parameters for it. With the use of a "+" sign, we can increase the total number of weights in the MLN and thus induce more complex distributions. Specifically, we can set a different weight for each partially ground formula obtained by grounding all variables in the formula corresponding to the $+$ symbol. For instance, in this case, for each possible grounding of the variables m_1, m_2, u and a in the formula, we will define a distinct weight. This allows us more degrees of freedom to model the data rather than using a single weight for the formula.

5. Next, we define several soft formulas that connect features of the dialogue utterances to `Mode`, `Act`, `Subact` and `ModeSwitch`. Let $\texttt{Feature}_1 \ldots \texttt{Feature}_N$ denote N features extracted from the utterances (we discuss the actual features in the next section), then, we encode these features using soft formulas of the form,

$$\texttt{Feature}_1(s,t,+f_1) \wedge \texttt{Feature}_2(s,t,+f_2) \ldots \texttt{Feature}_N(s,t,+f_N) \Rightarrow \texttt{Mode}(s,t,+m)$$
$$\texttt{Feature}_1(s,t,+f_1) \wedge \texttt{Feature}_2(s,t,+f_2) \ldots \texttt{Feature}_N(s,t,+f_N) \Rightarrow \texttt{ModeSwitch}(s,t)$$
$$\texttt{Feature}_1(s,t,+f_1) \wedge \texttt{Feature}_2(s,t,+f_2) \ldots \texttt{Feature}_N(s,t,+f_N) \Rightarrow \texttt{Act}(s,t,+a)$$
$$\texttt{Feature}_1(s,t,+f_1) \wedge \texttt{Feature}_2(s,t,+f_2) \ldots \texttt{Feature}_N(s,t,+f_N) \Rightarrow \texttt{Subact}(s,t,+u)$$

4.2 Joint Inference

Given the MLN specified in the previous section, the inference task is to jointly compute an assignment to all possible groundings of the query predicate, `Mode`. Specifically, we compute this assignment as a solution to the following optimization problem

$$\max_{\omega'} \sum_{h \in H} P(Q = \omega') \tag{2}$$

where H is the set ground atoms of hidden predicates and Q is the set of ground atoms of query predicates, ω' is an assignment on all atoms in Q. However, Eq. (2) which is an instance of the *marginal-MAP* (MMAP) inference problem involves both summation (summing out the hidden variables) and

maximization, and is well-known to be a very hard problem (Park and Darwiche, 2004). Instead, we approximate the solution to the MMAP problem with a solution to the following *Max a-posteriori* (MAP) inference problem which only involves maximization.

$$\max_{\omega} P(Q \cup H = \omega) \tag{3}$$

where ω is an assignment on all atoms in $Q \cup H$. To obtain an approximate MMAP assignment for only the atoms in Q, we simply project the complete solution obtained from the MAP problem in Eq. (3) on the atoms in Q. Note that even the MAP problem in Eq. (3) is NP-hard. However, several highly efficient off-the-shelf approximate MAP solvers can be used to obtain high-quality approximations. Notable examples include MaxWalkSAT (Kautz et al., 1997), dual-decomposition based solvers (Sontag and Globerson, 2011) and ILP based solvers such as Gurobi (Gurobi., 2013). In our experiments, we use Gurobi, a state-of-the-art ILP solver to compute the MAP solution for the MLN (Sarkhel et al., 2014). However, it turns out that a naive application of approximate MAP solvers to our problem is still infeasible in practice. For instance, suppose we have 500 students' dialogues in our dataset, and each dialogue has on average 100 utterances/time-steps, then, the formula, $\texttt{ModeSwitch}(s, t) \wedge \texttt{Mode}(s, t-1, m) \Rightarrow \neg \texttt{Mode}(s, t, m)$ itself has at least 1 million possible groundings. In other words, grounding the entire MLN and then applying MAP inference on the ground MLN quickly becomes infeasible. However, we notice that our MLN has a decomposable structure, i.e., the ground Markov network obtained by grounding the MLN with a single student's dialogue is independent of the ground Markov network obtained when we ground the MLN with the rest of students' dialogues. This means that we can decompose the MAP problem as,

$$\prod_{k} \max_{\omega_k} P(Q_k \cup H_k = \omega_k) \tag{4}$$

where Q_k and H_k are the query and hidden atoms specific to the dialogues of student the k-th student and ω_k is an assignment to all atoms in $\{Q_k, H_k\}$. Thus, using Eq. (4), we can essentially compute the MAP solution independently for each student dialogue using a standard MAP solver which greatly reduces the computational requirements of the solver and allows us to scale up joint inference over our large dataset of dialogues.

Next, we describe weight-learning for the soft formulas in our MLN. Specifically, we use gradient ascent to compute weights of the soft formulas that maximize the log-likelihood of our dataset. Note that, our model contains hidden variables that are not observed directly, i.e., atoms corresponding to `Act`, `Subact` and `ModeSwitch`. Due to the presence of these hidden variables in our model, the resulting log-likelihood function is no longer convex. Therefore, gradient ascent can get struck in local optima. We reduce the severity of the problem using *random restarts* (Selman et al., 1996). That is, we start gradient ascent from several different initialization points and average all the different weights that gradient ascent converges to when starting from these different initialization points. Note that in each step of gradient ascent, we need to compute the gradient as,

$$\mathbb{E}_w[N_i] - \mathbb{E}_w[N_i'] \tag{5}$$

where $\mathbb{E}_w[N_i]$ is the expected number of groundings of the i-th soft formula that are true given the current set of weights w w.r.t the MLN distribution $P(Q|H)$ and $\mathbb{E}_w[N_i']$ is the expected number of groundings of the i-th soft formula that are true in the dataset w.r.t the MLN distribution $P(Q)$. Both expectations are intractable to compute exactly. Therefore, we approximate these distributions with their respective MAP values. This means that, for each gradient ascent step, we run MAP inference twice and compute the approximate expectations and from the approximate expectations, we compute the approximate gradient direction. We continue updating the weights with the gradient until the weights converge. In order to reduce computation, we compute the weights only for a feasible set of groundings of the "+" variables in the soft formulas. For instance, consider soft formula 4 in the previous section. Here, the number of groundings of the "+" variables is equal to the product of $|Modes| * |Modes| * |Acts| *$

Classifier Type	Model Type	Features	#Classes
Act Classifier	$SVM^{multiclass}$	Unigrams, Bigrams, Number of Tokens, Ending Punctuation, Utterance number	17
Subact Classifier	$SVM^{multiclass}$	All features of Act Classifier, the output acts labeled by Act Classifier	61
Mode Classifier	$SVM^{multiclass}$	All features of Subact Classifier, the output Subacts labeled by Subact Classifier	17
Mode Switch Classifier	SVM	Merged features of Mode classifier for two successive utterances	2

Table 2: SVM Models for Act, Subact, Mode identification and Mode Switch detection.

Metric	Pipeline			Joint Model		
	Precision	Recall	F1	Precision	Recall	F1
Average	0.275	0.28	0.271	0.338	0.341	**0.332**
Weighted-Average	0.32	0.34	0.324	0.375	0.39	**0.378**

Table 3: 5-fold Cross Validation results for Mode labeling.

$|Subacts|$. However, the number of feasible combinations is much lower. That is, several combinations never occur in the training dataset and we assume that for all such cases, the weight is 0 (the likelihood that the formula is true/false is the same). We remove these cases from the set of ground formulas and compute weights only for the remaining set of feasible groundings.

4.3 Learning Feature Based Formulas

Unfortunately, the above weight learning procedure does not work very well to learn weights of the feature-based soft formulas (listed as 5. in the previous section). The number of weights that we need to learn corresponding to the feature-based formulas turns out to be extremely large. Specifically, grounding the "+" variables, we will have at least $O(N * d * |Modes| * |Acts| * |Subacts|)$, where N is the number of features, d is an upper-bound on the number of possible feature-values for a feature. For lexical features of the utterances such as unigrams, bigrams, etc. this number is extremely large. Thus, weight-learning for the MLN that includes the feature-based soft formulas is infeasible in our model. Instead, we utilize the flexibility of MLNs to incorporate the feature-based soft formulas implicitly. Specifically, we remove all the feature-based formulas from the MLN and learn the MLN weights using only the other formulas. We then derive weights for the feature-based formulas through a separate model and add this back into the MLN as described next.

We train an SVM-based pipeline system to label the acts, subacts, modes and mode-switches in sequence. That is, we use $SVM^{multiclass}$ to first label the acts. Using the labeled acts, we label the subacts, and using both the labeled acts and subacts, we label the modes. We detect mode-switches using a binary SVM classifier. The features used in each of these models are shown in Table 2. The SVM-based pipeline system yields confidence values in the form of hyper-plane distances for each dialogue utterance for every mode, act, subact and also whether a mode switch occurred. Specifically, given an utterance t for student s, we will have hyper-plane distances for `Act`(s,t,$+a$), `Subact`(s,t,$+u$), `Mode`(s,t,$+m$) and `ModeSwitch`(s,t) (Note that, these are the atoms in the RHS of the soft formulas specified in 5). We then add to the MLN, a unit clause corresponding to the RHS of each soft formula, with the weight of the unit clause given by the SVM confidence value, which we normalize into the range $[-1, 1]$. Thus, if the SVM classifier is confident that an utterance number t for student s has mode type M, it will output a large confidence value for the type M label, which in turn is encoded into the MLN as the formula `Mode`(s,t,M) with a large weight. This will then make it more likely that the atom `Mode`(s,t,M) will be set as true when computing the MAP solution for the overall joint model.

Type	Pipeline			Joint Model		
	Precision	Recall	F1	Precision	Recall	F1
Act	0.672	0.68	0.65	0.69	0.698	**0.68**
Subact	0.49	0.51	0.48	0.518	0.535	**0.513**

Table 4: 5-fold Cross Validation results for the hidden predicates, Act and Subact (weighted-average F1 scores).

5 Experiments

This section presents the details of our experimental setup and the results obtained. As already mentioned, we compared a pipeline approach with the MLN joint-inference approach.

5.1 Setup

We evaluate the performance of our joint model by comparing it with the SVM based pipeline system which uses the features outlined in Table 2. This system is similar to the one presented in Rus et al. (Rus et al., 2015) who used a related, but not identical dataset, except that Rus et al. use Conditional Random Fields to label the modes, while, here we use SVMs. The performance of Rus et al.'s mode identification system that uses the labels of acts and subacts that were detected using a supervised classifier, is similar to ones we present here.

For our joint model, we use Gurobi, a state-of-the-art ILP solver, to solve the MAP inference problem. That is, we ground the MLN with dialogue data from each student independently and solve the MAP problem for each such partially ground MLN independently using Gurobi. Note that this problem is embarrassingly parallel since each MAP solution can be computed independently of the others. Using this, we could run a single instance of MAP inference over the entire dataset in just a few minutes using a cluster of 5 8-core machines, each with 8GB RAM.

5.2 Results

Table 3 shows a comparison of the F1-scores, precision and recall obtained by running 5-fold cross validation. The scores are reported for simple average of the scores (average over all mode labels) and for the weighted average (average weighted by instances of a particular label). As seen here, the joint method clearly outperforms the pipeline method in every case, in terms of F1-score, precision and recall. The average F1-score we obtained using the joint method was nearly 6 points higher than the average F1-score obtained using the pipeline SVM classifier. Particularly, both precision and recall of mode identification improved over both metrics.

Next, we evaluated statistical significance of our results. Specifically, we ran 5-fold paired t-tests (cf. (Dietterich, 1998)) to determine if our results were significant. Our results showed that our results attained statistical significance at $p \leq 0.05$, i.e., we obtained $t = 3.75$ with $p = 0.009$.

In our next experiment, we evaluated the performance of our model on hidden predicates. Specifically, Table 4 shows a comparison of how well the systems perform in terms of labeling the hidden ground atoms (ground atoms of the `Act` and `Subact` predicates). Since joint inference takes advantage of inter-dependencies between modes, acts and subacts, the accuracy of labeling the hidden variables is also better in the joint model as compared to the pipeline SVM classifier. The improvement in act and subact labeling was slightly smaller than the improvement we got for our main task of mode labeling. However, as shown in Table 4, here again, we observed significant improvements in both precision and recall as compared to the pipeline system.

In our final experiment, we compared results over key pedagogical steps to evaluate the effect of joint inference in these steps. These results are shown in Table 5. The mode names are quite self-explanatory for Rapport Building, Problem Identification and Assessment. Scaffolding is a concept where the tutor scaffolds the learner who is working through the solution by giving hints. Sense Making is the concept of explanations for understanding purposes. Process Negotiation is discussing/confirming the process of how to go about solving the problem. As we see from the results, in most cases, the joint model is significantly

Mode-Label	Pipeline			Joint Model		
	Precision	Recall	F1	Precision	Recall	F1
Rapport Building	0.25	0.5	0.338	0.31	0.53	0.389
Scaffolding	0.2	0.34	0.258	0.22	0.54	0.312
Problem Identification	0.246	0.42	0.31	0.265	0.42	0.325
Assessment	0.06	0.34	0.11	0.28	0.27	0.28
Process Negotiation	0.279	0.47	0.35	0.275	0.56	0.37
Sense Making	0.20	0.21	0.21	0.22	0.32	0.26

Table 5: 5-fold Cross Validation results for modes important in tutoring.

better than the pipeline system. Particularly, in some cases such as Scaffolding, which is an important step that corrects learners when they are going in the wrong direction, there was nearly a 20 percent increase in recall. As such, in almost all modes, we observed improvements in both precision and recall, which clearly illustrates the benefit of our joint model.

6 Conclusion

In this paper, we presented a novel joint inference method to detect modes in human-to-human tutoring. Specifically, modes are high level abstractions of dialogue speech acts, which give us a much deeper understanding of the underlying process by which natural language tutoring occurs. This is an important sub-step in designing Intelligent Tutoring Systems since strategies taken by expert human tutors can be adapted to AI-based tutors. In this work, we exploited inter-dependencies between lower-level dialogue acts and the higher-level modes using joint inference. Specifically, we developed a Markov Logic Network (MLN) to encode the the joint dependencies between dialogue acts, subacts and modes using weighted first-order logic formulas. We then developed a scalable MAP inference strategy for our model by partially grounding the MLN in each inference sub-step instead of pre-grounding the full MLN. We demonstrated the effectiveness of our approach on a real-world dialogue-based tutoring dataset collected from 500 students and annotated by multiple expert tutors. We showed that our MLN-based joint model outperforms a pipeline model that we built using SVMs that detects modes, acts and subacts independently of each other.

Future work includes mode detection without pre-specifying the dialogue acts and modes, i.e., automatically induce the dialogue acts and modes in the dialogue using non-parametric unsupervised machine learning methods. We will also apply joint inference to other complex sub-problems in Intelligent Tutoring Systems such as semantic similarity matching, automatically generate the best subsequent tutoring strategies, and generating hints to a student based on student response. We will also explore utilizing advanced lifted inference methods (Venugopal and Gogate, 2012; Venugopal and Gogate, 2014) in tutoring systems.

This work makes substantial contributions towards discovering effective tutorial strategies using data-driven approaches which in turn will contribute to the development of effective intelligent tutoring systems that could provide affordable, effective, one-on-one instruction to any learner of any age, anytime (24/7), anyhwhere as long as an Internet-connected device is available. The impact of such effective educational technologies will be far-reaching.

Acknowledgements

The authors would like to thank the University of Memphis for partially supporting this work. This work was also partially supported by a contract from the Advanced Distributed Learning Initiative of the United States Department of Defense (Award W911QY-15-C-0070). The authors would also like to thank the anonymous reviewers for their inputs.

References

J.L. Austin. 1962. *How to Do Things with Words*. Oxford Press.

Srinivas Bangalore and Amanda Stent. 2009. Incremental parsing models for dialog task structure. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 94–102.

Kristy Elizabeth Boyer, Eun Young Ha, Robert Phillips, Michael D. Wallis, Mladen A. Vouk, and James C. Lester. 2010. Dialogue act modeling in a complex task-oriented domain. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 297–305.

Whitney L. Cade, Jessica L. Copeland, Natalie K. Person, and Sidney K. D'Mello. 2008. Dialogue modes in expert tutoring. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems (ITS)*, pages 470–479.

Michelene T.H. Chi, Stephanie A. Siler, Heisawn Jeong, Takashi Yamauchi, and Robert G. Hausmann. 2001. Learning from human tutoring. *Cognitive Science*, 25(4):471–533.

Thomas G. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.*, 10(7):1895–1923.

P. Domingos and D. Lowd. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool, San Rafael, CA.

Barbara Di Eugenio, Trina C. Kershaw, Xin Lu, Andrew Corrigan-Halpern, and Stellan Ohlsson. 2006. Toward a computational model of expert tutoring: A first report. In *Proeedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 503–508. AAAI Press.

Gurobi. 2013. *Gurobi Optimizer Reference Manual*. Gurobi Inc.

Hogyeong Jeong, Amit Gupta, Rod Roscoe, John Wagster, Gautam Biswas, and Daniel Schwartz. 2008. Using hidden markov models to characterize student behaviors in learning-by-teaching environments. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems (ITS)*, pages 614–625.

H. Kautz, B. Selman, and Y. Jiang. 1997. A General Stochastic Approach to Solving Problems with Hard and Soft Constraints. In D. Gu, J. Du, and P. Pardalos, editors, *The Satisfiability Problem: Theory and Applications*, pages 573–586. American Mathematical Society, New York, NY.

Tushar Khot, Niranjan Balasubramanian, Eric Gribkoff, Ashish Sabharwal, Peter Clark, and Oren Etzioni. 2015. Exploring markov logic networks for question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 685–694.

Blair Lehman, Sidney K. D'Mello, Whitney L. Cade, and Natalie K. Person. 2012. How do they do it? investigating dialogue moves within dialogue modes in expert human tutoring. In *Proceedings of the 11th International Conference on Intelligent Tutoring Systems (ITS)*, pages 557–562.

J. Marineau, Peter Wiemer-Hastings, D. Harter, B. Olde, P. Chipman, A. Karnavat, V. Pomeroy, Arthur Graesser, and the TRG. 2000. Classification of speech acts in tutorial dialog. In *Proceedings of the workshop on modeling human teaching tactics and strategies at the Intelligent Tutoring Systems 2000 conference*, pages 65–71.

Cristian Moldovan, Vasile Rus, and Arthur C. Graesser. 2011. Automated speech act classification for online chat. In *The 22nd Midwest Artificial Intelligence and Cognitive Science Conference*, pages 23–29.

Donald Morrison, Benjamin Nye, Borhan Samei, Vivek Varma Datla, Craig Kelly, and Vasile Rus. 2014. Building an intelligent pal from the tutor.com session database phase 1: Data mining. In *Educational Data Mining 2014*.

James D. Park and Adnan Darwiche. 2004. Complexity results and approximation strategies for map explanations. *J. Artif. Intell. Res. (JAIR)*, 21:101–133.

H. Poon and P. Domingos. 2007. Joint Inference in Information Extraction. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, pages 913–918, Vancouver, Canada. AAAI Press.

H. Poon and P. Domingos. 2008. Joint Unsupervised Coreference Resolution with Markov Logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 649–658, Honolulu, HI. ACL.

Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*.

Norbert Reithinger and Elisabeth Maier. 1995. Utilizing statistical dialogue act processing in verbmobil. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, pages 116–121.

Norbert Reithinger. 1995. Some experiments in speech act prediction. In *In Proceedings of the AAAI Spring Symposium on Empirical Methods in Discourse*.

Klaus Ries. 1999. Hmm and neural network based speech act detection. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*.

Vasile Rus, Sidney K. D'Mello, Xiangen Hu, and Arthur C. Graesser. 2013. Recent advances in conversational intelligent tutoring systems. *AI Magazine*, 34(3):42–54.

Vasile Rus, Nobal Niraula, Nabin Maharjan, and Rajendra Banjade. 2015. Automated labelling of dialogue modes in tutorial dialogues. In *Proceedings of the Florida Artificial Intelligence Research Society Conference*, pages 205–210.

Borhan Samei, Li Li, Haiying, Fazel Keshtkar, Vasile Rus, and Arthur C. Graesser. 2014. Context-based speech act classification in intelligent tutoring systems. In *Proceeedings of The 12th International Conference on Intelligent Tutoring Systems*, pages 236–241.

Somdeb Sarkhel, Deepak Venugopal, Parag Singla, and Vibhav Gogate. 2014. Lifted MAP inference for Markov Logic Networks. *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS-14)*.

John R. Searle. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press.

B. Selman, H. Kautz, and B. Cohen. 1996. Local Search Strategies for Satisfiability Testing. In D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, pages 521–532. American Mathematical Society, Washington, DC.

Riccardo Serafin and Barbara Di Eugenio. 2004. Flsa: Extending latent semantic analysis with features for dialogue act classification. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 692–699.

David Sontag and Amir Globerson. 2011. Introduction to Dual Decomposition for Inference. *Optimization for Machine Learning*.

Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Comput. Linguist.*, 26(3):339–373.

Deepak Venugopal and Vibhav Gogate. 2012. On lifting the gibbs sampling algorithm. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1664–1672.

Deepak Venugopal and Vibhav Gogate. 2014. Evidence-based clustering for scalable inference in markov logic. In *Proceedings of Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD*, pages 258–273.

Deepak Venugopal, Chen Chen, Vibhav Gogate, and Vincent Ng. 2014. Relieving the Computational Bottleneck: Joint Inference for Event Extraction with High-Dimensional Features. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 831–843. ACL.

Dialogue Act Classification in Domain-Independent Conversations Using a Deep Recurrent Neural Network

Hamed Khanpour, Nishitha Guntakandla, and **Rodney Nielsen**
University of North Texas
HiLT Lab
{hamedkhanpour, nishithaguntakandla}@my.unt.edu
rodney.nielsen@unt.edu

Abstract

In this study, we applied a deep LSTM structure to classify dialogue acts (DAs) in open-domain conversations. We found that the word embeddings parameters, dropout regularization, decay rate and number of layers are the parameters that have the largest effect on the final system accuracy. Using the findings of these experiments, we trained a deep LSTM network that outperforms the state-of-the-art on the Switchboard corpus by 3.11%, and MRDA by 2.2%.

1 Introduction

Dialogue Act (DA) classification plays a key role in dialogue interpretation, especially in spontaneous conversation analysis. Dialogue acts are defined as the meaning of each utterance at the illocutionary force level (Austin, 1975). Many applications benefit from the use of automatic dialogue act classification such as dialogue systems, machine translation, Automatic Speech Recognition (ASR), topic identification, and talking avatars (Král and Cerisara, 2012). Due to the complexity of DA classification, most researchers prefer to focus on the task-oriented systems such as restaurant, hotel, or flight, etc. reservation systems.

Almost all standard approaches to classification have been applied in DA classification, from Bayesian Networks (BN) and Hidden Markov Models (HMM) to feed forward Neural Networks, Decision Trees (DT), Support Vector Machines (SVM) and rule-based approaches.

Recently, the advancement of research in deep learning has led to performance upheavals in many Natural Language Processing (NLP) tasks, even leading Manning (2016) to refer to the phenomenon as a neural network "tsunami". One of the main benefits of using deep learning approaches is that they are not as reliant on handcrafted features; instead, they manufacture features automatically from each word (Turian et al., 2010), sentence (Lee and Dernoncourt, 2016; Kim, 2014), or even long texts (Collobert et al., 2011; Mikolov et al., 2013; Pennington et al., 2014). Inspired by the performance of recent studies utilizing deep learning for improving DA classification in domain-independent conversations (Ji et al., 2016; Lee and Dernoncourt, 2016; Kalchbrenner and Blunsom, 2013), we propose a model based on a recurrent neural network, LSTM, that benefits from deep layers of networks and pre-trained word embeddings derived from Wikipedia articles.

2 Related Work

Prior work has defined general sets of DAs for domain-independent dialogues that are commonly used in almost all research on DA classification (Jurafsky et al., 1997; Dhillon et al., 2004). The task of DA classification (sometimes called DA identification) is to attribute one member of a predefined DA to each given utterance. Therefore, DA classification is sometimes treated as short-text classification. Similar to many other traditional text classification methods, five sources of information have been used for DA classification tasks: lexical information, syntax, semantics, prosody, and dialogue history. Among all

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2012–2021, Osaka, Japan, December 11-17 2016.

proposed methods, those which used more sophisticated techniques for extracting lexical information, achieved the best results before deep learning was applied to the problem.

DA classification research started with handcrafting lexical features that yielded high quality results with an accuracy of 75.22% on the 18 DAs in the VERMOBIL dataset (Jekat et al., 1995). In general, Bayesian techniques were the most common approaches for DA classification tasks, which used a mixture of n-gram models together with dialogue history for predicting DAs (Grau et al., 2004; Ivanovic, 2005). In some studies, prosody information was integrated with surface-level lexical information to improve accuracy (Stolcke et al., 2000). Stolcke et al. (2000) reported the best accuracy on the core 42 DAs in the Switchboard corpus as 71%. This result was achieved by applying contextual information with HMM for recognizing temporal patterns in lexical information. Novielli and Strapparava (2013) investigated the sentiment load of each DA. They compared the accuracies of the classification before and after analyzing utterances in the Switchboard corpus by using Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2007) and postulated that affective analysis improved the accuracy.

Recently, approaches based on deep learning methods improved many state-of-the-art techniques in NLP including, DA classification accuracy on open-domain conversations (Kalchbrenner and Blunsom, 2013; Ravuri and Stolcke, 2015; Ji et al., 2016; Lee and Dernoncourt, 2016). Kalchbrenner and Blunsom (2013) used a mixture of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). CNNs were used to extract local features from each utterance and RNNs were used to create a general view of the whole dialogue. This work improved the state-of-the-art 42-tag DA classification on Switchboard (Stolcke et al., 2000) by 2.9% to reach 73.9% accuracy. Ji et al. (2016) presented a hybrid architecture that merges an RNN language model with a discourse structure that considers relations between two contiguous utterances as a latent variable. This approach improved the result of the state-of-the-art method by about 3% (from 73.9 to 77) when applied on the Switchboard corpus. The best result was achieved when the algorithm was trained to maximize the conditional likelihood. Ji et al. (2016) also investigated the performance of using standard RNN and CNN on DA classification and got the cutting edge results on the MRDA corpus (Ang et al., 2005) using CNN.

3 Our Model

Most deep learning variations were designed and studied in the late 1990s, but their true performance was not revealed until high-speed computers were commercialized and researchers were able to access significant amounts of data. Collobert et al. (2011) used a large amount of unlabeled data to map words to high-dimensional vectors and a Neural Network architecture to generate an internal representation. By adding a CNN architecture Collobert et al. (2011) built the SENNA application that uses representation in language modeling tasks. Their approach outperforms almost all sophisticated traditional NLP applications like part-of-speech-tagging, chunking, named entity recognition, and semantic role labeling without resorting to the use of any handcrafted features or prior knowledge which are usually optimized for each task. In this study, we designed a deep neural network model that benefits from pre-trained word embeddings combined with a variation of the RNN structure for the DA classification task.

For each utterance that contains l number of words, our model convert it into l sequential word vectors. Word vectors can be generated randomly with arbitrary dimensions or being set by a pre-trained word vectors using a variety of word-to-vector techniques (Mikolov et al., 2013; Pennington et al., 2014).

3.1 RNN-based Utterance Representation

Figure 1 illustrates a typical structure of an RNN. As can be seen, information from previous layers, h_{t-1}, is contributed to the succeeding layer's computations that generate h_t. Since almost all tokens, X_i, in a conversation are related to their previous tokens or words, we choose to use an RNN structure.

Given a list of d -dimensional word vectors, $X_1, X_2, ..., X_{t-1}, X_t, ...X_{t+n}$ in a given time step, t, we will have:

$$h_t = \sigma\left(W^{hh}h_{t-1} + W^{hd}X_t\right) \tag{1}$$

$$y_t = softmax\left(W^{(S)}h_t\right) \tag{2}$$

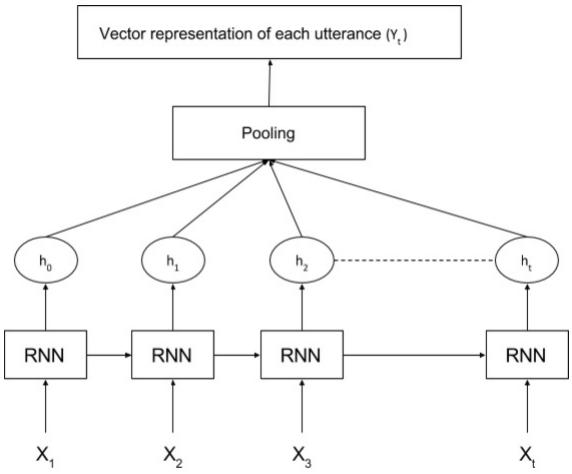


Figure 1: RNN structure for creating a vector-based representation of an utterance from its word.

where $W^{hh}\varepsilon\mathbb{R}^{\mathbb{h}\times\mathbb{h}}$ and $W^{hx}\varepsilon\mathbb{R}^{\mathbb{h}\times d}$ are weight matrices. σ represents logistic sigmoid function, and y_t, $y_t\varepsilon\mathbb{R}^{\mathbb{k}}$, is the class representation of each utterance and k denotes the number of classes for classification task.

In the pooling layer (Figure 1), our model takes all h vectors, $h_{1:t}$, and generate one vector. We can choose from three mechanisms: mean-, max- or last-pooling. Mean-pooling measures the average of all h vectors, max-pooling takes the greatest figure out of each h vector and last-pooling takes the last h vector (i.e., h_t).

Theoretically, RNNs should preserve the memory of previous incidents, but in practice when the gap between relevant information extends, RNNs fail to maintain relevant information. Hochreiter (1991) and Bengio et al. (1994) investigated the main reasons for RNNs' failures in detail. The other problem with RNN is the vanishing and exploding gradient that causes the learning process to be terminated prematurely (Mikolov et al., 2010; Pascanu et al., 2013).

Given the aforementioned problems with RNNs, we use Long Short Term Memory (LSTM), which is a variation of RNNs that is tuned to preserve long-distance dependencies as their default specificity. In DA classification, having the ability to connect related expressions of information that are distant from each other is important, particularly when it comes to classifying utterances as either subjective or objective, which is considered as one of the main sources of error in DA classification (Novielli and Strapparava, 2013). Classifying subjective versus objective texts is one of the major tasks in sentiment analysis in which LSTM-based approaches are shown to achieve high-quality results (Socher et al., 2013). Another reason for using LSTM is that it uses a *forget gate layer* to distill trivial weights, which belong to unimportant words from the cell state (see Eq. 4) . Figure 2 illustrates a standard structure of an LSTM cell.

As can be seen in Figure 2, we can define the LSTM cell at each time step *t* to be a set of vectors in $\mathbb{R}^d$:

$$i_t = \sigma\left(W^{(i)}X_t + U^{(i)}h_{t-1} + b^{(i)}\right) \tag{3}$$

$$f_t = \sigma\left(W^{(f)}X_t + U^{(f)}h_{t-1} + b^{(f)}\right) \tag{4}$$

$$o_t = \sigma\left(W^{(o)}X_t + U^{(o)}h_{t-1} + b^{(o)}\right) \tag{5}$$

$$u_t = \tanh\left(W^{(u)}X_t + U^{(u)}h_{t-1} + b^{(u)}\right) \tag{6}$$

$$c_t = i_t \bigodot u_t + f_{(t)} \bigodot c_{t-1} \tag{7}$$

$$h_t = o_t \bigodot \tanh(c_t) \tag{8}$$

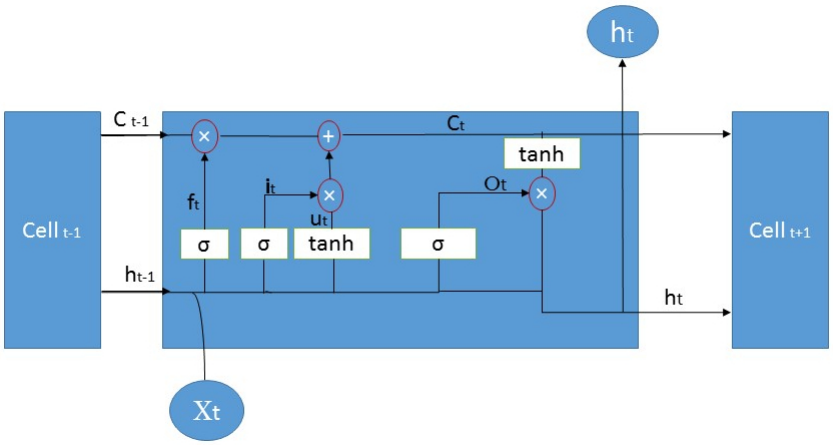


Figure 2: LSTM cell structure and its respective parameters (`http://colah.github.io`).

Where inputs are d dimensional vectors, i_t is the input gate, f_t is the forget gate, o_t is the output gate, c_t is the memory cell, h_t is the hidden state and $\odot$ represents element-wise multiplication.

c_t (Eq. 7) is the key part of LSTMs – it connects chains of cells together with linear interactions. In LSTMs, we have gates in each cell that decide dynamically which signals are allowed to pass through the whole chain. For example, the forget gate f_t (Eq. 4) decides to what extent the previous memory cell should be forgotten, the input gate (Eq. 3) manages the extent to which each cell should be updated, and the output gate manages the exposure of the internal memory state. The hidden layer h_t represents a gated, partial view of its cell state. LSTMs are able to view information over multiple time scales due to the fact that gating variables are assigned different values for each vector element (Tai et al., 2015).

3.2 Stacked LSTM

By arranging some LSTM cells back to back (Figure 2), the hidden layer, h_t, of each cell is considered as input for the subsequent layer in the same time step (Graves et al., 2013; Sutskever et al., 2014). The main reason for stacking LSTM cells is to gain longer dependencies between terms in the input chain of words.

In this study, we used stacked LSTMs with pre-trained word embeddings. Word embedding is distributional representations of words that are used to solve the data sparsity problem (Bengio et al., 2003). We trained word embeddings with 300-dimensional vectors by choosing the window and min-count equal to 5 (Mikolov et al., 2013).

4 Datasets

Since our study focuses on classifying DAs in open-domain conversations, we chose to evaluate our model on Switchboard (SwDA) (Jurafsky et al., 1997) and the five-class version of MRDA (Ang et al., 2005).

- **SwDA:** The Switchboard corpus (Godfrey et al., 1992) contains 1,155 five-minute, spontaneous, open-domain dialogues. Jurafsky et al. (1997) revised and collapsed the original DA tags into 42 DAs, which we use to evaluate our model. SwDA has 19 conversations in its test set.

- **MRDA:** The ICSI Meeting Recorder Dialogue Act corpus was annotated with the DAMSL tagset. This corpus is comprised of recorded multi-party meeting conversations. The MRDA contains 75 one-hour dialogues. There are several variations of the MRDA corpus but MRDA with 5 tags is commonly used in the literature.

  We used the list of files provided by Lee and Dernoncourt (2016) for creating the training, test, and development sets from the MRDA datasets.

5 Experimental Settings

We used the SwDA dataset to tune all hyperparameters including dropout, decay rate, word embeddings and the number of LSTM layers. All conversations in the training set were preprocessed and a randomized selection of one-third of them were utilized as a development set to allow the LSTM parameters to be trained over a reasonable number of epochs. We tuned one parameter value at a time and measured the accuracy on the development set, stopping when the accuracy on the development set did not change for 20 epochs. We used the NN packages provided by Lei et al. (2015) and Barzilay et al. (2016).

5.1 Word Embeddings

We tuned the word embedding parameters $method$, $corpus$ and $dimensionality$, while holding other parameters constant ($dropout = 0.5$, $decayrate = 0.5$ and $layersize = 2$). Specifically, we tested the methods *Word2vec* using the Gensim Word2vec package (Řehůřek and Sojka, 2010) and pretrained *Glove* word embeddings (Pennington et al., 2014). Word2vec embeddings were learned from Google News (Mikolov et al., 2013), and separately, from Wikipedia[1]. The Glove embeddings were pretrained on the 840 billion token Common Crawl corpus.

Method	**Resource**	**Dimension**	**Accuracy (%)**
Word2vec	Wikipedia	75	70.73
Word2vec	Wikipedia	150	71.85
Word2vec	Wikipedia	300	70.77
Word2vec	GoogleNews	75	71.26
Word2vec	GoogleNews	150	71.39
Word2vec	GoogleNews	300	71.32
Glove	CommonCrawl	75	69.28
Glove	CommonCrawl	150	69.71
Glove	CommonCrawl	300	69.40

Table 1: Accuracy using different word embedding techniques, corpora and vector dimensions.

Table 1 illustrates that the best results were consistently achieved by embeddings with 150-dimensions, and of those, Word2vec trained on Wikipedia had the best accuracy. Hence, these settings were used throughout the remainder of the experiments.

5.2 Decay Rate

LSTM uses standard backpropagation to adjust network connection weights (see Eq. 9), where E is the error and W_{ij} is the weight matrix between two nodes, i and j.

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}, \tag{9}$$

where η is the learning rate. To avoid overfitting, a regularization factor is added to Eq. 9 to penalize large changes in w_{ij}.

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}} - \eta \lambda w_{ij}. \tag{10}$$

The term $-\eta\lambda w_{ij}$ is the regularization factor and λ is the decay factor that causes w_{ij} decay in scale to its prior measure. We found that changing η does not impact the accuracy so we set $\eta = 1e-3$ and change λ to find the best fit for the data (Table 2).

As can be seen from Table 2, the positive trend of increasing accuracy fails after setting $\lambda = 0.8$. Therefore, we set $\lambda = 0.7$ in our experiments.

[1] https://dumps.wikimedia.org/enwiki/20160421

Accuracy (%)	λ
70.76	0.1
70.79	0.2
70.87	0.3
71.32	0.4
71.85	0.5
71.90	0.6
71.95	0.7
70.95	0.8

Table 2: The impact of changing λ on accuracy.

5.3 Dropout

Most of the recent studies that exploit deep learning approaches use the dropout technique (Hinton et al., 2012). Dropout is a kind of regularization technique that prevents the network from overfitting by discarding some weights. In each training cycle, it is possible that some neurons are co-adapted by randomly assigning zero to their weights. Dropout methods were originally introduced for feed-forward and convolutional neural networks but recently have been applied pervasively in the input embeddings layer of recurrent networks including LSTMs (Zaremba et al., 2014; Pachitariu and Sahani, 2013; Bayer et al., 2013). Bayer et al. (2013) report that standard dropout does not work effectively with RNNs due to noise magnification in the recurrent process which results in diminished learning. Since standard dropout is proven not to work effectively for RNNs, we apply the dropout technique proposed by Zaremba et al. (2014) for regularizing RNNs that is used by most studies in the literature employing LSTM models (Lei et al., 2015; Barzilay et al., 2016; Jaech et al., 2016; Swayamdipta et al., 2016; Lu et al., 2016). Zaremba et al. (2014) postulate that their approach reduces overfitting on a variety of tasks, including language modeling, speech recognition, image caption generation, and machine translation. We experimented with dropout probability settings in the range between 0.0 and 0.5.

Accuracy (%)	Dropout probability
71.95	0.5
72.01	0.4
72.05	0.3
72.15	0.2
72.55	0.1
73.29	0.0

Table 3: Impact of changing dropout on accuracy.

As can be seen in Table 3, any dropout at all hurt the accuracy. Hence, the value was set at 0.0 – dropout was not used in later tuning or in the final model.

5.4 Number of LSTM Layers

Finally, we tuned the number of layers. If you utilize only two layers, the model does not detect relevant tokens that are distant from each other. Conversely, if you use too many LSTM layers, the model will be prone to overfitting. We tested values in the range of 2 to 15. Table 4 illustrates our settings' performance on the development set – the accuracy increases up to a 10 LSTM cells before dropping significantly at 15.

5.5 Other Parameters

In addition to the aforementioned parameters, we investigated the impact of changing $L2\text{-}reg$, $pooling$, and $activation$ and finally set them to $1e-5$, $last\ pooling$, and $tanh$ respectively. These settings were

Accuracy (%)	No. of layers
73.29	2
73.61	5
73.92	10
72.90	15

Table 4: Impact of LSTM layers on accuracy.

consistent with previous findings in the literature and we did not observe significant improvements by changing these values.

6 Results and Discussion

In previous sections, we found the best setting for our model, with which we gained the best accuracy on the SwDA development set. In this section, we report our results on the SwDA and MRDA test set.

Model	Accuracy (%)
Our RNN Model	**80.1**
HMM (Stolcke et al., 2000)	71.0
CNN (Lee and Dernoncourt, 2016)	73.1
RCNN (Kalchbrenner and Blunsom, 2013)	73.9
DRLM-joint training (Ji et al., 2016)	74.0
DRLM-conditional training (Ji et al., 2016)	77.0
Tf-idf (baseline)	47.3
Inter-annotator agreement	84.0

Table 5: SwDA dialogue act tagging accuracies.

Table 5 shows the results achieved by our model in comparison with previous works. As a baseline, we consider the accuracy obtained from a Naive Bayes classifier using *tf-idf* bigrams as features (Naive Bayes outperformed other classifiers including SVM and Random Forest). Our model improved results over the state-of-the-art methods and the baseline by 3.11% and 32.85%, respectively.

We also applied our model to classify dialogue acts in the MRDA with 5 dialogue acts. To do so, we used the same settings as described above for classifying dialogue acts in SwDA (Table 5). Table 6 shows our results on the MRDA corpus.

Model	Accuracy (%)
Our RNN Model	**86.8**
CNN (Lee and Dernoncourt, 2016)	84.6
Graphical Model (Ji and Bilmes, 2006)	81.3
Naive Bayes (Lendvai and Geertzen, 2007)	82.0
Tf-idf (baseline)	74.6

Table 6: MRDA dialogue act tagging accuracies.

We calculate the baseline as before, by using *tf-idf* bigram features. The Random Forest classifier achieved the best result in comparison to other classifiers such as Naive Bayes and SVM. Our results in Table 6 show that our model outperformed the state-of-the-art method by 2.2%. It should be emphasized that our model achieved this result without being tuned on an MRDA development set.

7 Conclusion

In this study, we used a deep recurrent neural network for classifying dialogue acts. We showed that our model improved over the state-of-the-art in classifying dialogue act in open-domain conversational text.

We ran several experiments to realize the effects of setting each hyperparameter on the final results. We found that dropout regularization should be applied to LSTM-based structures (even for LSTM-adapted dropout methods that have been proven to have a positive impact on some datasets) cautiously to ensure that it does not have a negative impact on the accuracy of the system.

Acknowledgements

This research is partially supported by grant IIS-1262860 to UNT from the National Science Foundation.

References

Jeremy Ang, Yang Liu, and Elizabeth Shriberg. 2005. Automatic dialog act segmentation and classification in multiparty meetings. In *ICASSP (1)*, pages 1061–1064.

John Langshaw Austin. 1975. *How to do things with words*. Oxford university press.

Tao Lei Hrishikesh Joshi Regina Barzilay, Tommi Jaakkola, Katerina Tymoshenko, and Alessandro Moschitti Llu Marquez. 2016. Semi-supervised question retrieval with gated convolutions. *Naacl*.

Justin Bayer, Christian Osendorfer, Daniela Korhammer, Nutan Chen, Sebastian Urban, and Patrick van der Smagt. 2013. On fast dropout and its applicability to recurrent networks. *arXiv preprint arXiv:1311.0701*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *journal of machine learning research*, 3(Feb):1137–1155.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Rajdip Dhillon, Sonali Bhagat, Hannah Carvey, and Elizabeth Shriberg. 2004. Meeting recorder project: Dialog act labeling guide. Technical report, DTIC Document.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 517–520. IEEE.

Sergio Grau, Emilio Sanchis, Maria Jose Castro, and David Vilar. 2004. Dialogue act classification using a bayesian approach. In *9th Conference Speech and Computer*.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Sepp Hochreiter. 1991. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, page 91.

Edward Ivanovic. 2005. Dialogue act tagging for instant messaging chat sessions. In *Proceedings of the ACL Student Research Workshop*, pages 79–84. Association for Computational Linguistics.

Aaron Jaech, Larry Heck, and Mari Ostendorf. 2016. Domain adaptation of recurrent neural networks for natural language understanding. *arXiv preprint arXiv:1604.00117*.

Susanne Jekat, Alexandra Klein, Elisabeth Maier, Ilona Maleck, Marion Mast, and J Joachim Quantz. 1995. *Dialogue acts in VERBMOBIL*. Citeseer.

Gang Ji and Jeff Bilmes. 2006. Backoff model training using partially observed data: Application to dialog act tagging. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 280–287. Association for Computational Linguistics.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse relation language models. *arXiv preprint arXiv:1603.01913*.

Daniel Jurafsky, Rebecca Bates, Noah Coccaro, Rachel Martin, Marie Meteer, Klaus Ries, Elizabeth Shriberg, Audreas Stolcke, Paul Taylor, Van Ess-Dykema, et al. 1997. Automatic detection of discourse structure for speech recognition and understanding. In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pages 88–95. IEEE.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. *arXiv preprint arXiv:1306.3584*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Pavel Král and Christophe Cerisara. 2012. Dialogue act recognition approaches. *Computing and Informatics*, 29(2):227–250.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. *arXiv preprint arXiv:1603.03827*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. *arXiv preprint arXiv:1508.04112*.

Piroska Lendvai and Jeroen Geertzen. 2007. Token-based chunking of turn-internal dialogue act sequences. In *Proceedings of the 8th SIGDIAL Workshop on Discourse and Dialogue*, pages 174–181.

Liang Lu, Lingpeng Kong, Chris Dyer, Noah A Smith, and Steve Renals. 2016. Segmental recurrent neural networks for end-to-end speech recognition. *arXiv preprint arXiv:1603.00223*.

Christopher D Manning. 2016. Computational linguistics and deep learning. *Computational Linguistics*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

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

Nicole Novielli and Carlo Strapparava. 2013. The role of affect analysis in dialogue act identification. *Affective Computing, IEEE Transactions on*, 4(4):439–451.

Marius Pachitariu and Maneesh Sahani. 2013. Regularization and nonlinearities for neural language models: when are they needed? *arXiv preprint arXiv:1301.5650*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

James W Pennebaker, Roger J Booth, and Martha E Francis. 2007. Linguistic inquiry and word count: Liwc [computer software]. *Austin, TX: liwc. net*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Suman Ravuri and Andreas Stolcke. 2015. Recurrent neural network and lstm models for lexical utterance classification. *Proc. Interspeech, Dresden*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Greedy, joint syntactic-semantic parsing with stack lstms. *arXiv preprint arXiv:1606.08954*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Non-sentential Question Resolution using Sequence to Sequence Learning

Vineet Kumar
IBM Research Labs
New Delhi, India
vineeku6@in.ibm.com

Sachindra Joshi
IBM Research Labs
New Delhi, India
jasachind@in.ibm.com

Abstract

An interactive Question Answering (QA) system frequently encounters non-sentential (incomplete) questions. These non-sentential questions may not make sense to the system when a user asks them without the context of conversation. The system thus needs to take into account the conversation context to process the incomplete question. In this work, we present a recurrent neural network (RNN) based encoder decoder network that can generate a complete (intended) question, given an incomplete question and conversation context. RNN encoder decoder networks have been show to work well when trained on a parallel corpus with millions of sentences, however it is extremely hard to obtain conversation data of this magnitude. We therefore propose to decompose the original problem into two separate simplified problems where each problem focuses on an abstraction. Specifically, we train a semantic sequence model to learn semantic patterns, and a syntactic sequence model to learn linguistic patterns. We further combine syntactic and semantic sequence models to generate an ensemble model. Our model achieves a BLEU score of 30.15 as compared to 18.54 using a standard RNN encoder decoder model.

1 Introduction

Question Answering (QA) systems (Green Jr et al., 1961; Winograd, 1971; Woods and Kaplan, 1977; Hickl et al., 2006; Gobeill et al., 2009) enable a user to obtain precise information. A natural extension is an interactive and dialogue based QA system that allows a user to ask follow up or related questions. Interactive QA system however comes with its unique set of challenges. Users ask a follow up or related question by being as terse as possible, and they implicitly refer to concepts and entities in the past conversation. Table 1 depicts a few instances of follow up questions users may ask in an ongoing conversation.

Incomplete questions are a subset of non-sentential utterances (NSU) (Fernández, 2006). NSUs are incomplete utterances which make complete sense when seen in conjunction with the utterances in conversation. Table 1 illustrates some examples of NSU questions ($Q2$) a user might ask the system given a previous question ($Q1$) and an answer ($A1$). $R1$ refers to the intended complete question. Note that (a) and (c) need the previous question $Q1$, (b) needs previous answer $A1$, whereas (d) needs both $Q1$ and $A1$ to generate $R1$. The system thus either needs to restrict how users interact (Carbonell, 1983), or needs to handle the NSU questions by considering the conversation context. Restricting how users interact with a QA system is not natural, and thus can make the system hard to use. In this work, we focus on using the incomplete question and the conversation context to generate the resolved (intended) question. In the rest of the paper, we refer to this problem as NSU question resolution.

NSU resolution is an active area of research. One set of work deals with classifying NSU (Fernández et al., 2005). Another set of work proposes a rule or grammar based approach to resolve NSU (Carbonell, 1983; Dalrymple et al., 1991). Recently, a statistical based approach has been proposed for resolving NSU question (Raghu et al., 2015). However, this approach only focuses on the simpler problem of

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2022–2031, Osaka, Japan, December 11-17 2016.

(a)

Q1	how old was john rolfe when he died ?
A1	37
Q2	and how did he die ?
R1	how did john rolfe die ?

(b)

Q1	what animal has a 7 lettered name ?
A1	cheetah
Q2	and how fast can it run ?
R1	how fast can a cheetah run ?

(c)

Q1	what is greece 's national sport ?
A1	football
Q2	flower ?
R1	what is greece 's national flower ?

(d)

Q1	what do road runners eat ?
A1	small reptiles
Q2	how often ?
R1	how often do road runners eat small reptiles ?

Table 1: Examples of non-sentential questions in conversations:
(a) and (c) need the previous question $Q1$
(b) needs previous answer $A1$; (d) needs both $Q1$ and $A1$ to be resolved

resolving NSU based on previous questions, and thus will not be able to handle examples given in Table 1(b) and 1(d), where previous answer or a combination of previous question and answer is needed.

Recently, recurrent neural network (RNN) based encoder decoder networks have been applied successfully to the task of statistical machine translation (Cho et al., 2014; Bahdanau et al., 2014; Sutskever et al., 2014). RNN encoder decoder, also known as sequence to sequence learning, maps a variable length input sequence to a variable length output sequence. In this work, we approach the problem of NSU question resolution as sequence to sequence learning. We generate the input sequence by concatenating NSU question, previous question and answer. RNN encoder decoder is then used to learn a mapping of this input sequence to the resolved question.

RNN encoder decoder models have been successfully trained on huge parallel corpus of millions of sentences (Bahdanau et al., 2014; Cho et al., 2014; Sutskever et al., 2014). However, it is extremely hard to obtain conversation data of this magnitude. We have access to only 7220 conversations containing NSU questions, which were collected using Amazon Mechanical Turk (Raghu et al., 2015).

As we have a small dataset, we propose to decompose the original problem into two separate simplified problems where each problem uses an abstraction. These abstractions help the model training to focus on learning a specific aspect of the problem. Specifically, we train a syntactic sequence model to learn linguistic patterns, and a semantic sequence model to learn semantic patterns. We combine these two different models to generate an ensemble model, which can capture both linguistic and semantic patterns in NSU question conversations.

Our main contributions in this work are as follows:

1. We present a novel approach to handle non-sentential questions using the framework of sequence to sequence learning. Our approach is completely data driven, and can generate complete questions from a non-sentential question, given previous question and answer.

2. We propose a method to decompose the original NSU question resolution problem into two separate simplified abstractions that focus on learning a specific aspect of the problem. One such abstraction is semantic patterns in conversation data, that we learn with the help of a semantic sequence model.

3. We present a syntactic sequence model that focuses solely on learning linguistic patterns in conversations. Finally, we combine the semantic and syntactic sequence models to generate an ensemble model. Our ensemble model achieves a BLEU score of 30.15 as compared to 18.54 using a standard RNN encoder decoder.

Rest of this paper is organized as follows. We discuss related work in Section 2. Background needed to understand RNN encoder decoder model is discussed in Section 3. We present syntactic and semantic sequence models in Section 4 and Section 5 respectively. Finally, we discuss experiment settings and results in Section 6 and conclude in Section 7.

2 Related Work

NSUs were studied and classified into various classes by Ferna'ndez and Ginzburg (2002). One thread of work has focused on identifying and classifying NSUs into classes (Fernández et al., 2005; Rovira, 2006). Another thread of work has focused on resolving NSUs into complete intended utterances by building domain specific rules or grammar (Dalrymple et al., 1991; Carbonell, 1983). Writing rules or grammar is hard, extremely time consuming and may suffer with low recall. Therefore, we focus on a data driven and statistical approach.

Raghu et al. (2015) is the only work we know of that uses a statistical and data-driven model to resolve NSU questions. However their model cannot handle cases where previous answer or a combination of previous question and answer is needed to resolve a NSU question. For example, their approach cannot handle examples given in Table 1(b) and 1(d). Our approach does not have any such restrictions.

Sequence to sequence learning (Sutskever et al., 2014; Bahdanau et al., 2014; Cho et al., 2014) has been applied to a myriad applications. Some of the successful applications include statistical machine translation, speech translation (Duong et al., 2016), translating videos to sentences (Venugopalan et al., 2015), image captioning (Karpathy and Fei-Fei, 2015; Jia et al., 2015). Sequence to sequence learning has also been applied in modeling conversations (Li et al., 2016; Serban et al., 2016).

To the best of our knowledge, ours is the first work that approaches NSU question resolution as a sequence to sequence learning problem. Ours is also the first work that decomposes the original sequence to sequence learning problem, into separate simplified problems where each problem focuses on an abstraction.

3 Sequence to Sequence Learning

In this section, we discuss the framework of RNN encoder decoder model. This is followed by discussion on why it can be hard to train a RNN encoder decoder model using a small dataset. We finally formulate NSU question resolution as a sequence to sequence learning problem.

3.1 Background and Model Size

Sequence to sequence learning framework uses a recurrent neural network (RNN) to encode a variable-length input sequence to a fixed length vector, and then uses another RNN to decode the vector into a variable-length target sequence (Cho et al., 2014).

The model takes a source sentence (x) as input. Each sentence is a sequence of words, and each word is encoded using a one-hot encoding:

$$x = (x_1, x_2, \cdots, x_{t_x}), x_i \in \Re^{|V|}$$

The model outputs a target sentence (y), which is a sequence of words:

$$y = (y_1, y_2, \cdots, y_{t_y}), y_i \in \Re^{|V|}$$

where t_x and t_y respectively denote length of sequence x and y, and $|V|$ denotes the vocabulary size, and t_x need not be same as t_y. Note that compared to a neural machine translation model, we do not need a separate vocabulary for source and target, as source and target are in the same language (English).

RNN encoder first computes its forward state which are fixed length vectors $\vec{h}_i$:

$$\vec{h}_i = \begin{cases} (1 - \vec{z}_i) \circ \vec{h}_{i-1} + \vec{z}_i \circ \underline{\vec{h}_i} & \text{if i} > 0 \\ 0 & \text{if } i = 0 \end{cases}$$

where

$$\underline{\vec{h}_i} = tanh(\vec{W}\overline{E}x_i + \vec{U}[\vec{r}_i \circ \vec{h}_{i-1}])$$
$$\vec{z}_i = \sigma(\vec{W}_z\overline{E}x_+\vec{U}_z\vec{h}_{i-1})$$
$$\vec{r}_i = \sigma(\vec{W}_r\overline{E}x_+\vec{U}_r\vec{h}_{i-1})$$

$\overline{E} \in \Re^{m \times |V|}$ is the word embedding matrix. $\vec{W}, \vec{W}_z, \vec{W}_r \in \Re^{n \times m}$, $\vec{U}, \vec{U}_z, \vec{U}_r \in \Re^{n \times n}$ are weight matrices. m and n are word embedding dimensionality and number of hidden units respectively. σ is the logistic sigmoid function, $\circ$ is element wise multiplication.

RNN decoder is then initialized by a context vector $\vec{c}$. Typically a context vector is some combination of RNN Encoder's forward state vectors $\vec{h}_i$. Cho et al. (2014) and Sutskever et al. (2014) assign the context vector as $\vec{h}_{t_x}$, whereas Bahdanau et al. (2014) assign the context vector as a combination of RNN encoder hidden states ($\vec{h}_1, \vec{h}_2 \cdots \vec{h}_{t_x}$). The context vector c is then used to output sequence words.

Table 2 shows model size used by various RNN encoder decoder implementations. As $n \ll |V|$ and $m \ll |V|$, we can see that $\overline{E}$ dominates over other parameters $\vec{W}, \vec{W}_z, \vec{W}_r, \vec{U}, \vec{U}_z, \vec{U}_r$. This is usually not a problem when training data is large (of order of million sentences). However, for a small dataset, training a model with so many parameters does not work. We observed the same in our experiments.

We can reduce the vocabulary size, by replacing words that occur below a minimum frequency threshold with a special unknown symbol (UNK). This however, discards lots of useful information.

We present two new models: syntactic sequence (Section 4) and semantic sequence (Section 5) which can preserve and learn linguistic and semantic patterns respectively, while keeping the vocabulary size small.

Model	**Training data**	V	n	m
(Cho et al., 2014)	12M	15,000	1000	620
(Bahdanau et al., 2014)	12M	30,000	1000	620
(Sutskever et al., 2014)	12M	160,000	1000	620

Table 2: Model size for RNN Decoder. n is hidden layer size, m is word embedding size

3.2 Modeling NSU question resolution as sequence to sequence learning

We cast the problem of NSU question resolution as sequence to sequence learning. We concatenate the non-sentential question ($Q2$) and context ($Q1$, $A1$) to generate the source sequence. We use a special end of utterance symbol (END) to create the input sequence. This source sequence is then used to generate the resolved question ($R1$). For example, Table 3 depicts parallel corpus transformation for Table 1(c).

Source	what is greece 's national sport ? END football END flower ?
Target	what is greece 's national flower ?

Table 3: Parallel corpus formulation of Table 1(c)

Figure 1 depicts how RNN encoder decoder works. RNN encoder first processes the entire input sequence ($Q1$, $A1$, $Q2$) to a single fixed dimension vector (context vector $\vec{c}$). This vector is then used to initialize the RNN decoder. RNN decoder then samples output sequence by conditioning on previous sampled word, and the context vector.

4 Syntactic Sequence Model

We discussed in Section 3.1 that the parameters of RNN encoder decoder model are dominated by the size of vocabulary $|V|$. Even for a small dataset (1000s of sentences), $|V|$ may be of the order of 10,000. Thus for a small dataset, a RNN encoder decoder model has too many parameters to train it well.

We can reduce the vocabulary size by replacing out of vocabulary (OOV) with a special unknown symbol (UNK). However, we lose information by restricting vocabulary in this manner. In some cases, we just end up training the model to reproduce previous question $Q1$. For example, Table 1(c) is transformed such that $R1$ is exactly identical to $Q1$. Important information is lost that last OOV word (sport) in $Q1$ should be replaced by the OOV (flower) in $Q2$ to generate the complete question $R1$.

```
Q1: what is UNK 's national UNK ?
A1: UNK
Q2: UNK ?
R1: what is UNK 's national UNK ?
```

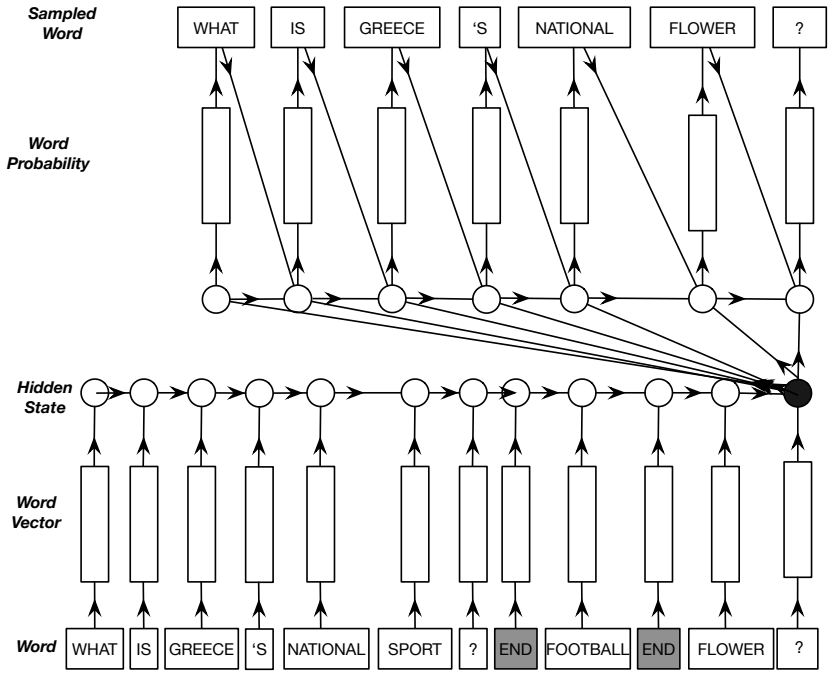


Figure 1: RNN based Encoder decoder for NSU question resolution

We can preserve linguistic structure by assigning a unique symbol to each OOV word. However, this does not help in reducing the vocabulary size. We can thus restrict assigning a new symbol only within a conversation($Q1, Q2, R1$) and reuse the symbols across conversations. Hence, it makes sense to assign symbols based on number of unknowns and its position in a single conversation. Table 4 (a) depicts how new symbols are assigned for the conversation in Table 1(c). Table 4(b) similarly shows how new symbols are assigned for the conversation in Table 1(b). Note how symbols(UNK1, UNK2, UNK3, UNK4) are shared across these two conversations. This allows the model to preserve (and learn) linguistic structure across different conversations.

(a)

Q1	what is UNK1 's national UNK2 ?	greece	UNK1
A1	UNK3	sport	UNK2
Q2	UNK4 ?	football	UNK3
R1	what is UNK1 's national UNK4 ?	flower	UNK4

(b)

Q1	what UNK1 has a UNK2 UNK3 name ?	animal	UNK1
A1	UNK4	7	UNK2
Q2	and how fast can it run ?	lettered	UNK3
R1	how fast can a UNK4 run ?	cheetah	UNK4

Table 4: Syntactic sequence training data for Table 1(c) and Table 1(b). Note how new symbols are assigned for each conversation, but shared across conversations.

Syntactic sequence model uses NSU question ($Q2$), conversation context ($Q1, A1$) and a symbol map, to generate the resolved question ($R1$). This symbol map helps in two important ways: it helps preserve the linguistic structure, and at the time of prediction it helps replace unknown symbol with the original word. We can also compare syntactic sequence model to a standard RNN encoder decoder model, where vocabulary is restricted and all OOV words are replaced with a single UNK. A standard RNN encoder decoder model will end up having UNK symbols as output. However, it is not possible to determine which word does this symbol correspond to, as there will be typically many UNK words in an input sequence. Syntactic sequence model addresses this problem by having a symbol map for the current conversation.

Syntactic sequence model however focuses solely on the position of OOV word in the sequence to assign a new unknown symbol and completely discards similarity between OOV words. In the next section (Section 5), we introduce a semantic sequence model that directly addresses this issue. Semantic sequence model focuses on learning semantic patterns from conversations.

5 Semantic Sequence Model

Syntactic sequence model focuses on learning linguistic patterns in conversations. However, it completely ignores the similarity of OOV words. This can lead to two different input sequences($Q1, A1, Q2$) to appear completely identical, even when they resolve to a different question $R1$. For example, Table 5 depicts two different input sequences with different $R1$, that look completely identical after assigning new unknown symbols based solely on position. Syntactic sequence model discards the information that OOV words 'Greece' and 'India' are similar, and 'flower' and 'sport' are similar (as compared to other tokens).

(a)

Q1	What is Greece 's national sport ?
A1	football
Q2	flower ?
R1	What is Greece 's national flower ?

Q1	What is UNK1 's national UNK2 ?
A1	UNK3
Q2	UNK4 ?
R1	What is UNK1 's national UNK4 ?

Greece	UNK1
sport	UNK2
football	UNK3
flower	UNK4

(b)

Q1	What is Greece 's national sport ?
A1	football
Q2	India ?
R1	What is India 's national sport ?

Q1	What is UNK1 's national UNK2 ?
A1	UNK3
Q2	UNK4 ?
R1	What is UNK4 's national UNK2 ?

Greece	UNK1
sport	UNK2
football	UNK3
India	UNK4

Table 5: Syntactic sequence input and output for two conversations with same $Q1$ but different $Q2$. Note how this model ends up assigning the same input sequence to (a) and (b)

We thus need a model that can exploit the similarity between OOV words and learn a higher level of abstraction. We can assign each OOV word a category number, based on a pre-learnt word category assignment. Each OOV word can then be assigned a new symbol based upon its word category index.

Semantic sequence model assigns a new symbol to each OOV word, based on the word category index. We learn the word category assignments by using a k-means algorithm (MacQueen and others, 1967), where pre-trained word vectors (Mikolov et al., 2013) are used as features.

Assigning new unknown symbols based on word similarity helps the model to focus on a powerful abstraction. The model learns that if a word of a particular category appears in a conversation, output will have words of a specific category. For example, Table 6 demonstrates how the model is trained to retain same output structure even with different NSU question ($Q2$). This is helpful as model can correctly be trained to preserve output structure at the level of word category, even with variations in input sequence.

Semantic sequence model takes as input a NSU question $Q2$, conversation context ($Q1, A1$) and a cluster symbol map. As compared to syntactic sequence model, we can have multiple OOV words assigned to the same cluster symbol token. We can replace the cluster symbol token (such as CL3) by replacing it with $Q2$ OOV word that was assigned to this cluster. We replace it with $Q2$ OOV word, as there is a greater chance that words in $Q2$ will appear in resolved utterance. For example, in Table 6(a), we can replace CL3 with flower.

(a)

Q1	What is CL1 's national CL3 ?
A1	CL3
Q2	CL3 ?
R1	What is CL1 's national CL3 ?

Greece	CL1
sport, football, flower	CL3

(b)

Q1	What is CL1 's national CL3 ?
A1	CL3
Q2	CL1 ?
R1	What is CL1 's national CL3 ?

Greece, India	CL1
sport, football	CL3

Table 6: Semantic sequence input and output for Table 5

6 Experiments and Results

6.1 Dataset

We evaluate our models on NSU question conversation data which was collected using Amazon Mechanical Turk (Raghu et al., 2015). NSU question conversation data has 7220 conversations. Each conversation consists of a previous question ($Q1$), previous answer ($A1$), NSU question ($Q2$), and a

resolved question ($R1$). Table 1 highlights a few examples from the dataset. This dataset however has many spelling mistakes, that were fixed manually using a spell checker. 6820 conversations were used for training and the remaining 400 were used as a validation set. We further lower case and then tokenize the text. RNN encoder decoder needs a parallel corpus of an input and output sequence for training. Input sequence is generated by concatenating question ($Q1$), answer ($A1$) and NSU question ($Q2$) with a special end of utterance symbol (END). We use resolved question ($R1$) as the output sequence. Table 3 lists a sample input and output sequence. There are a total of 12,603 word types, 134K words in input sequence text and 65K words in output sequence text.

6.2 Training and Model details

For all experiments, Bidirectional RNN encoder decoder with attention mechanism (Bahdanau et al., 2014) is used. Gated Recurrent unit (GRU) (Cho et al., 2014) is used as the hidden unit for RNN. We used Adam (Kingma and Ba, 2014) as the optimization algorithm with a learning rate of 0.005 and mini-batch size of 128. Although GRU does not suffer from the vanishing gradient problem, it can still suffer from exploding gradient (Graves, 2013; Pascanu et al., 2013). Thus, a hard constraint on norm of the gradient was enforced by scaling it when norm exceeds a threshold.

Word embedding matrix was initialized using pre-trained word vectors (Mikolov et al., 2013). We use an open source Theano (Theano Development Team, 2016) based implementation[1] for training all our models. Word embedding size m, hidden unit size n and regularization parameters are treated as hyper-parameters. We train with different configurations based on a combination of these hyper-parameters and select the model that gives the best BLEU score on held out set of 400 conversations.

6.3 Evaluation Metric

One possible method to evaluate our models is to manually compare the generated output sequence to gold standard (collected from a held out set). However, this method is slow, human intensive and prone to errors. We wanted a method that could automatically assign a score to the generated output sequence based on how similar it is to the gold standard. BLEU (Papineni et al., 2002) is a popular metric for evaluating statistical machine translation systems and fits our needs well. A corpus level BLEU score (based on average of four grams) on a held out dataset of 400 was computed to evaluate all our models. We use the standard evaluation script [2] used by machine translation community.

Experiment	V	**BLEU4**
All-Vocab	12,603	8.24
Freq-10	1519	17.76
Freq-20	808	**18.54**
semantic-seq-20	818	21.20
syntactic-seq-20	823	29.11
ensemble-20	823	**30.15**

Table 7: BLEU score on a held out set of 400. V refers to vocabulary size

6.4 Experiments

Section 3.1 highlighted that RNN encoder decoder model parameters are dominated by the size of vocabulary $|V|$, which can make the model difficult to train on a small dataset. To evaluate the effect of a large vocabulary on a small dataset, standard RNN encoder decoder is trained. We obtain low BLEU score for this model which has 12,603 words in vocabulary (All-Vocab). For further experiments, size of vocabulary is reduced by selecting only words that occur above a minimum threshold. All the remaining out of vocabulary words (OOV) are marked as UNK. We found that restricting vocabulary further leads to a drop in BLEU score as we generate many UNK words in the output sequence. Thus, we consider this standard RNN encoder decoder model with reduced vocabulary of 808 words as our baseline model for comparison with semantic and syntactic sequence models.

[1] https://github.com/nyu-dl/dl4mt-tutorial/tree/master/session2

[2] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

To train semantic sequence model, first all words (12,603) in original vocabulary are assigned clusters using k-means algorithm. Pre-trained word vectors (Mikolov et al., 2013) are used as word features. We use default parameters of scikit-learn (Pedregosa et al., 2011) with $k = 8$ clusters to assign the word clusters. We experimented with different cluster size, and found $k = 8$ to give the best results. Words with no word vectors are assigned a new UNK cluster, and words that are numbers are assigned a new NUM cluster. We thus have a total of 10 word clusters. Semantic sequence model shows improvement over the baseline.

Syntactic sequence model is trained by replacing OOV words in input sequence with unique UNK symbols based on its position, as described in Section 4. Maximum length of sequence symbol map for a conversation was found to be 15. Syntactic sequence model achieves significant gain in BLEU score over the baseline. We summarize all the results in Table 7.

We finally combine the best semantic and syntactic model to create an ensemble model. Ensemble model picks the output sequence which has maximum keywords overlap with NSU question $Q2$. The intuition behind this criteria is that keywords that appear in Q2 are likely to occur in the resolved question, and therefor a higher overlap of a candidate resolution with Q2 is likely to lead to a better resolution. Table 8 highlights model output for the same input sequence as generated by the best syntactic sequence and semantic sequence model. Ensemble model picks up the better among the two candidate resolutions.

Q1	A1	Q2	Gold	Syntactic	Semantic
who is the founder of usa today ?	al neuharth	and the new york times ?	who is the founder of the new york times ?	who is the founder of brazil ?	**who is the founder for the new york times ?**
where do zorse live ?	africa	and hulu ?	where do hulu live ?	what do hulu live ?	**where do hulu live ?**
who is the richest sport personality in south africa ?	ernie els	and in india ?	who is the richest sport personality in india ?	who is the richest sport in south africa ?	**who is the richest sport in india ?**
how many pounds in 125 kilograms ?	275.57375	and how many ounces ?	how many ounces are in 125 kilograms ?	how many pounds in UNK ounces ?	**how many ounces in kilograms ?**
what is the eye color of coyotes ?	yellow-brown	and that of wolves ?	what is the eye color of wolves ?	**what is the eye color of wolves ?**	what is the wolves color of ?
what does socio means ?	sociological	and what echo ?	what does echo means ?	**what does echo means ?**	what does socio start with ?
how many sides does a octagon have ?	eight	and a pentagon ?	how many sides does a pentagon have ?	**how many sides does a pentagon have ?**	how many times does a sides have ?
what is another word for portrait ?	represent	for ignorant ?	what is another word word for ignorant ?	**what is another word for ignorant ?**	what is another word for?
what is the posterior part of the brain called ?	cerebellum	and the anterior ?	what is the anterior part of the brain called ?	**what is the anterior part of the brain called ?**	what is the definition of the brain called ?
what sport originated from africa ?	wrestling	and in the united states ?	what sport originated in the united states ?	**what sport originated in the united states ?**	what sport is in a united states ?

Table 8: Model output for syntactic and semantic sequence models. Ensemble model picks the ones highlighted in **bold**

7 Conclusion

In this work we approach non-sentential question resolution in conversations as a sequence to sequence learning problem. Sequence to sequence learning models have been shown to work well when trained on a parallel corpus with millions of sentences. However, dataset of this magnitude is extremely hard to get for NSU question conversations.

We thus propose to decompose the original problem of NSU question resolution into two separate simplified problems. Each of these simpler problems focuses on an abstraction. Specifically we train a semantic sequence model that learns semantic patterns in conversations, and a syntactic sequence model that learns linguistic patterns in conversations. We finally combine the syntactic and semantic sequence model to generate an ensemble model. Our ensemble model achieves a BLEU score of 30.15 when compared to 18.54 on a standard RNN encoder decoder model with same vocabulary size.

As future work we wish to explore learning much simpler abstractions such as entity and concepts. Ensemble model is created using simple rules that pick the output sequence which has maximum overlap with NSU question. One can learn a statistical ensemble model too that uses other richer features from the simpler abstract models.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Jaime G Carbonell. 1983. Discourse pragmatics and ellipsis resolution in task-oriented natural language interfaces. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pages 164–168. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, aglar Gulehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Mary Dalrymple, Stuart M. Shieber, and Fernando Pereira. 1991. Ellipsis and higher-order unification. *CoRR*, cmp-lg/9503008.

Long Duong, David Chiang, Steven Bird, and Trevor Cohn. 2016. An attentional model for speech translation without transcription.

Raquel Fernández and Jonathan Ginzburg. 2002. Non-sentential utterances in dialogue: A: Corpus-based study.

Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. 2005. Using machine learning for non-sentential utterance classification. In *6th SIGdial Workshop on Discourse and Dialogue*.

Raquel Fernández. 2006. Non-sentential utterances in dialogue: Classification, resolution and use. *Unpublished Ph. D. thesis, University of London*.

Julien Gobeill, E Patsche, D Theodoro, A-L Veuthey, C Lovis, and P Ruch. 2009. Question answering for biology and medicine. In *2009 9th International Conference on Information Technology and Applications in Biomedicine*, pages 1–5. IEEE.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. 1961. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224. ACM.

Andrew Hickl, Patrick Wang, John Lehmann, and Sanda M. Harabagiu. 2006. Ferret: Interactive question-answering for real-world environments. In *ACL*.

Xu Jia, Efstratios Gavves, Basura Fernando, and Tinne Tuytelaars. 2015. Guiding the long-short term memory model for image caption generation. In *ICCV*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B. Dolan. 2016. A persona-based neural conversation model. *CoRR*, abs/1603.06155.

James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Dinesh Raghu, Sathish Indurthi, Jitendra Ajmera, and Sachindra Joshi. 2015. A statistical approach for non-sentential utterance resolution for interactive qa system. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 335.

Raquel Fernández Rovira. 2006. *Non-sentential utterances in dialogue: Classification, resolution and use*. University of London.

Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond J. Mooney, and Kate Saenko. 2015. Translating videos to natural language using deep recurrent neural networks. In *NAACL*.

Terry Winograd. 1971. Procedures as a representation for data in a computer program for understanding natural language. Technical report, DTIC Document.

William A Woods and R Kaplan. 1977. Lunar rocks in natural english: Explorations in natural language question answering. *Linguistic structures processing*, 5:521–569.

Context-aware Natural Language Generation for Spoken Dialogue Systems

Hao Zhou, Minlie Huang, Xiaoyan Zhu
State Key Laboratory of Intelligent Technology and Systems,
National Laboratory for Information Science and Technology,
Dept. of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China
tuxchow@gmail.com, aihuang@tsinghua.edu.cn, zxy-dcs@tsinghua.edu.cn

Abstract

Natural language generation (NLG) is an important component of question answering(QA) systems which has a significant impact on system quality. Most tranditional QA systems based on templates or rules tend to generate rigid and stylised responses without the natural variation of human language. Furthermore, such methods need an amount of work to generate the templates or rules. To address this problem, we propose a Context-Aware LSTM model for NLG. The model is completely driven by data without manual designed templates or rules. In addition, the context information, including the question to be answered, semantic values to be addressed in the response, and the dialogue act type during interaction, are well approached in the neural network model, which enables the model to produce variant and informative responses. The quantitative evaluation and human evaluation show that CA-LSTM obtains state-of-the-art performance.

1 Introduction

Natural language generation (NLG), the task of generating natural language from a knowledge base or a logical form representation, is an important component of dialogue or question answering system. NLG can be treated as a single-turn dialogue generation. Traditional approaches to NLG problem are mostly rule-based or template-based (Bateman and Henschel, 1999; Busemann and Horacek, 2002). However, these methods tend to generate rigid and stylised language without the natural variation of human language. In addition, they need a heavy workload to design the templates or rules.

Recently due to the growth of artificial neural networks and the increase of labeled data available on the Internet, data-driven approaches are developed to attack the NLG problem (Ritter et al., 2011; Shang et al., 2015). Shang et al. (2015) and Serban et al. (2015) apply the RNN-based general encoder-decoder framework to the open-domain dialogue response generation task. Although their model can generate the relevant and variant responses according to the input text in a statistical manner, the quality and content of responses depend on the quality and quantum of the training corpus. Wen et al. (2015) propose a task-oriented NLG model that can generate the responses providing the correct answers given the dialogue act (for instance, confirm or request some information), including the answer information. However the context information, such as the input question and dialogue act, is ignored. Yin et al. (2016) propose a neural network model that can generate answers to simple factoid questions based on a knowledge base. But a large error rate is observed due to the complex architecture introduced.

In this paper, we deal with the NLG problem in this setting: given a question, the corresponding dialogue act, and the semantic slots to be addressed in the response, how to generate a natural language response in a dialogue. We present a statistical task-oriented NLG model based on a Context-Aware Long Short-term Memory network (CA-LSTM), which adopts the general encoder-decoder framework to incorporate the question information, semantic slot values, and dialogue act type to generate correct answers. The major departures from prior work lie in:

- We design a context-aware generation framework for NLG. The framework incorporates the embedding of question and dialogue act type, and semantic values to be satisfied in the generated

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2032–2041, Osaka, Japan, December 11-17 2016.

response.

- We propose an attention mechanism that attends the key information in question conditioned on the current decoding state of the decoder. Thus the question embedding is dynamically changed over the decoding states.

- We propose to encode dialogue act type embedding to enable the model to generate variant answers in response to different act types. In this way, the response quality is substantially improved for particular act types.

2 Related Work

Traditional NLG methods divide the task into three phases: text planning, sentence planning, and surface realization (Walker et al., 2002), which can be solved by rules or templates in traditional approach (Mirkovic and Cavedon, 2011). The quality of language generated by rule-based or template-based methods mostly depends on the handcrafted rules or templates. Even if they can ensure the adequacy, fluency and readability of the language, the need of writing and maintaining the rules or templates is a large burden to these methods. Furthermore, the rule-based or template-based methods tend to generate rigid and nonvariant responses. To address the problem, Oh and Rudnicky(2000) propose a statistical approach which can learn from data to generate variant language using a class-based n-gram language model. However, due to the limits of the model and the lack of semantically-annotated corpora, the statistical approach cannot ensure the adequacy and readability of the generated language. Thus the statistical approach has not yet been employed widely compared to the rule-based or template-based methods.

Recently due to the maturity of artificial neural networks and the rich annotated data available on the Internet, data-driven statistical approaches are developed to address the NLG problem. These methods can be divided into two categories according to the corpus type, one is open-domain chat-based NLG, the other is task-oriented NLG for solving specific tasks.

2.1 Open-domain Chat-based NLG

Chat-based NLG aims to generate relevant and coherent responses in either single-turn or multi-turn dialogues. Shang et al.(2015) propose a Neural Responding Machine (NRM), a neural network-based chatbot NLG for Short-Text Conversation, which is trained on a large amount of one-round conversation data collected from a microblogging service. NRM takes the general encoder-decoder framework: NRM encodes the input text to the latent representation and then decodes the encoded information to generate responses. Rather than the traditional NLG task which includes text planning, sentence planning, and surface realization phases, NRM formalizes the NLG task as a general decoding process based on the encoded latent representation of the input text. Serban et al. (2015) propose a hierarchical recurrent encoder-decoder neural network to the open domain dialogue. In addition to the input text, the hierarchical model encodes the context information to generate the response.

Data-driven statistical approaches have also been studied for the text planning phase. In the text planning phase, NLG chooses the proper information of every sentence to be presented to users. The generation models mentioned above are trained by predicting the system response in a given conversational context using the maximum-likelihood estimation (MLE) objective so that they tend to generate nonsense responses such as "I dont know". To address this problem, Li et al. (2016) applies deep reinforcement learning to model long-term reward in chatbot dialogue which can plan the information in the response and avoid generating the nonsense responses in the dialogue.

2.2 Task-oriented NLG for Specific Domain

The statistical methods mentioned above are designed for open-domain chatbots, which emphasize on generating relevant and fluent responses according to the input text. While these methods are not suitable for task-solving scenarios (for instance, dialogue systems for restaurant and hotel reservation), which aims at providing correct answers to the input questions, because the responses of these methods are generated from the training data, which can not contain correct answers to any questions. Wen et al.

(2015) propose a statistical NLG based on a semantically controlled Long Short-term Memory (SC-LSTM) recurrent network which partially solves this problem. In the sentence planning phase, SC-LSTM generates the answers containing the slot tokens according to the dialogue act and slot-value pairs; in the surface realization phase, SC-LSTM replaces the slot tokens with the correct answers to the input questions. However, SC-LSTM generates responses with given answers information (the dialogue act and slot-value pairs) regardless of the input questions and generated answers. A Neural Generative Question Answering (GENQA), which can generate answers to simple factoid questions based on a knowledge base, addresses the task-oriented NLG problem further (Yin et al., 2016). However after unifying understanding of question, generation of answer, and retrieval of relevant facts in a knowledge-base into an end-to-end framework, GENQA introduces more errors in answers.

3 The Context-Aware LSTM Model (CA-LSTM)

3.1 Overview

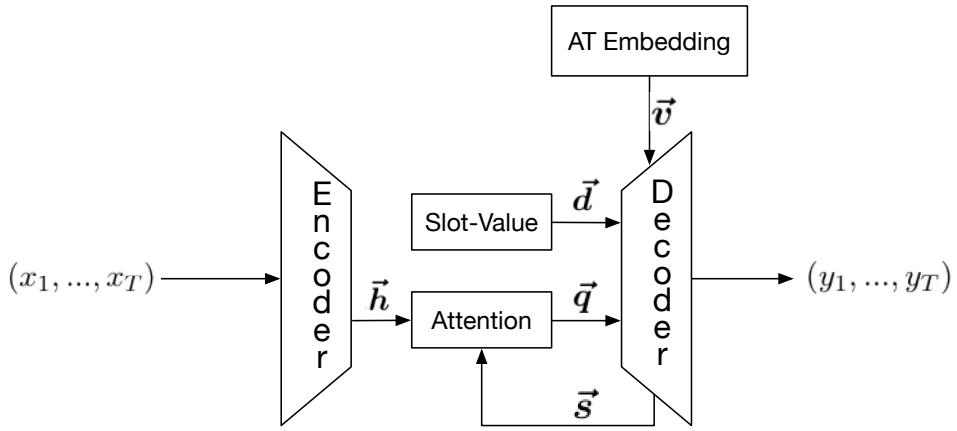


Figure 1: The general framework and dataflow of CA-LSTM. $\vec{h}$ is the hidden representation of the input question $(x_1, ..., x_T)$, $\vec{q}$ is the question vector generated by attention on $\vec{h}$ and the state of decoder $\vec{s}$, $(y_1, ..., y_T)$ is the output answer decoded with the question vector $\vec{q}$, the slot-value vector $\vec{d}$ and the dialogue act type (AT) embedding $\vec{v}$.

The Context-Aware LSTM (CA-LSTM) is built on the general encoder-decoder framework for sequence-to-sequence learning (Sutskever et al., 2014), as shown in Figure 1. Let $\boldsymbol{X} = (x_1, ..., x_T)$ and $\boldsymbol{Y} = (y_1, ..., y_T)$ denote the input question and output response respectively. The encoder converts the question sequence $\boldsymbol{X}$ to the hidden representation of the question sequence $\boldsymbol{h} = (h_1, ..., h_T)$. Then the hidden vector $\boldsymbol{h}$ is converted to the high dimensional question vector $\boldsymbol{q}$ after being attended with the current state of decoder. Finally, with the input of the question vector $\boldsymbol{q}$, the slot-value vector $\boldsymbol{d}$ and the embedding vector $\boldsymbol{v}$ of dialogue act type, the decoder generates the answer sequence $\boldsymbol{Y}$. Specifically, the slot-value vector $\boldsymbol{d}$[1] is a one-hot representation of the slot-value pairs which can regularize the generation process and ensure the information adequacy of answers as suggested by (Wen et al., 2015). By providing to the decoder the context information, including question vector and the act type embedding vector, CA-LSTM can generate context-aware responses which are related to the input question and dialogue acts.

The encoder and decoder are both based on Long Short-term Memory for its ability of modeling sequences of arbitrary lengths (Hochreiter and Schmidhuber, 1997). The procedure of generating variant responses has two components: the forward generator and the backward reranker, which share the same CA-LSTM network structure but with different parameters. To make use of the context information, the generator handles sequences in the forward direction and the reranker in the backward direction. The CA-LSTM network is introduced in Section 3.1 3.5 and the reranker is discussed in Section 3.6.

[1]The slot values are a set of values that must be satisfied in the response to be a correct answer, such as the price and type of requested restaurants.

3.2 Context-aware Decoding

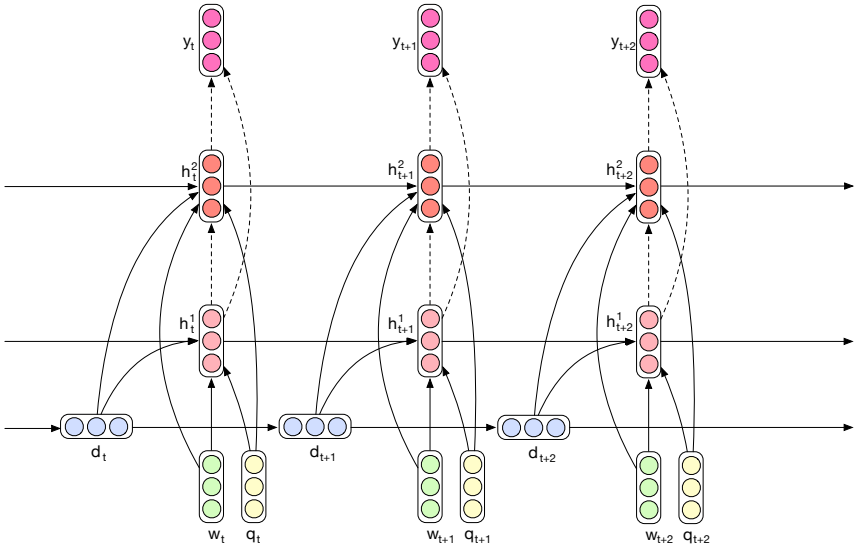


Figure 2: The graphical model of the decoder. w_t is the word embedding of the input token, q_t and d_t is the question vector and slot-value vector respectively, h_t^1 and h_t^2 are the hidden layer representations, and y_t is the probability distribution of the next token.

The decoder is based on the RNN language model (RNNLM), which takes the word embedding w_t of a token as the input at every time step t and outputs the probability distribution y_t of the next token w_{t+1} conditioned on the hidden layer representations h_t^1 and h_t^2. In addition to using the traditional LSTM cell in the RNNLM, we add the question vector $\boldsymbol{q}$ and the slot-value vector $\boldsymbol{d}$ in the network, as shown in Figure 2.

Deep neural networks can improve the network performance by learning multiple levels of feature abstraction. In order to learn more rich features, we adopt a two-layer Stacked-LSTM in the model, which has been proved effective in speech recognition (Graves et al., 2013) and other tasks. We apply the dropout operation on the dashed connections to prevent co-adaptation and overfitting, as shown in Figure 2. The Stacked-LSTM network is defined as follows,

$$i_t^n = \sigma(W_{wi}^n w_t + W_{hi}^n h_{t-1}^n + W_{hhi}^n h_t^{n-1} + W_{qi}^n q_t + b_i^n) \quad (1)$$

$$f_t^n = \sigma(W_{wf}^n w_t + W_{hf}^n h_{t-1}^n + W_{hhf}^n h_t^{n-1} + W_{qf}^n q_t + b_f^n) \quad (2)$$

$$o_t^n = \sigma(W_{wo}^n w_t + W_{ho}^n h_{t-1}^n + W_{hho}^n h_t^{n-1} + W_{qo}^n q_t + b_o^n) \quad (3)$$

$$\hat{c}_t^n = tanh(W_{wc}^n w_t + W_{hc}^n h_{t-1}^n + W_{hhc}^n h_t^{n-1} + W_{qc}^n q_t + b_c^n) \quad (4)$$

$$c_t^n = f_t^n \odot c_{t-1}^n + i_t^n \odot \hat{c}_t^n + tanh(W_{dc}^n d_t) \quad (5)$$

$$h_t^n = o_t^n \odot tanh(c_t^n) \quad (6)$$

where n and t denote the n^{th} layer in space and the t step in time respectively, σ is the sigmoid function, i_t^n, f_t^n, $o_t^n \in [0,1]^n$ are respectively the input gate, forget gate and output gate , $\hat{c}_t^n$ is the candidate cell value and c_t^n is the true cell value, and h_t^n is the hidden layer vector at time step t and layer n, all of which have the same dimension as the hidden layer vector. q_t is the question embedding vector and d_t is the slot-value vectors which will be described soon later.

Recalling that the task-oriented NLG aims at providing correct answers in responses, we thus use the slot-value vector $\boldsymbol{d}$ as the sentence planning cell to regularize the generation process and ensure the information adequacy of responses. The $\boldsymbol{d}$ vector stores a set of slot values that must be satisfied for the generated response to be qualified as a correct answer. The sentence planning cell is defined by the following equations,

$$r_t = \sigma(W_{wr} w_t + \sum_n \alpha_n W_{hr}^n h_{t-1}^n) \quad (7)$$

$$d_t = r_t \odot d_{t-1} \tag{8}$$

where $r_t \in [0,1]^d$ is the reading gate, d_0 is the initial one-hot slot-value vector by setting the corresponding bit to 1 and d_t is the slot-value vector at time step t which updates at every time step with the reading gate r_t. d_t influences the cell value c_t of the traditional LSTM cell by adding a nonlinear transformation $tanh(W_{dc}d_t)$ as shown in Equation 5.

Finally, the output probability distribution y_t of the next token w_{t+1} is computed as follows, and the next token w_{t+1} is generated by sampling from the distribution y_t.

$$w_{t+1} \sim y_t = P(w_{t+1} \mid w_t, w_{t-1}, , w_0, q_t, d_t) = softmax(\sum_n W_{ho}^n h_t^n) \tag{9}$$

3.3 Attention-based Encoding

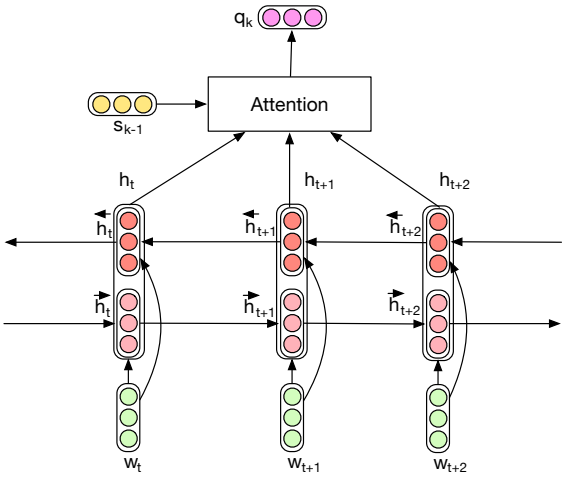


Figure 3: The graphical model of the encoder. h_t is the hidden layer representation of the input question at time step t for encoder, s_{k-1} is the state of the decoder at time step $k-1$ for decoder, q_k is the question vector at time step k for decoder generated by attention of $\boldsymbol{h}$ and the state of decoder s_{k-1}.

The encoder is built on Long Short-term Memory network. To model the question sequence with both of the preceeding and following contexts, we apply a Bidirectional-LSTM architecture (Graves et al., 2005) to the encoder as shown in Figure 3. The $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ are the two directional hidden representations which are computed by iterating the forward layer from $t = 1$ to T and the backward layer from $t = T$ to 1 respectively. By concatenating the $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$, we obtain the hidden representation of the question sequence $h_t = [\overrightarrow{h}_t ; \overleftarrow{h}_t]$.

Although we can feed the hidden layer representation h_T at time step T to the decoder as the question vector q_k, this method has its shortcomings: the question is processed at one time instead of being dynamically attended as the decoder proceeds. To address this problem, the attention mechanism (Bahdanau et al., 2014) is applied to generate question vector according to the hidden layer representation $\boldsymbol{h}$ and the state of the decoder s_{k-1}, which is computed as follows,

$$q_k = \sum_{t=1}^{T} \alpha_{kt} h_t \tag{10}$$

$$\alpha_{kt} = \frac{exp(e_{kt})}{\sum_{j=1}^{T} exp(e_{kj})} \tag{11}$$

$$e_{kt} = v_a^T tanh(W_a s_{k-1} + U_a h_t) \tag{12}$$

$$s_{k-1} = [h_{k-1}^1; h_{k-1}^2] \tag{13}$$

where α_{kt} is the weight of every hidden layer representation h_t of the question sequence, and s_{k-1} is the state of the decoder by concatenating h_{k-1}^1 and h_{k-1}^2 of the decoder. By weighted average of the hidden

layer representation $\boldsymbol{h}$, the decoder can obtain the critical information of the question sequence when generating the next token.

3.4 Act Type Embedding

Dialogue act consists of two parts: the act type that denotes the goal of the response to be generated, such as confirm and recommend, and the slot values which should be satisfied in the response, such as the price range or area of a requested restaurant. The act type information can be very useful to generate the response, however, it has not been fully exploited by SC-LSTM. In order to make the best use of the act type information, we embed the act type to an n-dimentional vector v_d which is randomly initialized and subsequently learned through the iterative training. By concatenating the act type embedding vector v_d and the word vector w_t and feeding them to the network, the act type information is able to influence the generation process at a global level. The computation of input gate, forget gate, output gate, candidate cell and reading gate of the network are updated by the following equations,

$$i_t^n = \sigma(W_{wi}^n[w_t; v_d] + W_{hi}^n h_{t-1}^n + W_{hhi}^n h_t^{n-1} + W_{qi}^n q_t + b_i^n) \quad (14)$$

$$f_t^n = \sigma(W_{wf}^n[w_t; v_d] + W_{hf}^n h_{t-1}^n + W_{hhf}^n h_t^{n-1} + W_{qf}^n q_t + b_f^n) \quad (15)$$

$$o_t^n = \sigma(W_{wo}^n[w_t; v_d] + W_{ho}^n h_{t-1}^n + W_{hho}^n h_t^{n-1} + W_{qo}^n q_t + b_o^n) \quad (16)$$

$$\hat{c}_t^n = tanh(W_{wc}^n[w_t; v_d] + W_{hc}^n h_{t-1}^n + W_{hhc}^n h_t^{n-1} + W_{qc}^n q_t + b_c^n) \quad (17)$$

$$r_t = \sigma(W_{wr}[w_t; v_d] + \sum_n \alpha_n W_{hr}^n h_{t-1}^n). \quad (18)$$

3.5 Training

The objective function is based on the cross entropy error between the predicted token distribution y_t and the gold distribution p_t in the training corpus. And to regularize the slot-value vector d_t, the cost function is modified to the following equation as suggested by (Wen et al., 2015),

$$F(\theta) = \sum_t p_t^\top log(y_t) + \parallel d_T \parallel + \sum_{t=0}^{T-1} \eta \xi^{\|d_{t+1} - d_t\|} \quad (19)$$

where d_T is the slot-value vector at the last time step T , and η and ξ are constants set to 10^4 and 100, respectively. To minimize $\parallel d_T \parallel$ is used to encourage the responses to provide adequate required slots. And the last term $\sum_{t=0}^{T-1} \eta \xi^{\|d_{t+1} - d_t\|}$ discourages the reading gate from turning off more than one bit of slot-value vector in a single time step. The parameters of CA-LSTM network are randomly initialized except for the word embeddings which are initialized by pre-trained 300-dimension word vectors (Pennington et al., 2014). The trade-off weights α are set to 0.5 as mentioned in Section 3.2 and 3.4. The dimension of hidden layer and act type embedding are set to 80 and 20 respectively. The decoder takes the two hidden layer Stacked-LSTM network with a 50% dropout rate. The network is trained with back propagation through time (Werbos, 1990) by treating each user question and system response turn as a mini-batch. AdaDelta (Zeiler, 2012) is applied to optimise the parameters, and the word embeddings are fine tuned through the iterative training. The forward generator and backward reranker are based on the same CA-LSTM network while the parameters are independent.

3.6 Reranking

In the decoding phase, the decoder generates a candidate set of response sequences by randomly sampling the probability distribution of next token. In the reranking phase, the candidate set is reranked by the sentence $score$. As a result, the top-n responses of the candidate set are chosen as the output responses. By combining the cost $F_f(\theta)$ of the forward generator as defined in Eq. (19), the cost $F_b(\theta)$ of the backward reranker (also in Eq. (19)) and the penalty of the error slot, the $score$ is defined as follows,

$$score = -(F_f(\theta) + F_b(\theta) + \lambda \frac{p+q}{N}) \quad (20)$$

where p, q are the number of missing and redundant slots respectively, N is the number of slots required, and λ is set to 100000 to penalize the response with wrong answers.

act types	inform, inform only match inform no match, confirm, hello select, request, reqmore, goodbye
slot names	name, type, *pricerange, postcode price, phone, address, *area, *near *food, *goodformeal, ***kids-allowed**

Table 1: The details about the dialogue act. **bold**=binary slots, *=slots can take the value of *dont care*.

Method	BLEU-4
hdc	0.451
kNN	0.602
classlm	0.627
SC-LSTM	0.731
CA-LSTM	0.775
CA-LSTM+att	0.783
CA-LSTM+att+emb	**0.790**

Table 2: BLEU-4 score of the top 5 responses.

4 Experiments

4.1 Dataset Description

To evaluate the performance of CA-LSTM, we adopt the SF Restaurant dataset as used in (Wen et al., 2015), which is a corpus of a spoken dialogue system providing information about restaurants in San Francisco. It has around 5000 user question and system response turns sampled from about 1000 dialogues. The act types and slot-value pairs are labeled in the dataset. The details about the dialogue act are provided in Table 1. The training, validation and testing set are partitioned in the ratio of 3:1:1. And upsampling w.r.t act type is applied to make the corpus more uniform similar to (Wen et al., 2015).

4.2 Implementation Details

We use Theano (Bergstra et al., 2010) to implement the proposed model. For each dialogue act and input question, we generate 20 responses and select the top 5 responses as the output after reranking. The BLEU-4 metric (Papineni et al., 2002) implemented by NLTK (Bird, 2006) is used for quantitative evaluation. And the references set of the BLEU-4 metric are built by grouping the references of the same dialogue acts after delexicalising the responses and lexicalizing them by the correct values. Since the performance of CA-LSTM depends on initialisation, the results shown below are averaged over 5 randomly initialised CA-LSTM and the corpus are partitioned after random shuffle as well.

4.3 Quantitative Evaluation

We compare our proposed model with several baselines including: the handcrafted generator (hdc), k-nearest neighbour (kNN), class-based LMs (classlm) as proposed by Oh and Rudnicky (2000), the 2-hidder-layer semantically conditioned LSTM network (SC-LSTM) proposed by Wen et al. (2015). For our own method, we experiment with several settings: the basic setting (denoted by CA-LSTM), the Context-Aware LSTM with attention (CA-LSTM+att) which encodes the question vector with an attention mechanism, and the Context-Aware LSTM with attention and act type embeddings (CA-LSTM+att+emb). The result is shown in Table 2. As we can see, the performances of our methods have been greatly improved compared to the baselines shown in the first block (hdc,kNN,classlm and SC-LSTM). By combining more context information (attention and act type embeddings), the performance of CA-LSTM is further improved correspondingly. And the Context-Aware LSTM with attention and act type embeddings (CA-LSTM+att+emb) obtains the best overall performance.

4.4 Human Evaluation

Method	Informativeness	Naturalness
classlm	2.28	2.32
SC-LSTM	2.62	2.63
CA-LSTM	**2.78**	**2.74**

Table 3: Human evaluation for the quality of top 5 responses on two metrics (rating out of 3).

Pref.%	classlm	SC-LSTM	CA-LSTM
classlm	-	16.9	15.7
SC-LSTM	83.1	-	25.3
CA-LSTM	**84.3**	**74.7**	-

Table 4: Pairwise preference among the three systems.

We recruit 10 judges for human evaluation experiments. For each task, the three systems(classlm, SC-LSTM and CA-LSTM with attention and act type embeddings which is denoted by CA-LSTM for simplicity) are used to generate 5 responses. Judges are asked to score each of them in terms of informativeness and naturalness (rating scale is 1,2,3), and also asked to state a preference between any two of them. The informativeness is defined as whether the response provides all the information contained in the DA and the naturalness is defined as whether the response could plausibly have been produced by a human as proposed by Wen et al.(2015). We test 200 DAs and 1000 responses per system in total. The result of human evaluation for the quality of response is shown in Table 3. As can be seen, CA-LSTM outperforms the baseline methods in both metrics of informativeness and naturalness significantly ($p < 0.05$, Student's t-test). Besides, CA-LSTM is preferred by judges as shown in Table 4 where CA-LSTM is much more perferred than SC-LSTM.

4.5 Case Study

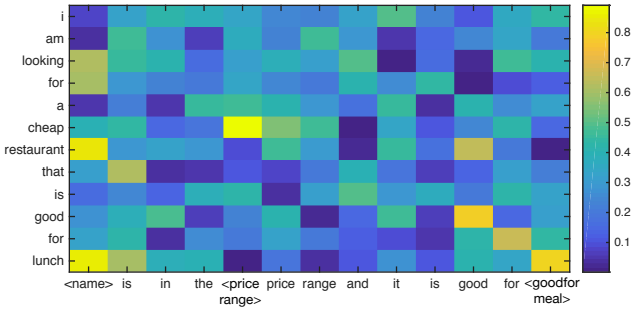


Figure 4: Attention matrix visualization

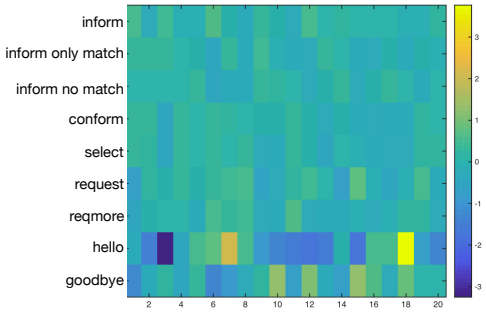


Figure 5: AT embedding visualization

Figure 4 visualizes the attention matrix for a pair of input question and output response. As can be seen, the weights of keywords in the question are strengthened when the decoder generates the relevant tokens, for example the weights of "restaurant", "lunch", "looking", "for" are augmented when the decoder generates the slot token "<name>".

Figure 5 visualizes the learned AT embeddings. The x-axis indicates the dimension index of AT embeddings and the color indicates the value of the corresponding dimension. The similar act types have similar AT embeddings such as "inform" and "inform only match", "request" and "reqmore", while dissimilar act types have different AT embeddings such as "hello" and "goodbye".

CA-LSTM can generate more coherent responses than SC-LSTM for particular act types such as "inform no match". Quantitative evaluation shows that CA-LSTM achieves 0.858 BLEU-4 score compared to 0.772 of SC-LSTM for the act type of "inform no match". For this act type, CA-LSTM can generate negation responses, such as " there is no basque restaurant that allows child -s. ", while SC-LSTM tends to ignore the negation information and generates responses like " there are basque restaurant -s that allow kid -s .".

5 Conclusion and Future Work

In this paper, we have proposed a statistical task-oriented NLG model based on a Context-Aware Long Short-term Memory (CA-LSTM) recurrent network. The network can learn from unaligned data without any heuristics, and it can generate variant responses and provide correct answers in response to the input information. Both quantitative evaluation and human evaluation show that CA-LSTM obtains the state-of-the-art performance. We also reveal the influence of the attention mechanism and act type embeddings with case studies. As future work, we would explore the NLG task in different domains and scenarios which need to consider more context information in the generation process.

6 Acknowledgements

This work was partly supported by the National Basic Research Program (973 Program) under grant No. 2013CB329403, the National Science Foundation of China under grant No.61272227/61332007.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

John Bateman and Renate Henschel. 1999. From full generation to near-templates without losing generality. In *Proceedings of the KI'99 workshop," May I Speak Freely*.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Stephan Busemann and Helmut Horacek. 2002. A flexible shallow approach to text generation. *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 238–247.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks*, pages 799–804. Springer.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

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

Jiwei Li, Will Monroe, Alan Ritter, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Danilo Mirkovic and Lawrence Cavedon. 2011. Dialogue management using scripts, October 18. US Patent 8,041,570.

Alice H Oh and Alexander I Rudnicky. 2000. Stochastic language generation for spoken dialogue systems. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3*, pages 27–32. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John Mcintyre Conference Centre, Edinburgh, Uk, A Meeting of Sigdat, A Special Interest Group of the ACL*, pages 583–593.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Hierarchical neural network generative models for movie dialogues. *arXiv preprint arXiv:1507.04808*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1577–1586.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Marilyn A Walker, Owen C Rambow, and Monica Rogati. 2002. Training a sentence planner for spoken dialogue using boosting. *Computer Speech & Language*, 16(3):409–433.

Tsung Hsien Wen, Milica Gasic, Nikola Mrksic, Pei Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1711–1721.

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

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2972–2978.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Weakly-supervised text-to-speech alignment confidence measure

Guillaume Serrière, Christophe Cerisara, Dominique Fohr, Odile Mella
LORIA URM 7503, 54506 Vandoeuvre-les-Nancy, France
guillaume.serriere@telecomnancy.net
{cerisara,fohr,mella}@loria.fr

Abstract

This work proposes a new confidence measure for evaluating text-to-speech alignment systems outputs, which is a key component for many applications, such as semi-automatic corpus anonymization, lips syncing, film dubbing, corpus preparation for speech synthesis and speech recognition acoustic models training. This confidence measure exploits deep neural networks that are trained on large corpora without direct supervision. It is evaluated on an open-source spontaneous speech corpus and outperforms a confidence score derived from a state-of-the-art text-to-speech aligner. We further show that this confidence measure can be used to fine-tune the output of this aligner and improve the quality of the resulting alignment.

1 Introduction

This work focuses on the text-to-speech alignment (T2SA) task, which consists in temporally aligning a given speech sound file with its known text transcription. The standard objective quality metric is the expected alignment error, measured in seconds and defined as $\mathcal{L} = E[|\hat{t} - t|]$, where t is the gold timestamp of a word boundary, and $\hat{t}$ the corresponding timestamp estimated by the aligner (Keshet et al., 2005).

Text-to-speech alignment is an important task for many applications, including: (i) Lip-syncing in cartoons production and film dubbing; (ii) Anonymization of audio corpus; (iii) Pre-processing of audio corpora for training new speech recognition systems; (iv) Indexing audio-visual corpora for browsing and querying; (v) Sampling sounds for speech synthesis; (vi) Second-language learning.

We propose in this work a novel confidence measure for detecting erroneous word boundaries at the output of an existing T2SA system. Accurately detecting misplaced word boundaries is crucial to reduce post-processing costs in every previous application. For instance, only the most reliable segments may be chosen for acoustic model training and speech synthesis, and manual corrections may be limited to the less reliable boundaries for lip syncing and corpus anonymization.

The proposed confidence measure is computed by a deep neural network (DNN) that is trained on a large corpus without any manually annotated word boundaries. We show on a gold corpus of French spontaneous speech that the proposed model is able to detect correct boundaries with a significantly better accuracy than the acoustic models used in the T2SA system, thanks to the acoustic features automatically captured by the deep neural model on the large unlabelled corpus. We further show that the proposed confidence measure may be used to post-process the T2SA output and improve its precision.

2 Related works

Every text-to-speech aligner faces three main challenges: (i) Handling imperfect transcriptions; (ii) Supporting noisy acoustic conditions; (iii) Finding the globally optimal alignment on long (up to a few hours) audio files. Many solutions have been proposed to address these issues. For instance, "anchor-based" approaches (Moreno et al., 1998; de Jong et al., 2006; Hazen, 2006) automatically infer high-confidence words timestamps at regular intervals in a long audio file in order to enable regular batch

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2042–2050, Osaka, Japan, December 11-17 2016.

alignment between two successive anchors. The issue of aligning highly imperfect text to speech may be addressed with standard acoustic adaptation (Zhao et al., 2005), or by performing recognition at the phoneme level only with monophthongs and fricatives, which appear to be more robust to noise than other phonemes (Haubold and Kender, 2007). Complementary, better phonetization models of unknown words may also be used (Bigi, 2013). More generally, various models have been proposed for phoneme alignment, such as discriminative shallow large-margin alignment models in (Keshet et al., 2005), non-neural unsupervised acoustic models in (Milde, 2014) and in (Lanchantin et al., 2015), where two DNNs are used respectively for acoustic modelling for speech transcription and for segmenting the speech file into speech and non-speech segments. A remarkable HMM-based architecture is also proposed in (Brognaux and Drugman, 2016), where the acoustic models are trained solely on the target corpus to align. The authors of (Yuan et al., 2013) demonstrate the importance of producing high-precision temporal limits with dedicated models, and propose in (Stolcke et al., 2014) a neural network to fine-tune the alignment. We follow this line of work, but rather focus on estimating the actual quality of the proposed boundaries with a confidence measure. A confidence measure is proposed in the aligner ALISA (Stan et al., 2016), but its role is to filter-out wrongly recognized sentences. Conversely, few publications address the problem of detecting reliable temporal boundaries after T2SA. (Paulo and Oliveira, 2004) proposes a confidence measure that is based on a synthetic speech signal, while (Keshet et al., 2005) discriminatively trains base functions that define an alignment confidence measure, but which is not evaluated per se: thanks to the decomposability property of the base functions, this measure is rather used with a dynamic programming algorithm to output a final alignment. Our work is, to the best of our knowledge, the first proposal to use the modelling potential of deep networks to compute successful confidence measures of text-to-speech alignment outputs.

3 Proposed model

3.1 Model description

The proposed model is shown in Figure 1. Two models, respectively called the *Boundary inspector* and the *Boundary selector*, are built to compute a confidence measure that any candidate word boundary is correct or not. Both the *Boundary inspector* and *selector* take as input an acoustic window of ± 0.05s around the candidate word boundary, plus two categorical inputs representing the left and right phonemes. They can thus be viewed as acoustic models that are specialized in identifying boundaries between two segments, as opposed to classical acoustic models that are designed to discriminate between phonemes that may generate a given segment.

The *Boundary inspector* is a standard feed-forward deep network with two output neurons, which encodes the probability that the central input frame [1] t is a true word boundary. It thus focuses on a single frame, the middle one, and makes a decision about it. It is completed with another model, the *Boundary selector*, which rather considers simultaneously all possible candidate frames in the interval $t \pm 5$, and decides which one is the most likely to be the target word boundary. Because we know that consecutive frames are more correlated than distant frames, we use a recurrent LSTM model to capture correlation between frames. Because no privileged direction is assumed, we use a bi-directional LSTM. The output of this LSTM is then merged with the contextual phonetic information in a feedforward network with 11 outputs: one for each input frame. Both models are finally combined with a deep feedforward network called *Aggregator*, which is trained separately on another corpus.

3.2 Training

Our choice to use a Deep Neural Network (DNN) is motivated by the potential of deep networks to infer complex hierarchical features from data that would have been difficult to design by hand. But this is only possible on large training corpora, while only our small gold corpus is manually annotated with temporal boundaries. We thus have to rely on one of the common deep learning "tricks" for building a large enough training corpus, such as transfer learning, the use of auxiliary tasks or data augmentation. In this

[1] A frame is a time-segment of 10ms length encoded into an acoustic vector of dimension 39 composed of 12 MFCC (Mel-Feature Cepstral Coefficients) plus their derivative and acceleration.

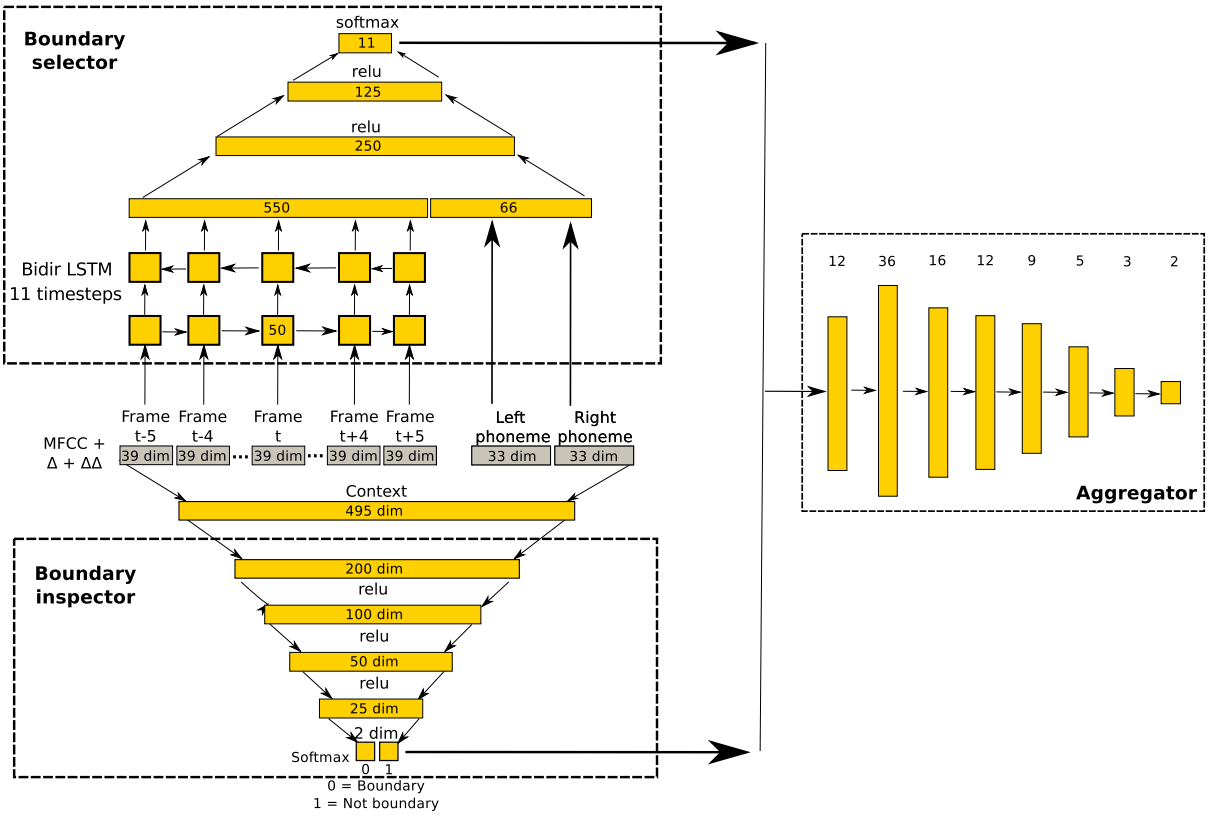


Figure 1: Proposed model. The input vector is composed of 11 frames plus 2 phonemes, shown in the center. Below, the *Boundary inspector* detects whether the middle frame t is a true word boundary or not. Above, the *Boundary selector* is composed of a bi-directional LSTM with 11 timesteps (only 5 are shown) plus 3 feed-forward layers that select the most likely word boundary frame in the input segment. Both models outputs are fed into the *Aggregator* model, on the right, which outputs a confidence probability that frame t is a word boundary.

work, we have decided to combine two state-of-the-art French text-to-speech aligners, ASTALI (Fohr et al., 2015) and JTrans (Cerisara et al., 2009) [2] in order to align part of the open-source ORFEO corpus, composed of 3 million words of French spontaneous speech manually transcribed and available at `http://www.projet-orfeo.fr`.

We then compare both ASTALI and JTrans alignments on this corpus and consider that any word boundary that has the same timestamp in both alignments, within a tolerance of ± 0.02s, is correct[3]. This procedure allows us to automatically build a large training corpus of positive examples which is then completed with 3 times more negative instances obtained by randomly sampling frames that are distant from any ASTALI and JTrans word boundary by at least 0.04s, leading to a training corpus of 377662 examples, which is used to train both the *Boundary inspector* and *selector*.

The same process is used on another set of files from the ORFEO corpus to create a second training corpus of 105406 examples, on which both model output probabilities are computed and used to train the *Aggregator*. During the training of each model, 20% of the training corpus is further reserved to compute a validation loss.

The evolution of the training and validation loss for the three models is shown in Figure 2. These curves suggest that overfitting is not a major issue at this stage. The hyper-parameters of the DNN, including the number and size of the layers, have been set-up empirically with a few trials and errors

[2] JTrans is available on github `https://github.com/synalp/jtrans` and ASTALI is released by its authors

[3] The exact timestamp chosen for this positive temporal limit is the average of the timestamps proposed by JTrans and ASTALI. The tolerance of 0.02s is standard in the phoneme alignment literature (Hosom, 2009).

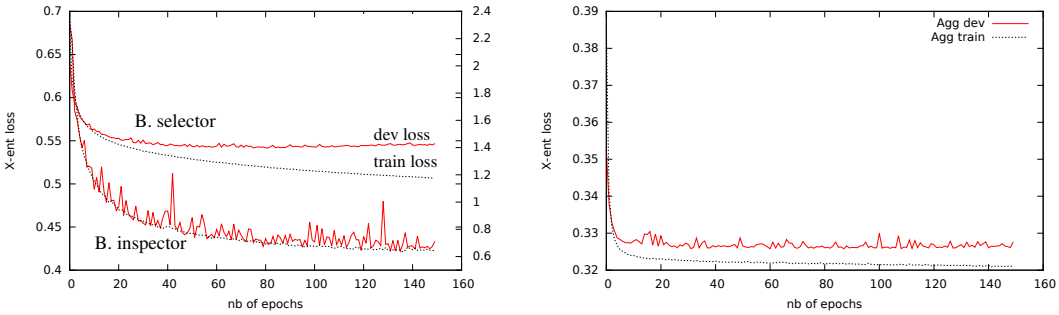


Figure 2: Loss curves during training of the models.

on the training and validation corpus. In particular, we have not used any automatic hyper-parameter tuning strategy. This search of an appropriate model topology has mainly been driven by our motivation to design an architecture that is deep enough to model rich transformations and at the same time that limits the number of parameters to prevent overfitting. Note however that we have only tried a few alternative hyperparameters and thus that the proposed topology is certainly not optimal. The DNN has been implemented with Keras (Chollet, 2015) and trained on these positive and negative instances with the ADAM stochastic gradient descent for 150 epochs.

3.3 Test

The proposed system is evaluated on a gold corpus that is composed of 10988 words extracted from the original ORFEO corpus, and for which 16264 word boundaries, obtained with ASTALI, have been manually corrected. There is no overlapping between this gold corpus and the previous corpora.

At test time, JTrans is run on the test corpus to compute candidate temporal limits of words.

An example of inputs/outputs is shown in Figure 3. Let t be a temporal word boundary[4] given by JTrans, with h_l and h_r respectively the left and right phonemes that are separated by t. For instance, in Figure 3, $t = 186$, h_l =õ and h_r =s.

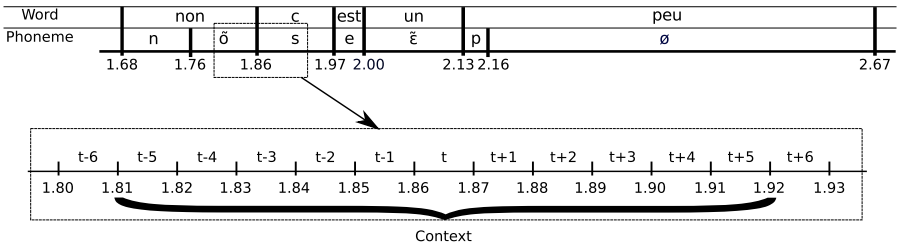


Figure 3: Example of a segmented sentence with its context for the fragment: “no, that’s a bit [...]”

The *Boundary inspector* and *selector* are run on the temporal limits proposed by JTrans and their output probabilities are then passed to the *Aggregator*, which finally returns, for each JTrans temporal limit t, the probability that t is correct or not.

4 Evaluation

4.1 Confidence measure evaluation

Similarly to most other confidence measures in the literature (Yu et al., 2011), we evaluate the proposed confidence measure as a detector of correct vs. erroneous examples. We evaluate next its performances with a Detection Error Tradeoff (DET) curve, which is easier to interpret than the ROC curve (Martin et al., 1997).

[4]Time variables such as t represent an integer number of frames since the start of the audio file. Hence, $t + 2$ is the second frame after the JTrans limit t.

We compare our proposed model first with a baseline confidence measure derived from the acoustic Hidden Markov Models (HMM) used in the text-to-speech alignment process. Let $(w_i)_{1\leq i\leq N}$ be the sequence of words in the transcription. For ease of notation, we assume here without loss of generality that every word is modelled with a single-state Hidden Markov Model (HMM); in fact, every word is actually composed of a sequence of phonemes, and every phoneme is modelled by an HMM with 3 emitting states. However, this hierarchy of models would lead to excessively long equations, and we prefer to simplify the presentation of this baseline.

For a given possible alignment, let the random variable $Q_t = i$ with $1 \leq i \leq N$ represents the index of the word aligned with frame t. By definition, in the context of text-to-speech alignment, a confidence measure for the transition $(Q_t = i, Q_{t+1} = i + 1)$ is given by the posterior probability:

$$P(Q_t = i, Q_{t+1} = i + 1|X, \lambda)$$

where λ represents the parameters of the acoustic models used in JTrans, and $X = (X_t)_{1\leq t\leq T}$ represents all observed acoustic frames.

For our baseline confidence measure, we rely on the acoustic models used in the JTrans system. These acoustic models are Hidden Markov Models, and it is thus well known that the previous posterior can be computed with the forward-backward algorithm:

$$P(Q_t = i, Q_{t+1} = i + 1|X, \lambda) = \frac{\alpha_i(t)a_{i,i+1}\beta_{i+1}(t + 1)b_{i+1}(X_{t+1})}{\sum_{k=1}^{N}\sum_{l=1}^{N}\alpha_k(t)a_{k,l}\beta_l(t + 1)b_l(X_{t+1})}$$

where $b_i(X_t)$ is the observation likelihood of frame X_t in state i. $b_i(X_t)$ is modeled in JTrans by a Gaussian Mixture Model. $a_{i,j} = P(Q_{t+1} = j|Q_t = i)$ is the prior transition probability between words i and j, which is irrelevant in the context of text-to-speech alignment, where we just set $a_{i,i} = \frac{1}{2}$ and $a_{i,i+1} = \frac{1}{2}$. α and β are respectively the matrices of forward and backward probabilities, which can be computed recursively:

$$\alpha_j(t) = \left[\sum_{i=1}^{N}\alpha_i(t - 1)a_{ij}\right] b_j(X_t)$$

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij}b_j(X_{t+1})\beta_j(t + 1)$$

We first evaluate the quality of the DNN as a detector of correct limits, assuming that any JTrans output boundary is a correct limit when its distance to the corresponding gold limit is smaller than 0.02s, as done during training. The corresponding DET curve is shown in Figure 4.

In the DET plot, the closer the curve is to the bottom-left origin, the better it is. We can observe that the proposed confidence measure is a better detector of true boundaries than the acoustic baseline for all possible detection thresholds. The first row in Table 1 also shows the Equal Error Rate (EER), which is the intersection between the $y = x$ diagonal and the DET. With 36% of equal errors, the proposed confidence measure is significantly better than random and it is the first efficient confidence measure for word boundaries based on acoustic information that we are aware of.

While the EER is a good summary of the DET curve, it can only be computed assuming knowledge of the true labels. The next rows in Table 1 thus show standard detection performances at another operating point, the median, which corresponds to the threshold that tags half of the corpus as positive, and half as negative. The proposed model is then compared with a second baseline, called *JTrans/ASTALI agreement*, which tags every JTrans boundary as positive when it lies within the ± 0.02s interval around the corresponding ASTALI limit. This baseline has already been used to automatically annotate the training corpora of the DNN models (see Section 3.2), except that for training, all boundaries tagged as negative are removed, while they are used here to compute the detection metrics in Table 1.

The acoustic baseline confidence measure is not significantly better than random, which confirms for text-to-speech alignment what has already been reported in the literature for speech recognition, i.e., that

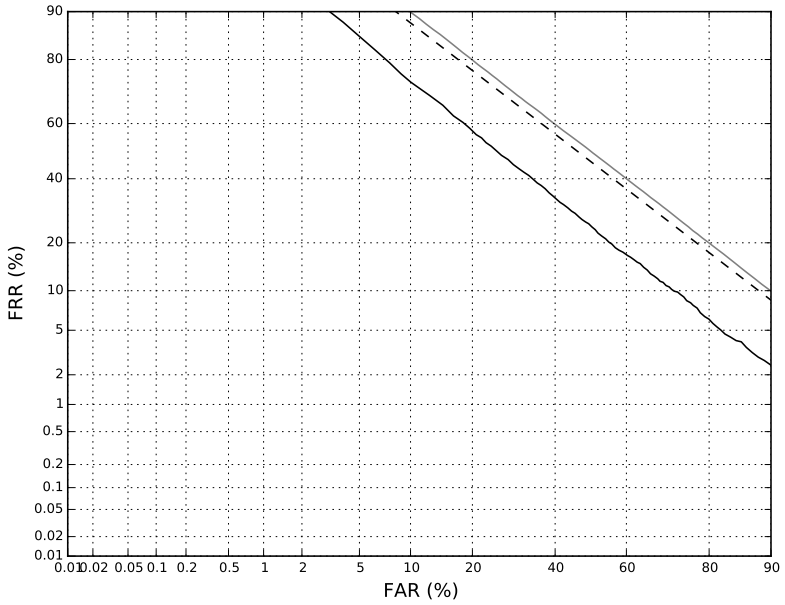


Figure 4: Detection Error Tradeoff curves for detecting word boundaries. The X-axis is the False Accept Rate, while the Y-axis is the False Reject Rate. Three curves are shown, from the worst (top-right corner) to the best (bottom-left corner): random baseline detector (straight grey line), baseline (dash) and our proposed DNN (plain line).

	Acoustic baseline	Proposed model	JTrans / ASTALI agreement
Equal Error Rate (EER)	48%	**36%**	
Precision (median)	43%	52%	60%
Recall (median)	53%	69%	45%
F1 (median)	48%	**60%**	51%

Table 1: Detection performances at fixed operating points of the proposed DNN and two baselines: an acoustic baseline, which computes the posterior of each boundary given JTrans' acoustic models, and a deterministic baseline that tags a boundary as correct when JTrans and ASTALI give close timestamps.

confidence measures based solely on acoustic observation likelihoods usually fail to reliably detect correct words (Willett et al., 1998). This is why state-of-the-art confidence measures for speech recognition mainly exploit other types of features, in particular language-model features (Seigel, 2013). However, language-model information is irrelevant in text-to-speech alignment applications, which makes the task of detecting reliable word boundaries especially challenging. With an F1 of 60%, the performances obtained with our proposed DNN-based confidence measure are thus encouraging, because:

- Our DNN only exploits the same information as speech acoustic models, i.e., acoustic observations and phoneme identities;
- It is trained without manual supervision, only exploiting agreement between two automatic T2SA systems.
- Despite the relatively weak precision of 60% for annotating positive labels in the DNNs' training corpus, the DNN is able to learn relevant acoustic information and provide the first working confidence measure for detecting true word boundaries.

The JTrans/ASTALI agreement baseline has a low recall of 45%. Although this low recall penalizes

its performances as a confidence measure, we can note that the recall is actually not crucial when this JTrans/ASTALI agreement approach is used to automatically annotate training examples for the proposed deep models, because all negative limits are discarded, as explained in Section 3.2. In fact, it may even be preferable to tune this automatic annotation method so that its precision is further increased, at the expense of an even lower recall, so that the positive examples that are kept have a higher likelihood of being correct. However, the JTrans/ASTALI agreement baseline may not easily be tuned in order to increase its precision above 60%. An interesting future work would then be to replace this agreement process with another detector, like the proposed DNN itself, for which the operating point can be tuned, and eventually iteratively retrain the DNN in a self-training fashion on larger corpora, without the need to rely on two different aligners.

4.2 Enhanced aligner evaluation

We propose next a simple post-processing module that enhances the precision of the original T2SA system. This fine-tuning algorithm basically detects suspicious JTrans temporal limits and replaces them by temporal limits with a higher confidence measure in their neighbourhood. It proceeds as follows:

Algorithm 1: Simple fine-tuning of JTrans' output alignment

- For every JTrans output word boundary t:
  - For every distance $d \in 1, 2, 3, 4, 5, \cdots, D$ up to a *maximum distance* D:
    - Compute both DNN output probabilities at distance d from t: $P(t-d)$ and $P(t+d)$
    - Pick the best of both frames $\hat{t} = \arg\max_{t' \in t-d, t+d} P(t')$
    - If the new frame is better than the original JTrans frame $P(\hat{t}) > P(t)$ and better than a minimum confidence threshold $P(\hat{t}) > \delta$, then move the word boundary to $\hat{t}$ and continue with the next boundary t.

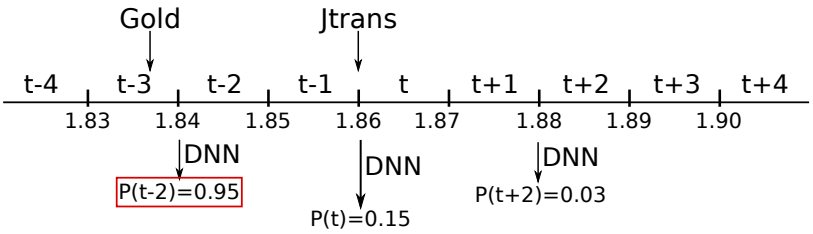


Figure 5: Fine-tuning example. The Jtrans initial limit is 1.86s. The DNN output probability is computed for frames (185,187) first, and then for frames (184,188). The best output is obtained for 1.84s.

Figure 6 plots the original (top horizontal dashed line) and resulting alignment error for various D and δ. Although the global impact of our fine-tuning algorithm is small, it is positive for all D and δ. Because our fine-tuning algorithm just looks for the most confident limits in a neighbourhood of the original JTrans boundary, the iterative application of the confidence measure onto more and more distant frames increases the probability of misclassification. Furthermore, whenever it moves a word boundary, the resulting impact on the previous or following words should be handled, for instance with a Viterbi algorithm. So this experiment is merely a proof of concept that confirms the possibility to post-process a text-to-speech aligner output with the proposed confidence measure; the main focus of this work is rather confidence measure evaluation, which may benefit to many other applications, as discussed in the introduction. These results are thus encouraging to further pursue efforts into investigating weakly supervised deep neural networks for fine-tuning existing text-to-speech aligners.

5 CONCLUSIONS AND FUTURE WORKS

We develop a weakly supervised approach that exploits two existing text-to-speech aligners to automatically annotate a corpus for training a deep neural network-based confidence measure without direct

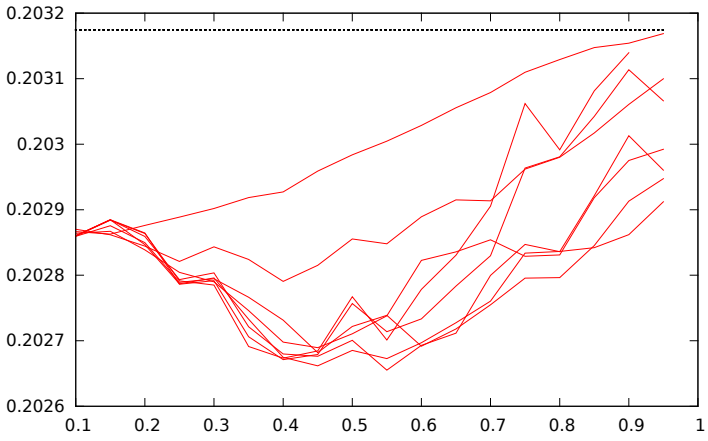


Figure 6: Average alignment error (Y-axis, in seconds) for various maximum distance D (from ± 1 to ± 8 frames), as a function of the minimum confidence threshold (X-axis). The top dashed horizontal line is the original JTrans alignment error.

supervision. We propose two different types of neural networks for this task and combine them within a single model. For now, all three components of the proposed model are trained independently, but we plan to train them jointly in a future work to further improve the resulting model. Experimental results show that the proposed confidence measure outperforms a baseline acoustic confidence measure derived from the original text-to-speech aligner. We further show that it outperforms another baseline, which results from a voting ensemble of both original text-to-speech aligners. This is, to the best of our knowledge, the first good performances ever reported for confidence measure detection of true word boundaries. The performances reached are also interesting because the deep models only exploit acoustic information, which has been shown to be otherwise unsuccessful for confidence estimation in the context of speech recognition, and because these models are trained without manual supervision. These results open the way to further improvements in automatic annotation of unlabelled corpora for text-to-speech alignment, for instance by iteratively re-labelling the training corpus with the proposed model setup in high-precision mode and retraining new confidence models. We further apply the trained confidence measure with a simple corrective algorithm that fine-tunes the output timestamps given by the original text-to-speech aligner. This experiment shows encouraging results for improving text-to-speech alignments thanks to the proposed confidence measure. Possible ways to improve these results include designing a better exploration strategy for fine-tuning the initial alignment, as well as investigating other DNN topologies.

The complete source code as well as links to all datasets is available at `https://github.com/cerisara/speechAlignConfidence`.

Acknowledgments

This work has been partly funded by the ANR ORFEO project. Some experiments presented in this paper have been made possible thanks to the donation of a GPU Titan X card by Nvidia, and were carried out using the Grid'5000 testbed, which is supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see `https://www.grid5000.fr`).

References

B. Bigi. 2013. A phonetization approach for the forced-alignment task. In *Proc. LTC*.

S. Brognaux and T. Drugman. 2016. Hmm-based speech segmentation: Improvements of fully automatic approaches. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(1):5 – 15, January.

C. Cerisara, O. Mella, and D. Fohr. 2009. Jtrans, an open-source software for semi-automatic text-to-speech alignment. In *Proc. INTERSPEECH*, Brighton, UK, September.

F. Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

F. de Jong, R. Ordelman, and M. Huijbregts. 2006. Automated speech and audio analysis for semantic access to multimedia. In *Proc. International Conference on Semantic and Digital Media Technologies*, Athens, Greece, December.

D. Fohr, O. Mella, and D. Jouvet. 2015. De l'importance de l'homogénéisation des conventions de transcription pour l'alignement automatique de corpus oraux de parole spontanée. In *8èmes Journées Internationales de Linguistique de Corpus (JLC2015)*, Orléans, France, September.

A. Haubold and J. R. Kender. 2007. Alignment of speech to highly imperfect text transcriptions. In *Proc. IEEE Conf. on Multimedia and Expo*, July.

T. J. Hazen. 2006. Automatic alignment and error correction of human generated transcripts for long speech recordings. In *Proc. Interspeech*, pages 1606–1609.

J.-P. Hosom. 2009. Speaker-independent phoneme alignment using transition-dependent states. *Speech Communication*, 51(4):352–368, April.

J Keshet, S Shalev-Shwartz, Y Singer, and D. Chazan. 2005. Phoneme alignment based on discriminative learning. In *Proc. Interspeech*, pages 2961–2964.

P. Lanchantin, M. Gales, P. Karanasou, X. Liu, Y. Qian, L. Wang, P. Woodland, and C. Zhang. 2015. The development of the cambridge university alignment systems for the multi-genre broadcast challenge. In *Proc. ASRU*.

A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. 1997. The det curve in assessment of detection task performance. Technical report, DTIC Document.

B. Milde. 2014. Unsupervised acquisition of acoustic models for speech-to-text alignment. Master's thesis, Univ. Darmstadt, April.

P. J. Moreno, C. Joerg, J.-M. Van Thong, and O. Glickman. 1998. A recursive algorithm for the forced alignment of very long audio segments. In *Proc. ICSLP*, December.

S. Paulo and L. C Oliveira. 2004. Automatic phonetic alignment and its confidence measures. In *Advances in Natural Language Processing*, pages 36–44. Springer.

M. S. Seigel. 2013. *Confidence Estimation for Automatic Speech Recognition Hypotheses*. Ph.D. thesis, Univ. of Cambridge, December.

A. Stan, Y. Mamiya, J. Yamagishi, P. Bell, O. Watts, R.A.J. Clark, and S. King. 2016. Alisa: An automatic lightly supervised speech segmentation and alignment tool. *Computer Speech & Language*, 35:116 – 133.

A. Stolcke, N. Ryant, V. Mitra, J. Yuan, W. Wang, and M. Liberman. 2014. Highly accurate phonetic segmentation using boundary correction models and system fusion. In *Proc. ICASSP*, Florence, May. IEEE SPS.

D. Willett, A. Worm, C. Neukirchen, and G. Rigoll. 1998. Confidence measures for hmm-based speech recognition. In *ICSLP*, volume 98, pages 3241–3244. Citeseer.

D. Yu, J. Li, and L. Deng. 2011. Calibration of confidence measures in speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(8):2461–2473.

J. Yuan, N. Ryant, M. Liberman, A. Stolcke, V. Mitra, and W. Wang. 2013. Automatic phonetic segmentation using boundary models. In *Proc. Interspeech*, Lyon, August. ISCA - International Speech Communication Association.

Y. Zhao, L. Wang, M. Chu, F. K. Soong, and Z. Cao. 2005. Refining phoneme segmentations using speaker-adaptive context dependent boundary models. In *Proc. Interspeech*, Lisbon, Portugal, September.

Domainless Adaptation by Constrained Decoding on a Schema Lattice

Young-Bum Kim
Microsoft
Redmond, WA
ybkim@microsoft.com

Karl Stratos*
Bloomberg L. P.
New York, NY
me@karlstratos.com

Ruhi Sarikaya†
Amazon
Seattle, WA
rsarikay@amazon.com

Abstract

In many applications such as personal digital assistants, there is a constant need for new domains to increase the system's coverage of user queries. A conventional approach is to learn a separate model every time a new domain is introduced. This approach is slow, inefficient, and a bottleneck for scaling to a large number of domains. In this paper, we introduce a framework that allows us to have a single model that can handle all domains: including unknown domains that may be created in the future as long as they are covered in the master schema. The key idea is to remove the need for distinguishing domains by explicitly predicting the schema of queries. Given permitted schema of a query, we perform constrained decoding on a lattice of slot sequences allowed under the schema. The proposed model achieves competitive and often superior performance over the conventional model trained separately per domain.

1 Introduction

Recently, there has been much investment on the personal digital assistant (PDA) technology in industry (Sarikaya, 2015). Apple's Siri, Google Now, Microsoft's Cortana, and Amazon's Alexa are some examples of personal digital assistants. Spoken language understanding is an important component of these examples that allows natural communication between the user and the agent (Tur, 2006; El-Kahky et al., 2014; Kim et al., 2015a; Kim et al., 2016b). PDAs support a number of scenarios including creating reminders, setting up alarms, note taking, scheduling meetings, finding and consuming entertainment (i.e. movie, music, games), finding places of interest and getting driving directions to them. The number of domains supported by these systems constantly increases, and whether there is a method that allows us to easily scale to a larger number of domains is an unsolved problem (Kim et al., 2015d; Kim et al., 2016a).

The main reason behind the need for additional domains is that we require a new set of schema (i.e., query topics), composed of intents, and slots for processing user queries in a new category. For example, a query in the TAXI domain is processed according to domain-specific schema that is different from those in the HOTEL domain. This in turn requires collecting and annotating new data, which is time consuming and expensive. Once the data is prepared, we also need to build a new system (i.e., models) for this specific domain. In particular, slot modeling is one of the most demanding components of the system in terms of costs in annotation and computation.

In this paper, we introduce a new approach that entirely removes the costs traditionally associated with enlarging the set of supported domains while significantly improving performance. This approach uses a single model to handle all domains: including unknown domains that may be created in the future using a combination of intents and slots in the master schema. The key idea is to remove the need for distinguishing domains by explicitly predicting topics/schema of queries. Thus we obviate the need and directly predict the schema from queries by multi-label classification (either with an RNN or with binary

* Work done while at Columbia University.

† Work done while at Microsoft.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2051–2060, Osaka, Japan, December 11-17 2016.

logistic regressions). Given permitted schema of a query, we perform constrained decoding on a lattice of slot sequences allowed under the schema.

In experiments on slot tagging 17 Cortana personal digital assistant domains, we observe that our single model outperforms each of the 17 models trained separately on different domains. This is because our model is able to leverage the data in all domains by reusing the same slots. It can be viewed as a form of domain adaptation, although "domainless adaptation" may be a more accurate description since we remove the need for distinguishing domains!

2 Background

2.1 Domain Adaptation

The goal of domain adaptation is to jointly leverage multiple sources of data (i.e., domains) in attempt to improve performance on any particular domain. There is a rich body of work in domain adaptation for natural language processing. A notable example is the feature augmentation method of Daumé III (2009), who propose partitioning the model parameters to those that handle common patterns and those that handle domain-specific patterns. This way, the model is forced to learn from all domains yet preserve domain-specific knowledge.

Another domain adaptation technique used in natural language processing utilizes unlabeled data in source and target distributions to find shared patterns (Blitzer et al., 2006; Blitzer et al., 2011). This is achieved by finding a shared subspace between the two domains through singular value decomposition (SVD). Unlike the feature augmentation method of Daumé III (2009), however, it does not leverage labeled data in the target domain.

This work is rather different from the conventional works in domain adaptation in that we remove the need to distinguish domains: we have a single model that can handle arbitrary (including unknown) domains. Among other benefits, this approach removes the error propagation due to domain misclassification. Most domain adaptation methods require that we know the data's domain at test time (e.g., the feature augmentation method). But in practice, the domain needs to be predicted separately by a domain classifier whose error propagates to later stages of processing such as intent detection and slot tagging.

2.2 Constrained Decoding

In a later section, we perform constrained decoding on a lattice of possible label sequences. This technique was originally proposed for transfer learning by Täckström et al. (2013). Suppose we have sequences that are only partially labeled. That is, for each token x_j in sequence $x_1 \dots x_n$ we have a set of allowed label types $\mathcal{Y}(x_j)$. Täckström et al. (2013) define a constrained lattice $\mathcal{Y}(x, \tilde{y}) = \mathcal{Y}(x_1, \tilde{y}_1) \times \dots \times \mathcal{Y}(x_n, \tilde{y}_n)$ where at each position j a set of allowed label types is given as:

$$\mathcal{Y}(x_j, \tilde{y}_j) = \begin{cases} \{\tilde{y}_j\} & \text{if } \tilde{y}_j \text{ is given} \\ \mathcal{Y}(x_j) & \text{otherwise} \end{cases}$$

We compute the most likely sequence in the lattice for a given observation sequence x under model θ as:

$$y^* = \arg\max_{y \in \mathcal{Y}(x, \tilde{y})} p_\theta(y|x)$$

3 Methods

In this section, we describe our domainless prediction framework. It consists of two stages:

1. Given a query, we perform *multi-label classification* to predict a set of allowed schema for the query.
2. Given the predicted schema, we perform *constrained decoding* on the lattice of valid slot sequences.

Since this framework does not involve domain prediction at all, given a query in an unknown domain we can still use the same model to infer its slot sequence, as long as the new domain is composed of existing slots and intents. In cases where the new domain needs an a new intent or slot, the underlying generic models have to updated with the updated schema.

3.1 Schema Prediction

The first stage produces a set of label types that serve as constraints in the second stage. To this end, we use Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) (Figure 1). The LSTM processes the given query to produce a fixed-size vector where the input at each time step is the word embedding corresponding to the word used at the time. We initialize these word embeddings with GloVe vectors (Pennington et al., 2014). Then the network maps the query vector to a distribution over schema types.

In more detail, we first map each word of the utterance into d-dimensional vector space using an embedding matrix of size V by d (which is trained along with other parameters in the network), where V is the size of the vocabulary. Then we map the sequence of the word vectors, $\{x_1, \dots x_T\}$, to LSTM outputs $\{h_1, \dots h_T\}$ where we take the last output to be a d-dimensional summary vector of the utterance $s = h_T$. We then use parameters $W \in \mathbb{R}^{k \times d}$ and $b \in \mathbb{R}^k$ where k is the number of slot types and compute

$$\hat{y} = \text{softmax}\left(Ws + b\right)$$

Thus $\hat{y}_i \in [0, 1]$ is the probability of slot i for the given utterance. To train the model, we minimize the sum of squared errors $||\hat{y} - y||$ (we could certainly use other metrics such as the KL divergence, but we did not pursue this direction).

At test time, we need to perform multi-label classification with the predicted probabilities of slot types $\hat{y} \in [0, 1]^k$. We achieve this by thresholding. But rather than using 0.5 as the threshold, we use the *minimum* probability of a ground-truth schema type from the training data. This results in predictions that are very high in recall at the expense of some precision. This trade-off is suitable in our setting, since these labels are only constraints in the second stage: while missing true labels causes the second stage to fail, over-predicting labels does not.

Since the minimum probabilities of ground-truth schema types are observed in the training data, we can train an separate model (SVM) to predict the threshold value for unseen inputs. In summary, given a test utterance, we first use the LSTM network to compute a distribution of slot types $\hat{y}$, next use the trained regressor to predict a suitable value of threshold, and take labels that have probabilities higher than the threshold. Figure 1 illustrates the process.

3.2 Constrained Decoding

In sequence learning, given a sample query $x_1 \dots x_n$, the decoding problem is to find the most likely tag sequence among all the possible sequences, $y_1 \dots y_n$:

$$f(x_1 \dots x_n) = \arg\max_{y_1 \dots y_n} p(x_1 \dots x_n, y_1 \dots y_n)$$

Here, given constraints $\tilde{y}$ from the first stage, we can simply define a constrained lattice *lattice* $\mathcal{Y}(x, \tilde{y}) = \mathcal{Y}(x_1, \tilde{y}) \times \dots \times \mathcal{Y}(x_n, \tilde{y})$ by pruning all tags not licensed by the constraints, as shown in Figure 2. Then, to find the most likely tag sequence which does not violate the given constrained lattice, we perform the decoding in the constrained lattice:

$$f(x_1 \dots x_n, \tilde{y}) = \arg\max_{\mathcal{Y}(x, \tilde{y})} p(x_1 \dots x_n, y_1 \dots y_n)$$

In experiments, we train a single sequence labeling model (CRF) on all domains, but at test time apply this constrained decoding with slot types predicted by the model in Section 3.1.

3.3 Relation to the Union Method

One of the most naive baselines in domain adaptation is to simply train a single model on the union of all data in different domains; at test time, the model predicts labels for any input regardless of which domain it comes from. Since our approach uses all data as well, it can be seen as a variation on this naive method.

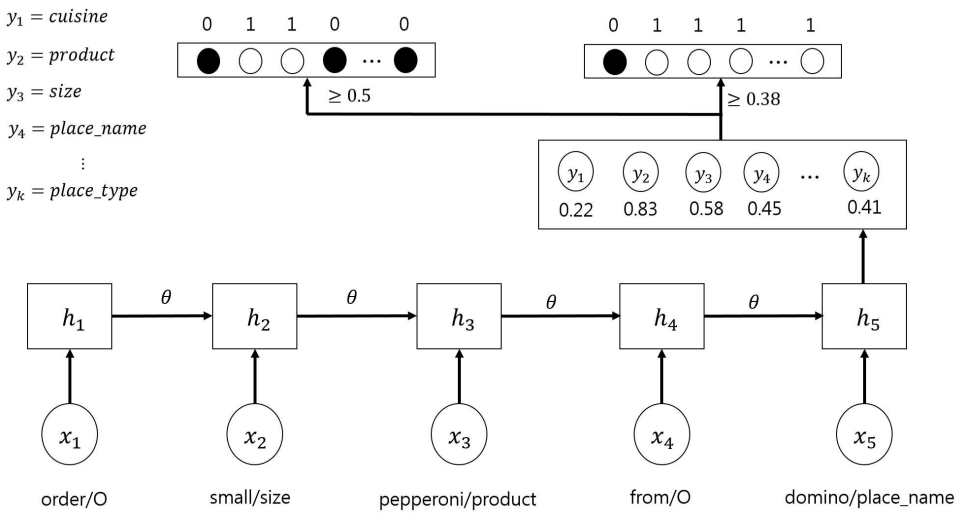


Figure 1: Illustration of schema prediction. In given a query, "order small pepperoni from domino", the word "small", "pepperoni" and "domino" is tagged as *size*, *product* and *place_name*, respectively. Therefore, LSTM multi-label classification model should predict a set of slots a query would be tagged with. When we fix a threshold for final result to 0.5, we can get two permitted labels, *product* and *size*. Whereas, we select different threshold corresponding to each query such as 0.38 in this example, we can obtain four permitted labels, *product*, *size*, *place_name* and *place_type*.

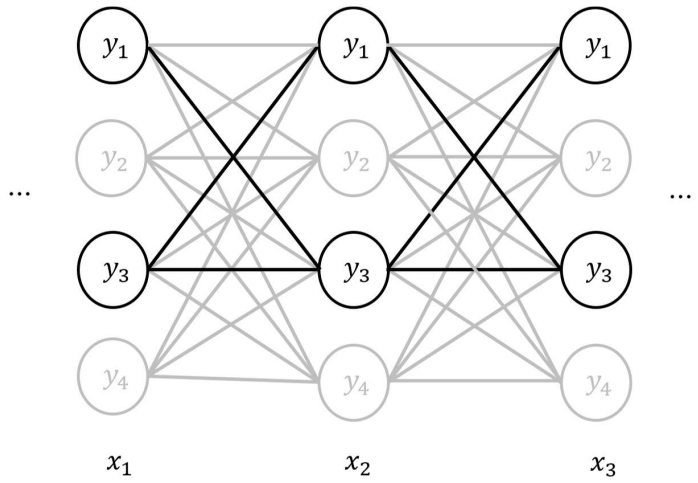


Figure 2: Constrained Lattice: Disabling nodes and transition while decoding the lattice to honor given constraints of domain schema.

The naive method also implicitly makes a decision on the domain of a query when the model predicts domain-specific labels. But it is well-known that this approach typically, unlike ours, yields poor performance. We conjecture that the reason for poor performance is the following. In the union method, the model must perform the *segmentation* as well as labeling of slots, which involves predicting labels in the BIO format (B: begin, I: inside, O: outside) (Ramshaw and Marcus, 1999). In comparison, our method only predicts possible labels and delegates inference to constrained decoding. Thus it can potentially

make more efficient use of labeled data.

4 Experiments

In this section, we turn to experimental findings to provide empirical support for our proposed methods.

4.1 Setting

	# of labels	# of shared label	#train	#test	#dev	#vocab	Description
Alarm	8	6	178K	14K	13K	3628	Set alarms
Calendar	20	16	220K	17K	15.6K	11574	Set events in calendar
Comm.	21	13	780K	72K	29K	69817	Make a call and sent text
Entertain.	15	5	173K	11K	9.3K	17521	Search movies and music
Events	6	4	12K	6K	5K	835	Purchase tickets to events
Hotel	17	9	7.3K	5.7K	4.9K	8172	Book hotel
Mediactrl	10	8	132K	19K	16K	12802	Set up a music player
Mvtickets	7	7	13K	8.2K	7.8K	2298	Buy movie tickets and find showtime
Mystuff	18	12	6.4K	3.6K	3.2K	8824	Find files and attachments
Note	3	1	7.8K	2.9K	2.5K	4756	Find, edit and create a note
Ondevice	6	5	259K	9.4K	6.4K	5386	Control the device
Orderfood	11	10	20K	2.7k	2.6K	3745	Order food using app
Places	32	19	488K	9.4K	8.7K	51611	Find location and direction
Reminder	16	12	338K	21.8K	18K	27823	Find, edit and create reminders
Reservations	12	11	17K	4K	3K	2920	Make a restaurant reservations
Taxi	10	10	10.7K	4.9K	3.1K	451	Find and book a cab
Weather	9	2	302K	5.5K	5.2K	12344	Ask weather
Overall	131	62	2964K	217K	153K	245K	

Table 1: Data sets used in the experiments. For each domain, the number of unique slots, the number of examples in the training, development, and test sets, input vocabulary size of the training set, and short description about domain.

To test the effectiveness of the proposed approach, we apply it to a suite of 17 Cortana personal assistant domains for slot (label) tagging tasks, where the goal is to find the correct semantic tags of the words in a given user utterance. For example, a user could say "reserve a table at joeys grill for thursday at seven pm for five people". Then the goal is to tag "joeys grill" with `restaurant`, "thursday" with `date`, "seven pm" with `time`, and "five" with `number_people`. The data statistics and short descriptions about the 17 domains are shown in Table 1. As the table indicates, the domains have very different granularity and diverse semantics. The total numbers of training, test and development queries across domains are 2964K, 217K and 153K, respectively. Note that we keep domain-specific slots such as `alarm_state`, but there are enough shared labels across domains. To be specific, we have shared 62 labels among 131 labels. In ALARM domain, there are 6 shared slots among 8 slots.

4.2 Results

In all our experiments, we follow same setting as in (Kim et al., 2015b; Kim et al., 2015c). We trained Conditional Random Fields (CRFs)(Lafferty et al., 2001) and used n-gram features up to $n = 3$, regular expression, lexicon features, and Brown Clusters (Brown et al., 1992). With these features, we compare the following methods for slot tagging[1]:

- *In-domain*: Train a domain-specific model using the domain-specific data covering the slots supported in that domain.
- *Binary*: Train a binary classifier for each slot type, assuming prediction for each slot type is independent of one another. Then combine the classification result with the slots needed for a given schema. For each binary slot tagger targeting a specific slot type, the labeled data is programatically

[1]For parameter estimation, we used L-BFGS (Liu and Nocedal, 1989) with 100 as the maximum number of iterations and 1.0 for the L2 regularization parameter.

Domain	In-domain	Binary	Post	Const(CRFs)	Const(LSTM)
Alarm	92.89	74.49	89.81	93.56	94.23
Calendar	90.03	75.62	82.14	88.57	88.16
Communication	92.94	84.17	86.93	92.14	90.39
Entertainment	93.83	83.28	91.26	93.17	94.37
Events	85.84	69.84	78.30	85.00	85.80
Hotel	91.25	73.86	77.45	91.12	90.81
Mediacontrol	86.39	83.70	86.22	87.07	86.43
Movietickets	91.75	85.39	87.03	91.06	91.35
Mystuff	87.92	51.30	80.48	84.88	82.46
Note	87.60	51.25	71.67	84.32	83.87
Ondevice	93.59	70.13	88.26	94.27	94.08
Orderfood	93.52	83.34	90.74	92.84	91.84
Places	91.75	75.27	87.69	89.55	90.96
Reminder	89.31	72.67	81.38	88.57	88.27
Reservations	92.68	86.10	91.07	93.56	94.32
Taxi	88.27	76.91	85.50	90.32	89.65
Weather	96.27	89.12	94.38	96.44	96.50
Average	90.93	75.67	85.31	90.38	90.21

Table 2: F1 scores for models which can handle all domains.

mapped to create a new labeled data set, where only the target label is kept while all the other labels are mapped `other` label.

- *Post*: Train a single model with all domain data, take the one-best parse of the tagger and filter-out slots outside the a given schema.

- *Const*: Train a single model with all domain data and then perform constrained decoding using a given schema.

To evaluate performance of the constrained decoding approach without schema prediction, we compare the performance among possible models, which can handle all domains in Table 2. Here, a schema is given from a pre-trained domain classifier with an average accuracy of 97%.

We consider *In-domain* as a plausible upper bound of the performance, yielding 90.93% of F1 on average. Second, *Binary* has the lowest performance of 75.67%. When we train a binary classifier for each slot type, the other slots that provide valuable contextual information are ignored. This leads to the degradation in tagging accuracy. Third, *Post* improves F1 scores across domains, resulting in 85.31% F1 on average. Note that this technique does not handle ambiguities and data distribution mismatches due to combining multiple domain specific data with different data sizes. Finally, *Const(CRF)* leads to consistent gains across all domains, achieving 90.38%, which almost matches the *In-domain* performance. *Const(CRF)* performs better than *Binary* because *Const(CRF)* constrains the best path search to the target domain schema. It does not consider the schema elements that are outside of the target domain schema. By doing so, it addresses the training data distribution issue as well as overlap on various schema elements.

Also, we performed experiments with LSTM for slot tagging by masking scores of predicted class labels from predicted schema. LSTM is one of the most popular deep learning techniques for sequence tagging (Bahdanau et al., 2014; Dyer et al., 2015), but we observe that the LSTM results (*Const(LSTM)*) on our dataset are very similar to that of CRFs (*Const(CRF)*, as shown in Table 2. In the following experiments, we mostly focus on the *Const* version of CRFs for simplicity.

The main results of constrained decoding with different schema prediction methods are shown in Table 3. *Bin* approach trains k binary logistic regression classifier for each slot type. Each binary classifier

Constrained by		Query					Domain
	In-domain	Bin_{Fix}	Bin_{Min}	Mult_{Fix}	Mult_{Min}	GoldQ	PredD
Alarm	92.89	90.56	84.89	91.37	96.29	97.74	93.56
Calendar	90.03	87.6	80.03	89.17	91.86	92.08	88.57
Comm.	92.94	91.28	77.94	91.89	93.29	95.97	92.14
Entertain.	93.83	92.41	81.83	91.9	95.54	96.5	93.17
Events	85.84	82.71	74.84	84.23	88.26	89.57	85
Hotel	91.25	89.25	82.25	90.75	92.23	93.21	91.12
Media.	86.39	82.94	85.39	84.58	92.99	93.99	87.07
Mvtickets	91.75	87.86	72.75	86.16	91.67	92.8	91.06
Mystuff	87.92	83.09	83.92	85.12	90.36	91.86	84.88
Note	87.6	81.84	78.6	83.38	87.58	88.42	84.32
Ondevice	93.59	91.87	69.59	92.22	97.79	98.6	94.27
Orderfood	93.52	91.25	76.52	93.62	95.92	96.24	92.84
Places	91.75	89.82	79.75	87.59	94.27	96.8	89.55
Reminder	89.31	86.1	71.31	87.18	93.89	94.07	88.57
Reservations	92.68	89.57	85.68	91.29	94.28	96.28	93.56
Taxi	88.27	86.63	72.27	89.07	95.42	97.11	90.32
Weather	96.27	94.65	89.27	95.8	98.5	99.11	96.44
Average	90.93	88.20	79.23	89.14	93.55	94.73	90.38

Table 3: F1 scores for *Const* with various schema prediction methods across 17 personal assistant domains.

determines if a query has a specific slot or not, while *Mult* approaches use a single LSTM to predict a set of allowed schema for a query. Subscript Fix denotes that a fixed threshold (0.5) is used to make a decision of positive versus negative label, and Min denotes that the threshold is set to be the minimum of positive label thresholds, which hence gives the maximum recall rate. *GoldQ* denotes the decoding was constrained by true schema for a query and and *PredD* denotes the decoding was constrained by a predicted domain. Here *In-domain* also uses domain classifier.

In the preliminary experiments, we observed that there are significant performance improvements when performing constrained decoding given a gold standard schema of a query *(GoldQ)*. However, it is very difficult to get similar performance by the predicted schema. The main reason is because it does not guarantee recall. As you can see, all methods based on schema prediction except for $\textit{Mult}_{Min}$, fail to achieve any improvement compared to *In-domain* and *PredD*. So, we use the minimum probability of a ground-truth schema type per query to increase recall. Using predicted minimum boundary $\textit{Mult}_{Min}$ finally boost up performance up to 93.55%, huge relative error reduction of 33% over *In-domain* approach.

Unlike previous experiments, the experiments shown in Table 4 assume that the true domain and its schema are given. So, there are no domain classification error. $\textit{MULT}_{Min}$ removes the predicted schema elements that are outside of the true domain schema. *In-domain* yields 92.53% F1 score. $\textit{MULT}_{Min}$ boosts the performance to 93.99%.

To further compare multi-classification approach to binary approach, we show performance for multi-class labeling task in Table 5. Unlike slot tagging performance, $\textit{Mult}_{Fix}$ has the highest F1 score because of its precision. $\textit{Mult}_{Min}$ has high recall at the slight expense of precision. However, binary logistic regression (Bin_{Min}) fails to keep reasonable precision. This is because logistic regression models are over-fitted to each label, minimum boundary is very low and thus it causes a lot false positives.

For the last scenario shown in Table 6, we assume that we do not have training data for the test domains. The amount of test data is about 2k. The $\textit{Mult}_{Min}$ performs reasonably well, yielding 96.48% on average. Interestingly, for the *Bus* domain, we can get almost perfect tagging performance of 99.5%. Note that all tags in *Bus* and *Ferry* domains are fully covered by our single model, but the *ShopElectric*

	In-domain	MULT$_{Min}$
Alarm	95.53	96.23
Calendar	90.37	92.46
Comm.	93.08	94.29
Entertain.	94.48	95.84
Events	89.47	90.02
Hotel	93.16	93.68
Mediactrl	90.87	92.7
Mvtickets	92.5	92.98
Mystuff	88.76	89.82
Note	89.48	90.58
Ondevice	95.27	97.26
Orderfood	94.35	96
Places	93.18	95.19
Reminder	90.22	92.77
Reservations	93.34	95.01
Taxi	91.37	94.2
Weather	97.64	98.83
Average	92.53	93.99

Table 4: F1 score for *In-domain* and *MULT$_{Min}$* across domains for constrained with predicted multi labels given true domain schema.

	Bin$_{Fix}$			Bin$_{Min}$			Mult$_{Fix}$			Mult$_{Min}$		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Alarm	87.7	67.6	76.4	65.2	99.2	78.7	88.2	86.5	87.3	69.3	99.1	81.6
Calendar	83.7	83.4	83.5	72.8	99.2	86.1	90.1	80.7	85.1	70.5	99.7	82.4
Comm.	78.7	86.1	82.2	66.1	98.8	79.2	86.5	85	85.7	74.8	98.9	85.2
Entertain.	87.4	72.6	79.3	58.4	99.7	73.5	88.3	84.6	86.4	63.3	99.1	77.4
Events	74.8	75.6	75.2	62.2	99.1	83.5	84.3	85.9	85.1	70.8	98.8	82.6
Hotel	87.3	85.9	86.6	58.2	98.4	73.1	84.9	89.2	87	82.2	98.2	89.6
Mediactrl	91.9	68.6	78.5	69.9	99.8	82.1	86	92.4	89.1	79.8	99.5	88.6
Mvtickets	81.3	77	79.1	72.4	99.3	83.8	86.4	87.7	87.1	73.2	99.3	84.3
Mystuff	87.6	82.4	84.9	61.7	98.9	79.5	78.8	82.4	80.5	78.3	98.5	87.4
Note	84.7	77.7	81.1	64.5	98.7	77.8	92.3	87.5	89.8	67.3	98	80
Ondevice	87.2	84.7	85.9	72.3	99.3	83.4	89.2	85.5	87.3	75.7	98.6	85.9
Orderfood	85	70.2	76.9	69.1	99.4	81.4	92.4	82.5	87.2	82.5	99.1	90.2
Places	89.9	83.5	85.9	73.1	99.6	85.9	87.2	70.7	75.8	68.9	99	75.7
Reminder	86.5	75.9	80.9	71.9	98.5	84.7	91.4	84.8	88	83.5	99.2	90.4
Reservations	90.7	85.2	87.9	51.1	99.3	67.5	90.7	80.5	85.3	79.9	99.5	88.6
Taxi	87.7	82	84.8	71.9	99.3	84.6	94.4	88.5	91.3	78.5	98.9	87.7
Weather	85	72.2	78.1	52.6	99.4	68.8	91.6	88.2	89.9	65.9	99.5	79.3
	85.7	78.3	81.6	65.5	99.2	79.6	88.4	84.9	86.4	74.4	99	84.5

Table 5: Multi-labeling task performance for schema prediction methods.

	Ferry	Bus	ShopElectric.	AVG.
Mult$_{Min}$	96.86	99.5	93.08	96.48

Table 6: F1 scores for *Mult$_{Min}$* across new domains which do not have domain specific training data.

domain is partially covered.

5 Conclusion

In this paper, we proposed a solution for scaling domains and experiences potentially to a large number of use cases by reusing existing data labeled for different domains and applications. The single slot tagging coupled with schema prediction and constrained decoding achieves competitive and often superior performance over the conventional model trained in per domain fashion. This approach enables creation of new virtual domains through any combination of slot types covered in the single slot tagger schema, reducing the need to collect and annotate the same slot types multiple times for different domains.

Acknowledgements

We thank Minjoon Seo and anonymous reviewers for their constructive feedback.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics.

John Blitzer, Sham Kakade, and Dean P Foster. 2011. Domain adaptation with coupled subspaces. In *AISTATS*, pages 173–181.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Hal Daumé III. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Ali El-Kahky, Derek Liu, Ruhi Sarikaya, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2014. Extending domain coverage of language understanding systems via intent transfer between domains using knowledge graphs and search query click logs. IEEE, Proceedings of the ICASSP.

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

Young-Bum Kim, Minwoo Jeong, Karl Stratos, and Ruhi Sarikaya. 2015a. Weakly supervised slot tagging with partially labeled sequences from web search click logs. In *Proceedings of the NAACL*. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, Xiaohu Liu, and Ruhi Sarikaya. 2015b. Compact lexicon selection with spectral methods. In *Proc. of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2015c. Pre-training of hidden-unit crfs. In *Proc. of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 192–198.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015d. New transfer learning techniques for disparate label sets. *ACL. Association for Computational Linguistics*.

Young-Bum Kim, Alexandre Rochette, and Ruhi Sarikaya. 2016a. Natural language model re-usability for scaling to different domains. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016b. Scalable semi-supervised query classification using matrix sketching. In *Proc. of Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

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

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Ruhi Sarikaya. 2015. *The technology powering personal digital assistants*. Keynote at Interspeech, Dresden, Germany.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12.

Gokhan Tur. 2006. Multitask learning for spoken language understanding. In *In Proceedings of the ICASSP*, Toulouse, France.

Sub-Word Similarity based Search for Embeddings: Inducing Rare-Word Embeddings for Word Similarity Tasks and Language Modelling

Mittul Singh[1,2,3] Clayton Greenberg[1,2,3] Youssef Oualil[1,3] Dietrich Klakow[1,2,3] *
[1]Spoken Language Systems (LSV)
[2]Saarbrücken Graduate School of Computer Science, Saarland Informatics Campus
[3]Collaborative Research Center on Information Density and Linguistic Encoding
Saarland University, Saarbrücken, Germany
{firstname.lastname}@lsv.uni-saarland.de

Abstract

Training good word embeddings requires large amounts of data. Out-of-vocabulary words will still be encountered at test-time, leaving these words without embeddings. To overcome this lack of embeddings for rare words, existing methods leverage morphological features to generate embeddings. While the existing methods use computationally-intensive rule-based (Soricut and Och, 2015) or tool-based (Botha and Blunsom, 2014) morphological analysis to generate embeddings, our system applies a computationally-simpler sub-word search on words that have existing embeddings. Embeddings of the sub-word search results are then combined using string similarity functions to generate rare word embeddings. We augmented pre-trained word embeddings with these novel embeddings and evaluated on a rare word similarity task, obtaining up to 3 times improvement in correlation over the original set of embeddings. Applying our technique to embeddings trained on larger datasets led to on-par performance with the existing state-of-the-art for this task. Additionally, while analysing augmented embeddings in a log-bilinear language model, we observed up to 50% reduction in rare word perplexity in comparison to other more complex language models.

1 Introduction

Word embeddings have been successfully applied to many NLP tasks (Collobert and Weston, 2008; Collobert, 2011; Socher et al., 2011; Socher et al., 2012; Hermann and Blunsom, 2014; Bengio and Heigold, 2014; Yang et al., 2015), and these systems often achieved state-of-the-art performance. This success has been ascribed to embeddings' ability to capture regularities traditionally represented in core NLP features. Most of these embeddings were trained on large amounts of data, allowing them to have good coverage of the relevant vocabularies. However, embeddings often still cannot satisfactorily represent *rare words*, i.e. words with few occurrences in training data.

To generate useful embeddings for words too rare for standard methods to handle, Luong et al. (2013) and Botha and Blunsom (2014) leveraged the segmentation tool, Morfessor (Creutz and Lagus, 2005), while Cotterell et al. (2016) used morphological lexica to generate rare-word embeddings. In general, these methods added resource-based knowledge to their systems in order to form word vector representations, showing impressive performance gains over methods which did not address the rare words problem.

In contrast, Soricut and Och (2015) applied an automatic method to induce morphological rules and transformations as vectors in the same embedding space. More specifically, they exploited automatically-learned prefix- and suffix-based rules using the frequency of such transformations in the data and induced a morphological relationship-based word graph. Then, they searched over this graph for rules that best infer the morphology of the rare words. The embeddings were then estimated using these rare-word explaining rules. In this method, creating and tuning this morphological graph could lead to a high initial cost.

* This work was supported by the Cluster of Excellence for Multimodal Computing and Interaction, the German Research Foundation (DFG) as part of SFB 1102, the EU FP7 Metalogue project (grant agreement number: 611073) and the EU Malorca project (grant agreement number: 698824).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2061–2070, Osaka, Japan, December 11-17 2016.

Language	V	RW	#ENF	Coverage
German	36602	15715	13103	99.9
Tagalog	22492	10568	8407	98.1
Turkish	24840	13624	9555	99.0
Vietnamese	6423	1332	305	69.1

Table 1: This table reports various statistics for different language datasets used for language modelling. The last column shows the coverage of our method in percentage.

Task	V	#ENF	Coverage
Rare Word (Luong et al., 2013)	2951	1073	100
Gur65 (Gurevych, 2005)	49	4	100
Rare Word + Google News	2951	173	100

Table 2: This table reports various statistics of a few language word similarity datasets used in our experiments. The last column shows the coverage of our method in percentage.

In order to overcome this cost and still be able to automatically induce rare word representations, we propose a sub-word similarity-based search. This technique maps a rare word to a set of its morphologically-similar words and combines the embeddings of these similar words to generate the rare word's representation (further discussed in Section 2). These generated embeddings can then be combined with existing word embeddings to be applied in various tasks.

In Section 3, we evaluate our embeddings on word similarity tasks. For further evaluation, in Section 4, we instantiate a log-bilinear language model (Mnih and Hinton, 2007) with our word embeddings and analyse their perplexity performance on rare words over various language modelling corpora. Finally, we summarise our findings in Section 5.

2 Rare-Word Embeddings

Rare words form a large part of a language's vocabulary. This is illustrated in Table 1, which reports the vocabulary size and number of rare words (RW) with zero (out-of-vocabulary words) or one training set occurrence for our corpora. As shown in this table, rare words constitute 10%-50% of the vocabulary. Further, it is widely known that in English, roughly half of all tokens in a given corpus occur only once. Thus, it is essential to handle rare words properly to obtain good performance.

In the context of word embeddings-related tasks, training good word embeddings can incur huge computational costs (Al-Rfou et al., 2013). So, in this work, we focus on augmenting readily available embeddings rather than creating new ones from scratch. To increase the availability of resources for many languages, Al-Rfou et al. (2013) released[1] pre-trained word embeddings for more than one hundred languages. These pre-trained word embeddings, namely *Polyglot*, were constructed by applying the method outlined in Bengio et al. (2009) on Wikipedia text, which vary in size from millions of tokens to a few billion tokens.

Among other available pre-trained word embeddings, Google released `word2vec` (Mikolov et al., 2013)-based embeddings[2] trained on their English News dataset (about 100 billion tokens). In our experiments, we applied both of these embeddings sets to jump start generating the rare word embeddings for different languages.

2.1 Inducing Rare-Word Embeddings

Statistics about the various language modelling corpora and word similarity tasks that we used in our experiments are shown in Table 1 and Table 2. In these tables, along with the vocabulary size and number of rare words, we also report the number of words for which the embeddings were not found (ENF = Embedding Not Found) in the pre-trained embedding sets. For most of the language and pre-trained embedding pairs, number of ENFs formed a large share of the vocabulary for word similarity tasks and of rare-word set size for language modelling tasks. Hence, we estimated the missing word embeddings before using them in our tasks.

We first provide a high level description of the steps of our method to induce the word embeddings for these missing rare words, followed by detailed description of each step. For a given set of pre-trained embeddings with a finite vocabulary V_E applied to a task with vocabulary V_T and a finite set of given rare words $RW = \{w | w \notin V_E \& \ w \in V_T\}$, we apply the following steps:

[1] `https://sites.google.com/site/rmyeid/projects/polyglot`
[2] `https://code.google.com/archive/p/word2vec/`

1. Map every word $w \in V_E$ to its sub-word features
2. Index $w \in V_T$ using its sub-word features
3. Search the index for matches of $w' \in RW$
4. For every $w' \in RW$, combine matched words' embeddings to generate its embedding

Step 1: Map words to sub-words
Although a word may be rare, substrings of that word are, in general, less rare. Hence, we start by breaking down each word $w \in V$ into its constituent N-sized sub-word units: $D_N(w)$. For example, given the sub-word size $N = 3$:

$$D_N(language) = \{lan, ang, ngu, gua, uag, age\}$$

In our experiments, we worked with value of $N = 3$. However, it remains to be seen how using differently sized sub-word units or even morphemes affects the performance of this method. Note that our procedure does not formally require that sub-word units be of equal length, so linguistically-sensible morphemes may be used if the resource is available for that language.

Step 2: Index word using its sub-words
Pre-trained sets of embeddings can cover large numbers of words already (for example, *Polyglot* embeddings have 100K words in their vocabulary). So, performing substring searches and comparisons can become quite computationally expensive. To speed up the search for sub-word units, we create an inverted index on words. For each $w \in V$, we treat $D_N(w)$ as a document and feed it into a search engine-based indexer. In this work, we used Lucene[3] (McCandless et al., 2010) to index the words.

Step 3: Search for matches of a rare word
Next, we break down the rare word $w' \notin V$ into its sub-word units ($D_N(w')$) and search for $D_N(w')$ using the index. We restrict the search results set to the top K results, denoted by $R^K(w')$. $R^K(w')$ contains the words having similar sub-word units as w', hence, containing words which are sub-word similar to w'. In our experiments, we fixed $K = 10$.

Step 4: Generating rare-word embeddings
To estimate the word embedding of $w' \in RW$, we compute the weighted average of embeddings (v) of the rare-word matches. For this weighted average, we employ a string similarity function S, such that

$$v_{w'} = \sum_{w:D_N(w) \in R^K(w')} S(w', w) \times v_w$$

The above method particularly hinges on the third step, where we utilise sub-word similarity of morphologically similar words to search for rare word alternatives, leading to embedding combination in the fourth step. Hence, we refer to the above technique as **S**ub-**Word** **S**imilarity based **S**earch (**SWordSS**: pronounced swordz). The SWordSS embeddings ($\{v_{w'} : w' \in RW\}$) are used along with $\{v_w : w \in V\}$ to perform rare word-related tasks.

In the fourth step, we apply different string similarity functions (S), described in the list below, to average different embeddings of matches from the third step. These different similarity functions help provide a more morphologically-sensible scoring of matches and eventually are used to weight the inputs of the final rare word embeddings.

- Jaccard Index, Jaccard (1912) computes the size of the character intersection over the size of the character union. Therefore, order of characters is not considered by this metric. Frequent characters such as vowels lead to uninteresting intersections, and short words could possibly suffer from an unfair floor.

[3] `https://lucene.apache.org/`

- Jaro similarity, Jaro (1989) considers the number of matching characters in corresponding positions and the number of transpositions detected. So, order of characters does matter for this metric. Insertions and deletions are treated similarly, and the frequency and length effects from Jaccard could also affect this metric.

- Most frequent K Characters similarity, Seker et al. (2014) considers the counts of the top K characters in each string. Thus, if the "root morphemes" are long enough to create nontrivial count statistics, this metric may, too, favor a more linguistic similarity, but as before, shorter strings could have unwanted effects.

- Subsequence Kernels, Lodhi et al. (2002) create automatically-generated features based on sequences of characters within the strings to be compared. Therefore, those sequences that do not cross morpheme boundaries could be especially helpful for estimating morphological similarity.

- Tversky coefficient, Tversky (1977) breaks down the union in the Jaccard index, allowing different weights for the denominator intersection, those characters that only appear in the first string, and those characters that only appear in the second string. These metaparameters allow the metric some flexibility that the others do not.

In our experiments on rare word-related tasks, we mostly observed that using SWordSS led to high coverage rates, also presented in Table 1 and Table 2. We note that whenever words w' resulted in zero matches in our experiments, they were either removed completely (in case of word similarity tasks) or substituted with random vectors (in case of language modelling tasks, Section 4).

3 Word Similarity Task

To test the efficacy of SWordSS embeddings, we evaluated them on two standard word similarity tasks. In such tasks, the correlation between the human annotator ratings of word pairs and the scores generated using embeddings was calculated. A good set of embeddings would achieve a high correlation.

Specifically, we evaluated the SWordSS embeddings on Luong et al. (2013)'s English Rare Words dataset with 2034 word pairs (Luong2034) and also evaluated these embeddings on a German word similarity task (Gurevych, 2005) with 65 word pairs (Gur65).

3.1 Experimental Setup

For the German word similarity task, we used only *Polyglot* word embeddings, which are 64-dimensional vectors. For English along with *Polyglot* word embeddings, we used the Google News word2vec embeddings, which are 300-dimensional vectors.

As a baseline, we used the existing pre-trained word embeddings, which are compared to their augmented SWordSS versions. While augmenting the pre-trained set with the SWordSS embeddings, we also explored various string similarity functions to be used in the fourth step (Section 2.1), namely, Jaccard Index (SWordSS$_{ji}$), Jaro similarity (SWordSS$_{jaro}$), Most Frequent K Characters similarity (SWordSS$_{mfk}$), Subsequence Kernels (SWordSS$_{ssk}$) and Tversky Coefficient (SWordSS$_{tc}$).

To evaluate the effect of these string similarity functions, we also implemented a constant similarity function ($S(w, w') = 1$, where w and w' are words) used in the fourth step, denoting the corresponding embeddings by SWordSS$_1$. Finally, we also compared the SWordSS embeddings to SO2015 (Soricut and Och, 2015), which also applies morphological analysis to generate missing word embeddings quite similar to SWordSS embeddings.

3.2 Results

Using SWordSS embeddings definitely increased the correlation with humans in comparison to the original on the Gur65 task (shown in Table 3), though the different string similarity functions except the constant function (SWordSS$_1$) led to correlations in a very close range, showing that particularly for German, different similarity functions behave very similarly. Henceforth, we only report the best correlation coefficient after applying these functions.

Word Vectors	Gur65
Polyglot	28.5
Polyglot+SWordSS$_{ji}$	37.5
Polyglot+SWordSS$_{jaro}$	37.1
Polyglot+SWordSS$_{mfk}$	37.2
Polyglot+SWordSS$_{ssk}$	36.9
Polyglot+SWordSS$_{tc}$	**37.6**
Polyglot+SWordSS$_{1}$	35.8

Table 3: Spearman's rank correlation (%) based evaluation of various string similarity functions used to generate augmented word vectors for the German word similarity task (Gur65)

Task	Luong2034	
Word Vectors	*Polyglot*	*Google News*
SO2015 w/o morph	-	44.7
SO2015 w/ morph	-	**52.0**
w/o SWordSS	9.7	45.3
w/ SWordSS$_{1}$	28.9	51.3
w/ SWordSS$_{sim}$	30.4	51.4

Table 4: Spearman's rank correlation (%) based evaluation of techniques with and without morphological features used to generate representations for the word similarity task.

Next, we compared SWordSS versions of *Polyglot* embeddings and Google News Embeddings on the Luong2034 task. When the SWordSS versions were compared to the original (labelled w/o SWordSS) it led to a higher correlation, as shown in Table 4. However, for each set of embeddings, the difference between SWordSS$_1$ and SWordSS$_{sim}$ remained small. The correlations for the SWordSS version of *Polyglot* were still lower than the correlation rates reported by SO2015. This was due to the difference in initial quality of embeddings used by each method. As *Polyglot* embeddings trained on a lesser amount of data than SO2015, they were easily outperformed.

In Table 4, we addressed this lower performance issue by replicating our experiment using Google News `word2vec` embeddings to jump start the SWordSS versions for the Luong2034 task. Using these embeddings, trained on a larger dataset than used by *Polyglot*, led to SWordSS versions having on-par results with the SO2015 results for the Luong2034 task.

Overall the SWordSS technique was able to drastically improve pre-trained embeddings performance on the above word similarity tasks. Even though SWordSS-augmented Google News embeddings did not significantly outperform SO2015, this method provides a simpler sub-word search based alternative to the graph search over morphological relationships performed by SO2015. Furthermore, by applying sub-word search in the third step as shown in Section 2.1, SWordSS overcomes the need for creating and tuning the graph of morphological relationships as required by SO2015.

4 Word Embeddings in Language Models

Training language models (LMs) using an expanded vocabulary (having more word types than contained in the training corpus) requires assigning probabilities to words which are not present in the training set. Traditionally, these rare words are assigned a default value of probability in conventional N-gram and long short term memory (LSTM)-based reccurrent neural network LMs (Sundermeyer et al., 2012). This is usually not beneficial for spoken term detection and automatic speech recognition systems made for low resourced languages, since presence of rare words in speech queries is high (Logan et al., 1996; Logan et al., 2005).

To avoid this misrepresentation of rare words, we apply SWordSS embeddings in a language modelling framework. Specifically, a log-bilinear language model (LBL) (Mnih and Hinton, 2007). In our experiments, when the SWordSS embeddings were used to initialise an LSTM's input layer, the system obtained the same perplexity values as the LSTM initialised with random embeddings. This observation suggests that the LBL framework is better suited than LSTMs for this naïve way of initialising neural language models with SWordSS embeddings and improving perplexity on rare words.

LBL predicts the next word vector $p \in \mathbb{R}^d$, given a context of $n-1$ words, as a transformed sum of context word vectors $q_j \in \mathbb{R}^d$, as:

$$p = \sum_{j=1}^{n-1} q_j C j$$

where $C_j \in \mathbb{R}^{d \times d}$ are position-specific transformation matrices. p is compared with the next word w's representation r_w. This comparison is performed using the vector dot product and then is used in a

softmax function to obtain the probability of the next word as follows:

$$p(w_i|w_{i-n+1}^{i-1}) = \frac{\exp(p \cdot r_w + b_w)}{\sum_{v \in V} \exp(p \cdot r_v + b_v)}$$

where b is the bias term encoding the prior probability of word type w.

First, Q the collection of context word vectors (q_j) and R the collection next word representations (r_w) are initialised with the pre-trained word embeddings. Thereafter, we train the LBL using stochastic gradient descent.

Previously, extensions to class based and factor based formulations have provided impressive improvements over regular N-gram LMs for morphological languages (Botha and Blunsom, 2014). But, these LMs do not provide straightforward ways of incorporating pre-trained word embeddings, so we use the original LBL because of the ease with which it incorporates pre-trained embeddings in its formulation.

4.1 Data

To evaluate the SWordSS embeddings for language modelling, we used the Europarl-v7 corpus of German (de) language as processed by Botha and Blunsom (2014). We also performed language modelling experiments with the SWordSS embeddings on Tagalog (tl), Turkish (tr) and Vietnamese (vi) corpora, which include transcriptions of phone conversations collected under the IARPA Babel Program language collection releases babel106b-v0.2f, babel105-v0.5 and babel107b-v0.7 respectively.

The German corpus was processed to have no out-of-vocabulary words (OOVs), however, it still had a lot of low frequency words (see Table 2). Contrastingly, the Babel corpora have OOVs as well as other low frequency words.

The Babel corpora were provided with training and development sets. We divided the existing development set into two halves to use one as the test set and the other half as the new development set. The statistics on these corpora are summarised in Table 5.

Statistics	de	tl	tr	vi
Train	1000K	585K	239K	985K
Dev	74K	30K	5K	65K
Test	73K	31K	6K	60K
Voc Size	37K	22K	25K	6K

Table 5: Statistical summary of corpora used for the language modelling experiments. Information corresponding to a language is presented in a column.

In Tables 1 & 2, we had shown that even though a lot of rare-word embeddings are missing from the pre-trained set, SWordSS was able to generate and obtain high coverage rates for such words, giving this method added benefit in the context of rare words.

4.2 Experimental Setup

Before evaluating the SWordSS embeddings for predicting rare words, we used all the OOVs to expand the corresponding vocabulary. SWordSS embeddings for all the words in the expanded vocabulary were used to initialise LBL framework as described in Section 4. A bigram version of this LBL (LBL2$_{SWordSS}$) was further trained on language corpora before being evaluated.

We compare our LBL2$_{SWordSS}$ model with the conventional Modified-Kneser-Ney five-gram LM (MKN5) (Kneser and Ney, 1995; Chen and Goodman, 1996) and also with the bigram (LBL2) based log-bilinear LM. As a more powerful baseline, we also trained an LSTM based RNN LM to compare with LBL2$_{SWordSS}$. Moreover, we compare the LBL2$_{SWordSS}$, with a character aware language model (Kim et al., 2015), denoted as CCNN-LSTM. The CCNN-LSTMs were chosen for comparison because of their ability to use character-based features to implicitly handle OOVs and rare words. For training each of these LMs, we used the expanded vocabulary as used by LBL2$_{SWordSS}$. In training neural network-based language models, we restricted the number of parameters to have a similar number of parameters as LBL2$_{SWordSS}$.

4.3 Perplexity Experiments

We compare the language models described in Section 4.2 using perplexity values calculated on test sets of different languages, shown in the Table 6.

Language Model	German		Tagalog		Turkish		Vietnamese	
	PPL	RW1PPL	PPL	RW1PPL	PPL	RW1PPL	PPL	RW1PPL
MKN5	364.2	559K	162.6	420K	478.9	139K	120.8	174K
LBL2	391.1	404K	171.4	204K	649	**94K**	137.6	**100K**
LSTM[4]	323.1	596K	134.7	343K	489.8	110K	**102.1**	457K
CCNN-LSTM	**315.7**	636K	**117.4**	354K	**408.7**	168K	182.7	516K
LBL2$_{SWordSS}$	369.4	**260K**	167.2	**167K**	513.2	110K	136.4	143K
#PAR	4.7 M		2.9 M		3.2 M		0.8 M	

Table 6: Perplexities on test set (PPL), RW1 perplexities (RW1PPL) in thousands and number of parameters (#PAR) for LBL and LSTM LMs in millions, presented on four language corpora

As shown in Table 6, LBL2$_{SWordSS}$ was able to outperform the conventional LBL2 comfortably on all the corpora except Vietnamese. For Vietnamese, LBL2$_{SWordSS}$ and LBL2 performed comparably. Due to SWordSS' low coverage of Vietnamese vocabulary, initialising LBL2 with SWordSS embedding led to only a marginal performance gain.

Overall in terms of test set perplexity, CCNN-LSTM outperformed LBL2$_{SWordSS}$ comfortably on most language corpora. However, on Vietnamese (in which characters represent meaning units rather than sounds) CCNN-LSTM suffered and the LSTM outperformed the other language models. In comparison to LSTM and CCNN-LSTM, LBL2$_{SWordSS}$'s lower performance on test data was expected as the former are more non-linearly complex language models.

However, for tasks like spoken term detection, having low perplexities on most frequent set of words is not good enough and hence, we compare LMs on the perplexity of a rare-word based test set. To perform this comparison, we computed perplexity only on rare words (RW1PPL), i.e. with training-set frequency of one, present in the test set. As shown in Table 6, we observe that LBL2$_{SWordSS}$ performed better than the LSTM-based LMs across various languages in terms of RW1PPL.

We note that CCNN-LSTM model cannot include SWordSS embeddings easily. Hence, they are not directly comparable to LBL2$_{SWordSS}$, as the latter has more information at its disposal.

4.4 Performance on OOVs and Rare Words

To further compare the performance of the aforementioned language models on rare words, we analyse perplexities of such words (RWPPL) in the test set as a variation of the frequency classes of these words in the training set. This variation is displayed in Figure 1.

For OOVs (rare words with zero training-set frequency), LBL2$_{SWordSS}$ outperformed the other language models built with similar number of parameters, on the Tagalog and Turkish corpora. In these cases, LBL2$_{SWordSS}$ reduced rare-word perplexities by a factor of two over the character-feature rich CCNN-LSTM, whose design allows it to implicitly handle rare words.

Even for rare words with training set frequency up to one, LBL2$_{SWordSS}$ reduced perplexity up to a factor of 2.5 times with respect to CCNN-LSTM, on the German, Tagalog and Turkish corpora. Interestingly on these particular language corpora, Figure 1 shows that LBL also performed better than both the LSTM-based LMs in modelling OOV and rare words of frequency up to ten.

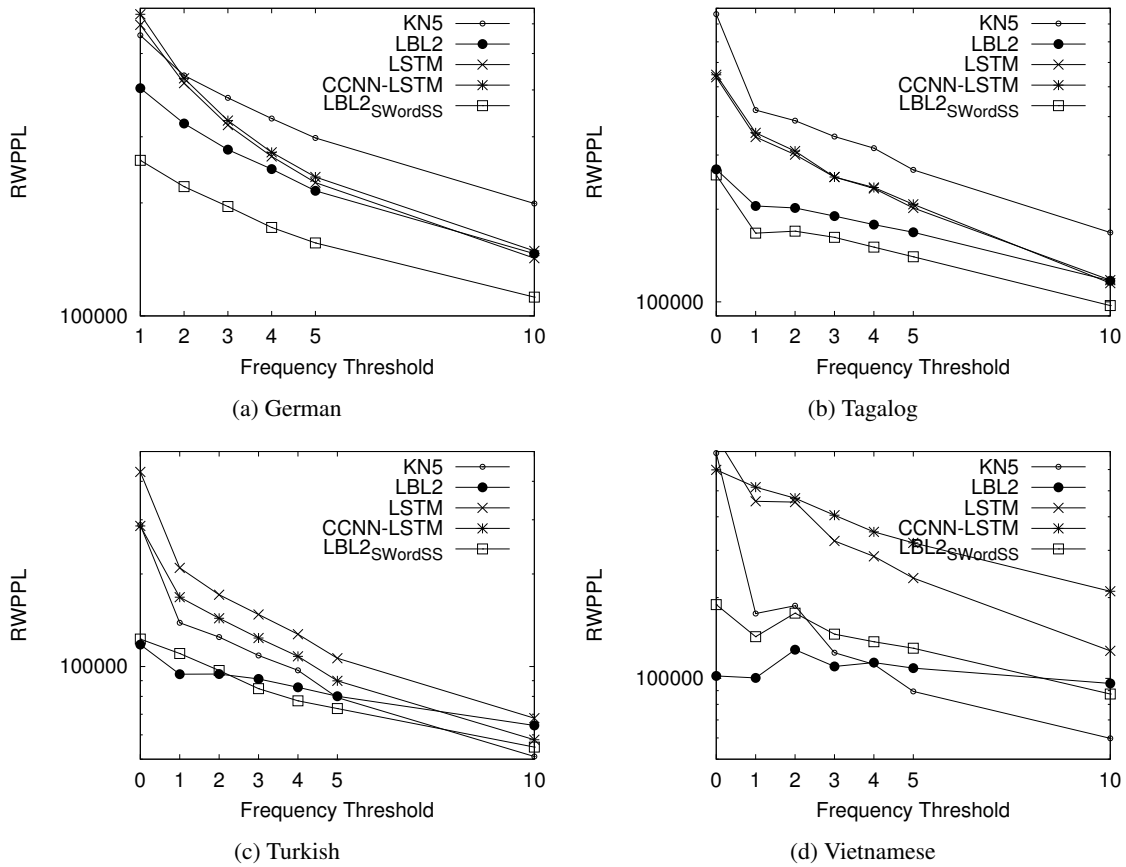
For Vietnamese, LBL alone was able to improve OOV and RW1 words over the other LMs. We attribute this to lower coverage of Vietnamese rare words by SWordSS than for other languages. Instead adding SWordSS embeddings harmed the prediction of OOV and RW1 words.

These perplexity improvements stared to wane when higher frequency words were included into the rare word set, across the different languages. Nevertheless, for languages with rich morphology, initialising LBL with SWordSS embeddings reduced perplexities on rare words.

5 Conclusion

In this paper, we introduced SWordSS, a novel sub-word similarity based search for generating rare word embeddings. It leverages the sub-word similarity in morphologically rich languages to search for close

[4]when initialised with SWordSS embeddings it obtained the same perplexity values



(a) German (b) Tagalog

(c) Turkish (d) Vietnamese

Figure 1: Variation of rare-word perplexity versus threshold on frequency of training-set words on German, Tagalog, Turkish and Vietnamese corpora

matches of a rare word, and then combines these close matches to estimate the embedding of a rare word.

Even though SWordSS is an unsupervised approach like Soricut and Och (2015), it differs from latter in the way it utilises the morphological information. The latter automatically induces morphological rules and transformations to build a morphological word graph. This graph is then tuned and used to induce embedding of a rare word. Instead, SWordSS replaces the overhead of induction of rules and creation of graph by searching a sub-word inverted index to find rare-word matches and combining their embeddings to estimate rare-word embedding.

To test the SWordSS technique, we augmented pre-trained embeddings and then evaluated them on word similarity tasks. The augmented embeddings outperformed the initial set of embeddings drastically. However, it lagged behind the state-of-the-art performance of Soricut and Och (2015). But, by employing embeddings trained on larger corpora, SWordSS was able to perform comparably on a rare-word task.

We also investigated the effects of using SWordSS augmented embeddings for modelling rare words. To perform this experiment, we trained LBL$_{SWordSS}$ LM and compared it with language models like the character aware LM, LSTM-based RNN LM restricted to similar size. On almost all datasets, the character aware LM outperformed the other LMs with respect to perplexity on complete test sets. But on rare words, SWordSS showed up to 50 % reduced perplexity values in comparison to other LMs. Hence, SWordSS embeddings contributed substantially in modelling rare-word tasks.

In future work, we plan to incorporate SWordSS embeddings into more complex LMs than LBL and further analyse the different string similarity functions used in SWordSS's formulation.

Acknowledgments

We would like to thank anonymous reviewers for their comments, which helped improve this paper. We are also immensely grateful to Rose Hoberman for her comments on an earlier version of this manuscript.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August. Association for Computational Linguistics.

Samy Bengio and Georg Heigold. 2014. Word embeddings for speech recognition. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association*, pages 1053–1057, Singapore, September.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA. ACM.

Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. *CoRR*, abs/1405.4273.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, California, USA, June. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *International Conference on Artificial Intelligence and Statistics*.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1651–1660, Berlin, Germany, August. Association for Computational Linguistics.

M. Creutz and K. Lagus. 2005. Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora Using Morfessor 1.0. Technical report, Helsinki University of Technology.

Iryna Gurevych, 2005. *Natural Language Processing – IJCNLP 2005: Second International Joint Conference, Jeju Island, Korea, October 11-13, 2005. Proceedings*, chapter Using the Structure of a Conceptual Network in Computing Semantic Relatedness, pages 767–778. Springer Berlin Heidelberg, Berlin, Heidelberg.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. *CoRR*, abs/1404.4641.

Paul Jaccard. 1912. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50, February.

Matthew A. Jaro. 1989. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models. *CoRR*, abs/1508.06615.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444.

Beth Logan, Pedro Moreno, Jean-Manuel Van Thong, et al. 1996. An experimental study of an audio indexing system for the web. In *Proceedings of the 4th International Conference of Spoken Language Processing*. Citeseer.

B. Logan, J. M. Van Thong, and P. J. Moreno. 2005. Approaches to reduce the effects of oov queries on indexed spoken audio. *IEEE Transactions on Multimedia*, 7(5):899–906, Oct.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113.

Michael McCandless, Erik Hatcher, and Otis Gospodnetic. 2010. *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*. Manning Publications Co., Greenwich, CT, USA.

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

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 641–648, New York, NY, USA. ACM.

Sadi Evren Seker, Oguz Altun, Ugur Ayan, and Cihan Mert. 2014. A novel string distance function based on most frequent K characters. *CoRR*, abs/1401.6596.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D. Manning, and Andrew Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea, July. Association for Computational Linguistics.

Radu Soricut and Franz Och. 2015. Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1627–1637, Denver, Colorado, May–June. Association for Computational Linguistics.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *INTERSPEECH*, pages 194–197.

Amos Tversky. 1977. Features of similarity. *Psychological review*, 84(4):327.

Yiming Yang, Hanxiao Liu, Jaime Carbonell, and Wanli Ma. 2015. Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 159–168, New York, NY, USA. ACM.

Semi-automatic Detection of Cross-lingual Marketing Blunders based on Pragmatic Label Propagation in Wiktionary

Christian M. Meyer and **Judith Eckle-Kohler** and **Iryna Gurevych**
Ubiquitous Knowledge Processing (UKP) Lab
and Research Training Group AIPHES
Technische Universität Darmstadt, Germany
https://www.ukp.tu-darmstadt.de

Abstract

We introduce the task of detecting cross-lingual marketing blunders, which occur if a trade name resembles an inappropriate or negatively connotated word in a target language. To this end, we suggest a formal task definition and a semi-automatic method based the propagation of pragmatic labels from Wiktionary across sense-disambiguated translations. Our final tool assists users by providing clues for problematic names in any language, which we simulate in two experiments on detecting previously occurred marketing blunders and identifying relevant clues for established international brands. We conclude the paper with a suggested research roadmap for this new task. To initiate further research, we publish our online demo along with the source code and data at http://uby.ukp.informatik.tu-darmstadt.de/blunder/.

1 Introduction

Large companies increasingly advertise and sell their products in international markets. Developing a marketing campaign for a new country requires tremendous translation efforts in order to bridge language and cultural boundaries. A particular problem often occurs if an established product, brand, or company name is introduced to a new, foreign market without being adapted to local habits and language use. This may yield offensive, embarrassing, or (at best) funny results causing excessive remedial cost and maybe even the withdrawal of a product from the new market. Such a *marketing blunder* can have multiple different reasons. A commercial for a men's fragrance showing a man with his dog failed, for instance, in Islamic countries where dogs are considered unclean. Dalgic and Heijblom (1996) distinguish possible reasons, including political, ethical, and legal issues, different traditions, inappropriate language, etc.

In this work, we focus on *cross-lingual marketing blunders*, which are a result of using inappropriate or negatively connotated expressions or translations for naming a company, brand, or product. One example for this is using the word *mist*, which usually describes fabulous, enigmatic, lightweight, or mystic things in English. A British car manufacturer, for example, chose the word to advertise their *Silver Mist* model. In German, the false friend *Mist* means, however, dung or manure, and it is a frequently used slang expression to describe a futile, cheap, or broken thing, nonsense, or an annoying, tedious situation. This pejorative meaning has caused the car manufacturer to rename its product (Room, 1982; Felser, 2010). A more recent example is the announcement of the 7th edition of a smartphone in Asia. The company's original English slogan "This is 7" has been changed for the Hong Kong market, as the pronunciation of the numeral seven (jyutping: *cat1*) is very similar to a vulgar expression for the male genitals (*cat6*), which would cause funny reactions when combined with "This is" on a product's advertisement poster.[1]

Spotting a marketing blunder can be very time-consuming and expensive for companies, especially if they do not operate local branches in all their target countries. With the emergence of the world wide web, a myriad of start-ups and small companies is struggling with this issue when planning an international online shop. They face two major problems:

(1) The absence of large-scale resources yielding *clues* for potential marketing blunders. Since many blunders are caused by false friends used in colloquial speech, multilingual dictionaries covering the

[1] See for example http://qz.com/777628/ (September 9, 2016)

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2071–2081, Osaka, Japan, December 11-17 2016.

standard language are of limited help. Although there are specialized monolingual slang dictionaries such as the *McGraw-Hill's American Slang Dictionary* (Spears, 2007) for the U.S., it is very challenging to keep these dictionaries up-to-date and to provide them for a large number of languages.

(2) The absence of tools that assist the process of identifying the *relevant clues* from these resources. Obviously, not every word that exists in a target language is problematic for marketing a product. The English word *fog* is, for instance, a false friend of the Hungarian *fog* (English: *tooth*). Neither meaning has a negative connotation per se that would impede the use of *fog* in a successful marketing campaign within those countries. A tool assisting the detection of marketing blunders thus needs to separate relevant clues from irrelevant ones in order to reduce the manual effort.

In the present paper, we propose a novel method and tool for assisting copywriters and sales promoters with the detection of cross-lingual marketing blunders. Our method is primarily based on disambiguated translations and pragmatic labels extracted from Wiktionary (`http://www.wiktionary.org`), for which we create a large inter-lingual index and a retrieval process. We evaluate our approach in two experiments: (1) detecting previously occurred marketing blunders and (2) finding evidence for potential blunders in established brand names. The detection of cross-lingual marketing blunders is a new task in natural language processing and, to the best of our knowledge, there are yet no existing tools that assist copywriters in avoiding such blunders. This is why we aim at introducing a formal task definition, a first, freely available dataset, and a novel knowledge-based method to initiate further research in this direction. Based on our results and error analysis, we lay out a research agenda for this new task. Apart from detecting marketing blunders in trade names, we believe that this research strand is enabling for many other tasks, including the identification of problematic product slogans, acronyms (e.g., of scientific proposals), and names of persons, institutions, and projects that should not be misinterpreted in foreign languages.

2 Related Work

Marketing blunders are yet mostly discussed in management and marketing research. Ricks (2006) reports a large number of previously occurred blunders in international business, including a separate chapter on product and company names. Knight (1995) and Dalgic and Heijblom (1996) discuss a few number of cases in detail. These works aim at finding new management strategies for inter-cultural marketing (cf. Jallat and Kimmel, 2002), rather than providing actual assistance and tools for copywriters.

In another strand of research, linguists have studied the properties of the language used in advertising, including teasers, slogans, and names. Cook (1992) and Janich (2013) give comprehensive introductions to the linguistic analysis of marketing language. While these works are mostly concerned with the question of how positive connotations and rhetorical figures enhance the value of a product, they also touch on the issues of cross-cultural and cross-lingual marketing communication. However, none of these works describes specific properties or methods to detect potential naming blunders. In addition to that, there are dictionaries on slang and pejorative expressions like the ones by Spears (2007) or Küpper (1984), as well as specialized dictionaries on trade names, such as Room (1982). Slang dictionaries are limited in their up-to-dateness (as indicated by the old publication years), word coverage, and range of available languages. Specialized name dictionaries are generally of little use for this task, as it is often the essence of a marketing campaign to create new, previously unused product or brand names.

In natural language processing, there are previous works which address the automatic generation and retrieval of slogans and creative names. Veale (2011) presents a search engine for creative text retrieval, which assists copywriters to find metaphors and unusual word combinations. Özbal and Strapparava (2012) describe an automatic approach to generate neologisms that can be used as product names or slogans. Both works are focused on the English language and on the identification or generation of *good* names, whereas our work aims at the detection of *problematic* names without focusing on a particular language or target market. Our task is similar to automatically distinguishing cognates and false friends. Inkpen et al. (2005) use orthographic similarity metrics for this task including Soundex, which we also propose for our method. Follow-up works by Mitkov et al. (2007), Gomes and Lopes (2011), Beinborn et al. (2013), and Ciobanu and Dinu (2014) propose different edit distance, machine translation, and seman-

tic similarity methods for identifying cognates. While false friends play a crucial role for the detection of marketing blunders, the existing approaches cannot be used directly, because they are specific for a certain language pair and do not make any assumptions on the (negative) connotations of a trade name. Kondrak and Dorr (2004) adapt orthographic and phonetic similarity methods to find confusable drug names. Unlike our tool which retrieves pragmatically marked sense descriptions, their work is limited to identifying similar forms.

The recognition of words and phrases associated with opinions and emotions is the goal of sentiment analysis. There have been multiple attempts to construct large sentiment lexicons, such as SentiWordNet (Esuli and Sebastiani, 2006). Banea et al. (2008) propose a bootstrapping approach to induce such lexicons from a small, manually defined seed list. Our approach is similar, since we propagate pragmatic information based on lexical relations. However, we do not require a seed list and we consider relations across many languages. While monolingual resources are not suitable for our task at all, the existing multilingual sentiment lexicons are severely limited in size. The NRC Word-Emotion Association Lexicon (Mohammad and Turney, 2013) is among the largest and available in about 40 languages, but since it has been automatically translated, it does not distinguish word senses and lacks slang and dialects. In recent work, Vo and Zhang (2016) propose a neural network architecture to build sentiment lexicons relying on emoticons as distant supervision signals. Although they currently publish only English and Arabic lexicons, such (almost) unsupervised methods are promising for building multilingual lexicons.

3 Task Formalization

Let T denote a product, brand, or company name. Our goal is to develop a method $\mathcal{M}$ retrieving a set of clues $C = \mathcal{M}(T)$ for a given T, which can be used by copywriters to decide whether T should be accepted or rejected as a name. We consider each clue $c = (w, \ell, d) \in C$ as a tuple of a word form w of language ℓ and a textual description d, which paraphrases the (potentially problematic) meaning of w in ℓ. While T is not specific to any language, a copywriter is typically only fluent in a few number of languages, which is why d must be in a language spoken by the end user (hereafter *output language*). For the example $T =$ "Silver Mist", a method could return the following clues:

#	form w	language ℓ	description d
1	Silber	German (deu)	A shiny gray color.
2	mist	English (eng)	A layer of fine droplets or particles.
3	Mist	German (deu)	Manure; animal excrement.
4	miist	Seri (sei)	An animal of the family Felidae.
5	miste	Danish (dan)	To lose something.
6	silver mine	English (eng)	A mine for silver ore.

The first clue refers to the German translation of *silver*, which has the similar word form *Silber*. The second and third clues address the word *mist* with its meanings in English and German. The fourth and fifth clues refer to similar word forms of *mist* in Danish and Seri (an isolated language spoken in Mexico). The last clue addresses a multi-word expression which has a similar form to the entire name T. When deciding if T should be rejected as a name, only the third and fifth clues are helpful, since the specified meanings are negatively connotated and thus do not enhance the value of a product with such a name. We say a clue c is *relevant* for T if it should be considered in the decision of accepting or rejecting the name. Note that there might be good reasons for a copywriter to ignore the fact that *Mist* has a negatively connotated meaning in German. In section 5, we discuss such cases.

The primary objective of a method $\mathcal{M}$ is to return relevant clues for as many names as possible and thus obtain a high recall. While this is obviously important for maximizing the usefulness of the method, a secondary objective is to retrieve *only* relevant clues and thus obtain a high precision. The rationale behind this is to minimize the amount of information that needs to be manually checked by humans.

4 Proposed Method

Our solution is based on the following considerations:

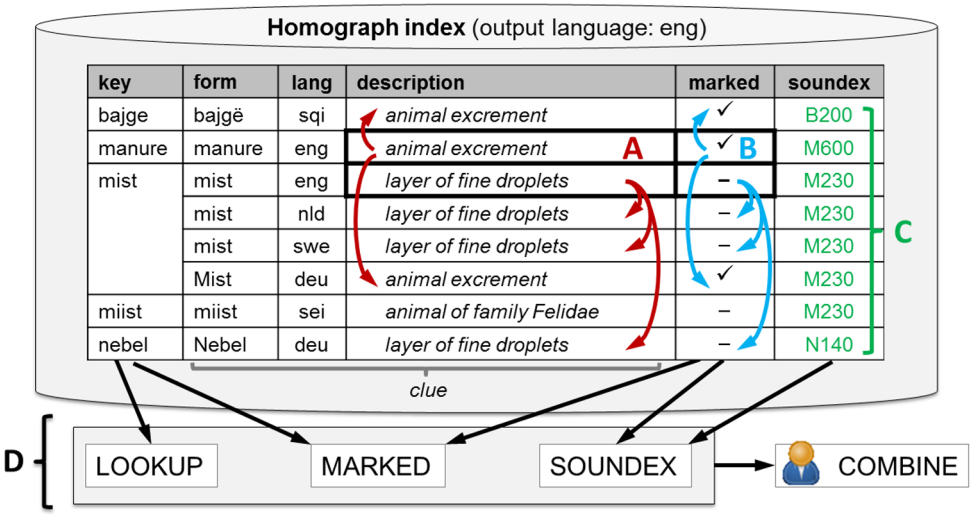


Figure 1: Running example of our method showing sense definition (A) and pragmatic label propagation (B), Soundex representation (C), and the practical usage scenario based on different query methods (D)

(1) The input name T can be of any language. Approaches using *monolingual* corpora or knowledge bases are therefore of little help or yield a tool that can only be used for a single target market. Instead, we aim at covering many languages and markets.
(2) The description of a clue should be limited to a few predefined output languages understood by the copywriters using the system.
(3) A solution must deal with non-standard language varieties, such as slang and dialects, to detect vulgarities, negative connotations, etc.
(4) Crowdsourcing platforms might prove helpful, but do not return instant feedback, require to divulge T before its official announcement, and are highly biased towards certain languages (cf. Pavlick et al., 2014).
(5) We lack training data for the marketing blunder task, which is why we do not consider data-driven or machine-learning approaches yet. Large annotated multilingual datasets including non-standard language will be necessary to learn a generalizing model (cf. section 7).

We propose using data from Wiktionary, which is particularly suitable for this task, since it contains lexicon entries and translations in many languages and a broad diversity of technical domains, colloquial language, slang, and dialects (cf. Meyer and Gurevych, 2012). As the users of our tool, we assume copywriters speaking English and German, which is a realistic example for central European marketing agencies. Our method is, however, not limited to these two output languages.

In the remaining section, we describe our family of methods $\mathcal{M}(T)$, which first segment the input name T into a sequence of k tokens $T = t_1 t_2 \dots t_k$. The methods then create the set Q of all token n-grams in T and retrieve clues for each $q \in Q$ from a huge inter-lingual index. We create this so-called *homograph index* by extracting and propagating sense definitions, translations, and pragmatic labels from Wiktionary.[2] The four steps of this approach are explained below and summarized in figure 1. We use the running example $T =$ "Silver Mist", for which our methods retrieve clues from the homograph index for each token n-gram $q \in Q = \{$*Silver*, *Mist*, *Silver␣Mist*$\}$.

Step A: Propagating sense definitions. The example of the *Silver Mist* car suggests that a large share of cross-lingual marketing blunders is due to false friends (i.e., two words with the same pronunciation or written form, but different meanings). This is why we first create a *homograph index* of words sharing the same word form. Each index entry consists of a normalized word form (the *key*) that points to one or multiple clues (i.e., triples of word form w, language ℓ, and textual description d).

Initially, the homograph index contains only clues extracted from all Wiktionary word senses of the

[2]We use the DKPro JWKTL software to extract this information: `https://dkpro.github.io/dkpro-jwktl/`

output languages (e.g., English and German). We create the key by converting the lemma of the Wiktionary word sense to lower case and removing special characters and diacritics. As the textual description d of the corresponding clues, we use the senses' definitions. In a subsequent step, we extend the homograph index by adding all translations of these word senses. We apply the same normalization technique for the translated word form to obtain the key. The key *han* thus points to clues in eleven languages, including the lemma forms *Han* (e.g., English), *han* (Turkish), *hän* (Finnish), *hăn* (Frisian), *hån* (Swedish), and *hắn* (Vietnamese). This multilingual homograph index already enables queries in any language for which translations are encoded in Wiktionary. In order to determine the clues for the translated word forms, we propagate the sense definition from the output language to the translation language. This is possible, because each Wiktionary translation is associated with a specific word sense. This way, copywriters can decide to accept or reject a name using a proper sense description instead of a bare word translation, which is especially necessary for polysemous words: Showing only the English word *arm* as a translation of the Polish *broń* would not be helpful, as it remains unclear if the potentially unwanted weapon sense or the unproblematic body part sense of *arm* is meant. In figure 1 (A), we show the homograph index creation for the two English output language lemmas *manure* and *mist*. We add the translations of the corresponding Wiktionary word senses as new index entries, and we propagate the sense description from the thick-framed cells to the foreign language entries.

Our final homograph index consists of 1.3 million normalized word forms referring to about 3.0 million clues covering 2,022 languages.[3] Using this index, we can define our first method for retrieving clues, which we call LOOKUP. Given the input name T and the set of all token n-grams Q, the LOOKUP method normalizes each $q \in Q$ using the same technique as for index keys and then looks up every normalized q in our index. For $T =$ "Silver Mist", LOOKUP returns 48 clues from six languages.

Step B: Propagating pragmatic labels. An important goal of our approach is separating relevant from irrelevant clues. The Dutch and Swedish forms of *mist* are, for instance, cognates of the English word form and thus carry the same, unproblematic meaning. Likewise, false friends without any negative connotation, such as the English and Hungarian *fog* discussed above, yield irrelevant clues that should be ignored. In a dictionary, the corresponding entries for these words are usually *unmarked* – i.e., they are not associated with a particular language variety, but considered standard language. As opposed to that, the German *Mist* is marked as "umgangssprachlich" ("slang") and as "verärgerte Äußerung" ("annoyed utterance") in Wiktionary, which are good indicators to avoid using *Mist* in a product name. We call these markings *pragmatic labels* (Wiegand et al., 2010). For our purposes, we are interested in sociological labels (the diastratic variety) that mark jargon used by a certain culture, social group, or social class (e.g., *army slang*, *argot*, *children's language*), register and style labels (the diaphasic variety) that mark word senses used in certain communicative situations (e.g., *colloquial*, *informal*, *slang*), and evaluative labels (the diaevaluative variety) that mark offensive words and words with a certain connotation (e.g., *pejorative*, *rude*, *derogatory*). Note that this goes beyond sentiment lexicons, which typically focus on the diaevaluative variety. In Wiktionary, pragmatic labels are specified at the beginning of a sense definition, usually enclosed in parentheses, typed in italics, or separated by a colon. We extract all the labels used for the word senses of the output languages. Of the 2,440 distinct labels used at least three times, we manually select 245 labels that belong to one of the three label categories and we enrich our homograph index by storing whether a word sense is marked by one of these labels. In a subsequent step, we propagate the labels from the output languages to the other languages by following the translation links. The underlying assumption is that a pragmatic label of a word sense in one language is conserved in another language, given that the translation is correct. Figure 1 (B) shows an example for English: The output language word sense of *manure* is marked as slang. We propagate this marking to the equivalent German word sense *Mist* and the Albanian (sqi) word sense *bajgë*. About 63,000 clues of the homograph index are marked by at least one of the 245 pragmatic labels. Based on this, we define a second method called MARKED, which looks up each normalized token n-gram of Q in the homograph index (equivalent to LOOKUP), but returns only clues marked by a pragmatic label.

[3] But note the long tail: 1,015 languages have only one clue.

Step C: Similar word form representations. Marketing blunders are of course not limited to forms with the exact same written form. Choosing the word forms *misd*, *misth*, or *miist* could, for instance, cause similar reactions in German-speaking markets as their pronunciation is highly similar to *Mist*. This is why we propose a third method SOUNDEX, which queries the homograph index for marked word forms starting with the same three letters and having the same *Soundex* representation (Russell, 1918) as the queried token n-gram (but not being identical to it). Soundex is an algorithm that returns a pseudo-phonetic representation of a given English word, which consists of the word's initial letter followed by at least three digits denoting groups of similar consonants. The main idea of the algorithm is that two words with similar pronunciations return similar Soundex representations: The Soundex representation of both *mist* and *miist* is M230, but M62352 for *marketing*. Although the Soundex algorithm is designed for the English language, we apply it to any token n-gram and leave the development of a language-independent pseudo-phonetic model to future research. To allow for faster queries, we precompute the Soundex representations for all clues of our homograph index as shown in figure 1 (C). When using the SOUNDEX method, we can then create a Soundex representation for each token n-gram $q \in Q$ and retrieve all marked clues that have the same Soundex representation in a simple homograph index lookup.

Step D: Practical, semi-automatic usage scenario. We combine the three methods by first querying the homograph index using MARKED. If this search does not yield relevant clues, we query the index using SOUNDEX and, analogously, we query the index using the LOOKUP method if there are still no relevant clues. We thus define the method COMBINE as the sequential application of MARKED, SOUNDEX and LOOKUP, see figure 1 (D). The rationale behind this is to simulate a practical usage situation of our approach: Since only a fraction of the entries are marked by pragmatic labels, we first present those to a user (MARKED). If she or he finds evidence for a marketing blunder, no further lookup is required for the given name. Otherwise, the user can check for marked forms with a similar form representation (SOUNDEX) and only turn towards reading all entries (LOOKUP) if there are still no relevant clues.

5 Evaluation

To evaluate our approach, we conduct two experiments: (1) We measure the performance of our four methods using a newly created dataset of previously occurred cross-lingual marketing blunders. (2) We apply our method to a large dataset of international brand names to check for potential blunders.

Marketing blunder dataset. As a novel evaluation dataset for this task, we extract the marketing blunder examples discussed by Ricks (2006, § 3) and provided on the homepage of the British consultancy *Commisceo Global*.[4] We omit examples that are not related to a name or whose name or translation is not explicitly provided. Ricks notes, for instance, that a U.S. food manufacturer wrongly translated the name of their mascot "Jolly Green Giant" into Arabic as "intimidating green ogre", but does not provide the exact Arabic form, which would be required to properly simulate the tool-assisted detection of this blunder. For each blunder in this dataset, we store the problematic name T, a remark on the vendor or type of product, and a short textual explanation of the blunder. In addition to that, we manually group the blunders into the following four categories:

- *vulgar:* names containing vulgar, rude, or offensive expressions,
- *sexual:* names with sexual innuendos,
- *negative:* names with negative connotations or suggesting negative properties,
- *intent:* names containing an expression with a different, unrelated meaning in a certain language causing astonishment and distraction among potential customers.

The fruit drink *Pavian* is an example for the *intent* group since *Pavian* means *baboon* in German, which caused distraction among customers, although the word is not negatively connotated. Our initial dataset consists of 44 cross-lingual marketing blunders, which we make publicly available for other researchers.

Experiment 1. We apply our four methods to each problematic name of this novel marketing blunder dataset. In total, our methods returned 1,494 clues. In order to judge a clue relevant or irrelevant, we

[4] http://www.commisceo-global.com/blog/cross-cultural-marketing-blunders (accessed: 2016-05-02)

	MARKED	SOUNDEX	LOOKUP	COMBINE
Detected marketing blunders:	18 / 44	18 / 44	28 / 44	34 / 44
intent	0 / 3	0 / 3	3 / 3	3 / 3
negative	3 / 15	8 / 15	10 / 15	14 / 15
sexual	7 / 14	4 / 14	7 / 14	8 / 14
vulgar	8 / 12	6 / 12	8 / 12	9 / 12
Retrieved clues:	105 / 151	85 / 247	341 / 1202	229 / 517
Precision P:	.70	.34	.28	.44
Recall R:	.41	.41	.64	.77
F_1 score:	.52	.37	.39	.56
F_2 score:	.45	.39	.51	.67

Table 1: Evaluation results for our marketing blunder dataset

ask two human raters to annotate this set of retrieved clues. The raters agree on 95 % of the judgments yielding an inter-rater agreement of $\kappa = .87$ (using Cohen's kappa). Based on this agreement, we consider the annotations reliable (cf. Artstein and Poesio, 2008). For obtaining a gold standard, we ask an additional adjudicator to decide on the 76 ties.

Table 1 summarizes the evaluation results. We report the number of detected blunders over the total number of blunders both for the whole dataset and separately for each blunder category. The table additionally provides the total number of relevant and retrieved clues as well as the precision, recall, F_1 and F_2 scores. We define the precision P as the ratio of relevant clues to the total number of retrieved clues (i.e., a method is more precise if it returns more relevant clues) and recall R as the proportion of detected marketing blunders in the dataset (i.e., a method has a higher recall if it is able to detect more marketing blunders). The F_1 and F_2 scores follow the standard definitions of being the (weighted) harmonic mean between precision and recall. In accordance with our task's primary objective (see section 3), the F_2 score prefers high recall over high precision.

The basic LOOKUP method yields a recall of .64 indicating that our homograph index is able to effectively detect cross-lingual marketing blunders. As the low precision indicates, there are, however, a large number of irrelevant clues that are retrieved by this simple index lookup. As opposed to that, we find a high precision for the MARKED method, as the index entries marked with pragmatic labels yield relevant clues in over 70 % of the cases. The MARKED method is, however, not suitable to detect marketing blunders of the categories *intent* and *negative*, which causes a low recall. It should be noted that it is not surprising to find the recall of MARKED lower than that of LOOKUP, because the former returns a subset of the latter. This is different for SOUNDEX, which facilitates the detection of marketing blunders that remain unseen by the other methods. We find that our semi-automatic usage simulation COMBINE yields the most reasonable trade-off between precision and recall, since it achieves the highest recall of the three methods and a higher precision than LOOKUP and SOUNDEX. With this method, we are able to detect 34 of the 44 cross-lingual marketing blunders ($R = .77$) and we achieve the highest F_1 and F_2 scores. For the corresponding marketing campaigns, the copywriters would have to examine a total of 517 clues (on average 12 per blunder); 229 of them are relevant (on average 5 per blunder; $P = .44$).

Experiment 2. In our second experiment, we aim at finding potential marketing blunders in existing names on a larger scale. To this end, we use all 998 brand names of the BrandPitt corpus (Özbal et al., 2012) and retrieve clues using our tool. It is important to note that this corpus mostly contains top-tier international brands, such as *Pizza Hut*, *Jaguar*, and *IKEA*, whose names are established for many years and thought over by leading marketing agencies. Initially, we therefore did not expect to find many relevant clues. Our tool returns a total of 756 clues using MARKED, 3,549 using SOUNDEX, and 17,270 using LOOKUP. Given the high number of brand names and clues, we focus on the clues returned by MARKED (i.e., the first step of our simulation) and ask two human raters to judge them relevant or irrelevant for deciding whether or not to use a name for a particular region in the world. The raters find

154 (rater 1) and 192 (rater 2) of the 756 clues to be relevant. This corresponds to returning relevant clues for 70 (rater 1) and 88 (rater 2) of the 215 names for which MARKED returned at least one clue. The raters agree in 90 % of the cases ($\kappa = .72$).

Finding this many relevant clues is surprising, which is why we carefully checked the annotated data. We indeed find very helpful clues, which – in our opinion – should at least be known to the corresponding marketing agencies. The name of the animation studio *Pixar* means, for example, to urinate in the Catalan language. The name of the Norwegian confectionery *NERO* and the German software producer *Nero* means *brainiac* in Finnish, which is marked derogatory in some contexts, and the related form *ñero* is a rough equivalent of *thug* or *gangsta* in Colombia. The name of the Russian car manufacturer *Lada* has a related form *låda* in Swedish meaning *box*, including a pejorative meaning for unattractive houses and cars. Though being differently pronounced, the relationship could still be problematic in written communication (e.g., a poster with barely visible ring diacritic). In the Darfur and Chad language Fur, *martìn* is a vulgar form for the buttocks, which the raters consider relevant for the *Aston Martin* car brand. The popular coffee bar name *Thanks a Latte* might be less suitable for the German market, where *Latte* is a colloquial word for erected penis. The name of the *Coco Pops* cereals might prove problematic in French markets, where *coco* means cocaine. Though being relevant, many of these clues are of course not problematic, since the brand names are already well-established. However, we consider our tool helpful for new names, which lack this brand strength. Another important finding from this experiment is that there are often relevant clues for which a more salient word sense exists that prevents misinterpretation. The related forms *cocó* and *cocô* mean, for instance, *shit* in Portugal and Brazil, respectively, which we consider highly relevant for the *Coco Pops* example. Since there is, however, also the frequent form *coco* in both languages meaning coconut, it is unlikely that the product name is misinterpreted.

6 Discussion and Error Analysis

MARKED works well for detecting the blunder categories *vulgar* and *sexual*, whereas LOOKUP predominantly retrieves clues for the *intent* and *negative* categories. Since we designed SOUNDEX to only return marked entries, it is likewise less suitable for *negative* and *intent*. For *negative* blunders, additional knowledge from sentiment lexicons could be helpful to recognize relevant clues containing a negative connotation. In order to do so, we require either language-independent sentiment analysis tools or a sense-disambiguated notion of sentiment for Wiktionary word senses, which can be used for propagating sentiment information across languages. Existing lexicons, such as the NRC Word-Emotion Association Lexicon (Mohammad and Turney, 2013), provide a good starting point.

Blunders of the *intent* category are much harder and most likely require copywriters to read all clues returned by LOOKUP. Semantic relatedness measures might prove useful for identifying false friends with highly different meanings. Especially in the BrandPitt dataset, we note, however, that there are some names using pragmatically marked or ambiguous words intentionally. The *Get Lost Magazine*, for instance, includes the English phrase *get lost* which raises negative associations of rudely being asked to leave. Since the magazine is about adventure traveling, the copywriters use this name on purpose to create an interesting name with multiple interpretations. This illustrates why we consider it important to model marketing blunder detection as a semi-automatic task leaving the final decision to humans.

None of our current methods is able to detect blunders whose text contains a problematic word as a substring. The product name *FARTFULL*, for instance, needs to be split into *fart* and *full*, before an index lookup can yield relevant clues. The large number of substring combinations would, however, yield a huge number of irrelevant clues if all combinations are queried. The English words *fartherer* or *penny-farthing* contain, for instance, the substring *fart*, but do not lead to a vulgar interpretation right away. A similar problem occurs for inflected word forms (e.g., *Vicks*). Relying on automatic lemmatization or morphological analysis is problematic, since such tools are usually language-specific. For our task, they would need to cover essentially any language. Trimming suffixes of different lengths might be a solution, but would not work for highly agglutinative languages.

The use of Soundex representations for identifying similar forms works well in some cases, for example for finding the Finnish *hullu* (IPA: [ˈhulːu]) meaning insane, which has a similar pronunciation as

the American streaming service *Hulu* ([huˈlu]). For the Japanese *creap*, SOUNDEX retrieves the English *creep* (annoying person) and the French *crevé* (extremely fatigued). While the former is relevant for detecting this blunder, the latter might have a somewhat similar pronunciation in English (e.g., [kɹiːv]), but definitely not in French ([kʁə.ve]). The same is true for *Lada* and the Swedish *Låda* ([ˈloː.da]). Such problems are due to Soundex being designed for English. Since Wiktionary also contains pronunciation information, a future method could lookup index entries with similar or equal IPA representation, given that they are available at a large-scale and for a large number of languages. Recently, Deri and Knight (2016) introduced a new grapheme-to-phoneme model for almost any language, which we consider highly relevant for indexing language-independent pseudo-phonetic representations.

Ricks (2006) also notes the English form *crap* as a blunder cause for *creap*, since it has a similar spelling. None of our methods, however, returns *crap* as a clue for this name. An obvious solution would be the use of string similarity metrics, such as Levenshtein's edit distance. We indeed tried this metric, but found that it returns a huge number of irrelevant clues. We therefore suggest to develop a modified edit distance metric for this task, which, for example, puts more weight on editing letters with similar shape. Finally, our tool cannot retrieve clues for potential marketing blunders across different scripts. For the *Bardak* machines, a method would have to transliterate the Latin spelling to the Cyrillic бардак in order to find the problematic meaning of a whorehouse. Future work should incorporate state-of-the-art transliteration systems for as many language pairs as possible.

7 Future Research Demands and Dissemination

Along with this paper, we publish our newly compiled marketing blunder dataset, homograph index, and annotations. In addition to that, we provide a web interface which implements the three steps of our COMBINE method. It allows retrieving clues for arbitrary names. Besides product, brand, and company names, our tool can retrieve clues for acronyms (e.g., of proposals) and person or organization names, which might cause misinterpretations in foreign languages.[5]

With these materials, our Wiktionary-based method and the detailed error analysis, we lay the foundation for the new natural language processing task of detecting marketing blunders. This task raises a number of important research challenges:

- Finding good names is a creative process, for which copywriters intentionally deviate from known patterns. This makes it hard to model the overall detection task with standard pattern recognition algorithms.
- Our task formulation is based on the notion of clues, which explain *why* a name is considered problematic. As opposed to that, many current tasks use a classification setup (e.g., a binary classification into *problematic* and *unproblematic*), which only indicates *if* there is a problem.
- Evaluating a name is highly subjective, as there might be intended ambiguity, jokes and language games, varying association and brand strength, etc. We therefore introduce marketing blunder detection as a semi-automatic task, which are typically very difficult to evaluate.
- Detecting marketing blunders makes most sense if *all* languages are considered, which is especially challenging for poorly documented languages. The necessity to limit the number of output languages raises another interesting challenge of separating object and meta language.

There is a large variety of future projects around this task. First and foremost, we require larger datasets for developing and evaluating our methods. While Wiktionary is continually updated by its community, future versions will yield improved coverage. For supporting additional output languages beyond English and German, it is necessary to scrape other language versions and associate the pragmatic label system with the existing index in a one-time effort. Further knowledge bases and corpora of colloquial language (e.g., from Twitter) may prove useful for background knowledge. For evaluation data, it will be interesting to cooperate with marketing researchers and copywriters. Previously occurred marketing blunders might be collected in a crowdsourcing effort. The second important strand of research will be better blunder detection methods for retrieving relevant clues. The most important issues to solve are

[5] https://github.com/UKPLab/coling2016-marketing-blunders

the intelligent segmentation of names containing problematic words, the automatic transliteration and phonetic representation of the index entries, and the integration of multilingual sentiment lexicons and analysis methods.

8 Conclusion

In this paper, we introduced the task of detecting cross-lingual marketing blunders. In addition to a formal task definition, we proposed a knowledge-based method, which propagates pragmatic labels to translated word senses from Wiktionary. Our final tool assists copywriters who design new names and accept or reject a suggested name based on a number of clues returned by the automatic method. We evaluated our work in two experiments. On a newly created dataset of previously occurred marketing blunders, we were able to detect 78 % of the problematic names. Our second experiment identified between 150 and 200 relevant clues for a large collection of top-tier international brand names.

We find that Wiktionary is well-suited for this task, as it contains many languages and a large number of pragmatic labels allowing us to process non-standard language varieties, such as slang. These language varieties are often absent from newswire corpora and expert-build dictionaries and therefore remain underresearched in our community.

We put a special focus on the error analysis and learned that follow-up work needs to find better tools and methods for computing language- and script-agnostic orthographic and phonetic similarity, for interpreting negative connotations that appear as substrings, and for language-independent sentiment and morphological analysis. To establish the new marketing blunder detection task, we finally discussed its main challenges and demands in order to suggest a research agenda to the scientific community. In future work, it will be especially interesting to cooperate with marketing agencies, in order to study how copywriters use a blunder detection tool for yet unknown product, brand, or company names.

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant № I/82806, by the German Research Foundation under grant № GU 798/17-1, and by the German Institute for Educational Research (DIPF). We would like to thank the anonymous reviewers for their helpful comments.

References

Ron Artstein and Massimo Poesio. 2008. Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics*, 34(4):555–596.

Carmen Banea, Rada Mihalcea, and Janyce Wiebe. 2008. A Bootstrapping Method for Building Subjectivity Lexicons for Languages with Scarce Resources. In *Proceedings of the Sixth International Language Resources and Evaluation*, pages 2764–2767, Marrakech, Morocco.

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2013. Cognate Production using Character-based Machine Translation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 883–891, Nagoya, Japan.

Alina Maria Ciobanu and Liviu P. Dinu. 2014. Automatic Detection of Cognates Using Orthographic Alignment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 99–105, Baltimore, MD, USA.

Guy Cook. 1992. *The Discourse of Advertising*. The INTERFACE series. London: Routledge.

Tevfik Dalgic and Ruud Heijblom. 1996. International Marketing Blunders Revisited: Some Lessons for Managers. *Journal of International Marketing*, 4(1):81–91.

Aliya Deri and Kevin Knight. 2016. Grapheme-to-Phoneme Models for (Almost) Any Language. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 399–408, Berlin, Germany.

Andrea Esuli and Fabrizio Sebastiani. 2006. SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 417–422, Genoa, Italy.

Georg Felser. 2010. Intercultural Marketing. In Alexander Thomas, Eva-Ulrike Kinast, and Sylvia Schroll-Machl, editors, *Handbook of Intercultural Communication and Cooperation*, chapter 1.3, pages 228–242. Göttingen: Vandenhoeck & Ruprecht.

Luís Gomes and José Gabriel Pereira Lopes. 2011. Measuring Spelling Similarity for Cognate Identification. In Luis Antunes and H. Sofia Pinto, editors, *Progress in Artificial Intelligence: Proceedings of the 15th Portuguese Conference on Artificial Intelligence*, volume 7026 of *Lecture Notes in Computer Science*, pages 624–633. Berlin/Heidelberg: Springer.

Diana Inkpen, Oana Frunza, and Grzegorz Kondrak. 2005. Automatic Identification of Cognates and False Friends in French and English. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 251–257, Borovets, Bulgaria.

Frédéric Jallat and Allan J. Kimmel. 2002. Marketing in culturally diverse environments: The case of Western Europe. *Business Horizons*, 45(4):30–36.

Nina Janich. 2013. *Werbesprache: Ein Arbeitsbuch*. Tübungen: Narr, 6th edition.

Gary A. Knight. 1995. Educator Insights: International Marketing Blunders by American Firms in Japan—Some Lessons for Management. *Journal of International Marketing*, 3(4):107–129.

Grzegorz Kondrak and Bonnie Dorr. 2004. Identification of Confusable Drug Names: A New Approach and Evaluation Methodology. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 952–958, Geneva, Switzerland.

Heinz Küpper. 1984. *Illustriertes Lexikon der deutschen Umgangssprache*. Stuttgart: Klett.

Christian M. Meyer and Iryna Gurevych. 2012. Wiktionary: A new rival for expert-built lexicons? Exploring the possibilities of collaborative lexicography. In Sylviane Granger and Magali Paquot, editors, *Electronic Lexicography*, chapter 13, pages 259–291. Oxford: Oxford University Press.

Ruslan Mitkov, Viktor Pekar, Dimitar Blagoev, and Andrea Mulloni. 2007. Methods for extracting and classifying pairs of cognates and false friends. *Machine Translation*, 21(1):29–53.

Saif Mohammad and Peter Turney. 2013. Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence*, 29(3):436–465.

Gözde Özbal and Carlo Strapparava. 2012. A Computational Approach to the Automation of Creative Naming. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 703–711, Jeju Island, Korea.

Gözde Özbal, Carlo Strapparava, and Marco Guerini. 2012. Brand Pitt: A Corpus to Explore the Art of Naming. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, pages 1822–1828, Istanbul, Turkey.

Ellie Pavlick, Matt Post, Ann Irvine, Dmitry Kachaev, and Chris Callison-Burch. 2014. The Language Demographics of Amazon Mechanical Turk. *Transactions of the Association for Computational Linguistics*, 2:79–92.

David A. Ricks. 2006. *Blunders in International Business*. Malden: Blackwell Publishing.

Adrian Room. 1982. *Dictionary of Trade Name Origins*. London: Routledge & Kegan Paul.

Robert C. Russell. 1918. *Index*. United States Patent 1,261,167, filed October 25, 1917, published April 2, 1918.

Richard A. Spears. 2007. *McGraw-Hill's American Slang Dictionary*. Chicago: McGraw-Hill, 2nd edition.

Tony Veale. 2011. Creative Language Retrieval: A Robust Hybrid of Information Retrieval and Linguistic Creativity. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 278–287, Portland, OR, USA.

Duy Tin Vo and Yue Zhang. 2016. Don't Count, Predict! An Automatic Approach to Learning Sentiment Lexicons for Short Text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 219–224, Berlin, Germany.

Herbert Ernst Wiegand, Michael Beißwenger, Rufus H. Gouws, Matthias Kammerer, Angelika Storrer, and Werner Wolski, editors. 2010. *Wörterbuch zur Lexikographie und Wörterbuchforschung / Dictionary of Lexicography and Dictionary Research*, volume 1 (Systematische Einführung / Systematic Introduction, A–C). Berlin/New York: de Gruyter.

Ambient Search: A Document Retrieval System for Speech Streams

Benjamin Milde[1,2], **Jonas Wacker**[1], **Stefan Radomski**[2],
Max Mühlhäuser[2], and **Chris Biemann**[1]

[1] Language Technology Group / [2] Telecooperation Group
Computer Science Department
Technische Universität Darmstadt, Germany

Abstract

We present Ambient Search, an open source system for displaying and retrieving relevant documents in real time for speech input. The system works ambiently, that is, it unobstructively listens to speech streams in the background, identifies keywords and keyphrases for query construction and continuously serves relevant documents from its index. Query terms are ranked with Word2Vec and TF-IDF and are continuously updated to allow for ongoing querying of a document collection. The retrieved documents, in our case Wikipedia articles, are visualized in real time in a browser interface. Our evaluation shows that Ambient Search compares favorably to another implicit information retrieval system on speech streams. Furthermore, we extrinsically evaluate multiword keyphrase generation, showing positive impact for manual transcriptions.

1 Introduction

Recent advancements in Automated Speech Recognition (ASR) and Natural Language Understanding (NLU) have proliferated the use of personal assistants like Siri[1] or Google Now[2], with which people interact naturally with their voice. However, the activation of such systems has to be specifically triggered and they are targeted to an (ever–growing) set of anticipated question types and commands.

When taking part in a conversation or listening to a lecture, people may want to look up helpful information or check facts. Manually checking this information or interacting with a personal assistant would hamper the flow of the discussion, respectively distract from the lecture. In the following, we present *Ambient Search*, a system that ambiently researches relevant information, in the form of proposing relevant documents to users in conversations or users who passively listen to spoken language. In contrast to other personal assistants, our system is not specifically triggered, it unobtrusively listens to speech streams in the background and implicitly queries an index of documents. We see the following utility in our approach: The assistant stays in the background and does not disturb the user. Access to the displayed snippets is on demand and the user can access the information in context without the need to formulate a specific query.

On the other hand, these advantages are fundamentally based on how well the system is able to retrieve relevant documents, as the system's utility diminishes when proposing a lot of irrelevant documents. In this paper, we also evaluate how well the system is able to retrieve relevant Wikipedia articles in spite of average speech recognition word error rates (WER) of 15.6% on TED talks and show that it finds more relevant articles compared to another implicit information retrieval system on speech streams.

The next section discusses related research, while we give an overview and technical details of our approach in Section 3. We evaluate keyword recognition and retrieval relevance in Section 4, and conclude in Section 5.

[1] http://www.apple.com/ios/siri/

[2] https://www.google.com/landing/now/

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2082–2091, Osaka, Japan, December 11-17 2016.

2 Related Work

The Remembrance Agent (Rhodes and Starner, 1996) is an early prototype of a continuously running automated information retrieval system, which was implemented as a plugin for the text editor Emacs[3]. Given a collection of the user's accumulated email, Usenet news articles, papers, saved HTML files and other text notes, it attempts to find those documents that are most relevant to the user's current context. Rhodes and Maes (2000) defined the term *just-in-time information retrieval agents* as "a class of software agents that proactively present information based on a person's context in an easily accessible and non-instrusive manner". A person's context can be a very broad term in this regard. E.g. Jimminy (Rhodes, 1997) represents a multimodal extension of the Remembrance Agent. Dumais et al. (2004) introduced an implicit query (IQ) system, which serves as a background system when writing emails. It also uses Term Frequency – Inverse Document Frequency (TF-IDF) scores for keyword extraction, like the Remembrance Agent.

Other systems cover a user's explicit information needs, e.g. Ada and Grace are virtual museum guides (Traum et al., 2012), that can suggest exhibitions and answer questions. The Mindmeld[4] commercial assistant can listen to conversations between people to improve the retrieval results by fusing the users location information with from transcripts extracted keywords. The FAME interactive space (Metze et al., 2005) is a multi-modal system that interacts with humans in multiple communication modes in order to suggest additional information to them. Although FAME supports speech recognition and voice commands, it only listens to conversations for a longer period of time when it guesses a conversation's topic and can suggest documents with explicit commands. Another class of systems try to record the content of a conversation or speech stream and visualize it using a network of terms: E.g. SemanticTalk (Biemann et al., 2004) iteratively builds a structure similar to a mind map and can also visualize conversation trails with respect to background documents.

The most similar approach to *Ambient Search* was presented by Habibi and Popescu-Belis (2015), extending earlier work of an Automatic Content Linking Device (ACLD) (Popescu-Belis et al., 2000). It uses an LDA topic model for the extraction of keywords and the formulation of topically separated search queries. The extracted set of keywords as well as the ultimately returned set of document recommendations fulfill a trade-off between topical coverage and topical diversity.

Because this system can be considered a state-of-the-art system of implicit information retrieval in speech streams, we compare our approach to this one in the evaluation in Section 4, alongside a TF-IDF-based baseline. A major difference to Habibi and Popescu-Belis (2015), that operates on complete speech transcriptions only, is that our implementation is also able to retrieve relevant documents in real time, e.g. process live speech input.

3 Our Approach to Ambient Search

Our approach is based on five major processing steps, as depicted in Figure 1. These steps are carried out in real-time and in a streaming fashion, i.e. we make use of a new transcription hypothesis as soon as it is available.

At first, the speech signal is streamed into an ASR system (1). It emits the partial sentence hypothesis and also predicts sentence boundaries. Once a full sentence has been hypothesized, new keywords and keyphrases are extracted in the current sentence, if available (2). These terms are then ranked (3) and merged with the ones from previous sentences. A query is then composed, which is submitted to a precomputed index of documents (4).

Eventually, the returned documents are also aggregated (5a), i.e. previously found documents decay their score over time and newer documents are sorted into a list of n best documents. This list is thus sorted by topical relevance of the documents and by time, with newer documents having precedence. Finally, the n best relevant documents are presented to the user (5b) and updated as soon as changes become available. Alongside the n best documents, a time line of previously suggested articles is also maintained and displayed. The next subsections provide further details on the individual major processing steps.

[3] `https://www.gnu.org/software/emacs/`

[4] `http://www.mindmeld.com`

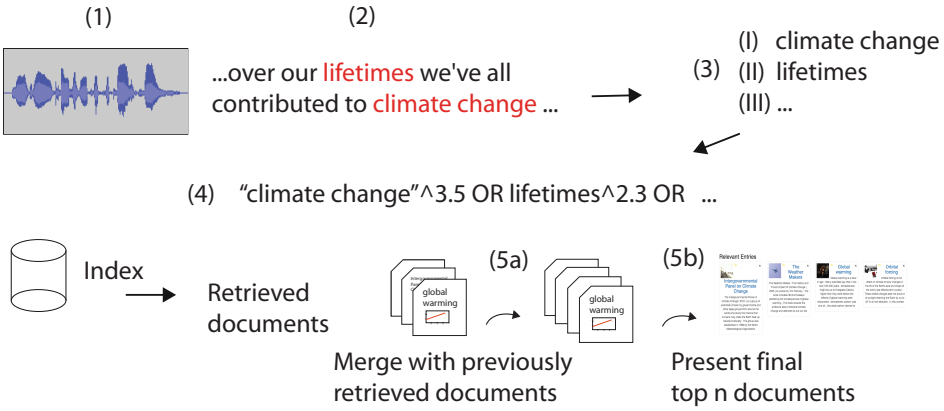


Figure 1: Processing steps of *Ambient search*

3.1 Speech Decoding

We use the popular Kaldi (Povey et al., 2011) open-source speech recognition framework and acoustic models based on the TED-LIUM corpus (Rousseau et al., 2014). We make use of online speech recognition, i.e. models that transcribe speech in real-time and emit partial transcription hypothesis, as opposed to offline models that operate on already recorded and complete utterances. The models were built using the standard recipe for online acoustic models based on a DNN-HMM acoustic model and i-vectors. We also make use of the TED-LIUM 4-gram language model (LM) from Cantab Research (Williams et al., 2015). The vocabulary of the speech recognizer is determined by its phoneme dictionary[5] and is confined to about 150k words. The online speech recognizer achieved an average WER of 15.6% on TED talks that we selected for the evaluation in Section 4.

We make use of kaldi-gstreamer-server[6], which wraps a Kaldi online model into a streaming server that can be accessed with websockets. This provides a bi-directional communication channel, where audio is streamed to the server application and partial and full sentence hypothesis and boundaries are simultaneously returned as JSON objects.

3.2 Keyphrase Extraction

A keyphrase, as opposed to a single keyword, can consist of one or more keywords that refer to one concept. We first precompute a DRUID (Riedl and Biemann, 2015) dictionary on a recent Wikipedia dump with scores for single adjectives or nouns and noun phrases. The restriction to only use adjectives and nouns is a common one in keyword detection, c.f. (Liu et al., 2010). DRUID is a state-of-the-art unsupervised measure for multiword expressions using distributional semantics. Intuitively, DRUID finds multiword expressions by combining an uniqueness measure for phrases with a measure for their incompleteness. Uniqueness in this context is based on the assumption that multiword expressions (MWEs) can often be substituted by a single word without considerably changing the meaning of a sentence.

The uniqueness measure uq(t) is computed with a distributional thesaurus, as the ratio of all similar unigrams of a term t divided by the number of n-grams similar to t. The incompleteness (ic) measure serves to punish incomplete terms in that it counts the number of times that the same words appear next to a term. The final DRUID measure for any term t is the subtraction of the incompleteness measure from the uniqueness measure: $DRUID(t) = uq(t) - ic(t)$. This helps to rank incomplete multiwords lower than their complete counterparts, e.g. 'red blood' is ranked lower than 'red blood cell'.

DRUID is implemented as a JoBimText (Biemann and Riedl, 2013) component, which can be down-

[5]The Kaldi TEDLIUM recipe uses CMUDICT (`http://www.speech.cs.cmu.edu/cgi-bin/cmudict`) plus a few automatically generated entries

[6]`https://github.com/alumae/kaldi-gstreamer-server`

loaded from the JoBimText project website[7] alongside precomputed dictionaries for English.

3.3 Term Ranking

We first precompute IDF and Word2Vec (Mikolov et al., 2013) lookup tables for all unique words in the Simple English Wikipedia and for all multiword terms in our DRUID dictionary. Word2Vec (CBOW) is our source of semantic similarity. We train it on stemmed text and treat multiwords as found with DRUID as opaque units. The final word embedding lookup table maps (in our case stemmed) word and phrase ids into a 100 dimensional continuous vector space. The model exploits the distributional properties of raw text for semantic similarity and the distance between embeddings can be used as a word and phrase similarity measure.

Using the lookup tables, we build a term ranking measure as follows. We extract all keyphrases from the last 10 sentences with a DRUID score of $\geq c$ and filter all stop words and any word that is not an adjective or noun, as determined by an off-the-shelf part of speech (POS)-tagger[8]. The cutoff constant c can be used to tune the amount of generated multiword candidates, with useful values ranging from 0.3 to 0.7 (see also Section 4). All multiwords and any single word remaining after filtering is proposed as a candidate. We then compute the average Word2Vec vector over all candidate terms. Finally, we score each candidate term according to the cosine distance of each term word vector to the average term word vector of the last 10 sentences and multiply this with the TF-IDF score of the given term:

$$tr(term_k, trans) = d_{cos}\left(w2v(term_k), \frac{1}{|terms|}\sum_{i=1}^{|terms|} w2v(term_i)\right) \cdot TFIDF(term_k, trans)$$

where tr is the term ranking function that ranks a $term_k$ out of the set of all candidate $terms$ to a given transcript $trans$. $w2v$ yields the embedding of the given term, TFIDF yields the TFIDF score of the given term and transcript (IDF computed on the background Wikipedia corpus) and d_{cos} is the standard cosine similarity.

This ranking measure tr can be interpreted as a combination of the distance to the core topic (Word2Vec) and the general importance of the term for the text window (TF-IDF). We use the measure to extract up to 10 highest ranked candidate keywords and keyphrases in the text window. For the first third of Alice Bows Larkin's TED talk on climate change[9], the system would rank the terms as: "climate change", future, emissions, "negative impacts", potential, profound, workplaces, behaviors, gas.

3.4 Index Queries

We use Elastic Search[10] and stream2es[11] to build an index of the Simple English Wikipedia[12]. We index all articles, including special pages and disambiguation pages and use a query filter to obtain only regular articles when querying the index. We build an OR query where at least 25% of the query terms should match (by setting the "minimum_should_match" parameter), also assigning the scores obtained in the last section to the individual terms in the query.

With the example ranking from the previous section the query would be:

```
"climate change"^23.111 future^13.537 emissions^9.431 "negative impacts"^3.120
   potential^2.985 profound^2.679 workplaces^2.562 behaviors^2.368 gas^1.925
```

It would return the following Wikipedia articles (ranked by Elastic Search in that order):

[7] http://jobimtext.org/components/druid/

[8] The POS tagger we use is from the spacy library (http://spacy.io)

[9] https://www.ted.com/talks/alice_bows_larkin_we_re_too_late_to_prevent_climate_change_here_s_how_we_adapt

[10] https://www.elastic.co/

[11] https://github.com/elastic/stream2es

[12] https://simple.wikipedia.org

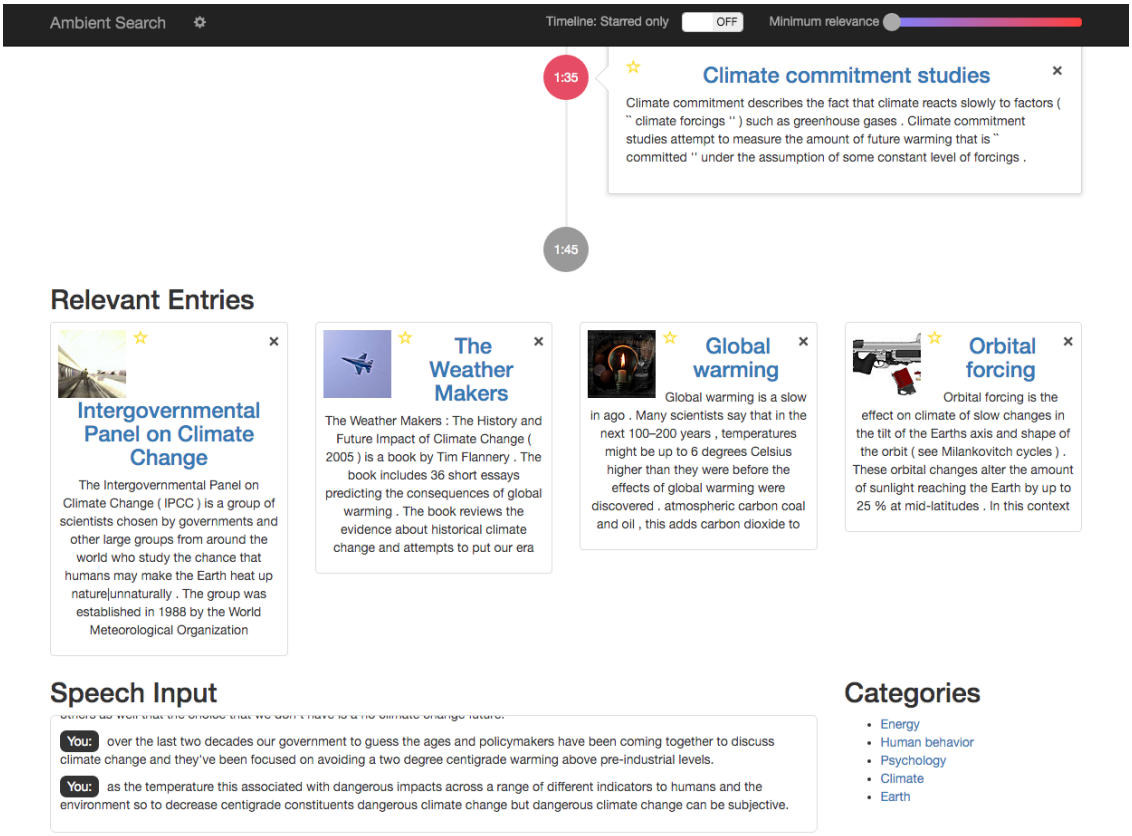


Figure 2: Screenshot of the system after listening to the first minutes of the TED talk "We're too late to prevent climate change - here is how we adapt" by Alice Bows Larkin [9]

```
"Intergovernmental Panel on Climate Change", "Global warming", "Climate change", "
    Global dimming", "The Weather Makers", "Greenhouse effect", "United Kingdom
    Climate Change Programme", "Ocean acidification", ...
```

This process is repeated for every new sentence and the scores of older retrieved documents decay (are multiplied with $d = 0.9$), to allow newer documents to rank higher.

3.5 Visual Presentation

Figure 2 gives a visual impression of our system, after it had been listening for a few minutes to Alice Bows Larkin's TED talk on climate change. We show excerpts of up to four relevant Wikipedia documents to the user. Clicking on such a document opens up a modal view to read the Wikipedia article. Articles are either retrieved online or using an offline version of the Simple English Wikipedia using XOWA[13]. Articles can be starred, to quickly retrieve them later and also removed, to signal the system that the article was irrelevant. When newer and more relevant articles are retrieved, older articles move into a timeline, which is constructed above the currently retrieved articles. The newest articles are at the bottom of the page and the page keeps automatically scrolling to the end, like a terminal, if the user does not scroll up. In the timeline, the relevance of a document is also visually displayed with different coloring of an element's circular anchor. The user can also regulate the threshold for minimum document relevance.

3.6 Implementation Details

We encapsulate the processing steps outlined in Section 3 into the following Python programs:

[13] https://gnosygnu.github.io/xowa/

(1) A *Kaldi client program*, that either uses the system's microphone or an audio file, streaming it in real time to obtain partial and full transcription hypothesis. (2) A *relevant event generator program*, that searches for new keywords and keyphrases and queries the elastic search index to obtain relevant documents. (3) The *Ambient Search server*, which sends out appropriate events to the browser view, to display the current top n relevant documents and to move older documents into a timeline.

We make use of message passing to communicate inputs and results of the individual programs using a redis-server[14]. Word2Vec and TF-IDF vectors are computed with the Gensim (Řehůřek and Sojka, 2010) package, while DRUID is precomputed as a list with JoBimText[15]. The Ambient Search web page is using HTML5/JS and Bootstrap[16] and connects to an ambient server instance running on the Python micro-framework Flask[17]. The web page is updated using Server Sent Events (SSE) or Long Polling as a browser fallback. This enables a reversed information channel, where the server pushes descriptions of new relevant documents to the browser client as it becomes available.

4 Evaluation

We base our evaluation on 30 fragments of 10 recent TED talks, which we downloaded as mp3 files from the TED.com website. These talks are not part of the TED-LIUM training dataset. In the following, we evaluate the proposed keywords and keyphrases, as well as the proposed documents from the in real-time transcribed audio file.

4.1 Keyphrase and Document Retrieval

We had two annotators manually pick terms (keywords and keyphrases) that are central to the topic of the talk and those that would cover a user's potential additional information needs. What should be included as a term can be very subjective, the inter-annotator agreement is $\kappa = 0.45$, with one annotator choosing 292 terms in total and the other 580. The overlapping set which we use in our evaluation consists of 206 terms and 460 other terms were chosen by only one of the annotators.

Finally, we also measure directly how relevant the retrieved documents are: We focus on an evaluation of the top-ranked documents returned by our ambient IR system for a particular TED talk fragment, since only top documents are suggested to the user of *Ambient Search*. The Normalized Discounted Cumulative Gain (NDCG) measure (Järvelin and Kekäläinen, 2002) is a popular choice to evaluate search engines and also takes into account the ranking of the proposed documents.

We evaluate on the top-5 returned documents of the complete system. We had two annotators that used the standard relevance scale from 0-3, where 0 means irrelevant and 3 very relevant. NDCG directly measures how relevant the returned documents are. While the effort is considerably higher, since different system outputs have to be judged, NDCG measures the end-to-end performance of the system. For computing NDCG, we pool all judgments across systems, obtaining an average of 27.7 relevance judgments per fragment, following standard practices for IR evaluations (Clarke et al., 2012). We use the standard NDCG measure with $k = 5$:

$$NDCG_k = \frac{DCG_k}{IDCG_k}$$

$$DCG_k = (rel_1 + \sum_{i=2}^{k} \frac{rel_i}{log_2 i})$$

where rel_i is a documents average relevance score in respect to the speech input. The Ideal Discounted Cumulative Gain (IDCG) assumes the best ranking of all possible relevant documents found in the set of all pooled judgements of a given transcript. The DCG on this optimal ranking, with respect to the set of documents retrieved by all systems for a particular transcript, is then used to compute IDCG.

[14] http://redis.io/
[15] http://jobimtext.org/components/druid
[16] http://getbootstrap.com/
[17] http://flask.pocoo.org/

4.2 Results

Keyword/keyphrase extraction method	Mean Recall (Std. Dev. in %)	Mean Precision (Std. Dev. in %)	Mean NDCG (Std. Dev. in %)
(1) TF-IDF baseline no MWEs, no filtering	26.97% (16.74%)	24.33% (15.42%)	0.188 (20.0%)
(2) TF-IDF baseline no MWEs, stopword filtering	39.24% (15.36%)	34.33% (10.86%)	0.387 (27.8%)
(3) TF-IDF baseline no MWEs, full filtering	40.91% (13.55%)	36.42% (10.99%)	0.426 (27.8%)
(4) TF-IDF baseline with MWEs (c=0.3), full filtering	43.22% (18.22%)	37.09% (16.61%)	0.392 (27.4%)
(5) Habibi and PB original implementation	36.68% (15.37%)	32.00% % (11.66%)	0.427 (28.0%)
(6) Habibi and PB our prep., without MWEs	43.76% (16.78%)	39.24% % (12.75%)	0.465 (24.1%)
(7) Our proposed method with MWEs (c=0.3)	**48.52% (21.55%)**	41.89 % (15.82%)	0.453 (25.7%)
(8) Our proposed method with MWEs (c=0.5)	48.08% (17.63%)	**42.42 % (13.20%)**	0.469 (26.9%)
(9) Our proposed method with MWEs (c=0.7)	48.48% (19.15%)	**42.42 % (13.45%)**	0.471 (26.1%)
(10) Our proposed method without MWEs	44.87% (17.24%)	40.08% (14.03%)	**0.481 (26.8%)**

Table 1: Comparison of TF-IDF baseline keyword and keyphrase extraction methods, the proposed LDA based keyword extraction method by Habibi and Popescu-Belis (2015) and our proposed method based on DRUID, Word2vec and TF-IDF. The comparison is based on the same Kaldi transcriptions and the same training resources (Simple English Wikipedia from May 2016).

Keyword/keyphrase extraction method	Mean Recall (Std. Dev. in %)	Mean Precision (Std. Dev. in %)	Mean NDCG (Std. Dev. in %)
(11) Habibi and PB our prep., without MWEs	43.99% (15.26%)	39.33 % (12.63%)	0.476 (21.7%)
(12) Our proposed method with MWEs (c=0.3)	51.75% (20.43%)	45.67 % (16.47%)	0.518 (24.8%)
(13) Our proposed method with MWEs (c=0.5)	52.19% (19.09%)	46.19 % (15.27%)	0.574 (22.1%)
(14) Our proposed method with MWEs (c=0.7)	**52.68% (17.20%)**	**46.85 % (14.76%)**	**0.602 (22.1%)**
(15) Our proposed method without MWEs	47.81% (17.28%)	43.52 % (16.09%)	0.578 (25.2%)

Table 2: Comparison of the proposed LDA based keyword extraction method by Habibi and Popescu-Belis (2015) and our proposed method based on DRUID, Word2vec and TF-IDF on manual TED talk transcripts.

In Table 1, we show a comparison of different methods for automatic keyword extraction on TED talk transcriptions (as produced by kaldi-gstreamer-server / the Kaldi online model). All methods use the same resources, i.e. they are all pretrained on the same Simple English Wikipedia dump from May 2016. However, our proposed method and the TF-IDF baseline can also produce terms that are DRUID multiwords, whereas the original implementation of Habibi and Popescu-Belis (2015) can only produce single keywords. All methods where allowed to produce maximally 10 words in the keyword evaluation – partially covered keyphrases where also counted as a hit for the direct keyword evaluation and a multiword term was counted as multiple words. In the NDCG evaluation, we allow each system to produce an equal number of 10 terms.

For the TF-IDF baselines (1-4), preprocessing is the most important performance factor, with the best results obtained by filtering stop words and any words that are not adjectives and nouns. However, while DRUID multiwords help to gain much better keyword recognition scores, it did not achieve a better

NDCG score on speech transcripts. We also saw good results using the method proposed by Habibi and Popescu-Belis (2015), with the diversity constraint (λ) set to the default value of 0.75, which was the optimal value in the domain of meeting transcripts. However, we noticed that the publicly available Matlab implementation of this method[18] only removed stopwords as part of its preprocessing (5). When we use our preprocessing as input (6), we can improve both keyword and NDCG evaluation scores significantly.

Our proposed methods (7-9) with enabled multiword keyphrases seem to better represent the content of fragments, as shown by the keyword judgements. Again, DRUID further improved keyword recognition scores, but it did not achieve a better NDCG score on speech transcripts. The best NDCG score using speech transcripts was obtained with our proposed method *without* using multiwords (10). We experimented with different values of c: 0.3, 0.5 and 0.7, which all lowered NDCG scores. On average, this translates to 2.16, 1.56 and 0.53 multiword terms per query respectively. The numbers are slightly lower if we use manual transcripts (1.9, 1.4 , 0.5).

We also evaluated our methods on manual transcriptions (11-15), see Table 2. Here the picture is different, as using DRUID can improve NDCG scores. However, only the highest cutoff factor of 0.7 (producing the smallest number of multiword candidates) yielded the best performance, suggesting that the number of added multiword candidates is an important parameter in the query generation. The scores on manual transcriptions can also be understood as the theoretically best possible scores for each method, assuming a perfect speech transcription system. If we compare them, we find that imperfect transcriptions have a high impact on system performance for all methods, as NDCGs are considerably higher with manual transcripts. If we correlate WER with our method in (10), we only observe a weak negative correlation of -0.193. If we use multiword terms the negative correlation is higher with a coefficient of -0.293. The comparison system from Habibi and Popescu-Belis in (6) has the lowest negative correlation of -0.118 and it does not seem to gain as much in the NDCG evaluation on perfect transcriptions as our system.

4.3 Error Analysis

If we look at fragments individually and compare our method (10) to Habibi and Popescu-Belis (2015), we find that in 15 transcription fragments their system has a higher NDCG score and in 14 our system scores higher, with one equal score. On average, in cases where our system scored higher, WER was 14.8%, and where it scored lower WER was 16.8%. For example, for all 3 fragments of the talk "Kids Science" by Cesar Harada[19], where the accent of the speaker deteriorates WER to 33.4%, our system returns much more irrelevant documents. Our average NDCG for the talk is 0.180, while Habibi and Popescu-Belis' system scores 0.454. Among the articles found by our system are "National School Lunch Program", "Arctic Ocean", "Fresh water", "Water", "Coal preparation plant", "Coal mining", "Plant" with most of the articles being irrelevant to the talk.

Word errors and resulting erroneous search terms are responsible for most irrelevant documents, e.g. "in coal power plant" appears in the transcript instead of "nuclear power plant". On the other hand, in this example Habibi and Popescu-Belis' system finds better matching articles, like "Microscope", "Light Microscope", "Mangrove" , "Fresh water", "River delta", "Fishing" which can be attributed to finding the keyword "microscope" and otherwise picking simpler keywords like "ocean", "sea", "fishing" and "river", which our system entirely misses. This changes when we run the systems on the manually transcribed texts, as e.g. our system with enabled multiword terms (9) then finds "nuclear power plant", which helps to retrieve very relevant documents ("Nuclear reaction", "Nuclear chemistry", "Nuclear power plants").

Moreover, if we enable the use of multiword terms in our method with c=0.7, we observe that NDCG was improved by the keyphrase enabled method in 9 cases, but also decreased in 11, with the other 10 transcripts remaining unchanged. If WER is poor, the keyphrase enabled methods do not seem to contribute to improving NDCG performance and tend to lower it. E.g. in the 5 transcripts with the

[18] https://github.com/idiap/DocRec

[19] http://www.ted.com/talks/cesar_harada_how_i_teach_kids_to_love_science

highest WER (19.8-40.9%, average: 26.4%), 3 scores are lowered and 2 unchanged. If we group all cases where the NDCG performance drops, we observe an average WER of 16.9% vs. 12.3% for the cases where they help to improve the NDCG score (average of all transcripts is 15.6%). This further suggests that a query generation with multiword terms helps more in cases where word error rates are low.

Interestingly, in 5 out of 30 transcripts, no multiword terms are found with c=0.7 but NDCG values were still slightly lower in all cases compared to our single word method. While the set of terms in all queries were nearly unchanged, their ranking was affected. This might be attributed to how we build IDF and Word2Vec models: multiwords are opaque units in the models. This can change the dense vectors and IDF values for the constituents of multiwords compared to training on single words and thus affect ranking scores. However, in some of the automatic transcriptions, only constituents of the correct multiwords can be found because of transcription errors, so that our method has to rank the constituent instead of the full multiword.

5 Conclusion

We presented *Ambient Search*, an approach that can show and retrieve relevant documents for speech streams. Our current prototype uses Wikipedia pages, as this provides a large document collection for testing purposes with an universal coverage of different topics.

Our method compares favorably over previous methods of topic discovery and keyword extraction in speech transcriptions. We explored the use of multiword terms as keyphrases, alongside single word terms. Our proposed extraction method using Word2Vec (CBOW) embeddings and TF-IDF is fairly simple to implement and can also be adapted quickly to other languages as it does not need any labelled training data. The only barrier of entry can be the availability of a speech recognition system in the target language.

We have started first efforts to build open source speech recognition models for German in (Radeck-Arneth et al., 2015) and have plans to support this language in future prototypes of *Ambient Search*. These speech models target distant speech recognition and could help to apply Ambient Search to more challenging situations in the future, e.g. distant conversational speech.

We also plan to evaluate a more dynamic approach to query generation, where the number of terms is dynamically chosen and not simply capped at a maximum number of term candidates after ranking. As the proposed use of multiword terms seems to be somewhat dependent on the quality of the transcription, it might also make sense to include likelihood information of the speech recognition system. Our evaluation on manual transcriptions also suggests that there is quite a large headroom for our system to benefit from any future reductions in WER of the online speech recognition component.

For actual live deployment and usage in discussions, lectures or business meetings, confidential information can be present in the speech streams. A privacy aspect has already been addressed by *Ambient Search*: the speech recognition is not carried out "in the cloud" and can be deployed on one's own infrastructure. Similarly, an offline version of the Simple English Wikipedia and a corresponding search index is used to retrieve and find articles. It can be entirely circumvented that personal information is ever transmitted through the internet – a vital aspect for the acceptance of such an application.

We have published the source code of *Ambient Search* under a permissive license on Github[20], along with all pretrained models, a demonstration video, evaluation files and scripts that are necessary to repeat and reproduce the results presented in this paper.

Acknowledgments

This work was partly supported by the Bundesministerium für Bildung und Forschung (BMBF), Germany, within the Dialog+ project within the program KMU-innovativ. We also want to thank Alexander Hendrich for contributing to improve the HTML5/JS display client and Michelle Sandbrink for helping out with the relevance judgements of the retrieved documents.

[20] https://github.com/bmilde/ambientsearch

References

C. Biemann and M. Riedl. 2013. Text: Now in 2D! A Framework for Lexical Expansion with Contextual Similarity. *Journal of Language Modelling*, 1(1):55–95.

C. Biemann, K. Böhm, G. Heyer, and R. Melz. 2004. SemanticTalk: Software for Visualizing Brainstorming Sessions and Thematic Concept Trails on Document Collections. In *Proc. ECML/PKDD*, pages 534–536, Pisa, Italy.

C. Clarke, N. Craswell, and E. Voorhees. 2012. Overview of the TREC 2012 Web Track. In *Proc. TREC*, Gaithersburg, MD, USA.

S. Dumais, E. Cutrell, R. Sarin, and E. Horvitz. 2004. Implicit Queries (IQ) for Contextualized Search. In *Proc. SIGIR*, page 594, Sheffield, UK.

M. Habibi and A. Popescu-Belis. 2015. Keyword Extraction and Clustering for Document Recommendation in Conversations. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(4):746–759.

K. Järvelin and J. Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Z. Liu, W. Huang, Y. Zheng, and M. Sun. 2010. Automatic Keyphrase Extraction via Topic Decomposition. In *Proc. EMNLP*, pages 366–376, Cambridge, MA, USA.

F. Metze, P. Gieselmann, H. Holzapfel, T. Kluge, I. Rogina, A. Waibel, M. Wölfel, J. Crowley, P. Reignier, D. Vaufreydaz, F. Bérard, B. Cohen, J. Coutaz, S. Rouillard, V. Arranz, M. Bertrán, and H. Rodriguez. 2005. The FAME Interactive Space. In *Proc. International Workshop on Machine Learning for Multimodal Interaction*, pages 126–137, Edinburgh, United Kingdom.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proc. NIPS*, pages 3111–3119, Lake Tahoe, NV, USA.

A. Popescu-Belis, J. Kilgour, P. Poller, A. Nanchen, E. Boertjes, and J. De Wit. 2000. Automatic Content Linking: Speech-based Just-in-time Retrieval for Multimedia Archives. In *Proc. SIGIR*, page 703, Athens, Greece.

D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely. 2011. The KALDI Speech Recognition Toolkit. In *Proc. IEEE ASRU*, Waikoloa, HI, USA.

S. Radeck-Arneth, B. Milde, A. Lange, E. Gouvêa, S. Radomski, M. Mühlhäuser, and C. Biemann. 2015. Open Source German Distant Speech Recognition: Corpus and Acoustic Model. In *Proc. TSD*, pages 480–488, Pilsen, Czech Republic.

R. Řehůřek and P. Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proc. LREC*, pages 45–50, Valletta, Malta.

B. Rhodes and P. Maes. 2000. Just-in-time Information Retrieval Agents. *IBM Systems Journal*, 39(3):686.

B. Rhodes and T. Starner. 1996. Remembrance Agent: A Continuously Running Automated Information Retrieval System. In *Proc. Practical Application Of Intelligent Agents and Multi Agent Technology*, pages 487–495.

B. Rhodes. 1997. The Wearable Remembrance Agent: A System for Augmented Memory. *Personal and Ubiquitous Computing*, 1(4):218–224.

M. Riedl and C. Biemann. 2015. A Single Word is not Enough: Ranking Multiword Expressions Using Distributional Semantics. In *Proc. EMNLP*, pages 2430–2440, Lisbon, Portugal.

A. Rousseau, P. Deléglise, and Y. Estève. 2014. Enhancing the TED-LIUM Corpus with Selected Data for Language Modeling and More TED Talks. In *Proc. LREC*, pages 3935–3939, Reykjavik, Iceland.

D. Traum, P. Aggarwal, R. Artstein, S. Foutz, J. Gerten, A. Katsamanis, A. Leuski, D. Noren, and W. Swartout. 2012. Ada and Grace: Direct Interaction with Museum Visitors. In *Proc. IVA*, pages 245–251, Santa Cruz, CA, USA.

W. Williams, N. Prasad, D. Mrva, T. Ash, and T. Robinson. 2015. Scaling Recurrent Neural Network Language Models. In *Proc. ICASSP*, pages 5391–5395, Brisbane, Australia.

Semi-supervised Gender Classification with Joint Textual and Social Modeling

Shoushan Li, Bin Dai, Zhengxian Gong, Guodong Zhou*
Natural Language Processing Lab
School of Computer Science and Technology, Soochow University, China
lishoushan@suda.edu.cn, bdai@stu.suda.edu.cn
zhxgong@suda.edu.cn, gdzhou@suda.edu.cn

Abstract

In gender classification, labeled data is often limited while unlabeled data is ample. This motivates semi-supervised learning for gender classification to improve the performance by exploring the knowledge in both labeled and unlabeled data. In this paper, we propose a semi-supervised approach to gender classification by leveraging textual features and a specific kind of indirect links among the users which we call "s*ame-interest*" links. Specifically, we propose a factor graph, namely Textual and Social Factor Graph (TSFG), to model both the textual and the "*same-interest*" link information. Empirical studies demonstrate the effectiveness of the proposed approach to semi-supervised gender classification.

1 Introduction

Gender classification is a fundamental task with regard to infer user's gender from the user-generated data. Recently, this task is getting increasingly more attention in some prevailing research fields, such as social network analysis and natural language processing. Applications developed from gender classification have enormous commercial value in personalization, marketing and judicial investigation (Mukherjee and Liu, 2010; Burger *et al.*, 2001; Volkova *et al.*, 2013).

In social media, conventional methods handle gender classification as a supervised learning problem over the past decade (Corney *et al.*, 2002; Ciot *et al.*, 2013). In supervised learning approaches, both user-generated textual and user social link features are verified to be effective for gender classification. For instance, in Figure 1, it is easy to infer *User* ***c*** to be a *female* through analyzing her saying "*I'm gonna be a* ***mom****!!* " Meanwhile, it is also possible to infer *User* ***c*** is more likely to be a *female* through analyzing her social link since she follows a cosmetic-selling *User* "***Dior***".

Although supervised methods have achieved remarkable success for gender classification, their good performances always depend on a large amount of labeled data, which often need expensive labor costs and long production time. How to learn a classification model with low dependence on the large-scale labeled data becomes an important and challenging problem in gender classification.

In this paper, we propose a semi-supervised learning approach to alleviate the above problem in supervised gender classification. Instead of using a large scale of labeled data, we exploit a small scale of labeled data and large amount of unlabeled data to train the model. Our semi-supervised approach employs both user-generated textual knowledge and user social link information. The basic motivation of our approach lies in the observation that social link information might be helpful to infer user gender. Specifically, we focus on the "*following*" link and think that two users who follow the same particular user could have the same gender. For instance, in Figure 1, *User* ***b***, *User* ***c*** and *User* ***d*** follow the same user named ***Dior*** and they are thought to be indirectly linked. Once *User* ***b*** and *User* ***c*** are correctly classified to be *female* with textual features, *User* ***d*** is more likely to be *female* since she is indirectly linked to *User* ***b*** and *User* ***c***.

* Corresponding author

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2092–2100, Osaka, Japan, December 11-17 2016.

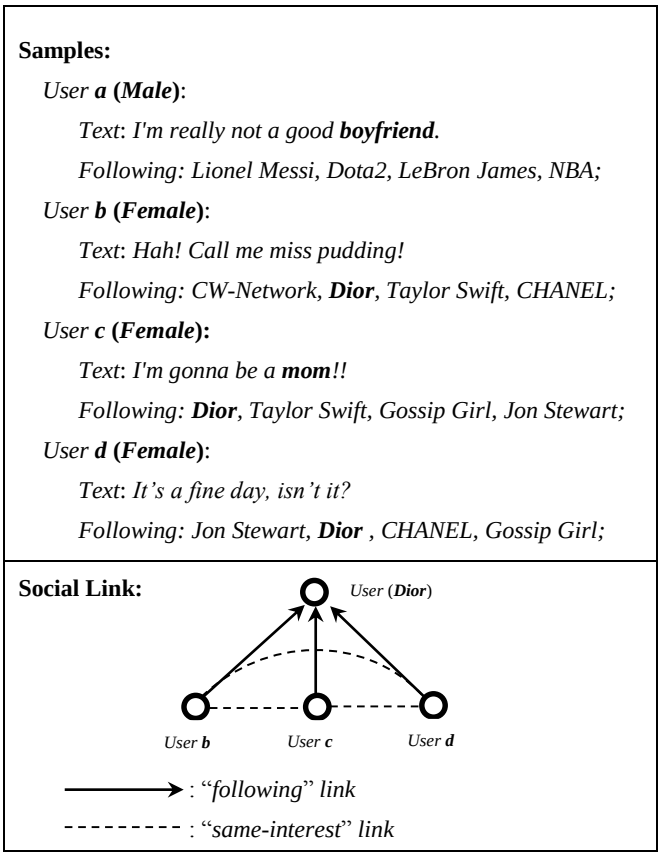


Figure 1: An example of Text and concerns in social media

Specifically, we propose a factor graph, namely Textual and Social Factor Graph (TSFG), to model both the textual and user social link information. Here, a social link between two users happens when the two users follow the same user. For instance, in Figure 1, *User **b*** and *User **c*** both follow the user named ***Dior***. These two users are thought to be linked with an indirect link, called "*same-interest*" link. In our TSFG approach, both the textual features and social links are modeled as various factor functions and the learning task aims to maximize the joint probability of all these factor functions. Empirical evaluation demonstrates the effectiveness of our TSFG approach to capture the inherent user social link. To the best of our knowledge, this work is the first attempt to incorporate both the textual and social information in semi-supervised gender classification.

The remainder of this paper is organized as follows. Section 2 overviews related work on gender classification. Section 3 introduces data collection and analysis. Section 4 describes our TSFG approach to gender classification. Section 5 presents the experimental results. Finally, Section 6 gives the conclusion and future work.

2 Related Work

In the last decade, gender classification has been studied in two main aspects: supervised learning and semi-supervised learning.

As for supervised learning, gender classification has been extensively studied in several textual styles, such as Blog (Nowson and Oberlander, 2006; Peersman *et al.*, 2011; Gianfortoni *et al.*, 2011), E-mail (Mohanmad *et al.*, 2011), YouTube (Filippova, 2012) and Micro-blog (Rao *et al.*, 2010; Liu et al., 2013). These studies mainly focus on employing various kinds of textual features such as character, word, POS features and their *n*-gram features to train the classifier. More recently, some studies focus on some specific application scenarios on supervised gender classification, such as multi-lingual gender classification (Ciot *et al.*, 2013; Alowibdi *et al.*, 2013) and interactive gender classification (Li *et al.*, 2015).

As for semi-supervised learning, gender classification has been studied with much less previous studies. Ikeda *et al.* (2008) propose a semi-supervised approach to gender classification in blog. Their

main idea is to utilize a sub-classifier to measure the relative similarity between two blogs so as to capture the classification knowledge in the unlabeled data. More recently, Burger *et al.* (2011) mention the importance of using unlabeled data and directly apply a self-training approach to perform semi-supervised learning for gender classification. Wang *et al.* (2015) employ both non-interactive and interactive texts as two different views in their co-training approach for semi-supervised gender classification.

Unlike the studies above, our study focuses on both textual features and social links for semi-supervised gender classification.

3 Data Collection and Analysis

The data is collected from Sina Micro-blog[2], the most famous Micro-blogging platform in China. In this platform, local users publish short messages and are allowed to follow other users to listen to their messages. From the website, we crawl each user's homepage which contains the user information (e.g. Name, gender, and, verified type), messages and following users. The data collection process starts from some randomly selected users, and then iteratively gets the data of their followers and followings. We remove some unsuitable users that meet one of the following two conditions: (1) verified organizational users that are verified as organization; (2) the non-active users that have less than 50 followers or 50 followings.

In total, we obtain about 10000 user homepages, from which we randomly select a balanced data set containing 1000 male and 1000 female users. Let $Fo(u_i)$ denotes the set of u_i 's all "following" users; F_{male} denotes the set of all male users' "following" users; F_{female} denotes the set of all female users' "following" users. F_{male} and F_{female} can be calculated as following:

$$F_{male} = \bigcup_{u_i \in S_{male}} Fo(u_i) \tag{1}$$

$$F_{male} = \bigcup_{u_i \in S_{female}} Fo(u_i) \tag{2}$$

Where S_{male} and S_{female} denote the sets of male and female users respectively.

Table 1 shows the statistics about the numbers of "following" users of all male and female users. From this table, we can see that there are many users who are only followed by male users or female users. Specifically, in our data set, 143389 users are followed by only male users and 119504 users are followed by only female users. Thus, these gender-sensitive followings are good clues to infer each user's gender.

	#of "*following*" users
$\lvert F_{male} \rvert$	162116
$\lvert F_{female} \rvert$	138231
$\lvert F_{male} \cap F_{female} \rvert$	18727
$\lvert F_{male} \rvert - \lvert F_{male} \cap F_{female} \rvert$	143389
$\lvert F_{female} \rvert - \lvert F_{male} \cap F_{female} \rvert$	119504

Table 1: Statistics of the following users

4 Textual and Social Factor Graph Model

A factor graph consists of two layers of nodes, i.e., variable nodes and factor nodes, with links between them. The joint distribution over the whole set of variables can be factorized as a product of all factors.

[2] http://weibo.com/

4.1 Model Definition

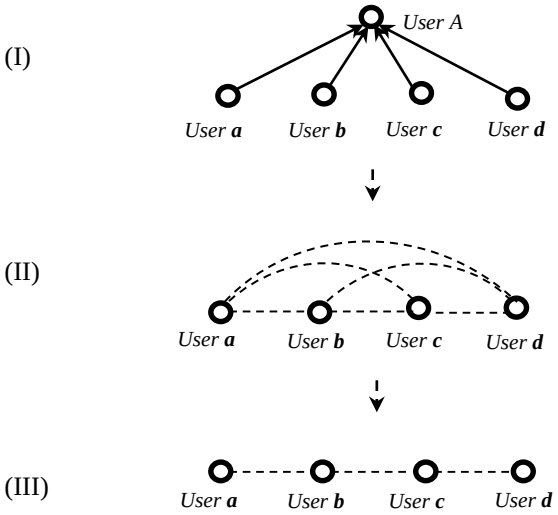


Figure 2: An example for illustrating the user links where

(I) shows the "following" links among all users;
(II) shows the "same-interest" links among the four users;
(III) shows the simplified four "same-interest" links among the four users.

Formally, let $G=(V,E)$ represent an instance network, where V denotes a set of the involved users in our data set. $E \subset V \times V$ is a set of relationships between users. Specifically, if a user u_i and a user u_j have the same following (i.e., the same-interest link), there is an edge $e_{ij}, e_{ij} \in E$, linking the two users u_i and u_j.

The "following" link: If a user u_i follows another user u_j, there is a "following" link between u_i and u_j. For instance, Figure 2(I) shows an example where four users, namely *User a*, *User b*, *User c*, and *User d*, are in our data set and each of them follows *User A*. Thus there are four "following" links among these five users.

The "same-interest" link: If a user u_i and a user u_j follows the same user, there is a "same-interest" link between u_i and u_j. For instance, Figure 2(II) shows six "same-interest" links among the four users, i.e., *User a*, *User b*, *User c*, and *User d*. The "same-interest" links derived from "following" links as showed in Figure 2(I).

Suppose that there are N users who have the same interest, the number of the same-interest links is C_N^2. However, when N is large, the number of the links is too large, which might make our factor graph model difficult to learn. Therefore, we simplify the link model by deleting $C_N^2-(N-1)$ links, only reserving a link line containing N-1 links, as shown in Figure 2(III).

We model the above network with a factor graph and our objective is to infer the gender categories of instances by learning the following joint distribution:

$$P(Y|G)=\prod_i \prod_k f(X_i,y_i) h_k(y_i,H(y_i)) \tag{3}$$

Where two kinds of factor functions are used.

1) Textual feature factor function: $f(X_i,y_i)$ denotes the traditional textual feature factor functions associated with each text representation of the user u_i, i.e., X_i. The textual feature factor function is instantiated as follows:

$$f(X_i,y_i)=\frac{1}{Z_1}\exp\left(\sum_j \alpha_j \Phi(x_{ij},y_i)\right) \tag{4}$$

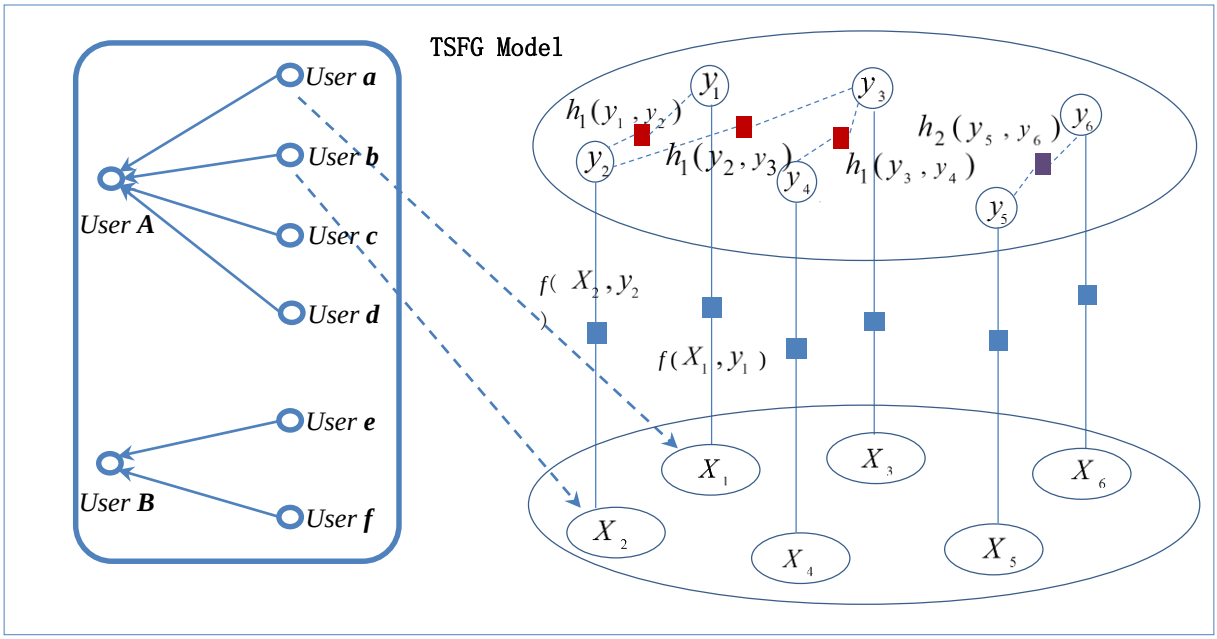


Figure 3: An example of TSFG where six instances are involved:

User ***a****, User* ***b****, User* ***c****, User* ***d****, User* ***e****, and User* ***f****.*

Note: each instance is represented as X_i . $f(\cdot)$ represents a factor function for modeling textual features. $h(\cdot)$ represents a factor function for modeling the "*same-interest*" link between two instances.

Where $\Phi(x_{ij}, y_i)$ is a feature function and x_{ij} represents a textual feature, i.e., a word feature in this study.

2) Social link factor function: $h_k(y_i, H(y_i))$ denotes the "same-interest" relationship among the users who follow the same user $u_k, u_k \in F_{male} \bigcup F_{female}$. $H(y_i)$ is the label set of the users linked to y_i. The social link factor function is instantiated as follows:

$$h_k(y_i, H(y_i)) = \frac{1}{Z_2} \exp\left\{ \sum_{y_i^l \in H(y_i)} \beta_{ikl} \left(y_i - y_i^l \right)^2 \right\} \tag{5}$$

Where β_{ikl} is the weight of the function, representing the degree if influence of the two instances y_i and y_i^l.

Figure 3 gives an example of our textual and social factor graph (TSFG) where six users, i.e., *User* ***a****, User* ***b****, User* ***c****, User* ***d****, User* ***e****,* and *User* ***f****,* are involved.

4.2 Model Learning

Learning the DFG model is to estimate the best parameter configuration $\theta = (\{\alpha\}, \{\beta\})$ to maximize the log-likelihood objective function $L(\theta) = \log P_\theta(Y|G)$, i.e.,

$$\theta^* = \arg\max L(\theta) \tag{6}$$

In this study, we employ the gradient decent method to optimize the objective function. For example, we can write the gradient of each α_j with regard to the objective function:

$$\frac{\partial L(\theta)}{\partial \alpha_j} = E\left[\Phi(x_{ij}, y_i)\right] - E_{P_{\alpha_j}(Y|G)}\left[\Phi(x_{ij}, y_i)\right] \tag{7}$$

Where $E\left[\Phi(x_{ij}, y_i)\right]$ is the expectation of feature function $\Phi(x_{ij}, y_i)$ given the data distribution. $E_{P_{\alpha_j}(Y|G)}\left[\Phi(x_{ij}, y_i)\right]$ is the expectation of feature function $\Phi(x_{ij}, y_i)$ under the distribution $P_{\alpha_j}(Y|G)$

given by the estimated model. Figure 4 illustrates the detailed algorithm for learning the parameter α. Note that LBP denotes the Loopy Belief Propagation (LBP) algorithm which is applied to approximately infer the marginal distribution in a factor graph (Frey and MacKay, 1998). A similar gradient can be derived for the other parameters.

Input: Learning rate η

Output: Estimated parameters α

Initialize $\alpha \leftarrow 0$

Repeat

1) Calculate $E\left[\Phi\left(x_{ij}, y_i\right)\right]$ using LBP
2) Calculate $E_{P_{\alpha_{ij}}(Y|G)}\left[\Phi\left(x_{ij}, y_i\right)\right]$ using LBP
3) Calculate the gradient of α according to Eq. (7)
4) Update parameter α with the learning rate η

$$\alpha_{\text{new}} = \alpha_{\text{old}} + \eta \frac{L(\alpha)}{\alpha}$$

Until *Convergence*

Figure 4: The learning algorithm for TSFG model

It is worth noting that we need to perform the LBP process twice for each iteration: One is to estimate the original distribution of unlabeled instances which are denoted as $y_i = ?$ and the other is to estimate the marginal distribution over all pairs. In this way, the algorithm essentially leverage both the labeled data and unlabeled data to optimize the complete network.

4.3 Model Prediction

With the learned parameter configuration θ, the prediction task is to find a Y^{T*} which optimizes the objective function, i.e.,

$$Y^{T*} = \arg\max P\left(Y^T \middle| Y^{L+U}, G, \theta\right) \tag{8}$$

Where Y^{T*} are the labels of the instances in the testing data and Y^{L+U} are the labels (or estimated labels) of the instances in the labeled and unlabeled data.

Again, we utilize LBP to calculate the marginal probability of each instance $P\left(y_i \middle| Y^{L+U}, G, \theta\right)$ and predict the label with the largest marginal probability. For all instances in the test data, the prediction indicated above is performed iteratively until converge.

5 Experimentation

We have systematically evaluated our TSFG approach to semi-supervised gender classification.

5.1 Experimental Settings

Data Setting: The data set contains 2000 users, as described in Section 3. From this data set, we select 200 users as initial labeled data, 1400 users as unlabeled data, and the remaining 400 users as the test data.

Features: Three types of textual features, including bag-of-words, f-measure, and POS pattern features, are adopted in our experiments. These features yield the state-of-the-art performance in gender classification (Mukherjee and Liu, 2010). To get word and POS features, we use the toolkit ICTCLAS[3] to perform word segmentation and POS tagging on the Chinese text.

[3] http://www.ictclas.org/ictclas_download.aspx

Classification Algorithm: For supervised learning, various of classification algorithms are available. As suggested by Li *et al.* (2015), we apply maximum entropy (ME) for supervised gender classification. Specifically, the ME algorithm is implemented with the Mallet Toolkit[4]. For semi-supervised learning, we implement our TSFG approach, together with some baselines.

Evaluation Measurement: The performances are evaluated using the standard *precision*, *recall*, and *F-score* in each gender category. For overall evaluation, we use macro-average *F-score* over both gender categories, which is denoted as F_{macro}.

Significance test: *T*-test is used to evaluate the significance of the performance difference between two approaches (Yang and Liu, 1999).

5.2 Experimental Results

For thorough comparison, several gender classification approaches are implemented including:

- **Baseline(Textual)**: employing ME classifier and textual features with only initial labeled data (without any unlabeled data).
- **Baseline(Textual+Social)**: employing ME classifier and both textual and social features with only initial labeled data (without any unlabeled data). Social features are extracted by considering each user ID of the followers of a user as a word.
- **Self-training(Textual)**: employing ME classifier and textual features with both labeled data and unlabeled data using self-training.
- **Self-training(Textual+Social)**: employing ME classifier and both textual and social features with both labeled data and unlabeled data using self-training.
- **Co-training(Textual)**: employing ME classifier and textual features with both labeled data and unlabeled data using co-training. We implement the co-training algorithm by randomly splitting the feature space into two disjoint feature subspaces as two views (Nigam and Ghani, 2000).
- **Co-training(Textual+Social)**: employing ME classifier and both textual and social features with both labeled data and unlabeled data using co-training. We implement the co-training algorithm by randomly splitting the feature space into two disjoint feature subspaces as two views (Nigam and Ghani, 2000).
- **TSFG:** our approach as described in Section 4.

	Male			*Female*			*Total*
Approach	*Precision*	*Recall*	*F-score*	*Precision*	*Recall*	*F-score*	F_{macro}
Baseline(Textual)	0.760	0.650	0.700	0.694	0.795	0.741	0.721
Baseline(Textual+Social)	0.800	0.700	0.747	0.733	0.825	0.776	0.762
Self-Training(Textual)	0.714	0.710	0.711	0.711	0.715	0.713	0.712
Self-Training(Textual+Social)	0.754	0.735	0.744	0.741	0.760	0.751	0.747
Co-Training(Textual)	0.725	0.700	0.712	0.710	0.735	0.722	0.717
Co-Training(Textual+Social)	0.784	**0.745**	0.764	0.757	0.795	0.776	0.770
TSFG	**0.961**	0.735	**0.833**	**0.785**	**0.970**	**0.868**	**0.851**

Table 2: Performance comparison of different approaches to semi-supervised gender classification

Table 2 shows the performance comparison of different approaches to gender classification. From this table, we can see that:

(1) Social BOW features are helpful in both supervised and semi-supervised learning approaches.
(2) Self-training fails to exploit unlabeled data to improve the performance and it performs even worse than the baseline approaches.

[4] http://mallet.cs.umass.edu/

(3) Co-training is effective for semi-supervised gender classification when both textual and social features are employed. This result indicates that the use of social features in semi-supervised gender classification in co-training is beneficial, although the improvement is rather limited, about 1%.

(4) Our approach TSFG performs best among all semi-supervised learning approaches. Moreover, the improvement over the two baselines is remarkable, 13% higher than Baseline(Textual) and 8.9% higher than Baseline(Textual+Social). Significance test shows that our approach significantly outperforms co-training (p-value<0.01)

Figure 5 shows the performances of our approach and the two baseline approaches when varying the sizes of the initial labeled data. From this figure, we can see that social features are always helpful for gender classification and Baseline(Textual+Social) consistently outperforms Baseline(Textual). Our approach fails to take effect when the size of the initial labeled data is too small (10 labeled instances in each category). When the size of the initial data is larger than 20 instances in each category, our TSFG approaches consistently performs much better than the two baseline approaches. Significance test shows that our TSFG approach significantly outperforms both Baseline(Textual) and Baseline(Textual+Social) when the size of the initial labeled instance is larger than 20 in each gender category (p-value<0.01).

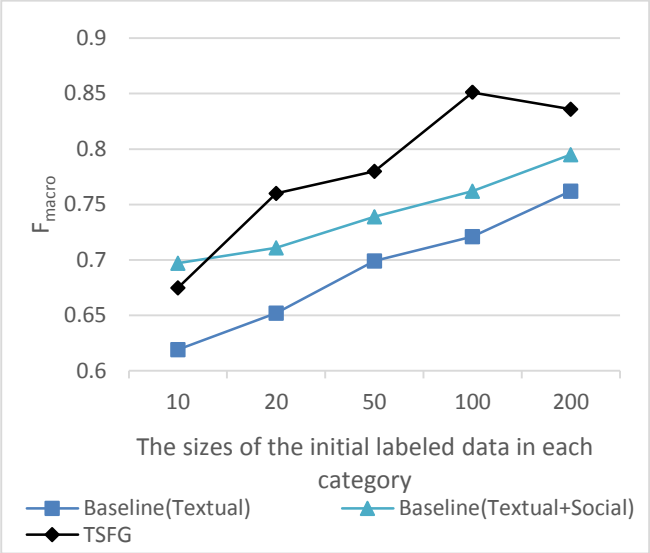


Figure 5: The performances of our approach and the two baseline approaches when varying the sizes of the initial labeled data.

6 Conclusion

In this paper, we propose a novel approach to semi-supervised gender classification in social media. In our approach, we first define a social link named "*same-interest*" link which models an indirect link between two users who follow the same user. Then, we propose a factor graph-based approach, namely Textual and Social Factor Graph (TSFG), where both the textual features and "*same-interest*" social links are modeled as various factor functions. Finally, we employ the graph to leverage both the labeled data and unlabeled data to optimize the complete network. Empirical studies show that our TSFG approach successfully exploits unlabeled data to improve the performance, remarkably outperforming other semi-supervised learning approaches.

In our future work, we would like to improve our semi-supervised learning approach by leveraging some other kinds of link information. Furthermore, we will apply our TSFG approach to some other NLP tasks where both textual and social features are available, such as user age prediction (Rosenthal and McKeown, 2011) and user occupation classification (Preotiuc-Pietro *et al.*, 2015).

Acknowledgements

This research work has been partially supported by five NSFC grants, No.61273320, No.61375073, No.61331011, No.61305088 and No.61373096.

Reference

Burger J. and J. Henderson and G. Kim and G. Zarrella. 2011. Discriminating Gender on Twitter. In *Proceedings of EMNLP-11*, pp. 1301–1309.

Corney M., O. Vel, A. Anderson and G. Mohay. 2002. Gender-Preferential Text Mining of E-mail Discourse. In *Proceedings of ACSAC-02*, pp. 282-289.

Ciot M., M. Sonderegger and D. Ruths. 2013. Gender Inference of Twitter Users in Non-English Contexts. In *Proceedings of EMNLP-13*, pp. 1136–1145.

Filippova K. 2012. User Demographics and Language in an Implicit Social Network. In *Proceedings of EMNLP-12*, pp. 1478–1488.

Frey B. and D. MacKay. 1998. A Revolution: Belief Propagation in Graphs with Cycles. In *Proceedings of NIPS-98*, pp.479–485.

Gianfortoni P., D. Adamson and C. Rosé. 2011. Modeling of Stylistic Variation in Social Media with Stretchy Patterns. In *Proceedings of EMNLP-11*, pp. 49–59.

Ikeda D., H. Takamura and M. Okumura. 2008. Semi-Supervised Learning for Blog Classification. In *Proceedings of AAAI-08*, pp.1156-1161.

Li S., J. Wang, G. Zhou and H. Shi. 2015 Interactive Gender Inference with Linear Programming. In *Proceedings of IJCAI-15*, pp. 2341-2347.

Liu N., Y. He, Q. Chen, M. Peng and Y. Tian. 2013. A New Method for Micro-blog Platform Users Classification Based on Infinitesimal-time. *Journal of Information & Computantional Science*. 10:9 (2013) 2569–2579.

Mukherjee A. and B. Liu. 2010. Improving Gender Classification of Blog Authors. In *Proceedings of EMNLP-10*, pp. 207-217.

Mohammad S. and T. Yang. 2011. Tracking Sentiment in Mail: How Genders Differ on Emotional Axes. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis(2011)*, pp.70-79.

Nigam, K. and R. Ghani. 2000. Analyzing the Effectiveness and Applicability of Co-training. In *Proceedings of CIKM-2000*, 86-93.

Nowson S. and J. Oberlander. 2006. The Identity of Bloggers: Openness and Gender in Personal Weblogs. In *Proceeding of AAAI-06*, pp. 163-167.

Peersman C., W. Daelemans, L. Van Vaerenbergh. 2011. Predicting Age and Gender in Online Social Networks. In *Proceedings of SMUC-11*, pp. 37-44.

Preotiuc-Pietro D., V. Lampos and N. Aletras. 2015. An Analysis of the User Occupational Class through Twitter Content. In *Proceedings of ACL-15*, pp. 1754-1764.

Rao D., D. Yarowsky, A. Shreevats and M. Gupta. 2010. Classifying Latent User Attributes in Twitter. In *Proceeding SMUC '10 Proceedings of the 2nd international Workshop on Search and Mining User-Generated Contents*, pp. 37-44.

Rosenthal S. and K. McKeown. 2011. Age Prediction in Blogs: A Study of Style, Content, and Online Behavior in Pre- and Post-Social Media Generations. In *Proceedings of ACL-11*, pp.763-772.

Volkova S., T. Wilson and D. Yarowsky. 2013. Exploring Demographic Language Variations to Improve Multilingual Sentiment Analysis in Social Media. In *Proceedings of EMNLP-13*, pp. 1815–1827.

Wang J., Y. Xue, S. Li and G. Zhou. 2015 Leveraging Interactive Knowledge and Unlabeled Data in Gender Classification with Co-training. In *Proceedings of DASFAA-15*, pp. 246-251.

Yan X. and L. Yan. 2006. Gender inference of Weblog Authors. In *Proceedings of AAAI-06*, pp.228-230.

Yang Y. and X. Liu. 1999. A Re-Examination of Text Categorization Methods. In *Proceedings of SIGIR-99*, pp. 42-49.

Predicting proficiency levels in learner writings by transferring a linguistic complexity model from expert-written coursebooks

Ildikó Pilán, Elena Volodina
Språkbanken, University of Gothenburg
Sweden
ildiko.pilan@svenska.gu.se
elena.volodina@svenska.gu.se

Torsten Zesch
Language Technology Lab
University of Duisburg-Essen
Germany
torsten.zesch@uni-due.de

Abstract

The lack of a sufficient amount of data tailored for a task is a well-recognized problem for many statistical NLP methods. In this paper, we explore whether data sparsity can be successfully tackled when classifying language proficiency levels in the domain of learner-written output texts. We aim at overcoming data sparsity by incorporating knowledge in the trained model from another domain consisting of input texts written by teaching professionals for learners. We compare different domain adaptation techniques and find that a weighted combination of the two types of data performs best, which can even rival systems based on considerably larger amounts of in-domain data. Moreover, we show that normalizing errors in learners' texts can substantially improve classification when in-domain data with annotated proficiency levels is not available.

1 Introduction

Data sparsity is a recognized problem in many machine learning based NLP approaches since the creation of data specifically collected and annotated for a certain task or language is time-consuming and costly. Previous attempts to overcome data sparsity include transferring knowledge between different types of data through the application of models from languages and tasks where sufficient data exists to the ones where data is unavailable or sparse (Daumé III and Marcu, 2006). A common case of such a transfer learning scenario is *domain adaptation*, where training and test data belong to different domains (e.g. text genres) referred to as *source domain* and *target domain* respectively.

In our experiments, we aim at exploring the plausibility of domain adaptation as a strategy for overcoming data sparsity in the context of foreign and second language (L2) learning. More specifically, we operationalize *domain* as the type of text involved in the language learning process: on the one hand, texts from coursebooks intended for L2 learners (referred to as *L2 input texts* in this paper), and on the other hand, essays created by learners (*L2 output texts*). Our goal is to predict L2 language development stages in terms of linguistic complexity in the latter category, i.e. learner-produced texts. These stages are commonly referred to as *proficiency levels* in second language acquisition and language testing. Levels range from 'absolute beginner' to 'advanced language user' with increasing linguistic complexity as learners progress with the levels. A scale of such levels, very influential both in Europe and outside, is the CEFR – Common European Framework of Reference for Languages (Council of Europe, 2001).

In previous work, NLP methods have been successfully applied to both assessing proficiency levels in L2 input texts collected from coursebooks and output texts written by learners (see section 2). However, the two text types have always been considered separately, while we argue that there is a shared linguistic content between the two that can be used for knowledge transfer. Specifically, the output of learners is a subset of the linguistic input that they are able to understand (Barrot, 2015). Thus, incorporating knowledge from coursebook texts representing L2 input may improve the classification of proficiency levels in L2 output text. Decreasing the need for a large amount of L2 output data is particularly appealing since acquiring this type of text poses a number of challenges including copyright issues, anonymization of sensitive information, and often even digitizing hand-written material (Megyesi et al.,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2101–2111, Osaka, Japan, December 11-17 2016.

2016; Mendes et al., 2016; Volodina et al., 2016). Since an increasing amount of people learn foreign languages worldwide either out of necessity or as a personal interest, systems targeting the needs of this user group are especially valuable. Within this context, the automatic assessment of proficiency levels in learner-produced texts would be a powerful tool for increasing both learners' autonomy and teaching professionals' efficiency.

Research Questions In particular, this paper aims at answering the following research questions: (i) Can we overcome the lack of a sufficient amount of learner output data by incorporating knowledge from L2 input texts when performing proficiency level classification? (ii) What kind of domain adaptation technique performs best in this context? (iii) Does normalizing errors in L2 learner output benefit proficiency level classification in a domain adaptation setting?

The motivation behind error normalization is that learner output typically contains errors which may influence the performance of automatic taggers and parsers and thus, classification performance. Therefore, error normalization may allow for a more precise calculation of feature values and a more successful transfer from and to a non error-prone domain. The amount and type of errors, i.e. degree of incorrectness, however, is not explicitly considered as an indicator of proficiency for L2 learner output in our experiments in order to keep comparability with coursebook texts. Unlike linguistic complexity, incorrectness is not a relevant aspect for L2 input texts as these are authored by teaching professionals and are supposed to be relatively error-free examples of language use.

Our target language of choice is Swedish, a language considerably less resource-rich than English and for which a CEFR-level classification model of L2 learners' writing is not available yet, despite the clear need for breaking down CEFR descriptors into linguistic constituents that characterize proficiency levels for each individual language (Little, 2011; North, 2007).

Main Findings We find that, in the absence of annotated learner-written data, using a classification model trained only on coursebook texts is a viable alternative if learner errors are normalized. Furthermore, if a small amount of learner output data is available and it is combined with L2 input texts, it can even outperform a model trained only on the few in-domain instances, resulting in a prediction quality matching that of in-domain state-of-the-art systems for other languages. In a domain adaptation setting, normalizing learner errors proved to yield a substantial improvement for features based on token, character and sentence counts as well as for features based on the CEFR-level distribution of tokens.

2 Text Categorization in the Language Learning Context

The automated evaluation of learner output is primarily a text classification task which aims at determining the quality of writing and assigning an appropriate label from a given set, for example a score or grade on the continuum between pass-fail (*essay scoring*) or a level indicating learning progress (*proficiency level classification*). In a L2 learning scenario, a longer piece of learner-written text is a popular means to assess learners' proficiency level. The human assessment of learner output, however, is both time-consuming and prone to subjectivity. Different linguistic dimensions need to be taken into consideration usually requiring several iterations of re-reading and different factors may influence the decision, such as negative attitude to a learner, hunger, bad mood, and boredom. Therefore, the number of initiatives to complement (or even replace) human assessment with a more objective and more efficient supervised machine learning system has been increasing the past years, with essay grading (Burstein and Chodorow, 2010) as an important application field.

2.1 Automatic Essay Scoring

Automatic essay scoring (AES) has been an active research area since 1990s, targeting mostly English (Burstein and Chodorow, 2010; Miltsakaki and Kukich, 2004; Page, 2003). Recently, with the availability of annotated learner corpora for other languages, automatic essay grading has expanded to cover also other languages, e.g. German (Zesch et al., 2015) and Swedish (Östling et al., 2013), to name just a few.

In its nature, AES has mostly relied on machine learning approaches, exploring both supervised (Yannakoudakis et al., 2011) and unsupervised methods (Chen et al., 2010) with different degrees of success.

Östling et al. (2013) have looked at Swedish upper secondary school essays, i.e. first language learner essays, and automatically assessed them in terms of a four-point scale of performance grades with an accuracy of 62%. The authors found that this result exceeded the agreement rate between two human assessors which was as low as 45.8% which might indicate that human-like performance is a rather uncertain goal. Linguistic parameters that have over time been presumed to be strong predictors of writing quality have varied from shallow ones like text and word length (Page, 2003; Östling et al., 2013) to more sophisticated features using Latent Semantic Analysis (Landauer et al., 2003), cosine similarity (Attali and Burstein, 2006), discourse structure and stylistic features (Attali and Burstein, 2006).

2.2 Proficiency Level Classification

A closely related task to AES is classifying texts into L2 proficiency levels which consists of predicting at which language learning stage a text can be produced or understood by a L2 learner, rather than assigning a grade within a pass-fail range. The CEFR, the scale of proficiency levels adopted in our experiments, contains guidelines for the standardization of language teaching and assessment across languages and countries (Council of Europe, 2001). It provides a common metalanguage to talk about objectives, assessment, (Little, 2011), and it defines language competences at six proficiency levels (A1, A2, B1, B2, C1, C2) where A1 is the beginner level. Since the publication of the CEFR guidelines in 2001, several countries have adopted the system, but its practical application has proven to be rather non-straightforward since the descriptions of the competences at each level remain vague (Little, 2011; North, 2007).

The past few years have seen an increasing interest in the CEFR-level classification of both L2 input and output texts. In the case of coursebook texts such a classification has also been referred to as *L2 readability* and it has been investigated for, among others, French (François and Fairon, 2012), Portuguese (Branco et al., 2014), Chinese (Sung et al., 2015), Swedish (Pilán et al., 2015), and English (Xia et al., 2016).

Apart from L2 input texts, CEFR-level annotated L2 learner corpora are also available for a number of languages including but not limited to English (Nicholls, 2003), Estonian (Vajjala and Lõo, 2014) and German (Hancke and Meurers, 2013). Moreover, MERLIN (Wisniewski et al., 2013) is a trilingual learner corpus comprised of written productions of L2 learners of Czech, German, and Italian also linked to CEFR levels. Despite the availability of annotated corpora for several languages, the number of projects targeting the automatic CEFR-level classification of learner essays has remained rather limited. Previously reported results for this task in terms of accuracy include 61% for German (Hancke and Meurers, 2013) and 79% for Estonian (Vajjala and Lõo, 2014).

2.3 Domain Adaptation for Tasks Related to L2 Learning

While there is a lot of previous work on domain adaptation in general, relatively few approaches exist in the field of assessing learner output texts. Previous applications of domain adaptation to learner essays focused on exploring the transfer of models between different writing tasks that prompted students to produce the essays, e.g. expressing an opinion on a topic vs. summarizing a news article (Zesch et al., 2015; Phandi et al., 2015). Zesch et al. (2015) explore which features are transferable from one essay grading task to another task based on a different prompt. They find that by excluding some highly domain-specific features, the transfer loss can be reduced significantly without noticeable differences in overall performance.

A popular domain adaptation approach is EASYADAPT (Daumé III, 2007) that augments the original feature space with source- and target-specific versions. Phandi et al. (2015) successfully applied EASYADAPT for automatic essay scoring and Xia et al. (2016) for the CEFR-level classification of L2 input texts with native language texts as source domain.

3 Datasets

For our experiments, we use L2 Swedish data including learners' output, i.e. error-prone essays written by learners, as well as L2 input data for learners, i.e. relatively error-free texts written by experts for

		CEFR Levels				
		A2	**B1**	**B2**	**C1**	**Total**
Learner Output	**Texts**	83	75	74	88	**320**
	Tokens	18,349	29,814	32,691	60,095	**140,949**
Expert Input	**Texts**	157	258	288	115	**818**
	Tokens	37,168	79,124	101,297	71,723	**289,312**

Table 1: Overview of CEFR-level annotated Swedish datasets.

L2 learners primarily intended as reading material. Both types of data are manually labeled for CEFR levels and automatically annotated across different linguistic dimensions including lemmatization, part-of-speech (POS) tagging, and dependency parsing using the Sparv (previously known as 'Korp') pipeline (Borin et al., 2012).

3.1 L2 Output Texts

Our source of output texts is SweLL (Volodina et al., 2016), a corpus consisting of L2 Swedish learner essays on a variety of topics, manually linked to CEFR levels. The essays also contain meta-information on learners' mother tongue(s), age, gender, education level, the exam setting, and, in certain cases, topic and genre. The distribution of essays per level is given in Table 1.

The corpus includes some essays at A1 and C2 levels, but these classes were too under-represented to be included in our experiments. As for A1 level, this may depend on learners' limited ability to write due to the lack of familiarity with many linguistic constructs. In fact, the CEFR contains no descriptor for writing essays and reports at A1 level (Council of Europe, 2001, 62). C2 is lacking since courses at this level are not provided, and it is in general characterized as a near-native language competence.

Since SweLL consists of learner-produced texts, it is likely that it contains some errors which, however, have not been annotated or normalized yet in the resource. The number of non-lemmatized tokens in the resource (i.e. tokens that could not be assigned baseforms during automatic annotation), which could indicate spelling errors or creative compounding at more advanced levels is higher at lower proficiency levels, but their amount always remains within a range of 5% and 8%.

3.2 L2 Input Texts

Our L2 input texts were collected from COCTAILL, a corpus of coursebooks used for teaching CEFR-based courses of L2 Swedish (Volodina et al., 2014). The coursebooks are divided into lessons (book chapters), each of which is labeled for the CEFR level it is aimed at. Each lesson contains a variety of elements including reading texts, exercises, lists, etc. Out of these only the texts intended for reading have been included in our dataset, whose CEFR level was derived from the level of the lesson they occurred in. Table 1 gives an overview of the distribution of these texts per level. For the same reasons as in section 3.1, C2 was not included in this dataset and A1 level has been omitted to keep the classes consistent between the two datasets.

4 Feature Set

We use the feature set presented in Pilán et al. (2015) designed for modeling linguistic complexity in input texts for L2 Swedish learners. These features rely on morpho-syntactic tags, information about the CEFR level of tokens, and aspects inspired by L2 Swedish curricula. Five sub-group of features can be distinguished in this set: length-based, (weakly) lexical, morphological, syntactic, and semantic features. The detailed list of features is presented in Table 2.

Count-based features rely on the number of characters and tokens (*tkn*), extra-long words being tokens longer than 13 characters. LIX (Läsbarhetsindex) is a traditional Swedish readability formula corresponding to the sum of the average number of words per sentence in the text and the percentage of

Count	Lexical	Syntactic	Morphological	
Sentence length	A1 lemma IS	Avg DepArc length	Modal V to V	Verb IS
Avg token length	A2 lemma IS	DepArc Len > 5	Particle IS	V variation
Extra-long token	B1 lemma IS	Max length DepArc	3SG pronoun IS	Function W IS
Nr characters	B2 lemma IS	Right DepArc Ratio	Punctuation IS	Lex tkn to non-lex tkn
LIX	C1 lemma IS	Left DepArc Ratio	Subjunction IS	Lex tkn to Nr tkn
Bilog TTR	C2 lemma IS	Modifier variation	PR to N	Neuter N IS
Square root TTR	Difficult W IS	Pre-modifier IS	PR to PP	CJ + SJ IS
Semantic	Difficult N&V IS	Post-modifier IS	S-VB IS	Past PC to V
Avg senses per token	OOV IS	Subordinate IS	S-V to V	Present PC to V
N senses per N	No lemma IS	Relative clause IS	ADJ IS	Past V to V
	Avg. KELLY log freq	PP complement IS	ADJ variation	Present V to V
			ADV IS	Supine V to V
			ADV variation	Relative structure IS
			N IS	Nominal ratio
			N variation	N to V

Table 2: Feature set.

tokens longer than six characters (Björnsson, 1968). Rather than a simple type-token ratio (TTR), we use a bi-logarithmic and a square root equivalent following Vajjala and Meurers (2012).

Lexical features incorporate information from the KELLY list (Volodina and Kokkinakis, 2012), a frequency-based word list compiled using a corpus of web texts (thus completely independent of our datasets), which also provides a suggested CEFR level per each lemma based on frequency bands. For some feature values, *incidence scores* (IS) are computed, in other words, instead of absolute counts, normalized values per 1000 tokens are considered to reduce the influence of sentence length. Lexical complexity is modeled with a set of weakly lexicalized features, i.e. we do not use word forms or lemmas themselves as features, but the IS of their corresponding CEFR levels instead. This aspect is especially important considering the limited size of our learner essay data. *Difficult* tokens are those that belong to levels above the overall CEFR level of the text. Moreover, we consider the IS of tokens not present in KELLY (OOV IS), the IS of tokens for which the lemmatizer could not identify a corresponding lemma (No lemma IS), as well as average KELLY log frequencies.

Morphological features include not only IS but also variational scores, i.e. the ratio of a category to the ratio of lexical tokens: nouns (N), verbs (V), adjectives (ADJ) and adverbs (ADV). The IS of all lexical categories as well as the IS of punctuation, particles, sub- and conjunctions (SJ, CJ) are taken into consideration. Nominal ratio (Hultman and Westman, 1977) is another readability formula proposed for Swedish that corresponds to the ratio of nominal categories, i.e. nouns, prepositions (PP) and participles to the ratio of verbal categories, namely pronouns (PR) adverbs, and verbs. Relative structures consist of relative adverbs, determiners, pronouns and possessives. Some features are inspired by L2 teaching material (Fasth and Kannermark, 1997) and they are based on fine-grained inflectional information such as the IS of neuter gender nouns and the ratio of different verb forms to all verbs.

Syntactic features are based, among others, on the length (depth) and the direction of dependency arcs (DepArc). Within this feature group, we consider also relative clauses as well as pre- and post-modifiers, which include, for example, adjectives and prepositional phrases respectively.

Semantic features build on information from the SALDO lexicon (Borin et al., 2013). We use the average number of senses per token and the average number of noun senses per nouns.

5 Experimental Setup

For all experiments, we use SVMs as implemented in WEKA (Hall et al., 2009) and the feature set presented in detail in section 4. Results are obtained using 10-fold cross-validation. We report the F_1 score, i.e. the harmonic mean of precision and recall, as well as quadratic weighted kappa (κ^2), a distance-based scoring function taking into consideration also the degree of misclassifications.

Experimental setup	Data used	# Training inst.	# Informing inst.
MAJORITY	D_T	288	320
IN-DOMAIN	D_T	288	320
SOURCE-ONLY	D_S	818	818
EASYADAPT	D_S with augmented features	818	1138
+FEATURE	D_T with D_S prediction as feature	288	1138
COMBINED	D_S + 60% of D_T	1010	1010
WEIGHTED	D_S ($w = 1$) + 60% of D_T ($w = 10$)	1010	1010
WEIGHTED-INSTSEL	Correctly classified D_S ($w = 1$) + 60% of D_T ($w = 10$)	505	1138

Table 3: Domain adaptation experimental setups.

5.1 Domain Adaptation

In a domain adaptation scenario, data from a source domain (D_S) is used to predict labels in a different, target domain (D_T). To overcome data sparsity, especially relevant for our learner essay data, we experiment with improving CEFR level classification by transferring information from our D_S consisting of L2 coursebook texts to D_T consisting of Swedish L2 learners' essays.

As baselines, we employ both assigning the most frequent label in the dataset (MAJORITY) and an IN-DOMAIN setup using only the learner essays in a cross-validation setup. We compare these to different domain adaptation scenarios inspired mostly by Daumé III and Marcu (2006) and Pan and Yang (2010) which differ in the type and the amount of data used as detailed in Table 3. We report the number of instances employed at the moment of training as well as the amount of instances from which information has been incorporated in some form in the final models.

In the **SOURCE-ONLY** setup, a model trained on all available source domain instances, i.e. coursebook texts, was applied directly to the target domain instances consisting of learner essays. **EASYADAPT** (Daumé III, 2007) is a feature augmentation approach which consists of triplicating the feature space by including three versions of each feature in the augmented equivalent: a general, a source-specific and a target-specific version. In more formal terms, to each feature vector x, the mapping function $\phi^S(x) = \langle x, x, 0\rangle$ is applied in the source domain and $\phi^T(x) = \langle x, 0, x\rangle$ in the target domain, 0 being a zero vector of length $|x|$. In **+FEATURE** we first train a model trained on the L2 input texts. Then, the CEFR label predicted by this system is incorporated as an additional feature for each essay instance and a new model is trained on the essays with this extra dimension. For **COMBINED** and **WEIGHTED** the training data includes not only D_S instances, but also 60% of D_T. In the WEIGHTED setup, an increased importance is given to D_T instances during training through the assignment of a higher weight (w). Finally, to obtain **WEIGHTED-INSTSEL**, we first train a model on the available D_T data and use that to classify D_S instances. Then those D_S instances that the essay-only model correctly classified are combined with 60% D_T, the latter ones receiving a weight of 10. Compared to WEIGHTED, in this setup we discard D_S instances that might be misleading when making predictions on D_T, due to differences in the underlying distributions in the two domains. A similar approach is presented in Jiang and Zhai (2007).

5.2 Error Normalization

Besides using learners' output texts in their original form, we investigate also the effects of error normalization on the domain-adapted strategies. By correcting errors we aim at bringing learners' texts closer to the standard language present in the coursebooks. Making the texts belonging to these two different domains more similar to each other may improve the domain-adapted classification performance. Moreover, since the annotation tools used were originally designed for dealing with standard Swedish, error normalization leads to a more reliable tagging and parsing, and hence to more precise feature values in the corrected learner output texts.

Previous error-normalization approaches include, among others, finite state transducers (Antonsen, 2012) and a number of, mostly hybrid, systems created within the CoNLL Shared Task on grammatical error correction for L2 English (Ng et al., 2014).

	ORIGINAL		ERROR-NORMALIZED	
	F_1	κ^2	F_1	κ^2
MAJORITY	.120	.000	.120	.000
IN-DOMAIN	.721	.886	.720	.872
SOURCE-ONLY	.438	.713	.620	.807
EASYADAPT	.503	.681	.533	.741
+FEATURE	.709	.879	**.802**	.864
COMBINED	.733	.863	.726	.885
WEIGHTED	**.747**	**.890**	.779	**.915**
WEIGHTED-INSTSEL	.733	.873	.795	.914

Table 4: Domain adaptation results with and without error normalization.

We use LanguageTool[1] (Naber, 2003), an open-source rule-based proof-reading program available for multiple languages which detects not only spelling, but also some grammatical errors (e.g. inconsistent gender use in inflected forms). We propose a two-step algorithm consisting of first obtaining correction candidates from LanguageTool and then ranking these candidates based on a word co-occurrence measure. As a first step, we identify errors in the learner essays and a list of one or more LanguageTool correction suggestions, as well as the *context*, i.e. the surrounding tokens for the error within the same sentence. When more than one correction candidate is available, as an additional step, we make a selection based on *Lexicographers' Mutual Information* (LMI) scores (Kilgarriff et al., 2004). Here we assume a positive correlation between a correction candidate co-occurring with a context word and being the correct version of the word intended by the learner. We check LMI scores for each LanguageTool correction candidate paired with the lemma of each available noun, verb, and adjective in the context based on a pre-compiled list of LMI scores. We create this list using a Korp API (Borin et al., 2012) providing LMI scores computed based on a customizable set of corpora. We use a variety of modern Swedish corpora totaling to more than 209 million tokens for our list of LMI scores. Only scores for noun-verb and noun-adjective combinations have been included with a threshold of LMI ≥ 50. When available, we select the correction candidate maximizing the sum of all LMI scores for the context words. In the absence of LMI scores for the pairs of correction candidates and context words, the most frequent word form in Swedish Wikipedia texts is chosen as a fallback.

Once correction candidates are ranked, each erroneous token identified by LanguageTool is replaced in the essays by the top ranked correction candidate. The normalized texts are then annotated again and feature values are re-computed.

6 Results and Discussion

Table 4 presents the results of our domain adaptation experiments first without error normalization (*original*) and then with corrected errors (*error-normalized*). In the case of the non-normalized essays, the in-domain baseline obtained using only the small amount of learner output texts in a cross-validation setup is .721 F_1 and .886 κ^2. Compared to this, transferring a model based on coursebook texts directly (SOURCE-ONLY) results in a considerable performance drop (-.283 F_1 and -.173 κ^2). When using the essays in their original, noisy form, the best performing domain adaptation setup is the weighted combination of L2 input and output texts, which outperforms even the in-domain baseline both in terms of F_1 and κ^2.

The obtained domain-adaptation results are comparable to state-of-the-art in-domain systems for other languages, like the system for Estonian described in Vajjala and Lõo (2014) with an F_1 of .78, or the one for German (Hancke, 2013) with .71 F_1 for a feature selected model distinguishing 5 classes. It is worth

[1] www.languagetool.org

	IN-DOMAIN		SOURCE-ONLY (Original)		SOURCE-ONLY (Error-norm.)	
Feature Group	F_1	κ^2	F_1	κ^2	F_1	κ^2
All	.721	.886	.438	.713	.620	.807
Count	.499	.740	.106	-.003	*.335*	***.708***
Lexical	**.625**	**.826**	**.318**	**.507**	**.378**	.626
Syntactic	.511	.665	.118	.066	.106	.030
Morphological	.538	.743	.297	.403	.291	.419
Semantic	.299	.198	.087	.000	.087	.000

Table 5: Performance of individual feature groups.

noting, however, that both of these systems required a considerably (about three times) larger annotated in-domain corpus. This shows that additional coursebook data can benefit the classification of language proficiency levels in learner output texts, especially if only a small amount of annotated in-domain data is available.

Error Normalization Our error-normalization method corrects in total 5,080 errors in the essays which amounts to 3.6% of all tokens in the data. In absence of error-annotated Swedish resources, we manually evaluate the method by inspecting 120 normalized items out of which we find 83 correct, corresponding to 69% accuracy. Out of the normalized tokens, about 87% are categorized as spelling errors by LanguageTool. Moreover, the choice of correction candidate is based on LMI scores in 24% of all cases.

Since our feature set does not target learner errors specifically (to be able to maintain comparability when applied to coursebook text), we do not expect error normalization to influence classification results with IN-DOMAIN. Our experiment results in Table 4 show, in fact, that correcting learner errors does not have any statistically significant effect in the IN-DOMAIN setup, but it does improve performance to a great extent for most domain-adapted cases. This latter would support the hypothesis that correcting spelling and grammatical errors increases the similarity between the target and the source domain. The gain is especially large (+.182 F_1) in the case of the SOURCEONLY setup, which does not rely on annotated essays. EASYADAPT, which has been successfully used in an AES task previously (Phandi et al., 2015), is outperformed by most other domain adaptation methods in our case, independently from error normalization.

In terms of F_1, +FEATURE using the predictions of a classifier trained on the L2 input texts performs best (.802 F_1), however, the degree of misclassifications indicated by κ^2 is smallest with WEIGHTED (.915), as in the case of the essays without error normalization. After error correction, WEIGHTED-INSTSEL achieves approximately the same quality of performance for all measures as the aforementioned two best performing models WEIGHTED and +FEATURE. These all improve over the IN-DOMAIN baseline by about .07 F_1 and .03 κ^2.

These results show that the knowledge transfer from L2 input texts can be substantially boosted by normalizing errors in the learner-produced texts.

Contribution of Feature Groups In the next step, we investigate the contribution of individual feature groups to the classification performance both in- and cross-domain with the SOURCE-ONLY setup which does not presuppose the availability of annotated in-domain data. Results for our ablation test are shown in Table 5.

The most predictive features in- and cross-domain on both the original and on the normalized essays are lexical features measuring the proportion of tokens per CEFR level in the texts. Morphological features also preserve their strong predictive power when transferred between L2 input and output texts. The informativeness of syntactic and count features is very low in the cross-domain setting with the original essays, but the latter category transfers much better after error-normalizing L2 output texts. A

potential explanation could be that error normalization includes also corrections of capitalization and whitespaces which might contribute to an improved detection of sentence boundaries, a central element in most of these features. Lexical features also benefit from error correction, presumably due to a more precise estimation of the CEFR-level distribution of tokens.

Direction of Misclassifications Finally, to investigate whether the transferred coursebook model predicts learner-written texts to be of higher or lower proficiency levels compared to the available annotations, we perform regression using SMO and the SOURCEONLY setup, transforming CEFR levels into numeric values. We use the normalized essays for this purpose since the automatic annotation is presumably more precise in these texts compared to their original version. Predictions within a distance of 0.5 from the numeric value representing the actual CEFR level are considered sufficiently close for being considered correct, thus the amount of errors is computed based only on cases exceeding this margin. The regression model produces .800 correlation and 1.120 RMSE (root mean squared error). We find that 64% of the erroneous predictions consider essays to be of a lower level than they actually are. This could be a data-driven confirmation of the pedagogical observation that learners' output texts are typically of a lower linguistic complexity compared to the L2 input texts written for them within the same CEFR level.

7 Conclusions

In this work we investigated the benefits of using texts from language learning coursebooks to classify proficiency levels in learner-written texts, since the latter type of data is especially costly to collect. Moreover, our experiments provide useful insights into how some simple domain adaptation techniques compare to each other for this task. Training only on source domain data did not yield a successfully transferable model between the L2 input and output texts if errors were not normalized in the learner-produced essays. With such a normalization, however, using only coursebook texts as training data produced a result rather close to what learning only from a small amount of essays did. Joining domains was useful, especially when weighted target domain instances were added to all, or a subset of the coursebook data, and learner errors were normalized. We showed that, with these two steps, it is possible to outperform a model based only on a limited amount of in-domain data. Furthermore, our results are competitive even compared to systems for other languages that make use of a considerably larger amount of in-domain data.

In the future, it would be informative to repeat the experiments for other languages, where we expect similar results. Additional domain adaptation techniques could also be explored for this task, for example, the identification of shared priors and kernel transformations. Alternatives to the current error normalization could be investigated in order to identify a broader range of incorrect tokens more precisely. More reliable error correction methods may yield further improvement to transferring classification models between these domains.

References

Lene Antonsen. 2012. Improving feedback on L2 misspellings-an FST approach. In *Proceedings of the SLTC 2012 workshop on NLP for CALL; Lund; 25th October; 2012*, number 080, pages 1–10. Linköping University Electronic Press.

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater® v. 2. *The Journal of Technology, Learning and Assessment*, 4(3).

Jessie Barrot. 2015. Comparing the linguistic complexity in receptive and productive modes. *GEMA Online Journal of Language Studies*, 15(2):65–81.

Carl Hugo Björnsson. 1968. *Läsbarhet*. Liber.

Lars Borin, Markus Forsberg, and Johan Roxendal. 2012. Korp - the corpus infrastructure of Språkbanken. In *LREC*, pages 474–478.

Lars Borin, Markus Forsberg, and Lennart Lönngren. 2013. SALDO: a touch of yin to WordNet's yang. *Language Resources and Evaluation*, 47(4):1191–1211.

António Branco, João Rodrigues, Francisco Costa, João Silva, and Rui Vaz. 2014. Rolling out text categorization for language learning assessment supported by language technology. *Computational Processing of the Portuguese Language. Springer*, pages 256–261.

Jill Burstein and Martin Chodorow. 2010. Progress and New Directions in Technology for Automated Essay Evaluation. *Oxford University Press*.

Yen-Yu Chen, Chien-Liang Liu, Chia-Hoang Lee, Tao-Hsing Chang, et al. 2010. An unsupervised automated essay scoring system. *IEEE Intelligent systems*, 25(5):61–67.

Council of Europe. 2001. *Common European Framework of Reference for Languages: Learning, Teaching, Assessment*. Press Syndicate of the University of Cambridge.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, pages 101–126.

Hal Daumé III. 2007. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic. Association for Computational Linguistics.

Cecilia Fasth and Anita Kannermark. 1997. *Form i focus: övningsbok i svensk grammatik. Del B*. Folkuniv. Förlag, Lund.

Thomas François and Cédrick Fairon. 2012. An 'AI readability' formula for French as a foreign language. In *Proceedings of the EMNLP and CoNLL 2012*.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. In *The SIGKDD Explorations*, volume 11, pages 10–18.

Julia Hancke and Detmar Meurers. 2013. Exploring CEFR classification for German based on rich linguistic modeling. In *Proceedings of the Learner Corpus Research (LCR) conference*.

Julia Hancke. 2013. Automatic prediction of CEFR proficiency levels based on linguistic features of learner language. *Master's thesis, University of Tübingen*.

Tor G Hultman and Margareta Westman. 1977. *Gymnasistsvenska*. Liber.

Jing Jiang and ChengXiang Zhai. 2007. Instance Weighting for Domain Adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271, Prague, Czech Republic. Association for Computational Linguistics.

Adam Kilgarriff, Pavel Rychly, Pavel Smrz, and David Tugwell. 2004. Itri-04-08 the sketch engine. *Information Technology*, 105:116.

Thomas K Landauer, Darrell Laham, and Peter W Foltz. 2003. Automated scoring and annotation of essays with the Intelligent Essay Assessor. *Automated essay scoring: A cross-disciplinary perspective*, pages 87–112.

David Little. 2011. The Common European Framework of Reference for Languages: A research agenda. *Language Teaching, Vol 44.3*.

Beáta Megyesi, Jesper Näsman, and Anne Palmér. 2016. The Uppsala Corpus of Student Writings: Corpus Creation, Annotation, and Analysis. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Amália Mendes, Sandra Antunes, Maarten Janssen, and Anabela Gonçalves. 2016. The COPLE2 Corpus: a Learner Corpus for Portuguese. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Eleni Miltsakaki and Karen Kukich. 2004. Evaluation of text coherence for electronic essay scoring systems. *Natural Language Engineering*, 10(01):25–55.

Daniel Naber. 2003. A rule-based style and grammar checker. Master's thesis, Bielefeld University, Bielefeld, Germany.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant, editors. 2014. *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, Baltimore, Maryland.

Diane Nicholls. 2003. The Cambridge Learner Corpus: Error coding and analysis for lexicography and ELT. In *Proceedings of the Corpus Linguistics 2003 conference*, volume 16, pages 572–581.

Brian North. 2007. The CEFR illustrative descriptor scales. *The Modern Language Journal 91*.

Ellis Batten Page. 2003. Project essay grade: PEG. *M.D. Shermis and J.C. Burstein, editors, Automated essay scoring: A cross-disciplinary perspective*, pages 43–54.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359.

Peter Phandi, Kian Ming A. Chai, and Hwee Tou Ng. 2015. Flexible Domain Adaptation for Automated Essay Scoring Using Correlated Linear Regression. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 431–439, Lisbon, Portugal. Association for Computational Linguistics.

Ildikó Pilán, Sowmya Vajjala, and Elena Volodina. 2015. A readable read: Automatic Assessment of Language Learning Materials based on Linguistic Complexity. *To appear in International Journal of Computational Linguistics and Applications*. Available at `http://arxiv.org/abs/1603.08868`.

Yao-Ting Sung, Wei-Chun Lin, Scott Benjamin Dyson, Kuo-En Chang, and Yu-Chia Chen. 2015. Leveling L2 texts through readability: Combining multilevel linguistic features with the CEFR. *The Modern Language Journal*, 99(2):371–391.

Sowmya Vajjala and Kaidi Lõo. 2014. Automatic CEFR Level Prediction for Estonian Learner Text. *NEALT Proceedings Series Vol. 22*, pages 113–127.

Sowmya Vajjala and Detmar Meurers. 2012. On improving the accuracy of readability classification using insights from second language acquisition. In *Proceedings of the Seventh Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–173. Association for Computational Linguistics.

Elena Volodina and Sofie Johansson Kokkinakis. 2012. Introducing the Swedish Kelly-list, a new lexical e-resource for Swedish. In *Proceedings of LREC*, pages 1040–1046.

Elena Volodina, Ildikó Pilán, Stian Rødven Eide, and Hannes Heidarsson. 2014. You get what you annotate: a pedagogically annotated corpus of coursebooks for Swedish as a Second Language. *NEALT Proceedings Series Vol. 22*, pages 128–144.

Elena Volodina, Ildikó Pilán, Ingegerd Enström, Lorena Llozhi, Peter Lundkvist, Gunlög Sundberg, and Monica Sandell. 2016. SweLL on the rise: Swedish Learner Language corpus for European Reference Level studies. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Katrin Wisniewski, Karin Schöne, Lionel Nicolas, Chiara Vettori, Adriane Boyd, Detmar Meurers, Andrea Abel, and Jirka Hana. 2013. MERLIN: An online trilingual learner corpus empirically grounding the European reference levels in authentic learner data. In *ICT for Language Learning 2013, Conference Proceedings, Florence, Italy. Libreriauniversitaria. it Edizioni*.

Menglin Xia, Ekaterina Kochmar, and Ted Briscoe. 2016. Text Readability Assessment for Second Language Learners. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 12–22, San Diego, CA. Association for Computational Linguistics.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 180–189. Association for Computational Linguistics.

Torsten Zesch, Michael Wojatzki, and Dirk Scholten-Akoun. 2015. Task-Independent Features for Automated Essay Grading. In *Proceedings of the Building Educational Applications Workshop at NAACL*.

Robert Östling, André Smolentzov, Björn Tyrefors, and Erik Höglin. 2013. Automated Essay Scoring for Swedish. In *The 8th Workshop on Innovative Use of NLP for Building Educational Applications*.

User Classification with Multiple Textual Perspectives

Dong Zhang, Shoushan Li*, Hongling Wang, Guodong Zhou
Natural Language Processing Lab
School of Computer Science and Technology, Soochow University, China
dzhang@stu.suda.edu.cn, lishoushan@suda.edu.cn
hlwang@suda.edu.cn, gdzhou@suda.edu.cn

Abstract

Textual information is of critical importance for automatic user classification in social media. However, most previous studies model textual features in a single perspective while the text in a user homepage typically possesses different styles of text, such as *original message* and *comment from others*. In this paper, we propose a novel approach, namely ensemble LSTM, to user classification by incorporating multiple textual perspectives. Specifically, our approach first learns a LSTM representation with a LSTM recurrent neural network and then presents a joint learning method to integrating all naturally-divided textual perspectives. Empirical studies on two basic user classification tasks, i.e., gender classification and age classification, demonstrate the effectiveness of the proposed approach to user classification with multiple textual perspectives.

1 Introduction

User attribute classification, also namely user classification for short, is a task which aims to leverage user-generated content to automatically predict user's attributes, such as gender (Wang et al., 2015), age (Rao et al., 2010; Sap et al., 2014) and location (Cheng et al., 2010). Recently, the growth of online social networks provides the opportunity to perform user classification in a broader context (Bollen et al., 2011; Sadilek et al., 2012; Lampos and Cristianini, 2010; Zamal et al., 2012). Basically, user classification is a fundamental task not only in sociolinguistic studies, but also in many real applications, such as recommender systems, and online advertising (O'Connor et al., 2010; Preotiuc-Pietro et al, 2015).

Text style	***User A*** ***Gender: female***	***User B*** ***Gender: male***
Original Message	"*Just bought the lipstick, look beautiful?*"	"*The first day, hard work.*"
Retweeted Message	"*Seaweed mask, it is so remarkably efficient.*"	"*Love her, take her to see the sea.*"
Comment from others	*"Sister, you're so pretty!"*	"*Go to see my latest message*"
Comment to others	"*Thanks.*"	*"Sister, you're so pretty!"*

Table 1: Some examples of different text styles in two users' homepages in a social media

Currently, machine learning approaches have dominated the research on user classification where statistic classifiers are learned with labeled data and various kinds of features, such as textual features,

* Corresponding author

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2112–2121, Osaka, Japan, December 11-17 2016.

behavior features, and social connection features (Preotiuc et al., 2015, Lampos et al., 2016). Among these features, textual features are most popular and they are good clues to infer the user attributes (Zhu et al., 2015; Li et al., 2015). For example, in Table 1, User A publishes a text "*Just bought the lipstick, look beautiful*?" which could be used to infer the user to be a *female* since *females* are more likely to buy a lipstick.

However, user-generated text sometimes possesses different styles, especially in social media. For instance, in Table 1, a homepage in a social media contains at least four kinds of text, namely *Original Message*, *Retweeted Message*, *Comment From Others*, and *Comment To Others*. Almost all previous studies do not distinguish these different styles of text, which might hurt the classification performance. For instance, in Table 1, User A has a *Comment From Others* "*Sister, you're so pretty!*" and User B has the same text but belongs to a different text style, i.e., *Comment To Others*. When the classifier do not carefully differentiate these text styles but merely mix all textual information together, using the sample of User A as training data is more likely to classify User B to be the same gender due to the same text "*Sister, you're so pretty!*". Obviously, this is a wrong prediction because User B is a *male* and the word "sister" is used to call someone else. Therefore, a better way to leverage textual knowledge in social media should be able to distinguish different styles of text.

In this paper, we address the above challenge by proposing a novel approach called ensemble LSTM recurrent neural network. Specifically, we first consider the features from each style of text as a separate textual perspective. Then, we train a Long Short-Term Memory (LSTM) network for each textual perspective respectively. Third, we add a merge layer to combine all LSTM representations by joint learning so as to fuse all textual knowledge. Empirical studies demonstrate that our approach performs much better than many strong baseline approaches.

Note that the motivation of employing LSTM as our single-perspective learning approach is that LSTM equips with a special gating mechanism that controls access to memory cells and it is powerful and effective at capturing long-term dependencies (Bengio et al., 1994). This advantage is helpful for modeling text and thus this approach has been successfully applied to a variety of NLP tasks, such as machine translation (Bahdanau et al., 2015), sentiment analysis (Tang et al., 2015), and sequence labeling (Chen et al., 2015).

The remainder of this paper is organized as follows. Section 2 overviews related work on user classification. Section 3 introduces data collection. Section 4 proposes our multi-perspective ensemble LSTM approach with multiple textual perspectives for user classification. Section 5 evaluates our approach with a benchmark dataset. Finally, Section 6 gives the conclusion and future work.

2 Related Work

Over the last decade, many previous studies have been devoted to the research on user classification with multiple attributes, such as user gender and user age.

User gender classification has been extensively studied in several domains, such as Blog (Peersman et al., 2011; Gianfortoni et al., 2011), E-mail (Mohanmad et al., 2011), YouTube (Filippova, 2012) and Micro-blog (Liu et al., 2013). More recently, some studies focus on some specific application scenarios on gender classification, such as multi-lingual gender classification (Ciot et al., 2013; Alowibdi et al., 2013), inferring gender by crowd (Nguyen et al., 2014) and interactive gender classification (Li et al., 2015).

User age classification has been studied in two main domains, i.e., blog (Burger and Hender son, 2006) and social media (Machinnon and Warren, 2006). In the blog domain, Schler et al. (2006) focus on textual features extracted from the blog text, such as word context features and POS stylistic features. Burger and Henderson (2006) explore some social features, such as location, time, and friend features, related to blogger age. Other studies, such as Rosenthal and McKeown (2011) and Goswami et al. (2009) explore both the textual and social features in automatic age classification. In the social media domain, Mackinnon and Warren (2006) explore some kind of social features, i.e., the relationship between users to predict a user's age and country of residence in a social network. Peersman et al. (2011) apply a text categorization approach to age classification with textual features only, i.e., word unigrams and bigrams. More recently, Marquardt et al. (2014) propose a multi-label classification approach to predict both the gender and age of authors from texts. Specifically, besides the word features, they also adopt some sentiment and emotion features in their approach.

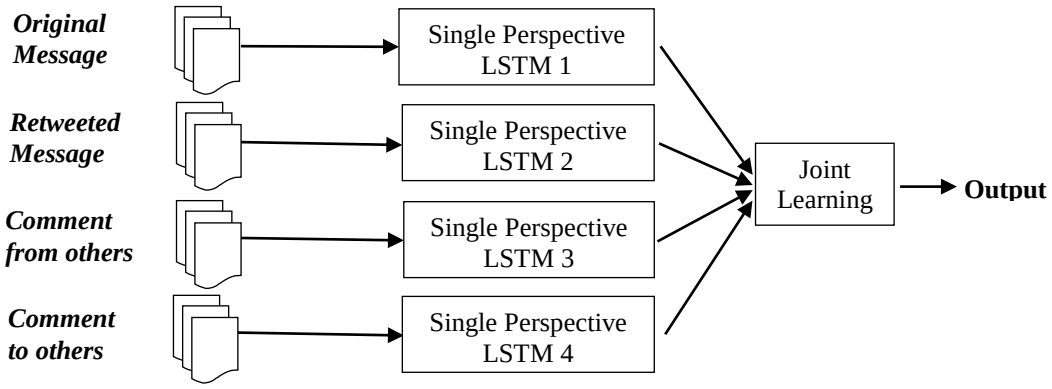


Figure 1: The framework of multi-perspective ensemble LSTM neural network

Some other user attributes, such as user location (Cheng et al., 2010), political orientation (Rao et al., 2010) and user occupational class prediction (Preotiuc-Pietro et al, 2015) are also popularly studied in recent years. Unlike all previous studies, this paper employs a deep learning approach to user classification and different styles of textual features are treated separately.

3 Data Collection

Our data are collected from Sina Micro-blog[1], a famous Micro-blogging platform in China. From the website, we crawl each user's homepage which contains user information (e.g., *name*, *age*, *gender*, *verified type*), and their posted messages. The data collection process starts from some randomly selected users, and iteratively gets the data of both their user attributes including gender and age. Different styles of text in each user's homepage are collected and they are:

1) ***Original message***: the messsages which are originally published by the user;
2) ***Retweeted message***: the messages which are retweeted by the user;
3) ***Comment from others***: the comments which are written by other users;
4) ***Comment to others***: the comments which are written by the user.

For gender classification, we randomly select 3000 *male* and 3000 *female* users for our empirical study and for age classification, we randomly focus on two age categories: *80s* (birthday between 1980 and 1989), *90s* (birthday between 1990 and 1999), each of which contains 3000 samples.

Table 2 shows the statistics about the average number of messages each user possessed in his/her homepage. From this table, we can see that each style of text has a decent number of messages or comments where *original message* and *comment to others* have more messages or comments than the other two styles.

	Gender		Age	
	Male	*Female*	*80s*	*90s*
Original message	148	158	154	153
Retweeted message	84	95	86	90
Comment from others	140	189	175	189
Comment to others	83	121	105	128

Table 2: Statistics about the average number of messages each user processed in his/her homepage

4 Our Approach

We treat the four styles of text as four textual perspectives for user classification and learn a multi-perspective ensemble LSTM recurrent neural network to make full use of all these perspectives. In general, our approach consists of two main components: (1) learning a new representation via a single-

[1] http://weibo.com/

perspective LSTM recurrent neural network of one type of user perspective. (2) employing a merge layer via joint learning to combine four different types of user perspectives. Figure 1 shows the framework overview of our approach and the two main components, i.e., single-perspective LSTM and multi-perspective ensemble LSTM via joint learning, will be discussed in detail.

4.1 Single perspective LSTM

In this study, we apply the implementation used by (Graves, 2013). The LSTM units at each time step t are defined to be a collection of vectors in $\mathbb{R}^d$: an input gate i_t, a forget gate f_t, an output gate o_t, a memory cell c_t and a hidden state h_t.

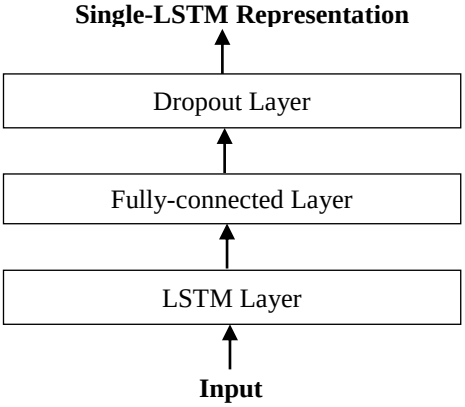


Figure 2: The framework of single perspective LSTM

Figure 2 illustrates the model architecture of our single-perspective LSTM where only one single LSTM layer is used. The input contains the representation of one type of textual perspective. According to the transition mode above, the input propagates through LSTM layer, Fully-connected layer and Dropout layer. The computing functions are given as following:

$$h^* = \phi\left(\omega^T h + b\right) \tag{1}$$

$$g = h^* \cdot D(p) \tag{2}$$

Where ϕ is the non-linear activation function, employed "*relu*" in our model and h is the output from LSTM layer. D denotes the dropout operator and p denotes a tune-able hyperparameter (the probability of retaining a hidden unit in the network).

4.2 Multi-perspective Ensemble LSTM via Joint Learning

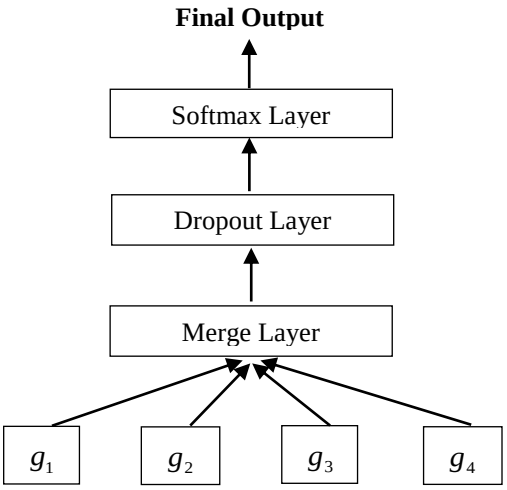


Figure 3: The framework of our multi-perspective ensemble LSTM approach

In order to distinguish the four types of textual perspectives and make full use of them legitimately, we propose a multi-perspective ensemble LSTM via joint learning to incorporate classification knowledge in *original message*, *retweeted message*, *comment from others* and *comment to others* separately. Figure 3 shows the framework of our multi-perspective ensemble LSTM approach where g_1, g_2, g_3 and g_4 are four LSTM representations learned from four single-perspective LSTM neural networks with four styles of textual perspectives.

The merge layer is designed to combine four types of user representation with a standard concatenation operation, i.e.:

$$g^* = [g_1; g_2; g_3; g_4] \tag{3}$$

Finally, a softmax output layer is used for classification. The model's prediction $label_{pred}$ is the class whose probability is maximal, specifically:

$$label_{pred} = \arg\max_i P(Y = i | x, W, U, V) \tag{4}$$

In our joint learning, the training objective is the penalized cross-entropy error, i.e.:

$$J = -\sum_{i=1}^{n_c} t_i \log y_i + \lambda \sum_{i=1}^{m} \left(\sum_{\varepsilon \in \omega} \left\| W_i^{\varepsilon} \right\|_F^2 + \sum_{\varepsilon \in \mu} \left\| U_i^{\varepsilon} \right\|_F^2 + \sum_{\varepsilon \in \nu} \left\| V_i^{\varepsilon} \right\|_F^2 \right) \tag{5}$$

Where $t \in \mathbb{R}^{n_c}$ is the one-hot represented ground truth and $y \in \mathbb{R}^{n_c}$ is the estimated probability for each class by softmax. (n_c is the number of target classes; m is the number of textual perspectives). In addition, W, U and V represent the corresponding weight matrices connecting them to the gates. $\| \cdot \|_F$ denotes the Frobenius norm of a matrix. $\omega = \{i, f, o, c\}$, $\mu = \{i, f, o, c\}$ and $\nu = \{i, f, o\}$ are the set of different gates (for W 's, U 's and V 's, respectively). λ is a hyperparameter that specifies the magnitude of penalty on weights.

To train our ensemble LSTM, we use Stochastic Gradient Descent with mini-batches. The set of parameters to learn is the set $\theta = \{W, U, V\}$ in each single LSTM RNN of user perspective. The gradients $\partial J / \partial \theta$ are achieved through the back propagation algorithm (a special case of the chain-rule of derivation). Specifically, in terms of W_i^{ε}, the update equation is given by:

$$W_i^{\varepsilon} := W_i^{\varepsilon} + \frac{\partial J}{\partial g^*} \cdot \frac{\partial g^*}{\partial g_i} \cdot \frac{\partial g_i}{\partial h_i^*} \cdot \frac{\partial h_i^*}{\partial h_i} \cdot \frac{\partial h_i}{\partial W_i^{\varepsilon}} \tag{6}$$

Where $\frac{\partial h_i}{\partial W_i^{\varepsilon}}$ in LSTM unit will be computed via back propagation though time (BPTT). In the same spirit, U_i^{ε} and V_i^{ε} could be obtained as following:

$$U_i^{\varepsilon} := U_i^{\varepsilon} + \frac{\partial J}{\partial g^*} \cdot \frac{\partial g^*}{\partial g_i} \cdot \frac{\partial g_i}{\partial h_i^*} \cdot \frac{\partial h_i^*}{\partial h_i} \cdot \frac{\partial h_i}{\partial U_i^{\varepsilon}} \tag{7}$$

$$V_i^{\varepsilon} := V_i^{\varepsilon} + \frac{\partial J}{\partial g^*} \cdot \frac{\partial g^*}{\partial g_i} \cdot \frac{\partial g_i}{\partial h_i^*} \cdot \frac{\partial h_i^*}{\partial h_i} \cdot \frac{\partial h_i}{\partial V_i^{\varepsilon}} \tag{8}$$

5 Experiments

In this section, we empirically evaluate the performance of our approach to user classification in social media.

5.1 Experimental Settings

Dataset: (1) **Gender classification**: the dataset contains 3000 *male* and 3000 *female* users and each user has four styles of text: *original message*, *retweeted message*, *comment from others* and *comment to others*. We randomly select 4200 (70%) users as training data, 600 (10%) users as development data and use the remaining 1200 (20%) users as test data. (2) **Age classification**: the data set contains 3000 *80s* (between 1980 and 1989) users and *90s* (between 1990 and 1999) users and each user has four

Parameter and Description	Value
Size of total unigram features	30000
Dimension of the LSTM layer output	128
Dimension of the fully-connected layer output	64
Dropout rate	0.5
Epochs of iteration	10

Table 3: Parameters setting in LSTM RNN

	ME	CNN	Parallel CNN	LSTM
Original message	0.843	0.843	0.849	0.863
Retweeted message	0.784	0.793	0.788	0.791
Comment from others	0.825	0.798	0.818	0.823
Comment to others	0.736	0.743	0.754	0.776
Average	*0.797*	*0.794*	*0.802*	***0.813***

Table 4: Performance comparison of different approaches with single textual perspective (**Gender Classification**)

styles of text: *original message*, *retweeted message*, *comment from others* and *comment to others*. We randomly select 4200 (70%) users as training data, 600 (10%) users as development data and use the remaining 1200 (20%) users as test data.

Representations: Each message text is treated as a bag-of-features and transformed into binary vectors encoding the presence or absence of each feature. The features include word unigrams, and two kinds of complex features, i.e., F-measure and POS sequence pattern features, which yield the state-of-the-art performance in user classification (Mukherjee and Liu, 2010).

Classification algorithms: (1) The maximum entropy (ME) classifier implemented with the public tool, Mallet Toolkits[2]. (2) The random forest classifier and adaboost classifier implemented with the public tool, scikit-learn[3]. (3) The CNN classifier implemented with the help of the tool Keras[4]. (4) The LSTM classifier implemented with the help of the tool Keras.

Parameters Setting: (1) The most important parameter of RF and ABC is *estimators*, which is set 500 via fine-tuning. (2) The parameters of LSTM are set as shown in Table 3.

Evaluation Measurement: The performance is evaluated using the standard accuracy measurement.

5.2 Experimental Results

Experimental Results on Single Textual Perspective

For thorough comparison, four approaches with single perspective are implemented:

- **ME**: the maximum entropy classifier with all the parameters default.
- **CNN**: the basic bow-CNN is proposed in (Johnson and Zhang, 2014).
- **Parallel CNN**: the extension of bow-CNN, which has two or more convolution layers in parallel to learn multiple types of embedding of small text regions, proposed in (Johnson and Zhang, 2014).
- **LSTM**: the single perspective LSTM introduced in Section 4.1.

Table 4 shows the performance comparison of four approaches to gender classification. From this table, we can see that the text style of *original message* performs best among all four styles of text no matter what classification approach is used. On average, CNN and Parallel CNN performs better than ME. Among the four approaches, LSTM perform best. Significance test shows that our LSTM approach significantly outperforms the other four approaches (p-value<0.05).

Table 5 shows the performance comparison of four approaches to age classification. From the table, we can see that the text style of *original message* performs best among all four styles of text no matter

[2] http://mallet.cs.umass.edu/
[3] http://scikit-learn.org/stable/
[4] https://github.com/fchollet/keras

	ME	CNN	Parallel CNN	LSTM
Original message	0.793	0.775	0.763	0.794
Retweeted message	0.707	0.699	0.733	0.745
Comment from others	0.736	0.761	0.757	0.759
Comment to others	0.745	0.751	0.744	0.760
Average	*0.745*	*0.747*	*0.749*	***0.765***

Table 5: Performance comparison of different approaches with single textual perspective (**Age Classification**)

Approach	RandomForest	Adaboost	Voting LSTM	Weighted_Sum LSTM	***Ensemble LSTM (Ours)***
Accuracy	0.791	0.803	0.853	0.885	***0.908***

Table 6: Performance comparison of five approaches with multiple textual perspective (**Gender Classification**)

Approach	RandomForest	Adaboost	Voting LSTM	Weighted_Sum LSTM	***Ensemble LSTM (Ours)***
Accuracy	0.763	0.744	0.801	0.816	***0.823***

Table 7: Performance comparison of five approaches with multiple textual perspective (**Age Classification**)

what classification approach is used. Similar to the results in gender classification, LSTM still perform best in age classification. Significance test shows that our LSTM approach significantly outperforms the other three approaches (p-value<0.05).

Experimental Results on Multiple Textual Perspectives

For thorough comparison, several ensemble learning approaches with multiple perspectives are implemented:

- **RandomForest**: a popular ensemble learning approach proposed by Strobl et al. (2007). In our implementation, we train multiple decision tree classifiers and employ random forest algorithm to combine them.
- **Adaboost**: a popular ensemble learning approach proposed by (Zhu et al., 2009). In our implementation, we mixture the data of all perspective and use each word feature to form a weak classifier and then combine all feature classifier with adaboost algorithm.
- **Voting LSTM**: we first use each single textual perspective to train a LSTM classifier and then use the voting rule (Kuncheva and Rodriguez, 2014) to combine the obtained label outputs from all single-perspective LSTM classifiers.
- **Weighted_Sum LSTM**: we first use each single textual perspective to train a LSTM classifier and then use weighted sum rule (Marler and Arora, 2010) to combine the obtained probability outputs from all single-perspective LSTM classifiers.
- **Ensemble LSTM (Our approach)**: our joint learning approach as introduced in Section 4.2.

Table 6 shows the performance comparison of all approaches to gender classification when multiple textual perspectives are used. From this table, we can see that, using multiple textual perspectives does not always outperform the best performed approach with a single textual perspective. For instance, when RandomForest and Adaboost are used, the performance of using multiple textual perspective are 0.791 and 0.803 respectively, which are worse than that of using the *Original message* perspective with LSTM classifier, i.e., 0.863. Our ensemble LSTM approach performs best and it performs much better than both the best-performed single perspective LSTM (as shown in Table 4) and other strong ensemble strategies with multiple textual perspectives, such as Voting LSTM and Weighted_Sum LSTM. Significance test shows that our ensemble LSTM approach significantly outperforms other approaches when multiple textual perspectives are used (p-value<0.05).

Table 7 shows the performance comparison of all approaches to age classification when multiple textual perspectives are used. From this table, we can see that our ensemble LSTM approach performs best and it is also performs better than other strong ensemble strategies with multiple textual perspectives, such as Voting LSTM and Weighted_Sum LSTM. Significance test shows that our ensemble LSTM approach significantly outperforms other approaches when multiple textual perspectives are used (p-value<0.05).

5.3 Effectiveness Analysis and Case Study

In order to further illustrate the superiority of our approach, we give a case study as following. Table 8 shows the selected features sorted by the feature selection method of information gain (IG) (Li et al., 2009) when the task of gender classification is considered. We extract the features from the *original message* text and the *retweeted message* text separately.

This table shows the top-10 IG features from the *original message* text and their ranks in the *retweeted message* text. N denotes the sequence number of the feature in the selected features. $|F_f|$ denotes the feature frequency in all samples of female. $|F_m|$ denotes the feature frequency in all samples of male. For instance, the sequence number of emoticon "rabbit" in *original message* is the first, the feature frequency in all samples of female is 5871, and the feature frequency in all samples of male is 1872. It is observed that this feature is usually used by a woman. From the table, we can see that many '*good*' features in *original message*, such as *emoticon [rabbit]*, *亲亲* (*kiss*) and *讨厌* (*hate*), are not ranked top in *retweeted message*. If we merely merge all styles of text, some '*good*' features in one textual perspective would not be as effective as in the scenario when they are separately treated.

Feature	Original message			Retweeted message		
	N	$\|F_f\|$	$\|F_m\|$	N	$\|F_f\|$	$\|F_m\|$
表情符-兔子 (emoticon [rabbit])	1	5871	1872	154	1553	960
亲亲 (kiss)	2	3700	978	104	1186	606
闺蜜 (ladybro)	3	588	103	1	1186	313
NBA	4	53	467	10	120	500
足球 (football)	5	169	1144	5	328	1561
球队 (team)	6	31	378	3	135	810
讨厌 (hate)	7	1854	773	797	1470	943
进球 (goal)	8	23	296	6	96	607
委屈 (grievance)	9	2358	802	--	--	--
男神 (dream guy)	10	1163	331	26	843	334

Table 8: The top-10 IG features from the *original message* text and their ranks in the *retweeted message* text

6 Conclusion

In this study, we propose a novel approach, namely ensemble LSTM, to user classification, which jointly learns textual features from different textual perspectives. Our contributions lie in two main aspects: First, the proposed LSTM approach with a single textual perspective performs much better than traditional approaches, such as ME and CNN, for user classification. Second, the proposed ensemble LSTM approach significantly outperforms both the approaches which use only one single textual perspective and several other ensemble approaches.

In our future work, we attempt to apply bidirectional LSTM in user classification to utilize both the bi-directional contexts. Moreover, in addition to the textual features, we would like to merge social features to further improve performance. What's more, we will apply our proposed multi-perspective ensemble LSTM model in some other tasks of user classification, such as user occupation classification and so on.

Acknowledgements

This research work has been partially supported by four NSFC grants, No.61273320, No.61375073, No.61331011, and No. 61402314.

Reference

Jalal S. Alowibdi, Ugo A. Buy and PHilip Yu. 2013. Language Independent Gender Classification on Twitter. In *Proceedings of IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 739-743.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning Long-term Dependencies with Gradient Descent is Difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter Mood Predicts the Stock Market. *Journal of Computational Science*, 2(1): 1-8.

John D. Burger and John C. Henderson. 2006. An Exploration of Observable Features Related to Blogger Age. In *Proceedings of AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs*, pages 15-20.

Zhiyuan Cheng, James Caverlee, and Kyumin Lee. 2010. You Are Where You Tweet: A Content-Based Approach to Geo-locating Twitter Users. In *Proceedings of CIKM*, pages 759-768.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long Short-term Memory Neural Networks for Chinese Word Segmentation. In *Proceedings of EMNLP*, pages 1197-1206.

Morgane Ciot, Morgan Sonderegger and Derek Ruths. 2013. Gender Inference of Twitter Users in Non-English Contexts. In *Proceedings of EMNLP*, pages 1136–1145.

Katja Filippova. 2012. User Demographics and Language in an Implicit Socail Network. In *Proceedings of EMNLP*, pages 1478-1488.

Philip Gianfortoni, David Adamson and Carolyn P. Ros é. 2011. Modeling of Stylistic Variation in Social Media with Stretchy Patterns. In *Proceedings of EMNLP*, pages 49–59.

Sumit Goswami, Sudeshna Sarkar and Mayur Rustagi. 2009. Stylo-metric Analysis of Bloggers' Age and Gender. In *Proceedings of AAAI Conference on Weblogs and Social Media*, pages 214-217.

Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *arXiv preprint arXiv:1308.0850*.

Rie Johnson, Tong Zhang. 2014. Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. *Eprint Arxiv*.

Ludmila I. Kunchev, Juan J. Rodr ́guez. 2014. A Weighted Voting Framework for Classifiers Ensembles. *Knowledge and Information Systems*, 38(2): 259-275.

Vasileios Lampos, Nikolaos Aletras, Jens K. Geyti, Bin Zou and Ingemar J. Cox. 2016. Inferring the Socioeconomic Status of Social Media Users based on Behaviour and Language. *European Conference on Information Retrieval. Springer International Publishing*, pages 689-695.

Vasileios Lampos and Nello Cristianini. 2010. Tracking the Flu Pandemic by Monitoring the Social Web. In *Proceedings of the 2nd International Workshop on Cognitive Information Processing*, pages 411-416.

Shoushan Li, Rui Xia, Chengqing Zong and Chu-Ren Huang. 2009. A Framework of Feature Selection Methods for Text Categorization. In *Proceedings of ACL*, pages 692-700.

Shoushan Li, Jingjing Wang, Guodong Zhou, and Hanxiao Shi. 2015. Interactive Gender Inference with Integer Linear Programming. In *Proceedings of IJCAI*, pages 2341-2347.

Nan Liu, Yanxiang He, Qiang Chen, Min Peng and Ye Tian. 2013. A New Method for Micro-blog Platform Users Classification Based on Infinitesimal-time. *Journal of Information & Computantional Science*. pages 2569-2579.

Ian Mackinnon and Robert Warren. 2006. Age and Geo-graphic Inferences of the LiveJournal Social Net-work. In *Proceedings of ICML*, pages 176-178.

R. Timothy Marler, Jasbir S. Arora. 2010. The Weighted Sum Method for Multi-objective Optimization: New Insights. *Structural and multidisciplinary optimization*, 41(6): 853-862.

James Marquardt, Golnoosh Farnadi, Gayathri Vasudevan, Marie-Francine Moens, Sergio Davalos, Ankur Teredesai and Martine De Cock. 2014. Age and Gender Identification in Social Media. In *Proceedings of CLEF 2014 Evaluation Labs*, pages 1129-1136.

Saif M. Mohammad, and Tony Yang. 2011. Tracking Sentiment in Mail: How Genders Differ on Emotional Axes. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pages 70-79.

Arjun Mukherjee and Bing Liu. 2010. Improving Gender Classification of Blog Authors. In *Proceedings of EMNLP*, pages 207-217.

Dong Nguye, Dolf Trieschnigg, A. Seza Doğruöz, Rilana Gravel, Mariet Theune, Theo Meder and Franciska de Jong. 2014. Why Gender and Age Prediction from Tweets is Hard: Lessons from a Crowdsourcing Experiment. *Association for Computational Linguistics*.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In *Proceedings of* ICWSM, pages 122–129.

Claudia Peersman, Walter Daelemans, Leona Vaerenbergh. 2011. Predicting Age and Gender in Online Social Networks. In *Proceedings of SMUC*, pages 37-44.

Daniel Preotiuc-Pietro, Vasileios Lampos and Nikolaos Aletras. 2015. An Analysis of the User Occupational Class through Twitter Content. In *Proceedings of ACL*, pages 1754-1764.

Daniel Preoţiuc-Pietro, Svitlana Volkova, Vasileios Lampos, Yoram Bachrach and Nikolaos Aletras. 2015. Studying User Income through Language, Behaviour and Affect in Social Media. *PloS one*, 10(9): e0138717.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying Latent User Attributes in Twitter. In *Proceedings of the 2nd International Workshop on Search and Mining Usergenerated Contents*, pages 37–44.

Sara Rosenthal and Kathleen McKeown. 2011. Age Prediction in Blogs: A Study of Style, Content, and Online Behavior in Pre- and Post-Social Media Genera-tions. In *Proceedings of ACL*, pages 763-772.

Adam Sadilek, Henry Kautz, and Vincent Silenzio. 2012. Modeling Spread of Disease from Social Interactions. In *Proceedings of ICWSM*, pages 322-329.

Maarten Sap, Gregory Park, Johannes Eichstaedt, Margaret Kern, Lyle Ungar, and H Andrew Schwartz. 2014. Developing Age and Gender Predictive Lexica over Social Media. In *Proceedings of EMNLP*, pages 1146–1151.

Jonathan Schler, Moshe Koppel, Shlomo Argamon and James Pennebaker. 2006. Effects of Age and Gender on Blogging. In *Proceedings of AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs*, pages 199-205.

Carolin Strobl, Anne-Laure Boulestei, Thomas Augustin. 2007. Unbiased Split Selection for Classification Trees Based on the Gini Index. *Computational Statistics & Data Analysis*, 52(1): 483-501.

Duyu Tang, Bing Qin, Ting Liu. 2015. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. In *Proceedings of EMNLP*, pages 1422-1432.

Jingjing Wang, Yunxia Xue, Shoushan Li and Guodong Zhou. 2015. Leveraging Interactive Knowledge and Unlabeled Data in Gender Classification with Co-training. Database Systems for Advanced Applications. *Springer International Publishing*, pages 246-251.

Faiyaz Al Zamal, Wendy Liu, and Derek Ruths. 2012. Homophily and Latent Attribute Inference: Inferring Latent Attributes of Twitter Users from Neighbors. In *Proceedings of ICWSM*, pages 387-390.

Zhu Zhu, Jingjing Wang, Shoushan Li and Guodong Zhou. 2015. Interactive Gender Inference in Social Media. *Database Systems for Advanced Applications. Springer International Publishing*, pages 252-258.

Ji Zhu, Hui Zou, Saharon Rosset and Trevor Hastie. 2009. Multi-class Adaboost. *Statistics and its Interface*, 2(3): 349-360.

Says Who…? Identification of Expert versus Layman Critics' Reviews of Documentary Films

Ming Jiang
School of Information Sciences
University of Illinois at Urbana-Champaign
{mjiang17@illinois.edu}

Jana Diesner
School of Information Sciences
University of Illinois at Urbana-Champaign
{jdiesner@illinois.edu}

Abstract

We extend classic review mining work by building a binary classifier that predicts whether a review of a documentary film was written by an expert or a layman with 90.70% accuracy (F1 score), and compare the characteristics of the predicted classes. A variety of standard lexical and syntactic features was used for this supervised learning task. Our results suggest that experts write comparatively lengthier and more detailed reviews that feature more complex grammar and a higher diversity in their vocabulary. Layman reviews are more subjective and contextualized in peoples' everyday lives. Our error analysis shows that laymen are about twice as likely to be mistaken as experts than vice versa. We argue that the type of author might be a useful new feature for improving the accuracy of predicting the rating, helpfulness and authenticity of reviews. Finally, the outcomes of this work might help researchers and practitioners in the field of impact assessment to gain a more fine-grained understanding of the perception of different types of media consumers and reviewers of a topic, genre or information product.

1 Introduction

Product reviews help customers to make purchase decisions, and producers to improve and develop goods (Hu & Liu, 2004; Kim et al., 2006; Mudambi & Schuff, 2010). Scalable NLP-based solutions have been developed to support various aspects of these decision-making processes:

(1) Describing, understanding and anticipating product ratings can help manufacturers to comprehend a market. Ranking reviews and reviewers further aids this step. The rating values per review are typically user-generated, ordinal variables, often on a 5-point scale (Jiang & Diesner, 2016; Pang & Lee, 2005).

(2) Identifying the trustworthiness or authenticity of reviews can assist in separating authentic from fudged reviews (Jindal & Liu, 2007; Jindal et al., 2010; Wu et al., 2010). Predicting this feature is more challenging than the previously mentioned ones as authenticity values are not explicitly provided by reviewers or readers, but need to be inferred from the content of reviews and related metadata.

(3) Predicting whether a review was written by an expert or a layman helps to differentiate the impact of the electronic word-of-mouth on the e-marketplace. McAuley and Leskovec (2013) studied the change of reviewers' expertise over time in order to improve personal recommender systems. Knowing the type of reviewer can also assist with asserting the credibility of reviewers (Basuroy et al., 2003; Flanagin & Metzger, 2013; Liu et al., 2008).

Besides commercially motivated analyses of product reviews, assessing the impact of information products such as books, films and other works of art on individuals, groups and society is another domain where knowing the type of author can be beneficial. In the context of impact assessment, *expert critics* (short experts in this paper) can be conceptualized as people with high standards of integrity and an extrinsic motivation for this task, such as writing reviews as part of their jobs as journalists. *Laymen reviewers* can be considered as ordinary customer who are intrinsically motivated to voluntarily provide this type of user-generated content based on their personal experience and points of view (Amblee & Bui, 2007; Chattoo & Das, 2014; Napoli, 2014; Rezapour & Diesner, 2017). We acknowledge the possibility that laymen might write expert-level reviews and vice versa.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2122–2132, Osaka, Japan, December 11-17 2016.

Creators and funders of works of art can use the knowledge about the type of writer to evaluate the impact of information products on the public, e.g. in terms of knowledge diffusion, framing, and sentiment. To provide some better understanding of this process, in this paper, we develop a binary classifier that predicts whether a review of a documentary film was authored by an expert or a layman. We focus on the domain of issue-focused documentaries to complement work based on feature films and box-office blockbusters. Our work also complements prior knowledge gained from studies that predict reviewer expertise based on personal ratings (Amblee & Bui, 2007; Flanagin & Metzger, 2013; Plucker et al., 2009), and/or the online behaviour of reviewers (Liu et al., 2008). We hypothesize that the type of reviewer can be inferred from characteristics of the text data, and address the following research question: Do different text patterns exist in reviews authored by experts versus laymen? If so, what features are unique to each group?

To the best of our knowledge, our study is the first one to apply machine-learning methods to computationally detect the type of author based on the content of reviews. We achieve an overall prediction accuracy of about 90.70% (F-measure), and explain the characteristics of each predicted class (expert versus layman).

The remainder of this paper is organized as follows. We review related work in section 2. Our corpus is described in section 3, and the methods in section 4. In the results section (5), we identify characteristics of each type of author and provide an error analysis. Finally, conclusions and future work are discussed in section 6.

2 Related Work

Nelson (1970, 1974) divides products into "search goods," i.e., tangible objects like cars, and "experience goods," i.e., intangible objects like films. While it might be possible to objectively evaluate search goods, e.g. in terms of their form, function and behaviour, rating experience goods, which are the subject of this study, may involve more personal perspectives, opinions, emotions and subjective judgment (Liu et al., 2008).

The majority of prior NLP-based solutions to commercially inspired review mining tasks can be divided into three groups: First, studies that predict the rating of products and the helpfulness of reviews (Ghose & Ipeirotis, 2011; Kim et al., 2006; Liu et al., 2008; Mudambi & Schuff, 2010; Yang et al., 2015; Zhang et al., 2015). These two tasks are fairly straightforward because user-generated ground-truth data on these features is available. Second, work that identifies the sentiment or opinion entailed in reviews. Knowledge about this feature might also help to explain or predict the former two features. This task requires the labelling of reviews with (values for) sentiment or opinion categories. Building such predictors is typically approached by using deterministic (look-up dictionaries) and/or probabilistic NLP techniques (de Albornoz et al., 2011; Pang & Lee, 2005; Turney, 2002). Third, work that focuses on summarizing the content or the gist of reviews to reduce the complexity of large text corpora (Hu & Liu, 2004; Li et al., 2010; Zhuang et al., 2006).

Studying reviewer expertise, which is the focus of this paper, is a minor branch in current review mining research. In prior work, this problem has mainly been approached by a) using empirical statistical investigations, such as counting average rating scores of experts versus novices, or correlating rating values with product consumption, b) conducting content analysis of reviews (de Jong & Burgers, 2013; Mackiewicz, 2009), and c) computational identifying the level of reviewers' expertise. As an example for the last type, Liu and colleagues (2008) used "reviewer expertise" as one of three variables for identifying the helpfulness of movie reviews. The authors operationalized expertise as frequent and highly positive-rated reviews per author and per pre-defined film genre. Their other two features were writing style and review timeliness. Combining all three features in a non-linear regression resulted in a helpfulness prediction accuracy of 71.2% (F-measure). The isolated contribution of the expertise feature was 51.8% (F-measure). In another study, which also falls into the last category, McAuley and Leskovec (2013) showed that users become more experienced in developing their taste for experience goods (tested for the product categories of beer, wine, fine foods and movies) with over-time exposure to these products. The authors found that the accuracy for predicting item ratings increases when users had higher levels of experience or expertise.

Our work differs from prior studies in that we focus on predicting reviewer expertise as a binary variable based on text-based features of reviews of issue-focused documentaries. The primary goal of

Abbreviation	Documentary	#Expert Reviews	#Valid Expert Reviews	#Layman Reviews	#Total Valid Reviews
SPSZM	Super Size Me	770	166	727	893
INJO	Inside Job	246	68	905	973
FOIN	Food Inc	129	65	2707	2772
GTKER	The Gatekeepers	85	47	178	225
TCOV	The Cove (fishing film)	78	45	485	530
CTFR	Citizenfour	97	44	238	282
AOKI	The Act of Killing	130	39	100	139
BLKFSH	Blackfish	69	35	1171	1206
EOTL	The End of the Line	40	33	67	100
FBCR	5 Broken Cameras	40	28	119	147
TTDS	Taxi To the Dark Side	220	27	52	79
HILI	House I Live In	45	26	221	247
HTSAP	How to Survive a Plague	42	19	79	98
HABA	Hell and Back Again	30	19	67	86
DWAR	Dirty Wars: The World Is a Battlefield	45	17	416	433
IVWAR	The Invisible War	36	16	231	247
PL3P	Paradise Lost 3 Purgatory	17	10	125	135
PAPR	Pandora's Promise	15	10	41	51
PDBTH	Pray the Devil Back to Hell	12	5	51	56
TALD	Through a Lens Darkly	10	4	10	14
SUM		**2156**	**723**	**7990**	**8713**

Table 1: Corpus statistics

this study is to detect indicative text features that can differentiate reviews written by experts from laymen.

3 Data

We collected a dataset that contained ground-truth or gold-standard data, i.e., expert versus layman reviews, for 20 documentaries. The films were selected based on their coverage of main social justice issues (as defined by philanthropic funders), including environmental issues, politics, public health, gender and ethnicity (Diesner et al., 2016). Table 1 shows the list of selected films, their abbreviation used in this paper, and the number of reviews per category.

Based on our reading of reviews on several popular film-rating sites, such as Rotten Tomatoes, Metacritic, Amazon and YouTube, we assume that layman reviews are mainly provided voluntarily. For these reviews, the full texts are provided on these websites. However, for expert reviews, only snippets and a link to the original source (e.g., major newspapers) are typically displayed. Due to copyright regulations and the terms of service for these pages, we could not access expert reviews from these review sites. Alternatively, we used LexisNexis Academic to collect comments written by professional critics that were published in newspapers and other sources. For these searches, the queries contained the film's title, name(s) of the director and/ or producer, and the keyword "review". The latter two items mainly served as disambiguators. For laymen reviews, we collected customer reviews from Amazon after obtaining Amazon's permission for this procedure. Even though reviews per author type were collected from a different platform (customer reviews from Amazon, expert reviews from LexisNexis Academic), we argue that the source does not determine or predict the type of author for the following reason: Many of these platforms list both types of reviews side by side, on the same platform. In other words, expert reviews from sources like Rotten Tomatoes are not written by Rotten Tomatoes, but come from the same sources that we used for our study – e.g., major newspapers.

The data collection involved some challenges. First, manual inspection of each article from LexisNexis was unavoidable as many texts were (soft) duplicates or poor fits, e.g., comments on multiple films with the target film being only briefly mentioned. We manually eliminated duplicates,

false positives and poor fits. In the end, 33.53% of the downloaded reviews were judged as valid data points for this study. The resulting number of valid instances per class is also shown in Table 1.

Second, the number of laymen reviews exceeds that of expert reviews. Therefore, for learning, we used the smaller set (i.e., expert reviews) as the defining upper bound for the number of instances considered per class, and randomly sampled an equally-sized number of reviews from the larger set.

4 Method

4.1 Features

Our features selection is guided by prior work that have shown that different aspects of writing style are useful indicators of review helpfulness and reviewer expertise. Based on this prior work, we chose three types of features (discussed in detail below): length features, lexical features and syntactic features.

The content of all considered reviews (N=1446; 723 per class) was pre-processed via stop word removal, stemming, and converting capitalization to lower case[1] for most lexical features except for sentiment analysis. We tested the impact of each routine on feature construction and prediction performance (F-measure), and selected the abovementioned techniques as they contributed most strongly to prediction performance.

4.1.1 Length Features

The length of both reviews and sentences per reviews were considered (Review length, Average sentence length) and computed by using the Stanford Parser (De Marneffe et al., 2006).

4.1.2 Lexical Features

As word choice may also characterize or correlate with each type of reviewer, we leveraged the top 250 unigrams according to the TF*IDF metric (unigram) as shown in Equation 1.

$$weight(w, C_f) = tf(w, C_f) \times idf(w) = c(w, C_f) \times \log\left(1 + \frac{N}{df(w)}\right) \quad (1)$$

$$weight(w, d) = tf(w, d) \times idf(w) = c(w, d) \times idf(w) \quad (2)$$

In this equation, C_f represents the corpus of all reviews per film, *w* is any term in C_f, c(w, C_f) is the number of occurences of *w* in C_f, *N* is the total number of reviews in the collection of a film, and *df(w)* is the number of reviews within the corresponded collection in which *w* appears.

Equation 2 calculates the TF*IDF per unigram per review, where d is the content per review.

We also used the informativeness per review as a feature (Equation 4) (Weaver & Shannon, 1949) by calculating information entropy. This metric is based on the average amount of information that each w carries per review as well as in the whole corpus per film, respectively (see Equation 3). The calculation is determined by the *w*'s normalized weight in review d and corpus C. As we focus more on the amount of information carried by w in corpus, the ratio parameter λ was set to 0.3 after experimenting with various values.

$$p(w) = \lambda \times \frac{weight(w, d)}{\sum_{w' \in d} weight(w', d)} + (1 - \lambda) \times \frac{weight(w, C_f)}{\sum_{w' \in C} weight(w', C_f)} \quad (3)$$

$$H(d) = \sum_{w \in d} [-p(w) \log_2 p(w)] \quad (4)$$

In addition, the emotionality of reviews has been shown to correlate with formal (more neutral) versus informal (more emotional) writing styles (Hu & Liu, 2004; Jiang & Diesner, 2016; Kim et al., 2006). To calculate emotionality, we reused the previously built, evaluated and widely used MPQA Subjectivity Lexicon (Wilson et al., 2005). Using this external lexical resources, we identified sentiment-loaded terms, summed them up per text, and normalized the sum of the number of sentiment words per valence type by text length (Sentiment%).

[1] Implemented by using an open-source package:
https://github.com/ijab/trec_file_ir/tree/master/bin/edu/pitt/sis/infsci2140/analysis

TR	Definition	Examples
Addition	Provide similar or further information	and, also, or, further
Introduction	Illustrate an argument with a detailed instance	for example, such as
Emphasis	Underline an argument	Even, very especially
Concession	Counter a previous argument	but, however, although
Causality	Describe cause and effect	because, since
Condition	Explain a precondition	if, unless
Order	Sequentially order	before, after
Summary	Conclusion	in a word

Table 2: Selected transition relationships

Syntax	Usage
Aux	Identify clause with non-main verb
Auxpass	Identify passive voice of clause with non-main verb
Csubj	Identify subject clause
Csubjpass	Identify passive voice of subject clause
Dobj	Identify direct object of clause
Iobj	Identify indirect object of clause
Nsubj	Identify nominal subject in clause
Nsubjpass	Identify passive nominal subject in passive clause
Mark	Identify finite clause that subordinate to another clause

Table 3: Selected syntax dependencies

Finally, we considered transition words and phrases to capture text cohesion, i.e., how ideas within a review relate to each other. We also leveraged an external lexical resource for this feature (Campbell et al.). Table 2 shows the list of main transition relationships (TR) used in this paper. We counted the number of transitions per review and normalized the value by review length (Transition%). We also calculated the ratio of each type of TR to capture individual preferences among TR per text (see Equation 5), where t is any transition term that appears in the review d and belongs to the i^{th} type of TR. N_{t_d} notes total number of transition terms in d.

$$Ratio(TR_i, d) = \frac{\sum_{t \in TR_i} tf(t,d)}{N_{t\_d}} \quad (5)$$

4.1.3 Syntax Features

The Stanford POS tagger was used to assign a single best fitting grammatical function (part of speech or POS) to every token. Per review, we calculated: 1) POS diversity, i.e., the number of unique POS tags, and normalized the value by total 36, which is the total number of POS tags considered by the tagger (Marcus et al., 1993), and 2) the prevalence of content bearing terms (see Equation 6), where tag is any POS tag that belongs to the i^{th} type of content words $Content\_W$ in review d. N_{tag_d} represents the total number of POS tags appeared in d. For each type of content words, we considered a set of POS tags shown as below:

- Nouns (i.e., NN, NNS, NNP & NNPS)
- Verbs (i.e., VB, VBD, VBG, VBN, VBP & VBZ)
- Adjectives (i.e., JJ, JJR & JJS)
- Adverbs (i.e., RB, RBR & RBS)

$$Ratio(Content\_W_i, d) = \frac{\sum_{tag \in Content\_W_i} count(tag,d)}{N_{tag\_d}} \quad (6)$$

Beyond the token level syntax features, we further used the Stanford NLP Parser to take the grammatical functions of words on the sentence level into account. Similar to our approach for using POS tags, for each review, we calculated: 1) syntax label diversity, where we normalized the number of unique syntax dependencies which appear in each review by the total number of dependency relations (N=48) given in the Stanford parser (De Marneffe et al., 2006), and 2) the ratio of each selected syntax dependency (see Table 3) to all dependencies occurring per review; using the Stanford typed dependencies for this task (De Marneffe & Manning, 2008).

4.2 Learning and Evaluation

After experimenting with various learning algorithms and observing SVM outperforming Naïve Bayes, we decided to present results based on training an SVM with a radial kernel. The classifier was implemented using the R package e1071 (Dimitriadou et al., 2011).

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad \begin{cases} Precision = \frac{TP}{TP+FP} \\ Recall = \frac{TP}{TP+FN} \end{cases} \qquad (7)$$

In order to assess the performance of our features, we conducted a 10-fold cross validation, where we used all reviews (expert and non-expert) for 18 films for training, and documents from the remaining 2 films for testing. To create comparatively similarly sized folds, we sorted films by decreasing numbers of reviews, and iteratively combined the two films from each end into one fold. This non-standard way of partitioning the data was chosen to enable result interpretation and error analysis not only on the class label basis, but to also be able to see if certain films, e.g. on certain topics or from different release dates, impact predictability.

We evaluate the performance of the proposed approach using the standard metrics of precision, recall and the F1 score (see Equation 7). Since we code the class labels with 0 for experts and 1 for laymen, TP (i.e., true positives) represents the number of layman reviews that are correctly predicted while FP (i.e., false positives) is the number of reviews that are mistakenly predicted as layman reviews. TN (i.e., true negatives) and FN (i.e., false negatives) are defined in the same way, but for expert reviews.

5 Results and Analysis

5.1 Experimental Results

The overall prediction accuracy of our classifier is 90.70% (F1 score) (Table 4). While F1 values are fairly similar across evaluation metrics and films, recall is lower than precision and has a larger standard deviation (7.52%).

In general, high performance correlates with higher numbers of training instances, but not vice versa (Table 4). This concern is moderate as the Pearson correlation coefficient for F1 and the number of instances per fold is -0.65.

Also, precision (94.02%) is higher than recall (87.90%) on average (Table 4), which indicates that $\frac{TP}{TP+FP} > \frac{TP}{TP+FN} \stackrel{\text{def}}{=} FP < FN$ (Equation 7). Since we have the same number of training instances for each class (i.e., $TP + FN = TN + FP$), we can infer that $FP < FN \stackrel{\text{def}}{=} TN > TP$. The results suggest that overall, expert reviews are more accurately predictable than layman reviews. With a further comparison of the standard deviation between precision (4.59%) and recall (7.52%), this finding suggests that expert reviews show lower in-group variability than layman comments. This might be due to professional norms and standards.

The isolated contribution of each feature to prediction accuracy is shown in Table 5 (sorted by decreasing contribution), and the actual values per feature per class are provided in Table 6.

The syntax features have the highest isolated impact, which indicates that out of the considered features, grammar use contributes the most for distinguishing the considered two groups. Looking into POS diversity and parser label diversity as shown in Table 6, expert reviews (0.61 for POS diversity; 0.57 for syntax labels diversity) feature more complex syntax than layman reviews (0.44 for POS

Fold No.	Documentary	Precision	Recall	F1	# Instances
1	SPSZM + TALD	88.36%	75.88%	81.65%	340
2	INJO + PDBTH	85.14%	86.30%	85.71%	146
3	FOIN + PAPR	92.31%	96.00%	94.12%	150
4	GTKER + PL3P	98.18%	94.74%	96.43%	114
5	TCOV + IVWAR	96.55%	91.80%	94.12%	122
6	CTFR + DWAR	96.23%	83.61%	89.47%	122
7	AOKI + HABA	95.65%	75.86%	84.62%	116
8	BLKFSH + HTSAP	100.00%	92.59%	96.15%	108
9	EOTL + HILI	91.80%	94.92%	93.33%	118
10	FBCR + TTDS	96.00%	87.27%	91.43%	110
Average/ Sum		**94.02%**	**87.90%**	**90.70%**	**1446**
Std Dev		4.59%	7.52%	5.15%	/

Table 4: Accuracy from 10-folds cross validation using all features

Feature Type	Feature	Precision	Recall	F1
Syntax	Parts of speech	**92.76%**	**90.01%**	**91.29%**
	Parse tree constituents	85.50%	85.52%	84.45%
Lexical	Transition words	84.74%	69.63%	76.29%
	Entropy	86.32%	68.83%	76.05%
	Unigrams	67.37%	82.75%	73.91%
	Sentiment	73.69%	41.12%	52.30%
Length	Review length	69.63%	74.82%	71.79%
	Avg sentence length	79.29%	64.85%	71.12%

Table 5: Isolated contribution per feature (highest value per column in bold)

diversity; 0.37 for syntax labels diversity). However, this feature may correlate with review length, which is considerably higher for experts (362 words) than for laymen (107 words). Also, experts use more nouns than laymen, while laymen use more verbs and adjectives.

The choice of salient words (unigrams) has a strong impact on recall, while text informativeness (entropy) and transition words rather contribute to precision. Given the aforementioned definition of precision and recall, this result hints at some uniformity or consistency of word choice in laymen reviews, and at higher vocabulary diversity as well as more coherent structure in expert reviews. Sentiment is the weakest contributor to the prediction.

Further analyzing the differences between both groups (Table 6), we find that experts, in comparison to laymen, write longer reviews and longer sentences, use more complex syntax, provide more new information (i.e., expert reviews have higher entropy values than layman reviews), have a higher diversity in their vocabulary, and use fewer emotional words. Some of these features might correlate with review length, but overall, these findings might be explainable by a professional text production style that reflects established norms and rules of journalistic writing. Based on our data, short reviews are the strongest defining feature for non-expert reviews. Layman reviews are also more opinionated and emphasize points made more strongly (i.e., high and fluctuating Sentiment%).

In addition to these quantitative analyses, we conducted a qualitative analysis by reading through the top 20 unigrams (based on TF*IDF) for each film to better understand difference in content and writing between experts and laymen. Table 7 provides an illustrative example for two randomly selected films, and we refer to this example in the following discussion of descriptive features per category. Overall, we find that experts frequently refer to 1) people involved in making and producing films ("director"; "morgan," "spurlock"), 2) film titles ("inside," "job"), 3) cinematographic concepts ("moore" as *Michael Moore style*), and 4) awards and festivals. Also, experts connect issues addressed in films to current affairs and higher level topics ("obesity"), and provide details or background information (e.g., "hubbard"; who frequently appeared in expert reviews of INJO, represents *Glenn Hubbard;* an economist who previously worked for the federal government). Expert reviews entail specific concepts ("obesity") and formalities ("Mr."), while laymen use more casual terms ("fat"; "bad"). Laymen reviews represent substantial engagement with the topic of a film ("eat"; "diet"), contextualize issues in peoples' regular lives ("people"; "day"; "money"; "job"; "school"; "healthy"), and contain more subjective terms ("good"; "bad").

Feature	Expert (AVG±STD)	Layman (AVG±STD)
Entropy	2.94±0.79	1.86±0.69
Sentiment%	0.09±0.03	0.12±0.12
Transition%	0.05±0.02	0.07±0.05
Ratio addition	0.63±0.24	0.44±0.33
Ratio example	0.02±0.07	0.01±0.05
Ratio emphasis	0.04±0.07	0.12±0.21
Ratio concession	0.15±0.17	0.10±0.18
POS diversity	0.61±0.16	0.44±0.19
Ratio NN	0.36±0.06	0.25±0.12
Ratio VB	0.14±0.04	0.18±0.07
Ratio JJ	0.09±0.03	0.11±0.12
Ratio RB	0.04±0.02	0.06±0.06
Syntax label diversity	0.57±0.17	0.37±0.19
Review length	362.00±421.56	107.50±188.65
Avg sentence length	27.52±13.88	15.16±8.66

Table 6: Values and variance of features per class

5.2 Error Analysis

The confusion matrix (Table 8) shows that our classifier predicts expert reviews with higher accuracy (93.36%) than laymen reviews (86.17%). More importantly, laymen are more likely to be mistaken for experts (13.83%) than

SPSZM	Expert	size, spurlock, mcdonald, super, film, year, days, director, big, moore, month, obesity, company, million, morgan, day, people, festival, burger, american
	Layman	mcdonalds, people, spurlock, movie, film, diet, mcdonald, eat, day, fat, make, school, healthy, bad, time, eating, good, watch, body, experiment
INJO	Expert	ferguson, inside, film, job, charles, crisis, director, men, mr, company, financial, global, banks, documentary, bankers, economic, end, hubbard, street, crash
	Layman	film, movie, wall, people, government, street, job, money, documentary, banks, great, financial, crisis, inside, world, watch, good, american, loans, made

Table 7: Top 20 Unigrams for Case Study

vice versa (6.64%). Looking into laymen reviews labeled as expert reviews, we find that these texts are long, detailed, and contain subject matter expertise. Expert reviews that got misclassified as laymen reports were typically short. These findings further substantiate our previously made point that some laymen write expert-level reviews, and vice versa.

To analyze our prediction errors more in depth, we selected a random sample (N=62) of misclassified reviews from both classes. We removed the class labels from these documents and asked two independent human annotators to code the texts as expert or layman reviews. Their inter-coder agreement was 45.2%, which suggests that categorizing these cases is also hard for humans.

We further discussed the label assignments with our two human coders. Trends emerging from their observations and our discussion are summarized in Table 9, where we synthesize the humans' feedback into a high or low value per identified feature and class. The features and values that the humans identified strongly overlap with those considered for supervised learning, e.g., level of detail (high for experts), subject matter expertise (high for experts), emotionality (high for laymen), and formal (experts) versus informal (laymen) writing styles. Beyond that, the close reading analysis also revealed additional features, e.g., differences in the usage of personal pronouns ("I" for laymen) and comparatives and superlatives (high for laymen), which can be used in future work, e.g. as new features. Overall, the majority of cases where both the classifier and the humans were incorrect are short expert reviews.

		Prediction	
		Expert	Layman
Truth	Expert	93.36%	6.64%
	Layman	13.83%	86.17%

Table 8: Error analysis

6 Discussion, Conclusions and Limitations

We have developed a binary classifier that predicts whether a review was authored by an expert or a layman with an accuracy of (90.70% (F-measure). Our work is novel with respect to its goal, focus, and potential applications. While prior work has focused on predicting commercially motivated

Features identified by human coders	**Expert**	**Layman**
Deep analysis including identification of different opinions about a given topic	High	Low
Technical details, e.g. running time, and screenings references, such as film festivals and award nominations	High	Low
Movie jargon, subject matter expertise about film-making ("guerrilla filmmaking style")	High	Low
Words and short phrases with strong emotions ("I strongly recommend this film to any ocean lover")	Low	High
Comparatives and superlatives ("Charles Ferguson made the best documentary")	Low	High
Personal pronouns used as self-reference to reviewer ("I")	Low	High
Questions to convey disbelief ("What about the effort to find the person, or people who did do it?")	Low	High
Casual, informal style ("In the second doc").	Low	High
Words with all letters in upper case ("Watch this film NOW").	Low	High
Smaller variability in vocabulary (e.g., duplicated words/phrases)	Low	High
Grammatical errors, sloppiness	Low	High

Table 9: Manually identified features and values

features, e.g. the rating, helpfulness, and sentiment of reviews, we aim to predict the type of author. We believe that this work enhances our understanding of the impact of issue-focused media, in our case documentary films, on different types of users.

Our results suggest that experts write comparatively lengthier and more detailed reviews with more complex grammar, higher entropy, and lower emotionality. Laymen are less object (noun) and more action (verb) oriented, and engage more emotionally with the content of a film. The relevance of these features was empirically demonstrated, and then manually verified and extended by human judges.

The generalizability of our findings is limited by several choices we made: First, we worked with data from two particular sources, i.e., expert reviews published mainly in major newspapers and retrieved from LexisNexis Academic, and laymen reviews collected from Amazon. Although our data come from different platforms, we argue that the considered text features are not a function of the type of source, but of the way in which experts versus laymen express their impressions of a film. Second, the first choice furthermore entails the assumption that user-generated reviews on Amazon are authored by laymen, while professional writers author expert reviews. We have shown that this assumption does not always hold: For the case of erroneous predictions, laymen are about twice as likely to be identified as experts than vice versa. Third, we also tried to use additional sources of reviews, but were constrained by the terms of service for these sites (e.g., Metacritic, Rotten Tomatoes). Fourth, since our primary goal is feature selection and analysis, we report results based on only one learning algorithm, namely SVM. In the future, we plan to explore the contribution of our binary classifier as a feature for predicting review ratings, helpfulness and authenticity. We will also test if prediction accuracy can be further increased by adding features that were detected by our human annotators during the error analysis process, namely the consideration of personal pronouns, comparatives and superlatives.

Finally, even though it is peripheral to the NLP work presented in this paper, looking at our results from a social or media impact assessment perspective, we find that laymen do engage with the content of a film, and contextualize issues raised in documentaries in their personal lives. These effects indicate public awareness and impact on information consumers. Our work might help researchers and practitioners in the field of impact assessment to understand how different groups of stakeholders reflect on a topic or a work of art (Barrett & Leddy, 2008; Chattoo & Das, 2014; Clark & Abrash, 2011; Diesner et al., 2014; Green & Patel, 2013; John & James, 2011; Napoli, 2014).

Acknowledgement

This work was supported by the FORD Foundation, grant 0155-0370, and by a faculty fellowship from the National Center of Supercomputing Applications (NCSA) at UIUC. We are also grateful to Amazon for giving us permission to collect reviews from their website. We also thank Sandra Franco and Harathi Korrapati from UIUC for their help with this paper.

Reference

Amblee, N., & Bui, T. (2007). Freeware downloads: An empirical investigation into the impact of expert and user reviews on demand for digital goods. In *Proceedings of the Americas Conference on Information Systems, AMCIS*, Paper 21.

Barrett, D., & Leddy, S. (2008). Assessing creative media's social impact. The Fledgling Fund.

Basuroy, S., Chatterjee, S., & Ravid, S. A. (2003). How critical are critical reviews? The box office effects of film critics, star power, and budgets. *Journal of Marketing, 67*(4), 103-117.

Campbell, G. M., Buckhoff, M., & Dowell, J. A. Transition Words. https://msu.edu/~jdowell/135/transw.html

Chattoo, C. B., & Das, A. (2014). Assessing the social impact of issues-focused documentaries: Research methods & future considerations. Center for Media and Social Impact, School of Communication at American University.

Clark, J., & Abrash, B. (2011). Social justice documentary: Designing for impact. Center for Social Media, School of Communication at American University.

de Albornoz, J. C., Plaza, L., Gervás, P., & Díaz, A. (2011). A Joint Model of Feature Mining and Sentiment Analysis for Product Review Rating. *Advances in Information Retrieval* (pp. 55-66). Berlin Heidelberg: Springer.

de Jong, I. K., & Burgers, C. (2013). Do consumer critics write differently from professional critics? A genre analysis of online film reviews. *Discourse, Context & Media, 2*(2), 75-83.

De Marneffe, M.-C., MacCartney, B., & Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of International Conference on Language Resources and Evaluation, LREC*, (pp. 449-454), (Vol. 6).

De Marneffe, M.-C., & Manning, C. D. (2008). Stanford typed dependencies manual. Technical report, Stanford University.

Diesner, J., Kim, J., & Pak, S. (2014). Computational impact assessment of social justice documentaries. *Journal of Electronic Publishing, 17*(3).

Diesner, J., Rezapour, R., & Jiang, M. (2016). Assessing public awareness of social justice documentary films based on news coverage versus social media. In *Proceedings of the iConference*.

Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D., & Weingessel, A. (2011). e1071: Misc Functions of the Department of Statistics (e1071), TU Wien. R package version 1.5-27.

Flanagin, A. J., & Metzger, M. J. (2013). Trusting expert-versus user-generated ratings online: The role of information volume, valence, and consumer characteristics. *Computers in Human Behavior, 29*(4), 1626-1634.

Ghose, A., & Ipeirotis, P. G. (2011). Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics. *IEEE Transactions on Knowledge and Data Engineering, 23*(10), 1498-1512.

Green, D., & Patel, M. (2013). Deepening engagement for lasting impact; A framework for measuring media performance and results. John S. and James L. Knight Foundation and Bill & Melinda Gates Foundation.

Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining, SIGKDD*, (pp. 168-177), (Vol. 04), ACM.

Jiang, M., & Diesner, J. (2016). Issue-focused documentaries versus other films: Rating and type prediction based on user-authored reviews. In *Proceedings of the 27th ACM Conference on Hypertext and Social Media*, (pp. 225-230), ACM.

Jindal, N., & Liu, B. (2007). Review spam detection. In *Proceedings of the 16th International Conference on World Wide Web, WWW*, (pp. 1189-1190), ACM.

Jindal, N., Liu, B., & Lim, E.-P. (2010). Finding unusual review patterns using unexpected rules. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM*, (pp. 1549-1552), ACM.

John, S., & James, L. (2011). Impact: A practical guide for evaluating community information projects. Knight Foundation.

Kim, S.-M., Pantel, P., Chklovski, T., & Pennacchiotti, M. (2006). Automatically assessing review helpfulness. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, EMNLP*, (pp. 423-430), Association for Computational Linguistics.

Li, F., Han, C., Huang, M., Zhu, X., Xia, Y.-J., Zhang, S., & Yu, H. (2010). Structure-aware Review Mining and Summarization. In *Proceedings of the 23rd International Conference on Computational Linguistics, COLING*, (pp. 653-661).

Liu, Y., Huang, X., An, A., & Yu, X. (2008). Modeling and predicting the helpfulness of online reviews. In *Proceedings of the IEEE International Conference on Data Mining, ICDM*, (pp. 443-452).

Mackiewicz, J. (2009). Assertions of expertise in online product reviews. *Journal of Business and Technical Communication, 24*(1), 3-28.

Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics, 19*(2), 313-330.

McAuley, J., & Leskovec, J. (2013). From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *In Proceedings of the 22nd international Conference on World Wide Web, WWW*, (pp. 897-908), ACM.

Mudambi, S. M., & Schuff, D. (2010). What makes a helpful review? A study of customer reviews on Amazon. com. *MIS quarterly, 34*(1), 185-200.

Napoli, P. (2014). Measuring media impact: An overview of the field. Media Impact Project, USC Annenberg Norman Lear Center.

Nelson, P. (1970). Information and consumer behavior. *The Journal of Political Economy, 78*(2), 311-329.

Nelson, P. (1974). Advertising as information. *The Journal of Political Economy, 82*(4), 729-754.

Pang, B., & Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *43rd Annual Meeting on Association for Computational Linguistics, ACL*, (pp. 115-124), Association for Computational Linguistics.

Plucker, J. A., Kaufman, J. C., Temple, J. S., & Qian, M. (2009). Do experts and novices evaluate movies the same way? *Psychology & Marketing, 26*(5), 470-478.

Rezapour, R., & Diesner, J. (2017). Classification and Detection of Micro-Level Impact of Issue-Focused Films based on Reviews. In *Proceedings of the 20th ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW)*, ACM.

Turney, P. D. (2002). Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL*, (pp. 417-424), Association for Computational Linguistics.

Weaver, W., & Shannon, C. E. (1949). The Mathematical Theory of Communication. *University of Illinois Press*. Urbana, Illinois.

Wilson, T., Wiebe, J., & Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, EMNLP*, (pp. 347-354), Association for Computational Linguistics.

Wu, G., Greene, D., & Cunningham, P. (2010). Merging multiple criteria to identify suspicious reviews. In *Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys*, (pp. 241-244), ACM.

Yang, Y., Yan, Y., Qiu, M., & Bao, F. S. (2015). Semantic Analysis and Helpfulness Prediction of Text for Online Product Reviews. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, ACL and the 7th International Joint Conference on Natural Language Processing, IJCNLP*, (pp. 38-44), (Vol. 2: Short Papers).

Zhang, R., Yu, W., Sha, C., He, X., & Zhou, A. (2015). Product-oriented review summarization and scoring. *Frontiers of Computer Science, 9*(2), 210-223.

Zhuang, L., Jing, F., & Zhu, X.-Y. (2006). Movie review mining and summarization. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management, CIKM*, (pp. 43-50), ACM.

Knowledge-Driven Event Embedding for Stock Prediction

Xiao Ding[†]*, **Yue Zhang**[‡], **Ting Liu**[†], **Junwen Duan**[†]
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{xding, tliu, jwduan}@ir.hit.edu.cn
[‡]Singapore University of Technology and Design
yue_zhang@sutd.edu.sg

Abstract

Representing structured events as vectors in continuous space offers a new way for defining dense features for natural language processing (NLP) applications. Prior work has proposed effective methods to learn event representations that can capture syntactic and semantic information over text corpus, demonstrating their effectiveness for downstream tasks such as event-driven stock prediction. On the other hand, events extracted from raw texts do not contain background knowledge on entities and relations that they are mentioned. To address this issue, this paper proposes to leverage extra information from knowledge graph, which provides ground truth such as attributes and properties of entities and encodes valuable relations between entities. Specifically, we propose a joint model to combine knowledge graph information into the objective function of an event embedding learning model. Experiments on event similarity and stock market prediction show that our model is more capable of obtaining better event embeddings and making more accurate prediction on stock market volatilities.

1 Introduction

Text mining techniques have been used to perform event-driven stock prediction (Ding et al., 2015). The main idea is to learn distributed representations of structured events (i.e. event embeddings) from text, and use them as the basis to generate textual features for predicting price movements in stock markets. Here the definition of *events* follows the open information extraction literature (Fader et al., 2011; Yates et al., 2007), which has seen applications in semantic parsing (Berant et al., 2013), information retrieval (Sun et al., 2015) and text mining (Ding et al., 2014). Formally, an event is defined as a tuple (A, P, O), where A represents the agent, P represents the predicate and O represents the object. For example, "Microsoft profit rises 11 percent" can be represented as the event tuple (A ="Microsoft profit", P ="rises", O ="11 percent"). In addition, the main advantages of *event embeddings* include (1) they can capture both the syntactic and the semantic information among events and (2) they can be used to alleviate the sparsity of discrete events compared with one-hot feature vectors. The learning principle is that events are syntactically or semantically similar should have similar vectors.

The *event embedding* method of Ding et al. (2015) is based on *word embeddings* of the agent, predicate and object of an event. Neural tensor networks are used to combine the embeddings of the three components into embedding vectors of events. For training, one component of gold-standard event is randomly flipped to synthesize negative examples. This form of event embedding method suffers from some limitations. First, the obtained event embeddings cannot capture the relationship between two syntactically or semantically similar events if they do not have similar word vectors. On the other hand, two events with similar word embeddings, such as "Steve Jobs quits Apple" and "John leaves Starbucks" may have similar embeddings despite that they are quite unrelated. One important reason for the problem is the lack of background knowledge in training event embeddings. In particular, if it is known that "Steve Jobs" is the CEO of "Apple", and "John" is likely to be a customer at "Starbucks", the two events can have very different embeddings according to their semantic differences.

*This work was done while the first author was visiting Singapore University of Technology and Design

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2133–2142, Osaka, Japan, December 11-17 2016.

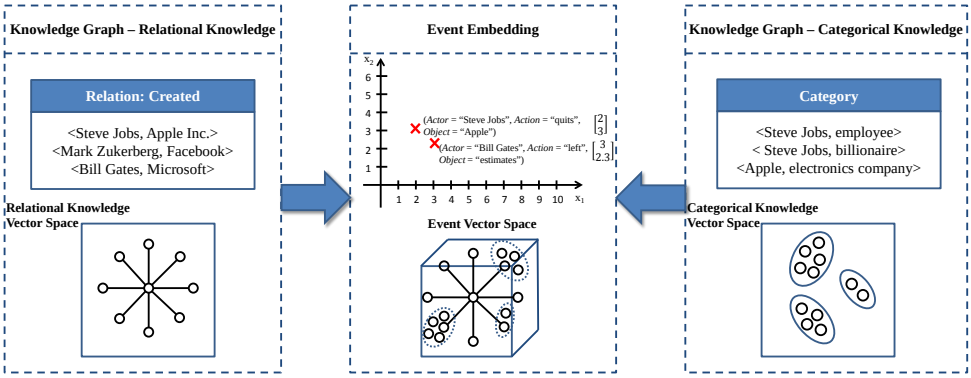


Figure 1: Incorporating knowledge graph into the learning process for event embeddings.

We propose to incorporate the external information from knowledge graphs, such as Freebase (Bollacker et al., 2008) and YAGO (Suchanek et al., 2007), into the learning process to generate better event representations. A knowledge graph stores complex structured and unstructured knowledge, and usually contains a set of vertices representing *entities* and a set of edges corresponding to the *relations* between entities. It commonly contains two forms of knowledge: *categorical knowledge* and *relational knowledge*. While categorical knowledge encodes the attributes and properties of certain entities, such as "programmer", "employee" for the person Mark Zuckerberg, relational knowledge encodes the relationship between entities, such as hasEconomicGrowth, isCEOof, etc. Both categorical knowledge and relational knowledge are useful for improving event embeddings. More specifically, categorical knowledge can be used for correlating entities with similar attributes, and relational knowledge can be used to differentiate event pairs with similar word embeddings.

We propose a novel framework for leveraging both categorical knowledge and relational knowledge in knowledge graphs for better event representations. As shown in Figure 1, we propose a coherent model to jointly embed knowledge graph and events into the same vector space, which consists of three components: the events model, the knowledge model, and the joint model. A neural tensor network is used to learn baseline event embeddings, and we define a corresponding loss function to incorporate knowledge graph information, by following recent work on multi-relation models (Socher et al., 2013).

Large-scale experiments on a YAGO corpus show that incorporating knowledge graph brings promising improvements to event embeddings. With better embeddings, we achieve better performance on stock prediction compared to the state-of-the-art methods.

2 Related Work

Stock Market Prediction There has been a line of work predicting stock markets using text information from daily news (Lavrenko et al., 2000; Schumaker and Chen, 2009; Xie et al., 2013; Peng and Jiang, 2015; Li et al., 2016). Pioneering work extracts different types of textual features from news documents, such as bags-of-words, noun phrases, named entities and structured events. Ding et al. (2014) show that structured events from open information extraction (Yates et al., 2007; Fader et al., 2011) can achieve better performance compared to conventional features, as they can capture structured relations. However, one disadvantage of structured representations of events is that they lead to increased sparsity, which potentially limits the predictive power. Ding et al. (2015) propose to address this issue by representing structured events using event embeddings, which are dense vectors. This paper proposes to leverage ground truth from knowledge graph to enhance event embeddings.

Knowledge Graph Embedding Recently, several methods have been explored to represent and encode knowledge graph (Bordes et al., 2013; Bordes et al., 2014; Chang et al., 2013; Ji et al., 2015; Lin et al., 2015) in distributed vectors. In this line of work, each entity is represented as a d-dimensional vector and each relation between two entities is modeled by using a matrix or a tensor. Most existing methods learn knowledge embeddings by minimizing a global loss function over all the entities and relations in

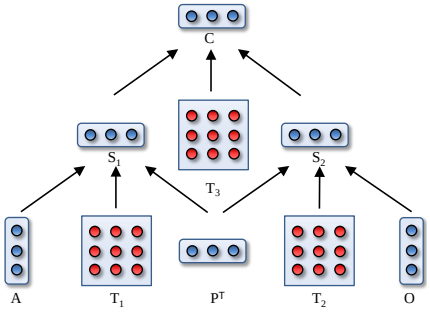


Figure 2: Baseline event-embedding model.

a knowledge graph. Entity vectors can encode global information over the knowledge graph, and hence are useful for knowledge graph completion (Socher et al., 2013). In this paper, we encode entity vectors into the learning process for event embeddings, so that information of knowledge graphs can be used for event-driven text mining and other tasks. Socher et al. (2013) has shown that previous work (Bordes et al., 2011; Jenatton et al., 2012; Bordes et al., 2012; Sutskever et al., 2009; Collobert and Weston, 2008) are special cases of their model, which is based on a neural tensor network. We follow Socher et al. (2013) and use tensors to represent relations in knowledge graph embeddings.

Our work is also related to prior research on joint embedding of words and knowledge graphs (Xu et al., 2014; Wang et al., 2014; Tian et al., 2016; Yang et al., 2014). Such work focuses on injecting semantic knowledge into distributed word representations, thus enhancing their information content. The resulting embeddings of words and phrases have been shown useful for improving NLP tasks, such as question answering and topic prediction. In comparison, our work integrates knowledge into vector representations of events, which was shown more useful than words for certain text mining tasks.

3 Knowledge-Driven Event Representations

We begin by introducing the baseline event embedding learning model, which serves as the basis of proposed framework. Then, we show how to model knowledge graph information. Subsequently, we describe the proposed joint model by integrating knowledge into the original objective function to help learn high-quality event representations. At the end of this section, we introduce the training process of the proposed framework in details.

3.1 Event Embedding

The goal of event embedding is to learn low-dimension dense vector representations for event tuples $E = (A, P, O)$, where P is the action or predicate, A is the actor or subject and O is the object on which the action is performed. We take the neural tensor network model of Ding et al. (2015) as the basis of our proposed framework. The architecture of neural tensor network for learning event embeddings is shown in Figure 2, where the bilinear tensors are used to explicitly model the relationship between the actor and the action, and that between the object and the action.

The inputs of the neural tensor network (NTN) are the word embeddings of A, P and O, and the outputs are event embeddings. We learn an initial word representation of d-dimensions ($d = 100$) from a large-scale financial news corpus, using the skip-gram algorithm (Mikolov et al., 2013). As most event arguments consist of several words, we represent the actor, action and object as the average of their word embeddings, respectively, allowing the sharing of statistical strength between the words describing each component (e.g. *Nokia's mobile phone business* and *Nokia*).

From Figure 2, $S_1 \in \mathbb{R}^d$ is computed by:

$$S_1 = g(A, P) = f\left(A^T T_1^{[1:k]} P + W \begin{bmatrix} A \\ P \end{bmatrix} + b\right), \quad (1)$$

where $T_1^{[1:k]} \in \mathbb{R}^{d \times d \times k}$ is a tensor, which is a set of k matrices, each with $d \times d$ dimensions. The bilinear tensor product $A^T T_1^{[1:k]} P$ is a vector $r \in \mathbb{R}^k$, where each entry is computed by one slice of the tensor ($r_i = A^T T_1^{[i]} P, i = 1, \cdots, k$). The other parameters are a standard feed-forward neural network, where $W \in \mathbb{R}^{k \times 2d}$ is the weight matrix, $b \in \mathbb{R}^k$ is the bias vector, and $f = tanh$ is the activation function. S_2 and C in Figure 2 are computed in the same way as S_1.

We also experiment with randomly initialized word vectors as the input of NTN, which are commonly used in previous work on structured embeddings from knowledge graphs (Bordes et al., 2011; Jenatton et al., 2012). In our case, pre-trained word embeddings give slightly better results as compared with randomly initialized embeddings.

We assume that event tuples in the training data should be scored higher than corrupted tuples, in which one of the event arguments is replaced with a random argument. Formally, the corrupted event tuple is $E^r = (A^r, P, O)$, which is derived by replacing each word in A with a random word w^r in our dictionary $\mathcal{D}$ (which contains all the words in the training data) to obtain a corrupted counterpart A^r. We calculate the *margin loss* of the two event tuples as:

$$L_{\mathcal{E}} = loss(E, E^r) = \max(0, 1 - g(E) + g(E^r)) + \lambda \|\Phi\|_2^2, \tag{2}$$

where $\Phi = (T_1, T_2, T_3, W, b)$ is the set of model parameters. The standard L_2 regularization is used, for which the weight λ is set as 0.0001. The algorithm goes over the training set for multiple iterations. For each training instance, if the loss $loss(E, E^r) = \max(0, 1 - g(E) + g(E^r))$ is equal to zero, the online training algorithm continues to process the next event tuple. Otherwise, the parameters are updated to minimize the loss using standard back-propagation (BP) (Rumelhart et al., 1985).

3.2 Knowledge Graph Embedding

Knowledge graph embedding is mainly used to encode whether two entities (e_1, e_2) are in a certain relationship R (e.g. (Steve Jobs, Apple Inc.) has a relation "created"). To incorporate categorical knowledge, we expand the definition of (e_1, R, e_2) so that e_2 can also be an attribute of e_1 and R can also be an attribute type (e.g. (Steve Jobs, Profession, CEO)). Inspired by recent studies on learning distributed representations of multi-relational data from knowledge graph (Socher et al., 2013), we use a neural tensor network framework for knowledge graph embedding.

There are two significant differences when a neural tensor network is used for knowledge graph embedding, compared to when it is used for event embedding. First, we model a relation type in a knowledge graph by using a tensor rather than a vector. This is because the number of relation types in knowledge graph is limited, and using a tensor can increase the expressive power. Second, we use a simpler neural tensor network model to learn knowledge graph embedding, which is easier to train. In contrast, the baseline event embedding model uses a recursive neural tensor network architecture to preserve the original structure of events.

The neural tensor network replaces a standard linear neural network layer with a bilinear tensor layer, which directly relates the two entity vectors across multiple dimensions. The model computes the probability that two entities are in a certain relationship by the following function:

$$g(e_1, R, e_2) = \mu_R^T f\left(e_1^T H_R^{[1:k]} e_2 + V_R \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + b_R\right), \tag{3}$$

where $f = tanh$ is the activation function, applied element-wise, $H_R^{[1:k]} \in \mathbb{R}^{d \times d \times k}$ is a tensor that consists of a set of $d \times d$ matrices, and the bilinear tensor product $e_1^T H_R^{[1:k]} e_2$ results in a vector $x \in \mathbb{R}^k$, where each entry is computed by on slice $i = 1, \cdots, k$ of the tensor: $x_i = e_1^T H_R^{[i]} e_2$. The other parameters for the relation R are the standard form of a neural network, where $V_R \in \mathbb{R}^{k \times 2d}$ and $b_R \in \mathbb{R}^k$.

The main method for training relation embeddings in knowledge graphs is similar to that for training the baseline event embeddings — each relation tuple in the training set $T^{(i)} = (e_1^{(i)}, R^{(i)}, e_2^{(i)})$ should receive a higher score than a corrupted tuple, in which one of the entities is replaced with a random

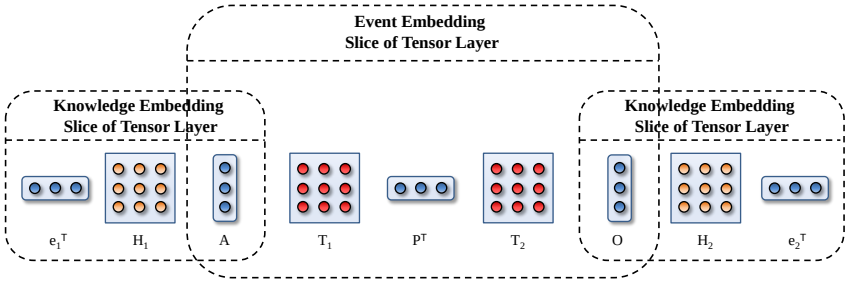


Figure 3: Architecture of the joint embedding model (only showing the tensor layer).

entity. Given a gold-standard tuple $T^{(i)} = (e_1^{(i)}, R, e_2^{(i)})$, the corresponding corrupted tuple is denoted as $T_c^{(i)} = (e_1^{(i)}, R^{(i)}, e_c^{(i)})$, where $e_c^{(i)}$ is the corrupted version of $e_2^{(i)}$. The set of all parameters is $\Omega = \{\mu, H, V\}$. We minimizing the following objective:

$$L_{\mathcal{K}} = \sum_{i=i}^{N} \sum_{m=1}^{M} max\left(0, 1 - g(T^{(i)}) + g(T_c^{(i)})\right) + \lambda \|\Omega\|_2^2, \tag{4}$$

where N is the number of training tuples, and the training objective scores the correct relation tuple higher than its corrupted version up to a margin of 1. For each correct tuple, we sample M randomly corrupted counterparts. We use standard L_2 regularization of all parameters, weighted by the hyperparameter λ.

3.3 Joint Knowledge and Event Embedding

Given a training event corpus $\mathcal{E}$ and a set $\mathcal{K}$ of relation tuples extracted from a knowledge graph, our model jointly minimizes a linear combination of the loss functions on both events and knowledge:

$$L = \alpha L_{\mathcal{E}} + (1 - \alpha) L_{\mathcal{K}} \tag{5}$$

where $\alpha \in [0, 1]$ is a model parameter to weight the two loss functions (the best development results were obtained with $\alpha = 0.4$). $\mathcal{E}$ and $\mathcal{K}$ share the same parameters — the embedding vectors for entities in events and their corresponding entities in knowledge graph are required to be the same.

Figure 3 shows the architecture of the proposed joint model. The algorithm is centralized on the event tuple (A, P, O), which is used to train the tensor values T_1, T_2 and T_3 in Figure 2. Two relations in the knowledge graph, H_1, H_2, which share entities A and O with the event, are shown on the two sides of Figure 3, respectively. By the shared entities A and O between events and knowledge relations. T_1, T_2, T_3, H_1 and H_2 can be trained simultaneously. Here relational knowledge can help incorporate information of the learned event representations by utilizing ground-truth relations between entities. Categorical knowledge can encode the entity information of the learned event representations by clustering similar entities together. The base event embedding model can preserve the structures of events.

Training The joint model is trained by taking the derivatives of the joint objective function with respect to the four groups of parameters T_1, T_2, H_1 and H_2, respectively. We have four derivatives for the i'th slice of the full tensor:

$$\frac{\partial g(e_1, R, A)}{\partial H_1^{[i]}} = \mu_i f'(z_i) e_1 A^T, z_i = e_1^T H^{[i]} A + V_i \begin{bmatrix} e_1 \\ A \end{bmatrix} + b_i; \frac{\partial g(A, P)}{\partial T_1^{[i]}} = \mu_i f'(z_i) A P^T, z_i = A^T T^{[i]} P + W_i \begin{bmatrix} A \\ P \end{bmatrix} + b_i$$

$$\frac{\partial g(O, R, e_2)}{\partial H_2^{[i]}} = \mu_i f'(z_i) O e_2^T, z_i = O^T H^{[i]} e_2 + V_i \begin{bmatrix} O \\ e_2 \end{bmatrix} + b_i; \frac{\partial g(P, O)}{\partial T_2^{[i]}} = \mu_i f'(z_i) P O^T, z_i = P^T T^{[i]} O + W_i \begin{bmatrix} P \\ O \end{bmatrix} + b_i$$

Table 1: Statistics of datasets.

	Training	Development	Test
#documents	442,933	110,733	110,733
#words	333,287,477	83,247,132	83,321,869
#events	295,791	34,868	35,603
time interval	02/10/2006 - 18/06/2012	19/06/2012 - 21/02/2013	22/02/2013 - 21/11/2013

z_i denotes the i'th element of the hidden tensor layer. We use minibatched L-BFGS (Nocedal, 1980) for optimization, which converges to a local optimum of our non-convex objective function. The rest of the model parameters, including μ, T_3, W, b and V, are trained in the same way as the baseline single models, for which the derivatives are calculated using standard back-propagation. It is the shared entities A and O that allow information exchange in training the values of T_1, T_2, H_1 and H_2, thereby improving the vector representations of structured events through relational and categorical knowledge.

4 Experiments

The performance of our knowledge-powered event embedding model is compared with state-of-the-art baselines by evaluating the quality of learned event embeddings on two tasks: event similarity and stock market prediction.

4.1 Experimental Settings

We use publicly available financial news from Reuters and Bloomberg over the period from October 2006 to November 2013, released by Ding et al. (2014). There are 106,521 documents in total from Reuters News, from which we extracted 83,468 structured events. From Bloomberg News, there are 447,145 documents, from which 282,794 structured events are extracted.

The structured events are extracted from news text using Open IE (Fader et al., 2011) and dependency parsing (Zhang and Clark, 2011), by strictly following the method of Ding et al. (2015). The timestamps of the news are also extracted, for alignment with stock price information. We conduct stock market prediction experiments on predicting the Standard & Poor's 500 stock (S&P 500) index and its individual stocks, obtaining indices and prices from Yahoo Finance. Detail statistics of the training, development (tuning) and test sets are shown in Table 1. For training knowledge-driven event embeddings, we use YAGO as the knowledge graph. The full knowledge graph consists of 10 million entities and 120 million facts, in more than 100 predefined relation types. We extract a sub knowledge graph of two thousand entities and more than 30 thousand relations for the experiments, which contains knowledge relevant to our news event data.

4.2 Task Description

Event Similarity We investigate whether the similarity between vector event representations is consistent with human-labeled event similarity, and whether better event representation is more useful for stock prediction. As there is no publicly available event similarity evaluation data, we conduct a set of human evaluations. Each event pair is associated with three independent human judgments on similarity and relatedness on a scale from 0 to 5, where 0 means that the event pair is completely dissimilar, and 5 indicates that the event pair has a strong similarity relation. For example, (Steve Jobs, quits, Apple) and (Steve Ballmer, quits, Microsoft) received an average score of 4.6, while (Steve Jobs, quits, Apple) and (John, leaves, Starbucks) received an average score of 0.4.

Similarity scores are computed by cosine similarity of embedding vectors for each event pair, based on which a ranked list is constructed. It is compared to the ranked list produced by the manual similarity scores according to human judgments. To evaluate the consistency between two ranking lists, we use Spearman's Rank Correlation ($\rho \in [-1, 1]$). A higher ρ corresponds to better event representation vectors. We compare our knowledge-driven event embedding (denoted as KGEB) with the baseline method proposed by Ding et al. (2015) (denoted as EB), and discrete event vectors with semantic lexicons based generalization method proposed by Ding et al. (2014) (denoted as DE).

Table 2: Experimental results on event similarity and its effect on S&P 500 index prediction. The improvement is significant at $p < 0.05$.

Methods	Spearman's Rank Correlation	Acc	MCC
DE	0.437	58.83%	0.1623
EB	0.591	64.21%	0.4035
KGEB	**0.616**	**66.93%**	**0.5072**

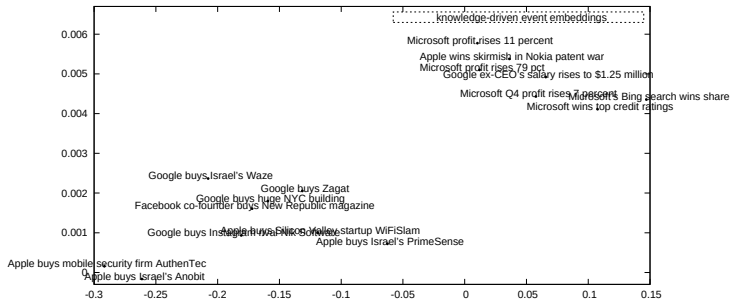


Figure 4: Two-dimensional PCA projection of 100-dimensional knowledge-driven event vectors.

Stock Prediction The stock prediction task can be treated as a binary classification problem. Following Tetlock et al. (2008), we automatically align 1,782 instances of daily trading data with news documents from the previous day. Specifically, we use the news information in day $t-1$ to predict price movements of stock market in day t. The output classification result [Class +1] represents that the stock closing price will increase compared with the opening price in day t, and [Class -1] represents that the stock closing price will decrease compared with the opening price in day t. Following Das and Chen (2007) and Xie et al. (2013), the standard measure of accuracy (Acc) and Matthews Correlation Cofficient (MCC) are used to evaluate the performances on S&P 500 index prediction and individual stock prediction.

The baseline methods are three state-of-the-art news-based stock market prediction systems: Luss and d'Aspremont et al. (2012) propose using bags-of-words to represent news documents, and constructing the prediction model by using Support Vector Machines (SVMs); Ding et al. (2014) report a system that uses structured event tuples $E = (A, P, O)$ to represent news documents, and investigate the complex hidden relationships between events and stock price movements by using a standard feedforward neural network; Ding et al. (2015) learn event embeddings for representing news documents, and build a prediction model based on a deep convolutional neural network. Ding et al. (2015) show that deep convolutional neural networks (CNN) are more powerful than SVMs and standard feedforward neural networks. As a result, we use CNN as the prediction model.

4.3 Results

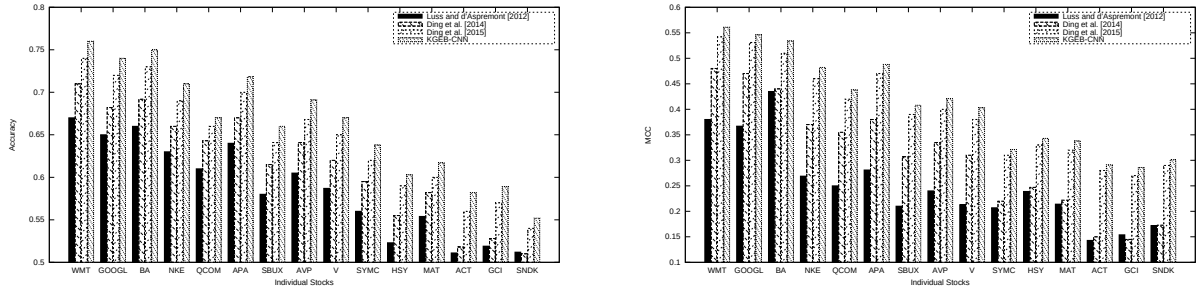
Event Similarity As shown in Table 2, we compare the performance of different event representation methods and their effects on S&P 500 index prediction. We find that although discrete event vectors are generalized by semantic lexicons (WordNet and VerbNet), the performance of KGEB is dramatically better than DE. This is mainly because the word coverage of semantic lexicons is limited, and the discrete representation is highly sparse. KGEB achieves better performance compared with EB on this task. The main performance gain results from better similarity between semantically related but lexically different events, thanks to the integration of knowledge. For example, "Steve Jobs quits Apple" and "John leaves Starbucks" have dissimilar vectors although they share similar word embeddings, as Steve Jobs is the CEO of Apple Inc. but John has no relationship with Starbucks encoded in knowledge graph. With the best event representation method, KGEB-based stock prediction achieves the best performance.

Figure 4 shows case studies on events about Google, Apple, Microsoft and Facebook. In particular, we apply two-dimensional PCA projection on the 100-dimensional event embeddings. It can be seen from the figure that by incorporating knowledge graph information, the joint model allows those events that correspond to the same semantic or topic to be close to each other.

S&P 500 Index Prediction We test the influence of knowledge-driven event embeddings on stock prediction by comparing KGEB with bag-of-words representations (Luss and d'Aspremont, 2012), structured

Table 3: Experimental results on index prediction.

	Acc	MCC
Luss and d'Aspremont (2012)	56.38%	0.0711
Ding et al. (2014)	58.83%	0.1623
WB-CNN	60.57%	0.1986
Ding et al. (2015)	64.21%	0.4035
KGEB-CNN	**66.93%**	**0.5072**



(a) Accuarcy (b) MCC

Figure 5: Experimental results on individual stock prediction (companies are named by ticker symbols).

event representations (Ding et al., 2014), baseline event embedding representations (Ding et al., 2015) and word embedding representations on the test dataset. A word embedding input (WB) consists of the sum of each word vector in a document; it addresses sparsity in word-based inputs, and can serve as a baseline embedding method. The experimental results are shown in Table 3. We find that:

(1) Comparison between the word-based models and event-based models (e.g. Luss and d'Aspremont (2012) vs Ding et al. (2014), WB-CNN vs Ding et al. (2015), WB-CNN vs KGEB-CNN) shows that events are more capable for representing news documents for stock prediction.

(2) Comparison between Ding et al. (2015) and KGEB-CNN shows that knowledge-driven event embeddings are more powerful than the baseline event embeddings. The main reason is that knowledge graph provides valuable ground-truth knowledge, which is helpful for learning better event embeddings. For example, "Chrysler recalls 919,545 Jeep SUVs" and "GM recalls nearly 474,000 cars" can be related, as "Chrysler" and "GM" are two automobile manufacturers, and Jeep SUV is a car model, which are recorded in the knowledge graph.

Individual Stock Prediction We compare our knowledge-driven event embeddings with the baseline methods on individual stock prediction, using the 15 companies selected by Ding et al. (2015) from S&P 500. The list consists of samples from high-ranking, middle-ranking, and low-ranking companies from S&P 500 according to the Fortune Magazine. The results are shown in Figure 5 (as space is limited, we only show comparison between KGEB-CNN and the three baselines). We find that knowledge-driven event embeddings achieve consistently better performances compared to the three baseline methods, on both S&P 500 index prediction and individual stock prediction. In most previous work, the accuracies of individual stock prediction are higher when only company-related news are used as inputs, compared with when sector-related news are used (Ding et al., 2014). This is because it is difficult to investigate the relationship among companies, and therefore news about other companies can be noise for predicting the stock prices of a company. However, knowledge graph can provide attributes of entities and relations between them, hence it is possible to learn more information from related companies to help decide the direction of individual stock price movements. For example, given that the news "GM recalls nearly 474,000 cars" leads to its stock price decrease in the training data, it can be predicted that Ford shares will fall next day according to the news "Ford is recalling about 433,000 2015 Focus" in the test data.

5 Conclusion

High-quality event representations are valuable for many text mining and NLP downstream applications. This paper proposed to incorporate knowledge graph into the learning process of event embeddings, which can encode valuable background knowledge. Experimental results on event similarity and stock

prediction showed that knowledge-powered event embeddings can improve the quality of event representations and benefit the downstream application.

Acknowledgments

We thank the anonymous reviewers for their constructive comments, and gratefully acknowledge the support of the National Key Basic Research Program of China (973 Program) of China via Grant 2014CB340503, the National Natural Science Foundation of China (NSFC) via Grant 61472107 and 71532004, the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, October.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, number EPFL-CONF-192344.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence and Statistics*, pages 127–135.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-relational latent semantic analysis. In *EMNLP*, pages 1602–1612.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Sanjiv R Das and Mike Y Chen. 2007. Yahoo! for amazon: Sentiment extraction from small talk on the web. *Management Science*, 53(9):1375–1388.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2014. Using structured events to predict stock price movement: An empirical investigation. In *Proc. of EMNLP*, pages 1415–1425, October.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *Proceedings of IJCAI*, BueNos Aires, Argentina, August.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proc. of EMNLP*, pages 1535–1545.

Rodolphe Jenatton, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski. 2012. A latent factor model for highly multi-relational data. In *Advances in Neural Information Processing Systems*, pages 3167–3175.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of ACL*, pages 687–696.

Victor Lavrenko, Matt Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. 2000. Mining of concurrent text and time series. In *KDD-2000 Workshop on Text Mining*, pages 37–44.

Qing Li, Jun Wang, Feng Wang, Ping Li, Ling Liu, and Yuanzhu Chen. 2016. The role of social sentiment in stock markets: a view from joint effects of multiple information sources. *Multimedia Tools and Applications*, pages 1–31.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

Ronny Luss and Alexandre d'Aspremont. 2012. Predicting abnormal returns from news using text classification. *Quantitative Finance*, (ahead-of-print):1–14.

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

Jorge Nocedal. 1980. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782.

Yangtuo Peng and Hui Jiang. 2015. Leverage financial news to predict stock price movements using word embeddings and deep neural networks. *arXiv preprint arXiv:1506.07220*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. Learning internal representations by error propagation. Technical report, DTIC Document.

Robert P Schumaker and Hsinchun Chen. 2009. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):12.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks For Knowledge Base Completion. In *Advances in Neural Information Processing Systems 26*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Rui Sun, Yue Zhang, Meishan Zhang, and Donghong Ji. 2015. Event-driven headline generation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 462–472, July.

Ilya Sutskever, Joshua B Tenenbaum, and Ruslan R Salakhutdinov. 2009. Modelling relational data using bayesian clustered tensor factorization. In *Advances in neural information processing systems*, pages 1821–1828.

Paul C Tetlock, Maytal Saar-Tsechansky, and Sofus Macskassy. 2008. More than words: Quantifying language to measure firms' fundamentals. *The Journal of Finance*, 63(3):1437–1467.

Fei Tian, Bin Gao, En-Hong Chen, and Tie-Yan Liu. 2016. Learning better word embedding by asymmetric low-rank projection of knowledge graph. *Journal of Computer Science and Technology*, 31(3):624–634.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119. Citeseer.

Boyi Xie, Rebecca J. Passonneau, Leon Wu, and Germán G. Creamer. 2013. Semantic frames to predict stock price movement. In *Proc. of ACL (Volume 1: Long Papers)*, pages 873–883, August.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1219–1228. ACM.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 645–650.

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. 2007. Textrunner: open information extraction on the web. In *Proc. of NAACL: Demonstrations*, pages 25–26. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151.

Distributed Representations for Building Profiles of Users and Items from Text Reviews

Wenliang Chen[†], **Zhenjie Zhang**[‡], **Zhenghua Li**[†], **Min Zhang**[†]
[†]School of Computer Science and Technology, Soochow University, China
[‡]Advanced Digital Sciences Center, Illinois at Singapore Pte. Ltd., Singapore
{wlchen, zhli13, minzhang}@suda.edu.cn
zhenjie@adsc.com.sg

Abstract

In this paper, we propose an approach to learn distributed representations of users and items from text comments for recommendation systems. Traditional recommendation algorithms, e.g. collaborative filtering and matrix completion, are not designed to exploit the key information hidden in the text comments, while existing opinion mining methods do not provide direct support to recommendation systems with useful features on users and items. Our approach attempts to construct vectors to represent profiles of users and items under a unified framework to maximize word appearance likelihood. Then, the vector representations are used for a recommendation task in which we predict scores on unobserved user-item pairs without given texts. The recommendation-aware distributed representation approach is fully supported by effective and efficient learning algorithms over massive text archive. Our empirical evaluations on real datasets show that our system outperforms the state-of-the-art baseline systems.

1 Introduction

With the prosperity of Internet-based e-commerce in the last decade, millions of item (e.g., product/serivce) comments are now available online and even more are flooding in an explosive manner. Yelp (www.yelp.com), as the largest restaurant comment web site, for example, attracts more than 140 million users to contribute new text comments every month, covering almost all restaurants in USA. It is widely believed that text comments contain much richer information on users as well as items than dull scores, with far more precise descriptions on personal preferences and item features. To fully unleash the potential value underneath the text comments, huge research efforts are now devoted to the design and development of new technologies to capture and understand the semantics in the comments, particularly those associated with users and items. The text analytical outcomes are expected to support decision making and provide new business opportunities in e-commerce.

Recommendation is recognized as one of the most important components in e-commerce systems, driving the growth of profits. Traditional recommendation systems rely on scores over user-item pairs under a bipartite graph model, such that accurate score prediction over unobserved user-item pair helps the system to identify potential interested buyers. A huge bulk of prediction and recommendation strategies are proposed in this domain, e.g. collaborative filtering and matrix completion (Sarwar et al., 2001; Koren et al., 2009). Because of the limited information included in each individual score, such recommendation strategies commonly suffer from the cold-start problem (Zhou et al., 2011), which returns poor recommendations to newcoming users. While text mining on comment archive may provide important information to the recommendation engine, even for fresh users, the integration of such information adds new complexity and challenges to the system, since the text mining results are not directly useful to the recommendation algorithms. In particular, most of the opinion mining techniques available in the literature (Liu and Zhang, 2012; Melville et al., 2002) do not distinguish attributes on user preference from attributes of items. The recommendation engines are thus unable to understand the underlying reasons behind the good and poor opinions from the users on the items. While there are

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2143–2153, Osaka, Japan, December 11-17 2016.

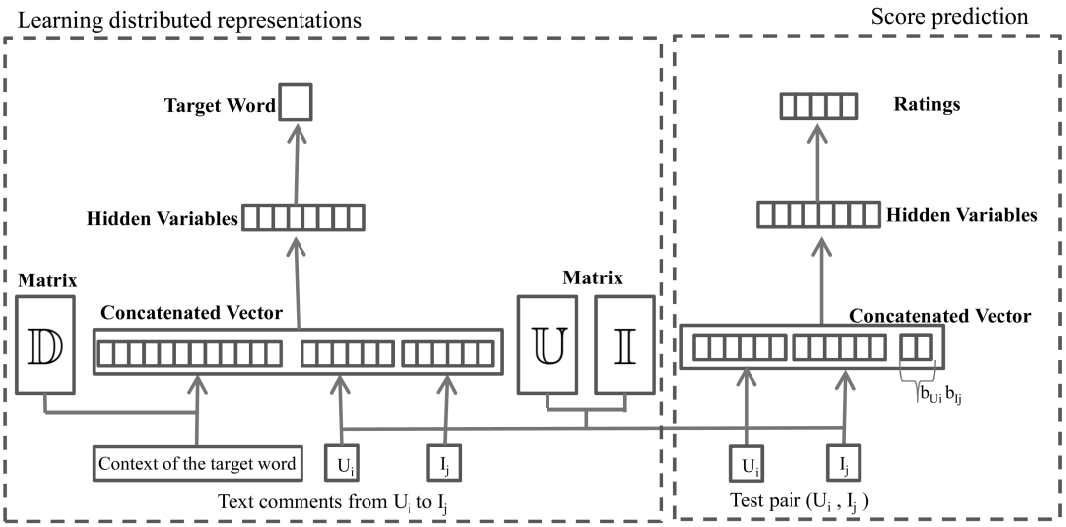


Figure 1: Our framework for learning distributed representations for recommendation

attempts on tightly connecting recommendation and text mining, e.g. (McAuley and Leskovec, 2013; Almahairi et al., 2015), these studies make strong assumptions on the generative mechanism behind the text comments and scores, which may not correctly reflect the interaction between user preferences and item attributes appropriately.

To better exploit the value of text comments for recommendation engine, we propose a simple yet effective approach inspired by the successful distributed representation models on text semantics extraction (Le and Mikolov, 2014; Mikolov et al., 2013). In our approach, we learn distributed representations from text comments to represent users and items. Our approach has a number of unique advantages, which make it a promising alternative to existing feature extraction methods used in comment-based recommendation systems. Firstly, the distributed representations on users and items are aligned, enabling the employment of a huge cluster of classification approaches on prediction and recommendation. Secondly, distributed representations do not rely on any assumption on the generative procedures over the user-generated contents. This feature generally avoids the potential bias on the analytical outcomes, caused by inappropriate setting on the priors, and finally saving computation overheads. Thirdly, the distributed representations work well on both long and short texts. It is thus likely to enhance the usefulness of the mining outcomes, by training over text data from multiple sources, e.g. food blogs and restaurant comments.

In our approach, we adapt the vectorization techniques used in the training of distributed representations over words and documents, to address the problem of factor decomposition between user preferences and item attributes. User vectors and item vectors are introduced as independent representations to the target users and items correspondingly and are learned by a neural network. In the model, the user vector is shared across the comments written by the same user and the item vector is shared across the comments written on the same item. Then, the vector representations are included in the training of the task of score prediction in which we predict rates on unobserved user-item pairs without given texts. The experimental results on real datasets show that our approach generates a significant margin of performance advantage over existing methods in the task of score prediction.

2 Preliminaries

In this section, we define the task of score prediction and introduce the background of distributed representations of words and documents.

2.1 Score Prediction

Assume that we have a fixed group of n users $\mathbb{U} = \{U_1, U_2, \ldots, U_n\}$ and a fixed group of m items $\mathbb{I} = \{I_1, I_2, \ldots, I_m\}$. Here, an item I_j can be either a product or a service, provided to all the users in $\mathbb{U}$ without any restriction. Each comment C_k in database $\mathbb{C} = \{C_1, C_2, \ldots, C_l\}$ composes a sequence of L_k words $\{w_1, w_2, \ldots, w_{L_k}\}$, with every word drawn from a known dictionary $\mathbb{D}$. Each comment C_k is also uniquely associated with a user $U_i \in \mathbb{U}$, an item $I_j \in \mathbb{I}$ and a score S_k. It indicates that user U_i purchases item I_j with score S_k and detailed text comments in C_k. The recommendation system is expected to return a group of items to each user U_i, such that the user U_i is more likely to buy these items with high scores and good comments. Therefore, the problem of recommendation is usually transformed into the score prediction problem, as formally defined below.

Problem 1 ***Score Prediction*** *Given $(\mathbb{U}, \mathbb{I}, \mathbb{C})$, the problem of score prediction is to estimate the score over unseen user-item pair (U_i, I_j) without text comments, where $U_i \in \mathbb{U}$ and $I_j \in \mathbb{I}$.*

Obviously, the score prediction is quite different from text-based sentiment classification which has text information in the testing stage. The recommendation system is more efficient, if the score prediction returns more accurate estimations.

2.2 Distributed Representations

Here, we briefly review the approaches of learning distributed representations of words and documents(Mikolov et al., 2013; Le and Mikolov, 2014). Word distributed representation denotes semantical meanings of the words extracted from large-scale text archive in a unified way (Mikolov et al., 2013). Each word is represented by a vector of fixed length L, such that 1) semantically similar words are close to each other in the new vector space; and 2) the semantics of words are compositional by vector operations. To simplify the notation, we use w to denote a word in the dictionary, as well as its corresponding vector representation. Given a sequence of words, i.e. $w_1, \ldots, w_T$, the distributed representations of the words are calculated by maximizing the likelihood of observing the words in the sequence, based on the neighbor words in the sequence. Mathematically, with the specified neighborhood width d, this objective is formalized as

$$\frac{1}{T}\sum_{i=1}^{T} \log \Pr\left(w_i \mid w_{i-d}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+d}\right).$$

For words at the beginning and end of the sequence, special null words are inserted as the padding words. To generate the vector representations of the words, the algorithm runs an optimization over the vectors as well as a group of weights, with a weight vector v_{w_i} of length $(2d-1)L$ associated with every word w_i. Given the representations of the words in form of vectors, the vector representations of the words in the neighborhood of w_i are concatenated, resulting in a long vector x_i of length $(2d-1)L$. The probability of observing w_i in the context of x_i is estimated by the following equation:

$$\Pr\left(w_i \mid x_i\right) = softmax(W \cdot x_i + b). \tag{1}$$

where W is the weight matrix connecting the input and the output and b is a bias vector. To efficiently optimize the weights and word representations, a number of optimization tricks, including hierarchical softmax and negative sampling are suggested in (Mikolov et al., 2013), in order to train the representations in reasonable time. The detail of learning procedure can be found in the paper of (Mikolov et al., 2013).

Paragraph (or document) vector is an extension of word distributed representation, by introducing a global vector indicating the topics of the whole paragraph (Le and Mikolov, 2014). When generating the representation of the neighborhood x_i, the paragraph vector, another vector of fixed length across paragraphs, is included. The paragraph vector is effective on capturing the context of the words, which can be used to enhance the accuracy of the word prediction model and extract overall topics of the whole paragraph.

3 Our Approach

Our approach for learning distributed representations is inspired by the models for learning the vectors of words and documents in (Le and Mikolov, 2014; Mikolov et al., 2013). In our solution, as presented in Figure 1, there are three types of distributed representations built and utilized in the learning phase, including word representation, user representation and item representation. Each representation is a vector of fixed length in its respective domain, say W, U and I respectively. Intuitively, the word representation denotes the conceptual meanings of the words in a latent space. Similarly, the user representation denotes the personal preferences over the items, and item representation denotes the important attributes of the items. The dimensions of the user representations, for example, are expected to indicate personal preference and behavior patterns, such as 1) whether the user prefers cheap items; or 2) whether the user prefers better service to good foods during dining in restaurants. We hereby emphasize that such preferences and profiles are automatically extracted from the data, without any supervision or manual labeling involved.

When the context is clear, we use U_i (resp. I_j) to denote both the user identity (resp. item identity) as well as its corresponding representation vector. Similarly, we misuse w to denote a word in dictionary $\mathbb{D}$ and its word representation vector. There are two parts in our approach: 1) Learning distributed representations of words, users, and items; 2) Score prediction for pairs of users and items without text comments.

3.1 The Learning Model

During the learning phase, the contexts are fixed-length and sampled from a sliding window over the text comments. The matrix $\mathbb{D}$, with stacked word vectors, is shared across all the text comments. However, the user vector is shared across the comments written by the same user and the item vector is shared across the comments written on the same item.

By looking up the matrices, we obtain the vectors of users, items, and words. After concatenating all relevant vectors in order, including user vector, item vector and neighborhood word vectors, a bi-layer neural network model is built on top of the concatenated vector to predict the words as shown in the left part of Figure 1.

Suppose that the target word is w and the input (the concatenated vector) is X_w. A intermediate layer is applied on the input vector X_w to generate a fixed length vector with L_h binary variables V_h. Each variable $v_i \in V_h$ is independently activated, based on the weight assignment on the edges between input vector and intermediate layer, following the logistic function:

$$\Pr(v_i = 1) = \frac{1}{1 + \exp\left(-\left(\sum_{x_j \in X_w} x_j \cdot w_{ij}\right)\right)}.$$

where w_{ij} is the weight connecting from the j^{th} dimension of the input to the i^th dimension of V_h. From the intermediate layer, our model predicts the word occurrence with another *prediction layer*. On this layer, the system employs exactly the same softmax function used in the existing word vector and paragraph vector models. The prediction function is,

$$\Pr\left(w \mid V_h\right) = softmax(W_w^h \cdot V_h + b^h).$$

where W_w^h is the weight vector connecting V_h to w and b^h is the bias vector.

Based on the model above, the training of the model with the text comment archive is in two folds. Firstly, the weights in the neural network are optimized to reflect the correlation between user/item/word representations and the word occurrence likelihood. Secondly, the construction of the representation identifies meaningful semantics into the vectors, especially for the user representation and item representation. When there are multiple users and items available in the comment database, the result representations automatically reveal the actual preferences of the users and the attributes of the items, because common topics across comments are merged and appear as a dimension in the representations.

3.2 Training the Parameters

During training, the parameters we should learn include the weights in the above equations and the vector representations of users, items and words. We use stochastic gradient descent to learn the parameters. Then the *backpropagation* algorithm is employed to push the gradients to the concatenated vector level, and thus used to update the weights on the edges as well as the user/item/word vectors by exactly the same mechanism. At every step of the learning phase, we sample a fixed-length context from a random comment.

To train the weights from hidden variables to the target words, we apply *negative sampling*. The negative sampling method is a simplified variation of Noise Contrastive Estimation (Gutmann and Hyvarinen, 2012; Mhih and Teh, 2012). To compute the probabilities efficiently, we use the negative sampling method proposed by (Mikolov et al., 2013), which approximates the probability by the correct example and K negative samples for each instance. The formulation to compute $\log(\Pr(w_i|V_h))$ is,

$$\log \sigma(z_{w_i}) + \sum_{k=1}^{K} \mathbb{E}_{w_k \sim P(w)}[\log \sigma(-z_{w_k})] \tag{2}$$

where $\sigma(z_w) = 1/(1+exp(-z_w))$, $z_w = W_w^h \cdot V_h + b^h$, w_i is the target word, V_h is the vector of hidden variables, and $P(w)$ is the noise distribution on the data. We set K as 5 in our experiments as used in (Mikolov et al., 2013). We perform the iterative update after predicting the target word,

$$\theta \leftarrow \theta - \alpha(\frac{\partial \sum log(\Pr(w_i|V_h))}{\partial \theta}) \tag{3}$$

where α is the learning rate and θ is the set of parameters to learn that includes the weights of the model. The initial value of α is 0.025. During training, the learning rate is halved if the log-likelihood does not improve significantly after one update. Then, the training stops if it does not improve again.

The computation efficiency on the training procedure is excellent, because most of the computation time is spent on the training between hidden variables and target words. The training between concatenated vectors and hidden variables is much more efficient, since the number of variables involved is highly constrained, depending on the specified size of the word neighborhood and the size of the hidden variable layer.

After the learning phase, we obtain the distributed representations of users and items shown as vectors. They are used in the score prediction task.

3.3 Predicting Scores

As is described in Problem 1, the problem of score prediction is to estimate the score of a user before he/she actually purchases the item. Thus, we do not have the text comment for the given pair when testing. We build a regression/classification model on top of the user/item representations. Specifically, a prediction model is a parameterized function $F(U_i, I_j)$ with inputs of user/item vectors and outputs of score estimation. While any regression model can be employed as the parameterized function $F(\cdot)$, we use the network shown in the right part of Figure 1. First, we learn a non-linear transformation which can project the input user/item representations into a space where it becomes linearly separable. The space is a intermediate layer referred to as a hidden layer. Then, a logistic regression classifier takes the transformed vectors for predicting the scores.

As indicated in Figure 1, the input layer contain the user and item representations and the average scores (b_{U_i} and b_{I_j}) of user and item, and the output layer has possible ratings.

Given the input x, $h(x) = s(W^1 x + b^1)$ represents the hidden layer, where W^1 is the weight matrix connecting the input vector to the hidden layer and b^1 is a bias vector for the transformation. Here, we use $tanh(a) = (e^a - e^{-a})/(e^a + e^{-a})$ for function s since it is fast when training the parameters.

Specifically, the prediction is then obtained by applying:

$$P(y|x) = softmax(W^2 h(x) + b^2) \tag{4}$$

	Yelp	Dianping	Amazon
review#	1.6M	1.2M	1.2M
item#	60,785	43,141	246,200
users#	366,715	350,936	641,380
review# per item	25.81	27.98	4.73
review# per user	4.27	3.44	1.81
word# per review	128.93	91.78	167.39

Table 1: Statistics of data set

where W^2 is the weight matrix connecting the hidden layer to the output layer and b^2 is a bias vector for the classifier. We use stochastic gradient descent to learn all parameters of the model, including W^1, b^1, W^2 and b^2. Finally, the prediction of the model y_{pred} is the class whose probability is maximal:

$$y_{pred} = \text{argmax}_i P(Y = y_i|x) \tag{5}$$

4 Experiments

4.1 Dataset

We evaluate the systems on datasets from a variety of public sources. The first one is from Yelp Dataset Challenge[1], which contains about 1.6M reviews. The second one is from *Dianping* (www.dianping.com), which consists of user reviews on the restaurants located in China. We also include about 1.2M reviews from Amazon previously used for opinion analysis in (Jindal and Liu, 2008). In the datasets, each review contains a user ID, restaurant/product ID, numeric rating (from 1 to 5), and detailed comment text. We filter out the reviews that do not have comment texts. The detailed statistics of the data sets are summarized in Table 1. The reviews are written in two different languages, Yelp and Amazon using English, and Dianping using Chinese. The experiments with different languages and domains demonstrate the genericity of our approach.

For each dataset, we randomly split the whole set into two parts: 80% as training data and 20% as test data. We also randomly select 10% of training data as development data to tune the parameters of the systems.

4.2 Score Prediction

We report the mean absolute error (MAE) of the score predictor, i.e., $MAE = \frac{1}{N}\sum_{u,i\in T}|\hat{r}_{ui} - r_{ui}|$, where T is the test set, N is the total number of predicted ratings, $\hat{r}_{ui}$ and r_{ui} are the predicted rating and the user assigned rating scores for user u and item i, respectively.

We compare with four systems: 1) CF: This is a user-based Collaborative Filtering (CF) system in which we follow the description of (Sarwar et al., 2001). 2) FM: This is a latent factor model implemented in libfm[2](Rendle, 2012). 3) CTR: This is a collaborative topic regression model adapted from (Wang and Blei, 2011)[3], which is proposed to recommend scientific articles in citation networks. 4) HFT: This is the Hidden Factors as Topics (HFT) model proposed by (McAuley and Leskovec, 2013), which is a state-of-the-art system using text reviews[4]. The first two systems use rating information only and the other two systems utilize text reviews. The settings of all the systems are tuned on the development sets. According to the results on the development sets, we set user/item vectors size as 100 and number of hidden units as 50 in our system.

Table 2 shows the experimental results of score prediction, where DRT is our system. FM is better than the traditional Collaborative Filtering (CF) system. This shows that the factor model is very powerful. HFT and CTR achieve similar scores to FM on two datasets and perform better than FM on the Amazon

[1]http://www.yelp.com/dataset_challenge/

[2]http://libfm.org/

[3]https://www.cs.princeton.edu/∼chongw/software/ctr.tar.gz

[4]http://cseweb.ucsd.edu/∼jmcauley/code/code_RecSys13.tar.gz

data. Our system provides the best performance on all the datasets and is significantly better than other systems ($p < 10^{-5}$).

System	Yelp	Dianping	Amazon
CF	1.0004	0.6906	0.9924
FM	0.9276	0.6609	0.8936
CTR	0.9243	0.6619	0.8693
HFT	0.9221	0.6615	0.8658
DRT	0.9064	0.6316	0.8165

Table 2: Results of score prediction (MAE)

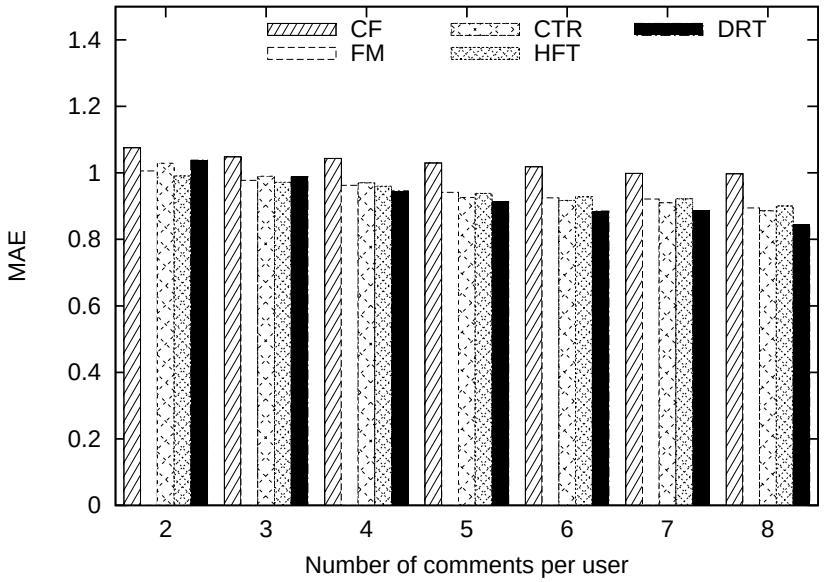


Figure 2: MAE vs number of training reviews per user (Yelp)

The number of training reviews is important for recommendation systems. We thus investigate the effect of training reviews per user. Figure 2- 4 show the MAE scores with varying number of training reviews per user. The trend of the results shows that all the systems are getting better scores, with increasing number of comments. When the number of training reviews grows larger, the performance gap between DRT and CF also grows. This fact indicates that our approach can make use of text comments to improve performance. The results also show that with 2 or 3 training reviews, DRT performs best on 3 among 6 cases, while with 4 or more training reviews, DRT can beat other approaches. In future work, we will try to improve the bad cases.

We also investigate the effect of different lengths of comments to our system (DRT). To reduce the effect of numbers of training reviews per user, we calculate the average lengths of comments written by the users who only have 4-10 training reviews. We ignore the users whose average comment length is no more than 20 words. We group the users into several groups by setting every 20 words as one BIN. For example, BIN-40 includes the users whose average comment length is in (20,40] and BIN-100 includes the users whose average comment length is in (80,100]. Figure 5 shows the relation between the MAE scores and the comment lengths. From the figure, we find that DRT roughly gets consistent performance along with the increase of comment lengths. However, this fact is against what we expect: the longer texts might result in better performance. We check some long comments in three datasets and find that most of the long comments indeed have more detailed information, but also bring some noise. For example, some users like to describe the experiences in other restaurants before commenting on the target restaurant. The results indicate that our system can partial reduce the effect of noisy information and achieve similar performance on both long and short texts.

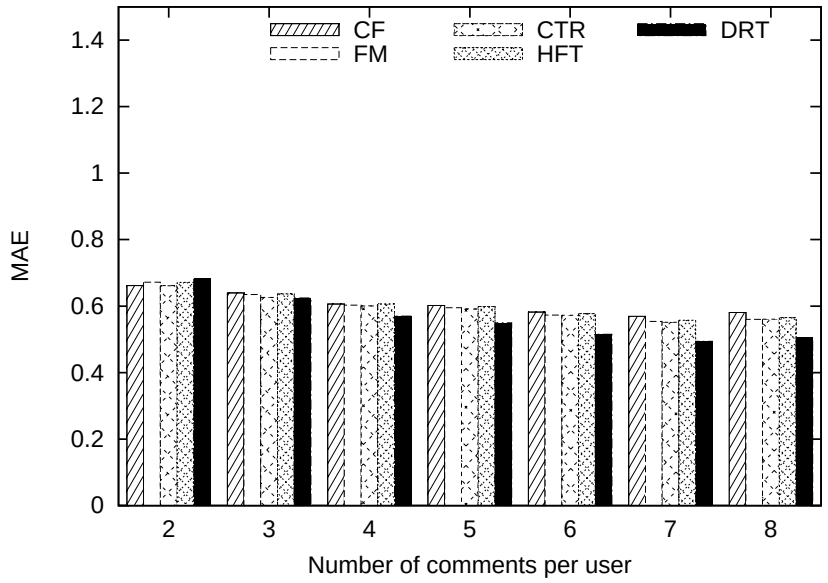


Figure 3: MAE vs number of training reviews per user (Dianping)

5 Related Work

In this paper, we utilize user comments to enhance recommendation systems. Text mining techniques are now widely used on comment data on the Internet to support a large variety of applications in e-commerce. Liu et al. (2005) discuss the possibility of comparing comments available online to identify important opinions of the users on the items. Zhai et al. (2011) present an approach to select important sentences in the product comments. Zhang et al. (2015) design a customized solution to restaurant comment summarization on dishes. Topic modeling techniques based on Dirichlet allocation are recently popular in text analytics (Blei et al., 2003), which is particularly effective on long documents to find meaningful topics (Blei et al., 2004). However, we hereby emphasize that all these techniques are not directly helpful to improve recommendation systems.

A huge bulk of recommendation algorithms are proposed in the literature. Collaborative filtering is known as the most popular approach (Su and Khoshgoftaar, 2009; Sarwar et al., 2001) and matrix factorization has recently emerged as an effective strategy to improve the effectiveness of collaborative filtering (Koren et al., 2009; Rendle, 2012). Latent Factorization Model decomposes the matrix of user-item features and identifies the features connected to users and items respectively. Some other invariants, such as max-margin matrix factorization (Rennie and Srebro, 2005) and probabilistic matrix factorization (Salakhutdinov and Mnih, 2008), are proposed to improve the robustness of the factorization outputs. Singh and Gordon (2008) further encode genres of movies and roles of actors in movies as binary relations into a collective matrix factorization model. All these approaches require the generation of matrix with aligned features, and only can use very limited information besides scores.

There are a handful of attempts to incorporate rich text information into recommendation engines. Musat et al. (2013) revise the original collaborative filtering approach, by modeling user similarity based on the feature words shown in their text comments. In their approach, the feature words are nouns with the highest opinion counts. Zhang et al. (2014) adopt a similar strategy to support personalized recommendation based on text comments. Their approach utilizes existing sentiment analysis tools (Lu et al., 2011) to generate the feature words. These words are fed into latent factor model (Koren et al., 2009) to build profiles on the users and items. However, the feature extraction procedure used in their method does not distinguish user preference and item features in the opinion. He et al. (2015) tries to build tripartite graph model with user, item and discussion topic engaged. The recommendation system exploits the tripartite graph model to make personalized recommendation decisions. In their approach, the feature words are identified similarly as Zhang et al. (2014). All the above studies

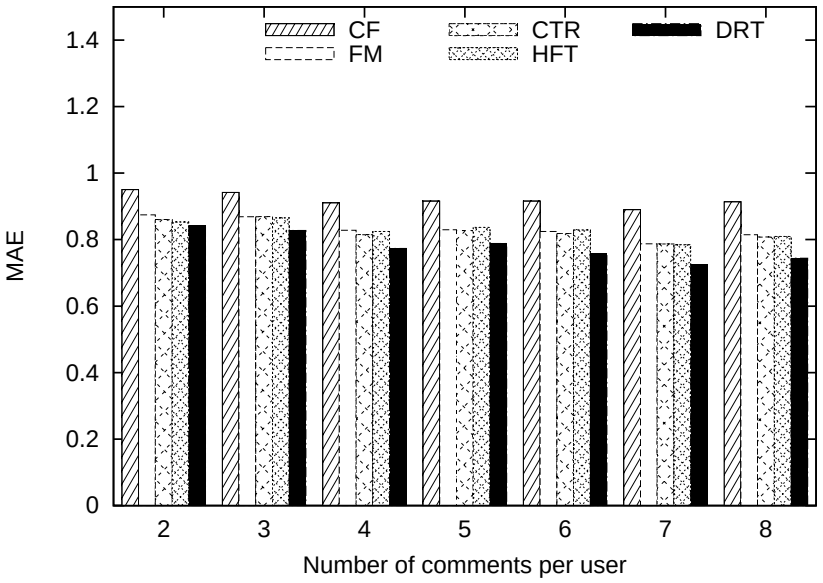


Figure 4: MAE vs number of training reviews per user (Amazon)

identify the feature words directly extracted from comments.

Recently some researchers try to learn latent aspects from comments (Wu and Ester, 2015). Wang et al. (2011) builds a regression model over the topic results from LDA. Their approach does not try to identify user profiles, and thus only applicable to score prediction on the text comments. Wang and Blei (2011) combine the merits of traditional collaborative filtering and topic modeling which provides a latent structure to recommend scientific articles in citation networks. Tang et al. (2015) learn representations of users and item for sentiment classification. Although similar concepts of user and item representations are used in (Tang et al., 2015), there are two major differences. Firstly, the problem we try to solve is completely different: they work on review sentiment classification, while our work focuses on score prediction without text. Secondly, the models are different: their approach includes user/product representation in CNN to enhance score prediction accuracy over text review, while out proposal attempts to maximize the likelihood of individual words. McAuley and Leskovec (2013) use a topic model to extract user and item profiles based on text comments. Although related, such approaches differ from ours in that they represent latent topics by keywords which are limited in representations. The approach of McAuley and Leskovec (2013) is the most relevant work to ours, but there are a number of limitations in their work. Firstly, they assume there is a one-to-one correspondence between the entries in the user profile and item profile. Secondly, their optimization relies on the assumption of LDA, such that the words are drawn independently from an unknown distribution. Our approach does not have any strong assumption on the generation mechanism.

6 Conclusion

In this paper, we have presented a simple yet effective approach to learn distributed representations of users and items from large amounts of text reviews. In our approach, the user vectors and item vectors are learned by a neural network. The user representation denotes the personal preferences over the items, while the item representation denotes the important attributes of the items. Finally, the distributed representations are used in recommendation systems. When tested on the datasets from different domains and languages, our systems achieve better performance than the state-of-the-art baseline systems.

Acknowledgments

Wenliang Chen, Zhenghua Li, and Min Zhang are supported by the National Natural Science Foundation of China (Grant No. 61572338, 61502325, and 61373095). This work is also partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization. Zhenjie Zhang

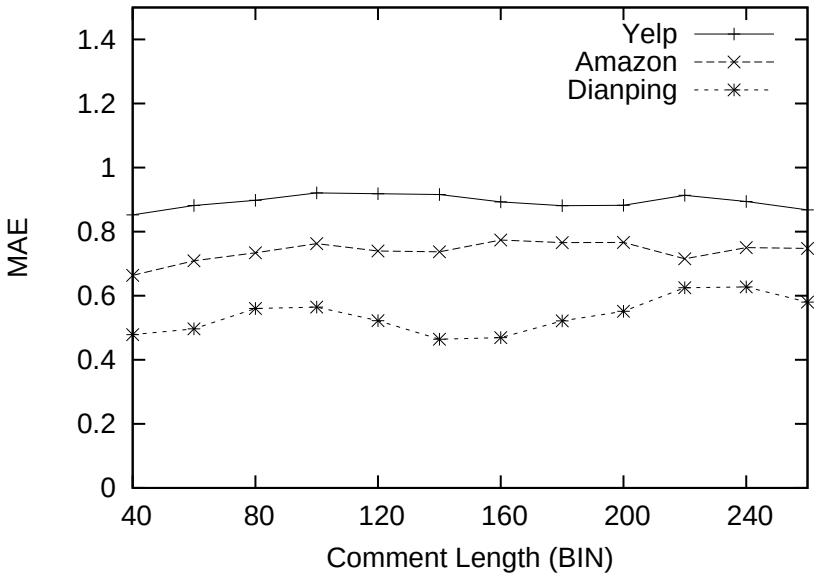


Figure 5: MAE vs average comment length (BIN) per user

is supported by the research grant for the Human Centered Cyber-physical Systems Programme at the Advanced Digital Sciences Center from Singapores A*STAR. We would also thank the anonymous reviewers for their detailed comments, which have helped us to improve the quality of this work.

References

Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. 2015. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 147–154. ACM.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

David M Blei, T Griffiths, M Jordan, and J Tenenbaum. 2004. Hierarchical topic models and the nested chinese restaurant process. *NIPS*, 16:106–114.

Michael Gutmann and Anpo Hyvarinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistica. *The Journal of Machine Learning Research*, 13:307–361.

Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 219–230. ACM.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196.

Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In *Mining text data*, pages 415–463. Springer.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. In *Proc of WWW*, pages 342–351. ACM.

Yue Lu, Malú Castellanos, Umeshwar Dayal, and ChengXiang Zhai. 2011. Automatic construction of a context-aware sentiment lexicon: an optimization approach. In *WWW*, pages 347–356.

Julian J. McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, pages 165–172.

Prem Melville, Raymond J. Mooney, and Ramadass Nagarajan. 2002. Content-boosted collaborative filtering for improved recommendations. In *AAAI*, pages 187–192.

Andriy Mhih and Hyee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of ICML*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Claudiu Cristian Musat, Yizhong Liang, and Boi Faltings. 2013. Recommendation using textual opinions. In *IJCAI*.

Steffen Rendle. 2012. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May.

Jason D. M. Rennie and Nathan Srebro. 2005. Fast maximum margin matrix factorization for collaborative prediction. In *ICML*, pages 713–719.

Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, pages 880–887.

Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW Conference*, pages 285–295.

Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *KDD*, pages 650–658. ACM.

Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1014–1023, Beijing, China, July. Association for Computational Linguistics.

Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *KDD*, pages 448–456. ACM.

Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In *KDD*, pages 618–626.

Yao Wu and Martin Ester. 2015. Flame: A probabilistic model combining aspect based opinion mining and collaborative filtering. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 199–208. ACM.

Zhongwu Zhai, Bing Liu, Lei Zhang, Hua Xu, and Peifa Jia. 2011. Identifying evaluative sentences in online discussions. In *AAAI*.

Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*, pages 83–92.

Rong Zhang, Zhenjie Zhang, Xiaofeng He, and Aoying Zhou. 2015. Dish comment summarization based on bilateral topic analysis. In *ICDE*, pages 483–494.

Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. 2011. Functional matrix factorizations for cold-start recommendation. In *SIGIR*, pages 315–324. ACM.

Improving Statistical Machine Translation with Selectional Preferences

Haiqing Tang, Deyi Xiong*, Min Zhang and Zhengxian Gong
Soochow University, Suzhou, China
hqtang@stu.suda.edu.cn
{dyxiong, minzhang, zhxgong}@suda.edu.cn

Abstract

Long-distance semantic dependencies are crucial for lexical choice in statistical machine translation. In this paper, we study semantic dependencies between verbs and their arguments by modeling selectional preferences in the context of machine translation. We incorporate preferences that verbs impose on subjects and objects into translation. In addition, bilingual selectional preferences between source-side verbs and target-side arguments are also investigated. Our experiments on Chinese-to-English translation tasks with large-scale training data demonstrate that statistical machine translation using verbal selectional preferences can achieve statistically significant improvements over a state-of-the-art baseline.

1 Introduction

Lexical translation error is one of the most urgent issues for statistical machine translation (SMT). Although phrase-based SMT can deal with local context dependencies well, it performs rather poorly with long-distance dependencies and therefore causes a lot of lexical translation errors. Verbs and their arguments form such long-distance dependencies and play important roles in translation as they build skeletons of sentences. However, many SMT systems are not sufficient to capture long-distance dependencies between arguments and their dominating verbs. Verbs and arguments are often either incorrectly translated or not translated at all according to the error study by Wu and Fung (2009a).

In order to address this issue, predicate-argument structures (PAS), which identify semantic frames within sentences by marking predicates, and labeling arguments with semantic roles, have been explored for SMT via various approaches in recent years. Wu and Fung (2009b) employ target-side PAS to pick out the most suitable translations among translation candidates after the decoding procedure is completed. Gildea (2010) integrates the PAS knowledge into decoding through projecting source-side PAS to the target-side via word alignments. In this paper, we are particularly interested in long-distance dependencies between verbs and their arguments in a predicate-argument structure. We propose to utilize selectional preferences (SPs) to handle these verb-argument dependencies for SMT.

Selectional preferences place semantic restrictions on words, with which words can co-occur in different syntactic patterns. To be more specific, the SPs of a verb can characterize the semantic restrictions that the verb imposes on its arguments. Violating these restrictions inevitably makes sentence senses odd or implausible. For example, in the sentence *"The ball drinks a potato."*, both subject and object preferences for the verb "drink" are violated. SPs have proven useful for numerous applications, e.g., semantic role labeling (Gildea and Jurafsky, 2002), pronoun resolution (Bergsma et al., 2008), textual inference (Pantel et al., 2007), word-sense disambiguation (Resnik, 1997) and many more. Therefore, we have sufficient theoretical foundation to believe that SPs between verbs and arguments can be used to alleviate translation errors that we pointed out above.

Our work consists of two parts: modeling SPs for verbs and incorporating SPs into an SMT system. In particular, we focus on the verb-object (v, obj) and verb-subject $(v, subj)$ selectional preference instances which can be extracted from our target-side corpus. SPs are computed in two ways: conditionally

*Corresponding author

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2154–2163, Osaka, Japan, December 11-17 2016.

probabilistic SPs and topic-based SPs. The former calculates conditional probabilities between verbs and arguments as the strengths of SPs in a traditional way. The latter builds a class-based SP model using topics as semantic classes of arguments. All these calculated SPs are monolingual SPs. Since we model SPs for translation, we are also interested in cross-lingual SPs, i.e., selectional preferences of source-side verbs over corresponding target-side arguments. Taking (v, obj) semantic restriction as an example, we want to extract source-side verbs v_s and target-side objects obj_t from our word-aligned bilingual corpus. With these semantic instances, we define a bilingual SP model to calculate cross-lingual SP strength that a source-side v_s impose on its target-side obj_t. We integrate SPs into a state-of-the-art phrase-based SMT system. Experiments on large-scale translation display that SPs can achieve an improvement of up to 0.83 BLEU points over our baseline.

To the best of our knowledge, this is the first attempt to successfully incorporate selectional preferences into SMT. Our contributions are as follows.

- We propose various models to incorporate target-side monolingual selectional preferences into SMT.
- We also present a model for cross-lingual selectional preferences.
- In order to address the unknown word issue in SP modeling, we further introduce a word embedding based similarity model.
- Finally, we conduct experiments and in-depth analysis to demonstrate how these SP models work for SMT.

The remainder of this paper is organized as follows. Section 2 introduces related studies about SPs induction and application. Section 3 elaborates our methods to learn verbal SPs from a large-scale corpus and three SP models for SMT. Section 4 discusses how to deal with unknown words in SPs. Section 5 describes how we integrate the verbal SPs into SMT. Section 6 reports the experimental results. In the last section, we conclude with future directions.

2 Related Work

Recent two decades have witnessed increasing efforts on automatic acquisition of SPs for verbs as well as wide applications of SPs in NLP tasks. Resnik (1996) is a pioneer on the induction of SPs from corpus, proposing a class-based approach named selectional association that uses WordNet synsets to provide conceptual classes for nouns co-occurring with a specific predicate in a particular relation. Li and Abe (1998) also rely on WordNet and use the principle of Minimum Description Length to find a suitable generalization level of a noun. But entirely relying on WordNet to generalize nouns to semantic classes has a fatal disadvantage because WordNet is lack of coverage of proper nouns. Therefore, Rooth et al. (1999) propose a probabilistic latent variable model using Expectation-Maximization (EM) clustering algorithm to induce class-based SPs. Erk (2007) investigates a similarity-based model which takes advantage of a corpus-based distributional similarity metrics between arguments for SPs. More recently, a number of researchers come up with methods modeling SPs via unsupervised topic models where topics express a set of latent classes for preferences with different grammatical relations. Séaghdha (2010) describes a model using latent Dirichlet allocation (LDA) (Blei et al., 2003) to compute SPs composed of a predicate and a single argument. In contrast, Ritter et al. (2010) study acquiring selectional preferences of a predicate and multiple arguments with topic models.

SPs are useful for numerous NLP tasks. Resnik (1997) uses automatically acquired SPs for word sense disambiguation. Zapirain et al. (2009) employ SPs to process semantic role classification in a large dataset. Many researchers apply SPs to conduct pseudo-disambiguation tasks (Van de Cruys, 2014; Erk, 2007) in order to evaluate the performance of their methods of acquiring SPs. In contrast to plenty of applications of SPs in monolingual tasks, rather few efforts are devoted to incorporate SPs into SMT. To the best of our knowledge, we are the first to model SPs in the context of SMT.

From the perspective of verb and argument translation, the most related work to ours is Xiong et al. (2012). They propose two translation models to incorporate source-side PAS into SMT. One is the predicate translation model exploring both lexical and semantic contexts to predict target-side predicates. The other is the argument reordering model which estimates the direction of target-side arguments movement relative to their predicates. The significant difference is that they separately model the translation of verbs and arguments while we model them in a unified fashion via SPs.

3 Selectional Preference Model

Most approaches represent SPs for verbs as a function $\sigma : (v, r, c) \rightarrow s$ that maps each verb v and the semantic class c of its argument with respect to role r to a real-valued selectional preference strength s (Light and Greiff, 2002). The higher the value of s is, the more arguments semantically fit their dominating verb. In this paper, we are interested in the degree to which an object or subject semantically fits a given verb. Additionally, we are wondering which semantic relation is more helpful for a phrase-based machine translation. We propose two approaches: a conditional probability-based method and a topic-based method to model verbal SPs. Due to the space limit, we only describe how we compute the SP strength of (v, obj). The strength of $(v, subj)$ can be calculated in a similar way.

3.1 Conditional Probability-Based SPs

The conditional probability-based method is the most primitive corpus-based way to capture SPs that a verb imposes on its arguments. The conditional probability can be computed as follows.

$$P(n|v, r) = \frac{f(v, r, n)}{f(v, r)} \tag{1}$$

where $f(v, r, n)$ represents the number of times that a noun n co-occurs with a verb v in a grammatical relation r. Considering r as a relation of a direct object of v, n is correspondingly specified as the headword of r. Thus we simplify formula (1) to calculate the SP strength between a verb and its object as follows.

$$P_c(obj|v) = \frac{f(v, obj)}{f(v)} \tag{2}$$

where obj is the headword of the object of v.

3.2 Topic-Based SPs

Topic-based SP is a typical of class-based SP that models how well a particular class of words fits a verb. We use latent topics that are learned from a collection of documents as our semantic classes. We choose the most widely used LDA (Blei et al., 2003) topic model to infer topics for our arguments. Each word in our corpus is assigned a topic. Then we compute the SP for a verb and its object headword as follows.

$$\begin{aligned} P_t(obj|v) &= \sum_{tp \in T} P(tp|v) P(obj|v, tp) \\ &\approx \sum_{tp \in T} P(tp|v) P(obj|tp) \end{aligned} \tag{3}$$

where T denotes the collection of topics that the current obj belongs to and tp stands for a topic assignment of the object. The first part $P(tp|v)$ can be calculated with relative counts that a verb co-occurs with objects that are assigned a topic tp. The second part $P(obj|tp)$ can be directly retrieved from the per-topic word distribution of topic tp over words computed by the LDA topic model.

3.3 Bilingual SPs

The two models introduced above are used to calculate SPs for verbs only on the target side. We also want to model cross-lingual SPs that source-side verbs impose on their corresponding target-side arguments.

Translation Category / Selectional Pairs	#0#	#1#	#2#	#3#
target-side (v, obj)	10.67%	27.46%	48.32%	13.54%
target-side $(v, subj)$	4.95%	34.45%	48.79%	11.80%

Table 1: Proportion of source-side verb-argument translation categories on the development set.

We therefore adapt the above two models to compute bilingual SPs. The conditionally probabilistic bilingual SP variant is calculated as follows.

$$P_{cbil}(obj_t|v_s) = \frac{f(v_s, obj_t)}{f(v_s)} \tag{4}$$

where obj_t is the target translation of obj_s. If obj is translated into a multi-word phrase, we use the first word of the phrase as obj_t.

For the bilingual topic-based SP model, we still use the LDA topic model to infer topics on the target language. We compute bilingual topic-based SPs via the following formula.

$$\begin{aligned} P_{tbil}(obj_t|v_s) &= \sum_{tp \in T} P(tp|v_s) P(obj_t|v_s, tp) \\ &\approx \sum_{tp \in T} P(tp|v_s) P(obj_t|tp) \end{aligned} \tag{5}$$

where T denotes the set of topics that the current obj_t belongs to and tp is the topic assigned to the object by LDA. $P(tp|v_s)$ is calculated with counts that the source-side verb v_s co-occurs with an object whose target-side counterpart is labeled with a topic tp. $P(obj_t|tp)$ is calculated by the LDA model.

4 SPs of Unseen Words

Conditional probability-based SPs cannot make any predictions for object headwords that have never occurred in our extracted selectional preference instances (v, obj). As our corpus cannot cover any phenomena in real world, a zero co-occurrence count between v and obj is not sufficient to show that the v has no selectional preference for that obj as its object. The method we employ to obtain source-side verb-argument pairs corresponding target-side verb-argument pairs during decoding has an obvious defect that word alignments directly affect the generation of translations. In this case, there may be many unseen word combinations. In order to investigate the proportion of unseen word combinations during decoding, we define four labels to classify these generated phrases. Label #0# represents phrases whose verb or object is translated into "#NULL#" due to incorrect word alignments. Phrases whose verbs or arguments are unseen in our trained SP models are labeled with #1#, #2# respectively. The remaining phrases appearing in our trained SP models are annotated with label #3#.

Table 1 shows the distribution of these phrases over the four categories on our development set (see details in Section 6.1). There are 53,268 (v, obj) pairs and 42,385 $(v, subj)$ pairs generated on the target side during decoding. Among these phrases, Only 13.54% (v, obj) pairs and 11.80% $(v, subj)$ pairs appear in our trained SP models. Most phrases, accounting for nearly 50%, are those whose argument headwords are unseen for our trained SP models. Hence, it is quite necessary to take some measure to model the SPs that verbs impose on their unseen argument headwords.

Instead of assigning a uniform value as the selectional strength for those unseen verb-argument combinations, we exploit a similarity-based model to compute SPs of a verb for an unseen argument headword during decoding, similar to the model by Erk (2007). Assuming (v, w_{un}) is a generated selectional preference instance according to word alignment information and w_{un} is an unseen object headword of v. The formulation to compute the selectional strength that v imposes on w_{un} is as follows.

$$P_c(w_{un}|v) = \sum_{w \in Seen(obj)} sim(w_{un}, w) \times wt_{obj}(w) \tag{6}$$

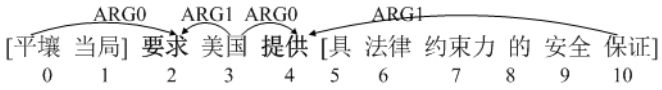


Figure 1: A source sentence with its predicate-argument structure. The verbs in the sentence are bold.

where $Seen(obj)$ is the set of seen headwords for an argument obj of a verb v, $sim(w_{un}, w)$ is the similarity between the seen and potential headword, and $wt_{obj}(w)$ is the weight of a seen headword w.

For the headword weight $wt_{obj}(w)$, we employ the selectional preference that the verb v imposes on the seen headword w to compute the value. $sim(w_{un}, w)$ is calculated with word2vec[1] and the similarity metric: Cosine. After each word on the target-side corpus is projected into a multidimensional vector space, $sim(w_{un}, w)$ is computed as follows.

$$Sim(\overrightarrow{w_{un}}, \overrightarrow{w}) = \frac{\overrightarrow{w_{un}} \bullet \overrightarrow{w}}{||\overrightarrow{sw_{un}}|| \times ||\overrightarrow{w}||} = \frac{\sum_i (a_i \times b_i)}{\sqrt{\sum_i {a_i}^2 \times \sum_i {b_i}^2}} \tag{7}$$

where a_i and b_i are the value of ith dimension of their word embeddings.

5 Decoding

In this section, we mainly elaborate how to integrate the proposed SP models into a phrase-based SMT system built on bracketing transduction grammars (BTG) (Wu, 1997). Before we introduce the integration algorithm for SP models, we define two functions F and G on a source sentence and its predicate-argument structure following Xiong et al. (2012). We use the sentence in Figure 1 as an example to make the two functions easier to be understood.

- $F(i, j)$: The function is used to find positions of all verbs and their object headwords pairs from the predicate-argument structure. These pairs are completely located within the source span (i, j). For example, in Figure 1, $F(0, 4)$=$\{$(2,3)$\}$, $F(0, 10)$=$\{$(2,3), (4,10)$\}$ while $F(0, 2)$=$\{\}$ because the object headword "美国" is located outside of the span $(0, 2)$ and $F(5, 10)$=$\{\}$ for the reason that the verb "提供" is located outside of the span $(5, 10)$.

- $G(i, k, j)$: The function finds positions of all verbs and their object headwords pairs that cross two neighboring spans (i, k) and $(k+1, j)$. It can also be formulated as $F(i, j) - (F(i, k) \cup F(k+1, j))$. In Figure 1, $G(0, 4, 10)$=$F(0, 10) - (F(0, 4) \cup F(5, 10))$=$\{$(4,10)$\}$.

In order to calculate SP strengths of target-side verbs and arguments as well as bilingual verb-argument pairs, we store word alignment information for each phrase pair in the phrase table. Given a source sentence with its predicate-argument structure, if a BTG lexical rule is applied to translate a source phrase c spanning (i, j) to a target phrase e, we use $F(i, j)$ to detect all verb-object pairs and build a translation set $A(i, j) = \{(v_t, obj_t), (\cdots), \cdots\}$ to store corresponding verb-object translations on the target side through word alignments. Since our decoder is a log-linear model which is easy to incorporate new features, we define another function P_r to calculate the score of SPs as a new feature over span (i, j) as follows.

$$P_r(A(i, j)) = \prod_{(v_t, obj_t) \in A(i,j)} P_{\cdot}(obj_t | v_t) \tag{8}$$

where $P_{\cdot}(obj_t|v_t)$ can be the conditionally probabilistic model P_c or topic-based model P_t. For the bilingual SP models, we only need to change v_t to its source-side counterpart v_s.

If a BTG merging rule is applied to combine its two sub-spans (i, k), $(k + 1, j)$ in a straight $((i, k) + (k + 1, j) \rightarrow (i, j))$ or inverted order $((k + 1, j) + (i, k) \rightarrow (i, j))$, we directly use $P_r(A(i, k))$ and $P_r(A(k + 1, j))$ that have been already computed for the two sub-spans (i, k) and $(k + 1, j)$ in

[1] https://code.google.com/archive/p/word2vec/

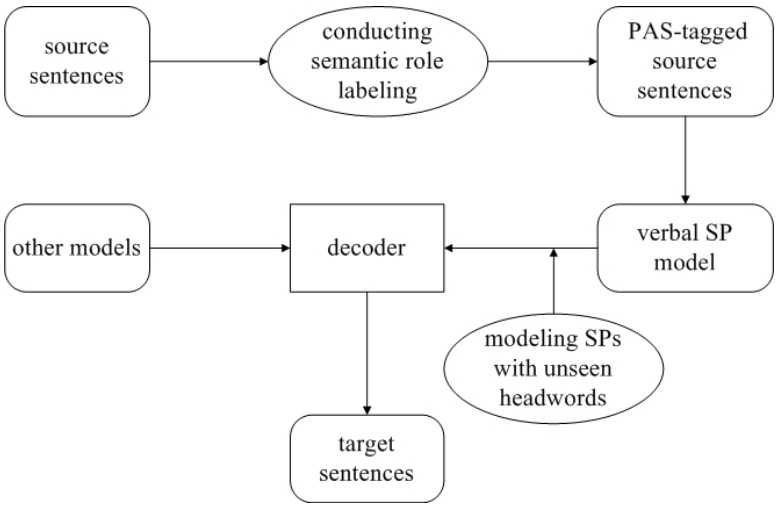


Figure 2: Architecture of SMT system equipped with verbal SPs.

the dynamic programming decoding algorithm. In this way, we only need to set another translation set $B(i,j) = \{(v_t, obj_t), (\cdots), \cdots\}$ to store the translations of source-side verb-object pairs found by $G(i,k,j)$ according to word alignments and calculate $P_r(B(i,j))$ for verb-object pairs that cross the two sub-spans.

In order to expedite the decoding process, we compute corresponding SPs for each (v, obj) semantic pairs extracted from the training corpus before decoding and load them when decoding instead of computing them on the fly. As for unseen object headword w_{un} of a verb, we use Eq. (6) to model SPs when integrating conditional probability-based SP model and Eq. (3) to model SPs when integrating topic-based SP model. We store the selectional strength of (v, w_{un}) for each unknown word so as to avoid repetitive computation. Figure 2 shows the architecture of the SMT system equipped with verbal SPs translation model. Since the system we used is based on a CKY-style decoder, the integration algorithm introduced here can be easily adapted to other CKY-based decoding systems such as the hierarchical phrasal system (Chiang, 2007).

6 Experiments

In order to validate the effectiveness of our SMT system enhanced with SPs, we perform a series of experiments on Chinese-to-English translation, which are trained with massive data. Specially, we aim at investigating:

- Whether integrating SPs into SMT can improve the system translation accuracy.
- Which can achieve better performance, conditionally probabilistic SP model or topic-based SP model?
- Whether semantic similarity-based approach is more reasonable than assigning a uniform value as the selectional strength that a verb imposes on its unseen argument headwords.
- Whether bilingual SPs are more effective than monolingual SPs for SMT.

6.1 Setup

The baseline is a state-of-the-art BTG-based phrasal system (Xiong et al., 2006). Our training data corpora[2] consist of 2.9M sentence pairs with 80.9M Chinese words and 86.4M English words. We ran GIZA++ on these corpora in both directions and then applied the "grow-diag-final" refinement rule to obtain final word alignments. Then we used all these word-aligned corpora to generate our phrase table.

[2]The corpora include LDC2003E14, LDC2004T07, LDC2005T06, LDC2005T10 and LDC2004T08 (Hong Kong Hansards/Laws/News).

Model	NIST04	NIST05
Base	36.40	33.69
Base+$P_c(obj_t\|v_t)$	36.93*	34.22**
Base+$P_c(obj_t\|v_t)$+$P_c(obj_{t_{un}}\|v_t)$	37.09*	34.43**
Base+$P_c(sub_t\|v_t)$	36.89	34.19*
Base+$P_c(sub_t\|v_t)$+$P_c(sub_{t_{un}}\|v_t)$	36.99*	34.37**
Base+$P_{cbil}(obj_t\|v_s)$	37.15**	34.21**

Table 2: Results of conditionally probabilistic SPs with two selectional relations: (v, obj) and (v, sub). **/*: significantly better than the baseline at $p < 0.01$ and $p < 0.05$ respectively.

Our 4-gram language model was trained on the Xinhua section of the English Gigaword corpus using the SRILM toolkit with modified Kneser-Ney smoothing.

In order to automatically learn SPs for verbs, we first parsed all source sentences using Stanford Parser and then ran the Chinese semantic role labeler (Li et al., 2010) on all source parse trees to annotate semantic roles for all verbs. At the same time, we ran SENNA on the target side to not only parse all target sentences but also conduct semantic role labeling for all verbs. It is easy to extract (v_t, obj_t) pairs or (v_t, sub_t) pairs after we obtained semantic roles on both sides. As for extracting (v_s, obj_t) selectional tuples, we first extracted (v_s, obj_s) pairs from source sentences with PAS and then used word alignments to get the target-side translation obj_t of obj_s. We used GibbsLDA++ to infer topics for our topic-based SP models. We set the number of topics from 50 to 350 with an incremental interval 50. We found the best number of topics according to results on our development set.

We trained word embeddings with word2vec using continuous bag-of-words model (Mikolov et al., 2013). The word vector dimensionality was set to 200 and we set the value of threshold for occurrence of words to 0.00001. Values of other parameters such as the training algorithm and the size of the window were all set by default.

We adopted the NIST MT03 evaluation test data as our development set, and the NIST MT04, MT05 as the test sets. We used the case-insensitive BLEU-4 (Papineni et al., 2002) to evaluate translation quality and run MERT (Och, 2003) three times. We finally recorded average BLEU scores over the three runs for all our experiments and used MultEval toolkit[3] to perform the significance test.

6.2 Results

Our first group of experiments is to investigate whether a simple conditional probability method for modeling SPs is able to improve translation accuracy in terms of BLEU. Moreover, we also would like to know whether the similarity-based SP model for unseen argument headwords will achieve further improvements. Experimental results are shown in Table 1. From the experiments which are conducted only using monolingual SPs, we can find that the verb-object SP model $P_c(obj_t|v_t)$ performs slightly better than $P_c(sub_t|v_t)$ on both test sets. Using semantic similarity metric rather than a uniform value to evaluate the selectional preference of a verb for its unseen argument can achieve better performance. It can also be observed that bilingual SPs marginally outperform than monolingual SPs on average. All SP models in this table are statistically better than the baseline on the test set MT05.

Our second group of experiments is to validate whether the topic-based SPs are more effective than conditionally probabilistic SPs in improving the accuracy of lexical choice. Table 2 shows our results. First, we have observations similar to what we have found in Table 1: verb-object SPs are better than verb-subject SPs while cross-lingual SPs better than monlingual SPs. Second, comparing Table 2 against Table 1, we find that topic-based SPs are better than conditional probabilistic SPs with a uniform value for unseen headwords, but similar to that with a similarity-based SP model for unseen headwords.

Analysis on translations reveals that our SP models are helpful for reducing verb-argument translation errors. Due to the space limit, we only show two translation examples. Figure 3 displays a translation example which shows that the system equipped with verbal SP model can solve the problem that the

[3]https://github.com/jhclark/multeval

Model	NIST04	NIST05
Base	36.40	33.69
Base+$P_t(obj_t\|v_t)$	37.11*	34.36**
Base+$P_t(sub_t\|v_t)$	37.07*	34.30**
Base+$P_{tbil}(obj_t\|v_s)$	37.23**	34.35**

Table 3: Results of topic-based SPs with two relations: (v, obj) and (v, sub). **/*: significantly better than the baseline at $p < 0.01$ and $p < 0.05$ respectively.

Figure 3: A translation example shows that verbal SPs can help SMT system alleviate the translation error that verb is not translated at all. The verbs in the sentence are bold.

baseline is unable to translate each verb in the source sentence to a target string. From the example, we can easily find that the baseline cannot correctly translate a $(verb, obj)$ selectional tuple like (参与，任务) where only obj "任务" is translated. Instead, in the system enhanced with verbal SPs, in addition to the object, $verb$ "参与" is also correctly translated into a target string. Figure 4 shows another example to demonstrate that verbal SPs are useful for selecting the proper translation for an object. The source word "发放" is not translated at all in the baseline while it is translated into "release" by our SP model.

7 Conclusion

We have presented three different models to compute SPs on verb-object and verb-subject pairs and successfully integrate them into a phrase-based SMT system. From a series of experiments on Chinese-to-English translation, we have found:

- Verbal SPs can significantly improve SMT in alleviating translation errors of verbs and their arguments.
- Verb-subject SPs perform similarly to verb-object SPs but slightly worse.

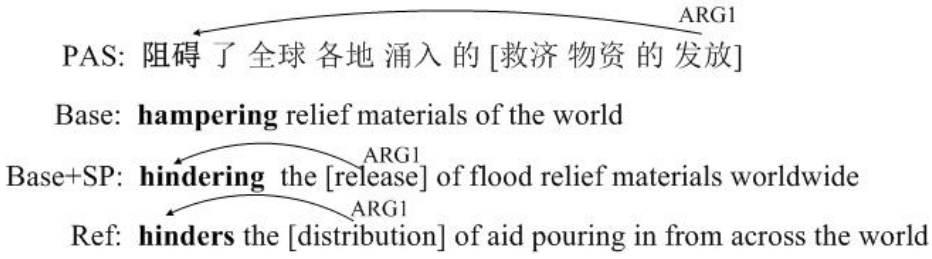


Figure 4: A translation example shows that verbal SPs can help SMT system alleviate the translation error that the argument of a verb is not translated at all. The verbs in the sentence are bold.

- Similarity-based SPs is helpful for conditional probability-based SP model to evaluate the SPs of a verb's unseen argument headword.
- Topic-based SPs are better than conditionally probabilistic SPs and bilingual SPs marginally better than monolingual SPs.

In the future, we would like to acquire bilingual SPs using a neural network approach (Van de Cruys, 2014). We also want to model SPs for verbs that are unseen in the training corpora and to explore a unified method to obtain SPs that a verb impose on its subject and object at the same time.

Acknowledgements

The authors were supported by National Natural Science Foundation of China (Grant Nos. 61403269, 61432013, 61525205, 61273319 and 61305088) and Natural Science Foundation of Jiangsu Province (Grant No. BK20140355). We also thank the anonymous reviewers for their insightful comments.

References

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

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of Machine Learning Research*, 3:993–1022.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Katrin Erk. 2007. A simple, similarity-based model for selectional preferences. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, volume 45, page 216.

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

Daniel Gildea. 2010. Semantic role features for machine translation. In *International Conference on Computational Linguistics*, pages 716–724.

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

Junhui Li, Guodong Zhou, and Hwee Tou Ng. 2010. Joint syntactic and semantic parsing of chinese. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1108–1117. Association for Computational Linguistics.

Marc Light and Warren Greiff. 2002. Statistical models for the induction and use of selectional preferences. *Cognitive Science A Multidisciplinary Journal*, 26(3):269–281.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard H Hovy. 2007. Isp: Learning inferential selectional preferences. In *HLT-NAACL*, pages 564–571.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Philip Resnik. 1996. Selectional constraints: An information-theoretic model and its computational realization. *Cognition*, 61(1):127–159.

Philip Resnik. 1997. Selectional preference and sense disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How*, pages 52–57. Washington, DC.

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

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

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

Tim Van de Cruys. 2014. A neural network approach to selectional preference acquisition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 26–35.

Dekai Wu and Pascale Fung. 2009a. Can semantic role labeling improve smt? pages 218–225.

Dekai Wu and Pascale Fung. 2009b. Semantic roles for smt: a hybrid two-pass model. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 13–16.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 521–528. Association for Computational Linguistics.

Deyi Xiong, Min Zhang, and Haizhou Li. 2012. Modeling the translation of predicate-argument structure for smt. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 902–911. Association for Computational Linguistics.

Beñat Zapirain, Eneko Agirre, and Lluís Màrquez. 2009. Generalizing over lexical features: Selectional preferences for semantic role classification. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 73–76. Association for Computational Linguistics.

Hierarchical Permutation Complexity for Word Order Evaluation

Miloš Stanojević **Khalil Sima'an**
Institute for Logic, Language and Computation (ILLC)
University of Amsterdam
{initial.last}@uva.nl

Abstract

Existing approaches for evaluating word order in machine translation work with metrics computed *directly* over a *permutation* of word positions in system output relative to a reference translation. However, every permutation factorizes into a permutation tree (PET) built of *primal permutations*, i.e., atomic units that do not factorize any further. In this paper we explore the idea that permutations factorizing into (on average) *shorter primal permutations* should represent *simpler ordering* as well. Consequently, we contribute *Permutation Complexity*, a class of metrics over PETs and their extension to forests, and define *tight metrics*, a sub-class of metrics implementing this idea. Subsequently we define example tight metrics and empirically test them in word order evaluation. Experiments on the WMT13 data sets for ten language pairs show that a tight metric is more often than not better than the baselines.

1 Introduction

MT evaluation involves at least two factors, word order (syntactic) and adequacy (semantic). Conceivably, MT system developers could use diagnostic tools based on metrics dedicated to each factor separately. Word order metrics are frequently used to evaluate pre-ordering components, e.g., (Herrmann et al., 2011; Bisazza and Federico, 2013), or for analyzing specific reordering phenomena, e.g., (Bisazza and Federico, 2013; Xiang et al., 2011; Braune et al., 2012). Other uses include, ordering component tuning, e.g., (Gao et al., 2011; Neubig et al., 2012; DeNero and Uszkoreit, 2011; Katz-Brown et al., 2011; Hall et al., 2011), measuring divergence between languages (Birch et al., 2008), and matching gene sequences in bioinformatics (Eres et al., 2004).

For evaluating word order, a permutation is induced between a system output and the corresponding reference translation. Existing work uses metrics over permutations such as Kendall's tau (Lapata, 2006; Birch and Osborne, 2011), Spearman (Isozaki et al., 2010), Hamming, Ulam (Birch et al., 2010) and Fuzzy Score (Talbot et al., 2011). Approximately, Kendall's tau, Spearman and Hamming measure correct individual position or correct relative pairs, whereas Ulam and Fuzzy Score measure monotone units (contiguous or not).

A word order metric measures how similar a permutation is to the monotone (or identity) permutation. Here we advocate the idea that a suitable metric must *also* assign similar values to similar permutations. Crucially, factorizing a permutation into a Permutation Tree (PET) reveals its atomic building blocks, called *primal* permutations (Albert and Atkinson, 2005; Gildea et al., 2006). In this view, permutations that factorize into similar PETs should be similar. Some previous work (Stanojević and Sima'an, 2014a; Stanojević and Sima'an, 2014b) has used PETs for evaluation, but without attempting to explain the effect of factorization. Next we motivate the idea that, all other things being equal, *the more factorizable a permutation the simpler it is* in terms of ordering.

Informally, a PET for permutation π is a tree where the nodes are labeled with *operators* (Figure 1). The fringe of every subtree in a PET is a *sub-permutation* of π, i.e., a contiguous sub-sequence isomorphic with a permutation.[1] Consider $\pi_a = \langle 6, 1, 4, 2, 3, 5 \rangle$ and $\pi_b = \langle 6, 1, 5, 2, 3, 4 \rangle$. Their PETs (two

[1] Akin to a phrase pair in MT.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2164–2173, Osaka, Japan, December 11-17 2016.

left-most in Figure 1) are built from monotone $\langle 1,2 \rangle$ or inverted $\langle 2,1 \rangle$ operators only. Two local inversions $\langle 2,1 \rangle$ could turn each of π_a and π_b into monotone. Permutation $\pi_c = \langle 2,4,5,6,1,3 \rangle$ (right-most Figure 1) demands $\langle 2,4,1,3 \rangle$ at the root to bring it to monotone. In contrast, $\pi_d = \langle 6,2,4,1,5,3 \rangle$ does not yield to factorization because it does not properly contain sub-permutations; Non-factorizable permutations are called *primal permutations*,[2] and they constitute the *atomic building blocks for all permutations* (Albert and Atkinson, 2005) – see Section 3. Hence, π_d demands itself to convert it into monotone. In this view, π_a and π_b signify potentially simpler ordering than π_c, which is simpler than π_d.

Conveniently, PETs show two aspects of permutations: *recursive grouping and primal building blocks*. In this paper we introduce *Permutation Complexity*, a class of metrics over PETs, exploiting hitherto untapped discerning properties of permutations. **A.** Similarity: different permutations often share primal permutations. **B.** Factorizability: some permutations factorize into shorter primal permutations but others do not. **C.** Hierarchy: factorizing permutations exposes their hierarchical grouping. Practically speaking, metrics over PETs should be attractive because they parameterize in terms of primal permutations and bracketing structure.

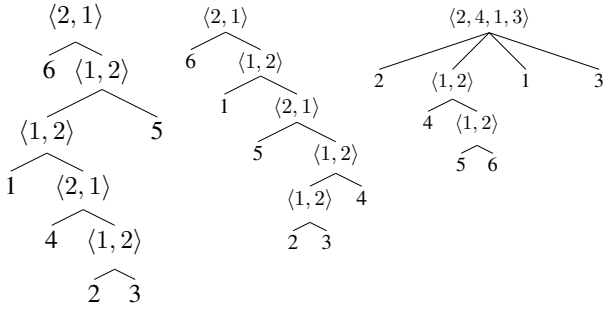


Figure 1: Three permutations and their PETs

From our Permutation Complexity viewpoint we see PET factorization as *compression* using a *code book* of primal permutations. Consequently, we introduce *tight* metrics, a sub-class that assigns a smaller complexity to a PET than to any less factorized structure of the same permutation, with the intuition that more factorization should reveal simpler building blocks. In this paper, we contribute: (1) Foundational formalization of tight complexity metrics over permutations, (2) An extension of PETs (Gildea et al., 2006) to forests to capture the potential relevance of brackettings for evaluation, (3) Novel metrics for reordering evaluation, and (4) Experiments on system ranking in MT. Our experiments show that the new tight (and semi-tight) metrics perform competitively over a range of language pairs, which provides the first evidence for a complexity-based factor in evaluation.

2 Existing metrics (Baselines)

We define evaluation metrics in the range $[0,1]$ with the interpretation *the higher the score the better*. Whilst this is natural for MT evaluation, for a formal treatment of complexity, as in Section 4, it is natural that complexity is interpreted as "the higher the more complex". The two notions are easily converted to each other after normalization.

A permutation π over $[1..n]$ (subrange of the positive integers) is a bijective function from $[1..n]$ to itself. To represent permutations we will use angle brackets as in $\langle 2,4,3,1 \rangle$. Given a permutation π over $[1..n]$, the notation π_i $(1 \leq i \leq n)$ stands for the integer in the i^{th} position in π; $\pi(i)$ stands for the index of the position in π where integer i appears; and π_i^j stands for the (contiguous) sub-sequence of integers $\pi_i, \ldots \pi_j$. The *length* of π is simply $|\pi| = n$.

$$\textsc{Kendall}(\pi) = \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\delta[\pi(i)<\pi(j)]}{(n^2-n)/2}$$

$$\textsc{Hamming}(\pi) = \frac{\sum_{i=1}^{n}\delta[\pi_i=i]}{n}$$

$$\textsc{Spearman}(\pi) = 1 - \frac{3\sum_{i=1}^{n}(\pi_i-i)^2}{n(n^2-1)}$$

$$\textsc{Ulam}(\pi) = \frac{LCS(\pi,\textsc{Id}_1^n)-1}{n-1}$$

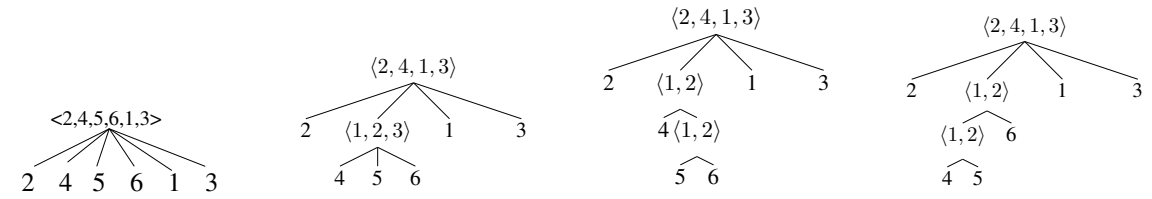
$$\textsc{Fuzzy}(\pi) = 1 - \frac{c-1}{n-1}$$

where c is # of monotone sub-permutations

Figure 2: Common metrics over permutations

The baselines are the existing metrics over permutations, including KENDALL's tau, HAMMING and ULAM used in (Birch and Osborne, 2010; Birch and Osborne, 2011; Birch et al., 2010; Isozaki et al., 2010); SPEARMAN rho used in (Isozaki et al., 2010); and FUZZY *Reordering Score* used in (Talbot et al., 2011), which is a reordering measure extracted from

[2] Also known as simple or non-decomposable (Brignall, 2010) – note the analogy with prime numbers.



(a) Original permutation (b) Factorization step (c) PET (d) Another PET

Figure 3: Permutation factorization leading to PETs

METEOR (Denkowski and Lavie, 2011). Figure 2 lists the definitions of these metrics. In these definitions, LCS stands for Longest Common Subsequence, Kronecker $\delta[a]$ which is 1 if $(a = true)$ else zero, and $\text{ID}_1^n = \langle 1, \cdots, n \rangle$ which is the identity permutation over $[1..n]$. Next we present an alternative view of permutations.

3 Factorization and order complexity

In factorization we seek to decompose a permutation to reveal a tree of its atomic ordering patterns. Figure 3 shows the factorization process applied to $\pi = \langle 2, 4, 5, 6, 1, 3 \rangle$. It starts out by representing π as a tree with root decorated with π itself (Figure 3a). In every step we seek the *minimal number* of adjacent *sub-permutations*. For $\langle 2, 4, 5, 6, 1, 3 \rangle$ this minimal number is four, namely $\{2\}$, $\{4, 5, 6\}$, $\{1\}$ and $\{3\}$. The first step leads to Figure 3b, where the sub-permutations are represented as subtrees with roots decorated with operators (permutations) over their child nodes. Applying factorization recursively to $\langle 4, 5, 6 \rangle$ leads to choices in binarization because both $\{4, 5\}$ and $\{5, 6\}$ are sub-permutations (Figures 3c and 3d). Next we summarize the formal results underlying factorization.

Primal permutations[3] are permutations that do not properly contain sub-permutations. Example common primal permutations are $\langle 1, 2 \rangle$, $\langle 2, 1 \rangle$ and $\langle 2, 4, 1, 3 \rangle$. Primal permutations signify the *atomic reorderings*. The following result shows they are also the *building blocks* of all permutations.

Factorization (Albert and Atkinson, 2005) Every permutation π can be written[4] as $\sigma[\pi_1, \cdots, \pi_m]$, where σ is *primal and unique*, and each π_j is a sub-permutation of π. If $m \geq 4$ then $\pi_1, \cdots, \pi_m$ are *unique*.

The uniqueness of σ and $\pi_1, \cdots, \pi_m$ for $m \geq 4$ is crucial for efficiency (Section 5). We call m the **arity** of π, written $\mathbf{a}(\pi)$ (or simply $\mathbf{a}$). For example, $\langle 4, 2, 3, 1 \rangle$ has arity 2: $\sigma = \langle 2, 1 \rangle$, $\pi_1 = \langle 4, 2, 3 \rangle$ and $\pi_2 = \langle 1 \rangle$. Applying Albert & Atkinson's result recursively factorizes π into PETs, see (Gildea et al., 2006) and Section 5 for efficient algorithms.

Permutation complexity is the class of metrics over PETs. This class includes *ground metrics* over primal permutations (operators), and higher-order metrics over PETs for other factorizable permutations. In a trivial sense, useful here, the existing baseline metrics can be seen as operating over weaker kinds of factorization which leave (parts of) π unfactorized.

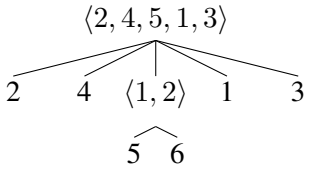


Figure 4: Weak factorization

Weak factorization A permutation π is a *weak factorization (WF)* of itself, represented as a single node with operator equivalent to π. The process applies recursively factorizing an operator in a given weak factorization τ into any number of sub-permutations (not necessarily minimal). Weak factorization may terminate at any point.

Intuitively, we would like permutation complexity metrics to be sensitive to factorization into primal permutations. This can be achieved by imposing a partial order over the different weak factorizations

[3] Also known as simple or non-decomposable permutations.

[4] The notation $\sigma[\pi_1 \cdots \pi_m]$ stands for a sequence of sub-permutations $\pi_1 \cdots \pi_m$ which is permuted by σ.

of the same permutation, assigning minimal complexity to PET factorizations. Next we formalize the notion of *tight metrics* implementing this intuition.

4 Permutation complexity: Tight metrics

A complexity metric $C(\cdot)$ is a function from weak factorizations to non-negative reals.

Tight/Semi-tight metrics A complexity metric $C(\cdot)$ is **tight** for a non-primal permutation π iff for every two weak factorizations $\tau_x \neq \tau_y$ of π holds: if τ_x factorizes into τ_y then $C(\tau_x) > C(\tau_y)$. A **semi-tight** metric fulfills the weaker requirement $C(\tau_x) \geq C(\tau_y)$ for all cases except when τ_x is the flattest weak factorization (single node) and $C(\tau_y)$ is a PET, where it strictly requires $C(\tau_x) > C(\tau_y)$.

A metric $C(\cdot)$ is tight iff it is tight for all π. It is semi-tight if it is semi-tight for at least one π and tight otherwise.

Let wf be a weak factorization and let $\mathcal{O}_{wf}$ be the multi-set of (non-leaf) node operators (permutations). We now narrow our attention to functions $F(\cdot)$ over the multi-set $\mathcal{O}_{wf}$, i.e., $C(wf) = F(\mathcal{O}_{wf})$. This means that we are disregarding the bracketing structure of wf. In Section 5 we incorporate bracketings by extending this framework to forests.

What should metric $F(\mathcal{O}_{wf})$ fulfill to be tight? We will parameterize $F(\cdot)$ with a ground metric $C_o(\cdot)$ over node operators, i.e., $F_{C_o}(\cdot)$. The idea here is that higher-order metrics over PETs can better delegate local operator complexity to a dedicated ground metric defined directly over operators, particularly primal permutations.

An operator complexity function $C_o(op)$ monotone non-decreasing[5] in operator length $|op|$ implements the idea that longer primal permutations are more complex,[6] cf. (Brignall, 2010) Theorem 2.2.:

> *Every primal permutation of length $n \geq 2$ contains another primal permutation of length $(n-1)$ or $(n-2)$ (Schmerl and Trotter, 1993).*

For example, $\langle 2,4,1,5,3\rangle$ contains $\langle 2,4,1,3\rangle$, and the latter contains $\langle 2,1\rangle$.[7]

Now we look at functions $F_{C_o}(\cdot)$ that are *monotone increasing* in the arithmetic average of $C_o(\cdot)$: $\textsc{Avg}_{C_o}(wf) = \frac{1}{|\mathcal{O}_{wf}|} \times \sum_{op \in \mathcal{O}_{wf}} C_o(op)$. For semi-tightness $\textsc{Max}_{C_o}(wf)$ is suitable. The following theorem says that a metric is tight if it assigns lower complexity to a permutation factorizing into a PET with shorter average primal permutation length.

Theorem 1 A metric $C(\cdot)$ is tight (semi-tight) if for some $C_o(\cdot)$, monotone non-decreasing in *operator length*, metric $C(\cdot)$ is monotone increasing (respectively non-decreasing) in $\textsc{Avg}_{C_o}(\cdot)$.

Proof Assume τ_0 factorizes to τ_j in a number of steps $j \geq 1$. In every step τ_{i-1} to τ_i, one operator op in τ_{i-1} of length $|op| > 2$ factorizes into $\sigma[op_1, \cdots, op_m]$, where $m \geq 2$. By the nature of factorization, $|\sigma| = m$ and $\sum_{i=1}^{m} |op_i| = |op|$. Let $\mathcal{O}^-$ stand for multi-set $\mathcal{O}$ excluding op. The average length of operators in τ_i is $\frac{1}{|\mathcal{O}_{\tau_i}|} \times (\sum_{p \in \mathcal{O}_{\tau_i}} |p|)$; it can be rewritten into $\frac{1}{m+|\mathcal{O}_{\tau_{i-1}}|} \times (m + |op| + \sum_{p \in \mathcal{O}^-_{\tau_{i-1}}} |p|)$ and again into $\frac{1}{m+|\mathcal{O}_{\tau_{i-1}}|} \times (m + \sum_{p \in \mathcal{O}_{\tau_{i-1}}} |p|)$. The desired inequality $\frac{m+\sum_{p \in \mathcal{O}_{\tau_{i-1}}} |p|}{m+|\mathcal{O}_{\tau_{i-1}}|} < \frac{\sum_{p \in \mathcal{O}_{\tau_{i-1}}} |p|}{|\mathcal{O}_{\tau_{i-1}}|}$ holds under the condition that $\sum_{p \in \tau_{i-1}} |p| > |\mathcal{O}_{\tau_{i-1}}|$, i.e., the average length of operators in a weak factorization is greater than the number of non-leaf nodes. The latter is a tautology because the branching factor of any node is always two or more. Tightness (semi-tightness) follows if $C(\cdot)$ is monotone increasing (respectively monotone non-decreasing) in $\textsc{Avg}_{C_o}(\cdot)$ when C_o is monotone non-decreasing in operator length. □

In other words, a metric assigning lower complexity to more factorizable permutations (by average operator length) is tight, i.e., *it allows comparing permutations by the smallest complexity assigned to them within this framework.* In Figure 3, a tight metric $C(\cdot)$ assigns $C(3a) > C(3b) > C(x)$ for

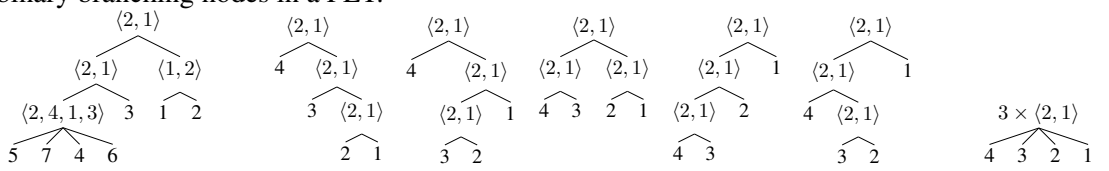
[5] $C_o(op)$ could be *monotone increasing*, but the weaker requirement is sufficient. Practically, we could parameterize $C_o(op)$ in operator-clusters and train it on data.

[6] A further practical requirement is $C_o(op) = 0$ iff $op = \langle 1,2\rangle$. But this is not necessary for tightness.

[7] By definition a primal permutation cannot be a sub-permutation of another primal permutation.

$x \in \{3c, 3d\}$. A semi-tight metric assigns complexity scores such that either $C(3a) \geq C(3b) > C(x)$ or $C(3a) > C(3b) \geq C(x)$ for $x \in \{3c, 3d\}$.

An example tight metric is the number of nodes in a PET; the more nodes the shorter the average length of primal operators. Beside maximum operator length, another semi-tight metric is the number of non-binary branching nodes in a PET.



(a) PET $\langle 5, 7, 4, 6, 3, 1, 2\rangle$. (b) Five different PETs for $\pi = \langle 4, 3, 2, 1\rangle$. (c) Flattened PET

Figure 5: In 5a and 5b PETs for different permutations. In 5c a flattened PET for the five in 5b.

5 From permutation trees to forests

For many computational purposes, a single *canonical PET* is sufficient, cf. (Gildea et al., 2006). A single PET can be computed in linear-time, cf. (Uno and Yagiura, 2000; Zhang and Gildea, 2007). Crucial for efficiency is the *uniqueness* of the sub-permutations for factorizations of arity $\mathbf{a} \geq 4$ (see Albert and Atkinson's result – Section 3), i.e., there is a single choice for a set of *split points* between adjacent sub-permutations. For arity $\mathbf{a} = 2$, there are at most $(n-1)$ choices for a single split point, if $|\pi| = n$. This is also crucial for our Permutation Forest algorithm defined next.

Some permutations factorize into multiple alternative PETs (see Figure 5b). The alternative PETs of π can be packed into an $O(n^2)$ permutation forest (PEF).

Flattened PET In a PET, chains of binary operators, either all inverted or all monotone, can be flattened leading to a *special kind* of packed representation. For example, the monotone operators in the PETs in Figures 3c and 3d can be flattened into the representation in Figure 3b. Therefore, we distinguish this from regular factorization by writing the encapsulated chain explicitly as in $3 \times \langle 2, 1\rangle$ in Figure 5c. This notation means that all binarizations of this order are allowed under that node. Various metrics defined in the next section are computed without the need to unpack this representation (e.g., the number of possible binarizations), which leads to algorithms over PEFs in $O(n)$. In the sequel we refer to function FLAT(PET) which "flattens" PET in this fashion.

A **permutation forest** (akin to a parse forest) $\mathcal{F}$ for π (over $[1..n]$) is a data structure consisting of a subset of $\{[[i, j, \mathcal{I}_i^j]] \mid 0 \leq i \leq j \leq n\}$, where $\mathcal{I}_i^j$ is a (possibly empty) set of *inferences* for π_{i+1}^j. If π_{i+1}^j is a sub-permutation and it has arity $\mathbf{a} \leq (j - (i+1))$, then each inference consists of a $\mathbf{a}$-tuple $[p, l_1, \ldots, l_{\mathbf{a}-1}]$, where the **operator** p is the permutation of the $\mathbf{a}$ sub-permutations ("children" of π_{i+1}^j), and for each $1 \leq x \leq (\mathbf{a}-1)$, l_x is a "split point" which is given by the index of the last integer in the x^{th} sub-permutation in π.

Function PEF$(i, j, \pi, \mathcal{F})$;
`# Args: sub-perm. π over [i..j] and forest F`
Output: Parse-Forest $\mathcal{F}(\pi)$ for π;

begin
1. if $([[i, j, \star]] \in \mathcal{F})$ then return $\mathcal{F}$; `#memoization`
2. $\mathbf{a} := \mathbf{a}(\pi)$;
3. if $\mathbf{a} = 1$ return $\mathcal{F} := \mathcal{F} \cup \{[[i, j, \emptyset]]\}$;
4. For each set of split points $\{l_1, \ldots, l_{\mathbf{a}-1}\}$ do
5. $\quad p := \text{RANKLISTOF}(\pi_1^{l_1}, \pi_{(l_1+1)}^{l_2}, \ldots, \pi_{(l_{\mathbf{a}-1}+1)}^{n})$;
6. $\quad \mathcal{I}_i^j := \mathcal{I}_i^j \cup [p, l_1, \ldots, l_{\mathbf{a}-1}]$;
7. $\quad$ For each $\pi_v \in \{\pi_1^{l_1}, \pi_{(l_1+1)}^{l_2}, \ldots, \pi_{(l_{\mathbf{a}-1}+1)}^{n}\}$ do
8. $\quad\quad \mathcal{F} := \mathcal{F} \cup PermForest(\pi_v)$;
9. $\mathcal{F} := \mathcal{F} \cup \{[[i, j, \mathcal{I}_i^j]]\}$;
10. Return $\mathcal{F}$;
end;

Figure 6: Pseudo-code of permutation-forest factorization algorithm. Function $\mathbf{a}(\pi)$ returns the arity of π. Function RANKLISTOF$(r_1, \ldots, r_m)$ employs *Counting Sort* (Cormen et al., 2001) to sort the sub-permutations $r_1, \ldots, r_m$ as integer ranges in $O(n)$, and returns a permutation p over $[1..m]$ signifying their order. The top-level call is PEF$(\pi, 0, n, \emptyset)$. We will use PEF(π) thereby overloading PEF$(\cdot)$.

Let us exemplify the inferences on $\pi = \langle 4,3,2,1\rangle$ (see Figure 5b) which factorizes into pairs of sub-permutations ($\mathbf{a} = 2$): a split point can be at positions with index $l_1 \in \{1,2,3\}$. Each of these split points (factorizations) of π is represented as an *inference* for the *same root node* which covers the whole of π (placed in entry $[0,4]$); an inference here consists of the permutation $\langle 2,1\rangle$ (swapping the two ranges covered by the children sub-permutations) together with $\mathbf{a}-1$ indexes $l_1,\ldots,l_{\mathbf{a}-1}$ signifying the split points of π into sub-permutations: since $\mathbf{a} = 2$ for π, then a single index $l_1 \in \{1,2,3\}$ is stored with every inference. For the factorization $((4,3),(2,1))$ the index $l_1 = 2$ signifying that the second position is a split point into $\langle 4,3\rangle$ (stored in entry $[0,2]$) and $\langle 2,1\rangle$ (stored in entry $[2,4]$). For the other factorizations of π similar inferences are stored in the permutation forest.

Figure 6 shows a simple top-down factorization algorithm which starts out by computing the arity $\mathbf{a}$ using function $\mathbf{a}(\pi)$. If $\mathbf{a} = 1$, a single leaf node is stored with an empty set of inferences. If $\mathbf{a} > 1$ then the algorithm computes all possible factorizations of π into $\mathbf{a}$ sub-permutations (a sequence of $\mathbf{a}-1$ split points) and stores their inferences together as $\mathcal{I}_i^J$ associated with a node in entry $[[i,j,\mathcal{I}_i^j]]$. Subsequently, the algorithm applies recursively to each sub-permutation.

The Albert and Atkinson uniqueness results for $\mathbf{a} \geq 4$ implies that the number of sets of split points is exactly one. For $\mathbf{a} = 2$ there are at most $n-1$ such sets. This means that line 4 in Figure 6 is at most linear in n. Similarly for $\mathbf{a} \geq 4$ line 7 does at most $(n-1)$ recursive calls, and for $\mathbf{a} = 2$ only two. In total, this algorithm has time complexity $O(n^3)$.

6 Evaluation metrics by factorization

So far we presented the Permutation Complexity class of metrics and defined tightness. In this section we present example tight and semi-tight metrics.

The (semi-)tight metrics presented next are linear-time in permutation length. Each of these metrics concentrates on one aspect of PETs/PEFs: factorization extent ($|\textsc{Pet}|$), bracketing freedom ($\#\textsc{Pets}$), and maximum arity ($\textsc{Max}_{|\mathbf{Op}|}$). These example metrics are summarized in Figure 7.

$$
\begin{aligned}
|\textsc{Pet}|(\pi) &= \frac{\textsc{Count}_{node}(PET(\pi))-1}{n-2} \\
\#\textsc{Pets}(\pi) &= \frac{\textsc{Count}_{pet}(\textsc{Pef}(\pi))-1}{\textsc{Count}_{pet}(\textsc{Pef}(\textsc{Id}^n))-1} \\
\textsc{Max}_{|\mathrm{Op}|}(\pi) &= 1 - \frac{MaxOp(PET(\pi))-2}{n-2}
\end{aligned}
$$

Figure 7: Summary of metrics: $\textsc{Count}_{node}$ is number of nodes in $PET(\pi)$; $MaxOp(PET)$ is maximum operator length in PET; $\textsc{Count}_{pet}(PEF)$ returns count of PETs in PEF.

$|\textbf{PET}|(\pi)$ is the ratio of number of nodes in a PET of π to the number of nodes in PET for $\textsc{Id}^n$. This is a *tight metric cf. Section 4.*

$\#\textbf{PETs}(\pi)$ is the ratio of number of different PETs that π factorizes into to this number for a fully monotone permutation. This metric is semi-tight: consider a flattened PET together with a complexity function based on average operator length – taking into account that flattened nodes receive operator length expressed as monotone decreasing in the Catalan number.

$\textbf{MAX}_{|\mathbf{Op}|}(\pi)$ is one minus the normalized maximum operator length in a PET of π (normalized by the range of lengths, i.e., $[2..n]$).

Having defined tight and semi-tight metrics, next we will evaluate these metrics against a gold standard: human judgements in MT.

7 Experimental setting

Data We use human rankings of translations from WMT13 (Bojar et al., 2013) for ten language pairs with a diverse set of MT systems.

Meta-evaluation We conduct **system level** meta-evaluation by following the method used in (Macháček and Bojar, 2013). All MT systems were first ranked by the ratio of the times they were judged to be better than some other system. All the metrics that we tested compute system level scores for the same systems and then we rank systems by that score (per each metric). The rankings that are

		English-Czech	English-Russian	English-French	English-Spanish	English-German
baselines	HAMMING	0.868 ± 0.033	0.511 ± 0.056	0.911 ± 0.016	0.806 ± 0.056	0.851 ± 0.024
	KENDALL	0.849 ± 0.03	0.511 ± 0.039	0.907 ± 0.014	0.844 ± 0.076	0.918 ± 0.019
	SPEARMAN	0.852 ± 0.029	0.508 ± 0.041	0.907 ± 0.014	0.848 ± 0.074	0.915 ± 0.019
	FUZZY	0.854 ± 0.03	0.498 ± 0.044	0.92 ± 0.014	0.818 ± 0.058	0.897 ± 0.018
	ULAM	0.851 ± 0.029	0.507 ± 0.041	0.914 ± 0.014	0.844 ± 0.07	0.908 ± 0.022
tight	|PET|	0.853 ± 0.029	0.515 ± 0.042	0.907 ± 0.013	0.866 ± 0.074	0.923 ± 0.018
semi	#PETS	0.879 ± 0.053	0.538 ± 0.103	0.904 ± 0.016	0.797 ± 0.052	0.819 ± 0.03
	$\text{MAX}_{\|\text{Op}\|}$	0.849 ± 0.029	0.513 ± 0.043	0.907 ± 0.013	0.864 ± 0.074	0.924 ± 0.018
	BLEU	0.895 ± 0.028	0.574 ± 0.057	0.897 ± 0.034	0.759 ± 0.078	0.786 ± 0.034

Table 1: Correlation with human judgement out of English.

		Czech-English	Russian-English	French-English	Spanish-English	German-English
baselines	HAMMING	0.878 ± 0.028	0.761 ± 0.035	0.984 ± 0.012	0.88 ± 0.033	0.851 ± 0.021
	KENDALL	0.887 ± 0.026	0.831 ± 0.021	0.969 ± 0.012	0.831 ± 0.064	0.905 ± 0.016
	SPEARMAN	0.881 ± 0.025	0.831 ± 0.02	0.967 ± 0.014	0.826 ± 0.066	0.905 ± 0.017
	FUZZY	0.931 ± 0.016	0.81 ± 0.023	0.977 ± 0.009	0.889 ± 0.029	0.894 ± 0.015
	ULAM	0.909 ± 0.026	0.83 ± 0.021	0.974 ± 0.009	0.86 ± 0.054	0.895 ± 0.015
tight	|PET|	0.895 ± 0.026	0.839 ± 0.021	0.965 ± 0.012	0.818 ± 0.064	0.918 ± 0.014
semi	#PETS	0.878 ± 0.034	0.698 ± 0.038	0.959 ± 0.021	0.883 ± 0.042	0.786 ± 0.039
	$\text{MAX}_{\|\text{Op}\|}$	0.895 ± 0.025	0.838 ± 0.021	0.966 ± 0.012	0.819 ± 0.064	0.921 ± 0.014
	BLEU	0.936 ± 0.036	0.651 ± 0.041	0.993 ± 0.014	0.879 ± 0.051	0.902 ± 0.017

Table 2: Correlation with human judgement into English.

produced by all metrics are compared with human judgment using Spearman rank correlation coefficient. When there are no ties, Spearman correlation can be expressed by $\rho = 1 - \frac{6 \sum d_i^2}{n(n^2-1)}$, where $d_i = y_i - x_i$ represents a distance in ranks given by humans and the metric for system i.

Statistical significance We use bootstrap re-sampling with 1000 samples for computing statistical significance. We apply the t-test and we consider a difference significant if $p < 0.05$.

		HAMMING	KENDALL	SPEARMAN	FUZZY	ULAM
Tight	|PET|	**5/4/1**	**7/2/1**	**6/2/2**	**5/4/1**	**5/4/1**
S-tight	$\text{MAX}_{\|\text{Op}\|}$	**5/4/1**	**5/2/3**	**6/2/2**	5/5/0	**5/4/1**
S-tight	#PETS	2/6/2	3/7/0	3/7/0	2/8/0	3/7/0

Table 3: Pairs-wise comparison over 10 language pairs. In the triple $N/B/D$: N is number of language pairs where the new metric significantly outperforms the baseline, B is baseline outperforms new metric and D is the number of language pairs where the difference is insignificant (draw). Bold show the cases where $N > B$.

Evaluating reordering All the tested metrics are defined on the sentence level. Since words in the reference or system translations might not be aligned, we introduce a brevity penalty for the ordering component as in (Isozaki et al., 2010).[8] After scaling sentence-level reordering score $ordering(\pi)$ by a brevity-penalty $BP(|\pi|, |ref|)$, we interpolate the result with a reordering-free (bag-of-words) lexical score $F1(ref, sys)$, i.e.,[9] $SenScore(ref, sys) = \alpha \times F1(ref, sys) + (1 - \alpha) \times BP(|\pi|, |ref|) \times ordering(\pi)$, where π is the permutation represent-

[8]This is the same as in BLEU with the small difference that instead of taking the length of system and reference translation as its parameters, it takes the length of the system permutation and the length of the reference.

[9]$F1 = 2 \times \frac{precision \times recall}{precision + recall}$, where $precision(ref, sys) = \frac{|ref \cap sys|}{|sys|}$ and $recall(ref, sys) = \frac{|ref \cap sys|}{|ref|}$, assuming each of ref and sys is represented as a bag of words.

ing the word alignment between sys and ref. The interpolation parameter was fixed $\alpha = 0.5$, weighing both lexical and reordering metrics equally, to avoid introducing preference for one over the other, but in principle this could be tuned on human rankings.

We score a system $\mathcal{S}$ by aggregating $SenScore$ weighted by reference length over reference-system pairs in the system's corpus $\mathcal{C}_\mathcal{S}$ and normalize: $Score(\mathcal{S}) = \frac{\sum_{(ref,sys)\in\mathcal{C}_\mathcal{S}} |ref| \times SenScore(ref,sys)}{\sum_{(ref,sys)\in\mathcal{C}_\mathcal{S}} |ref|}$.

Word alignments We align system and reference translations directly using the METEOR aligner (Denkowski and Lavie, 2011), which implements beam search over all possible monolingual alignments that could be built with exact, stem, WordNet and paraphrase match, where each matching mode is weighted depending on language pair.[10]

All metrics in our experiments are interpolated in the same manner with lexical component and brevity penalty, and are fed with the same input permutations.

Results The scores for translation into-English are in Table 2. Table 1 shows the results for the out-of-English direction. We also include BLEU-Moses straight from WMT13 tables for an impression regarding a known full metric. The present tight/semi-tight metrics outperform the baselines on six language pairs (English into Czech/ Russian/ Spanish/ German, and out of Russian and German). But the baselines prevail on four (English into French, and out of Czech, French and Spanish). We hypothesize that English-French shows local reordering where hierarchical factorization has small effect. The results for French- and Spanish-English might be explained similarly. For English-Russian and English-Czech, #PETs (bracketing freedom) is superior, likely because Russian and Czech allow freer order than English which is difficult for MT systems to capture. English-Russian shows low correlations for all metrics (including BLEU), suggesting that either all systems participating are judged of lower quality, or that human judgements are less consistent. For Czech-English, FUZZY, which outperforms all metrics, concentrates on monotone patterns suggesting that Czech-English MT systems in WMT13 differ mainly in how well they obtain correct phrases/blocks in their translations rather than long-distance ordering.

Comparison between ten metrics over ten language pairs is difficult. Hence, we present a pair-wise comparison between the metrics. Table 3 shows for each new metric $\mathcal{N}$ and baseline $\mathcal{B}$ a ratio $N/B/D$ where N is the number of language-pairs where statistically significant improvement by $\mathcal{N}$ over $\mathcal{B}$ is found, B is the reverse situation and D is the number of draws (insignificant difference).

Table 3 shows clearly that the tight metric |PET| performs more often than not better than each of the baselines. Semi-tight metric $\text{MAX}_{|\text{Op}|}$ concerns factorizability and performs as well as FUZZY outperforming the other baselines. Semi-tight metric #PETs concerns bracketing freedom and performs worse than many baselines, suggesting that for most language pairs bracketing freedom, which does not always favor more factorization, is not sufficient. Furthermore, tight and semi-tight metrics |PET| and $\text{MAX}_{|\text{Op}|}$ outperform the *not* semi-tight metrics suggesting that the improvement comes from (semi-)tightness rather than arbitrary functions over trees.

Our results exemplify that factorizing word order mismatch might have higher chance of correlating with human evaluation than the baselines. The tight and semi-tight metrics tested here are simple instantiations that illustrate the general class. More effective variants do more justice to the complexity of primal permutations. Furthermore, different metrics cover different dimensions of complexity. The results show that the importance of a dimension depends on the language pair.

8 Conclusions

The factorized representations of permutations as PETs and PEFs bring together two ingredients (1) grouping words into blocks, and (2) factorization into primal permutations. In this paper we propose a class of metrics, Permutation Complexity, define and show tightness for a sub-class, extend PETs to PEFs and explore example (semi-)tight evaluation metrics exploiting both the hierarchical and primality dimensions. Experiments with WMT13 data show that tight or semi-tight metrics compare favorably

[10]We also make METEOR minimize the number of unaligned words using "-t maxcov".

to the baselines in correlation with human evaluation. Our results can be seen as novel evidence suggesting that tightness might constitute a guiding principle for word order evaluation. The metrics presented in this work only exemplify the range of possible metrics based on the same intuition. In future work we aim at further ordering of the space of metrics, exploring a variety of new complexity metrics, and testing their value on various (evaluation) tasks.

Acknowledgments

This work is supported by STW grant nr. 12271 and NWO VICI grant nr. 277-89-002.

References

Michael H. Albert and Mike D. Atkinson. 2005. Simple permutations and pattern restricted permutations. *Discrete Mathematics*, 300(1-3):1–15.

Alexandra Birch and Miles Osborne. 2010. LRscore for Evaluating Lexical and Reordering Quality in MT. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 327–332, Uppsala, Sweden, July. Association for Computational Linguistics.

Alexandra Birch and Miles Osborne. 2011. Reordering Metrics for MT. In *Proceedings of the Association for Computational Linguistics*, Portland, Oregon, USA. Association for Computational Linguistics.

Alexandra Birch, Miles Osborne, and Philipp Koehn. 2008. Predicting success in machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 745–754, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alexandra Birch, Miles Osborne, and Phil Blunsom. 2010. Metrics for MT evaluation: evaluating reordering. *Machine Translation*, pages 1–12.

Arianna Bisazza and Marcello Federico. 2013. Dynamically shaping the reordering search space of phrase-based statistical machine translation. *TACL*, 1:327–340.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August. Association for Computational Linguistics.

Fabienne Braune, Anita Gojun, and Alexander Fraser. 2012. Long-distance reordering during search for hierarchical phrase-based SMT. In *Proceedings of the European Association for Machine Translation (EAMT12)*, pages 177–184, Trento, Italy.

Robert Brignall. 2010. A survey of simple permutations. In Steve Linton, Nik Ruškuc, and Vincent Vatter, editors, *Permutation Patterns*, volume 376 of *London Math. Soc. Lecture Note Ser.*, pages 41–65. Cambridge Univ. Press, Cambridge.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 2(33):201–228.

Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. 2001. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition.

John DeNero and Jakob Uszkoreit. 2011. Inducing sentence structure from parallel corpora for reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 193–203, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proceedings of the EMNLP 2011 Workshop on Statistical Machine Translation*.

Revital Eres, Gad M. Landau, and Laxmi Parida. 2004. Permutation pattern discovery in biosequences. *Journal of Computational Biology*, 11(6):1050–1060.

Yang Gao, Philipp Koehn, and Alexandra Birch. 2011. Soft dependency constraints for reordering in hierarchical phrase-based translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 857–868, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daniel Gildea, Giorgio Satta, and Hao Zhang. 2006. Factoring Synchronous Grammars by Sorting. In *ACL*.

Keith Hall, Ryan McDonald, and Slav Petrov. 2011. Training structured prediction models with extrinsic loss functions. In *Domain Adaptation Workshop at NIPS*, October.

Teresa Herrmann, Jochen Weiner, Jan Niehues, and Alex Waibel. 2011. Analyzing the potential of source sentence reordering in statistical machine translation. In *Proceedings of International Workshop on Spoken Language Translation (IWSLT)*.

Liang Huang, Hao Zhang, Daniel Gildea, and Kevin Knight. 2009. Binarization of synchronous context-free grammars. *Computational Linguistics*, 35(4):559–595.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 944–952, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jason Katz-Brown, Slav Petrov, Ryan McDonald, Franz Och, David Talbot, Hiroshi Ichikawa, Masakazu Seno, and Hideto Kazawa. 2011. Training a parser for machine translation reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 183–192, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mirella Lapata. 2006. Automatic Evaluation of Information Ordering: Kendall's Tau. *Computational Linguistics*, 32(4):471–484.

Matouš Macháček and Ondřej Bojar. 2013. Results of the WMT13 Metrics Shared Task. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 45–51, Sofia, Bulgaria, August. Association for Computational Linguistics.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. 2012. Inducing a discriminative parser to optimize machine translation reordering. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 843–853, Stroudsburg, PA, USA. Association for Computational Linguistics.

James H. Schmerl and William T. Trotter. 1993. Critically indecomposable partially ordered sets, graphs, tournaments and other binary relational structures. *Discrete Mathematics*, 113(1-3):191–205.

Miloš Stanojević and Khalil Sima'an. 2014a. Evaluating Word Order Recursively over Permutation-Forests. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 138–147, Doha, Qatar, October. Association for Computational Linguistics.

Miloš Stanojević and Khalil Sima'an. 2014b. Fitting Sentence Level Translation Evaluation with Many Dense Features. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 202–206, Doha, Qatar, October. Association for Computational Linguistics.

David Talbot, Hideto Kazawa, Hiroshi Ichikawa, Jason Katz-Brown, Masakazu Seno, and Franz Och. 2011. A Lightweight Evaluation Framework for Machine Translation Reordering. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 12–21, Edinburgh, Scotland, July. Association for Computational Linguistics.

Takeaki Uno and Mutsunori Yagiura. 2000. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309.

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 746–754.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 3(23):377–403.

Bing Xiang, Niyu Ge, and Abraham Ittycheriah. 2011. Improving reordering for statistical machine translation with smoothed priors and syntactic features. In *Proceedings of the Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation*, SSST-5, pages 61–69, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hao Zhang and Daniel Gildea. 2007. Factorization of Synchronous Context-Free Grammars in Linear Time. In *NAACL Workshop on Syntax and Structure in Statistical Translation (SSST)*, pages 25–32.

Interactive Attention for Neural Machine Translation

Fandong Meng[1*] **Zhengdong Lu**[2] **Hang Li**[2] **Qun Liu**[3,4]

[1]AI Platform Department, Tencent Technology Co., Ltd.
fandongmeng@tencent.com
[2]Noah's Ark Lab, Huawei Technologies
{Lu.Zhengdong,HangLi.HL}@huawei.com
[3]ADAPT Centre, School of Computing, Dublin City University
[4]Key Lab of Intelligent Information Processing, Institute of Computing Technology, CAS
qliu@computing.dcu.ie

Abstract

Conventional attention-based Neural Machine Translation (NMT) conducts dynamic alignment in generating the target sentence. By repeatedly reading the representation of source sentence, which keeps fixed after generated by the encoder (Bahdanau et al., 2015), the attention mechanism has greatly enhanced state-of-the-art NMT. In this paper, we propose a new attention mechanism, called INTERACTIVE ATTENTION, which models the interaction between the decoder and the representation of source sentence during translation by both reading and writing operations. INTERACTIVE ATTENTION can keep track of the interaction history and therefore improve the translation performance. Experiments on NIST Chinese-English translation task show that INTERACTIVE ATTENTION can achieve significant improvements over both the previous attention-based NMT baseline and some state-of-the-art variants of attention-based NMT (i.e., coverage models (Tu et al., 2016)). And neural machine translator with our INTERACTIVE ATTENTION can outperform the open source attention-based NMT system Groundhog by 4.22 BLEU points and the open source phrase-based system Moses by 3.94 BLEU points averagely on multiple test sets.

1 Introduction

Neural Machine Translation (NMT) has made promising progress in recent years (Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015a; Jean et al., 2015; Luong et al., 2015b; Tang et al., 2016; Wang et al., 2016; Li et al., 2016; Tu et al., 2016; Shen et al., 2016; Zhou et al., 2016), in which attention model plays an increasingly important role. Attention-based NMT represents the source sentence as a sequence of vectors after a RNN or bi-directional RNN (Schuster and Paliwal, 1997), and then simultaneously conducts dynamic alignment with a gating neural network and generation of the target sentence with another RNN. Usually NMT with attention model is more efficient than its attention-free counterpart: it can achieve comparable results with far less parameters and training instances (Jean et al., 2015). This superiority in efficiency comes mainly from the mechanism of dynamic alignment, which avoids the need to represent the entire source sentence with a fixed-length vector (Sutskever et al., 2014).

However, conventional attention model is conducted on the representation of source sentence (fixed after generated) only with reading operation (Bahdanau et al., 2015; Luong et al., 2015a). This may let the decoder tend to ignore past attention information, and lead to over-translation and under-translation (Tu et al., 2016). To address this problem, Tu et al. (2016) proposed to maintain tag vectors in source representation to keep track of the attention history, which encourages the attention-based NMT system to consider more untranslated source words. Inspired by neural turing machines (Graves et al., 2014), we propose INTERACTIVE ATTENTION model from the perspective of memory reading-writing, which provides a conceptually simpler and practically more effective mechanism for attention-based NMT. The NMT with INTERACTIVE ATTENTION is called NMT$_{|\mathsf{A}}$, which can keep track of the interaction history with the representation of source sentence by both reading and writing operations during translation.

[*] The majority of this work was completed when the first author studied at Institute of Computing Technology, Chinese Academy of Sciences.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2174–2185, Osaka, Japan, December 11-17 2016.

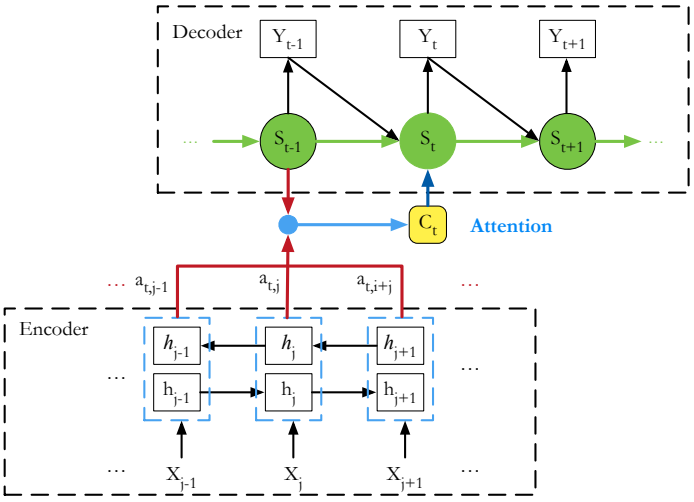


Figure 1: Illustration for attention-based NMT.

This interactive mechanism may be helpful for the decoder to automatically distinguish which parts have been translated and which parts are under-translated.

We test the efficacy of NMT_{IA} on NIST Chinese-English translation task. Experiment results show that NMT_{IA} can significantly outperform both the conventional attention-based NMT baseline (Bahdanau et al., 2015) and coverage models (Tu et al., 2016). And neural machine translator with our INTERACTIVE ATTENTION can outperform the open source attention-based NMT system Groundhog by 4.22 BLEU points and the open source phrase-based system Moses by 3.94 BLEU points.

RoadMap: In the remainder of this paper, we will start with a brief overview of attention-based neural machine translation in Section 2. Then in Section 3, we will detail the INTERACTIVE ATTENTION-based NMT (NMT_{IA}). In Section 4, we report our empirical study of NMT_{IA} on a Chinese-English translation task, followed by Section 5 and 6 for related work and conclusion.

2 Background

Our work is built upon the attention-based NMT (Bahdanau et al., 2015), which takes a sequence of vector representations of the source sentence generated by a RNN or bi-directional RNN as input, and then jointly learns to align and translate by reading the vector representations during translation with a RNN decoder. Therefore, we take an overview of the attention-based NMT in this section before detail the NMT_{IA} in next section.

2.1 Attention-based Neural Machine Translation

Figure 1 shows the framework of attention-based NMT. Formally, given an input source sequence $\mathbf{x}=\{x_1, x_2, \cdots, x_N\}$ and the previously generated target sequence $\mathbf{y}_{<\mathbf{t}}=\{y_1, y_2, \cdots, y_{t-1}\}$, the probability of the next target word y_t is

$$p(y_t|\mathbf{y}_{<\mathbf{t}}, \mathbf{x}) = softmax(f(\mathbf{c}_t, y_{t-1}, \mathbf{s}_t)) \quad (1)$$

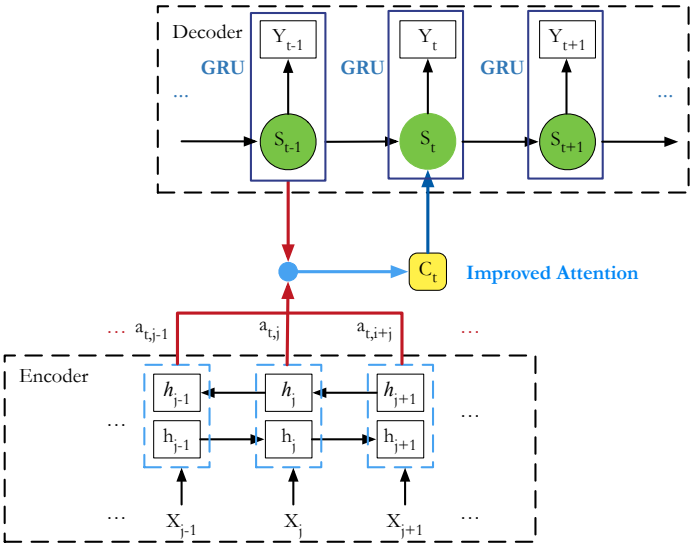


Figure 2: Illustration for improved attention model of NMT.

where $f(\cdot)$ is a non-linear function, and $\mathbf{s}_t$ is the state of decoder RNN at time step t which is calculated as

$$\mathbf{s}_t = g(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c}_t) \tag{2}$$

where $g(\cdot)$ can be any activation function, here we adopt a more sophisticated dynamic operator as in Gated Recurrent Unit (GRU) (Cho et al., 2014). In the remainder of the paper, we will also use GRU to stand for the operator. And $\mathbf{c}_t$ is a distinct source representation for time t, calculated as a weighted sum of the source annotations:

$$\mathbf{c}_t = \sum_{j=1}^{N} a_{t,j}\mathbf{h}_j \tag{3}$$

Formally, $\mathbf{h}_j = [\overrightarrow{\mathbf{h}_j}^T, \overleftarrow{\mathbf{h}_j}^T]^T$ is the annotation of x_j, which is computed by a bi-directional RNN (Schuster and Paliwal, 1997) with GRU and contains information about the whole input sequence with a strong focus on the parts surrounding x_j. And its weight $a_{t,j}$ is computed by

$$a_{t,j} = \frac{exp(e_{t,j})}{\sum_{k=1}^{N} exp(e_{t,k})} \tag{4}$$

where $e_{t,j} = \mathbf{v}_a^T tanh(\mathbf{W}_a\mathbf{s}_{t-1} + \mathbf{U}_a\mathbf{h}_j)$ scores how well $\mathbf{s}_{t-1}$ and $\mathbf{h}_j$ match. This is called automatic alignment (Bahdanau et al., 2015) or attention model (Luong et al., 2015a), but it is essentially reading with content-based addressing defined in (Graves et al., 2014). With the attention model, it releases the need to summarize the entire sentence with a single fixed-length vector (Sutskever et al., 2014; Cho et al., 2014). Instead, it lets the decoding network focus on one particular segment in source sentence at one moment, and therefore better resolution.

2.2 Improved Attention Model

The alignment model $a_{t,j}$ scores how well the output at position t matches the inputs around position j based on $\mathbf{s}_{t-1}$ and $\mathbf{h}_j$. Intuitively, it should be beneficial to directly exploit the information of y_{t-1} when

reading from the representation of source sentence, which is not implemented in the original attention-based NMT (Bahdanau et al., 2015). As illustrated in Figure 2, we add this implementation into the attention model, inspired by the latest implementation of attention-based NMT[1]. This kind of attention model can find a more effective alignment path by using both previous hidden state $\mathbf{s}_{t-1}$ and the previous context word y_{t-1}. Then, the calculation of $e(t,j)$ becomes

$$e_{t,j} = \mathbf{v}_a^T tanh(\mathbf{W}_a \tilde{s}_{t-1} + \mathbf{U}_a \mathbf{h}_j) \tag{5}$$

where $\tilde{\mathbf{s}}_{t-1} = \mathbf{GRU}(\mathbf{s}_{t-1}, \mathbf{e}_{y_{t-1}})$ is an intermediate state tailored for reading from the representation of source sentence with the information of y_{t-1} (its word embedding being $\mathbf{e}_{y_{t-1}}$) added. And the calculation of update-state $\mathbf{s_t}$ becomes

$$\mathbf{s}_t = \mathbf{GRU}(\tilde{\mathbf{s}}_{t-1}, \mathbf{c}_t) \tag{6}$$

3 Interactive Attention

In this section, we will elaborate on the proposed INTERACTIVE ATTENTION-based NMT, called NMT$_{\mathsf{IA}}$. Figure 3 shows the framework of NMT$_{\mathsf{IA}}$ with two rounds of interactive read-write operations (indicated by the yellow and red arrows respectively), which adopts the same prediction model (Eq. 1) with improved attention-based NMT. With annotations $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N\}$ of the source sentence $\mathbf{x} = \{x_1, x_2, \cdots, x_N\}$, we take $\mathbf{H}$ as a memory, which contains N cells with the jth cell being $\mathbf{h}_j$. As illustrated in Figure 3, INTERACTIVE ATTENTION in NMT$_{\mathsf{IA}}$ contains two key parts at each time step t: 1) attentive reading from $\mathbf{H}$, and 2) attentive writing to $\mathbf{H}$. Since the content in $\mathbf{H}$ changes with time, we will add time stamp on $\mathbf{H}$ (hence $\mathbf{H}^{(t)}$) and its cells (hence $\mathbf{h}_j^{(t)}$).

At time t, the state $\mathbf{s}_{t-1}$ first meets the prediction y_{t-1} to form an "intermediate" state $\tilde{\mathbf{s}}_{t-1}$, which can be calculated as follows

$$\tilde{\mathbf{s}}_{t-1} = \mathbf{GRU}(\mathbf{s}_{t-1}, \mathbf{e}_{y_{t-1}}) \tag{7}$$

where $\mathbf{e}_{y_{t-1}}$ is the word-embedding associated with the previous prediction word y_{t-1}. This "intermediate" state $\tilde{\mathbf{s}}_{t-1}$ is used to read the source memory $\mathbf{H}^{(t-1)}$

$$\mathbf{c}_t = \mathbf{Read}(\tilde{\mathbf{s}}_{t-1}, \mathbf{H}^{(t-1)}) \tag{8}$$

After that, $\tilde{\mathbf{s}}_{t-1}$ is combined with $\mathbf{c}_t$ to update the new state

$$\mathbf{s}_t = \mathbf{GRU}(\tilde{\mathbf{s}}_{t-1}, \mathbf{c}_t) \tag{9}$$

Finally, the new state $\mathbf{s}_t$ is used to update the source memory by writing to it to finish the interaction in a round of state-update

$$\mathbf{H}^{(t)} = \mathbf{Write}(\mathbf{s}_t, \mathbf{H}^{(t-1)}) \tag{10}$$

The details of $\mathbf{Read}$ and $\mathbf{Write}$ in Eq. 8 and 10 will be described later in next section.

From the whole framework of NMT$_{\mathsf{IA}}$, we can see that the new attention mechanism can timely update the representation of source sentence along with the update-chain of the decoder RNN state. This may let the decoder keep track of the attention history during translation. Clearly, INTERACTIVE ATTENTION can subsume the coverage models in (Tu et al., 2016) as special cases while conceptually simpler. Moreover, with the attentive writing, INTERACTIVE ATTENTION potentially can modify and add more on the source representation than just history of attention, and is therefore a more powerful model for machine translation, as empirically verified in Section 4.

[1] https://github.com/nyu-dl/dl4mt-tutorial/tree/master/session2

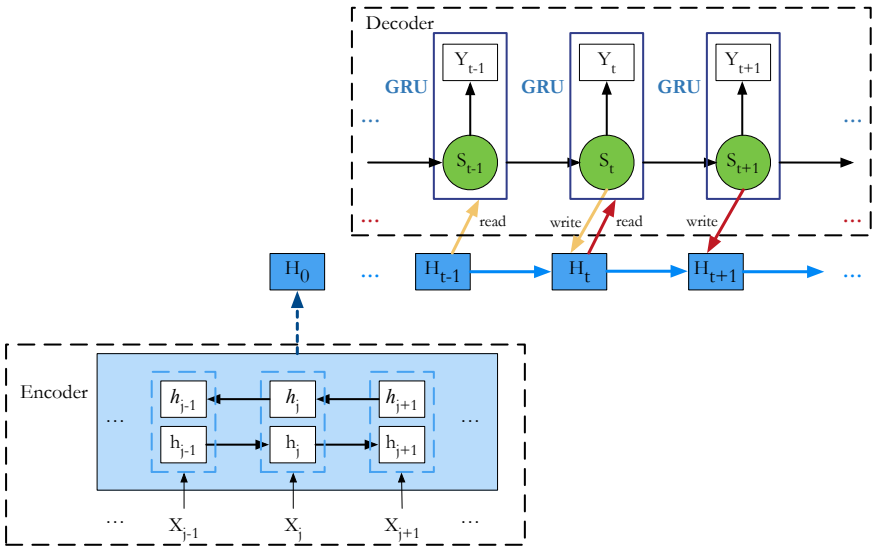


Figure 3: Illustration for the NMT$_{|A}$. The yellow and red arrows indicate two rounds of interactive read-write operations.

3.1 Read and Write of Interactive Attention

Attentive Read Formally, $\mathbf{H}^{(t')} \in \mathbb{R}^{n \times m}$ is the memory in time t' after the decoder RNN state update, where n is the number of memory cells and m is the dimension of vector in each cell. Before the state $\mathbf{s}$ update at time t, the output of reading $\mathbf{c}_t$ is given by

$$\mathbf{c}_t = \sum_{j=1}^{n} \mathbf{w}_t^R(j)\mathbf{h}_j^{(t-1)} \tag{11}$$

where $\mathbf{w}_t^R \in \mathbb{R}^n$ specifies the normalized weights assigned to the cells in $\mathbf{H}^{(t-1)}$. We can use content-based addressing to determine $\mathbf{w}_t^R$ as described in (Graves et al., 2014) or (quite similarly) use the reading mechanism such as the attention model in Section 2. In this paper, we adopt the latter one.[2]

Attentive Write Inspired by the writing operation of neural turing machines (Graves et al., 2014), we define two types of operation on writing to the memory: FORGET and UPDATE. FORGET is similar to the forget gate in GRU, which determines the content to be removed from memory cells. More specifically, the vector $\mathbf{F}_t \in \mathbb{R}^m$ specifies the values to be forgotten or removed on each dimension in memory cells, which is then assigned to each cell through normalized weights $\mathbf{w}_t^W$. Formally, the memory ("intermediate") after FORGET operation is given by

$$\tilde{\mathbf{h}}_i^{(t)} = \mathbf{h}_i^{(t-1)}(1 - \mathbf{w}_t^W(i) \cdot \mathbf{F}_t), \qquad i = 1, 2, \cdots, n \tag{12}$$

where

- $\mathbf{F}_t = \sigma(\mathbf{W}_F, s_t)$ is parameterized with $\mathbf{W}_F \in \mathbb{R}^{m \times m}$, and σ stands for the $Sigmoid$ activation function;
- $\mathbf{w}_t^W \in \mathbb{R}^n$ specifies the normalized weights assigned to the cells in $\mathbf{H}^{(t)}$, and $\mathbf{w}_t^W(i)$ specifies the weight associated with the ith cell in the same parametric form as $\mathbf{w}_t^R$.

[2] Wang et al. (2016) verified the former one for the read operation on the external memory.

UPDATE is similar to the update gate in GRU, deciding how much current information should be written to the memory as the added content

$$\mathbf{h}_i^{(t)} = \tilde{\mathbf{h}}_i^{(t)} + \mathbf{w}_t^W(i) \cdot \mathbf{U}_t, \qquad i = 1, 2, \cdots, n \tag{13}$$

where $\mathbf{U}_t = \sigma(\mathbf{W}_U, \mathbf{s}_t)$ is parameterized with $\mathbf{W}_U \in \mathbb{R}^{m \times m}$, and $\mathbf{U}_t \in \mathbb{R}^m$. In our experiments, the weights for reading (i.e., $\mathbf{w}_t^R$) and writing (i.e., $\mathbf{w}_t^W$) at time t are shared when conducting interaction with the source memory.

3.2 Optimization

The parameters to be optimized include the embedding of words on source and target languages, the parameters for the encoder, the decoder and other operations of NMT$_{|\mathsf{A}}$. The optimization is conducted via the standard back-propagation (BP) aiming to maximize the likelihood of the target sequence. In practice, we use the standard stochastic gradient descent (SGD) and mini-batch with learning rate controlled by AdaDelta (Zeiler, 2012).

4 Experiments

We report our empirical study of NMT$_{|\mathsf{A}}$ on Chinese-to-English translation task in this section. The experiments are designed to answer the following questions:

- Can NMT$_{|\mathsf{A}}$ achieve significant improvements over the conventional attention-based NMT?
- Can NMT$_{|\mathsf{A}}$ outperform the attention-based NMT with coverage model (Tu et al., 2016)?

4.1 Data and Metric

Our training data consist of 1.25M sentence pairs extracted from LDC corpora[3], with 27.9M Chinese words and 34.5M English words respectively. We choose NIST 2002 (MT02) dataset as our development set, which is used to monitor the training process and decide the early stop condition. And the NIST 2003 (MT03), 2004 (MT04), 2005 (MT05), 2006 (MT06) datasets are used as our test sets. The numbers of sentences in NIST MT02, MT03, MT04, MT05 and MT06 are 878, 919, 1788, 1082, and 1664 respectively. We use the case-insensitive 4-gram NIST BLEU[4] as our evaluation metric, with statistical significance test (*sign-test* (Collins et al., 2005)) between the proposed models and the baselines.

4.2 Training Details

In training the neural networks, we limit the source and target vocabulary to the most frequent 30K words for both Chinese and English, covering approximately 97.7% and 99.3% of two corpus respectively. All the out-of-vocabulary words are mapped to a special token UNK. We initialize the recurrent weight matrices as random orthogonal matrices. All the bias vectors are initialized to zero. For other parameters, we initialize them by sampling each element from the Gaussian distribution of mean 0 and variance 0.01^2. The parameters are updated by SGD and mini-batch (size 80) with learning rate controlled by AdaDelta (Zeiler, 2012) ($\epsilon = 1e^{-6}$ and $\rho = 0.95$). We train the NMT systems with the sentences of length up to 50 words in training data, and set the dimension of word embedding to 620 and the size of the hidden layer to 1000, following the settings in (Bahdanau et al., 2015). We also use dropout for our baseline NMT systems and NMT$_{|\mathsf{A}}$ to avoid over-fitting (Hinton et al., 2012). In our experiments, dropout was applied on the output layer with dropout rate setting to 0.5.

Inspired by the effort on easing the training of very deep architectures (Hinton and Salakhutdinov, 2006), we use a simple pre-training strategy to train our NMT$_{|\mathsf{A}}$. First we train a regular attention-based NMT model (Bahdanau et al., 2015). Then we use the trained NMT model to initialize the parameters of NMT$_{|\mathsf{A}}$ except for those related to the operations of INTERACTIVE ATTENTION. After that, we fine-tune all the parameters of NMT$_{|\mathsf{A}}$.

[3]The corpora include LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06.

[4]ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl

SYSTEMS	MT03	MT04	MT05	MT06	AVERAGE
Moses	31.61	33.48	30.75	31.07	31.73
Groundhog	30.96	33.09	30.61	31.12	31.45
RNNsearch*	33.42	36.04	33.60	32.24	33.83
NMT$_{\mathsf{IA}}$	35.09*	37.73*	35.53*	34.32*	35.67

Table 1: BLEU-4 scores (%) of the phrase-based SMT system (Moses), NMT baselines: Groundhog and RNNsearch* (our implementation of improved attention model as described in Section 2.2), and our INTERACTIVE ATTENTION model (NMT$_{\mathsf{IA}}$). The "*" indicates that the results are significantly ($p<0.01$) better than those of all the baseline systems.

4.3 Comparison Systems

We compare our NMT$_{\mathsf{IA}}$ with four systems:

- **Moses** (Koehn et al., 2007): an open source phrase-based translation system[5] with default configuration. The word alignments are obtained with GIZA++ (Och and Ney, 2003) on the training corpora in both directions, using the "grow-diag-final-and" balance strategy (Koehn et al., 2003). The 4-gram language model with modified Kneser-Ney smoothing is trained on the target portion of training data with the SRILM toolkit (Stolcke and others, 2002),
- **Groundhog**: an open source NMT system[6] implemented with the conventional attention model (Bahdanau et al., 2015).
- **RNNsearch***: our in-house implementation of NMT system with the improved conventional attention model as described in Section 2.2.
- **Coverage Model**: state-of-the-art variants of attention-based NMT model (Tu et al., 2016) which improve the attention mechanism through modeling a soft coverage on the source representation by maintain a coverage vector to keep track of the attention history during translation.

4.4 Main Results

The main results of different models are given in Table 1. Before proceeding to more detailed comparisons, we first observe that

- RNNsearch* outperforms Groundhog, which is implemented with the conventional attention model as described in Section 2.1, by 2.38 BLEU points averagely on four test sets;
- RNNsearch* only exploit sentences of length up to 50 words with 30K vocabulary, but can achieve averagely 2.10 BLEU points higher than the open source phrase-based system Moses, which is trained with full training data.

Clearly from Table 1, NMT$_{\mathsf{IA}}$ can achieve significant improvements over RNNsearch* by 1.84 BLEU points averagely on four test sets. We conjecture it is because our INTERACTIVE ATTENTION mechanism can keep track of the interaction history between the decoder and the representation of source sentence during translation, which may be helpful for the decoder to automatically distinguish which parts have been translated and which parts are under-translated.

4.5 INTERACTIVE ATTENTION Vs. Coverage Model

Tu et al. (2016) proposed two coverage models to let the NMT system to consider more about untranslated source words. Basically, they maintain a coverage vector for each hidden state for source to keep track of the attention history and feed the coverage vector to the attention model to help adjust future attention.

[5]http://www.statmt.org/moses/

[6]https://github.com/lisa-groundhog/GroundHog

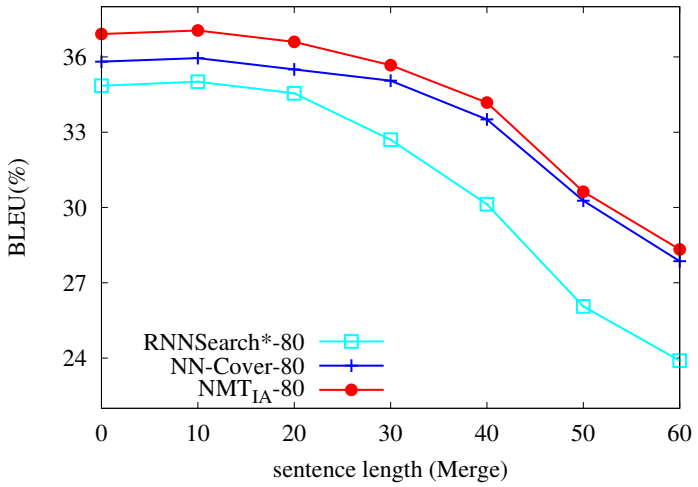


Figure 4: The BLEU-4 scores (%) of generated translations on the merged four test sets with respect to the lengths of source sentences. The numbers on X-axis of the figure stand for sentences *longer than* the corresponding length, e.g., 40 for source sentences with > 40 words.

SYSTEMS	MT03	MT04	MT05	MT06	AVERAGE
RNNsearch*-80	33.34	37.10	33.38	33.70	34.38
NN-Cover-80	33.69	38.05	35.01	34.83	35.40
NMT$_{|A}$-80	35.69*+	39.24*+	35.74*+	35.10*	36.44

Table 2: BLEU-4 scores (%) of the conventional attention-based model (RNNsearch*-80), the neural network based coverage model (NN-Cover-80) (Tu et al., 2016) and our INTERACTIVE ATTENTION model (NMT$_{|A}$-80). "-80" means the models are trained with the sentences of length up to 80 words, which is consistent with the setting in (Tu et al., 2016). The "*" and "+" denote that the results are significantly ($p<0.01$) better than those of RNNsearch*-80 and NN-Cover-80 respectively.

Although we do not maintain a coverage vector, our INTERACTIVE ATTENTION can potentially do similar things, therefore subsuming coverage models as special cases. We hence compare our INTERACTIVE ATTENTION model with the coverage model in (Tu et al., 2016). There are two coverage models proposed in (Tu et al., 2016), including linguistic coverage model and neural network based coverage model (NN-Cover). Since the neural network based coverage model generally yields better results, we mainly compare with the neural network based coverage model. Although the coverage models are originally implemented on Groundhog in (Tu et al., 2016), they can be easily adapted to the "RNNsearch*". Following the setting in (Tu et al., 2016), we conduct the comparison with the training sentences of length up to 80 words. Clearly from Table 2, our NMT$_{|A}$-80 outperforms the NN-Cover-80 by +1.04 BLEU scores averagely on four test sets.

A more detailed comparison between conventional attention model (RNNsearch*-80), neural network based coverage model (NN-Cover-80) (Tu et al., 2016) and NMT$_{|A}$-80 suggests that our NMT$_{|A}$-80 is quite consistent on outperforming the conventional attention model and the coverage model. Figure 4 shows the BLEU scores of generated translations on the test sets with respect to the length of the source sentences. In particular, we test the BLEU scores on sentences longer than $\{0, 10, 20, 30, 40, 50, 60\}$ in the merged test set of MT03, MT04, MT05 and MT06. Clearly, on sentences with different length, NMT$_{|A}$-80 always yields consistently higher BLEU scores than the conventional attention-based NMT and the enhanced version with the neural network based coverage model. We conjecture that with the attentive writing (described in Section 3.1), INTERACTIVE ATTENTION potentially can modify and add more on the source representation than just history of attention, and is therefore a more powerful model for machine translation.

We also provide some actual translation examples (see Appendix) to show that our INTERACTIVE ATTENTION can get better performance then baselines, especially on solving under-translation problem. We think the interactive mechanism of NMT_{IA} is helpful for the decoder to automatically distinguish which parts have been translated and which parts are under-translated.

5 Related Work

Our work is related to recent works that focus on improving attention models (Luong et al., 2015a; Cohn et al., 2016; Feng et al., 2016). Luong et al. (2015a) proposed to use global and local attention models to improve translation performance. They use a global one to attend to all source words and a local one to look at a subset of source words at a time. Cohn et al. (2016) extended the attention-based NMT to include structural biases from word-based alignment models, which achieved improvements across several language pairs. Feng et al. (2016) added implicit distortion and fertility models to attention-based NMT to achieve translation improvements. These works are different with our INTERACTIVE ATTENTION approach, as we use a rather generic attentive reading while at the same time performing attentive writing.

Our work is inspired by recent efforts on attaching an external memory to neural networks, such as neural turing machines (Graves et al., 2014), memory networks (Weston et al., 2014; Meng et al., 2015) and exploiting an external memory (Tang et al., 2016; Wang et al., 2016) during translation. Tang et al. (2016) exploited a phrase memory for NMT, which stores phrase pairs in symbolic form. They let the decoder utilize a mixture of word-generating and phrase-generating component, to generate a sequence of multiple words all at once. Wang et al. (2016) extended the NMT decoder by maintaining an external memory, which is operated by reading and writing operations of neural turing machines (Graves et al., 2014), while keeping a read-only copy of the original source annotations along side the "read-write" memory. These powerful extensions have been verified on Chinese-English translation tasks. Our INTERACTIVE ATTENTION is different from previous works. We take the annotations of source sentence as a memory instead of using an external memory, and we design a mechanism to directly read from and write to it during translation. Therefore, the original source annotations are not accessible in later steps. More specially, our model inherited the notation and some simple operations for writing from (Graves et al., 2014), while NMT_{IA} extends it to "unbounded" memory for representing the source. In addition, although the read-write operations in INTERACTIVE ATTENTION are not exactly the same with those in (Graves et al., 2014; Wang et al., 2016), our model can also achieve good performance.

6 Conclusion

We propose a simple yet effective INTERACTIVE ATTENTION approach, which models the interaction between the decoder and the representation of source sentence during translation by using reading and writing operations. Our empirical study on Chinese-English translation shows that INTERACTIVE ATTENTION can significantly improve the performance of NMT.

Acknowledgements

Liu is partially supported by the Science Foundation Ireland (Grant 13/RC/2106) as part of the ADAPT Centre at Dublin City University. We sincerely thank the anonymous reviewers for their thorough reviewing and valuable suggestions.

References

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

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

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 876–885, San Diego, California, June.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*, pages 531–540.

Shi Feng, Shujie Liu, Mu Li, and Ming Zhou. 2016. Implicit distortion and fertility models for attention-based encoder-decoder NMT model. *CoRR*, abs/1601.03317.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China, July.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*, pages 48–54.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL on interactive poster and demonstration sessions*, pages 177–180, Prague, Czech Republic, June.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. Towards zero unknown word in neural machine translation. In *Proceedings of IJCAI*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China, July.

Fandong Meng, Zhengdong Lu, Zhaopeng Tu, Hang Li, and Qun Liu. 2015. Neural transformation machine: A new architecture for sequence-to-sequence learning. *CoRR*, abs/1506.06442.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

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

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of ACL*, pages 1683–1692, Berlin, Germany, August.

Andreas Stolcke et al. 2002. SRILM-an extensible language modeling toolkit. In *Proceedings of ICSLP*, volume 2, pages 901–904.

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

Yaohua Tang, Fandong Meng, Zhengdong Lu, Hang Li, and Philip L. H. Yu. 2016. Neural machine translation with external phrase memory. *CoRR*, abs/1606.01792.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of ACL*, pages 76–85, Berlin, Germany, August. Association for Computational Linguistics.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Memory-enhanced decoder for neural machine translation. In *Proceedings of EMNLP*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR*, abs/1410.3916.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *CoRR*, abs/1606.04199.

APPENDIX: Actual Translation Examples

In appendix we give some example translations from RNNsearch*-80, NN-Cover-80 and NMT$_{\text{IA}}$-80, and compare them against the reference. We highlight some correct translation segments (or under-translated by baseline systems) in blue color and wrong ones in red color.

Example Translations

src	北韩 指称 这 项 核子 僵局 系 仅 涉及 美国 的 双边 议题 ， 其他 国家 进行 干预 只 会 使 问题 复杂化 。
ref	North Korea said the nuclear stalemate is a bilateral topic of discussion with United States only. The interference of other countries will only complicate the issue.
RNNsearch*-80	north korea claimed that the nuclear stalemate was only involved in bilateral issues in the united states , and other countries will find it more complicated .
NN-Cover-80	the north korea said that it had only involved bilateral talks in the united states and other countries would interfere with the issue .
NMT$_{\text{IA}}$ -80	north korea claimed that this nuclear stalemate was only related to the us bilateral agenda , and interference in other countries could only complicate the problem .

src	平壤 采取 上述 行动 之后 四 天 ， 联合国 安全 理事会 的 五 个 常任 理事国 都 为 此 一 危机 采取 预防性 外交 行动 。
ref	Four days after Pyongyang made the above move, five permanent members of the UN Security Council have all taken preventive diplomatic actions on this crisis.
RNNsearch*-80	pyongyang has taken these actions four days ago , and the five permanent members of the un security council have taken precautions against this crisis .
NN-Cover-80	in a four - day operation , the five permanent members of the un security council have taken preventive diplomatic actions for the crisis .
NMT$_{\text{IA}}$ -80	in the four days after pyongyang took the above action , the five permanent members of the un security council have taken preventive diplomatic actions for this crisis .

src	菲律宾 政府 原本 计划 于 本 月 稍后 就 恢复 正式 和谈 一 事 与 菲共 举行 初步 磋谈 。
ref	The Philippine government originally planned to hold preliminary discussions with the Philippine communists on the resumption of formal peace talks later this month.
RNNsearch*-80	the philippine government originally planned to hold a preliminary meeting with <UNK> on friday .
NN-Cover-80	the philippine government plans to resume formal peace talks with <UNK> later this month .
NMT$_{\text{IA}}$ -80	the philippine government originally planned to hold a preliminary discussion on the resumption of formal peace talks later this month .

src	他 表示 ： 「 显然 的 ， 我们 需要 做 的 第一 件 事情 是 将 伊拉克 安全 部队 国际化 ， 另 一 件 事情 是 将 过渡 政府 交给 联合国 。 」
ref	He said: "Obviously, the first thing we need to do is to internationalize the security force in Iraq. The other thing is to turn the transitional government over to the United Nations."
RNNsearch*-80	he said : " obviously , the first thing we need to do is to <UNK> iraqi security forces to the united nations . "
NN-Cover-80	he said : " obviously , we need the first thing to internationalize the security forces in iraq , and another thing is to hand over the transitional government to the united nations . "
NMT$_{\text{IA}}$ -80	he said : " obviously , the first thing we need to do is to internationalize the security forces in iraq , and the other is to send the transitional government to the united nations . "

Get Semantic With Me! The Usefulness of Different Feature Types for Short-Answer Grading

Ulrike Padó
Hochschule für Technik Stuttgart
Schellingstr. 24
70174 Stuttgart
ulrike.pado@hft-stuttgart.de

Abstract

Automated short-answer grading is key to help close the automation loop for large-scale, computerised testing in education. A wide range of features on different levels of linguistic processing has been proposed so far. We investigate the relative importance of the different types of features across a range of standard corpora (both from a language skill and content assessment context, in English and in German). We find that features on the lexical, text similarity and dependency level often suffice to approximate full-model performance. Features derived from semantic processing particularly benefit the linguistically more varied answers in content assessment corpora.

1 Introduction

Computerised testing is becoming ubiquitous in the educational domain, and automated and semi-automated grading of tests is in high demand to relieve the workload of teachers (especially in the context of Massive Open On-line Courses or repeated testing for continuous feedback during the academic year). NLP is a key technology to close or at last narrow the automation loop for grading of free-text answers and essays. We focus on the automated grading of *short-answer questions* (i.e., assessment questions that require a free-text answer up to two or three sentences in length). For this task, the training data consists of a question, at least one reference answer and several student answers. Systems then predict answer accuracy as a binary *correct-incorrect* decision or as a more fine-grained multi-class problem (or even a regression task predicting points). In contrast to the related essay grading task, correct student answers stay closer to the reference answer than good essays might to an example essay.

As is often the case in young research areas, an important contribution was made by the Semeval-2013 shared task (Dzikovska et al., 2013), which introduced standard evaluation benchmarks. Two large data sets are now available with performance standards for system comparison.

On the shared task data, researchers have experimented with various features based on linguistic processing, from syntactic information (used by a majority of entries in SemEval-2013, Dzikovska et al. (2013)) to deep semantic representations (Ott et al., 2013) or Textual Entailment (TE) systems (Zesch et al., 2013). Others, staying closer to the surface level, have recently experimented with sophisticated measures of textual similarity (Jimenez et al., 2013; Sultan et al., 2016) or inferring informative answer patterns (Ramachandran et al., 2015). However, similar to other NLP tasks (like, for example, TE), one of the biggest challenges remains beating the lexical baseline: At SemEval-2013, the baseline consisting of textual similarity measures comparing reference and student answer frequently was not outperformed.

Given the large feature space proposed so far and the lack of consensus about where to find the most useful features, we ask whether there are any regularities in the predictiveness of features across corpora. Is there a hierarchy of features that are always, sometimes or never useful across different corpora and languages? Are features from deep linguistic processing informative over and above the lexical baseline, or are they subsumed by the more shallow features? And, finally, do the optimal feature sets differ with corpus characteristics like language or elicitation task?

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2186–2195, Osaka, Japan, December 11-17 2016.

We will investigate these questions as follows: Section 2.1 defines our task and hypotheses. Section 2.2 introduces the corpora and 3 describes the features. Section 4 specifies our implementation of the experiments. We first validate our feature set against literature results in Section 5, then look at feature predictiveness individually in Section 6 and in combination in Section 7. Section 8 concludes.

2 Background: Task and Data

We look for highly predictive features for the short-answer grading task that generalise over different corpora. We also ask how much information can be drawn from more abstract features from deeper levels of linguistic processing that is not covered by the strong NGram and text similarity baselines.

2.1 Task and Hypotheses

We investigate these questions by looking at unseen-question 2-way classification of short answers. In this task, the test data contains only questions (and the corresponding answers) not seen during training, but from a similar domain as the training data. We choose this task because it makes no assumptions about pre-existing student answers for each question, which maps well to small-scale testing in many educational settings. The task is binary classification of answers as *correct* or *incorrect*.

We select data from the spectrum of available short-answer corpora (see, e.g., the excellent overview article by Burrows et al. (2015)) according to two criteria: Language and elicitation task. There are two predominant tasks: On the one hand, there are corpora that assess content mastery in specific knowledge domains. These corpora contain answers by mostly highly proficient speakers. On the other hand, there are learner corpora assessing language students' reading comprehension by asking questions about the content of a text. Answers to these questions are characterised by learner mistakes and the heavy influence of *lifting* answers from the reading text, with the result of overall less variation within answers. In order to vary language, we use one German and one English corpus for each mode (see 2.2 below).

We hypothesise that the higher levels of answer variation in content-assessment corpora as opposed to the language-skill corpora will necessitate features from deeper processing levels to uncover parallels between student and reference answer. We do not expect corpus language to have a big effect, except perhaps in the usefulness of pre-processing like lemmatisation for inflection-rich German.

2.2 Data

We use the corpora listed in Table 1. The SciEntsBank (SEB) and Beetle corpora are the SemEval-2013 corpora (Dzikovska et al., 2013), which we consider as one data set. Both contain content assessment questions from science instruction, in English. CSSAG is a set of German content assessment questions about programming in Java; the data set used here is an extension of the corpus described in Padó and Kiefer (2015), following the same design principles. CREG (Meurers et al., 2011b) and CREE (Meurers et al., 2011a) are language-skill corpora. Learners of German and English, respectively, read texts and answered questions about their content.

	#Questions/ #Answers	#Q/#A (Test Set)	Task	Language
SEB (Dzikovska et al., 2013)	135/4969	16/733	Content	English
Beetle (Dzikovska et al., 2013)	47/3941	9/819		
CSSAG (Padó and Kiefer, 2015)	31/1926	NA		German
CREG (Meurers et al., 2011b)	85/543	NA	Language	
CREE (Meurers et al., 2011a)	61/566	NA		English

Table 1: Corpus sizes and characteristics

For Beetle and SEB, ample test sets exist which we use in Section 5 to validate our full models before delving into feature analysis. For the smaller corpora, there are no separate test sets[1] and the data sets

[1]There is a designated test set for CREE, but it repeats some questions from the training set, so it is not appropriate for the unseen question task. We therefore combined development and test data for CREE.

were considered too small to create them. Following Hahn and Meurers (2012), we present results for leave-one-question-out cross-validation, where we hold out each question in turn and train models on the remaining data.

All corpora contain the question texts, at least one reference answer per question and the student answers with a human-assigned correctness judgment. All corpora have explicit *correct/incorrect* annotation, except for CSSAG. CSSAG answers are scored in half-point steps up to a maximum number of points per question (usually 1 or 2). We convert CSSAG scores into binary labels by mapping all answers with more than 50% of points to *correct* and all other answers to *incorrect*.

3 Features

We compute established literature features on five different levels of linguistic processing. In the order of processing complexity, these are NGram features, text similarity features, dependency features, abstract semantic representations in the LRS formalism (Richter and Sailer, 2004) and entailment votes from a TE system, which we treat as a black box.

In the unknown question setting, all features are computed in relation to the reference answer given for each question (the question is also considered for some features, see below). Features usually code the overlap between units (NGrams, dependency relations, etc.) in the reference and student answer. We use the reference answer as the basis, so the features express the percentage of reference answer units shared between student and reference answer. The higher the percentage, the more completely does the student answer cover the reference. If the percentage is lower, the student answer is probably incomplete. The inverse percentage can of course also be computed; where the corresponding features performed well, we include them also. Wherever there is more than one reference answer, we use the maximum overlap of all the answer options, assuming that graders will evaluate student answers according to the most similar reference answer.

Table 2 gives an overview over the feature set. In more detail, the features are:

NGram features measure the overlap in uni-, bi- and trigrams between reference and student answer. NGrams are computed on both tokens and lemmas (to raise coverage).

Similarity measures compare reference and student answer on the text level. We use Greedy String Tiling (GST, Wise (1996)), a string-based algorithm popular in plagiarism detection that deals well with insertions, deletions and re-arrangement of the text.[2] We also use the classical Cosine measure as a vector-based approach and compute the Levenshtein edit distance between the texts. The measures are run on lemmatised text before (with stop words, WSW) and after stop word filtering (SWF). Stop word filtering includes removal of words in the question (*question word demotion*, Mohler et al. (2011)). The rationale is that students should be graded on the new information they provide over and above the concepts mentioned in the question. We chose not to use similarity measures that need external resources (such as WordNet or large corpora), since they may not be equally appropriate for the different corpus domains and show inconsistent performance.

Dependency features code the overlap between the student and reference answer dependency relations in terms of lemmatised triples of governor, dependency type and dependent.

Semantics features are derived by the parsing and alignment component in CoSeC (Hahn and Meurers, 2012). It constructs LRS (Lexical Resource Semantics, Richter and Sailer (2004)) analyses of the texts and attempts to align the components. We then compute the overlap in aligned components between reference and student answers as well as the question and student answer. The motivation for the latter measure is similar to question-word demotion in that high overlap between question and answer may point to question copying with little additional content.

TE decisions are computed using the Excitement Open Platform[3] (EOP, Magnini et al. (2014)). Dzikovska et al. (2013) propose constructing the Text from question and student answer and using

[2] Minimum string length is four characters.

[3] `http://hltfbk.github.io/Excitement-Open-Platform/`

the reference answer as the Hypothesis that may or may not be entailed by the Text. For us, this led to many false-positive entailment judgments by the TE system, most likely because the longer the Text, the easier it becomes to construct relations between Text and Hypothesis. We therefore use only the student answer as the Text. Our features are the entailment decision itself and the confidence score returned by the system. If there are multiple reference answers, the student answer may well entail one, but not the others. Therefore, we record any Entailment decision and its confidence score over any Non-Entailment decision, and in the case of only Non-Entailment decisions, record the lowest confidence score to capture the judgment closest to Entailment. This means that a high score size correlates with a positive decision and a low score with a negative decision.[4]

Feature Group	Feature Names
NGram	Unigram(Token,Lemma), Bigram(Token,Lemma), Trigram(Token,Lemma)
Similarity	GST(WSW,SWF), Cosine(WSW,SWF), Levenshtein(WSW,SWF)
Dependency	SRDependency, RSDependency
Semantics	LRS-QS, LRS-RS, LRS-SR
TE	TEDecision, TEConfidence

Table 2: Overview of the feature set

4 Method

We pre-processed the corpora with the DKPro pipeline (Eckart de Castilho and Gurevych, 2014), using the OpenNLP segmenter[5], the TreeTagger for POS tags and lemmas (Schmid, 1995) and the MaltParser (Nivre, 2003) for dependency parses. All tools (including the LRS parser and the EOP TE system) are used as-is without additional evaluation and tuning on our data.

Since our goal is to gain insight into the contribution of the different feature groups, we consider only one learning algorithm and do not investigate ensemble learning (although this is a common and promising approach in the literature (Dzikovska et al., 2013)). For our small data sets, overfitting is a concern. We therefore use decision trees, namely the J48 implementation in the Weka machine learning toolkit (Hall et al., 2009), which addresses overfitting by a pruning step built into the algorithm.

We report unweighted average F1 scores for comparability with Dzikovska et al. (2013), and, for the full models, accuracy for comparison to Hahn and Meurers (2012) and Meurers et al. (2011a). Tests for significance of differences between results are carried out by stratified shuffling (Yeh, 2000). The independent observations needed for this approach are the sets of answers belonging to one question.

5 Full Models and Literature Benchmarks

As the first step, we compare the performance of the decision tree algorithm and the whole feature set to the literature benchmarks for the data sets. We show that the model and features we chose achieve realistic performance to ensure that our analyses below are meaningful.

In addition to the benchmarks, we report the frequency baseline (always assign the more frequent class) and a lexical baseline (a decision tree trained with just the UnigramToken feature)[6].

For the binary grading task, the human upper bound for accuracy (measured as agreement between the raters) is in the high eighties. For the CREE and CREG corpora, grader agreement is reported as 88% (Bailey and Meurers, 2008) and 87% (Ott et al., 2012), respectively.

Table 3 lists the unweighted average F1 scores and accuracies for the different data sets. The SEB and Beetle figures are for the held-out test sets; for the other data sets, we report leave-one-question-out cross-validation results. All models outperform the frequency baseline.

[4]We use the MaxEntClassification algorithm with settings Base+WN+TP+TPPos+TS for English and settings Base+GNPos+DBPos+TP+TPPos+TS for German.

[5]`https://opennlp.apache.org/`

[6]Note that this baseline differs from the lexical baseline used in the SemEval-2013 evaluation, where a combination of similarity measures was used. The SemEval-2013 lexical baseline is the same for SEB and F=78.8 for Beetle.

	SEB		Beetle		CSSAG		CREG		CREE	
	F	Acc	F	Acc	F	Acc	F	Acc	F	Acc
Frequency Bsl	37.1*	59.0	36.7*	58.0*	38.4*	62.4*	37.4*	52.7*	44.4*	59.5*
UnigramToken Bsl	61.8*	64.0	69.8	72.9	67.4	72.0	78.6	78.6	66.5	81.3
Full model	52.8	61.5	68.1	70.8	66.6	69.3	81.5	82	70.2	80.4
Literature	62.9	–	66.6	–	–	–	–	86.3	–	88.4

Table 3: Performance of the full feature set in comparison to baselines and literature results. * indicates a significant difference between baseline and full model.

Four of the five models do not significantly differ from the UnigramToken baseline, although two numerically outperform it. This is a familiar picture from the SemEval-2013 competition and underscores the difficulty of the task.

A notable anomaly is the SEB model, which significantly underperforms on F-score and numerically underperforms on accuracy against both baselines. The leave-one-question-out cross-validation result for this model on the training set is comparable to the Beetle test set result at an F-score of 66.0 and accuracy of 67.7. We hypothesise that the training and test set for the SEB data differ substantially.

The SEB model performance of course also does not reach the literature result (although the cross-validation result of F=66.6 is comparable), while the Beetle model even numerically outperforms the literature benchmark (we compare to the median participant performance at SemEval-2013, Dzikovska et al. (2013)). For CSSAG, no prior literature results exist. The CREG result is roughly similar to the best model to date reported in Hahn and Meurers (2012). The literature result for CREE (Meurers et al., 2011a) is not completely comparable, as it was computed on the held-out test set that does not satisfy the unseen question task. Therefore, it is not surprising that our model does somewhat worse on a purely unseen question evaluation.

Overall, with the exception of the SEB model, we have been able to verify that out feature set and learner are able to approximate state-of-the-art results. We still include the SEB data set in our analyses below since the leave-one-question-out cross-validation result is much more consistent with the other models and we hypothesise a mismatch of test and training data.

6 Performance of Individual Features

For our analysis of feature impact, we first look at the performance of each feature individually. We train a decision tree with just that feature and report unweighted average F1 scores. We present only features that outperform the frequency baseline by at least 10 points F-score. The cells in Table 4 show the difference in F-score between the single-feature and full-model performance. Features that perform numerically close to the full model (within 15 percent of the F-score) are bold-faced.

We first discuss the table from the point of view of the different feature groups. As expected, the **NGram features** are strong and approximate full-model performance consistently. The higher-order NGrams drop off against the Unigrams since they are sparser. The lemmatised NGram features were introduced to potentially overcome this problem, but they consistently do less well than the token-level features. Analysis shows that lemmatisation yields higher overlap percentages between reference and student answer, but this figure now correlates less with answer accuracy. Apparently, there are important differences between reference and student answer on the token level that are lost through lemmatisation.

The **similarity measures** are also strong across the board. Among the measures we tested, Greedy String Tiling is the best predictor of response accuracy. Further, our results support the suggestion by Okoye et al. (2013) that stop words should not be removed, but we can qualify this recommendation: For measures like Greedy String Tiling and Levenshtein that explicitly operate on word sequences, stop word removal hurts performance. For the Cosine measure, on the other hand, removal is generally beneficial because it removes spurious overlap. Levenshtein edit distance is the least predictive of the similarity measures. This fits well with the analysis in Heilmann and Madnani (2013), who also see uneven performance of the model containing their edit-distance feature.

Feature Group	Feature	SEB	Beetle	CSSAG	CREG	CREE
NGrams	UnigramToken	**-4.3**	**-1.8**	**0.8**	**-2.9**	**-3.7**
	BigramToken	**-8.2**	**-1.4**	**-5.6**	**0**	–
	TrigramToken	-16.5	**-4.8**	-20.7	**-8.7**	–
	UniLemma	-21.4	-12.6	**-3.6**	-12.3	**-9.7**
	BiLemma	-12.9	**-6.1**	–	**-10.2**	-14.4
	TriLemma	–	-15.1	–	-20.7	–
Similarity	GST-WSF	-17.1	**-9**	**1.1**	**-10.7**	**-6.2**
	CosineWSF	-12.5	**-9.9**	–	**-11.5**	–
	LevenshteinWSF	–	-16.2	**0.6**	-29.8	–
	GST-WSW	**-7.7**	**-8**	**-1.4**	**-11.8**	**–2.2**
	CosineWSW	-15.1	**-9.1**	–	-22.2	–
	LevenshteinWSW	–	-20.8	**1**	-22.8	–
Dependency	RSDependency	**-7.5**	**-7**	**-9**	**-10.5**	–
	SRDependency	-13.5	-13.1	**-4.9**	–	–
Semantics	LRS-QS	–	**-9.3**	-16.6	–	–
	LRS-RS	-13.6	**-1.6**	**-3.4**	-24.9	–
	LRS-SR	-16.2	-13.7	–	-23.8	–
TE	TEDecision	–	–	-19.1	-34.3	-15.6
	TEConfidence	-12.9	-13.3	**-4.6**	**-10.7**	**2.2**
	Full model	66.0	72.6	66.6	81.5	70.2

Table 4: Performance of individual features across all data sets ($F_{feature} - F_{full\ model}$ for all $F_{feature}$ at least 10 points F-score above the Frequency baseline).

The **dependency** features are also informative for all corpora (except for CREE). Here and in the semantic features, we again find that the RS normalisation works better than the SR normalisation, that is, specifying how much of the reference answer is covered by the student answer predicts overall accuracy better than looking at how much of the student answer is present in the reference answer. This is because the latter direction does not accurately model incomplete student answers.

The **semantic representations** are highly predictive, but only for Beetle and CSSAG, although they are within 20% F-score for SEB. The overlap between question and student answer (QS) is probably informative for Beetle because of questions that ask about specific components in an electric circuit which have to be mentioned in a correct answer. This observation calls into question the usefulness of question word demotion in all situations. Specifically in Beetle, question word demotion can be counterproductive. Take, for example, the question *Why do you think those terminals and the negative battery terminal are in the same state?* with one reference answer *Terminals 1, 2 and 3 are connected to the negative battery terminal*, and its demoted version *1, 2 3 connected to*, which matches correct answers as well as incorrect answers which speak about a connection to the positive battery terminal.

The **TE** confidence feature works well for CSSAG and outperforms the full model for CREE by two points F-score. Performance for SEB and Beetle is within 20% F-score of the full model performance. Recall that the construction of the confidence value guarantees a correlation of high confidence with an Entailment decision and lower confidence with a Non-Entailment decision, so the feature carries most of the information of the nominal TEDecision feature in addition to the graded confidence.

In sum, all features are useful, but not in all cases. As expected, there are workhorse features like NGrams and similarity that strongly predict response accuracy across the board. We find that this general applicability extends to dependency features, as well. The more abstract semantic and TE features are useful in specific cases.

To further analyse these performance patterns, we now turn to an analysis from the point of view of the corpus types. We hypothesised in Section 2.1 that there would be little influence of language, but a noticeable difference between the corpora collected with different elicitation tasks.

First of all, there is indeed no discernible difference between the German and English corpora. Even NGram lemmatisation, while helpful for CSSAG and CREG, also approximates the full models for Beetle and CREE quite well and the best token-level NGram features always outperform the lemmatised features.

We do however see a clear difference between the content assessment and language-skill assessment corpora. The language-skill corpora (CREG and CREE) profit comparatively little from the deeper processing of the dependency and LRS features; NGram and text similarity features are however very strong predictors of response accuracy. This can be explained by the typical answers in these corpora: Since students' language proficiency is limited, they often lift all or part of their answers directly from the reading. The target answers are adapted from the same texts and therefore lexically similar. Lexical and string overlap are therefore sufficient to distinguish between a correct and an incorrect answer. Then why are the TE features so strikingly successful for these corpora? This can be explained by the fact that, for CREG, and especially for CREE, the reference answers are slightly re-formulated from the reading texts by the instructor, a highly proficient speaker of the target language (frequent changes include tense and contractions, but also paraphrasing, often with POS changes for the words involved). The generalisation strategies in the TE system help find the underlying semantic similarity that is obscured on the token level.

For the content assessment corpora, in addition to the NGram and similarity features, features from deeper levels of processing are always useful. (The TE confidence is within 20% F-score for SEB and Beetle, as are the "missing" semantic and dependency features.) The deeper processing levels apparently help uncover paraphrasing by (mostly) language-proficient test-takers.

From the analysis of individual feature performance, we thus find that the NGram and similarity surface features, as well as the dependency features, are predictive for every corpus, but the features from deeper processing are useful especially for the content assessment corpora.

7 Feature Groups and Combined Performance

The discussion in Section 6 showed a clear performance pattern for the different feature groups. One possible next step would be to combine all the features that are highly predictive of response accuracy into one model. However, the features are highly inter-correlated, so their joint performance does not necessarily exceed any single performance. Recall that for CREE, the TEConfidence feature alone outperforms the full model by two points F-score. To quantify the amount of inter-correlation, an average 67% of the variation in the UnigramToken feature can be explained by a linear combination of the other feature groups (*excluding* the NGram features) across the corpora. This explains the high UnigramToken baseline - the other features strongly co-vary with the NGram features and contribute relatively little additional information.

In order to find the most predictive feature combinations, we choose an extrinsically-motivated model-building strategy. We propose adding features in the order of processing effort necessary to produce them, with the motivation that any information that can be gained by simple means should not be duplicated by more costly methods. Starting from the NGram feature set, we incrementally add more features and monitor the performance to find the cut-off point at which the complete model performance has been approximated (or even out-performed).

Table 5 shows the results. Note that the NGram feature group as a whole often outperforms the UnigramToken baseline, since the higher-order NGram features in the feature group contribute additional information. Adding the TE features in the final step results in the full model performance. The intermediary results in bold face represent substantial increases in model performance. Underlined results numerically outperform the full model.

As expected after our discussion of feature patterns in Section 6, we find that for all corpora, subsets of the feature groups suffice to approximate full model performance. In four out of five cases, we even optimise performance by using fewer features.

There are few surprises in the feature groups that contribute substantially to model performance: Again, we see a strong reliance on the NGram and similarity features. For three out of the five corpora, the full model performance can be reached or exceeded just by these two feature groups. Adding dependency features further improves four out of five models (although the CSSAG improvement is negligibly small).

Feature Group	SEB	Beetle	CSSAG	CREG	CREE
UnigramToken Bsl	61.7	70.9	67.4	78.6	66.5
NGrams	**62.1**	**72.6**	**66.5**	**80.5**	**60.9**
+ Similarity	**62.6**	72.6	**69.4**	**82.5**	**67.4**
+ Dependency	**64.7**	70.9	69.5	**83.5**	**69.8**
+ Semantics	64.7	**73.4**	66.4	83.4	**70.7**
+ TE (full model)	**66.0**	72.6	66.6	81.5	70.2

Table 5: Performance in F-score when adding feature groups in order of processing effort. Boldfaced figures indicate a substantial contribution to model performance, underlined figures exceed full model performance.

SEB, Beetle and CREE profit from features from deep processing (semantics and TE). This matches our analysis above that the more varied language in content assessment corpora (and the highly-proficient paraphrasing that creates the reference answers from the reading text for CREE) can be successfully addressed by more abstract features from deeper levels of linguistic analysis.

For the individual corpora, there is a clear correspondence between feature groups with highly predictive features in Table 4 and useful feature groups in Table 5. Any feature group containing a feature that approximates full model performance within about four points F-score proves useful in incremental model construction. Interesting exceptions to this rule are CSSAG and CREE, where the semantic and TE features (CSSAG) or just the TE features (CREE) are individually predictive within four points F-score of the full model, but do not improve combined model performance. Since these features are added last, the information they contain appears to be already covered by the combination of the other feature groups. However, the best incremental CREE model still does not outperform the model only using TEConfidence (F=72.4). The CREG and SEB models profit from adding feature groups that alone are not extremely predictive (CREG: similarity and dependency features, SEB: similarity, dependency and TE features). These feature groups clearly add relevant new information given the backbone of NGram features.

In sum, we again find that the NGram and text similarity features are very predictive of response accuracy for all corpora. This is mirrored in the literature in the SemEval-2013 performance of the CU model (Okoye et al., 2013) that focuses on these feature types, or the strong results recently presented by Sultan et al. (2016), who use lexical overlap and vector-based text similarity features. Dependency features are also worth computing, as they further improve performance for four out of five models. Features from deeper linguistic processing levels are useful if the student answers differ from the reference answer by proficient paraphrasing (as opposed to insertions, deletions and re-orderings). This is the case whenever proficient speakers answer content assessment questions (or adapt the reference answer, as for the language skill assessment in CREE).

8 Conclusions

The goal of this paper was to identify highly predictive features for the short-answer grading task. We used five corpora from the content and language-skill assessment domains to ensure that our findings would generalise and verified that our full feature set approximates literature results.

The analyses found generally applicable features in the realm of shallow (Unigram) to medium (text similarity and dependency) linguistic analysis. Features on deeper processing levels were found to co-vary substantially with the shallow features. This explains why the lexical baseline is hard to break. Deeper features (semantic representations and TE) are however useful to model the higher levels of linguistic variation in our content-assessment corpora (as opposed to the language-skill corpora). There was no language-specific pattern to feature predictiveness.

These results serve as a starting point for future research into automated short-answer grading. Depending on the corpus type at hand, our feature recommendations can be used to quickly build a well-motivated basis model to expand by further deep or shallow features, according to corpus type.

Acknowledgements

The author would like to extend thanks to Michael Hahn for providing access to the LRS parser, to Valeriya Ivanova and Verena Meyer for their work on the CSSAG corpus and processing software and to Sebastian Padó for helpful discussions.

References

Stacy Bailey and Detmar Meurers. 2008. Diagnosing meaning errors in short answers to reading comprehension questions. In *Proceedings of the 3rd Workshop on Innovative Use of NLP for Building Educational Applications (BEA-3) at ACL'08.*, pages 107–115.

Steven Burrows, Iryna Gurevych, and Benno Stein. 2015. The eras and trends of automatic short answer grading. *Int J Artif Intell Educ*, 25:60–117.

Myroslava Dzikovska, Rodney Nielsen, Chris Brew, Claudia Leacock, Danilo Giampiccolo, Luisa Bentivogli, Peter Clark, Ido Dagan, and Hoa Trang Dang. 2013. SemEval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment. In *Proceedings of SemEval-2013*, pages 263–274.

Richard Eckart de Castilho and Iryna Gurevych. 2014. A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Michael Hahn and Detmar Meurers. 2012. Evaluating the meaning of answers to reading comprehension questions: A semantics-based approach. In *The 7th Workshop on the Innovative Use of NLP for Building Educational Applications*, pages 326–336.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1).

Michael Heilmann and Nitin Madnani. 2013. ETS: Domain adaptation and stacking for short answer scoring. In *Proceedings of SemEval-2013*, pages 275–279.

Sergio Jimenez, Claudia Becerra, and Alexander Gelbukh. 2013. Softcardinality: Hierarchical text overlap for student response analysis. In *Proceedings of SemEval-2013*.

Bernardo Magnini, Roberto Zanoli, Ido Dagan, Katrin Eichler, Günter Neumann, Tae-Gil Noh, Sebastian Pado, Asher Stern, and Omer Levy. 2014. The Excitement Open Platform for textual inferences. In *Proceedings of the ACL demo session*.

Detmar Meurers, Ramon Ziai, Niels Ott, and Stacey Bailey. 2011a. Integrating parallel analysis modules to evaluate the meaning of answers to reading comprehension questions. *Special Issue on Free-text Automatic Evaluation. International Journal of Continuing Engineering Education and Life-Long Learning (IJCEELL)*, 21(4):355–369.

Detmar Meurers, Ramon Ziai, Niels Ott, and Janina Kopp. 2011b. Evaluating answers to reading comprehension questions in context: Results for german and the role of information structure. In *Proceedings of the TextInfer 2011 Workshop on Textual Entailment*, pages 1–9, Edinburgh, Scottland, UK. Association for Computational Linguistics.

Michael Mohler, Razvan Bunescu, and Rada Mihalcea. 2011. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 752–762. ACL.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *IWPT 03*, pages 149–160.

Ifeyinwa Okoye, Steven Bethard, and Tamara Sumner. 2013. CU: Computational assessment of short free text answers - a tool for evaluating students' understanding. In *Proceedings of SemEval-2013*, pages 603–607.

Niels Ott, Ramon Ziai, and Detmar Meurers. 2012. Creation and analysis of a reading comprehension exercise corpus: Towards evaluating meaning in context. In Thomas Schmidt and Kai Wörner, editors, *Multilingual Corpora and Multilingual Corpus Analysis*, Hamburg Studies in Multilingualism (HSM), pages 47–69. Benjamins, Amsterdam.

Niels Ott, Ramon Ziai, Michael Hahn, and Detmar Meurers. 2013. CoMeT: Integrating different levels of linguistic meaning assessment. In *Proceedings of SemEval-2013*, pages 608–616.

Ulrike Padó and Cornelia Kiefer. 2015. Short answer grading: When sorting helps and when it doesn't. In *Proceedings of the Nodalida-2015 workshop*.

Lakshmi Ramachandran, Jian Cheng, and Peter Foltz. 2015. Identifying patterns for short answer scoring using graph-based lexico-semantic text matching. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 97–106.

Frank Richter and Manfred Sailer. 2004. Basic concepts of lexical resource semantics. In Arnold Beckmann and Norbert Preining, editors, *European Summer School in Logic, Language and Information 2003. Course Material I, volume 5 of Collegium Logicum*, pages 87–143. Publication Series of the Kurt Gödel Society, Vienna.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT-Workshop*.

Md Arafat Sultan, Cristobal Salazar, and Tamara Sumner. 2016. Fast and easy short answer grading with high accuracy. In *Proceedings of NAACL-HLT 2016*, pages 1070–1075.

Michael J. Wise. 1996. YAP3: Improved detection of similarities in computer program and other texts. In *SIGCSEB: SIGCSE Bulletin (ACM Special Interest Group on Computer Science Education)*, pages 130–134. ACM Press.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of COLING 2000*, pages 947–953.

Torsten Zesch, Omer Levy, Iryna Gurevych, and Ido Dagan. 2013. UKP-BIU: Similarity and entailment metrics for student response analysis. In *Proceedings of SemEval-2013*.

Automatically Processing Tweets from Gang-Involved Youth: Towards Detecting Loss and Aggression

Terra Blevins
Department of Computer Science
Columbia University
New York, NY, USA
tlb2145@columbia.edu

Robert Kwiatkowski
Department of Computer Science
Columbia University
New York, NY, USA
rjk2147@columbia.edu

Jamie Macbeth
Department of Electrical and Computer Systems Engineering
Fairfield University
Fairfield, CT, USA
jmacbeth@fairfield.edu

Kathleen McKeown
Department of Computer Science
Columbia University
New York, NY, USA
kathy@cs.columbia.edu

Desmond Patton
School of Social Work
Columbia University
New York, NY, USA
dp2787@columbia.edu

Owen Rambow
Center for Computational Learning Systems
Columbia University
New York, NY, USA
rambow@ccls.columbia.edu

Abstract

Violence is a serious problems for cities like Chicago and has been exacerbated by the use of social media by gang-involved youths for taunting rival gangs. We present a corpus of tweets from a young and powerful female gang member and her communicators, which we have annotated with discourse intention, using a deep read to understand how and what triggered conversations to escalate into aggression. We use this corpus to develop a part-of-speech tagger and phrase table for the variant of English that is used, as well as a classifier for identifying tweets that express grieving and aggression.

1 Introduction

The USA has the highest rate of firearm related deaths compared to other industrialized countries. Violence is particularly prevalent in cities like Chicago, which has seen a 40% increase in firearm violence in 2015; someone is shot every 2-3 hours in the city. The Chicago Police Department claims that gang violence is exacerbated by taunting between gang members on social media. Recent studies have shown that the new "digital street" is likely to have consequences for one's lived experiences (Moule et al., 2013; Patton et al., 2013; Pyrooz et al., 2015). Gangs that are highly organized have an increased likelihood of engaging in online behaviors that may include harassing others via the web (Moule et al., 2013).

In this paper, we work with a dataset of tweets posted by a young and particularly powerful female Chicago gang member, Gakirah Barnes, and people with whom she communicated. We use the dataset to develop a system that can automatically classify tweets as expressing either *loss*, grieving the death of friends or family who were shot, or *aggression*, threatening to harm others often in retribution for a loss. Tweets that don't fall into either category are classified as *other*. The ultimate goal of our work is to alert community outreach groups when aggressive tweets are identified so that they can intervene to alleviate a potentially violent situation. We are also interested in enabling interventions when youths are traumatized before grief turns to retribution. Our team includes social workers who labeled the data with discourse tags representing the intention behind the tweet, and computer scientists who developed the classification system in close consultation with the social workers.

The language used in the tweets is quite different from Standard American English and also from language used in Twitter by other populations. Sample tweets are shown in Figure 1, illustrating the many factors that characterize the form of these tweets: the use of dialectal (African American Vernacular English or AAVE) grammar and vocabulary (Rickford, 1999; Green, 2002), gang-related slang, non-standard orthography, emojis, and abbreviated expressions. Individual words do not always mean what

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2196–2206, Osaka, Japan, December 11-17 2016.

Tweet	Label	Youth Interpretation
If We see a opp Fuck it We Gne smoke em 😈	Aggression (Threat)	he mean like if he see opp he go kill him opp mean like the people he dont like
Dnt get caught on Dat 800 block lame ass Lil niggas Betta take Dat Shyt on stony spot	Aggression (Insult)	he saying them lil nigga better not get caught on the 800 block or they go kill them so he tell them if they wanna live they better stay on stony
Young niggas still getting shot babies still dying 🙏	Loss	he mean like teen keep die and babys and kid keep die

Figure 1: Tweets and a Chicago youth's interpretation of them.

they do in Standard American English. A Chicago youth from the neighborhood helped us interpret the tweets. His interpretations of the sample tweets are shown on the right in Figure 1.

Given this non-standard language, natural language tools that are widely used in the NLP community cannot be used for our task. Out-of-vocabulary words and abbreviated informal expressions mean that part-of-speech taggers are not accurate. Some words carry different meanings than in most other contexts (e.g., *smoke* in the first tweet of Figure 1 means 'kill') and thus even online slang dictionaries such as Wiktionary do not have accurate definitions for words in this context. In fact, 56.9% of the words in our corpus which are not in WordNet (Beckwith et al., 1991) have incorrect definitions in Wiktionary. The intuitions of the computer scientists on the team about the meaning of tweets was often incorrect and thus, interaction with the social workers was critical.

Our approach to classifying tweets features three key contributions:

- A new corpus that is annotated with discourse intention based on a deep read of the corpus, as well as POS tags.[1]
- NLP resources for the sub-language used by Chicago gang members, specifically a POS tagger and a glossary.
- A system to identify the emotion conveyed by tweets, using the Dictionary of Affect in Language.

We developed a part-of-speech (POS) tagger for the gang sublanguage and applied machine translation alignment to produce a phrase table that maps the vocabulary they use to Standard English. Features for the emotion classifier included the POS tags produced by our tagger as well as the Dictionary of Affect in Language's (DAL) quantitative scores representing the affect of words. (Whissell, 2009). In order to access the correct word in the DAL for each Twitter word, we used the glossary we derived to find the standard English terms corresponding to slang. Our supervised classifier is able to recognize *loss* tweets with 62.3% f-measure and *aggression* tweets with 63.6% f-measure, improving over the baseline by 13.7 points (aggression) and 5.8 points (loss).

In the following sections, we describe our annotated corpus, the sublanguage tools we developed and the classifier.

2 Related Work

Wijeratne et al. (2015) engineered a general surveillance platform that uses commonly available sentiment analysis tools as a component, but does not process the language of social media posts based on the specific language and culture of street gangs with an aim towards detecting aggression. Others have analyzed urban gangs' social media presence using spatialized network data (Radil et al., 2010) and automated the analysis of graffiti style features (Piergallini et al., 2014) to predict gang affiliation. Research has also studied the psychological impact of crime on urban populations by analyzing social media, finding that crime exposure over a year can result in negative emotion and anxiety (Valdes et al.,

[1]The dataset is available at http://dx.doi.org/10.7916/D84F1R07 .

2015). Yet others are analyzing news reports to build a database of gun violence incidents (Pavlick and Callison-Burch, 2016).

There has been work on POS tagging for Twitter (Derczynski et al., 2013; Owoputi et al., 2013), including for other languages (Rehbein, 2013). We discuss (Owoputi et al., 2013) in detail in Section 4.1 The most closely related work is (Jørgensen et al., 2016), which studies African American Vernacular English in three genres (movie scripts, lyrics, tweets). They use a very large unlabeled corpus. In contrast, we use a small labeled corpus and investigate domain adaptation using additional data. Given the short amount of time since the publication of (Jørgensen et al., 2016), we have not been able to obtain their data or system to compare to ours, which we intend to do in the future.

Other research has used statistical approaches to automatically characterize dialect variation in Twitter across cities and to show how the geographical distribution of lexical variation changes over time (Eisenstein, 2015). There has been quite a bit of work examining other kinds of phenomena on Twitter; researchers have developed systems to analyze accommodation (Danescu-Niculescu-Mizil et al., 2011), sentiment analysis (e.g., (Agarwal et al., 2011; Rosenthal et al., 2015)) and clues to geolocation (Dredze et al., 2016).

3 Our Corpus

3.1 Data Collection, Corpus, and Qualitative Analysis

To create our corpus, we analyzed publicially available Twitter communication from Gakirah Barnes, who became a gang member in Chicago at age 13 and was killed at age 17, as well as tweets from people who communicated with her. Barnes changed her Twitter handle to @TyquanAssassin in memory of her friend Tyquan Tyler, who was killed in 2012. She subsequently swore to avenge Tyler's death and became a known gang leader with 9 killings to her name before she was in turn shot and killed at age 17. We focus on Gakirah because she was highly active on Twitter, posting over 27,000 tweets from December, 2011 until her death on April 11, 2014. Her typical content ranged from discussing friends and intimate relationships to threats and taunts towards rival gangs and grieving the loss of friends killed due to gang or police violence. To start, we used Radian6[2], a social media tracking service, to capture several thousand tweets by, mentions of, and replies to @TyquanAssassin. We then applied a deep read to 718 of these tweets sent during a 34-day period starting on March 15th, 2014, two weeks before another of Gakirah's friends, Raason "Lil B" Shaw, was killed by the Chicago police (March 29th, 2014) and ending one week after Gakirah's death (Thursday, April 17th, 2014). A deep read is a type of textual analysis in which annotators use outside knowledge such as context to interpret textual data. They identify and describe subtle details of the tweet such as moments of escalation. (Patton et al., 2016) We selected this time period because it represents two violent events and the conditions for retaliation are feasible. Figure 1 shows three tweets from this period. We subsequently included 102 tweets from January 14th to January 20th of the same year in the analysis in order to create a test set.

Modeling a social work approach to conducting research, we created an interdisciplinary research team comprised of a social work researcher and computer scientists (Ford, 2014). The social workers developed the annotation categories based on work with two 18 year old African American men, from a Chicago neighborhood with high rates of violence, who we hired as research assistants. They were asked to interpret the 718 tweets from the 34-day period described above. The research assistants were provided an Excel spread sheet with Gakirah Twitter data which listed the author of the tweet, the content (excluding images), the URL to the specific Twitter page and the date and time with which the tweet was posted. They provided their initial reactions about the tweets including: their first impressions, general tone, emotion and explanation of language. They also interpreted emojis that were connected to text when they were able to access the URL for a specific tweet.

Next, the social work team used the Chicago youth interpretations to ensure they had an accurate understanding of the culture, context, and language embedded in the tweets. They then analyzed communications from Gakirah Barnes and other Twitter users in her network. The deep read analysis we developed was based on a coding process that related external events to expressed events. As part of the

[2]http://radian6.com

coding process, the social work team developed a codebook to reflect categories found in the data using a random sample of 50 Twitter communications from Gakirah and others in her Twitter network. The research team then used the codebook to code all 820 tweets. Given the context of the case study, (i.e. gang violence, aggression, and trauma), initial codes identified content in posts that were perceived as threatening or violent. We then focused on posts identified through our coding process and interpretations from youth research assistants as threatening or violent, and asked "why was this communicated?" To achieve this goal, we developed a 6 step process to understand how and what triggered conversations to escalate into aggression, a process we have termed the *Digital Urban Violence Analysis Approach*. These six steps include analyzing: a triggering event; the context about the author; the tweet content; information derived from the conversational network; the linguistic form and tone of the tweet; and finally, the next event or turning point. During this process we acquired a deeper understanding of the context surrounding the variation in Twitter communication. For example, we learned that aggressive and threatening communication was often times preceded by posts that reflected loss or grief. A total of 26 codes were developed through open coding which provided an explanation for why a threatening or violent post was communicated on Twitter. Critical to this process was the coding meetings or "member checking" where the social work team came together with the computer scientists to discuss the validity of codes. Chicago youth called in to discuss how they interpreted posts, the social work team described how they developed codes and identified emerging themes and the computer scientists often asked specifically about the qualitative coding process to better understand why certain text was coded as aggression or threat. A fuller description of the methodology can be found in (Patton et al., 2016).

Based on the coding meetings, we then engaged in a second round of coding, or selective coding, which was used to further examine why a category existed and to collapse the 26 codes further. We noticed that the majority of our codes fit into three broad categories: 1) aggression, 2) grief and 3) other. The collapsed aggression code contained examples of insults, threats, bragging, hypervigilance and challenges with authority. The collapsed grief code included examples of distress, sadness, loneliness and death. The "other" codes contained examples of general conversations between users, discussions about women, and tweets that represented happiness. The January data (the test set) was coded by two annotators; inter-annotator agreement on the test set is $\kappa = 0.62$, which is moderate agreement.

3.2 Data Used in Computational Experiments

The dataset used for our NLP experiments contains the 820 tweets from Gakirah and people with whom she communicated as just described. This data is partitioned into a training set of 616 tweets, a development set of 102 tweets, and a test set of 102 tweets. The training and development set come from March and April of 2014, and the test set consists of tweets from January of the same year.

We manually annotated this data set for part of speech (POS) tags. One annotator tagged the dev and train sets and another annotated the dev and test sets. Inter-annotator agreement was $\kappa = 0.80$ on the dev set. There is a large amount of domain specific language which our annotators frequently were unfamiliar with as well as many tweets with a variety of words used in a manner different from Standard English. One such example is the use of the word *ass* which at times can be used as an adverb, adjective, or noun, whereas in Standard English *ass* is almost always used exclusively as a noun. An example can be found in the second tweet in Figure 1, *lame ass Lil niggas*.[3] The first annotator interpreted *ass* as an intensifying modifier to the adjective *lame* and tagged it is an adverb, while the second annotator read it as the second in a string of three adjectives modifying *niggas*. Additionally, the noun phrase *stony spot* in the same tweet is read by the first annotator as a common noun phrase (an adjective modifying a noun) whereas the second annotator interpreted it as the name of a location and as such tagged both as proper nouns. These discrepancies and others like them lead to a difficult task, involving reconciling problems that do not exist in newswire data; for example, confusion between common and proper nouns account for 20% of the inter-annotator disagreement. Experiments to train a system to automatically produce

[3]Note that this is not the "Ass Camouflage Construction" (ACC) discussed by Collins et al. (2008) and others, in which a phrase of type *your ass* acts as a pronoun. Instead, this is an instance of the following unnamed construction: "An [AAVE] construction distinct from the ACC, one not common to standard colloquial English, involves the combination of adjectives or nouns with the nouns *ass* and *behind* to form complex adjectives only usable pre-nominally" (Collins et al., 2008, p.32).

POS annotation are described in Section 4.1.

For the classification experiments, we used the collapsed categories of *aggression* and *loss*. Tweets that do not have an aggression or loss label are grouped into a miscellaneous *other* group. We experimented with using the full data containing all three labels and a subset of the data containing only aggression and loss annotations; we describe these experiments in Section 5.

4 NLP Analysis for the Language of Twitter Posts

4.1 Part-of-Speech Tagging

Part of speech (POS) tagging is used as a source of features in many NLP classification tasks. Our data, being fairly different from most Standard English corpora, necessitated the creation of a tagger specific to this domain: the Stanford POS tagger trained on newswire achieves an accuracy of only 34.8% on our dev set (Table 1), and even the CMU Tweet-specific POS tagger (Owoputi et al., 2013) achieves only 81.5% (as compared to 91.5% on the CMU test set). We therefore hand-annotated our corpus with POS tags (see Section 3.2). We used the CMU tokenization scheme and tagset with minor changes. We tokenized the raw data using the CMU "twokenizer" for tweets, and then we performed a second tokenization step that splits all unicode emojis into individual emoji symbols separated by spaces. As our corpus had more acronyms and other miscellaneous words which CMU tags uniformly as "G" for garbage, we use the context of the word in the sentence to give it an appropriate tag (such as "N"). Furthermore, our data also includes many emojis as well as emoticons. CMU's tagset only had an "E" tag for emoticons but not for emojis; we tag all emojis with "E" as well. As such, our final tagset included all 25 tags of CMU tagset, with the exception of the "G" tag, resulting in 24 tags in the tagset. These differences caused an unfair decrease in accuracy for the CMU tagger which we want to use as a baseline in fair comparison; therefore, for evaluation of the CMU tagger we created a separate evaluation corpus on which to test CMU wherein all emojis were replaced with the emoticon ":)" and all "G" tags were not counted. Our own tagger was trained and tested on the unmodified data with all emojis preserved. A similar transformation was also necessary in converting the output from the Stanford Tagger due to the PTB tagset differing from the CMU tagset. The transformation is fairly straightforward as PTB tags have more detail than CMU tags. Additionally, because there are some CMU tags specific to Twitter language such as the "#" tag for hashtags or the "L" tag for words with contractions such as *I'm*, all such tags were not included in the accuracy rating for Stanford.

For features, we use word unigram and bigram features, the predicted tags from the previous two words ("Tags In Window"), character n-grams for the target word, and miscellaneous binary character features such as whether or not there was punctuation, capitalization, etc. in the word in question. Furthermore, our tagger also leverages Brown Clusters created by CMU for the task of POS tagging tweets.

Tagger	Oct27 Test	Dev Set	Test Set
Stanford	52.2%	34.8%	26.0%
CMU	**91.5**%	81.5%	78.0%
Our Tagger	90.3%	**89.8**%	**81.5**%

Table 1: POS Tagger Accuracy Results

We train our tagger on the entirety of the Oct27 CMU dataset containing around 1800 tweets as well as our manually annotated gang tweets training data (616 tweets – see Section 3.2). In order to leverage the similarity to standard tweets we made use of domain adaptation (Daumé III, 2007). We also tried an even simpler method: by adding an additional feature corresponding to the domain of the sentence in the training data as well as the domain of the sentence to be tagged, the classifier is able to effectively give a weighting to the value added by each of the domains when tagging the other. This simple method of domain adaptation performed slightly better than the Daumé method for this tagging task. In Table 2 we can see that the CMU data without domain adapatation adds 0.8%, and our simple domain adaptation adds another 0.9%.

The results on the CMU test set, our dev set, and our test set are shown in Table 1. The differences between our tagger and the CMU tagger on the dev and test sets are statistically significant ($p < 0.0001$, McNemar's test). There is a large difference between the dev and test set accuracy among all taggers,

Tagger	Dev Set
Our Tagger	**89.8**%
- Misc Char Features	89.7%
- Word Bigrams	89.5%
- Word Unigrams	89.1%
- Domain Adaptation	88.9%
- CMU data	88.1%
- Tags In Window	88.7%
- CMU Brown Clusters	88.0%
- Char n-grams	86.9%

Table 2: POS tagger feature ablation study: we show accuracy results when each listed feature is removed

due presumably to the difference in annotators. Accuracy for all three taggers is higher on the CMU test set than on the Gang dev and test sets as well. This is likely in part due to the very specific nature of the language in our tweets.

Table 2 shows an ablation study in which we remove one feature at a time. A lower result means that this feature contributes more. Surprisingly, the single most important feature is the character n-grams, followed by the CMU Brown Clusters. Because of the similarity of much of the vocabulary used, the Brown Clusters produce a reasonable increase in accuracy similar to the increase reported for CMU's tweet tagger. The CMU Brown Clusters had a hit rate of 93% for words in our corpus, excluding URLs, hashtags, user handles and emojis. The high hit rate, coupled with the fact that these clusters were derived from Twitter data, likely contributed to the value.

4.2 Extracting a Glossary

Another challenging NLP task involved the creation of a glossary for the gang tweets. Our method involved using the machine translation software Moses (Koehn et al., 2007). We glossed about 400 of the tweets from our corpus into Standard American English, and used MGIZA++ to extract an alignment. (We did not succeed in creating a phrase table, presumably because the corpus was too small.) From the alignments that Moses generated, we created a simplified phrasebook, mapping one gang tweet word to one or more English words. This approach was most effective in translating the many acronyms and abbreviations that exist in gang tweets.

5 Predicting Aggression in Twitter Posts

We experiment with three supervised classification systems to predict which tweets are aggressive or demonstrate loss. Two of our systems are Support Vector Machines (SVM) (Cortese and Vapnik, 1995); these experiments include ternary classification on the full dataset (TCF) and binary classification on the aggression-loss subset (BCS). Our TCF experiments include two binary classifiers, which classify tweets as aggression versus other and loss versus other; all tweets not classified as aggression or loss are labeled other. We also implemented an additional model, in an attempt to improve performance on the full dataset. This system is a cascading classifier (CC), which uses two SVM models. One model is trained to identify one class containing all aggression and loss tweets and a second class containing all other tweets using a binary classifier on the full dataset. This enables automatic generation of an aggression/loss subset. The tweets selected by the first SVM are then passed to a second model. This second model is the same model as the BCS for Loss and Aggression.

We compute features for these classifiers from our Twitter data, including unigrams, predicted POS tags, and emotion scores. For unigram features, Twitter handles are mapped to a common token, and URLs are handled similarly. Emojis behave as regular words for all features.

5.1 Emotion Features

Our approach to identifying aggression and loss in tweets depends on identifying the emotion expressed. We use the Dictionary of Affect in Language (DAL) in order to obtain the emotional content of individual words. The DAL is a lexicon that maps over 8000 English words to a three dimensional score. The three dimensions of this score are pleasantness; activation, which is a measure of a word's intensity; and imagery, which is a measure of the ease with which a word can be visualized. Our system extends the DAL with WordNet in order to identify the emotional content of Standard English words in our data that do not occur in the DAL following Rosenthal and McKeown (2013). For each word that is not found in the DAL and is found in WordNet, the synonyms from the first (most common) synset are searched against the DAL. We assume that the emotion of a synonym will be similar to that of the original word. Thus, if there is a match between the synonyms and the DAL, the emotion score of the synonym is used for the original word.

💯	HUNDRED POINTS SYMBOL
🙏	PERSON WITH FOLDED HANDS
😂	FACE WITH TEARS OF JOY
😈	SMILING FACE WITH HORNS
💸	MONEY WITH WINGS

Figure 2: The five most common emojis in our dataset and their unabbreviated descriptions.

A more difficult task is to apply the DAL to the nonstandard English and Twitter-specific elements of the tweets. We assume each token that is not found in the DAL or WordNet is not a Standard English word. We considered various lexicons for "translating" these tokens to standard English. One such lexicon is the phrasebook automatically generated through machine translation (Section 4.2). We also attempted to translate the tweets by using a larger knowledge base, Wiktionary. With Wiktionary, we considered the definition of a word to be its translation. Wiktionary contains an entry for about half (47.7%) of the nonstandard words in the tweets; however, due to the obscure nature of most of these words, only 45.1% of these definitions are correct. In comparison, the MT-generated phrasebook manages to identify a comparable number (43.6%) of the nonstandard words to Wikipedia. Additionally, the phrasebook is much more accurate on the words that it manages to translate than Wiktionary - 83.2% of the translations from the phrasebook are accurate. We thus use the MT-phrasebook we derived from the training data instead of Wiktionary as a translation lexicon in our final system.

We use a similar technique to obtain an emotion score for the emojis found in many of the tweets. Emojis are Unicode symbols that depict faces, animals, objects, and many other entities (Figure 2). They have recently become very popular in online communication, replacing the older "emoticon" (a facial expression depicted by punctuation, for example :)). Emojis are often used to contribute to or clarify the emotion of the words they accompany. Additionally, a significant number (12.6%) of non-stopword tokens in our data are emojis. Since emojis play a significant role in the overall emotional content of a tweet, it is imperative that we include the emotional content of these emojis when scoring the tweets for their overall emotion. We attempt to solve this problem by using an additional lexicon for emojis, which maps these symbols to a representative English word or phrase. Our Emoji Lexicon uses abbreviated versions of the Unicode "names," or informative glosses, that describe the symbol in words. Thus, similar to the process we use to translate nonstandard words and slang, we utilize this lexicon to obtain a English "translation" of each emoji we come across.

We obtained an emotion score for each tweet using these techniques. We preprocess the data, removing the stopwords and Twitter specific features that do not add emotional content, such as URLs and Twitter handles. For each nonstandard token, we search a translation lexicon (either the MT-generated phrasebook or the Emoji Lexicon) to obtain a Standard English translation. Once a translation is obtained for a nonstandard element, it is applied to the DAL system described above to obtain an emotion score. For words whose emotion scores are found directly in the DAL or through WordNet, the translation process is skipped. Once the three-dimensional emotion score of each individual word is identified, the scores are combined to represent the overall emotion of the tweet. A number of different methods of combining the emotion scores were tested; however, the best results were obtained by using, for each

Experiment	Label	Precision	Recall	F-measure
TCF	**Aggression**	0.525	0.600	0.560
	Baseline (unigrams)	0.462	0.514	0.486
TCF	**Loss**	0.500	0.625	0.556
	Baseline (unigrams)	0.500	0.688	0.578
TCF	Average of **Aggression** and **Loss**	0.513	0.613	0.558
TCF	**Aggression** or **Loss**	0.588	0.800	0.678
CC	**Aggression**	0.471	0.923	0.623
CC	**Loss**	0.483	0.933	0.636
CC	Average of **Aggression** and **Loss**	0.477	0.928	0.630
BCS	**Aggression**	0.868	0.943	0.904
	Baseline (unigrams)	0.906	0.829	0.866
BCS	**Loss**	0.750	0.938	0.833
	Baseline (unigrams)	0.813	0.813	0.813

Table 3: Experimental Results on the test set. TCF is a Ternary Classification on the Full dataset (the three classes being **Aggression**, **Loss**, and neither). We provide separate results for our two classes of interest, as well as the macro-average for the two classes. We also give results for a binary task in which we collapse **Aggression** and **Loss** into one class ("**TCF Aggression** or **Loss**"). CC is the Cascading Classifier whose first step is an identification of **Aggression** or **Loss** (the system in line labeled "**TCF Aggression** or **Loss**"), and whose second step is a binary classification on the positively identified data points from the first step using the BCS system. We again provide separate results for our two classes of interest and the macro-average. BCS is Binary Classification on the aggression-loss Subset of the training data.

dimension, the minimum and maximum scores across all words in the tweet.

5.2 Results

We experimented with different approaches to classifying our data according to the aggression, loss, and other categories. In addition to SVMs, we experimented with a number of ML approaches, but we found SVMs to work best for this task. The results are shown in Table 3, as well as the f-scores of baseline unigram models for each experiment.[4] The results are better on the aggression-loss subset (BCS) than on the full dataset (TCF). However, the aggression-loss subset does not represent real-world data, as all tweets that are not labeled loss or aggression were removed prior to the experiment. The Cascading Classifier (CC) was thus implemented in an attempt to achieve better results on a realistic dataset. The CC performs better than the unigram baselines and the TCF models on both the aggression and loss categories, with both improvements statistically significant (using randomization) at $p = 0.023$ for aggression and $p = 0.039$ for loss. For our task, the high recall of our system is beneficial; it ensures that all the tweets that could potentially escalate to violence are recommended to the user, so that they can decide whether or not to intervene.

We also report results on the development set and demonstrate the contributions made by the POS tags and emotion scores as features over the baseline with respect to the dev set (Table 4). Note that the POS tags were separated into two features: a unigram POS language model, and a bigram model. We only show results for those single features in combination with unigrams that improve over the baseline. The last line for each experiment/label pair represents the final feature set that was used by each experiment on the test set.

All of our features have an impact on our classifiers. Emotion scores are useful for classification on the aggression/loss subset of the data and for classification of the aggression label of the full dataset. POS tags are useful for almost all experiments with the exception of classification of the aggression label on

[4]The "other" category had a precision of 0.706, a recall of 0.462, and a 0.558% f-measure with the TCF classifier.

Experiment	Label	Features	F-measure
TCF	**Aggression**	unigrams (baseline)	0.609
		unigrams, bigrams	0.674
		unigrams, POS-unigrams	0.674
		unigrams, emotion score	0.659
		unigrams, bigrams, POS-unigrams, emotion score	0.741
TCF	**Loss**	unigrams (baseline)	0.756
		unigrams, POS-bigrams	0.818
TCF	**Aggression + Loss**	unigrams (baseline)	0.727
		unigrams, bigrams	0.738
		unigrams, POS-bigrams	0.812
		unigrams, bigrams, POS-bigrams	0.821
BCS	**Aggression**	unigrams (baseline)	0.866
		unigrams, bigrams	0.884
		unigrams, emotion score	0.914
		unigrams, bigrams, emotion score	0.926
BCS	**Loss**	unigrams (baseline)	0.708
		unigrams, POS-unigrams	0.766
		unigrams, emotion score	0.723
		unigrams, POS-unigrams, emotion score	0.800

Table 4: Results on the development set and a breakdown of impact of the feature sets. The first line given for each experiment and label is the unigram baseline, and the last line is the full feature set.

the subset and of the loss label on the full dataset. Since the subset models were used as part of the CC, the features for these models are also important to our cascading classifiers.

6 Conclusion

We have presented a new corpus of tweets written by young African Americans associated with gangs in Chicago. The tweets present a challenge to natural language processing since they exhibit many features that differentiate them from Standard American English and from a representative collection of English language tweets, and since they carry complex meaning in context. We have discussed a methodology which involves a close reading of tweets in conjunction with informants, and which leads to an annotation scheme for the tweets which interprets them in the social and communicative context. We have shown that we can use POS tagging at a reasonable level if we annotate a small corpus with POS tags. We have then used this POS tagger in conjunction with a glossary to develop a system that can tag tweets as expressing two categories from the annotation scheme, namely loss and aggression, with F-measures above 60% on our test set for both categories.

The work we describe in this paper is only a first step towards our goal of creating a tool that can alert social workers to the need to intervene, with the ultimate goal of reducing gang-related violence. In future work, we will extend our corpus to include more authors, more time periods, and greater geographical variation. We also intend to further investigate how close the relationship between expressions of aggression on Twitter and real world aggression is.

7 Acknowledgments

The research described here was supported in part by the National Science Foundation (NSF) under IIS-1422863. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF.

We thank the Social Media Listening Center at Clemson University for access to the Radian6 social media tracking platform. We also thank the anonymous reviewers for their thoughtful comments.

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of Twitter data. In *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, pages 30–38, Portland, Oregon, June. Association for Computational Linguistics.

Richard Beckwith, Christiane Fellbaum, Derek Gross, and George Miller. 1991. WordNet: A lexical database organized on psycholinguistic principles. In Uri Zernik, editor, *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, pages 211–232. Erlbaum.

Chris Collins, Simanique Moody, and Paul M. Postal. 2008. An AAE camouflage construction. *Language*, 84(1):29–68.

Corinna Cortese and Vladimir Vapnik. 1995. Support vector networks. *Machine Learning*, 20:273–297.

Cristian Danescu-Niculescu-Mizil, Michael Gamon, and Susan Dumais. 2011. Mark my words! Linguistic style accommodation in social media. In *Proceedings of WWW*, pages 745–754.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June. Association for Computational Linguistics.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *RANLP*, pages 198–206.

Mark Dredze, Miles Osborne, and Prabhanjan Kambadur. 2016. Geolocation for Twitter: Timing matters. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Jacob Eisenstein. 2015. Written dialect variation in online social media. In Charles Boberg, John Nerbonne, and Dom Watt, editors, *Handbook of Dialectology*. Wiley.

Heather Ford. 2014. Big data and small: Collaborations between ethnographers and data scientists. *Big Data & Society*, 1(2).

Lisa Green. 2002. *African American English: A Linguistic Introduction*. Cambridge University Press.

Anna Jørgensen, Dirk Hovy, and Anders Søgaard. 2016. Learning a POS tagger for AAVE-like language. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1115–1120, San Diego, California, June. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard K Moule, David C Pyrooz, and Scott H Decker. 2013. From 'What the f#@% is a Facebook?'to 'Who doesn't use Facebook?': The role of criminal lifestyles in the adoption and use of the Internet. *Social Science Research*, 42(6):1411–1421.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390, Atlanta, Georgia, June. Association for Computational Linguistics.

Desmond Upton Patton, Robert D Eschmann, and Dirk A Butler. 2013. Internet banging: New trends in social media, gang violence, masculinity and hip hop. *Computers in Human Behavior*, 29(5):A54 – A59.

Desmond Upton Patton, Kathleen McKeown, Owen Rambow, and Jamie Macbeth. 2016. Using natural language processing and qualitative analysis in gang violence: A collaboration between social work researchers and data scientists. In *Proceedings of Bloomberg Data for Good Exchange*.

Ellie Pavlick and Chris Callison-Burch. 2016. The gun violence database. In *Proceedings of Bloomberg Data for Good Exchange*.

Mario Piergallini, A Seza Dogruöz, Phani Gadde, David Adamson, and Carolyn Penstein Rosé. 2014. Modeling the use of graffiti style features to signal social relations within a multi-domain learning paradigm. pages 107–115.

David C. Pyrooz, Scott H. Decker, and Richard K. Moule Jr. 2015. Criminal and routine activities in online settings: Gangs, offenders, and the internet. *Justice Quarterly*, 32(3):471–499.

Steven M Radil, Colin Flint, and George E Tita. 2010. Spatializing social networks: Using social network analysis to investigate geographies of gang rivalry, territoriality, and violence in los angeles. *Annals of the Association of American Geographers*, 100(2):307–326.

Ines Rehbein. 2013. Fine-grained POS tagging of German tweets. In *Language Processing and Knowledge in the Web*, pages 162–175. Springer.

John Russell Rickford. 1999. *African American vernacular English: Features, evolution, educational implications*. Wiley-Blackwell.

Sara Rosenthal and Kathleen McKeown. 2013. Sentiment detection of subjective phrases in social media. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 478–482, Atlanta, Georgia, June. Association for Computational Linguistics.

Sara Rosenthal, Preslav Nakov, Alan Ritter, Veselin Stoyanov, Svetlana Kiritchenko, and Saif Mohammad. 2015. Semeval-2015 task 10: Sentiment analysis in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, Denver,CO.

Jose Manuel Delgado Valdes, Jacob Eisenstein, and Munmun De Choudhury. 2015. Psychological effects of urban crime gleaned from social media. In *Proceedings of the International Conference on Web and Social Media (ICWSM)*, Menlo Park, California, May. AAAI Press.

Cynthia Whissell. 2009. Using the revised dictionary of affect in language to quantify the emotional undertones of samples of natural language. *Psychological Reports*, 105:509–521.

Sanjaya Wijeratne, Derek Doran, Amit Sheth, and Jack L Dustin. 2015. Analyzing the social media footprint of street gangs. In *Intelligence and Security Informatics (ISI), 2015 IEEE International Conference on*, pages 91–96. IEEE.

Content-based Influence Modeling for Opinion Behavior Prediction

Chengyao Chen, Zhitao Wang, Yu Lei and Wenjie Li
Department of Computing,The Hong Kong Polytechnic University
{cscchen, csztwang, csylei, cswjli}@comp.polyu.edu.hk

Abstract

Nowadays, social media has become a popular platform for companies to understand their customers. It provides valuable opportunities to gain new insights into how a person's opinion about a product is influenced by his friends. Though various approaches have been proposed to study the opinion formation problem, they all formulate opinions as the derived sentiment values either discrete or continuous without considering the semantic information. In this paper, we propose a Content-based Social Influence Model to study the implicit mechanism underlying the change of opinions. We then apply the learned model to predict users' future opinions. The advantage of the proposed model is the ability to handle the semantic information and to learn two influence components including the opinion influence of the content information and the social relation factors. In the experiments conducted on Twitter datasets, our model significantly outperforms other popular opinion formation models.

1 Introduction

Social media services, such as Twitter, Facebook, etc. provide fast and effective platforms for people to receive messages from their neighbors/friends and express their own opinions. The online communication can gradually influence one's opinions (Anagnostopoulos et al., 2008). In fact, according to a marketing survey[1], 71% of the consumers said they are more likely to make a purchase based on social media referrals. Naturally, social media offers a great chance for companies to conduct marketing by influencing the opinions of their potential customers. In order to achieve that, exploring and understanding the intrinsic mechanism of opinion formation is of great importance.

Informational influence is a primary process for forming opinions on products in social media (Das et al., 2014). It describes the following scenario: when users lack the necessary information, they will seek for the opinions of their neighbors to update their beliefs. Taking the informational influence as premise, several models are proposed with different assumptions of how a person updates her/his own opinions according to the neighbors' opinions (Clifford and Sudbury, 1973; DeGroot, 1974; Hegselmann and Krause, 2002). In these models, opinions are pre-defined as statuses through discrete categories including positive, negative and neutral opinion (Hegselmann and Krause, 2002; Galam, 2002; De et al., 2014) or continuous scales of opinion strengths (Clifford and Sudbury, 1973; DeGroot, 1974; Yildiz et al., 2011; Chazelle, 2012). However, on most social media platforms, people exchange their views by posting and replying through textual messages. The summarized opinion status simplifies the opinion formation process, and ignores the effects of semantic information hidden in the exchanged content information. Even if two messages have the same opinion category, different semantic information of the contents may result in different effects on others' opinions. We take two postings of the product "Samsung Galaxy" as examples:

(1): *"I can't post gifs on this stupid Galaxy S6."*

(2): *"Just lost my new Galaxy S6 and very sad."*

(1) expresses complaints about a problem in the usage of the product, which may lead other users to an unfavorable impression on the product. However, (2) expresses the personal sad mood for the loss and

[1] http://www.socialmediatoday.com/content/30-statistics-how-social-media-influence-purchasing-decisions-infographic

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2207–2216, Osaka, Japan, December 11-17 2016.

no bad effect on "Samsung Galaxy" is transmitted through this message. The two examples show that the summarized opinion representations are not able to differentiate the opinion influences of different expressions on other users. Therefore, it is necessary to deeply explore the user-generated content in social communication, especially to understand the actual opinion influence derived from the messages during the communication.

The problem becomes discovering the underlying relevance between a person's opinion and the received content information. The intuitive solution is to employ the co-occurrence patterns of one's opinion and her/his neighboring messages. However, as the data grows, the patterns of co-occurrences can be sparse and ineffective for prediction. Vector representations of words and phrases have been successfully applied in many Natural Language Processing tasks (Bengio et al., 2003; Le and Mikolov, 2014). Through encoding the semantic information, word embedding makes it possible to overcome the curse of dimensionality.

Therefore, we propose a Content-based Social Influence Model (CIM) based on the neural network framework which encodes the content information with word embeddings. We represent each opinion word as a dense vector in the continuous space. We then compose the opinion word vectors of one's previous message and her/his neighboring messages to form the social opinion context vector and feed the vector to a softmax layer for opinion prediction. To construct the social opinion context vector, we incorporate two social relation factors, stubbornness and interpersonal influence. Stubbornness represents the degree a user insists on her/his previous opinion and interpersonal influence represents the influence one receives from neighbors. Also, the social relation factors are polarity-related which can be either positive or negative.

Different from previous opinion formation models which only learn the opinion influence of social relationships, our proposed model learns two opinion influence components, i.e., the opinion word embeddings and the social relation factors. The learned word vectors reflect the opinion influence of different opinion words during the discussion on a specific issue, and the social relation factors including stubbornness and interpersonal influence. Integrating these two components together, our model has the capability to describe the opinion formation process more accurately. In the experiments conducted on three Twitter datasets, our proposed model performs better than other state-of-art opinion influence models. Besides, we also study the expression of users with different influence powers. The analysis could be as a reference for companies to understand the different effects of different wordings and furthermore manage their social accounts better.

The rest of paper is organized as follows. We first review the related work in Section 2. Section 3 formulates the problem and describes the framework of our proposed model. Then, the experiments and evaluation are given in Section 4. Finally, we conclude and mention potential future works in Section 5.

2 Related Work

2.1 Opinion Influence Modeling

Opinion formation is a problem firstly studied by the researchers in the sociology and statistics areas. One notable work is proposed by DeGroot (DeGroot, 1974), which takes opinions as continuous values and assumes that one updates her/his opinions by averaging neighboring opinions. Hegselmann et al., (Hegselmann and Krause, 2002) propose the Flocking model with another assumption. They assume that people are influenced by others depending on how close their opinions are. Different from these two studies which represent opinions with continuous values, voter model represents opinion as discrete category (Clifford and Sudbury, 1973). In this model, a person selects only one of her/his neighbors uniformly at random, and takes the current opinion of the neighbor as her/his own opinion. A modification is termed as the Majority voter model (Krapivsky and Redner, 2003), where the user adopts the majority opinion in his/her neighborhood. Apart from the neighboring influence, another social relation factor stubbornness is considered in opinion prediction models. It represents the degree that one insists on her/his own opinion. The DeGroot model, Flocking model and the Voter model are extended with the idea of stubbornness (Acemoglu and Ozdaglar, 2011; Yildiz et al., 2013). Recently, De et al., (De et al., 2014) propose an asynchronous linear model (AsLM) based on the DeGroot model, which first

introduces the negative influence and proves the effectiveness of the proposed model on the social media dataset. However, existing models fail to consider the effects of content information on the opinion formation problem. Our work is the first try to integrate semantic information into opinion behavior modeling.

2.2 Neural Network in NLP Tasks

Recently, neural network has received great achievements in Natural Language Processing tasks, such as language modeling (Bengio et al., 2003), machine translation (Cho et al., 2014) and sentiment classification (Tang et al., 2014). One of the most useful neural network techniques for NLP is the word embedding, which learns vector representations of words (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013). The neural language model proposed by Bengio et al., (Bengio et al., 2003) uses the concatenation of several previous words (context) as the input of the feed-forward neural network, and then the encoded context vector is used to predict the next word (target word). Following the word embedding techniques, several models are extended to achieve the phrase-level and sentence-level representations by composing all vectors of words in the phrase or sentence together. The basic composition method is using weighted average of all word vectors (Zanzotto et al., 2010; Mikolov et al., 2013). In (Mikolov et al., 2013), they use a simple data-driven approach, where phrases are formed based on the unigram and bigram counts of the words. Furthermore, considering the syntactic structure of the phrases or sentences, a method combining the words by their orders in the syntactic tree is proposed (Socher et al., 2011).

The proposed content-based social influence model bears similarities with the neural language model. In the opinion formation tasks, we regard the neighboring opinions and one's previous opinion as the "contexts", and the "target" is one's future opinion category. The model has a more complex framework since the social relation factors including stubbornness and interpersonal influence are considered with the word embeddings.

3 Approach

3.1 Problem Definition

Formally, we denote the network of users who are interested in the same issue as $G = (V, E)$, where each vertex $u \in V$ represents a user, and each edge $e \in E$ represents a following friendship between two users. The number of users is N. The neighbor set for each user $u \in V$ is denoted by $F_u = \{v|(u, v) \in E\}$, whose size is $n(u)$.

Additionally, the opinion expression behavior of a user is formulated as a triple $< p, o, t >$ which represents that a user u posts a tweet p with the opinion category o at the timestamp t. There are three values of +1, 0, -1 for opinion category o indicating the "positive", "negative" and "neutral" sentiment respectively. Given a user u, his opinion behaviors are represented as a sequence of triples: $\{< p_u(1), o_u(1), t_u(1) >, \cdots, < p_u(i), o_u(i), t_u(i)) >, \cdots, < p_u(m(u)), o_u(m(u)), t_u(m(u)) >\}$

Furthermore, given the above definitions, we define the neighboring opinion set for each user u at each timestamp $t_u(i)$ as $C_u(i) = \{p_{F_u^1}(t_1), \cdots, p_{F_u^v}(t_v), \cdots, p_{F_u^{n(u)}}(t_{n(u)})\}$, where $t_u(i-1) < t_v < t_u(i)$ for each neighbor $v \in F_u$. It includes all the information u receives from his neighbors in F_u since previous posting time $t_u(i-1)$. Considering opinion words are the most representative parts to reflect one's opinion, we only keep the opinion words within each tweet. For brevity, we rewrite the neighboring opinion set as $C_u(i) = \{C_u^1(i), \cdots, C_u^v(i), \cdots, C_u^{n(u)}(i)\}$, where $C_u^v(i) = \{C_{u,1}^v(i), \cdots, C_{u,|C_u^v(i)|}^v(i)\}$ contains all opinion words in the tweet $p_{F_u^v}(t_v)$. If there does not exist a posting from a neighbor v during the time period, $C_u^v(i)$ is an empty set. Also, we represent the tweet $p_u(i)$ with the opinion words set $S_u(i) = \{S_{u,1}(i), \cdots, S_{u,|S_u(i)|}(i)\}$.

The problem can be defined as: given the neighboring opinion information received in previous timestamp $C_u(i)$ and previous personal opinion $S_u(i-1)$, our objective is to predict the future opinion category $o_u(i)$ at the timestamp $t_u(i)$.

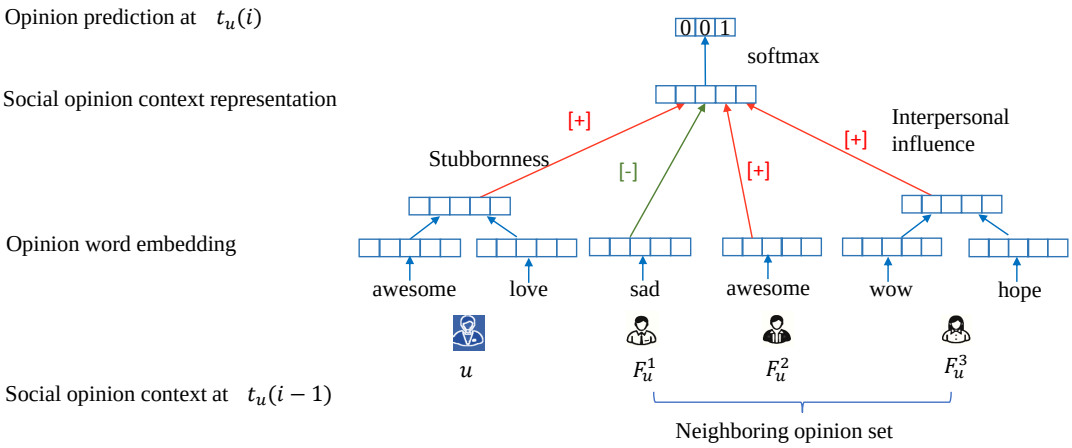


Figure 1: The graphical representation of the CIM on the opinion prediction

3.2 Framework

In this paper, we propose a novel influence model based on representation learning to solve the opinion prediction problem. Different from the existing models which learn the social relation factors including stubbornness and interpersonal influence for each user individually, our model proposes an unified framework by learning the opinion influence of the content information and the influence among social relationships together. We represent each opinion word as a dense vector, and present the composition method for the formation of social opinion context vector by concerning the polarity-related social relation factors (Section 3.2.1). Afterwards, the social opinion context vector is then used to predict one's opinion category (Section 3.2.2). Finally, we present how to learn the proposed model (Section 3.2.3). The graphical description for our proposed model is in Figure 1.

3.2.1 Social Opinion Context Composition with Polarity-related Influence

In this work, we represent each opinion word w as a low-dimensional continuous and real-valued vector $\Phi(w)$, with the dimension d. To obtain the representation of the opinions from u's vth friend, we sum the vectors of all opinion words in the set $C_u^v(i)$, and represent it as $\Phi(C_u^v(i))$. Given all neighboring opinion representations, the social opinion context vector $c_u(i)$ could be obtained by combining them together. Traditional composition methods form the phrase vector by combining word vectors with the weights obtained from the data, or applying the matrix transformation to the concatenation of word vectors (Le and Mikolov, 2014). In this work, we propose a composition method utilizing two social relation factors that have been commonly considered in previous influence models (Das et al., 2014; De et al., 2014). The social relation factors are used to describe the influence among users on the network. The stubbornness factor describes how much a person insists on her/his previous opinion, and the interpersonal influence represents the strength a neighbor has to change one's opinion. Because the interpersonal influence has the linear property (De et al., 2014), our method averages all the word vectors in the neighboring opinion set $C_u(i)$ and one's own previous opinion set $S_u(i-1)$ with the social relation factors. Formally, it is denoted as follows:

$$c_u(i) = \sum_{v=1}^{n(u)} tanh(\alpha_{uv})\Phi(C_u^v(i)) + tanh(\alpha_{u0})\Phi(S_u(i-1)) \tag{1}$$

where $\Phi(S_u(i-1)) = \sum_k^{|S_u(i-1)|} \Phi(S_{u,k}(i-1))$. α_{uv} represents the interpersonal influence on user u's opinion from the vth neighbor, and α_{u0} represents u's stubbornness. The two social relation factors are limited between -1 and 1 by using tangent function in Eq (2), which allows both positive and negative influence.

$$tanh(\alpha_{uv}) = \frac{e^{\alpha_{uv}} - e^{-\alpha_{uv}}}{e^{\alpha_{uv}} + e^{-\alpha_{uv}}} \tag{2}$$

The idea of polarity-related influence was firstly proposed by (De et al., 2014), and was proved quite effectively for opinion prediction on social network. The positive influence happens when a user trusts her/his friend, s/he will accept the opinion of her/his friend and express the same one. The negative influence implies that a user gets influenced by her/his friend, but to the opposite direction.

3.2.2 Opinion Prediction

Finally, social opinion context vector could be taken as the features to predict the future opinion category in the output layer. The output layer of our approach is expressed by the following equations.

$$P(o_u(i)|c_u(i)) = softmax(Vc_u(i) + b) \tag{3}$$

The softmax function represents the probability of current vector belonging to the jth class.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{4}$$

where $V \in \mathbb{R}^{K \times d}$, and $b \in \mathbb{R}^K$. K is the number of opinion categories, and it is set 3 in our model.

3.2.3 Learning

The model is parameterized by the social relation factors α, the word representation $\Phi(w)$ for each opinion word, and the output parameters V, b. The objective function we need to maximize is the log-likelihood of all opinion behavior sequences defined in Eq (5).

$$\mathcal{L}(O) = \sum_{u=1}^{N} \sum_{i=1}^{m(u)} \log P(o_u(i)|C_u(i), S_u(i-1)) \tag{5}$$

We learn the model using the stochastic gradient decent (SGD) algorithm. The dimensionality of the word embedding d is set as 30. During the training phrase, we normalize the gradients if the norm exceeds 1 (Pascanu et al., 2013). The training phrase stops when the training error has a decrease less than 1 or reaches the maximum iteration length of 100. The model is implemented by Theano library (Bastien et al., 2012).

4 Experiment

4.1 Data Collection

We select three well-known electronic products widely discussed on Twitter for the purpose of performance evaluation. They are "Samsung Galaxy", "Xbox" and "PlayStation". For each product, we collect all the tweets containing the product name, such as "Samsung Galaxy", published from 1st March, 2014 to 30th November, 2014 by using the Twitter streaming API[2]. We remove the inactive users with less than 30 tweets and the over active users with more than 1000 tweets. We also collect the following relationships among the users, and further construct the user network for each individual product.

Table 1 summarizes the statistics of the datasets. The "# of users" and the "# of avg friends" describe the size of the network. During each communication round, not all of a user's friends provide the suggestions, and the friends who actually post tweets and influence the user's future opinion are the active friends. Each communication round starts after a user posts a tweet, and ends when the user updates her/his opinion with a new tweet. Therefore, we define the average number of active friends by "# of avg active friends". The active level is denoted as $("\#ofavgactivefriends")/("\#ofavgfriends")$. It implies the interests of the users on the discussion of a product. From the statistics, we observe that the products "Samsung Galaxy" and "Xbox" are actively discussed by the users with the 39%, 37% active level respectively. However, the communication on the topic "PlayStation" is not as frequent as the communication on the other two topics.

[2]https://dev.twitter.com/streaming/overview

Table 1: Network statistics.

Topic	Samsung Galaxy	Xbox	PlayStation
# of users	8921	4358	5158
# of avg friends	14.42	9.58	11.83
# of avg active friends	5.65	3.58	3.33
active level	**0.39**	**0.37**	**0.28**

4.2 Opinion Processing

Many approaches has been proposed to analyze the sentiment from the text (Hu and Liu, 2004; Pang and Lee, 2008; Mukherjee et al., 2012). However, all these methods fail to explore the reason why people express or change their opinions. In our work, we take the sentiment of tweets as premise and discover the social influence during the communication. The Vader method recently proposed by (Hutto and Gilbert, 2014) has been proved better than typical state-of-art benchmarks on analyzing the sentiment of tweets with 96% accuracy on the Twitter dataset. With the constructed twitter-specific sentiment lexicon, Vader method considers the grammatical and syntactical rules to access the sentiment scores of tweets. We utilize the Vader method to score the sentiment of each tweet and to tag the sentiment category. The tweet with positive sentiment score is tagged as positive, the one with negative score is tagged as negative, and the one with zero score is tagged as neutral.

Additionally, we obtain all the opinion words with the following rules. For each tweet, all the opinions words included in the twitter-specific sentiment lexicon (Hutto and Gilbert, 2014) are extracted. If an opinion word follows a negation word, we retain the phrase "not"+"opinion word" instead of the original opinion word. For example, the opinion word extracted from the tweet "I don't like the Samsung Galaxy S6." is the phrase "not like". For the tweets only stating the facts without expressing an opinion, we use the word symbol "NeuW" to represent them. To alleviate the word sparsity, we only keep the opinion words that occur more than 50 times in the whole dataset and replace the infrequent opinion words with the corresponding symbols. The positive opinion words are replaced with the symbol "PosW", and the negative opinion words are replaced with the symbol "NegW". Finally, the numbers of the remaining opinion words for the topic "Samsung Galaxy", "Xbox", and "PlayStation" are 880, 1146 and 505, respectively.

4.3 Experimental Set-up

We compare the proposed model CIM with four baseline models, i.e., the DeGroot model, the Flocking model, the Voter model and the AsLM model. These models have different assumptions for the opinion formation process. To be fair, all baseline models incorporate the factor of personal stubbornness. It means that all models take the influence from one's previous opinion into account. For the DeGroot model (Acemoglu and Ozdaglar, 2011), the Flocking model (Hegselmann and Krause, 2002) and the AsLM model (De et al., 2014), each tweet is represented as a continuous sentiment score. For the Voter model with the assumption of the majority adoption (Krapivsky and Redner, 2003), each tweet is summarized by its opinion category. To further verify the effectiveness of the content information, we develop another influence model Content_SVM which is implemented with LIBSVM (Chang and Lin, 2011). The model trains SVM classifiers individually for each user by taking all the neighboring opinion words and the opinion words in one's previous tweet as features. To be consistent with the linear influence assumption, the linear kernel is used in SVM training process. The parameters of each model are set for their best performances experimentally.

We split the data into the training dataset and test dataset according to the posting time. The training dataset is constructed by using the data before the $m(u)-1$ timestamp for each user u. With the influence model learned from the training set, we predict the last opinion for each user.

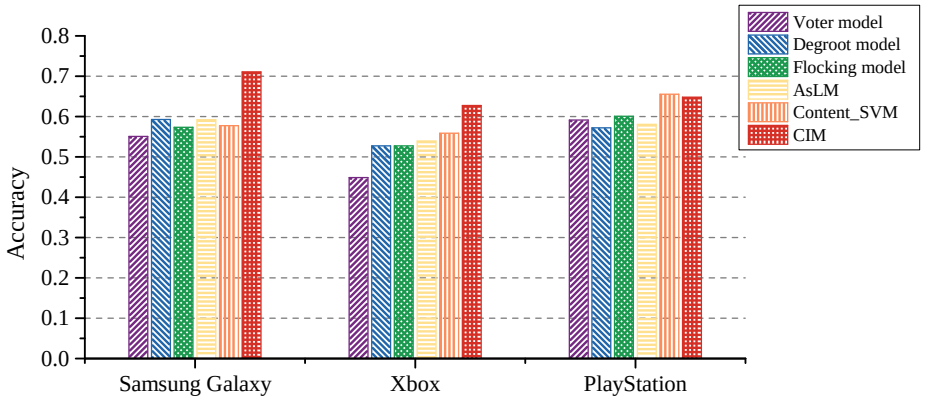


Figure 2: Performances on opinion prediction

4.4 Opinion Prediction Performances

We first evaluate the prediction accuracy for all the models. The results are displayed in Figure 2.

$$Accuracy = \frac{the\ number\ of\ correctly\ predicted\ users}{the\ number\ of\ all\ users}$$

The content-based models (Content_SVM and CIM) almost outperform the baseline methods in all three topics, which verifies that employing the detailed content information is more effective than only using the opinion statuses for opinion behavior prediction.

Meanwhile, CIM performs consistently much better than all baseline methods on the topics of "Samsung Galaxy", and "Xbox". Compared with other methods which only learns the social relation factors from opinion behaviors for each user individually, CIM encodes the semantic information into the dense vectors of the opinion words through learning from the opinion behaviors of all users. The good performance of CIM demonstrates its better ability to capture two types of opinion influence components including opinion influence of the opinion words and social relation factors together. However, CIM has a slightly lower accuracy compared with the best competitor on the topic "PlayStation". It can be attributed to the lower active level of users on the PlayStation than those on the other two topics. The insufficient communication histories over the network make it difficult to learn the actual influence of opinion words for opinion prediction, and may even harm the results.

4.5 The Effect on Opinion Category

For a more detailed analysis, we further evaluate the ability of CIM on predicting different opinion categories. We present the distributions of three opinion categories in both the training dataset and the test dataset in Table 2. The F1 score which considers both precision and recall, is used as the measurement on each opinion category. The experimental results are included in Table 3.

On the topics of "Samsung Galaxy" and "Xbox" with the active communication environment, CIM still has a significant improvement concerned with the evaluation metrics on all the three opinion categories. Specifically, the improvements compared with the best competitors on the positive opinion prediction and the negative opinion prediction are 17.7%, 21.5% for the topic "Samsung Galaxy" and 11.5%, 20.3% for the topic "Xbox" respectively. Compared with predicting the neutral opinions, forecasting the positive and negative opinions is more useful for companies to understand the customer needs and the brand reflection.

On the topic "PlayStation" with the relatively inactive communication, best performances of different evaluation metrics are obtained by different models. CIM performs well on the prediction of the positive and neutral opinions but poorly on the prediction of negative opinions. It reveals that the weakness of CIM is mainly on learning the negative opinion formation process when the communication is insufficient. We also note that the Voter model which performs poorly on the other two topics has better results on the "PlayStation" topic. Different from influence models based on the interpersonal influence,

Table 2: Opinion category statistics.

Topic	Samsung Galaxy		Xbox		PlayStation	
	Training set	Test set	Training set	Test set	Training set	Test set
% of negative opinion	11.05	14.61	16.33	6.81	11.99	19.73
% of positive opinion	19.96	19.95	41.56	26.88	25.03	19.28
% of neutral opinion	65.43	65.41	42.11	66.31	62.98	60.99

Table 3: Performances on three opinion categories.

Topic	Samsung Galaxy			Xbox			PlayStation		
	F1_Pos	F1_Neg	F1_Neu	F1_Pos	F1_Neg	F1_Neu	F1_Pos	F1_Neg	F1_Neu
Degroot	0.4950	0.1932	0.6913	0.5185	0.1935	0.6035	0.2531	0.1405	0.7064
Flocking	0.4449	0.2677	0.6780	0.4469	0.2069	0.6240	0.3513	**0.3711**	0.7125
AsLM	0.5812	0.2139	0.7028	0.5597	0.2298	0.6293	0.3210	0.2025	0.7338
Voter	0.4826	0.1762	0.6246	0.4637	0.1694	0.4709	**0.5655**	0.2688	0.6782
Content_SVM	0.4918	0.1436	0.6732	0.5972	0.2004	0.6106	0.5616	0.3410	0.7458
CIM	**0.6842**	**0.3253**	**0.7787**	**0.6658**	**0.2765**	**0.6677**	0.5568	0.1521	**0.7518**

the Voter model assumes that one will accept the mainstream view of her/his neighbors as the future opinion. The results indicate that when neighboring messages are insufficient, the group influence of all neighbors dominates. It motivates us to utilize the group influence with the interpersonal influence together for benefiting the opinion behavior prediction in the insufficient communication situation.

4.6 Analysis of Wording for Influential Users

With the learned model, the companies could get the insights into how to become an influential voice on the social media by improving their wordings. We analyze different expressions used by users with different social opinion influence degrees in the network. Based on the learned interpersonal influences, we calculate the influence strengths of Twitter users by averaging their outgoing influence strengths on their followers. Based on the influence strengths, we divide users into three groups. The users with influence strengths more than 0.5 are categorized as the positively influential users. The users with influence strengths less than -0.5 represent the negatively influential users. The remaining are regarded as the ordinary users with little influence.

We then extract the high frequent words from the users in different influence groups. The results show that the positively influential users more likely utilize the words describing the facts, e.g., ”security”, ”special” and impress”. However, the tweets posted by strong negative influential users are more emotional with the words like ”Woo”, ”Wow” or the emoticons ”o_o”. The analysis indicates that the detailed information about the products tends to make positive effects, while heavily emotional expressions may annoy people and influence them in the opposite direction.

5 Conclusions

In this paper, we propose to characterize the users' tweets with detailed opinion content instead of discrete opinion categories or continuous scores. To the best of our knowledge, this is the first attempt to incorporate the content information into opinion behavior modeling. Existing models only learn the social relation factors from the pre-defined opinion sequences. Differently, our proposed model based on the feed-forward neural network framework is capable of learning the opinion word representations which encodes the actual influence of the opinions words, and learn the two social relation factors from the opinion behaviors of all users. The experiments conducted on the Twitter dataset demonstrate the effectiveness of our proposed model on the opinion prediction. We also examine the expressions of users with different influence degrees, which could provide useful information for companies to manage their accounts. Based on the current work, we will further combine more influencing factors including the personal interests and group influence in the future model.

Acknowledgments

The work described in this paper was supported by Research Grants Council of Hong Kong (PolyU 5202/12E, PolyU 152094/14E), National Natural Science Foundation of China (61272291 and 61672445) and The Hong Kong Polytechnic University (4-BCB5, B-Q46C and G-YBJP).

References

Acemoglu, D. and Ozdaglar, A. (2011). Opinion dynamics and learning in social networks. *Dynamic Games and Applications*, 1(1):3–49.

Anagnostopoulos, A., Kumar, R., and Mahdian, M. (2008). Influence and correlation in social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 7–15. ACM.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D., and Bengio, Y. (2012). Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chazelle, B. (2012). Natural algorithms and influence systems. *Communications of the ACM*, 55(12):101–110.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Clifford, P. and Sudbury, A. (1973). A model for spatial conflict. *Biometrika*, 60(3):581–588.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Das, A., Gollapudi, S., and Munagala, K. (2014). Modeling opinion dynamics in social networks. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 403–412. ACM.

De, A., Bhattacharya, S., Bhattacharya, P., Ganguly, N., and Chakrabarti, S. (2014). Learning a linear influence model from transient opinion dynamics. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 401–410. ACM.

DeGroot, M. H. (1974). Reaching a consensus. *Journal of the American Statistical Association*, 69(345):118–121.

Galam, S. (2002). Minority opinion spreading in random geometry. *The European Physical Journal B-Condensed Matter and Complex Systems*, 25(4):403–406.

Hegselmann, R. and Krause, U. (2002). Opinion dynamics and bounded confidence models, analysis, and simulation. *Journal of Artificial Societies and Social Simulation*, 5(3).

Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Hutto, C. and Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International AAAI Conference on Weblogs and Social Media*.

Krapivsky, P. and Redner, S. (2003). Dynamics of majority rule in two-state interacting spin systems. *Physical Review Letters*, 90(23):238701.

Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mukherjee, S., Bhattacharyya, P., et al. (2012). Sentiment analysis in twitter with lightweight discourse analysis. In *COLING*, pages 1847–1864.

Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. (2013). How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Socher, R., Lin, C. C., Manning, C., and Ng, A. Y. (2011). Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136.

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., and Qin, B. (2014). Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1555–1565.

Yildiz, E., Acemoglu, D., Ozdaglar, A. E., Saberi, A., and Scaglione, A. (2011). Discrete opinion dynamics with stubborn agents. *Available at SSRN 1744113*.

Yildiz, E., Ozdaglar, A., Acemoglu, D., Saberi, A., and Scaglione, A. (2013). Binary opinion dynamics with stubborn agents. *ACM Transactions on Economics and Computation*, 1(4):19.

Zanzotto, F. M., Korkontzelos, I., Fallucchi, F., and Manandhar, S. (2010). Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1263–1271. Association for Computational Linguistics.

Data-driven learning of symbolic constraints for a log-linear model in a phonological setting

Gabriel Doyle
Department of Psychology
Stanford University
Stanford, CA 94305
gdoyle@stanford.edu

Roger Levy
Department of Brain and Cognitive Sciences
Massachusetts Institute of Techonology
Cambridge, MA 02139
rplevy@mit.edu

Abstract

We propose a non-parametric Bayesian model for learning and weighting symbolically-defined constraints to populate a log-linear model. The model jointly infers a vector of binary constraint values for each candidate output and likely definitions for these constraints, combining observations of the output classes with a (potentially infinite) grammar over potential constraint definitions. We present results on a small morphophonological system, English regular plurals, as a test case. The inferred constraints, based on a grammar of articulatory features, perform as well as theoretically-defined constraints on both observed and novel forms of English regular plurals. The learned constraint values and definitions also closely resemble standard constraints defined within phonological theory.

1 Introduction

Constraint-based models of language, often in the form of "maximum entropy" or "log-linear" models, are prominent in many applications and theoretical analyses in computational linguistics and psycholinguistics, including in text segmentation (Beeferman et al., 1999; Poon et al., 2009), machine translation (Och and Ney, 2002), syntactic alternation choice (Bresnan et al., 2007), and phonology (Goldwater and Johnson, 2003). Building successful models – and learning about human behavior from them – relies on the ability to identify relevant constraints, and this can be a difficult problem.

In this paper, we propose a system for learning both the values of and symbolic definitions for such constraints. We present a framework that combines observed data about linguistic outcomes with a flexible probabilistic context-free grammar of constraint structure to jointly infer (binary) feature values for multiple constraints and likely symbolic definitions for those constraints. We ground the model in a morphophonological setting, using the model to infer what phonological constraints affect the output form of regular English plurals, although it can be applied to other problems for which a constraint grammar can be defined.

The inference procedure moves beyond existing methods for learning *extensional* definitions of constraint values (Griffiths and Ghahramani, 2005; Görür et al., 2006; Doyle et al., 2014) from observational data to incorporate top-down information about likely *intensional* constraint definitions, improving both the applicability of the constraints and the theoretical basis for their values. We show that learning the constraints through this model performs as well as using pre-specified phonologically-standard constraints in explaining both observed and novel regular plural morphophonology. In addition, the structure of the learned constraints is similar to standard phonological constraints, showing that the model can be useful in both applications and theory-building.

2 Constraint-based models and the phonological test case

Our core problem is how to learn appropriate identities and weights for log-linear features in linguistic applications. In general, we assume some set of input types $\{x_i\}$, with $n_{i\cdot}$ instances of each type observed. The input type x_i is observed to produce n_{ij} instances of each outcome type y_j, and, as we are using a log-linear model, we assume that the number of observed input-output pairs (x_i, y_j) is proportional to the exponential of the weighted sum of the constraint values v_{ijk} over all constraints k. At least

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2217–2226, Osaka, Japan, December 11-17 2016.

some subset of these constraints k are unknown, and our goal is to learn the number of and values for these unknown constraints, as well as weights for both the known and unknown features.

Furthermore, we assume that the values of the unknown constraints are based on definitions that are generated from a symbolic grammar. This allows the model to inject theory- or observation-based structure into the learning process, improving the plausibility of the constraint values and allowing the researcher to identify likely definitions for the constraints to apply to unobserved inputs. At present, we limit ourselves to the case where the unknown constraints are binary and depend only on the outcome type y_j, a simplified case that is particularly relevant to phonological constraint acquisition. We discuss avenues for relaxing the binarity limitation in Section 7.

Gaps in log-linear phonological modeling We consider phonological theory as a test case because it has a well-established constraint-based framework, Optimality Theory (OT; Prince and Smolensky (1993)). But there is a gap in learning methods for OT-style phonology. Multiple methods have been proposed within OT for learning constraint weights or orderings (Tesar and Smolensky, 2000; Boersma and Hayes, 2001; Goldwater and Johnson, 2003) when the constraint definitions are known. None of these can learn constraint definitions, though three general tracks of research have pushed toward this goal. One track builds phonetically-grounded constraints based on the difficulty of producing or understanding the sound sequences (Hayes, 1999), but cannot produce constraints that lack such grounded motivations (Hayes, 1995). A second track learns constraints within a phonotactic problem, looking solely at attested output forms (Hayes and Wilson, 2008; Berent et al., 2012), but the phonotactic learning problem does not take input forms into account, and searches over a finite constraint set (instead of an infinite grammar). A third track uses data-driven learning to infer constraints (Doyle et al., 2014), but this method only learns which words violate a given constraint, and not a symbolic or intensional definition to apply it to novel words.

We propose a model to fill the gaps between these research tracks, by inferring constraints: 1) in the absence of articulatory motivation, 2) in the presence of input forms, and 3) with explicit, symbolic constraint definitions. The model uses a simple (but infinite) grammar of constraints to jointly learn a matrix of constraint violations, likely definitions for the constraints, and relative weights on the constraints that adequately explain the observed phonological forms.

Phonological constraints and log-linearity Traditional versions of OT do not employ log-linearity, so we work with the MaxEnt OT (MEOT; Goldwater and Johnson (2003)) framework, an extension that connects constraint-based phonology to the general class of log-linear models. (Traditional, non-log-linear, OT is approximated as the difference between weights on the MEOT constraints grow.) Some existing work on phonotactic and phonological constraint learning (Hayes and Wilson, 2008; Doyle et al., 2014) has been based in such a log-linear framework.

As with all OT frameworks, the core structure supposes that phonological forms are produced by starting with an input form, generating a set of output candidates, determining what constraints each candidate input-output pair violates, and selecting an output form based on the number and strength of the candidates' constraint violations. There are two types of constraints: those that depend on both the input and output ("faithfulness"), and those that depend only on the output ("markedness"). Each constraint has an associated weight, which is always non-positive; no constraint violations can make an output form more likely to be chosen. MEOT is a log-linear model, so summing the weights of all violated constraints provides each candidate's linear predictor, which is logit-transformed to a probability.

In terms of the general framework from the start of this section, faithfulness constraints are known, while the markedness constraints and weights for both constraint types are unknown.[1] In addition, we assume that the definitions for the markedness constraints are generated by a PCFG over phonological features of the sounds of the output candidates. Our specific grammar is discussed in Sect. 5.2.

[1]We limit ourselves to the learning of markedness constraints in this paper, as faithfulness constraints appear to be less arbitrary than markedness constraints (McCarthy, 2008), and may be representable as part of the output candidate generation process (Riggle, 2009).

3 Model structure

We represent the constraints as two matrices: F, the observed faithfulness constraints, which depend on both input and output forms; and M, the unobserved markedness constraints, which depend only on the output form. Each cell of F and M tells the number of violations of a constraint by a given input-output mapping. F_{ijk} is the number of violations of faithfulness constraint k by input-output pair type (x_i, y_j); M_{jl} is the number of violations of markedness constraint l by output candidate y_j. For each input x_i, some subset of the output forms $\{y_j\}$ are possible; this subset will be denoted $\mathcal{Y}(x_i)$. The weight vector w provides weights for both F and M, and is unobserved.

M is a non-parametric binary matrix with a known number of rows (candidates) but an unknown number of columns (constraints). Each column $M_{\cdot l}$ of the matrix M, which we will refer to as a "violation profile", is a binary vector of length J, the number of output candidates, specifying whether the candidate y_j violates this constraint. w is a vector of real numbers; within OT, weights are strictly negative, so we draw from $-\exp(\eta_w)$.

Previous work on constraint learning (Doyle et al., 2014) generated M through an Indian Buffet Process (Griffiths and Ghahramani, 2005), with the number of constraints L generated by a Poisson prior (with parameter α) and the violation profiles generated by a rich-get-richer scheme. In the present work, we retain the Poisson prior over L, but we want the violation profiles to be derived from symbolic constraint definitions d instead. The definitions are built from the underlying grammar G and specify whether each candidate y_j violates d. Within our model, we assume that a candidate y_j can be an exception to the definition d (switching a one to zero or vice versa in M_{jl}), and the number of exceptions is drawn from a exponential prior (Rational Rules framework; Goodman et al. (2008)). Thus, given a constraint definition d_l, the probability of it producing a violation profile $M_{\cdot l}$ is given by

$$p(M_{\cdot l}|d) \propto \exp(-bQ(M_{\cdot l}; y_\cdot, d_l)) \tag{1}$$

where $Q(M_{\cdot l}; y_\cdot, d_l)$ is the number of exceptions in $M_{\cdot l}$ given candidates $\{y_j\}$ and definition d_l, and b is the exception parameter, with larger b penalizing exceptions more strongly. As neither the true violation matrix M nor the true constraint definitions d are observed, we estimate the probability of a violation profile $M_{\cdot l}$ by marginalizing over possible constraint definitions (see Sect. 4.1).

The probability of whole observed corpus Y is the product of the probabilities across all observed input-output pairs:

$$p(Y|M, F, W) \propto \prod_i \frac{\left(\exp\left(\sum_{jk} w_{Fk} F_{ijk} + \sum_{jl} w_{Ml} M_{jl}\right)\right)^{n_{ij}}}{\left(\sum_{y_z \in \mathcal{Y}(x_i)} \exp\left(\sum_k w_{Fk} F_{izk} + \sum_l w_{Ml} M_{zl}\right)\right)^{n_{i\cdot}}} \tag{2}$$

In summary: F is observed, $w \sim -\exp(\eta_w)$, $L \sim \text{Poiss}(\alpha)$, $M_{\cdot l} \sim \exp(-bQ(M_{\cdot l}; y_\cdot, d_l))$, and $d_l \sim \text{PCFG}(G)$. We infer likely constraint matrices and weights M and w from their joint posterior distribution, which is proportional to the product of the probabilities of the data (Eqn. 2), constraints M, and weights w:

$$p(M, w|Y, F, \alpha, b, \eta_w, G) \propto p(Y|M, F, w)p(M|b, G)p(w|\eta_w) \tag{3}$$

4 Model Inference

For the model to find appropriate constraint structures, we use Markov Chain Monte Carlo (MCMC) inference over the space of constraint definitions d, markedness matrices M, and weight vectors w.

4.1 Inference over d

Inference on M requires knowledge of the prior over violation profiles (i.e., columns of M), but this is a sum over the infinite set of constraint definitions. To estimate this, we use importance sampling over constraint definitions d. For a given profile m, we start by drawing a constraint definition d from the PCFG, then Metropolis-Hastings sample through the space of constraint definitions, with three possible

move types of equal probability: subtree replacement, incision, and excision. In terms of the constraint definitions in Sect. 5.2, replacement changes a feature value, a phoneme, or a phoneme sequence; excision removes a feature, phoneme or phoneme sequence; and insertion adds a feature, phoneme, or phoneme sequence.

Subtree replacement The first move, subtree replacement, comes from Goodman et al. (2008). Subtree replacement chooses a non-terminal node uniformly randomly in the tree, and re-draws all of its children according to the PCFG probabilities. If a subtree replacement is to be made, the jump probability of moving from tree T to T' by redrawing the subtree S_X at node X is:

$$J_R(T';T) = \frac{1}{N_R} \cdot \prod_{r \in S'_X} p(r), \tag{4}$$

where N_R is the number of non-terminal nodes in T, S'_X is the new subtree with root X, and r ranges over the rules triggered by S'_X.

Node excision The second move is node excision, which promotes a subtree one level up in the tree, eliminating its parent node and sibling subtree. It selects a node X uniformly randomly from the set of nodes that can be excised (nodes with at least one grandchild Z that is also a valid child of X under the CFG). If no excisable nodes exist in the tree, the model attempts a different move type (replacement or insertion) instead. Excision removes a node Y – the child of X and parent of Z – from the tree, as well as the current sibling of Z (with its subtree). If an excision is to be made, the jump probability of choosing to excise between X and Z in tree T to yield tree T' is:

$$J_E(T';T) = \frac{1}{N_E} \cdot \frac{1}{N_{E;X}}, \tag{5}$$

where N_E is the number of nodes in T that have at least one excisable grandchild, and $N_{E;X}$ is the number of excisable grandchildren of X in T.

Node insertion The third move, node insertion, reverses node excision. A new node is inserted between a parent and child node, and the child node gets a new sibling subtree. Node selection for insertion works similarly to excision; a node is drawn uniformly randomly from the set of insertable nodes, those that have at least one child that could also be its grandchild. As with excision, if no insertable nodes exist, a different move type is attempted. Once an insertable node X is chosen, the model chooses a child node Z uniformly among its children that could be a grandchild of X. That node becomes a grandchild of X, and the model draws a new node Y from the PCFG, such that Y is a valid child of X, parent of Z, and sibling of the remaining child node of X (call this A). Finally, Z draws a new sibling B in its new lower position, according to the PCFG. Given that an insertion is to be made to the tree T, the probability of that insertion being node Y between X and Z is:[2]

$$J_I(T';T) = \frac{1}{N_I} \cdot \frac{1}{N_{I;X}} \cdot \frac{p(X \to AY)}{p(X \to A*)} \cdot \frac{p(Y \to ZB)}{p(Y \to (Z * \mid * Z))} \cdot \prod_{r \in S_B} p(r), \tag{6}$$

where N_I is the number of nodes in T that have at least one insertable child, and $N_{I;X}$ is the number of insertable children of X in T. The third fraction is the probability of choosing Y as the new child in T', and the fourth fraction is the probability of choosing B as the new sibling of Z, as well as whether Z is the left- or right-hand child of Y. The final term is the probability of the subtree S_B.

Acceptance probability Using the jump probabilities between trees given by the above equations, we can calculate the acceptance probability of a possible Metropolis move from T to T'. This is the product of the ratio of the forward and backward jump probabilities and the ratio of the trees given the current violation profile m:

[2]This equation assumes that Z is the right-hand child of X and the left-hand child of Y. If Z is the left-hand child of X or the right-hand child of Y or both, the probability is calculated similarly, but the third or fourth fraction changes to reflect the actual structure.

$$\frac{p(m|T')p(T')J(T;T')}{p(m|T)p(T)J(T';T)}. \tag{7}$$

The Metropolis method samples constraint definitions $\{d^{(1)}, \cdots, d^{(n)}\}$ from the posterior distribution $p(d|M_{\cdot l})$. These samples are used to estimate the probability of the violation profile m given the constraint grammar G by taking the harmonic mean of $p(M_{\cdot l}|d^{(t)})$ over all samples (Newton and Raftery, 1994).[3] This provides a prior for the columns of the matrix; coupled with the Poisson prior on the number of columns, we have a phonologically-motivated prior over matrices with an indefinite number of columns.

4.2 Inference over M

Inference on M uses five possible sampling moves, all of which rely on the estimates of $p(M_{\cdot l}|G)$ obtained above. Three of the sampling moves are equivalent to previous work with non-parametric binary constraint matrices (Görür et al., 2006; Doyle et al., 2014): columns may be removed or added, and each cell M_{jl} is Gibbs sampled, potentially changing whether candidate y_j violates constraint l.

We introduce two new moves – splitting or combining columns – to more efficiently move between constraint definitions. These can shift violations that explain the data well but are exceptions within their current column into a column where they fit better. Without them, moving violations between columns requires first removing them via Gibbs sampling, which may be very unlikely due to the loss in data likelihood from the loss of critical constraint violations.

A proposed split and its acceptance probability are drawn as follows. The likelihood of a violation M_{jl} being an exception within its profile $M_{\cdot l}$ is estimated from the proportion of samples from $p(d|M_{\cdot l})$ that mark the violation as exceptional. The set of violations V to be moved is drawn as a sequence of independent Bernoulli draws based on each violation's likelihood of being an exception. The exception likelihood is smoothed using a Beta-binomial distribution with parameter β, by taking the maximum a posteriori estimate of the likelihood:

$$p(m_{jl} \in V) = \frac{N_E + \beta - 1}{N_E + N_N + 2\beta - 2} \tag{8}$$

The number of Metropolis samples in which M_{jl} was an exceptional violation is N_E and a non-exceptional violation is N_N. Higher β increases the overall smoothing, and the effect of the smoothing decreases as more Metropolis samples are drawn. We set $\beta = 100$, as we expect substantial noise due to the size of the sample space.

4.3 Inference over w

After each matrix sample, we apply Metropolis-Hastings sampling on w. Our proposal distribution is $-\Gamma({w_k}^2/\eta_M, \eta_M/-w_k)$, which the current weight w_k as its mean. We set $\eta_M = 1$ as a default.

5 Experiment

5.1 English regular plural morphophonology

We test this model on the English regular plural system, which has one underlying form (/z/) with three attested output realizations: [z], [s], or [əz] (as in *hugs, huts*, and *hushes*, respectively). Two markedness constraints drive this alternation in the standard phonological analysis, which can be written in terms of the phonetic feature sequences they penalize: [-VOI][+VOI] and [+STR][+STR]. The former penalizes outputs where consecutive consonants do not agree in voicing, and the latter penalizes outputs where consecutive consonants are both strident (s,z,sh,ch). These are coupled with three faithfulness constraints, which penalize removing, adding, or changing a phoneme (MAX, DEP, and IDENT in OT terminology).

For this experiment, we consider four candidate outputs for each input: the bare singular form, plus forms with each of the three attested allomorphs of the regular plural suffix. The candidates for plural

[3] Harmonic mean estimation can be noisy and take a large number of iterations to converge (Neal, 1994), so we tested a range of violation profiles and found consistent convergence to the expected constraint definitions and profile probabilities within a few thousand samples.

hug (underlying /hʌgz/), for instance, are [hʌg], [hʌgz], [hʌgs], or [hʌgəz]. In general, the [əz] candidate wins only when the singular ends in a strident, the [s] candidate wins only when the singular ends in a voiceless non-strident, and the [z] candidate wins the rest of the time. The training set consists of the plural forms of 26 nouns, each understood by at least 89% of 18-month-old English learners (Dale and Fenson, 1996).[4] The model is given 100 examples of each plural, always using the standard pluralization.

5.2 The constraint grammar

Potential constraint definitions are sequences of phonological feature bundles. There are 23 phonological features, each capturing different characteristics of a sound; for instance, the phoneme [s] has phonological features including [+consonantal, +strident, –voiced], while the similar phoneme [z] has features including [+consonantal, +strident, +voiced]. A feature bundle within a constraint definition matches all phonemes with all of the bundle's features. Thus a definition [+consonantal, +strident][+consonantal, +voiced] matches the strings sz and zz, but not zs. Phoneme-to-feature mappings are based on Riggle (2012).[5] To make sure that model's success is not based on the grammar generating only definitions relevant to the English plural problem, we include Kleene stars, matching zero or more consecutive occurrences of a feature bundle. While some other phonological constraint definitions, such as vowel harmony, require Kleene stars, the English plural does not.

Note that the constraints are not necessarily binary when defined as sequences of feature bundles; a candidate can contain multiple sequences that violate a constraint. But because we are considering the effect of adding a suffix to a stem word, we can subtract the stem's violations from each candidate. Since all the candidates from a given input share the singular form as a stem, the same number of violations are subtracted from all of them, and the candidate probabilities within the log-linear model are unchanged.

5.3 Model parameters and implementation

We ran the model for 200 iterations in three trial runs, with deterministic annealing on the first 100. For estimating $p(M_{.l})$, 1000 burn-in samples were taken and discarded, and 250 additional samples (every second sample out of 500 to reduce autocorrelation) of d are averaged. For violation profiles $M_{.l}$ that reoccured, each time $p(M_{.l})$ was re-calculated, half as many additional samples were drawn (125, 62, ..., to a minimum of 25) and incorporated into the average. The parameters α and η_w are set to 1 and b is set to 10, to encourage fewer constraints and parity between violations and definitions. These fit the standard phonological assumption of phonologically-motivated and parsimonious constraints.

6 Results

We tested the model's performance in four ways: how well the learned structures explain observed plurals, how well they predict novel plurals, how accurately they reflect the standard violation profiles, and how interpretable the constraint definitions are. In all cases, we compared against a baseline of the standard phonological constraints that phonological theory suggests. This baseline M was derived from the two standard English markedness constraints, [+STR][+STR] and [-VOI][+VOI], with no exceptions. Baseline weights were sampled as in Sect. 4.3, with M held constant.

Explaining observed plurals The first test is to show that the model can learn a phonological system for the observed plurals. The model satisfies this goal if it predicts the observed forms at least as well as the baseline model. We calculate both the mean and MAP values of the data likelihood (Eq. 2) over the final 100 iterations of each of the three model runs, and report the across-run means in Table 1. t-tests found no significant differences between the learned and baseline performance on the training data.

Predicting novel plurals The second test of the model is whether its learned constraints extend to newly-encountered words. This is a crucial feature for human acquisition; children quickly learn to generalize morphophonological systems. It also represents an important model improvement, as Doyle

[4]Training words: *baby, ball, balloon, banana, bath, bird, blanket, book, car, chair, daddy, diaper, door, drink, eye, hug, key, kiss, kitty, mommy, nap, nose, phone, shoe, spoon, toothbrush*

[5]There is one deviation from Riggle's system: we do not specify voicing on sonorants, because sonorants do not have voiceless versions and do not trigger [-VOI][+VOI] violations.

	MAP LL	Mean LL	Min. Cand. Prob.	Mean Cand. Prob.
Model	-2.94	-7.91	.986	.995
Baseline	-5.16	-10.6	.974	.991

Table 1: Comparing the performance of the learned constraints to the baseline of the standard phonological constraint definitions. On the left, training data log-likelihoods on the left, based on values from the final 100 iterations for the three model runs. On the right, test set probability masses for the correct plural forms. The learned constraints perform as well as the phonologically standard constraints.

et al. (2014)'s constraint learning model was incapable of making such predictions due to its strictly-extensional constraints. For the test set, we used the 25 most frequent countable nouns in the Corpus of Contemporary American English (COCA; Davies (2008)) that take regular plurals, none of which were in the training set.[6] Five of these nouns end with phonemes that did not occur word-finally in the observed data, requiring the model to have made phonological generalizations from the training data.

To assess the predictive power of the learned constraints, we obtained constraint definitions by using the $p(d|m)$ Metropolis sampler to generate a distribution over definitions for each violation profile. Violation profiles $M_{\cdot l}$ are taken from the final iteration of each model run. For a new candidate y, the probability $m_{y;l}$ that y violates constraint l was estimated using constraints d drawn from the $p(d|M_{\cdot l})$ Metropolis sampler. $m_{y;l}$ was then used as the constraint value for the log-linear predictor. Both the model and baseline constraints correctly put the highest probabilities on the correct plural forms, as shown in Table 1. All correct plural forms received at least 98% of the probability mass under the model constraints, and there was no significant difference between the model and baseline predictions.

Violation profile accuracy The previous test showed that the constraint definitions effectively extend to unobserved forms. Now we want to examine their correspondence with phonological theory. First, we want to see if the right number of constraints was learned. Two of the model runs had two markedness constraints throughout the final 100 iterations, like the baseline. The third model run used four markedness constraints over its final 100 iterations, but the extra markedness constraints supplied violations that matched two of the faithfulness constraints (DEP and IDENT). Those faithfulness constraints' weights dropped to near zero in this run, though, meaning that all learning and baseline runs had five active constraints. In the runs with two markedness constraints, we tested how their violation profiles and definitions mapped to the baseline constraints.[7] Over the final 100 iterations, the learned violation profiles agree with their corresponding baseline violation profiles on an average of 98.9% of all candidates, showing that both constraint sets have similar phonological meanings.

Similarities in constraint definitions We compared the likely constraint definitions, as estimated by the Metropolis sampler for $p(d|m)$, for the learned and baseline violation profiles to their phonologically standard counterparts. Table 2 shows the most likely constraint definitions for each violation profile, given either the baseline violation profiles (based on the standard constraint definitions) and the two-constraint runs of the model. On three of the four learned constraints, the model agrees with the inferred definition given the baseline violations. Reasons for the deviations from the phonologically standard definitions are discussed in Sect. 7.

Experiment summary We performed four tests of the constraint learner. The model learned a set of constraints and weights that could explain observed data and effectively generalize to unobserved forms. In addition, we find that the constraint definitions it learns correspond with the definitions that come from a baseline set of constraints, although additional information is needed to identify the exact same constraints as the baseline set.

[6]These words are: *time, year, way, day, thing, world, school, state, family, student, group, country, problem, hand, part, place, case, week, company, system, program, question, government, number, night*

[7]The remaining analyses are limited to the two-constraint learning runs; the four-constraint solution represents convergence failure to a local optimum with joint probability (Eq. 3) well below the two-constraint solutions because of its lower probability M. Better exploration of the constraint space would move this run toward the two-constraint solution.

Standard	Baseline	Model Run 1	Model Run 2
[+STR][+STR]	[+STR][-SYL]	[+STR][-SYL]	[+STR][-SYL]
[-VOI][+VOI]	[-VOI][+VOI]	[-VOI][+VOI]	[-VOI,-STR][-SG,-HI]*[-NAS,+VOI]
Key: voi=voicing, str=strident, sg=spread glottis, hi=high, nas=nasal, syl=syllabic			

Table 2: The phonological standard definitions and the most likely constraint definitions inferred in the baseline and model runs. Baseline/model d likelihoods based on 10000 samples from $p(d|m)$.

7 Discussion

Definitional ambiguity Although the constraints extend seamlessly to new data and their violation profiles mostly match, Table 2 showed the constraint definitions don't quite match the standard definitions. This is because multiple definitions can have identical violation profiles, as there are many phonological features; for instance, based on the first constraint's violation profile, the model has learned to penalize sz and zz sequences, but not səz. Phonological theory says that this constraint's definition is [+STR][+STR], but given the available data, any feature that is negative for [z] and [s] but positive for [ə] (or vice versa) will produce the same violation pattern, and the model has no reason to prefer one to the other.[8]

The complex definition of the second constraint in the second model run arises similarly. Small differences (8% of violations) between the model and baseline violation profiles lead the model to infer this more complicated definition, which penalizes stems ending in voiceless non-stridents getting either the [əz] suffix (with [ə] matching the [-SG,-HI] bundle and [z] matching [-NAS,+VOI]) or the [z] suffix (with the Kleene star vacuously satisfied). Such stems should get the [s] suffix, so this constraint definition is consistent with the observed data, and overreaching by handling two constraints' function: penalizing [z] like the [-VOI][+VOI] constraint would, but also penalizing [əz], which is covered by the faithfulness constraint DEP.

Such definitional ambiguity can be reduced through simultaneous learning of multiple phonological phenomena. The [+STR][-SYL] definition could be ruled out by observing the faithful manifestation of s-initial onset clusters in English, as in *stop* or *spin*; the Kleene-star definition could be ruled out by faithful realizations of non-harmonious *kid* or *peg*. Such learning would also be more realistic, as learners generally observe and learn a range of phonological phenomena simultaneously.

Relaxing binarity One important remaining step is to allow for non-binary constraints in the model, which could be introduced in multiple ways. One possibility is to mimic non-binary constraints through multiple, overlapping binary constraints (Frank and Satta, 1998), though this would require changes to the current PCFG. Another possibility is to treat the existing binary matrix as an indicator of whether a constraint is violated and add a second matrix, with positive integer values, corresponding to the number of violations of that constraint. Griffiths and Ghahramani (2011) use a similar design to overcome the binary nature of an Indian Buffet Process for object recognition.

Theory testing Our model also represents a way to investigate the plausibility of different theoretical statements of a constraint, casting constraint selection through the lens of model comparison. In addition, if the underlying constraint grammar is varied, this model could be used to investigate the plausibility and effectiveness different potential grammars.

8 Conclusion

We presented a model for learning binary, symbolically-defined constraints in a log-linear model from a combination of observational data and an infinite grammar over constraint definitions. We tested this model on a morphophonological problem and showed that it accurately inferred the values of the constraints, and found appropriate constraint definitions (though with some issues of definitional ambiguity).

[8] In fact, $p(d|m)$ is approximately equal for a range of constraint definitions that include [+STR][-SYL], [+STR][+STR], [+STR][-LABIAL], and others.

Acknowledgements

We wish to thank Eric Baković, Klinton Bicknell, Dave Barner, Charles Elkan, Andy Kehler, the UCSD Computational Psycholinguistics Lab, the Phon Company, and the COLING reviewers for their discussions and feedback on this work. This research was supported by NSF award IIS-0830535 and an Alfred P. Sloan Foundation Research Fellowship to RL.

References

Doug Beeferman, Adam Berger, and John Lafferty. 1999. Statistical models for text segmentation. *Machine Learning*, 34:177–210.

Iris Berent, Colin Wilson, Gary F. Marcus, and Douglas K. Bemis. 2012. On the role of variables in phonology: Remarks on Hayes and Wilson 2008. *Linguistic Inquiry*, 43:97–119.

Paul Boersma and Bruce Hayes. 2001. Empirical tests of the Gradual Learning Algorithm. *Linguistic Inquiry*, 32:45–86.

Joan Bresnan, Anna Cueni, Tatiana Nikitina, and R. Harald Baayen. 2007. Predicting the dative alternation. In G. Bourne, I. Kraemer, and J. Zwarts, editors, *Cognitive Foundations of Interpretation*. Royal Netherlands Academy of Science, Amsterdam.

Philip S. Dale and Larry Fenson. 1996. Lexical development norms for young children. *Behavioral Research Methods, Instruments, and Computers*, 28:125–127.

Mark Davies. 2008. The Corpus of Contemporary American English: 450 million words, 1990-present.

Gabriel Doyle, Klinton Bicknell, and Roger Levy. 2014. Nonparametric learning of phonological constraints in Optimality Theory. In *Proceedings of the Association for Computational Linguistics*.

Robert Frank and Giorgio Satta. 1998. Optimality theory and the generative complexity of constraint violability. *Computational Linguistics*, 24:307–315.

Sharon Goldwater and Mark Johnson. 2003. Learning OT constraint rankings using a Maximum Entropy model. In *Proceedings of the Workshop on Variation within Optimality Theory*.

Noah Goodman, Joshua Tenebaum, Jacob Feldman, and Tom Griffiths. 2008. A rational analysis of rule-based concept learning. *Cognitive Science*, 32:108–154.

Dilan Görür, F. Jäkel, and Carl Rasmussen. 2006. A choice model with infinitely many latent features. In *Proceedings of the 23rd International Conference on Machine Learning*.

Thomas Griffiths and Zoubin Ghahramani. 2005. Infinite latent feature models and the Indian buffet process. Technical Report 2005-001, Gatsby Computational Neuroscience Unit.

Thomas Griffiths and Zoubin Ghahramani. 2011. The Indian Buffet Process: An introduction and review. *Journal of Machine Learning Research*, 12:1185–1224.

Bruce Hayes and Colin Wilson. 2008. A maximum entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*, 39:379–440.

Bruce Hayes. 1995. *Metrical Stress Theory: Principles and Case Studies*. U. of Chicago, Chicago.

Bruce Hayes. 1999. Phonetically driven phonology: the role of optimality theory and inductive grounding. In M. Darnell, E. Moravcsik, M. Noonan, F. Newmeyer, & K. Wheatley, editor, *Formalism and Functionalism in Linguistics, vol. 1*. Benjamins.

John McCarthy. 2008. *Doing Optimality Theory*. Blackwell.

Radford Neal. 1994. Response to approximate Bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society. Series B (Methodological)*, 56:3–48.

Michael Newton and Adrian Raftery. 1994. Approximate Bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society. Series B (Methodological)*, 56:3–48.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the ACL.*

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Alan Prince and Paul Smolensky. 1993. Optimality Theory: Constraint interaction in generative grammar. Technical report, Rutgers Center for Cognitive Science.

Jason Riggle. 2009. Generating contenders. *Rutgers Optimality Archive*, 1044.

Jason Riggle. 2012. Phonological feature chart (v. 12.12). December.

Bruce Tesar and Paul Smolensky. 2000. *Learnability in Optimality Theory*. MIT Press.

Chinese Tense Labelling and Causal Analysis

Hen-Hsen Huang, Chang-Rui Yang, and Hsin-Hsi Chen
Department of Computer Science and Information Engineering
National Taiwan University
No. 1, Sec. 4, Roosevelt Road, Taipei, 10617 Taiwan
{hhhuang, cjyang}@nlg.csie.ntu.edu.tw; hhchen@ntu.edu.tw

Abstract

This paper explores the role of tense information in Chinese causal analysis. Both tasks of causal type classification and causal directionality identification are experimented to show the significant improvement gained from tense features. To automatically extract the tense features, a Chinese tense predictor is proposed. Based on large amount of parallel data, our semi-supervised approach improves the dependency-based convolutional neural network (DCNN) models for Chinese tense labelling and thus the causal analysis.

1 Introduction

Causal analysis plays a crucial role in the applications such as event extraction (Hashimoto et al., 2012; 2014), causality inference (Tanaka et al., 2012), question-answering (Oh et al., 2013), and motivation identification (Nguyen et al., 2015). Compared to English, the topic of causal analysis in Chinese is rarely touched. In this work, we explore the role of tense information in Chinese causal analysis. As pointed by Mirza (2014), the causal relation and temporal information is correlated. In a causal relation, the cause intuitively precedes its effect. In other words, the tense information could be useful features in the tasks of causal analysis.

Two tasks of causal analysis are investigated in this study: causal type classification and causal directionality identification. The Chinese discourse relation corpus, Chinese Discourse Treebank (CDTB) (Li et al., 2014), is adopted as our dataset. In CDTB, six types of causality relations, **Purpose**, **Background**, **Hypothetical**, **Inference**, **Condition**, and **Cause-Result**, are defined.

A discourse relation connects two arguments. In the case of causality, one of the two arguments (e.g., *arg*1) presents a situation, and it is causally affected by the other argument (e.g., *arg*2). For example, the first part of the sentence (S1) shows a reason, and the second part, which is underlined, is its effect.

(S1) 由於產能不足，國內自給率不到四成，大部分要仰賴進口。 (Because of insufficient capacity, the domestic self-sufficiency rate is less than 40, most rely on imports.)

The direction of *arg*1 and *arg*2 is reversible. Like (S1), the reason is described in the former argument in most cases, and the effect is presented in the latter argument. However, (S2) shows a counter-example that presents the effect in the former part. To exactly extract the cause and the effect in natural language, causal directionality identification is required.

(S2) 西藏銀行部門積極調整信貸結構，以確保農牧業生產等重點產業的投入，加大對工業，能源，交通，通信等建設的正常資金供應量。(Tibet banking sector actively adjust credit structure in order to ensure the input of agricultural production and other key industries, and increase the industrial, energy, transportation, communications, construction of the normal supply of funds.)

There is no tense annotation in the CDTB. For this reason, we select all the samples of causality re-

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2227–2237, Osaka, Japan, December 11-17 2016.

lation from CDTB, and manually label the tense for each argument as ground-truth for the two tasks. To automatically extract the tense features, a Chinese tense predictor is required. The grammatical tense in English explicitly denotes the temporal information for a given text. In Chinese, however, the temporal information is communicated with aspect particles such as 了 (le) and 着 (zhe) and temporal adverbials such as 現在 ("now") and 明天 ("tomorrow") (Xue et al., 2008; Ge et al., 2015). In other words, it is more challenging to determine the tense in Chinese text. Thus, we propose a semi-supervised algorithm that learns to label tense information in Chinese text. With UM-Corpus, a large English-Chinese parallel corpus aligned at sentence-level (Tian et al., 2014), we generate a pseudo-labelled Chinese tense corpus by deriving the tense information from their English counterpart. Dependency-based convolutional neural network (DCNN) is trained to predict Chinese tense. We incorporate the semi-supervised Chinese tense predictor in the tasks of causal type classification and causal directionality identification. The experimental results are compared with the supervised approach and the ideal situation where human-labelled information is available.

The contribution of this paper is three-fold: (1) we transfer the tense information from English sentence to its Chinese counterpart based on sentence-aligned English-Chinese parallel corpus, (2) we train Chinese tense predictor with DCNN and use it to label tense markers on a Chinese sentence, and (3) we apply the tense information to identify causal type and causal directionality of a sentence. The rest of this paper is organized as follows. Section 2 surveys the related work. Section 3 describes the experimental materials. Section 4 shows our approach to Chinese tense labelling. Section 5 illustrates the use of tense information in causal type and causal directionality identification. Section 6 concludes this paper.

2 Related Work

Causal analysis attracts much attention in AI community for years. A variety of issues have been explored. One of the hottest topic is event analysis, where causal information plays a crucial role (Do et al., 2011; Riaz and Girju, 2013; 2014; Mirza and Tonelli, 2014; Kives et al., 2015). Other applications include generation of event causality hypotheses (Hashimoto et al., 2015), motivation identification (Nguyen et al., 2015), causality detection and extraction (Hashimoto et al., 2012; Mihaila and Ananiadou, 2013), causal inference (Tanaka et al., 2012), question answering (Oh et al., 2013), and future scenario generation (Hashimoto et al., 2014). The correlation between temporality and causality is studied by Mirza (2014) and Mirza and Tonelli (2014).

Unlike English, no grammatical tense is available in Chinese. Various approaches are explored to address the topic of Chinese tense prediction. Liu et al. (2011) propose an unsupervised method for Chinese tense labelling by learning from a Chinese-English parallel corpus. Zhang and Xue (2014) deal with Chinese tense inference by training a supervised model with various linguistic features on a Chinese tense corpus (Xue and Zhang, 2014). Following the unsupervised method by Liu et al. (2011), we develop a semi-supervised model that benefits from a large amount of data labelled by an accurate English tense predictor.

Neural networks such as recurrent neural network (RNN) and convolutional neural network (CNN) are very popular in NLP community. Kim (2014) releases a sentence classifier with convolutional neural network (CNN), where a sentence is represented as a sequence of word vectors (Mikolov et al., 2013). Based on Kim's work, Ma et al. (2015) propose the dependency-based CNN (DCNN) by adding the structure information features to the sentence representation. In this work, we employ DCNN for Chinese tense classification under supervised, unsupervised, and semi-supervised learning.

3 Linguistic Resources

Three types of corpora are used in this work. Section 3.1 describes the corpus for Chinese causal analysis. Section 3.2 and Section 3.3 introduce the corpora for developing our Chinese tense predictor.

3.1 Chinese Causality Corpus

There are few resources for Chinese causal analysis. In this work, we extract instances labelled with causality relation in the Chinese Discourse Treebank (CDTB) (Li et al., 2014) as the basis of our causality dataset. Similar to the English discourse corpus, e.g., Penn Discourse Treebank (PDTB) (Prasad

et al., 2008), CDTB is a Chinese corpus annotated with discourse information. A type of discourse relation is given to a pair of text spans (arguments). For instances of the explicit discourse relation, the connectives (discourse markers) are also annotated.

CDTB does not provide the information of causal directionality. Here we manually label the directionality for each instance extracted from CDTB. Table 1 summarizes the six types of the causality relation. The distributions of explicit/implicit and directionality are shown. Cause-Result, which appears more than 50%, is the majority. Most cases are implicit except Hypothetical and Condition. In terms of directionality, 73.1% of instances are in the direction of Reason-Effect. Furthermore, all the instances of Hypothetical, Inference, and Condition are Reason-Effect. In contrast, 78% of Purpose instances are Effect-Reason. In general, Chinese speakers tend to express the reason before the effect. We release the annotated tense corpus as a resource for NLP community.[1]

Causal Type	Number of Instances	Explicit or Implicit	Number of Instances	%	Directionality	Number of Instances	%
Purpose	332	Explicit	162	48.8%	Reason-Effect	73	22.0%
		Implicit	170	51.2%	Effect-Reason	259	78.0%
Background	127	Explicit	4	3.1%	Reason-Effect	98	77.2%
		Implicit	123	96.9%	Effect-Reason	29	22.8%
Hypothetical	69	Explicit	55	79.7%	Reason-Effect	69	100.0%
		Implicit	14	20.3%	Effect-Reason	0	0.0%
Inference	38	Explicit	3	7.9%	Reason-Effect	38	100.0%
		Implicit	35	92.1%	Effect-Reason	0	0.0%
Condition	71	Explicit	37	52.1%	Reason-Effect	71	100.0%
		Implicit	34	47.9%	Effect-Reason	0	0.0%
Cause-Result	677	Explicit	200	29.5%	Reason-Effect	612	90.4%
		Implicit	477	70.5%	Effect-Reason	65	9.6%
Total	1,314	Explicit	461	35.1%	Reason-Effect	961	73.1%
		Implicit	853	64.9%	Effect-Reason	353	26.9%

Table 1: Statistics of the causality relations in Chinese causality corpus.

3.2 Chinese Tense Corpus

Human-annotated and machine-generated Chinese tense corpora will be used to learn Chinese tense predictor. The human-annotated Chinese tense corpus was developed by Xue and Zhang (2014). Based on a word-aligned Chinese-English parallel treebank, tense, modality, eventually, and event types are manually annotated. Due to the copyright issue, only a subset of data is available for us. For every event, one of the seven tenses is labelled: “Past”, “Present”, “Future”, “Relative Past”, “Relative Present”, “Relative Future”, and “None”. We convert all the relative tenses to absolute ones. Finally, total 3,358 instances are extracted. Figure 1 shows the distribution of the human-annotated dataset used in the experiments.

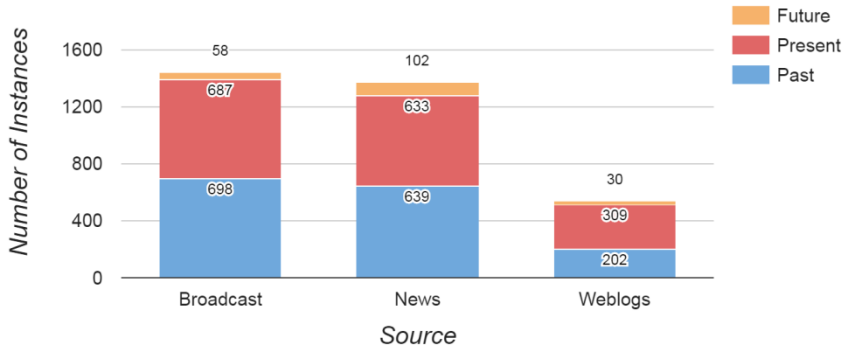


Figure 1: Distribution of instances extracted from the human-labelled Chinese tense corpus.

[1] http://nlg.csie.ntu.edu.tw/nlpresource/chinese_causality

3.3 English-Chinese Parallel Corpus

In contrast to the human-annotated Chinese tense corpus, a large English-Chinese parallel corpus, UM-Corpus (Tian et al., 2014), is adopted to label Chinese sentences with tense information. In UM-Corpus, text from eight domains are collected and aligned at sentence-level. A total of 2,215,000 sentences are released. How to develop the machine-generated Chinese tense corpus will be described in Section 4.2.

4 Chinese Tense Labelling

Because grammatical tense is inherent in an English sentence, tense prediction is relatively easier. A large amount of pseudo-labelled data can be generated by tense mapping between English-Chinese parallel sentences. Section 4.1 shows a rule-based tense predictor to determine the tense in the English side. Section 4.2 specifies how to transfer the tense information to its Chinese counterpart by bilingual verb alignment. Section 4.3 proposes a dependency-based convolutional neural network (DCNN) to predict Chinese tense.

4.1 Rule-based English Tense Predictor

Based on the definition of the Stanford typed dependencies[2], we develop a rule-based English tense predictor. For each of the 18 combinations among tenses, voices, and aspects, Table 2 presents the rules in the tense determination. Figure 2 illustrates the dependency tree of the sentence "He was being punished", where the verb "punished" is tagged as VBN (past participle verb), and its dependents contain aux(was/VBD) and auxpass(being/VBG). According to the rules in Table 2, the tense of this sentence is past, the voice is passive, and the aspect is progressive.

Tense	Voice/Aspect	Verb POS	Dep. Auxiliary Verb	Sample
Present	Active/Simple	VB, VBP, VPZ		I write.
	Active/Progressive	VBG	aux(am/VBP)	I am writing.
	Active/Perfect	VBN	aux(have/VBP)	I have written.
	Passive/Simple	VBN	auxpass(is/VBZ)	He is punished.
	Passive/Progressive	VBN	aux(is/VBZ), auxpass(being/VBG)	He is being punished.
	Passive/Perfect	VBN	aux(has/VBZ), auxpass(been/VBN)	He has been punished.
Past	Active/Simple	VBD		I wrote.
	Active/Progressive	VBG	aux(was)-VBD	I was writing.
	Active/Perfect	VBN	aux(had)-VBD	I had written.
	Passive/Simple	VBN	auxpass(was/VBD)	He was punished.
	Passive/Progressive	VBN	aux(was/VBD), auxpass(being/VBG)	He was being punished.
	Passive/Perfect	VBN	aux(had/VBD), auxpass(been/VBN)	He had been punished.
Future	Active/Simple	VB	aux(will/MD)	I will write.
	Active/Progressive	VBG	aux(will/MD), aux(be/VB)	I will be writing.
	Active/Perfect	VBN	aux(will/MD), aux(have/VB)	I will have written.
	Passive/Simple	VBN	aux(will/MD), auxpass(be/VB).	He will be punished.
	Passive/Progressive	VBN	aux(will/MD), aux(be/VB), auxpass(being/VBG)	He will be being punished.
	Passive/Perfect	VBN	aux(will/MD), aux(have/VB) auxpass(been/VBN)	He will have been punished

Table 2: Rules for English tense prediction with the information of POS tagging and dependency parsing.

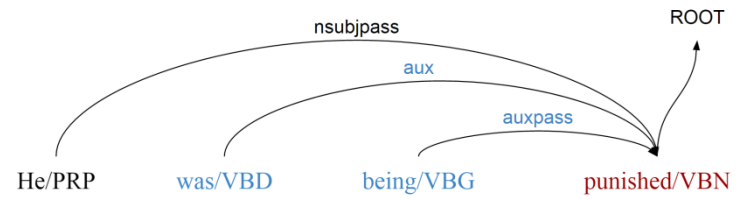


Figure 2: Dependency tree of the sentence "He was being punished".

[2] http://nlp.stanford.edu/software/dependencies_manual.pdf

We evaluate the performance of the English tense predictor on the dataset from the NTHU Academic Writing Database[3]. In this dataset, 1,171 English sentences are carefully annotated with linguistic information such as tense, voice, aspect, and argumentative zone. Our rule-based tense predictor achieves an accuracy of 91.98%. Error analysis shows that most wrongly labelled instances are due to the errors of POS tagging and dependency parsing. (S3) shows an example. The verb (VB) "image" is wrongly labelled as a noun (NN) by the Stanford tagger. Our rule-based English tense predictor is released as a tool[4].

(S3) "When combined with multiphoton excitation, both schemes can image thick samples with three-dimensional optical sectioning and much improved resolution."

4.2 Machine-generated Chinese Tense Corpus

As described in Section 3, total 2,215,000 English-Chinese parallel sentences are released in UM-Corpus. We perform Chinese word segmentation, POS tagging, and dependency parsing for the Chinese sentences with Stanford CoreNLP (Manning et al., 2014). UM-Corpus is aligned at sentence level, but a sentence may contain multiple verbs. For a sentence with multiple verbs, we employ the alignment tool GIZA++[5] to align English verbs with their Chinese counterparts (Och and Ney, 2003). However, not all cases are perfectly aligned. In the example shown in Figure 3, the English verb "go" is wrongly aligned with two Chinese tokens 那回 ("that time") and 去 ("go") because the Chinese word segmenter does not correctly separate 那 ("the") and 回 ("back"). To reduce the noise, we remove all the instances that fail to align. As a result, we obtain 615,521 Chinese instances with tense information as the pseudo-labelled corpus.

Table 3 shows the statistics of this corpus. On the one hand, instances of the *present* tense, which occupy 63.75%, are the majority. On the other hand, only 6.7% of the instances are with the *future* tense. Among all domains, the odd distribution of Law is observable. About 49.09% of the instances in the Law domain are in *future* tense because most legal provisions are made to regulate what will happen in the future. Microblog is the smallest domain, i.e., only 954 instances are found.

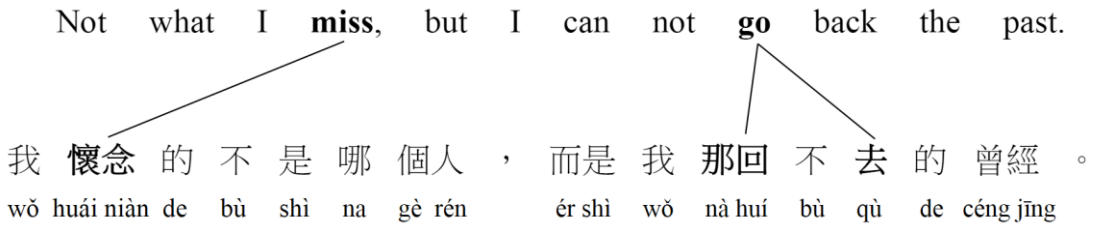


Figure 3: An imperfectly aligned case where the English verb "go" is aligned with two Chinese tokens due to word segmentation error.

4.3 DCNN-Based Chinese Tense Predictor

Tense labelling for a given sentence is a task of sentence classification. In this work, we employ the dependency-based convolutional neural network (DCNN) as the classifier (Ma et al., 2015). Based on the sentence classifier with CNN (Kim, 2014), the DCNN gains improvement by incorporating the information of linguistic structure. In addition to a sequence of word vectors like the skip-gram (Mikolov et al., 2013), the outcome of dependency parsing such as ancestor paths and siblings are added to the sentence representation. In this work, the skip-gram is trained on the Tagged Chinese Gigaword (CGW) corpus 2.0 (Graff et al., 2005; Huang, 2009), and a Chinese word is represented as a vector with a dimension of 400.

[3] http://writing.wwlc.nthu.edu.tw/writcent
[4] http://nlg.csie.ntu.edu.tw/nlpresource/english_tense_predictor
[5] http://www.statmt.org/moses/giza/GIZA++.html

Domains	Past		Present		Future		Total
	#	%	#	%	#	%	
Education	51,906	32.67%	98,954	62.28%	8,022	5.05%	158,882
Laws	1,370	4.56%	13,930	46.35%	14,754	49.09%	30,054
Microblog	155	16.25%	743	76.94%	56	5.87%	954
News	50,768	34.79%	88,812	60.86%	6,350	4.35%	145,930
Science	12,222	19.75%	46,065	74.45%	3,586	5.80%	61,873
Spoken	25,924	33.37%	48,313	62.19%	3,447	4.44%	77,684
Subtitles	25,735	29.68%	57,064	65.82%	3,898	4.50%	86,697
Thesis	14,416	26.97%	38,543	72.11%	488	0.98%	53,447
Total	182,496	29.65%	392,424	63.75%	40,601	6.70%	615,521

Table 3: Distribution of the machine-labelled Chinese tense corpus.

4.3.1 Unsupervised Learning for Chinese Tense Labelling

In the setting of unsupervised learning, we train the DCNN classifier on the machine-generated Chinese tense corpus, and test on the human-annotated Chinese tense corpus. The support vector machine (SVM) with RBF kernel and the random forest (RF) classifiers are also trained as baseline models. The hyperparamters of both classifiers are adjusted with grid search. The McNemar test is applied for significance testing at p=0.05. Table 4 shows the results in accuracies in the order of domain size. In general, the more the data, the better the performance. All the three models trained on the tiny Microblog dataset are superior to those trained on Law, the relatively larger dataset, because of the odd distribution of the Law domain. The DCNN significantly outperforms the other two models in most domains except for Microblog and Subtitles. DCNN with the data from all domains achieves the highest accuracy of 68.62% in the unsupervised approach. The performances of SVM and RF with all data are slightly decreased. That confirms the selection of pseudo data is crucial for traditional classifiers (Liu et al., 2011). In contrast, the DCNN model is not affected by this issue. That shows the high discriminative ability of the neural network model.

Domains	Number of Instances	DCNN	SVM	RF
Microblog	954	48.62%	50.14%	49.45%
Law	30,054	43.28%	41.06%	40.75%
Thesis	53,447	54.95%	53.81%	49.97%
Science	61,873	57.70%	56.69%	52.56%
Spoken	77,384	65.07%	62.90%	60.38%
Subtitles	86,697	55.43%	56.27%	56.63%
News	145,930	66.80%	64.38%	62.36%
Education	158,882	67.91%	64.70%	62.22%
All Domains	615,521	**68.62%**	62.20%	61.57%

Table 4: Experimental results of learning from pseudo-labelled data by domains.

4.3.2 (Semi-)Supervised Learning for Chinese Tense Labelling

This section evaluates our model under supervised and semi-supervised learning. Five-fold cross validation is performed on the 3,358 genuine instances. For each fold, one fifth of 3,358 genuine instances (human-annotated) are used for testing, and four-fifth of 3,358 genuine instances and various amounts of pseudo-labelled (machine-generated) data are used for training. Table 5 compares the accuracies of

Settings	# Genuine Data	# Pseudo Data	DCNN	SVM	RF
Supervised	3,358	0	66.77%	64.79%	65.92%
Unsupervised	0	615,521	68.62%	62.20%	61.57%
Semi-Supervised	3,358	10,000	68.00%	66.10%	62.85%
	3,358	20,000	68.59%	66.33%	61.99%
	3,358	100,000	67.97%	66.60%	63.80%
	3,358	300,000	68.56%	66.42%	64.13%
	3,358	615,521	**69.64%**	65.86%	63.83%

Table 5: Comparison of supervised, unsupervised, and semi-supervised learning for Chinese tense labelling.

supervised, unsupervised, and semi-supervised training. Our model gains improvement by adding the genuine instances to large pseudo-labelled corpus. Compared to supervised training, adding the pseudo-labelled data increases the performance of DCNN up to 69.64%. SVM is also improved under semi-supervised training. RF is a counter-example that performs best under supervised training, but still does not compete with DCNN.

5 Causal Analysis

The DCNN-based Chinese tense predictor is used to label the tense features to the instances in the causal corpus. Sections 5.1 and 5.2 confirm if the two tasks of causal analysis gain improvement from tense information. This work focuses on the correlation between tense information and causal analysis in Chinese text. The bag-of-word SVM classifiers with or without tense features are experimented to verify if the tense information improves the two tasks of causal analysis. The tense features consist of six binary values: *arg*1-*is-past*, *arg*1-*is-present*, *arg*1-*is-future*, *arg*2-*is-past*, *arg*2-*is-present*, and *arg*2-*is-future*. Three sources of tense features are compared: labelled by the supervised model (M_{super}), labelled by the semi-supervised model (M_{semi}), and labelled by human (M_h). Refer to Section 4.3.2, the supervised model is the DCNN-based Chinese tense predictor trained on 3,358 genuine data. The semi-supervised model is the DCNN-based Chinese tense predictor trained on the combination of 3,358 genuine and 615,521 pseudo data. The model with human-labelled tense, M_h, is an ideal model since human-labelled information is unavailable in real applications. Five-fold cross validation is performed. The hyperparameters are adjusted for the SVM (RBF) classifier with grid search. The McNemar test is applied for significance testing at p=0.05.

5.1 Causal Type Classification

In the task of causal type classification, the model predicts one of the six causal types for a given argument pair. The performances measured in accuracy and macro F-score are given in Table 6. Compared to the model with only Word feature (M_w), tense information indeed improves the performance of this task. M_h is significantly superior to M_w at p=0.05. Furthermore, it is surprising that M_{semi} competes with M_h.

Model	M_w		M_{super}		M_{semi}		M_h	
	Accuracy	F-Score	Accuracy	F-Score	Accuracy	F-Score	Accuracy	F-Score
Explicit	75.48%	44.74%	76.35%	44.93%	76.57%	46.48%	**77.00%**	**46.63%**
Implicit	59.78%	30.84%	60.60%	29.09%	**62.36%**	**32.48%**	62.25%	29.71%
Overall	65.28%	35.71%	66.11%	34.64%	67.33%	**37.38%**	**67.41%**	35.64%

Table 6: Experimental results of causal type classification.

The confusion matrices of M_h and M_{semi} are shown in Tables 7 and 8, respectively. M_h tends to predict an instance to Cause-Result, the largest type of the six. In contrast, M_{semi} is fairer that more instances are classified to minor types.

(S4) is an example which is correctly classified to Cause-Result by M_{semi}, but wrongly classified to Background by M_h. The part of effect is underlined, while the rest is the part of reason. This instance shows the grey zone between Cause-Result and Background. By definition, Cause-Result holds on a stronger factually cause-effect relation.

Types	Purpose	Background	Hypothetical	Inference	Condition	Cause-Result
Purpose	66.57%	0.00%	0.30%	0.00%	1.51%	31.63%
Background	6.30%	19.69%	0.00%	0.79%	0.00%	73.23%
Hypothetical	13.04%	0.00%	49.28%	0.00%	1.45%	36.23%
Inference	10.53%	7.89%	0.00%	5.26%	5.26%	71.05%
Condition	21.13%	1.41%	1.41%	0.00%	21.13%	54.93%
Cause-Result	6.20%	4.73%	0.74%	0.44%	0.89%	87.00%

Table 7: Confusion matrix of the model with human-labelled tense features.

Types	Purpose	Background	Hypothetical	Inference	Condition	Cause-Result
Purpose	72.29%	0.60%	0.30%	0.00%	1.20%	25.60%
Background	6.30%	31.50%	0.00%	0.79%	1.57%	59.84%
Hypothetical	17.39%	0.00%	47.83%	0.00%	1.45%	33.33%
Inference	10.53%	15.79%	0.00%	5.26%	5.26%	63.16%
Condition	28.17%	1.41%	1.41%	0.00%	22.54%	46.48%
Cause-Result	9.45%	6.20%	1.33%	0.44%	0.74%	81.83%

Table 8: Confusion matrix of the model with the tense features labelled by our semi-supervised tense predictor.

(S4) 僅中國陸上三大天然氣最富集的四川盆地，近四十多年來，已累計生產一千六百三十三億立方米天然氣。基本上解決了成都、重慶等一批大中城市的民用燃料，並形成以天然氣為原料的中國最大的維尼龍生產線四川維尼龍廠。 (Sichuan Basin, the only place with the three major natural gas resources in China, nearly forty years, has produced a total of 163.3 billion cubic meters of natural gas. This basically provided domestic fuel for Chengdu, Chongqing and other cities, and found the Sichuan Vinylon plant, China's largest production line of Vinalon using natural gas as raw materials)

5.2 Causal Directionality Identification

In the task of causal directionality identification, the binary classifier predicts one of the two direction (i.e., Reason-Effect and Effect-Reason) for a given argument pair. Refer to Table 1, 73.1% of instances in the direction of Reason-Effect, and no instances in the direction of Effect-Reason are found in the Hypothetical, Inference, and Condition types. Thus, only the performances of Purpose, Background, and Cause-Result are reported in Table 9. The results are consistent with the task of causal type classification. The ideal model M_h achieves the best performance and significantly outperforms M_w (p=0.05), and M_{semi} is second.

Model	M_w		M_{super}		M_{semi}		M_h	
	Accuracy	F-Score	Accuracy	F-Score	Accuracy	F-Score	Accuracy	F-Score
Purpose	87.04%	78.31%	88.25%	81.23%	88.85%	82.19%	91.26%	86.04%
Background	77.16%	43.55%	77.16%	43.55%	77.16%	43.55%	78.74%	50.39%
Cause-Result	90.84%	54.52%	90.84%	53.29%	90.84%	55.68%	91.13%	59.18%
Overall	88.18%	60.52%	88.53%	60.35%	88.71%	62.06%	89.76%	66.03%

Table 9: Experimental results of causal directionality identification.

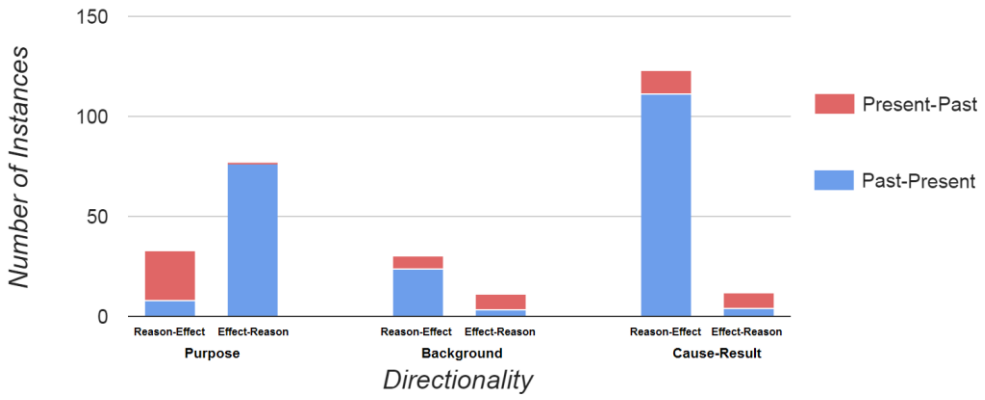


Figure 4: Relationship between causal directionality and chronology.

Figure 4 presents the relationship between causal directionality and chronology. Due to the sparseness of the tense of future, only the two transitions, Past to Present (forward) and Present to Past (reverse), are shown. The direction of Reason-Effect is the majority in the types of Background and Cause-Result, where the reason and the effect of most instances happen in the order of chronology. The type of Purpose is different. As the instance of Purpose shown in (S5), where the part of effect is underlined, while the rest is the part of reason. In the case of Purpose, the reason usually happens after the effect because the reason is the goal, and the effect is the manner to achieve to goal. The statistics reflects the special natural of the Purpose type.

(S5)香港特別行政區行政長官董建華今日（星期二）與四萬名信眾出席佛教界慶祝香港回歸祈福大會，為香港的繁榮安定及世界和平祝禱。(Today (Tuesday), Hong Kong Chief Executive Tung Chee-hwa and forty thousand faithful attended the Buddhist blessing event to celebrate the return of Hong Kong, for the prosperity and stability of Hong Kong and the world peace.)

6 Conclusion

This work investigates the role of tense information in Chinese causal analysis. We annotate the tense information on CDTB, and propose an approach that learns from parallel data for Chinese tense labelling. Our semi-supervised approach improves the performance of the DCNN and SVM models. The best model achieves an accuracy of 69.64% in Chinese tense labelling, while its outcome is useful information for the tasks of causal analysis.

Experimental results confirm the causal analysis tasks gain improvement from the tense features. Furthermore, we observe the high discriminative ability of the neural network model when the pseudo-labelled data are added to training set. Linguistics phenomena about causality and chronology are discussed with the evidence of data. We release the annotated tense corpus and a high performance rule-based English tense predictor for NLP community.

7 Acknowledgements

This research was partially supported by Ministry of Science and Technology, Taiwan, under grants MOST-104-2221-E-002-061-MY3 and MOST-105-2221-E-002-154-MY3, and National Taiwan University under grant NTU-ERP-104R890858. We are also very thankful to the Writing Center at National Tsing Hua University for providing us the NTHU Academic Writing Database, the NLP^{2}CT Laboratory at University of Macau for providing us the UM-Corpus, Chinese Language Processing Group at Brandeis University for providing us the Chinese Tense Corpus, and Professor Zhou Guodong for providing us the Chinese Discourse TreeBank.

References

Quang Xuan Do, Yee Seng Chan, Dan Roth. 2011. Minimally Supervised Event Causality Identification. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 294–303, Edinburgh, Scotland, UK.

Tao Ge, Heng Ji, Baobao Chang, and Zhifang Sui. 2015. One Tense per Scene: Predicting Tense in Chinese Conversations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (*Short Papers*), pages 668–673, Beijing, China.

David Graff, Ke Chen, Junbo Kong, and Kazuaki Maeda. 2005. Chinese Gigaword Second Edition LDC2005T14. Web Download. Philadelphia: Linguistic Data Consortium.

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, and Jong-Hoon Oh. 2015. Generating Event Causality Hypotheses through Semantic Relations. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2396- 2403.

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, Motoki Sano, Istvan Varga, Jong-Hoon Oh, and Yutaka Kidawara. 2014. Toward Future Scenario Generation: Extracting Event Causality Exploiting Semantic Rela-

tion, Context, and Association Features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 987–997, Baltimore, Maryland, USA.

Chikara Hashimoto, Kentaro Torisawa, Stijn De Saeger, Jong-Hoon Oh, and Jun'ichi Kazama. 2012. Excitatory or Inhibitory: A New Semantic Orientation Extracts Contradiction and Causality from the Web. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 619–630, Jeju Island, Korea.

Chu-Ren, Huang. 2009. Tagged Chinese Gigaword Version 2.0 LDC2009T14. Web Download. Philadelphia: Linguistic Data Consortium.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (EMNLP), pages 1746–1751, Doha, Qatar.

Christopher Kives, Stephen G. Ware, and Lewis J. Baker. 2015. Evaluating the Pairwise Event Salience Hypothesis in *Indexter*. In *Proceedings of the Eleventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (*AIIDE-15*), pages 30-36.

Yancui Li, Wenhe Feng, Jing Sun, Fang Kong, and Guodong Zhou. 2014. Building Chinese Discourse Corpus with Connective-driven Dependency Tree Structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), pages 2105–2114, Doha, Qatar.

Feifan Liu, Fei Liu, and Yang Liu. 2011. Learning from Chinese-English Parallel Data for Chinese Tense Prediction. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 1116–1124, Chiang Mai, Thailand.

Mingbo Ma, Liang Huang, Bing Xiang, and Bowen Zhou. 2015. Dependency-based Convolutional Neural Networks for Sentence Embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (*Short Papers*), pages 174–179, Beijing, China.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55-60.

Claudiu Mihaila and Sophia Ananiadou. 2013. What causes a causal relation? Detecting Causal Triggers in Biomedical Scientific Discourse. In *Proceedings of the ACL Student Research Workshop*, pages 38–45, Sofia, Bulgaria.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, 26, pages 3111-3119.

Paramita Mirza. 2014. Extracting Temporal and Causal Relations between Events. In *Proceedings of the ACL 2014 Student Research Workshop*, pages 10–17, Baltimore, Maryland USA.

Paramita Mirza and Sara Tonelli. 2014. An Analysis of Causality between Events and its Relation to Temporal Information. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2097–2106, Dublin, Ireland.

Dong Nguyen, Tijs van den Broek, Claudia Hauff, Djoerd Hiemstra, and Michel Ehrenhard. 2015. #SupportTheCause: Identifying Motivations to Participate in Online Health Campaigns. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2570–2576, Lisbon, Portugal.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19-51.

Jong-Hoon Oh, Kentaro Torisawa, Chikara Hashimoto, Motoki Sano, Stijn De Saeger, and Kiyonori Ohtake. 2013. Why-Question Answering using Intra- and Inter-Sentential Causal Relations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1733–1743, Sofia, Bulgaria.

Rashmi Prasad, Alan Lee, Nikhil Dinesh, Eleni Miltsakaki, Geraud Campion, Aravind Joshi, and Bonnie Webber. 2008. Penn Discourse Treebank Version 2.0 LDC2008T05. Web Download. Philadelphia: Linguistic Data Consortium, 2008. https://catalog.ldc.upenn.edu/LDC2008T05

Mehwish Riaz and Roxana Girju. 2013. Toward a Better Understanding of Causality between Verbal Events: Extraction and Analysis of the Causal Power of Verb-Verb Associations. In *Proceedings of the SIGDIAL 2013 Conference*, pages 21–30, Metz, France.

Mehwish Riaz and Roxana Girju. 2014. In-depth Exploitation of Noun and Verb Semantics to Identify Causation in Verb-Noun Pairs. In *Proceedings of the SIGDIAL 2014 Conference*, pages 161–170, Philadelphia, U.S.A.

Shohei Tanaka, Naoaki Okazaki, and Mitsuru Ishizuka. 2012. Acquiring and Generalizing Causal Inference Rules from Deverbal Noun Constructions. In *Proceedings of COLING 2012: Posters*, pages 1209–1218, COLING 2012, Mumbai, India.

Liang Tian, Derek F. Wong, Lidia S. Chao, Paulo Quaresma, Francisco Oliveira, Shuo Li, Yiming Wang, and Yi Lu. 2014. UM-Corpus: A Large English-Chinese Parallel Corpus for Statistical Machine Translation. In *Proceedings of the 9th International Conference on Language Resources and Evaluation* (*LREC*), Reykjavik, Iceland.

Nianwen Xue, Zhong Hua, and Kai-Yun Chen. 2008. Annotating "tense" in a Tense-less Language. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 3461-3466, Marrakech, Morocco.

Nianwen Xue and Yuchen Zhang. 2014. Buy one get one free: Distant Annotation of Chinese Tense, Event Type, and Modality. In *Proceedings of the 9th International Conference on Language Resources and Evaluation* (*LREC*), Reykjavik, Iceland.

Yuchen Zhang and Nianwen Xue. 2014. Automatic Inference of the Tense of Chinese Events Using Implicit Linguistic Information. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1902–1911, Doha, Qatar.

Exploring Topic Discriminating Power of Words in Latent Dirichlet Allocation

Kai Yang, Yi Cai*, Zhenhong Chen
School Of Software Engineering, South China University of Technology, China

Ho-fung Leung
Department of Computer Science and Engineering,
The Chinese University of Hong Kong,
Hong Kong

Raymond LAU
College of Business,
City University of Hong Kong,
Hong Kong

Abstract

Latent Dirichlet Allocation (LDA) and its variants have been widely used to discover latent topics in textual documents. However, some of topics generated by LDA may be noisy with irrelevant words scattering across these topics. We name this kind of words as topic-indiscriminate words, which tend to make topics more ambiguous and less interpretable by humans. In our work, we propose a new topic model named TWLDA, which assigns low weights to words with low topic discriminating power (ability). Our experimental results show that the proposed approach, which effectively reduces the number of topic-indiscriminate words in discovered topics, improves the effectiveness of LDA.

1 Introduction

Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and its variants are generative statistical topic models providing a powerful framework for finding topics in text documents. In generative process, each document is a mixture of several topics, and the generation of each word belongs to one of the document's topics (Heinrich, 2009).

Mimno et al. have found that LDA often produces topics that are not interpretable or meaningful (Mimno et al., 2011). According to our observation, most of topics (especially those considered uninterpretable) contain some words which are common in the corpus. For example, words like 'science', 'academic' or 'abstract' in a corpus about scientific publications will appear in most of topics. To explain this kind of words more clearly, we prepare another example showed in Table 1(a). The table shows the top 5 words for 5 topics generated by standard LDA from a corpus of reviews about smart phones. Word 'phone' can be easily recognized as a common word in the corpus about phones, and we can find that all the topics contain this word. For words which are likely to scatter across many topics are difficult to discriminate different topics, we denote this kind of words, such as 'phone', as **topic-indiscriminate words**. We use the term **topic discriminating power** to denote the ability of a word discriminating different topics. Topic-indiscriminate words have low topic discriminating power.

Table 1: An example about a result of standard LDA

(a) Result of standard LDA

Topic	Word
1	sound, headphones, **phone**, bass, card
2	screen, iphone, **phone**, display, ear
3	picture, **phone**, photo, video, gb
4	memory, sd, gb, **phone**, battery
5	android, **phone**, nexus, Samsung, google

(b) Document frequency (DF) of words

Word	DF	Word	DF
phone	2041	screen	1900
memory	1553	picture	1451
sound	1221	sd	928
android	915	iphone	837
headphones	428	card	389

*Corresponding author, e-mail: ycai@scut.edu.cn

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2238–2247, Osaka, Japan, December 11-17 2016.

These topic-indiscriminate words tend to bring some irrelevant words into topics, which make these topics less interpretable. We explain the cause of this negative effect using the following example. Subjectively, in Table 1(a), we can see that Topics 1, 2, 3, 4, 5 can be easily interpreted to topics about sound, screens, pictures, memory cards and Android systems respectively. Words such as 'card' in Topic 1, 'ear' in Topic 2 and 'battery' in Topic 4 seem to be irrelevant to other words in their topics, which make these topic hard to be understood. We consider that these words are brought into topics by topic-indiscriminate words. Figure 1 shows the word co-occurrence relationship about words in Topic 1. Each node represents a word, while two words are linked together if they co-occur in the same document. We can find that word 'card' only co-occur with 'phone' and never co-occur with other words. According to (Heinrich, 2005), if two words co-occur in the same document, these two words are more likely to be assigned at the same topic in LDA. Plausibly, word 'card' is assigned to Topic 1 because of the co-occurrence with word 'phone'. Hence, it is reasonable for us to consider that topic-indiscriminate words will result in worse performance of LDA.

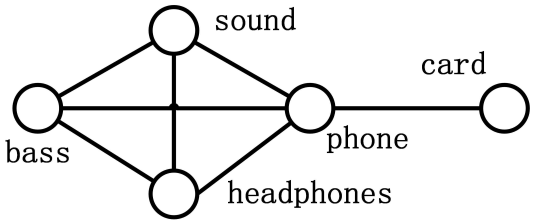


Figure 1: Graph about word co-occurrence

Wilson et al. claim that LDA should take weights of words in documents into consideration (Wilson and Chew, 2010). They consider that words which scatter across more documents are less important and should be given lower weights. We call this kind of word as document-indiscriminate words in our paper. Generally, stop words (e.g. 'the', 'of' and 'is') or common words are document-indiscriminate words, for the reason that these words appear in most of documents. Topic-indiscriminate words are a bit different from document-indiscriminate words, which is illustrated in the following example. Table 1(b) shows us top 10 most frequent words in the corpus of Table 1(a). We can find that words 'screen' and 'memory', which are kernel words for Topics 2 and 4, have high document frequency. On the other hand, they just appear in Topics 2 and 4 respectively. Therefore, they are document-indiscriminate words instead of topic-indiscriminate words, i.e. words with high topic discriminating power. In Wilson's approach, word 'screen' and 'memory' will be assigned lower weights to decrease their rankings in Topics 2 and 4. This will make topics less interpretable, as these words are important for people to understand topics. Hence, Wilson's approach has low ability to find out topic-indiscriminate words accurately, although it does well in finding document-indiscriminate words.

In this paper, we explore the topic discriminating power of LDA, and propose a new LDA model called Term Weighting LDA (TWLDA), which provides a way to measure this power according to supervised term weighting schemes. With our model, topic-indiscriminate words will be given lower weights and have less negative effect on the results of LDA. The reason why we apply supervised term weighting schemes to measure topic discriminating power is that they have been used to measure the discriminating power of words among categories in text categorization tasks (Lan et al., 2009). Words which concentrate on one topic can better discriminate that topic, and the topic discriminating power of these words are stronger. Hence, topic-indiscriminate words, whose topic discriminating power are weak, will be considered less important and be given lower weights in our proposed model. In summary, we conclude our contributions as follows: (a) We explore the topic discriminating power of words in LDA, and find that these words will make the generated topics less interpretable; (b) To solve the problem caused by topic-indiscriminate words, we propose a new model called TWLDA, which can measure the topic discriminating power of words and assign low weights to topic-indiscriminate words in order to reduce the negative effect caused by these words; (c) We explore our proposed TWLDA with different term weighting schemes, and find that supervised schemes, especially entropy-based supervised

schemes, have better performance than others; (d) We also conduct several experiments to demonstrate the effectiveness of TWLDA with different evaluation metrics.

2 Related Work

2.1 Latent Dirichlet Allocation

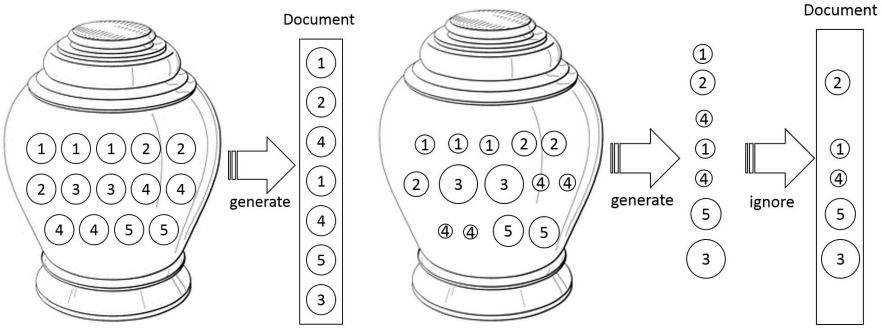
Latent Dirichlet Allocation (Blei et al., 2003) is a generative topic model. It assumes that the words in a document are drawn from a set of latent variables called topics which are distributions over words in the vocabulary.

However, some of the generated topics may mix unrelated or loosely-related words (Mimno et al., 2011). To tackle this problem, some knowledge-based topic models have been proposed in (Andrzejewski et al., 2009; Chen et al., 2013; Chen et al., 2014). These models use expert domain knowledge to guide LDA. For example, DF-LDA (Andrzejewski et al., 2009) takes domain knowledge in the form of must-links and cannot-links given by users. A must-links means that two words should be assigned to the same topic while a cannot-links means that two words should not. Besides, there are several models utilizing seed words provided by users (Burns et al., 2012; Jagarlamudi et al., 2012; Mukherjee and Liu, 2012). In some recent works, for example, GKLDA model (Chen et al., 2013) utilizes the general knowledge such as lexical knowledge to boost the performance.

2.2 Term Weighting Schemes and its Usage in LDA

Term weighting schemes are widely used to measure the importance of words in documents. They can be classified into supervised schemes and unsupervised schemes (Lan et al., 2009). The supervised schemes exploit category information of training documents while unsupervised schemes do not. There are many unsupervised schemes widely used in Information Retrieval (IR) tasks, such as tf, $tf \cdot idf$ (Sparck Jones, 1972) and some variants (Leopold and Kindermann, 2002; Paik, 2013). However, these schemes ignore the categories labels of each document. On the contrast, supervised schemes use the documents labeled with category information. Some supervised schemes are proposed recently, e.g., $iqf \cdot qf \cdot icf$ (Quan et al., 2011), rf (Lan et al., 2009) and some variants (Ko, 2012). Wang et al. propose some entropy-based term weighting schemes such as bdc which are based on the entropy of terms in categories (Wang et al., 2015). Wang et al. declare that bdc outperforms the state-of-the-art schemes, e.g. $tf \cdot idf$, $iqf \cdot qf \cdot icf$ and rf, in text categorization tasks.

Wilson et al. propose a model called WLDA, which applies term weighting schemes to weight terms in LDA. In their model, term weighting schemes are applied to measure the document discriminating power of words. Words which scatter across more documents are given relatively low weights. However, topic-indiscriminate words may not scatter across almost all the documents. Instead, they scatter across most of topics. Hence, the model proposed by Wilson et al. cannot give topic-indiscriminate words relatively low weights. To overcome this problem, we propose a new LDA model, which can give topic-indiscriminate words relatively low weights.



(a) Traditional generating process (b) Our proposed generating process

Figure 2: Word generating process

3 Generative Process Considering Weights of Words

Some of topics generated by LDA may be uninterpretable which contain irrelevant words. According to our observation, these words tend to scatter across many topics. We denote these words as **topic-indiscriminate words** due to the fact that they cannot discriminate different topics. We use the term **topic discriminating power** to denote the ability of words discriminating topics. Topics of LDA will be less interpretable if they mix with these topic-indiscriminate words. Hence, these words have much negative effect on the results of LDA.

To eliminate or alleviate the negative effect caused by topic-indiscriminate words, a possible way is to reduce the number of them occurring in documents. If the number of topic-indiscriminate words occurring in documents is discounted, the negative effect of these words to the results of LDA will also be alleviated (Heinrich, 2005). Inspired by this way, we propose a new generative process, which take weights of words into consideration. If a word gets a lower weight, the number of this word will be discounted more strongly in documents. Therefore, words with lower weights will have less negative effect on the results of LDA.

To explain our generative process, we describe the procedure of generating words from one topic in Figure 2. The urn represents the word distribution of a topic. Each ball has a mark number, which corresponds to a word in the vocabulary. The number of the balls is proportional to the number of words in the topic, while the size of balls represents its weights. In traditional LDA, each ball is considered having the same size (shown in Figure 2 (a)). In our proposed process, the sizes of balls are varied according to their weights (shown in Figure 2 (b)). The process of generating a word from a topic is as follows. Firstly, a ball is selected from the urn with the same process as traditional model. Secondly, we conduct a random choice to decide whether to put this ball into the document or not. Balls with large size are more likely to be put into the document. As the example shown in Figure 2 (b), balls '1' and '4' are smaller and are less likely to be put into the document.

4 Term Weighting LDA

According to the proposed generative process, we propose a new topic model called Term Weighting LDA (TWLDA). Section 3 has shown that words with lower weights generally have weaker negative effect on results of LDA. Since topic-indiscriminate words negatively affect the results of LDA, we expect to find out a way to give these words relatively low weights. In our work, we use supervised term weighting schemes to calculate weights of words. Supervised term weighting schemes are widely applied to measure the the importance of words in different categories in text categorization (Wang et al., 2015). We regard the topics as categories in documents. Topic-indiscriminate words, which scatter across many topics, will be considered unimportant and get relatively low weights by supervised schemes. However, the topics of words are unknown in the beginning. In order to obtain topics of words, an additional step is conducted before we calculate weights of words. In this step, we execute a topic model. This topic model can be standard LDA or other topic models, such as PLSI (Hofmann, 1999) and so on, which can find out the topics of words in documents. In summary, the proposed TWLDA consists of four main processes, which are shown as follows:

- Step 1: $\overrightarrow{\varphi'} \longleftarrow TopicModel()$
- Step 2: $\overrightarrow{\sigma} \longleftarrow Calculate(\overrightarrow{\varphi'})$
- Step 3: Discounting the number of words
- Step 4: Executing *xLDA* with the discounted values

In Step 1, a topic model is executed. Then a topic-word distribution $\overrightarrow{\varphi'}$ is generated by a this topic model.

In Step 2, according to the $\overrightarrow{\varphi'}$, we apply a supervised term weighting scheme to calculate weights of words $\overrightarrow{\sigma}$. Since supervised schemes have the ability to measure the topic discriminating power of words, in principle, all the supervised term weighting schemes can be applied here.

Step 3 is to discount the number of words by their weights. The number of words is diminished proportionally according to weights of words. Hence, the total discounted number of words in document m under topic k is calculated as follows:

$$n_m'^{(k)} = \sum_{t=1}^{t=V} \sigma_t n_{mkt} \tag{1}$$

where σ_t denotes the weight of word t, which is ranging from 0 to 1. n_{mkt} is the number of word t belonging to topic k in document m. Similarly, the total discounted number of word t under topic k is calculated as follows:

$$n_k'^{(t)} = \sum_{m=1}^{m=M} \sigma_t n_{mkt} \tag{2}$$

Step 4 is to execute the standard LDA or its variants, denoted as xLDA, using the discounted values calculated in Equations 1 and 2. Generally, xLDA can be standard LDA or its variants, such as GKLDA (Chen et al., 2013). The main procedures are the same as xLDA. We take standard LDA for example. In Gibbs Sampling process (Chatterji and Pachter, 2004), conditional probability of word t in document m under topic k is calculated using the following formula:

$$p(z_i = k|\overrightarrow{z}_{k,\neg i}, \overrightarrow{w}, \overrightarrow{\alpha}, \overrightarrow{\beta}) = \frac{n_{m,\neg i}'^{(k)} + \alpha_k}{\sum_{k=1}^{k=K}(n_{m,\neg i}'^{(k)} + \alpha_t)} \frac{n_{k,\neg i}'^{(t)} + \beta_k}{\sum_{t=1}^{t=V}(n_{k,\neg i}'^{(t)} + \beta_t)} \tag{3}$$

where $\overrightarrow{\alpha}$ and $\overrightarrow{\beta}$ are hyperparameters of the model. Equation 3 is mostly the same as the formula in the Gibbs Sampling process of traditional LDA (Geman and Geman, 1984). The difference is that those counting variables are replaced with the discounted values, such as $n_{m,\neg i}'^{(k)}$ and $n_{k,\neg i}'^{(t)}$, calculated in Step 3. Equation 3 shows that word t will have less probability to be assigned in topic k if the weight of word t is lower. As a result, words with lower weights will get lower ranking in topics. Hence, the negative effect on results of LDA caused by topic-indiscriminate words will be alleviated if their weights are relatively low. By replacing with the discounted values, other variants of LDA can also be executed in Step 4.

5 Experiment

In this section, we conduct experiments to verify the effectiveness of TWLDA. In the first experiment, we apply the following supervised term weighting schemes in TWLDA: $iqf \cdot qf \cdot icf$ and bdc. We also use unsupervised term weighting schemes $tf \cdot idf$ for comparison. Besides, we will compare the performance of TWLDA with standard LDA and WLDA, which is proposed in (Wilson and Chew, 2010). In our second experiments, we test the performance of TWLDA, WLDA and standard LDA if we do not delete stop words in the pre-processing step.

5.1 Datasets and Pre-processing

Datasets: We use two datasets in our experiments. 'dataset1' consists of online reviews from Amazon. There are totally 39,554 reviews mixed together. 'dataset2' has been used in (Chen et al., 2013), which consists of 8,958 reviews about camera and phone. We obtain it in the website [1].

Pre-processing: Reviews in 'dataset1' are preprocessed as follow. Firstly, words are converted into lower case, and the words with upper or lower case are treated as the same words. Secondly, all punctuations in documents are eliminated and only those alphabetic and numeric characters can be retained. Thirdly, we perform stemming and remove the stop words. In this work, we only use nouns,

[1] https://github.com/czyuan/GKLDA

adjective, verb and adverb. Besides, we do not preprocess $dataset2$ for it has been pre-processed in (Chen et al., 2013).

Parameter Setting: The iteration of TWLDA is set to 2500, which consists of 1000 iterations for the topic model in Step 1 and 1500 iterations for xLDA in Step 4 in Section 4. We set the iterations of preceding LDA model to 1000 due to the reason that most of topic models will converge within 1000 iterations in both two datasets which are used in our experiments. For the reason that small changes of α and β will not affect the results much (Jo and Oh, 2011; Titov and McDonald, 2008), we set $\alpha = 1$ and $\beta = 0.1$ as the setting in (Chen et al., 2013). The value of topic number K is fixed to 20.

5.2 Evaluation Metrics

In this section, we use two ways to evaluate the performance of our proposed model, one is quantitative evaluation and the other is qualitative evaluation. In quantitative evaluation, we use the Topic Coherence (Mimno et al., 2011) and $Precision@n$ as our evaluation metric. Topical Coherence (Mimno et al., 2011) is a metric commonly used to evaluate the performance of LDA, since it shows a well consistence with the judgement of human beings. In (Arora et al., 2012; Brody and Elhadad, 2010; Chen et al., 2014), Topical Coherence is used to compare the performance of different topic models. The better performance of topic model will get higher score in Topic Coherence. We also use $Precision@n$ (or $p@n$), a commonly used metric in information retrieval (Mukherjee and Liu, 2012; Zhao et al., 2010), for evaluation. Top words are more important in topic models, and we set *n* to 5, 10, 15 and 20. We ask two judges to label top 20 words in topics. Each topic is labeled as *correct* if it had more than half of its words related to each other; otherwise *incorrect*. Then, we asked these two judges to label each word of the top 20 words in topics which are labeled good. Since judges already had the conception of each topic in mind, each word was labeled *correct* if it consisted with the concept of the topic; otherwise *incorrect*. We use $p@n$ in two experiments shows in Section 5.3 and Section 5.4. Cohen's Kappa score for word labeling is showed in Table 2, which indicates high agreements between two judges with all the scores larger than 0.8 according to scale in (Landis and Koch, 1977).

Table 2: Cohen's Kappa for agreements of judges

	Topic Labeling	**Word Labeling**			
		p@5	p@10	p@15	p@20
Dataset1	0.858	0.806	0.830	0.879	0.859
Dataset2	0.832	0.842	0.875	0.892	0.872

Table 3: The number of correct topics

	dataset1	dataset2
bdc-TWLDA	15	14
WLDA	9	12
tf-idf-TWLDA	10	12
standard LDA	9	10
iqf-TWLDA	13	13

5.3 Comparison of Exiting LDA & TWLDA with Different Term Weighting Schemes

Different term weighting schemes in TWLDA can result in different performance. In our experiment, we firstly compare the performance of TWLDA using different state-of-the-art supervised term weighting schemes, such as $iqf \cdot qf \cdot icf$ and bdc. We also use $tf \cdot idf$ for comparision. We denote TWLDA using these schemes as tf-idf-TWLDA, iqf-TWLDA and bdc-TWLDA. Furthermore, we compare TWLDA with standard LDA and WLDA.

Quantitative Evaluation: For the reason that the process of Gibbs Sampling is random, we will get different results each time we run the model. We executed each model for 10 times, and calculate the average Topic Coherence value in each iteration. The results of the standard LDA using different term weighting schemes are shown in Figure 3. Figure 4 shows the average $precision@n$ of all good topics over two datasets, while Table 5.2 shows the number of correct topics. We find that:

- From the Topic Coherence results results, bdc-TWLDA and iqf-TWLDA outperform standard LDA and WLDA in both two datasets. bdc-TWLDA, which is an entropy-based scheme, performs the best. On the contrary, the results of TWLDA get worse when it apply tf-idf.

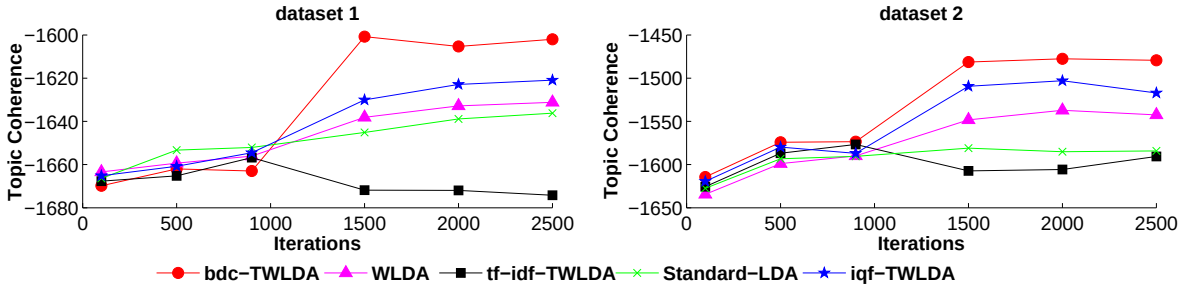


Figure 3: Comparison of TWLDA (xLDA is standard LDA) with different term weighting schemes on Topic Coherence Evaluation

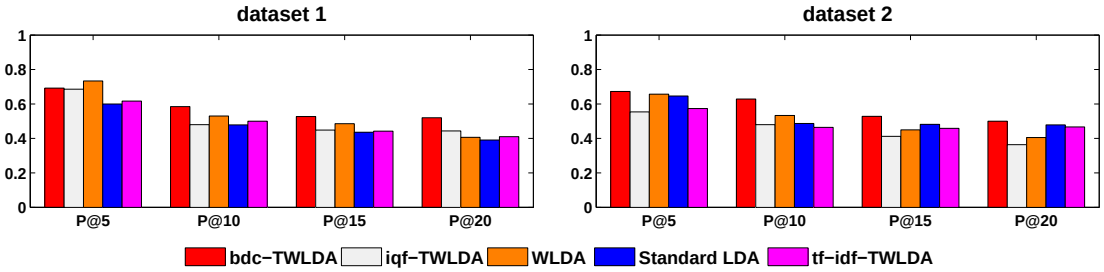


Figure 4: Average Precision@n (P@n) of coherent topics from two datasets

- From the $Precision@n$ results in Figure 4 and correct topic number in Table 5.2, bdc-TWLDA performs best and improve standard LDA by more than 10%. bdc-TWLDA generates most correct topics in both datasets, while iqf-TWLDA ranks second. Although iqf-TWLDA performs worse than WLDA in dataset2 in $Precision@n$ score, it performs better than WLDA in dataset1.

- In general, entropy-based term weighting scheme bdc performs best in both datasets. It corresponds to the experimental result of Wang et al. which shows that the bdc performs better than $iqf \cdot qf \cdot icf$. Supervised scheme $iqf \cdot qf \cdot icf$ also performs better than standard LDA at most of cases. However, tf-idf-LDA gets the worst results in both datasets.

Qualitative Results: Table 4 shows the qualitative results of LDA and bdc-TWLDA in two datasets. We choose the top 5 words of each topic generated respectively by LDA and bdc-TWLDA. We ask two judges to mark those 'bad' topics which are un-interpretable by human into red color. Although the labeling of topics may be subjective, we tried our best to have the consensus between two human judges. As the results shown in Table 4, standard LDA has 11 un-interpretable topics, while there are only 6 interpretable topics in the result of bdc-TWLDA. Furthermore, there are topic-indiscriminate words scattering across several topics, such as 'phone', 'time' and 'word'. In the results of dataset2, there are 10 uninterpretable topics in standard LDA and only 6 topics in bdc-TWLDA. We do not present the results of dataset2 for the limitation of space in our paper. We also do not show the results of WLDA here, which have 11 and 9 un-interpretable topics in dataset1 and dataset2 respectively. Overall, we can see that TWLDA shows higher performance than the standard LDA.

5.4 Performance of TWLDA without Eliminating Stop Words

To demonstrate that our approach also has good performance even though we do not eliminate stop words in the preprocessing step, we execute bdc-TWLDA, WLDA and standard LDA in the following situation: eliminating stop words and retain stop words. In this experiment, we only use dataset1, since dataset2 has been pre-processed and all the stop words have been deleted. We asked two judges to label correct topics (the labeling criteria are introduced in Section 5.2). The Cohen's Kappa value of these two judges are 0.891, which indicates they achieve high agreements. Figure 5 shows the number of correct topics in

Table 4: Quality comparison between standard LDA and bdc-TWLDA

(a) Standard-LDA

topic	word
0	**canon,well,digital,nikon,point**
1	**read,reading,games,video,videos**
2	ipad,mini,size,screen,display
3	ipad,apple,mini,love,product
4	phone,samsung,galaxy,nexus,android
5	battery,life,phone,long,time
6	**screen,phone,back,case,glass**
7	**easy,user,set,features,settings**
8	headphones,ear,sound,quality,buds
9	**amazon,price,google,buy,well**
10	**phone,buy,apple,know,back**
11	**ipad,mini,purchase,happy,product**
12	**phone,recommend,android,best,highly**
13	video,focus,mode,pictures,auto
14	bought,love,gift,loves,old
15	**time,easy,love,size,small**
16	sound,bass,headphones,price,quality
17	pictures,lens,quality,canon,zoom
18	**apps,ipad,apple,touch,free**
19	**month,plan,storage,working,work**

(b) $bdc - TWLDA$

topic	word
0	battery,life,memory,storage,gb
1	**recommend,product,highly,arrived,wifi**
2	sound,bass,music,headphones,hear
3	**ipad,mini,kindle,fire,set**
4	lens,canon,mm,picture,zoom
5	apps,wifi,internet,download,email
6	display,retina,muy,responsive,deal
7	size,small,carry,weight,hand
8	pictures,takes,quality,shots,zoom
9	nexus,google,phone,android,version
10	**money,amazon,wait,return,months**
11	ear,sony,buds,headphones,pair
12	happy,choice,glad,purchase,satisfied
13	manual,mode,video,settings,auto
14	**charge,half,phone,charging,search**
15	apple,products,ios,system,devices
16	canon,nikon,dslr,shoot,lens
17	gift,bought,card,loves,christmas
18	**reviews,front,know,piece,mind**
19	wifi,internet,data,home,web

different models when they eliminate and retain stop words. The number of correct topic in all the three models experiences a fall if they retain stop words, especially standard LDA which decreases from 9 to 2. We can also find that bdc-TWLDA still has high performance when it retain stop words.

Table 5: The number of correct topics in different models

Models	Eliminate stop words	Retain stop words	percentage of decrease
bdc-TWLDA	15	13	13%
WLDA	9	5	44%
Standard LDA	9	2	78%

5.5 Experimental Results Discussion

Our experiments show that the performance of TWLDA depends on the term weighting schemes we choose. The reason is that the capacities of different schemes measuring topic discriminating power are different. Entropy-based schemes like bdc perform the best. In information theory, words which are scattered in most of topics have larger entropy. The entropy of a word can well indicate those topic-indiscriminate words. We get the conclusion that entropy-based term weighting schemes are effective in TWLDA. In the experiments, supervised term weighting schemes outperform unsupervised term weighting schemes in TWLDA. Both bdc and $iqf \cdot qf \cdot icf$ perform better than the standard LDA, while $tf \cdot idf$ perform worse than standard LDA. The reason is that unsupervised term weighting schemes can just measure the document discriminating power of words, other than topic discriminating power.

6 Conclusions

In this paper, we firstly explore topic discriminating power of words in LDA. We observe that topics perform worse if they contain words with low topic discriminating power. These topic-indiscriminate words have negative effects on the results of LDA. In order to solve these problems, we proposed a new model called TWLDA. TWLDA can apply different supervised term weighting schemes to give topic discriminating words relatively low weights in LDA or variants of LDA. The results show that TWLDA has a significant performance while applying supervised term weighting schemes like *bdc*. The number of topic-indiscriminate words is reduced in topics generated by TWLDA with *bdc*.

7 Acknowledgements

This work is supported by National Natural Science Foundation of China (project no. 61300137, Tip-top Scientific and Technical Innovative Youth Talents of Guangdong special support program (No. 2015TQ01X633), Science and Technology Planning Project of Guangdong Province (No. 2013B010406004).

References

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 25–32. ACM.

Sanjeev Arora, Rong Ge, Yoni Halpern, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. 2012. A practical algorithm for topic modeling with provable guarantees. *arXiv preprint arXiv:1212.4777*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Samuel Brody and Noemie Elhadad. 2010. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 804–812. Association for Computational Linguistics.

Nicola Burns, Yaxin Bi, Hui Wang, and Terry Anderson. 2012. Extended twofold-lda model for two aspects in one sentence. In *Advances in Computational Intelligence*, pages 265–275. Springer.

Sourav Chatterji and Lior Pachter. 2004. Multiple organism gene finding by collapsed gibbs sampling. In *Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*, pages 187–193. ACM.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Discovering coherent topics using general knowledge. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 209–218. ACM.

Zhiyuan Chen, Arjun Mukherjee, and Bing Liu. 2014. Aspect extraction with automated prior knowledge learning. In *Proceedings of ACL*, pages 347–358.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741.

Gregor Heinrich. 2005. Parameter estimation for text analysis. Technical report, Technical report.

Gregor Heinrich. 2009. A generic approach to topic models. In *Machine Learning and Knowledge Discovery in Databases*, pages 517–532. Springer.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM.

Jagadeesh Jagarlamudi, Hal Daumé III, and Raghavendra Udupa. 2012. Incorporating lexical priors into topic models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 204–213. Association for Computational Linguistics.

Yohan Jo and Alice H Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824. ACM.

Youngjoong Ko. 2012. A study of term weighting schemes using class information for text classification. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 1029–1030. ACM.

Man Lan, Chew Lim Tan, Jian Su, and Yue Lu. 2009. Supervised and traditional term weighting methods for automatic text categorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4):721–735.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Edda Leopold and Jörg Kindermann. 2002. Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*, 46(1-3):423–444.

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 262–272. Association for Computational Linguistics.

Arjun Mukherjee and Bing Liu. 2012. Aspect extraction through semi-supervised modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 339–348. Association for Computational Linguistics.

Jiaul H Paik. 2013. A novel tf-idf weighting scheme for effective ranking. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 343–352. ACM.

Xiaojun Quan, Liu Wenyin, and Bite Qiu. 2011. Term weighting schemes for question categorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):1009–1021.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th international conference on World Wide Web*, pages 111–120. ACM.

Tao Wang, Yi Cai, Ho-fung Leung, Zhiwei Cai, and Huaqing Min. 2015. Entropy-based term weighting schemes for text categorization in VSM. In *27th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2015, Vietri sul Mare, Italy, November 9-11, 2015*, pages 325–332.

Andrew T Wilson and Peter A Chew. 2010. Term weighting schemes for latent dirichlet allocation. In *human language technologies: The 2010 annual conference of the North American Chapter of the Association for Computational Linguistics*, pages 465–473. Association for Computational Linguistics.

Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a maxent-lda hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 56–65. Association for Computational Linguistics.

Textual Entailment with Structured Attentions and Composition

Kai Zhao and **Liang Huang** and **Mingbo Ma**
School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, Oregon, USA
{kzhao.hf, lianghuang.sh, cosmmb}@gmail.com

Abstract

Deep learning techniques are increasingly popular in the textual entailment task, overcoming the fragility of traditional discrete models with hard alignments and logics. In particular, the recently proposed attention models (Rocktäschel et al., 2015; Wang and Jiang, 2015) achieves state-of-the-art accuracy by computing soft word alignments between the premise and hypothesis sentences. However, there remains a major limitation: this line of work completely ignores syntax and recursion, which is helpful in many traditional efforts. We show that it is beneficial to extend the attention model to tree nodes between premise and hypothesis. More importantly, this subtree-level attention reveals information about entailment relation. We study the recursive composition of this subtree-level entailment relation, which can be viewed as a soft version of the Natural Logic framework (MacCartney and Manning, 2009). Experiments show that our structured attention and entailment composition model can correctly identify and infer entailment relations from the bottom up, and bring significant improvements in accuracy.

1 Introduction

Automatically recognizing sentence entailment relations between a pair of sentences has long been believed to be an ideal testbed for discrete approaches using alignments and rigid logic inferences (Zanzotto et al., 2009; MacCartney and Manning, 2009; Wang and Manning, 2010; Watanabe et al., 2012; Tian et al., 2014; Filice et al., 2015). All of these methods are based on sparse features, making them brittle for unseen phrases and sentences.

Recent advances in deep learning reveal another promising direction to solve this problem. Instead of discrete features and logics, continuous representation of the sentence is more robust to unseen features without sacrificing performance (Bowman et al., 2015). In particular, the attention model based on LSTM can successfully identify the word-by-word correspondences between the two sentences that lead to entailment or contradiction, which makes the entailment relation inference more focused on local information and less vulnerable to misleading information from other parts of the sentence (Rocktäschel et al., 2015; Wang and Jiang, 2015).

However, conventional neural attention models for entailment recognition problem treat sentences as sequences, ignoring the fact that sentences are formed from the bottom up with syntactic tree structures, which inherently associate with the semantic meanings. Thus, using the tree structure of the sentences will be beneficial in inducing the entailment relations between parts of the two sentences, and then further improving the sentence-level entailment relation classification (Watanabe et al., 2012).

Furthermore, as MacCartney and Manning (2009) point out, the entailment relation between sentences is modular, and can be modeled as the composition of subtree-level entailment relations. These subtree-level entailment relations are induced by comparing subtrees between the two sentences, which are by nature a perfect match to be modeled by the attention model over trees.

In this paper we propose a recursive neural network model that calculates the attentions following the tree structures, which helps determine entailment relations between parts of the sentences. We model the entailment relation with a continuous representation.The relation representations of non-leaf nodes are recursively computed by composing their children's relations. This approach can be viewed as a *soft* version of Natural Logic (MacCartney and Manning, 2009) for neural models, and can make the recognized entailment relation easier to interpret.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2248–2258, Osaka, Japan, December 11-17 2016.

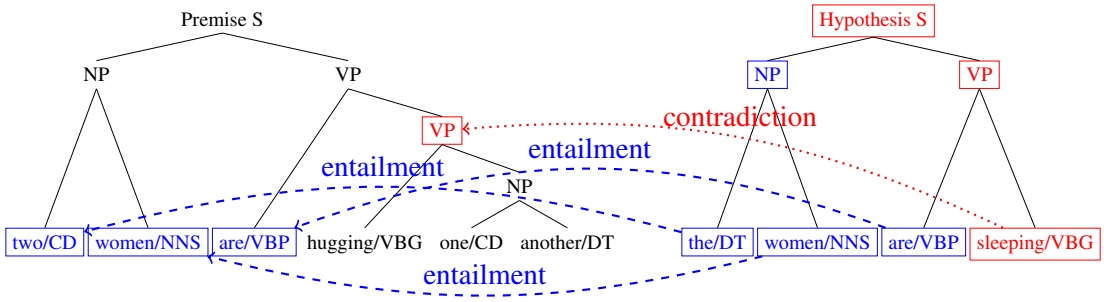


Figure 1: Exemplary trees for the premise sentence "two women are hugging one another" and the hypothesis sentence "the women are sleeping". The syntactic labels (NP, VP, CD, etc.) are not used in the model. The dashed and dotted lines show the lowest level of alignments from the hypothesis tree nodes to the premise tree nodes. The blue dashed lines mark the entailment relations, and the red dotted line marks the contradiction relation. In the hypothesis tree, tree nodes in blue squares are identified to be entailment from the premise, and nodes in red squared are identified to contradicts the premise. By composing these relations from the bottom up, we reach a conclusion that the sentence-level entailment relation is contradiction. Please also refer to Figure 5 for real examples taken from our experiments.

We make the following contributions:

1. We adapt the sequence attention model to the tree structure. This attention model directly works on meaning representations of nodes in the syntactic trees, and provides a more precise guidance for subtree-level entailment relation inference. (Section 2.2)

2. We propose a continuous representation for entailment relation that is specially designed for entailment composition over trees. This entailment relation representation is recursively composed to induce the overall entailment relation, and is easy to interpreted. (Section 2.3)

3. Inspired by the forward and reverse alignment technique in machine translation, we propose dual-attention that considers both the premise-to-hypothesis and hypothesis-to-premise directions, which makes the attention more robust to confusing alignments. (Section 2.4)

4. Experiments show that our model brings significant performance boost based on a Tree-LSTM model. Our dual-attention can provide superior guidance for the entailment relation inference (Figure 4). The entailment composition follows the intuition of Nature Logic and can provide a vivid illustration of how the final entailment conclusion is formed from bottom up (Figure 5). (Section 4)

2 Structured Attentions & Entailment Composition

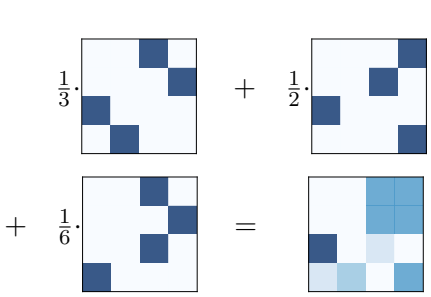
Here we first give an overview and formalization of our model, and then describe its components.

2.1 Formalization

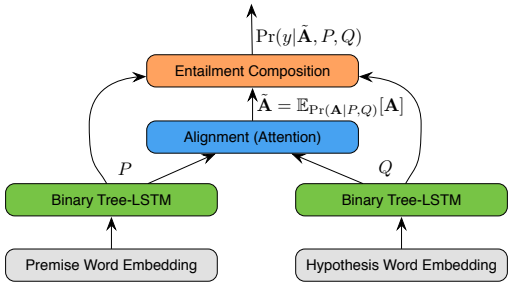
We assume both the premise tree and the hypothesis tree are binarized.

We use the premise tree and hypothesis tree in Figure 1 to demonstrate the process of our approach. The premise sentence is "two women are hugging one another", and the hypothesis sentence is "the women are sleeping".

Following the traditional approaches (MacCartney and Manning, 2009; Watanabe et al., 2012), we first find the alignments from hypothesis tree nodes to premise tree nodes (i.e., the dashed or dotted curves in Figure 1). Then we explore inducing the sentence-level entailment relations by 1) first computing the entailment relation at each node of the hypothesis tree based on the alignments, and then 2) composing the entailment relations at the internal hypothesis nodes from bottom up to the root in a recursive way. Our model resembles the work of Natural Logic (MacCartney and Manning, 2009) in the spirit that the entailment relation is inferred *modularly*, and composed *recursively*.



(a) Expected alignment over 3 alignments with probability distribution of $(\frac{1}{3}, \frac{1}{2}, \frac{1}{6})$. Each alignment is a matrix of $\{0,1\}^{4\times4}$, and the expected alignment is a matrix of $\mathbb{R}^{4\times4}$.



(b) Architecture of our neural network. Layers with the same color share the same parameters. The alignment module calculates the expected alignment $\tilde{\mathbf{A}}$. The Entailment Composition module infers the final entailment y.

Figure 2: Expected alignments calculation (a) and overview of the network architecture (b).

We formalize this entailment task as a structured prediction problem similar to Mnih et al. (2014), Ba et al. (2015), and Xu et al. (2015). The inputs are two trees: premise tree P, and hypothesis tree Q. The goal is to predict a label $y \in \{\text{contradiction}, \text{neutral}, \text{entailment}\}$. Note that although the output label y is not structured, we can still consider the problem as a structured prediction problem, because: 1) the input is a pair of trees; and 2) the internal alignments are structured.

More formally, we aim to minimize the negative log likelihood of the gold label given the two trees. The objective can be written in the online fashion as:

$$\ell = -\log \Pr(y|P,Q) = -\log \sum_{\mathbf{A}} \Pr(y, \mathbf{A}|P,Q)$$

$$= -\log \sum_{\mathbf{A}} \Pr(\mathbf{A}|P,Q) \cdot \Pr(y|\mathbf{A},P,Q) = -\log \mathbb{E}_{\Pr(\mathbf{A}|P,Q)}[\Pr(y|\mathbf{A},P,Q)],$$

where the structured latent variable $\mathbf{A} \in \{0,1\}^{|Q|\times|P|}$ represents an alignment. $|\cdot|$ is the number of nodes in the tree. $\mathbf{A}_{ij} = 1$ if and only if node i in Q is aligned to node j in P, otherwise $\mathbf{A}_{ij} = 0$.

However, enumerating over all possible alignments $\mathbf{A}$ takes exponential time, we need to efficiently approximate the above log expectation.

Fortunately, as Xu et al. (2015) point out, as long as the calculation $\Pr(y|\mathbf{A},P,Q)$ only consists of linear calculation, simple nonlinearities like $\tanh$, and softmax, we can have following simplification via first-order Taylor approximation:

$$\ell = -\log \mathbb{E}_{\Pr(\mathbf{A}|P,Q)}[\Pr(y|\mathbf{A},P,Q)] \approx -\log \Pr(y|\mathbb{E}_{\Pr(\mathbf{A}|P,Q)}[\mathbf{A}],P,Q)],$$

which means instead of enumerating over all alignments and calculating the label probability for each alignment, we can use the label probability for the expected alignment as an approximation:[1]

$$\tilde{\mathbf{A}} \triangleq \mathbb{E}_{\Pr(\mathbf{A}|P,Q)}[\mathbf{A}] \quad \in \mathbb{R}^{|Q|\times|P|} \tag{1}$$

Figure 2a shows an example of expected alignment calculation. The objective is simplified to

$$\ell \approx -\log \Pr(y|\tilde{\mathbf{A}},P,Q). \tag{2}$$

With this observation, we split our calculation into two steps as the top two modules in Figure 2b. First in the Alignment module, we calculate the expected alignments $\tilde{\mathbf{A}}$ using Equation 1 (Section 2.2). Then we calculate the node-wise entailment relation, propagate and compose the relation from bottom up to find out the final entailment relation (Equation 2) in the Entailment Composition module (Section 2.3). Both of these two modules rely on the composition of tree node meaning representations (Section 3).

2.2 Attention over Tree Nodes

First we calculate the expected alignments $\tilde{\mathbf{A}}$ between the hypothesis Q and the premise P (Equation 1):

$$\tilde{\mathbf{A}} = \mathbb{E}_{\Pr(\mathbf{A}|P,Q)}[\mathbf{A}].$$

[1]We use bold letter, $\mathbf{A}$, for binary alignments, and tilde version, $\tilde{\mathbf{A}}$, for the expected alignments in the real number space.

To simplify the calculation, we further approximate the global (binary) alignment $\mathbf{A}$ to be consisted of the alignment $\mathbf{A}_i \in \{0,1\}^{1\times|P|}$ of each tree node $i \in Q$ independently. $\mathbf{A}_i$ is the ith row of $\mathbf{A}$:

$$\mathbf{A} = [\mathbf{A}_1^T; \mathbf{A}_2^T; \ldots; \mathbf{A}_{|Q|}^T]^T,$$

$$\Pr(\mathbf{A}|P,Q) = \prod_i^{|Q|} \Pr(\mathbf{A}_i|P,Q).$$

$\Pr(\mathbf{A}_{i,j} = 1|P,Q)$ is the probability of the node $i \in Q$ being aligned to node $j \in P$, which is defined as:

$$\Pr(\mathbf{A}_{i,j} = 1|P,Q) \triangleq \frac{\exp(T_{2k,1}([\mathbf{h}_i;\mathbf{h}_j]))}{\sum_k \exp(T_{2k,1}([\mathbf{h}_i;\mathbf{h}_k]))}. \quad (3)$$

$\mathbf{h}_i, \mathbf{h}_j \in \mathbb{R}^k$ are vectors representing the semantic meanings of node i, j, respectively, whose calculation will be described in Section 3. $T_{2k,1}$ is an affine transformation from $\mathbb{R}^{2k}$ to $\mathbb{R}$. This formulation essentially is equivalent to the widely used attention calculation in neural networks (Bahdanau et al., 2014), i.e., for each node $i \in Q$, we find the relevant nodes $j \in P$ and use the softmax of the relevances as a probability distribution. In the rest of the paper, we use "expected alignment" and "attention" interchangeably.

The expected alignment of node i being aligned to node j, by definition, is:

$$\tilde{\mathbf{A}}_{i,j} = \Pr(\mathbf{A}_{i,j} = 1|P,Q) \cdot 1 = \Pr(\mathbf{A}_{i,j} = 1|P,Q).$$

2.3 Entailment Composition

Now we can calculate the entailment relation at each tree node and propagate the entailment relation following the hypothesis tree from bottom up, assuming the expected alignment is given (Equation 2):

$$\ell \approx -\log \Pr(y|\tilde{\mathbf{A}}, P, Q).$$

Let vector $\mathbf{e}_i \in \mathbb{R}^r$ denote the entailment relation in a latent relation space at hypothesis tree node $i \in Q$. At the root of the hypothesis tree. We can induce the final entailment relation from entailment relation vector $\mathbf{e}_{\text{root}}$. We use a simple $\tanh$ layer to project the entailment relation to the 3 relations defined in the task, and use a softmax layer to calculate the probability for each relation:

$$\Pr(y|\tilde{\mathbf{A}}, P, Q) = \text{softmax}(\tanh(T_{r,3}(\mathbf{e}_{\text{root}}))).$$

At each hypothesis node i, $\mathbf{e}_i$ is calculated recursively given the meaning representation at this tree node $\mathbf{h}_i$, the meaning representation of every node in the premise tree $\mathbf{h}_j, j \in P$, and the entailment from i's children, $\mathbf{e}_{i,1}, \mathbf{e}_{i,2}$:

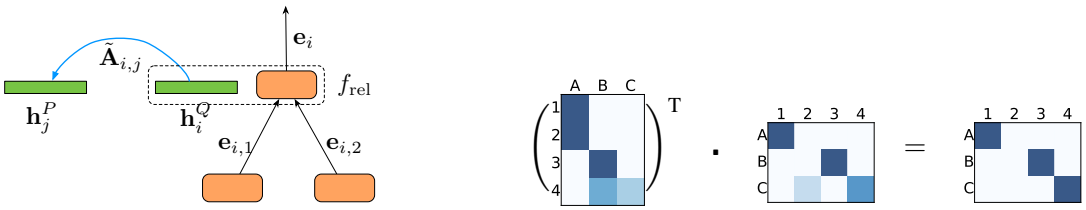
$$\mathbf{e}_i = f_{\text{rel}}([\mathbf{h}_i; \sum_{j\in P} \tilde{\mathbf{A}}_{i,j}\mathbf{h}_j], \mathbf{e}_{i,1}, \mathbf{e}_{i,2}) \quad (4)$$

Figure 3a illustrates the calculation of the entailment composition. We will discuss f_{rel} in Section 3.

2.4 Dual-attention Over Tree Nodes

We can further improve our alignment approximation in Section 2.2, which does not consider any structural information of current tree, nor any alignment information from the premise tree.

We can take a closer look at our conceptual example in Figure 1. Note that the alignments have, to some extent, a symmetric property: if a premise node j is most relevant to a hypothesis node i, then the hypothesis node i should also be most relevant to premise node j. For example, in Figure 1, the premise phrase "hugging one another" contradicts the hypothesis word "sleeping". In the perspective of the premise tree, the hypothesis word "sleeping" contradicts by the known claim "hugging one another". This suggests us to calculate the alignments from both side, and eliminate the unlikely alignment if it only exists in one side. This technique is similar to the widely used forward and reversed alignment technique in the machine translation area.



(a) Entailment composition at hypothesis tree node i. The composition is based on the meaning representation of current node $\mathbf{h}_i^Q$, the expected alignment for node i, $\tilde{\mathbf{A}}_i$, expected meaning representation of aligned premise tree node $\sum_{j \in P} \tilde{\mathbf{A}}_{i,j} h_j^P$, and known entailment relations from children nodes $\mathbf{e}_{i,1}, \mathbf{e}_{i,2}$.

(b) An example of dual-attention eliminating uncertainty in the alignment. In the left attention matrix, word "4" can be aligned to either "B" or "C". In the middle attention matrix, word "C" can be aligned to either "2" or "4". The element-wise product eliminates these uncertainty and results in the right attention matrix.

Figure 3: Entailment composition (a) and dual-attention calculation (b).

In detail, we calculate the expected alignments $\tilde{\mathbf{A}}$ from hypothesis to premise, and also the expected alignments $\tilde{\mathbf{A}}^R$ from premise to hypothesis, and use their element-wise product

$$\tilde{\mathbf{A}}^* = \tilde{\mathbf{A}} \cdot \tilde{\mathbf{A}}^R$$

as the attention to feed into the Entailment Composition module.[2] This element-wise product is a mimic of the intersection of two alignments in machine translation. Figure 3b shows an example.

In addition to our dual-attention, Cohn et al. (2016) also explore to use the structural information to improve the alignment. However, their approach requires introducing some extra terms in the objective function, and is not straightforward to integrate into our model. We leave adding more structural constraints to further improve the attention as an open problem to explore in the future.

3 Review: Recursive Tree Meaning Representations

Here we describe the final building block of our neural model.

In Section 2.2, we did not mention the calculation of the meaning representation $\mathbf{h}_i$ for node i in Equation 3, which represents the semantic meaning of the subtree rooted at node i. In general, $\mathbf{h}_i$ should be calculated recursively from the meaning representations $\mathbf{h}_{i,1}$, $\mathbf{h}_{i,2}$ of its two children if node i is an internal node, otherwise $\mathbf{h}_i$ should be calculated based on the word $\mathbf{x} \in \mathbb{R}^d$ in the leaf.

$$\mathbf{h}_i = f_{\text{MR}}(\mathbf{x}_i, \mathbf{h}_{i,1}, \mathbf{h}_{i,2}). \tag{5}$$

Similar is Equation 4, where the relation $\mathbf{e}_i$ is recursively calculated from the relation of its two children, as well as the meaning $\mathbf{h}_i$ comparing with the meaning of the premise tree:

$$\mathbf{e}_i = f_{\text{rel}}([\mathbf{h}_i; \sum_{j \in P} \tilde{\mathbf{A}}_{i,j} \mathbf{h}_j], \mathbf{e}_{i,1}, \mathbf{e}_{i,2}). \tag{6}$$

Note the resemblance between these two equations, which indicates that we can handle them similarly with the same form of composition function $f(\cdot)$.

We have various choices for composition function f. For example, we can use simple RNN functions as in Socher et al. (2013). Alternatively, we can use a convolutional layer to extract features from $\mathbf{x}_i, \mathbf{h}_{i,1}, \mathbf{h}_{i,2}$ and use pooling as aggregation to form $\mathbf{h}_i$. In this paper we choose Tree-LSTM model (Tai et al., 2015). Our model is independent to this composition function and any high-quality composition function is sufficient for us to infer the meaning representations and entailments.

Here we use Equation 5 as an example. Equation 6 can be handled similarly. Similar to the classical LSTM model (Hochreiter and Schmidhuber, 1997), in the binary Tree-LSTM model of Tai et al. (2015), each tree node has a state represented by a pair of vectors: the output vector $\mathbf{h} \in \mathbb{R}^{1 \times k}$, and the memory cell $\mathbf{c} \in \mathbb{R}^{1 \times k}$, where k is the length of the Tree-LSTM output representation. We use $\mathbf{h}$ as the meaning

[2] We need to normalize $\tilde{\mathbf{A}}^*$ at each row to make each row a probability distribution.

Method	k	$\|\theta\|_M$	Train	Test
LSTM sent. embedding (Bowman et al., 2015)	100	221k	84.8	77.6
Sparse Features + Classifier (Bowman et al., 2015)	-	-	99.7	78.2
LSTM + word-by-word attention (Rocktäschel et al., 2015)	100	252k	85.3	83.5
mLSTM (Wang and Jiang, 2015)	300	1.9m	92.0	86.1
LSTM-network (Cheng et al., 2016)	450	3.4m	88.5	86.3
LSTM sent. embedding (our implement. of Bowman et al. (2015))	100	241k	79.0	78.4
Binary Tree-LSTM (our implementation of Tai et al. (2015))	100	211k	82.4	79.9
Binary Tree-LSTM + simple RNN w/ attention	150	220k	82.4	81.8
Binary Tree-LSTM + Structured Attention & Composition	150	0.9m	87.0	86.4
+ dual-attention	150	0.9m	87.7	**87.2**

Table 1: Comparison between our structured model with other existing methods. Column k specifies the length of the meaning representations. $|\theta|_M$ is the number of parameters without the word embeddings.

representation of the tree node in the attention model. The LSTM transition calculates the state $(\mathbf{h}_i, \mathbf{c}_i)$ of node i with leaf word $\mathbf{x}_i \in \mathbb{R}^d$, and two children with states $(\mathbf{h}_{i,1}, \mathbf{c}_{i,1})$ and $(\mathbf{h}_{i,2}, \mathbf{c}_{i,2})$ respectively.

We can abuse the mathematics a little bit, and write the transition at an LSTM unit as a function:

$$[\mathbf{h}_i; \mathbf{c}_i] = \mathrm{LSTM}(\mathbf{x}_i, [\mathbf{h}_{i,1}; \mathbf{c}_{i,1}], [\mathbf{h}_{i,2}; \mathbf{c}_{i,2}])$$

In practice, we use the above $\mathrm{LSTM}(\cdot,\cdot,\cdot)$ function as $f_{\text{MR}}(\cdot,\cdot,\cdot)$, and $f_{\text{rel}}(\cdot,\cdot,\cdot)$. But we only expose the output $\mathbf{h}_i$ to the above layers, and keep the memory $\mathbf{c}_i$ visible only to the $\mathrm{LSTM}(\cdot,\cdot,\cdot)$ function.

Following Zaremba et al. (2014), function $\mathrm{LSTM}(\cdot,\cdot,\cdot)$ is summarized by Equations 7-9:

$$\begin{pmatrix} \mathbf{i}_i \\ \mathbf{f}_{i,1} \\ \mathbf{f}_{i,2} \\ \mathbf{o}_i \\ \mathbf{u}_i \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} T_{d+2k,k} \begin{pmatrix} \mathbf{x}_i \\ \mathbf{h}_{i,1} \\ \mathbf{h}_{i,2} \end{pmatrix} \tag{7}$$

$$\mathbf{c}_i = \mathbf{i}_i \odot \mathbf{u}_i + \mathbf{f}_{i,1} \odot \mathbf{c}_{i,1} + \mathbf{f}_{i,2} \odot \mathbf{c}_{i,2}, \tag{8}$$

$$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i), \tag{9}$$

where $\mathbf{i}_i$, $\mathbf{f}_{i,1}$, $\mathbf{f}_{i,2}$, $\mathbf{o}_i$ represent the input gate, two forget gates for two children nodes, and the output gate respectively. $T_{d+2k,k}$ is an affine transformation from $\mathbb{R}^{d+2k}$ to $\mathbb{R}^k$.

4 Empirical Evaluations

We evaluate the performances of our structured attention model and structured entailment model on the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015). The SNLI dataset contains ~ 570k sentence pairs. We use the binarized trees in SNLI dataset in our experiments.

4.1 Experiment Settings

Network Architecture

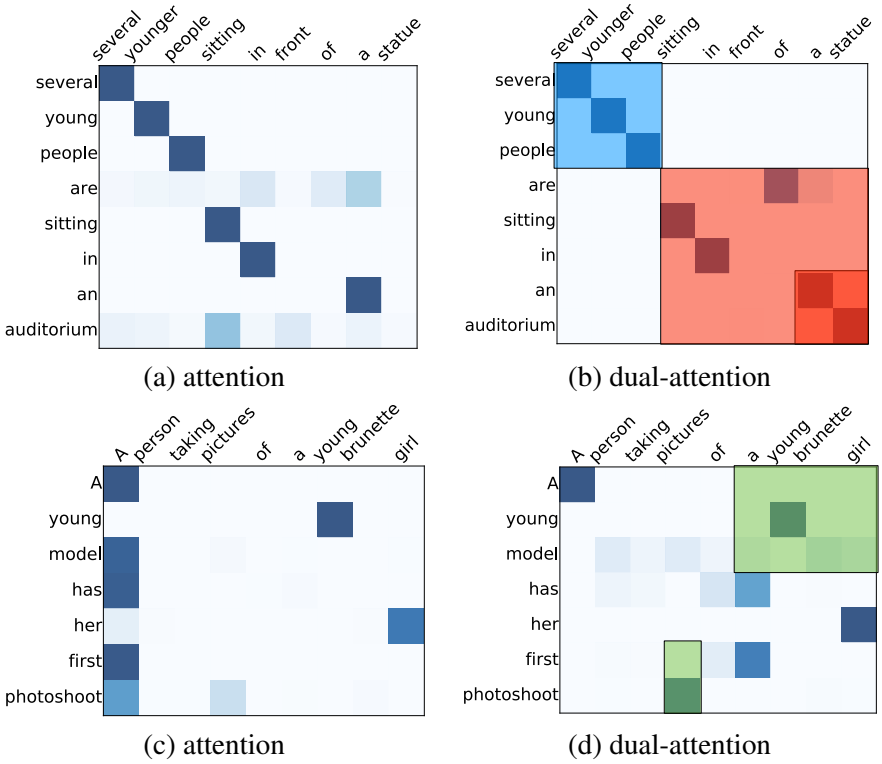
The general structure of our model is illustrated in Figure 2b. We omitted a dropout layer between the word embedding layers and the tree LSTM layers in Figure 2b. We use cross-entropy as the training objective.[3]

Parameter Initialization & Hyper-parameters

We use GloVe (Pennington et al., 2014) to initialize the word embedding layer. In the training we do not change the embeddings, except for the OOV words in the training set. For the parameters of the rest layers, we use a uniform distribution between -0.05 and 0.05 as initialization.

Our model is trained in an end-to-end manner with adam (Kingma and Ba, 2014) as the optimizer. We set the learning rate to 0.001, β_1 to 0.9, and β_2 to 0.999. We use minibatch of size 32 in the training. The dropout rate is 0.2. The length for the Tree-LSTM meaning representation $k = 150$. The length of the entailment relation vector $r = 150$.

[3]Our code is released at `https://github.com/kaayy/structured-attention`.



(a) attention (b) dual-attention

(c) attention (d) dual-attention

Figure 4: Attention matrices for exemplary sentence pairs. Note that, for brevity we *only show* the attentions between each word pair, and skip the attentions of tree nodes. Some important tree node alignments calculated by our model are highlighted using the colored boxes, where the colors of the boxes represent the entailment relations (see Figure 5). (a) (b) Premise: several younger people sitting in front of a statue. Hypothesis: several young people sitting in an auditorium. Dual-attention fixes the misaligned word "auditorium". (c) (d) Premise: A person taking pictures of a young brunette girl. Hypothesis: A young model has her first photoshoot. Dual-attention fixes the uncertain alignments for "photoshoot" and "model".

4.2 Quantitative Evaluation

We present a comparison of structured model with existing methods of LSTM-based sentence embedding (Bowman et al., 2015), LSTM with attention (Rocktäschel et al., 2015), Binary Tree-LSTM sentence embedding (our implementation of Tai et al. (2015)), mLSTM (Wang and Jiang, 2015), and LSTM-network (Cheng et al., 2016) in Table 1.

We first try Binary-Tree LSTM with a composition function f_{rel} of a recurrent network with attention as in Rocktäschel et al. (2015), which achieves an accuracy of 81.8. We find the training of this RNN is difficult due to the vanishing gradient problem.

Using Binary-Tree LSTM for entailment relation composition instead of the simple RNN brings $\sim$4.6 improvement. We observe that the vanishing gradient problem is greatly alleviated. Dual-attention further improves the tree node alignment, achieving another 0.8 improvement.

Our structured entailment composition model outperforms the similar mLSTM model, which essentially also uses an LSTM layer to propagate the "matching" information, but sequentially. With the help of dual-attention, our model outperforms mLSTM with a 1.1 point margin.

4.3 Qualitative Evaluation

Due to space constraints, here we highlight two examples in Figure 4 for both standard attention and dual-attention, and Figure 5 for entailment composition. To pick the most representative examples from the dataset needs careful consideration. Ideally random selection is most convincing. However, due to

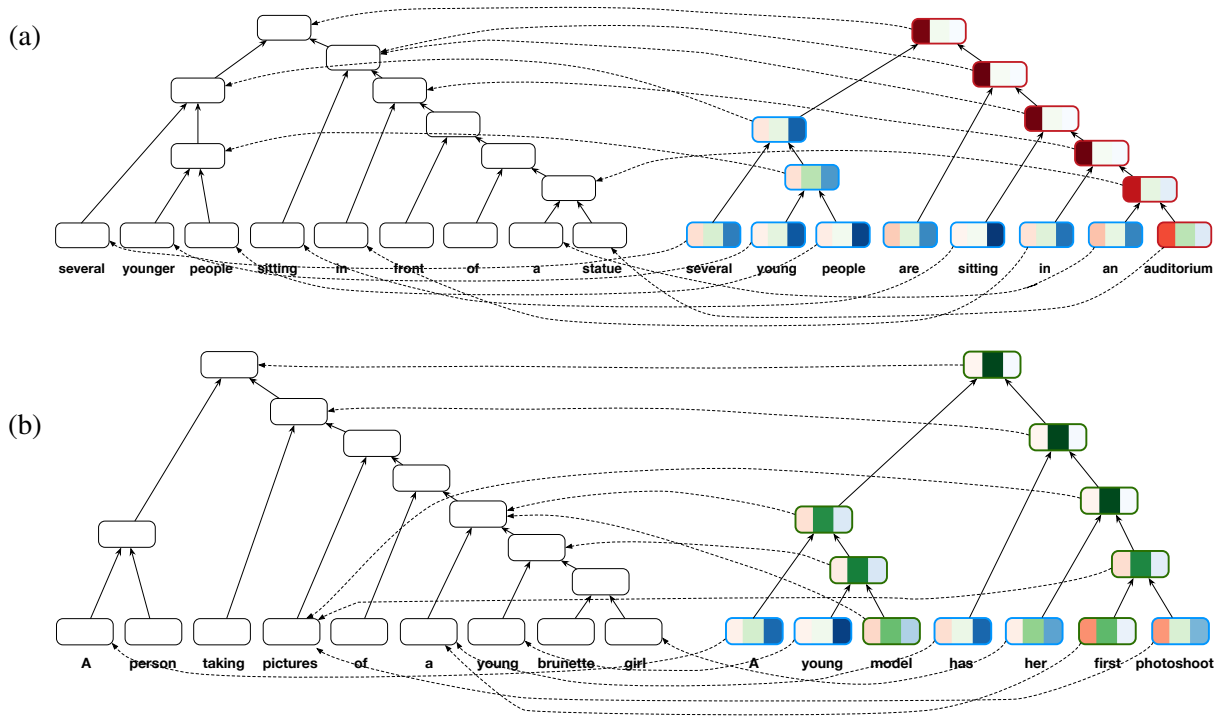


Figure 5: Examples illustrating entailment relation composition. (a) for Figure 4 (b); (b) for Figure 4 (d). For each hypothesis tree node, the dashed line shows to its most confident alignment. The three color stripes in each node indicate the confidences of the corresponding entailment relation estimation: red for contradiction, green for neutral, and blue for entailment. The colors of the node borders show the dominant estimation. Note: there is no strong alignment for hypothesis word "are" in (a).

the fact that most correctly classified examples in the datasets are trivial sentence pairs with only word insertion, deletion, or replacement, and many incorrectly classified examples in the datasets involves common knowledge, (e.g., "waiting in front of a red light" entails "waiting for green light", or "splashing through the ocean" contradicts "is in Kansas",) it is time-consuming to find meaningful insights from randomly selected examples. Here we manually choose two examples from the test set of the SNLI corpus, with consideration of both generality and non-triviality. They both involve complex syntactic structures and compositions of several relations. In addition, some examples that need more subtle linguistic insights are discussed in Section 4.4.

Our first example is shown in Figure 4 (a) and (b), with premise "several younger people sitting in front of a statue", and hypothesis "several young people sitting in an auditorium". Figure 4 (a) and (b) only show the word-level attention for brevity. In this example, note the hypothesis word "auditorium", which has no explicit correspondence in the premise sentence, but indeed has an implicit correspondence "statue" that indicates the conflict relation. The standard attention model aligns "auditorium" to "sitting" since they more frequently co-occur, leading to an incorrect relation of "entailment" (not shown in Figure 5). The dual-attention model correctly finds the alignment between "auditorium" and "statue" since "sitting" is more likely to be aligned to the same word in the premise. The colored boxes in Figure 4 (b) show some important tree node alignment calculated by our model. The colors represent the entailment relation based on the alignment, as shown in Figure 5 (a).

In Figure 5 (a), each tree node is filled with three color stripes, whose darknesses show the confidences of the corresponding entailment relations. For this example, the contradiction relation from "statue" and "auditorium" flips every tree node from bottom up and finally make the final result contradiction, similar to our concept example in Figure 1.

Another example with premise "a person taking pictures of a young brunette girl", and hypothesis "a

young model has her first photoshoot". The word-level attentions are shown in Figure 4 (c) (d). The standard attention is uncertain about two words: 1) word "model" has several meanings, making it hard to find the right alignment, but in the perspective of from premise to hypothesis, it is easier since a girl is more likely to be a model. 2) Similar is for hypothesis word "photoshoot", which can either be aligned to "a" or "pictures" but since "a" is aligned to other words, dual-attention aligns "photoshoot" to "pictures".

In Figure 5 (b), we can see that there are two parts in the hypothesis indicates that the relation should be neutral: 1) "a young brunette girl" is not necessarily a "a young model"; and 2) the "pictures" taken are not necessarily "her first photoshoot".

4.4 Discussion

Although many attention-based models, including our model, achieve superior results in the Stanford Natural Language Inference dataset, we still need to circumvent some problems to apply these neural models to more general textual entailment problems.

Despite those sentence pairs that require more common knowledge to find the entailment relations as we mentioned in Section 4.3, we are more interested in sentences that are difficult because they involve non-trivial linguistic properties.

Consider the following two pairs of sentences that are difficult for current attention and composition based models:

1.
   - Premise: The boy loves the girl.
   - Hypothesis: The girl loves the boy.

   Here the only difference between the two sentences is the order/structure of the words. To handle this problem the attention-based models should take the reordering into consideration when composing entailment relations.

2.
   - Premise: A stuffed animal on the couch.
   - Hypothesis: An animal on the couch.

   In this example, almost every hypothesis word occurs in the premise sentence, but it is difficult to infer that "a stuffed animal" is not "an animal". While in most cases the monotonicity of entailment suggests that a word deletion in the premise sentence either leads to entailment, e.g., "a cute animal" entails "an animal", or a reverse entailment, e.g., "some animal" reverse entails "animal" (See MacCartney and Manning (2009) for more details), but for words like "stuffed" it is quite different: their monotonicity directions depend on the nouns being modified, e.g., "a stuffed animal" does not entails "an animal", but "a stuffed toy" entails "a toy". This observation suggests that we might need to consider phrases like "stuffed animal" as a whole instead of treating the two words separately and then composing the entailment relations.

In addition, training of the neural models rely on large training corpora, which makes it difficult to directly apply neural models on traditional RTE datasets, e.g., the Pascal RTE dataset (Dagan et al., 2006) and the FraCaS dataset (Cooper et al., 1996), which are usually small and contain many named entities that are hard for neural models to identify.

5 Conclusion

We have presented an approach to model the composition of the entailment relation following the tree structure for the sentence entailment task. We adapted the attention model for tree structures. Experiments show that our model bring significant improvements in accuracy, and is easy to interpret.

Acknowledgments

We thank the anonymous reviewers for helpful comments. We are also grateful to James Cross, Dezhong Deng, and Lemao Liu for suggestions. This project was supported in part by NSF IIS-1656051, DARPA FA8750-13-2-0041 (DEFT), and a Google Faculty Research Award.

References

Jimmy Ba, Ruslan R Salakhutdinov, Roger B Grosse, and Brendan J Frey. 2015. Learning wake-sleep recurrent attention models. In *Advances in Neural Information Processing Systems*, pages 2575–2583.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September. Association for Computational Linguistics.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. *arXiv preprint arXiv:1601.01085*.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. Using the framework. Technical report, Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2015. Structural representations for learning relations between pairs of texts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, Beijing, China, July. Association for Computational Linguistics*.

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

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Bill MacCartney and Christopher D Manning. 2009. An extended model of natural logic. In *Proceedings of the eighth international conference on computational semantics*, pages 140–156. Association for Computational Linguistics.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

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

Ran Tian, Yusuke Miyao, and Takuya Matsuzaki. 2014. Logical inference on dependency-based compositional semantics. In *Proceedings of ACL*, pages 79–89.

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*.

Mengqiu Wang and Christopher D Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1164–1172. Association for Computational Linguistics.

Yotaro Watanabe, Junta Mizuno, Eric Nichols, Naoaki Okazaki, and Kentaro Inui. 2012. A latent discriminative model for compositional entailment relation recognition using natural logic. In *COLING*, pages 2805–2820.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

Fabio Massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. 2009. A machine learning approach to textual entailment recognition. *Natural Language Engineering*, 15(04):551–582.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

plWordNet 3.0 – a Comprehensive Lexical-Semantic Resource

Marek Maziarz[A], **Maciej Piasecki**[A], **Ewa Rudnicka**[A],
Stan Szpakowicz[B], **Paweł Kędzia**[A]
[A] Wrocław University of Technology, Wrocław, Poland
[B] University of Ottawa, Ottawa, Ontario, Canada
mawroc@gmail.com, maciej.piasecki@pwr.wroc.pl, ewa.rudnicka78@gmail.com,
szpak@eecs.uottawa.ca, pawel.kedzia@pwr.edu.pl

Abstract

We have released plWordNet 3.0, a very large wordnet for Polish. In addition to what is expected in wordnets – richly interrelated synsets – it contains sentiment and emotion annotations, a large set of multi-word expressions, and a mapping onto WordNet 3.1. Part of the release is enWordNet 1.0, a substantially enlarged copy of WordNet 3.1, with material added to allow for a more complete mapping. The paper discusses the design principles of plWordNet, its content, its statistical portrait, a comparison with similar resources, and a partial list of applications.

1 Introduction

WordNet (Fellbaum, 1998), developed at Princeton University and available on an open licence since the early 1990s, has proven useful in thousands of applications to English texts. It is not flawless, but it strikes a most reasonable balance between the formalisation of the descriptions of lexical meaning and the wide coverage required for practical applications. Wordnets for other languages have been built upon the WordNet blueprint, but almost none of them come close to WordNet's size and coverage. That limits their influence on language technology for those languages. It is therefore unclear whether the success of the "WordNet phenomenon" is not somehow restricted to English. It must also be noted that most of those wordnets have been translated, one way or another, from Princeton WordNet, mainly in order to reduce the workload and cost. This construction method does not quite take into consideration the peculiarities of the given language's lexical semantic systems, inasmuch as the lexical material and the network of relations strongly depend on the solutions specific to English.

The goal of the plWordNet team was to build a wordnet which provides a faithful and comprehensive description of the system of Polish lexical semantics. That is to say, its structure should represent accurately the lexico-semantic relations between lexical meanings in Polish, and be motivated only by observations derived from Polish language data. We were determined to avoid any form of translation from wordnets for other language, and even any kind of structure transfer. That was meant to keep our wordnet's structure free from the idiosyncrasies of the lexical systems of other languages. We also aimed to have a resource with good coverage with respect to lemmas, word senses and instances of lexico-semantic relations, so that the resulting language resource could be a strong basis for practical applications with a high chance of retrieving semantic knowledge. Finally, we assumed that our wordnet should be developed in close correspondence to language data collected from very large corpora, so that it could become a robust, faithful description of Polish usage.

We have been fortunate in the past 10 years to have almost continual funding at a level that allowed us to reach our goals without compromising these fundamental assumptions. It was a rare chance to carry out a long-term plan of building a very large wordnet without worrying too much about cost.[1] The main purpose of this paper is to present plWordNet, to square its final state with the assumptions, and to compare it with several other lexical resources. We will also refer to hundreds of plWordNet's known applications and thus try and show that the effort was worth the price.

2 plWordNet in brief

2.1 The plWordNet model

Wordnets have become standard lexical-semantics resources in NLP, and have found thousands of applications. A wordnet is now considered a basic language resource, expected to be available for any language.

[1] There were three major releases. The development was carried out by researchers (linguists and computational linguists), wordnet editors (supporting linguists) and programmers (developing and maintaining tools to support linguists' work), at the approximate cost of 40 person-years.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2259–2268, Osaka, Japan, December 11-17 2016.

The plWordNet project, arising from a wish to fill a gap in language technology for Polish, and clearly inspired by WordNet, aimed to produce a faithful description of the system of Polish lexical semantics.

It must be noted that several fundamental definitions in the WordNet paradigm, *e.g.,* those of a synset, near synonymy or lexicalised concepts, were not clear enough to be used operationally (Fellbaum, 1998; Vossen, 2002), and to achieve good consistency among wordnet editors – see a longer discussion in (Piasecki et al., 2009). We decided against the transfer method (Vossen, 2002), so as to avoid influencing plWordNet's structure with some properties alien to the Polish lexical system. We also could not adopt the merge method, because no dictionaries or other lexical resources on open licenses were available.[2] We proposed a *corpus-based wordnet development process* instead: a large text corpus is a primary data source, and language tools and systems help wordnet editors explore the corpus.[3]

The corpus has been the main knowledge source for all phases of the development, from the systematic extraction of lemmas for inclusion in plWordNet to the automated acquisition of lexico-semantic relations for presentation to the editors. Dictionaries and encyclopaedias complement language competence of the editors, all of them trained linguists, and in all linguistic matters editors have the last word. Detailed instructions ensure a high degree of consistency of those decisions.

Corpora contain words, with senses discernible by context. Groups of synonyms are not a natural phenomenon in texts. We decided to make the *lexical unit* (LU) the basic building block in plWordNet, rather than the synset as in WordNet (Piasecki et al., 2009). We defined the LU in a rather technical way as a triple: a lemma, its part of speech and its sense indicator. We assumed that one LU belongs to exactly one *synset*. The synset, however, has been defined indirectly – and operationally – as a group of LUs which share lexico-semantic *constitutive relations* and *constitutive features* (Maziarz et al., 2013). Examples of the former are hyponymy, hypernymy, meronymy and holonymy; of the latter, stylistic register, aspect, and semantic classes for adjectives and verbs. With this definition of the synset, a relation between two synsets in plWordNet can be treated as a shorthand for the fact that LUs from the two groups share links by certain relation, *e.g.,* hypernymy.

Each relation has been given a clear definition meant to allow wordnet editors to make consistent decisions. There also are linguistic substitution tests, with slots to be occupied by two LUs possibly in this relation. The tests, which support wordnet editors' decisions very effectively, are automatically filled and presented in a wordnet editing system called WordnetLoom (Piasecki et al., 2013). We adhere intentionally to the *minimal commitment principle*: lexico-semantic relations are grounded in the Polish linguistic tradition and language data in very large corpora; plWordNet's structure is derived from the relations in a way which depends on no particular theory of meaning.

2.2 The content

description layer	instances
lexico-semantic relations	>700K
glosses	>100K
usage examples	83K
links to Wikipedia	55K
sentiment annotation	30K

Table 1: Multilayered semantic description in plWordNet: the statistics.

The relations are the backbone of a wordnet: they jointly describe a word's meaning; definitions and usage example come next. plWordNet has over 40 different relation types (100 when counting subtypes). many of them link LUs from different parts of speech. In addition to relations, plWordNet describes meaning in several ways. Table 1 presents the statistics of these descriptions.

- **Semantic domains** (Princeton WordNet calls them *lexicographer files*) are broad lexical fields of a given LU. They are quite general (*e.g.,* `animals`, `artifacts`, `place`).
- **Stylistic labels** describe the lexical register of a given LU. There are 11 registers in plWordNet: non-standard, obsolete, regional, terminological, argot/slang, literary, official, vulgar, coarse, colloquial, general; words in some registers (*e.g.,* vulgar and coarse) can co-exist in a synset, but normally distinct registers mean distinct synsets. The register thus affects the network of lexical relations.

[2]Unrestricted availability was an essential point for us in view of what we wanted plWordNet's licence to be.

[3]The corpus grew in size from the initial ≈260 million words during the work on plWordNet 1.0, through ≈1.8 billion tokens for plWordNet 2.3, to ≈4.0 billion for plWordNet 3.0.

	synsets	lemmas	LUs	avs
GermaNet	101,371	119,231	131,814	–
PWN	117,659	155,593	206,978	1.74
enWN	125,500	165,712	218,611	1.74
plWN	**197,721**	**179,125**	**260,214**	1.32

Table 2: The count of synsets, lemmas and lexical units (LUs), and average synset size (avs), in PWN 3.1 (PWN), enWordNet 1.0 (enWN), plWordNet 3.0 (plWN) and GermaNet 10.0 (`http://www.sfs.uni-tuebingen.de/GermaNet/`).

- **Glosses** are short definitions, a very important element of plWordNet. They help the user to understand the network, and plWordNet editors to work with high effectiveness.
- **Usage examples** are sentences which illustrate a particular lexical meaning. They are exemplars for sense usage and also real corpus evidence. Usage examples in plWordNet are due to the liguists' intuition, or taken from corpora in the public domain or published on a Creative Commons Licence.
- **Links to *Wikipedia*** are added to those LUs whose meaning is an exact equivalent of a Wikipedia entry.
- **Semantic verb classes**, part of plWordNet's structure, generalise the Vendler classes for typical Polish verb usage. They influence the network's shape, since only verbs of the same class may be linked with hyponymy.[4]
- **Sentiment and emotion annotation** marks word meanings as discussed below.

Sentiment analysis or the construction of a sentiment lexicon, perhaps based on plWordNet, has been a frequently stated intended use of plWordNet once it became publicly available.[5] We met this expectation in a pilot project, in which about 30,000 noun and adjective LUs were annotated with basic emotions (Plutchik, 1980), fundamental human values and sentiment polarity, illustrated by usage examples (Zaśko-Zielińska et al., 2015). LUs rather than synsets were annotated, because LUs from the same synset can differ with respect to sentiment polarity.[6] Annotation covers the sentiment polarity of a sense on a 5-level scale, and basic emotions.) and LUs are the object of linguistic tests or are included in usage examples. The annotation was performed by a group separate from the plWordNet editors, so it also served as a form of verification of the plWordNet content.

The newest release of plWordNet, version 3.0, complements the preceding versions. After version 2.3, the work concentrated on a modified system of relations for adjectives (Maziarz et al., 2012) and on the expansion of the adjective sub-database; the construction of the adverb subnetwork,[7] supported by a semi-automated method based on adjective-adverb derivational relations (Maziarz et al., 2016); and a major increase of the number of lexicalised multi-word expressions (Dziob and Wendelberger, 2016).

3 Comparative analysis

3.1 The lexical net

A wordnet is a lexical net, so it can be evaluated with statistical measures suitable for graphs (Lewis, 2009). We consider graph size, network volume, average graph density, corpus coverage, clustering coefficient, distance measure and connectivity. A wordnet of good quality ought to have a large, dense network, covering contemporary corpora well, and showing traits of "small-worldness".

Network volume and density. Table 2 shows the number of synsets, lemmas and LUs in three manually and independently constructed wordnets: Princeton WordNet, plWordNet and GermaNet, together with enWordNet, our extension of Princeton WordNet. We can say that plWordNet is comparable in size to Princeton WordNet (and the 5% larger enWordNet), and almost twice as large as GermaNet. Table 3 shows that the volumes of the two resources are also comparable. plWordNet has 208K LU relation instances and 324K synset relation instances; the WordNet counts are 91K and 195K, respectively. Taking into account that in WordNet the average synset size is higher than the average synset size in plWordNet (Table 2) one may want to calculate an average relation density per LU. This measure approximates an

[4]For example, *zgubić*$_2$ and *stracić*$_1$ 'to lose' (`HAPPPENINGS`) or *wybudować*$_1$ 'build$_{\texttt{PERF}}$' and *zrobić*$_2$ 'do$_{\texttt{PERF}}$' (`PERFECTIVE ACTIONS`).

[5]An independent attempt has been made (Haniewicz et al., 2013; Haniewicz et al., 2014).

[6]For instance, *pies*$_2$ 'Canis lupus familiaris' is unmarked, while *pies*$_3$ 'cop (policeman)' is negatively marked.

[7]Adverbs are usually neglected in wordnet: there are none in GermaNet, and less than 3% of all lexical units in WordNet are adverbs. Their proper relational description is not easy, as witnessed by WordNet's low synset relation density of 0.03 (Table 1).

WordNet 3.1	verbs		nouns		adverbs		adjectives		all	
	N	ρ	N	ρ	N	ρ	N	ρ	N	ρ
LU relations	24,840	0.99	44,185	0.28	720	0.13	21,636	0.72	91,381	0.42
synset relations	16,827	1.22	145,338	1.62	109	0.03	23,491	1.29	185,765	1.48
all relation types	80,280	3.20	492,457	3.12	1,015	0.18	86,221	2.87	659,973	3.02
plWordNet 3.0	**verbs**		**nouns**		**adverbs**		**adjectives**		**all**	
	N	ρ	N	ρ	N	ρ	N	ρ	N	ρ
LU relations	48,744	1.50	98,376	0.58	12,542	1.14	48,894	1.02	208,556	0.80
synset relations	36,616	1.66	219,266	1.75	19,716	2.18	48,258	1.17	323,856	1.64
all relation types	127,065	3.92	494,893	2.94	43,551	3.94	118,574	2.47	784,083	3.02

Table 3: The volume of the lexical networks and relation density with regard to parts of speech. N is the number of relation instances, ρ is the relation density measured either for LUs, or synsets, or for all relation types.

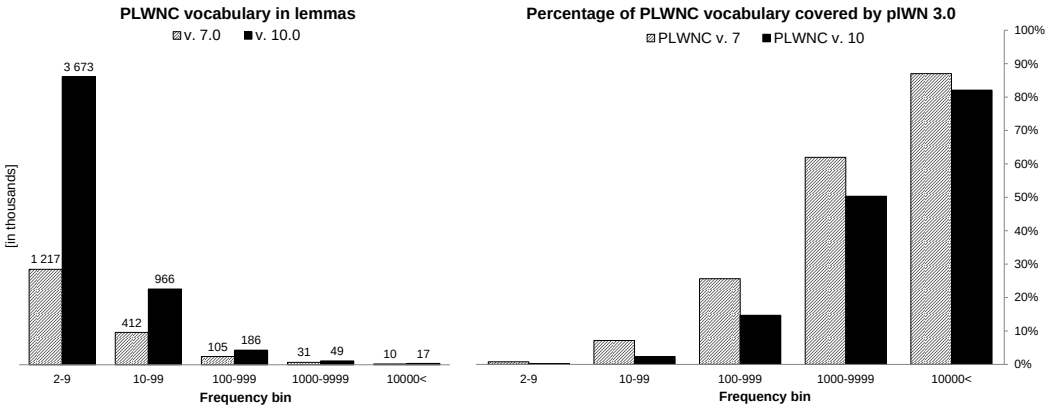


Figure 1: **Left**: The number of lemmas in PLWNC version 3.0 and 10.0 with regard to different freqency bins. The bin "100-999" contains those words that occur in the PLWNC 100 to 999 times. In agreement with Zipf's law, there are far more rare than frequent words in both corpora. **Right**: Coverage of the 7th and 10th version of PLWNC by plWordNet 3.0.

amount of information falling to a single LU, which in fact is very similar for both wordnets, 660K for WordNet and 785K for plWordNet, see row "all relation types" in the table.[8]

Corpus coverage. Figure 1, right, shows how well plWordNet 3.0's vocabulary covers PLWNC. plWordNet was developed on three corpora, the ICS PAS corpus (Przepiórkowski, 2004) (plWordNet 1.0, 250M tokens), plWordNet Corpus 7.0 (plWordNet 2.0 and 3.0, 1.8G tokens) and plWordNet Corpus 10.0 (plWordNet 3.0, 4.2G tokens). Note that the coverage of PLWNC 10.0 is lower than that of version 7.0. The chart also proves that plWordNet creators favoured more frequent lemmas over less frequent. Figure 2, left, presents the coverage of three versions of plWordNet (1.0, 2.0 and 3.0). The consecutive versions of plWordNet housed more and more low-frequenct lemmas. Now, words with frequencies lower than $f = 10$ account for merely 10% of plWordNet 3.0 (Figure 2, right).

Small world. Similarly to Princeton WordNet, plWordNet shows a small-world behaviour: short average path length and high clustering coefficient (Sigman and Cecchi, 2001).[9] In Figure 3 we plot the statistics for three versions of plWordNet (1.0, 2.0, 3.0), Princeton WordNet and a conglomerate, an effect of mapping from plWordNet 3.0 to WordNet 3.1 (WN-plWN3). For a classical random graph of plWordNet's size, a global clustering coefficient is close to $\frac{\langle k \rangle}{N} = 2.5 \times 10^{-5}$, where $\langle k \rangle$ is an average number of neighbours of a vertex (see ρ values in Table 3, we put here $\langle k \rangle = 3$), and N is the number of graph vertices (in this case synsets, see Table 2). The average path length for the random graph is very similar to the obtained values (see see Figure 3): $\frac{ln(N)}{ln(\langle k \rangle)} \approx 11$ (Omidi and Masoudi-Nejad, 2009).

For sure, plWordNet is denser now in terms of the clustering coefficient and the shorter path lengths than in the past (it is indeed a smaller world now). As compared to WordNet, plWordNet versions 2.0 and 3.0 have shorter average path length and higher clustering coefficient.

[8]This approximation was calculated thus: we choose synset relations within the same POS and multiply the number of relation instances by a square of the average synset size for a particular POS (synset relations are shorthand for relations between LUs from two synsets). If a synset relation holds between different POSs, we multiply the number of synset relations by the average synset sizes of the two distinct POSs.

[9]We calculate the classic *global clustering coefficient* (Opsahl, 2013).

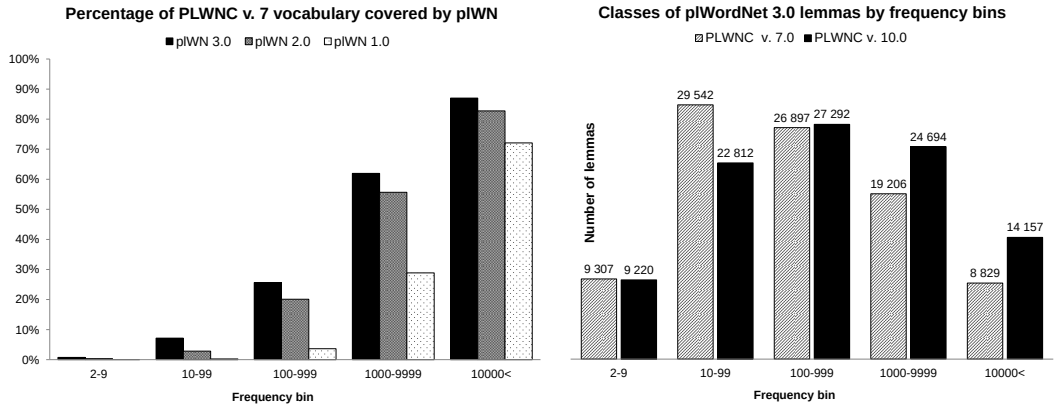


Figure 2: **Left**: Coverage of the 7th version of plWordNet Corpus (PLWNC) by three different stages of plWordNet development – versions 1.0 (from 2009), 2.0 (2013) and 3.0 (2016) – with regard to frequency bins. The bin "100-999" contains words which occur in the PLWNC 100-999 times. Percentages show how many lemmas in each corpus bin are found in plWordNet (version 1st, 2nd or 3rd). **Right**: The cardinality of frequency bins in plWordNet 3.0. Frequencies were calculated in two versions of plWordNet Corpus (7.0 i 10.0).

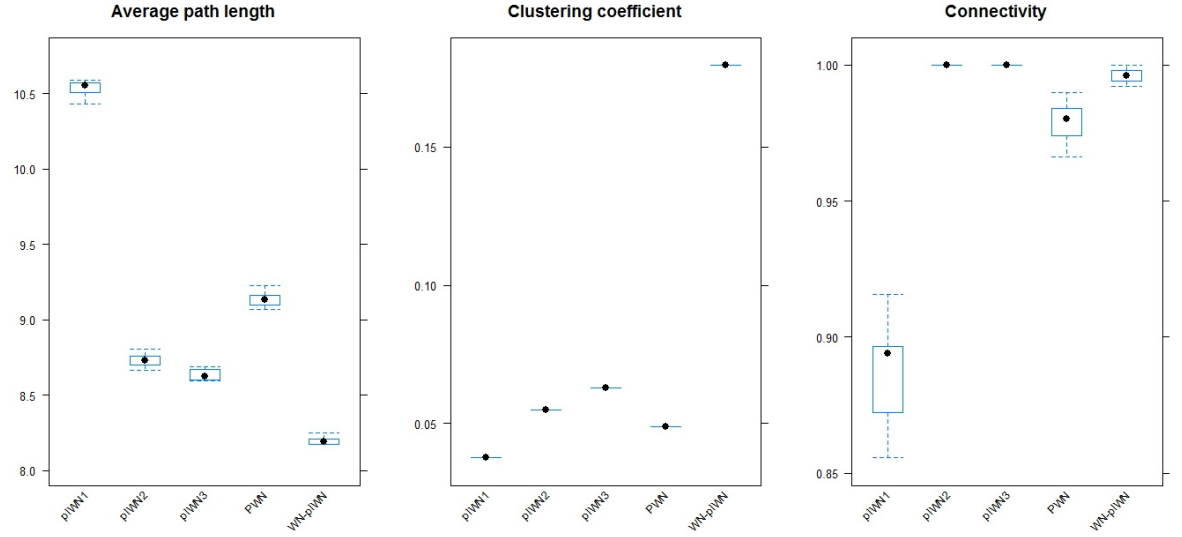


Figure 3: Average path length, clustering coefficient and connectivity in different lexical networks. plWN1, plWN2, plWN3: = plWordNet 1.0, 2.0, 3.0; PWN: WordNet 3.1, WN-plWN: mapping between plWordNet3.0 and WordNet 3.1. Clustering coefficients were calculated for the whole graphs. Average path lengths were obtained by randomly picking a pair of 2×500 synsets (without replacement) and seeking a way through the graph between the pairs; if a way could be found, the shortest path was chosen, and then the set of resulting calculations was averaged. The procedure was repeated 10 times for each graph. The connectivity was calculated simultaneously: it is a ratio of felicitously found paths.

Connectivity measures how often a path can be established between two synsets randomly chosen in a graph. For all wordnet versions, the statistic is high (>85%) or very high (>95%), with plWordNet 1.0 last in ranking and two other versions of plWordNet with the two highest ranks.

The mapping results, described in the next section, were very surprising. The merged networks of Polish and English lexical units gave impressive values of clustering coefficient (3 times larger than for plWordNet 3.0) and shortest path lengths. The conglomerate has small-world behaviour more than its separate parts. It seems that linking independently built resources creates a new quality.

3.2 Comparison by mapping

As noted, plWordNet has been developed independently from WordNet, without any transfer of structures between the two resources, thus avoiding any bias towards WordNet. Even so, the alignment of plWordNet and WordNet was needed for a variety of (bilingual and multilingual) applications and research tasks. We have designed a strategy of mapping plWordNet to WordNet (Rudnicka et al., 2012). The key element

I-relation	Noun	Adjective	Adverb	Total
I-Synonymy	36,367	4,077	448	40,892
I-Hyponymy	74,394	29,216	781	104,391
I-Hypernymy	4,121	167	51	4,339
I-Meronymy	6,982	-	-	6,982
I-Holonymy	3,471	-	-	3,471
I-Partial synonymy	4,339	1,544	4	5,887
I-Inter-register synonymy	1,672	54	22	1,748
I-Cross-categorial synonymy	-	19,286	-	19,286
Total	131,346	54,344	1,306	186,996

Table 4: Interlingual relation counts

of the strategy was a comparison of the two relation structures in order to find the corresponding nodes of synset graph structures and link them via one of eight interlingual relations (hierarchically ordered by varying strength and specificity). The mapping was done manually, in the WordNetLoom editor (Piasecki et al., 2013), bottom-up (leaves first), from plWordNet to WordNet. As a result, almost all plWordNet noun synsets are mapped in version 3.0, about $\frac{3}{4}$ of adjective synsets and about $\frac{1}{4}$ of adverb synsets.

The linguists' work was supported by an automatic prompt system which suggested interlingual links using a rule-based part-of-speech-sensitive algorithm, and a cascade dictionary (Kędzia et al., 2013; Rudnicka et al., 2015a). The final decisions, however, were made by linguists and the cost of the mapping process was comparable to that of editing plWordNet. That has turned out to be money well spent, for two reasons. The two interlinked, independently created wordnets provide a remarkable opportunity to run a comparative analysis; and the mapping process required a careful analysis of plWordNet's structure, so it was a kind of evaluation procedure.

Indeed, the mapping process enabled a comparative analysis and an evaluation of the lexical coverage and the construction methods of the two wordnets. The linguists discovered many gaps in the lexical coverage between plWordNet and Princeton WordNet, as well as numerous differences in the number, type and structure of synset and LU relations – all due to the different construction methods (Rudnicka et al., 2015b). These facts account for the final mapping results, with interlingual hyponymy counts doubling interlingual synonymy counts. This is illustrated in Table 4.

The results are striking. Interlingual synonymy was most highly favoured by the mapping procedure, yet its counts are much lower than those of interlingual hyponymy across all mapped categories. This is caused by the strict restrictions on the application of I-Synonymy. It could only be assigned given strong correspondence of the meanings and relation structures between plWordNet and WordNet synsets. Superficially, noun synset relation structures seem largely to correspond, with hyponymy forming the backbone of a relation network. However, on a closer look, various contrasts come to the fore.

First, plWordNet and WordNet differ in synset granularity, which affects relation structures. In general, plWordNet synsets are smaller and tend to include fewer lexical units than WordNet synsets. In plWordNet there are always distinct synsets for feminine, masculine and neuter forms, singular and plural, mass and count, diminutive, augmentative and stylistically marked forms. While mapping, we found many instances of mixed WordNet synsets grouping together marked and unmarked forms of such pairs. Moreover, the concept of hyponymy in plWordNet and in WordNet is different. plWordNet always understands hyponymy narrowly, as "and hyponymy": the hyponyms have to have all properties of their hypernym(s). That leads to many cases of multiple hyponymy, but it is always of the "and" type. WordNet also allows a more relaxed "or hyponymy", which lets hyponyms have *some* properties of their hypernym(s). We have also found places (both in plWordNet and in WordNet) where the same conceptual dependency was encoded variously by meronymy or by hyponymy.

Adjective and adverb relation structures diverge even more between plWordNet and WordNet than noun relations structures (Rudnicka et al., 2015a). In plWordNet, the adjective synset relation structure is a vertical, hyponymy-based network, partly similar to that for nouns. WordNet employs a completely different, horizontal dumbbell model, based on a rather vague "Similar to" relation. That has made designing an adjective mapping procedure a real challenge. We had to take into account the lexical unit relation network which displays more similarity to establish interlingual correspondence links between plWordNet and WordNet synsets. Since adverbs have been systematically derived from adjectives, we have also capitalised on the results of adjective mapping in designing the mapping procedure for adverbs. The relevant interlingual adjective relation links were copied to adverbs and presented in the form of automatic prompts to linguists. They verified them and introduced manual interlingual adverb links. That process

	plWordNet	WordNet
Nouns	2,733	43,575
Verbs	22,029	13,789
Adjectives	8,188	11,298
Adverbs	7,529	2,704
Total	40,479	71,366

Table 5: The number of synset not mapped yet in plWordNet 3.0 and Princeton WordNet 3.1.

also allowed for critical evaluation (sometimes followed by correction) of interlingual adjective links.

Having finished their work on mapping synsets from selected wordnet graphs (usually domain-restricted), bilingual linguists reported potential errors in plWordNet to the team responsible for the Polish side, who analysed and, if needed, corrected them. Despite meticulous quality control, it is inevitable that isolated errors – typos, flawed links, synsets too general or too specific – persist in plWordNet 3.0. Such errors will be rooted out when a reporting system for users has been implemented.

The mapping went in the usual "national wordnet to WordNet" direction. We were well-aware of substantial lexical, grammatical and cultural differences between English and Polish as well as different development processes of the two wordnets. Even so, we did not expect differences in the mapping coverage between the wordnets as large as those illustrated in Table 5.

The reasons for the discrepancies in the mapping coverage of nouns and adjectives have been already discussed. The mapping of adverbs has only started, while verbs have not been mapped yet.

In short, the results of mapping have shown large differences between plWordNet and WordNet in lexical content, coverage and relation structure. Differences in lexical content are due to lexico-grammatical differences between English and Polish and the existence of many lexical and cultural gaps between the two languages. Differences in lexical coverage are due to different construction methods of the two wordnets: merge method for WordNet and corpus-based method for plWordNet, as well as in the time span of their construction: mid 1990-ties to 2006 for WordNet 3.0 and 2005-2016 for plWordNet 3.0.

The differences in relation structure are due to different theoretical solutions assumed in the construction of two wordnets: lower vs higher synset granularity, "and" vs "or" hyponymy, and the use of hyponymy and meronymy to code the same conceptual distinctions. The effects of those differences are the prevalence of I-hyponymy over I-synonymy and the large part of WordNet not mapped yet, due to one-directional, plWordNet to WordNet mapping direction.

An I-hyponymy-based bilingual resource is clearly less valuable than one based on I-synonymy (due to the lower specificity of links). So, we have sought remedies. One idea was to exploit the existing I-hyponymy links to extend WordNet's coverage. The result was the construction of enWordNet 1.0, an extended version of WordNet. The lemmas of plWordNet leaf synsets linked by I-hyponymy to WordNet synsets were automatically translated by a large cascade dictionary. The obtained list of translations was then filtered by WordNet lemmas. Next, the results of this filtering were divided into lemmas for which the cascade dictionary found: (1) equivalents whose lemmas were not present in WordNet; (2) no equivalents; (3) equivalents whose lemmas were already present in WordNet.

Linguists started with the first group, carefully verifying the suggestions with corpora and all available resources; then they moved to the second group, trying to find equivalents on their own (in all available resources); lastly, they investigated the third group, verifying the existing mapping relations. Moreover, whenever linguists started work with a particular WordNet "nest", they were encouraged to look for its possible extensions on their own (not limiting themselves to cascade dictionary suggestions). The effect of that work is a substantially enlarged version of WordNet, with lexical material – some 10,000 lemmas – added in many places where a link from the Polish side would have been inaccurate. The result, enWordNet 1.0,[10] is also part of this release, which ought to encourage comparative studies and cross-lingual research.

4 Applications of plWordNet

Language resources are developed for applications: the higher the uptake, the better the perceived quality. plWordNet is a pivotal element of a system of language and knowledge resources; plWordNet's wide coverage helps a lot. The system has several layers, with plWordNet in the middle:

- top- and medium-level ontology SUMO with plWordNet semi-automatically mapped onto it (Kędzia and Piasecki, 2014),

[10]The symbol WordNet® is a registered trademark. We cannot use it.

- NELexion2, a very large lexicon of Polish Proper Names (PNs), ≈1.5 million, manually linked at the level of fine-grained semantic PN classes (Marcińczuk, 2016),
- a lexicon of ≈60,000 multiword expressions with syntactic structures described, linked to plWordNet's LUs by lemmas (Maziarz et al., 2015; Dziob et al., 2016),
- a syntactic-semantic lexicon of Polish valency frames (≈15,000 lemmas described) linked to plWordNet at the LU level and semantic restrictions of frame arguments (Kotsyba, 2014; Hajnicz, 2014).

The system is a very large network, linking knowledge elements to lexical meaning and descriptions of local syntactic-semantic structures. Given the mapping to WordNet, the system can be an anchor to a global Linked Data network,[11] a powerful cloud of heterogenous data webs. Manually crafted lexical-semantic resources could serve as a skeleton for the cloud, notably with plWordNet's comprehensive coverage. Lexical item descriptions therein would be the means of anchoring webs to text clouds.

plWordNet has become an important reference for research on the development of wordnets; (Fišer and Sagot, 2015) is the latest of numerous citations.

plWordNet's open license enables frequent use as a monolingual and bilingual dictionary: Web-based (`http://plwordnet.pwr.edu.pl`) via an Android application, and via WordnetLoom (Piasecki et al., 2013) (`http://ws.clarin-pl.eu/public/WordnetLoom-Viewer.zip`) a wordnet editor which offers advanced visual, graph-based browsing. plWordNet has also been included in a very large and popular Polish multilingual dictionary Lingo (`http://ling.pl`). Access to plWordNet as a dictionary amounts to tens of thousand of visits a month.

In addition to monolingual resources, plWordNet is part of multilingual resources, *e.g.,* WordTies (Pedersen et al., 2012), Open Multilingual WordNet (Bond and Foster, 2013) and multimodal resources, *e.g.,* the classification of gestures based on the verb categorisation in plWordNet (Lis and Navarretta, 2014). plWordNet was referred to in the resource for textual entailment (Przepiórkowski, 2015) and utilised for ontology mapping and linking ontology to lexicon (Jastrząb et al., 2016).

Assorted applications of plWordNet include language correction, relation extraction (Mykowiecka and Marciniak, 2014), text indexing (Kaleta, 2014), Text Mining (Maciołek and Dobrowolski, 2013), text classification (Wróbel et al., 2016; Mirończuk and Protasiewicz, 2016), Open Domain Question Answering (Przybyła, 2013), and use as a quasi-ontology in document structure recognition (Kamola et al., 2015).

Registered users of plWordNet declare its applications. Here is a selection of such declaration: education (at different levels) including Polish language teaching, building dictionaries, extraction of synonyms and semantically related words, detection of loanwords, cross-linguistic study on phonesthemes, classification of metaphorical expressions, corpus studies, grammar development, comparative and contrastive studies, language recognition, parsing disambiguation, semantic analysis of text, document similarity measures, semantic indexing of documents, semantic information retrieval, recommendation systems, construction of chatbots and dialogue systems, plagiarism detection, translation evaluation, data visualisation, research on complex networks and ontologies. An exceptional case is the practical use of plWordNet during the medical treatment of aphasia.

5 Always more to do

The release of plWordNet 3.0 is a caesura, but language resources never really reach a stable state. The wordnet is an NLP-friendly description of the Polish lexical system on a scale unheard of even in previously published large unilingual dictionaries.

And yet, each element of the system could stand improvement. For example, while many derivational relations (typical of strongly inflected languages such as Polish) have been introduced, there remains a motherlode of relations signalled by verbal prefixes, a highly productive operation similar to what phrasal verbs contribute to English. Relation density in plWordNet is quite satisfactory, but there can be semi-automatic methods of improving it further. Stylistic registers as a constitutive feature can lead the the natural introduction of sub-databases of specialised vocabulary for a variety of domains, interlinked across registers. Multi-word expressions and proper names need more work. Emotion annotations have to be extended onto the whole network.

Last but not least, user feedback in matters small (typos, omissions) and large (new functionalities, support for new kinds of applications) ought to be implemented.

Acknowledgment: work financed as part of the investment in the CLARIN-PL research infrastructure funded by the Polish Ministry of Science and Higher Education.

[11] `http://linkeddata.org/`

References

Francis Bond and Ryan Foster. 2013. Linking and Extending an Open Multilingual Wordnet. In *Proc. 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1352–1362, Sofia, Bulgaria.

Agnieszka Dziob and Michał Wendelberger. 2016. Extraction and description of multi-word lexical units in plWordNet 3.0. In *Proc. 8th Int. Global Wordnet Conference.*

Agnieszka Dziob, Michał Kaliński, Marek Maziarz, Maciej Piasecki, Adam Radziszewski, Stan Szpakowicz, and Michał Wendelberger. 2016. MWELexicon. Language resource published in CLARIN-PL repository, April.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database.* MIT Press.

Darja Fišer and Benoît Sagot. 2015. Constructing a poor man's wordnet in a resource-rich world. *Language Resources and Evaluation*, 49(3):601–635.

Elżbieta Hajnicz. 2014. Lexico-Semantic Annotation of Składnica treebank by means of plwn lexical units. In *Proc. Seventh Global Wordnet Conference*, pages 23–31, Tartu, Estonia.

Konstanty Haniewicz, Wojciech Rutkowski, Magdalena Adamczyk, and Monika Kaczmarek. 2013. Towards the Lexicon-Based Sentiment Analysis of Polish Texts: Polarity Lexicon. In *Computational Collective Intelligence. Technologies and Applications: 5th International Conference, ICCCI 2013, Craiova, Romania*, pages 286–295. Springer.

Konstanty Haniewicz, Monika Kaczmarek, Magdalena Adamczyk, and Wojciech Rutkowski. 2014. Polarity Lexicon for the Polish Language: Design and Extension with Random Walk Algorithm. In *Advances in Systems Science: Proc. International Conference on Systems Science 2013 (ICSS 2013)*, pages 173–182. Springer.

Tomasz Jastrząb, Grzegorz Kwiatkowski, and Paweł Sadowski. 2016. Mapping of Selected Synsets to Semantic Features. In *Beyond Databases, Architectures and Structures. Advanced Technologies for Data Mining and Knowledge Discovery: 12th International Conference, BDAS 2016, Ustroń, Poland*, pages 357–367. Springer.

Zbigniew Kaleta. 2014. Semantic text indexing. *Computer Science*, Vol. 15 (1):19–34.

Grzegorz Kamola, Michal Spytkowski, Mariusz Paradowski, and Urszula Markowska-Kaczmar. 2015. Image-based logical document structure recognition. *Pattern Analysis and Applications*, 18(3):651–665.

Paweł Kędzia, Maciej Piasecki, Ewa Rudnicka, and Konrad Przybycień. 2013. Automatic Prompt System in the Process of Mapping plWordNet on Princeton WordNet. *Cognitive Studies.* to appear.

Natalia Kotsyba. 2014. Using Polish Wordnet for Predicting Semantic Roles for the Valency Dictionary of Polish Verbs. In *Advances in Natural Language Processing: 9th International Conference on NLP, PolTAL 2014, Warsaw, Poland*, pages 202–207. Springer.

Pawel Kędzia and Maciej Piasecki. 2014. Ruled-based, Interlingual Motivated Mapping of plWordNet onto SUMO ontology. In *Proc. Ninth International Conference on Language Resources and Evaluation (LREC-2014), Reykjavik, Iceland*, pages 4351–4358.

Ted G. Lewis. 2009. *Network Science: Theory and Applications.* Wiley.

Magdalena Lis and Costanza Navarretta. 2014. Classifying the Form of Iconic Hand Gestures from the Linguistic Categorization of Co-occurring Verbs. In *Proc. 1st European Symposium on Multimodal Communication University of Malta; Valletta; October 17-18; 2013*, volume 101 of *Linköping Electronic Conference Proceedings*, pages 41–50. Linköping University Electronic Press.

Przemysław Maciołek and Grzegorz Dobrowolski. 2013. Cluo: web-scale text mining system for open source intelligence purposes. *Computer Science*, Vol. 14 (1)(1):45–62.

Michał Marcińczuk. 2016. NELexicon2. Language resource – lexicon of Polish Proper Names – published in CLARIN-PL repository, April.

Marek Maziarz, Stanisław Szpakowicz, and Maciej Piasecki. 2012. Semantic Relations among Adjectives in Polish WordNet 2.0: A New Relation Set, Discussion and Evaluation. *Cognitive Studies*, 12:149–179.

Marek Maziarz, Maciej Piasecki, and Stanisław Szpakowicz. 2013. The chicken-and-egg problem in wordnet design: synonymy, synsets and constitutive relations. *Language Resources and Evaluation*, 47(3):769–796. http://link.springer.com/article/10.1007/s10579-012-9209-9.

Marek Maziarz, Stan Szpakowicz, and Maciej Piasecki. 2015. A Procedural Definition of Multi-word Lexical Units. In *Proc. RANLP'2015*, pages 427–435, Hissar, Bulgaria.

Marek Maziarz, Stan Szpakowicz, and Michał Kaliński. 2016. Adverbs in plWordNet: Theory and Implementation . In *Proc. of GWC 2016*, pages 209–217.

Marcin Mirończuk and Jarosław Protasiewicz. 2016. A Diversified Classification Committee for Recognition of Innovative Internet Domains. In *Beyond Databases, Architectures and Structures. Advanced Technologies for Data Mining and Knowledge Discovery: 12th International Conference, BDAS 2016, Ustroń, Poland*, pages 368–383. Springer.

Agnieszka Mykowiecka and Małgorzata Marciniak. 2014. Attribute Value Acquisition through Clustering of Adjectives. In *Advances in Natural Language Processing: 9th International Conference on NLP, PolTAL 2014, Warsaw, Poland*, pages 92–104, Cham. Springer.

Saeed Omidi and Ali Masoudi-Nejad, 2009. *Computational Social Network Analysis: Trends, Tools and Research Advances*, chapter Network Evolution: Theory and Mechanisms, pages 191–224. Springer Science & Business Media.

Tore Opsahl. 2013. global clustering coefficient. *Social Networks*, 35(2).

Bolette Sandford Pedersen, Lars Borin, Markus Forsberg, Krister Lindén, Heili Orav, and Eirikur Rögnvalssson. 2012. Linking and Validating Nordic and Baltic Wordnets- A Multilingual Action in META-NORD. In *Proc. 6th International Global Wordnet Conference*.

Maciej Piasecki, Stanisław Szpakowicz, and Bartosz Broda. 2009. *A Wordnet from the Ground Up*. Wrocław University of Technology Press. http://www.eecs.uottawa.ca/~szpak/pub/A_Wordnet_from_the_Ground_Up.zip.

Maciej Piasecki, Michał Marcińczuk, Radosław Ramocki, and Marek Maziarz. 2013. WordNetLoom: a WordNet development system integrating form-based and graph-based perspectives. *International Journal of Data Mining, Modelling and Management*, 5(3):210–232.

Robert Plutchik. 1980. *EMOTION: A Psychoevolutionary Synthesis*. Harper & Row.

Adam Przepiórkowski. 2015. Towards a Linguistically-Oriented Textual Entailment Test-Suite for Polish Based on the Semantic Syntax Approach. *Cognitive Studies / Études Cognitive*, 15:177–191.

Adam Przepiórkowski. 2004. *The IPI PAN Corpus, Preliminary Version*. Institute of Computer Science PAS.

Piotr Przybyła. 2013. Question Classification for Polish Question Answering. In *Proc. of IIS 2013*, pages 50–56, Warsaw, Poland.

Ewa Rudnicka, Marek Maziarz, Maciej Piasecki, and Stan Szpakowicz. 2012. A Strategy of Mapping Polish WordNet onto Princeton WordNet. In *Proc. COLING 2012, posters*, pages 1039–1048.

Ewa Rudnicka, Wojciech Witkowski, and Michał Kaliński. 2015a. A Semi-automatic Adjective Mapping Between plWordNet and Princeton WordNet. In *Text, Speech, and Dialogue*, pages 360–368. Springer.

Ewa Rudnicka, Wojciech Witkowski, and Michał Kaliński. 2015b. Towards the Methodology for Extending Princeton WordNet. *Cognitive Studies*, 15(15):335–351.

Mariano Sigman and Guillermo A. Cecchi. 2001. Global organization of the Wordnet lexicon. In *Proc. National Academy of Sciences of the United States of America*, volume 99.

Piek Vossen. 2002. EuroWordNet. Technical report, Univ. of Amsterdam.

Krzysztof Wróbel, Maciej Wielgosz, Aleksander Smywiński-Pohl, and Marcin Pietron. 2016. Comparison of SVM and Ontology-Based Text Classification Methods. In *Proc. of ICAISC 2016*, pages 667–680, Zakopane, Poland. Springer.

Monika Zaśko-Zielińska, Maciej Piasecki, and Stan Szpakowicz. 2015. A Large Wordnet-based Sentiment Lexicon for Polish. In *Proc. RANLP 2015*, pages 721–730.

Time-Independent and Language-Independent Extraction of Multiword Expressions From Twitter

Nikhil Londhe
SUNY Buffalo
nikhillo@buffalo.edu

Rohini K Srihari
SUNY Buffalo
rohini@buffalo.edu

Vishrawas Gopalakrishnan
SUNY Buffalo
vishrawa@buffalo.edu

Abstract

Multiword Expressions (MWEs) are crucial lexico-semantic units in any language. However, most work on MWEs has been focused on standard *monolingual* corpora. In this work, we examine MWE usage on Twitter - an inherently multilingual medium with an extremely short average text length that is often replete with grammatical errors. In this work we present a new graph based, language agnostic method for automatically extracting MWEs from tweets. We show how our method outperforms standard Association Measures. We also present a novel unsupervised evaluation technique to ascertain the accuracy of MWE extraction.

1 Introduction

Apart from being just a social media platform, Twitter has emerged as an authoritative source of breaking news and subsequent discussions (Kwak et al., 2010; Hu et al., 2012). Most "global" news stories, from terrorist attacks, political news, sports events to celebrity updates, not only *trend* on Twitter within minutes of the actual event but often in multiple languages. One challenge thus, in understanding the full story is being able to process all languages involved. One way to do this could be by partitioning data into the constituent languages (Bergsma et al., 2012) as there exist several sophisticated tools for Twitter (Pak and Paroubek, 2010; Ritter et al., 2012; Owoputi et al., 2013; Kong et al., 2014) designed specifically for various languages (Avontuur et al., 2012; Abdul-Mageed et al., 2012; Rehbein, 2013). However, such an approach might not be able to process all languages. Further, it faces an added disadvantage of ignoring valuable semantic, temporal and cross-lingual relationships between the tweets. In fact these relationships could instead be utilized to not only better understand the underlying story but also generate resources for resource poor languages in question.

Thus, as a cursory step in understanding such hashtags, our work focuses on extracting multiword expressions (MWEs) from Twitter data streams. MWEs are great starting points from two perspectives: (a) they are statistically "idiosyncratic" (Sag et al., 2002) and thus, require no prior knowledge of the text or the corresponding language for extraction and (b) form a considerable portion of the vocabulary for a given language (Fellbaum, 1998). Furthermore, their importance for a variety of NLP tasks like POS tagging (Shigeto et al., 2013), deep parsing (Nivre and Nilsson, 2004), sentiment analysis (Moreno-Ortiz et al., 2013), translation (Ren et al., 2009; Carpuat and Diab, 2010) etc: cannot be overstated. Also, as we explore in Section 4, MWE usage on Twitter shows some unique characteristics stemming from the nature of the medium like acronym usage, temporal sensitivity, etc. and thus, motivating a stronger need to develop MWE extraction techniques specific to such data streams.

However, most work (Van de Cruys and Moirón, 2007; Ramisch et al., 2010; Sinha, 2011) on automatic MWE extraction has either relied on (a) the knowledge of POS patterns that constitute MWEs and the availability of POS annotated corpora, or (b) enumeration of all possible n-grams and ranking them using Association Measures (AMs) (Pedersen et al., 2011). A third branch of work also exists that instead uses parallel corpora (Da Silva et al., 1999) and exploits distributional dissimilarity between words

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2269–2278, Osaka, Japan, December 11-17 2016.

S. no	Tag type	POS tags	Examples
1	ADJP	JJ JJ	*petits blancs*, ginger redhead
2		NN JJ	day gay, reunion special
5	NP	JJ NN	*delicioso cctel*, *sozialen netzwerken*
6		DT NN	*la eurocopa*
7		NN NN	clapback season, skai jackson, **fra rou**, *asie pacifique*
8		NN RB	*la arranca*
9	ADVP	PP RB	*mal den*
10	VP	NN VB	*je suis*, *kuch lana*, *nahi aayenge*
11		RB VB	verbally attacked, *heit aber*
12		VB JJ	breaking federal
13		VB NN	cry blood, banish demons, minimize disruption, *evitar el*
14		VB RB	starts tonight, acted honorably
15		VB VB	gotta catch, **lets rt**

Table 1: Examples of extracted MWEs and their syntactic classification

to extract phrases. However, we do not consider this approach further given the target domain and only mention it here for completeness.

However, as outlined above, the very nature of our problem invalidates the first line of approach. It is impractical to build corresponding systems (namely POS taggers, POS patterns and candidate extraction) for every applicable language. As far as the second approach is concerned, it is usually effective over time invariant datasets where one time enumeration of all n-grams would suffice. However, our setting would require frequent regeneration of N-grams as the corpus increases over time. Hence, we would like to find methods that do not require enumerating all N-grams and can yet find statistically significant phrases. An added challenge, as we discuss in Section 2, when working with multilingual data is that of evaluation. Thus, we must also find ways to evaluate the extracted MWEs that involves minimal manual intervention.

Thus, the primary objectives of this work can be enumerated as:

- Propose a new graph based method for MWE extraction that can circumvent the challenges of Twitter language usage, temporal nature of Hashtags and possible enumeration of all N-grams.
- Propose an automatic evaluation technique for the extracted MWEs
- Additionally, analyze the variance in extracted MWEs across different variables

The rest of the paper is organized as follows. Starting with Section 2, we first discuss the problem setting in a little more detail and then present our method in Section 3. We show why a word graph based method can overcome the enumerated problems - multilingualism, lack of grammar and relatively free word ordering to name a few. Then in Section 4, we describe our novel evaluation technique and also compare the performance of our method against different AMs. Finally, we conclude by discussing the scope of future work and conclusions from our results in Section 5.

2 Related Work

In this section, we consider the problem of extracting MWEs from a text corpus and evaluating the accuracy and nature of the extracted MWEs. As discussed in Section 1, nuanced extraction techniques rely on POS annotated corpora at the very least. Firstly, the lack of POS taggers for all applicable languages would reduce the size of the workable dataset. For example, some resource poor languages like Malay, Indonesian etc: have very little work in the said regard (Adriani and Van Rijsbergen, 2000; Rais et al., 2011). Secondly, as shown by (Derczynski et al., 2013), POS taggers trained on longer documents perform poorly on tweets. Further the extraction patterns vary widely (Kunchukuttan and Damani, 2008; Green et al., 2011; Tsvetkov and Wintner, 2014) based on the underlying language and thus, making it computationally intractable. Finally, as shown by (Solorio et al., 2014), it is much harder to detect the individual languages within code switched short text documents. Further, there is even lack of availability of standard annotated corpora beyond a handful of languages (Solorio and Liu, 2008; Vyas et al., 2014) for code switched text and thus, almost little to no research even exists in extracting MWEs from such text. Thus, at the very least we need to look at techniques that do not rely on POS tags.

As far as candidate evaluation is concerned, most techniques discussed thus far focus on evaluating the *efficacy* of the *POS extraction patterns*. Hence, a common technique (Pearce, 2002; Ramisch et al., 2012) involves measuring recall against standard corpora. Note that such methods assume that an exhaustive language specific list of MWEs is available. However, since our task is primarily concerned with MWE "discovery", such standard lexicons may not be used. A common alternative involves manual evaluation. However, our initial efforts at manual evaluation proved to be tedious primarily due to unfamiliarity with some of the languages. This prompted us to develop an automatic evaluation technique that we present in Section 4.2 that uses the Twitter Search API.

However, this raises a related yet contrary question on MWE classification. For the extracted MWEs to be useful for downstream processing, some nomenclature must be developed. Some of the earliest work in MWE extraction and classification was done by Sag et al. (2002). They initially introduced a structural classification for MWEs that relies on the differences in compositionality and fixedness between the different MWEs. Later work by Schneider et al. (2014) on MWE usage in social media uses two classification schemes. One, that deals with compositionality and classifies MWEs as either *strong* or *weak* based on their opaqueness and a second, detailed syntactic classification that relies on POS tags. In a multilingual scenario however, it is much easier to determine POS tags for a foreign phrase than to judge the compositionality or opaqueness of the MWE itself. Thus, in continuation with the list provided by Schneider et al. (2014) that deals specifically with social media, we adopted an abridged version[1] as depicted in Table 1. The table lists the tag type, the POS tags used and some extracted examples. Note that this scheme is used only for the purpose of classification and not utilized for MWE evaluation. For languages other than English, we determine membership by examining the translation of the given foreign language phrase. We largely use this nomenclature for analysis as presented in Section 4.

Having thus presented an overview of related work, we now turn our attention to our main algorithm.

3 System description & algorithms

3.1 Constructing Word Graphs

Thus, so far we have established that the nature and size of tweets are an hindrance for the standard tokenization process. However, using word graphs would circumvent both problems. On one hand, they would allow us to capture co-occurrence and statistical information within the graph structure but at the same time allow relaxed word ordering. Thus, given a set of tweets for a hashtag, which we will refer to as a *dataset*, we could construct a single graph $G = (V, E)$ from all tweets as follows. The vertices V represent the set of all unique tokens that occur within the dataset and two vertices share an edge if they co-occur within a tweet. The edge weight is set to the co-occurrence probability of the participating vertices and each vertex is annotated with the occurrence probability of the underlying token. The token set is obtained by simple whitespace tokenization followed by lowercasing and removing all mentions, URLs, emojis/emoticons and # prefixes.

For such a graph, we further contend that the tokens represented by a pair of vertices constitute a MWE if (a) the said tokens frequently co-occur but (b) rarely occur with other tokens. This could be ascertained by using the edge weights and examining the vertex neighborhoods of the said vertices. To that end, we looked at similar problems in other domains and found the method as presented by Londhe et al. (2014) for *Product title matching* to be promising. The authors essentially demonstrate how word graphs for product titles can be utilized to detect equivalences using a community detection algorithm viz. CDAM (Community Detection for Approximate Matching). We thus implemented equivalent algorithms, collectively called GRePE (**G**raph **Re**duction for **P**hrase **E**xtraction) in our problem setting which we now present.

3.2 Extracting MWE candidates

A block diagram of our system components is shown in Figure 1. Overall, the two main system components are the Indexer and the Graph Reducer. The *Indexer* ingests a given dataset to convert it into

[1] we only use bigrams and treat proper nouns as any other nouns

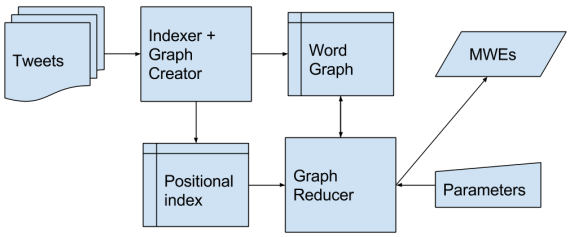


Figure 1: System block diagram

Algorithm 1 CROSS-VERTEX ENRICHMENT

Input: Pair of vertices V_1, V_2 and enrichment threshold η
Output: Enriched neighborhoods $\mathrm{N}^e(V_1), \mathrm{N}^e(V_2)$
for Vertex v in $\{V_1, V_2\}$ **do**
    Find v' in $\mathrm{N}(v)$ such that $p(v, v')$ is largest
    Define $\mathrm{N}_\eta(v)$ as all vertices x with $p(x, v) \geq \eta \times p(v, v')$
    Initialize $\mathrm{N}^e(v) = \mathrm{N}_\eta(v)$
    Let $v_o = V \setminus v$, $\mathrm{N}'_\eta(v) \coloneqq \mathrm{N}(v) \setminus \mathrm{N}_\eta(v)$, $\mathrm{C}(v) = \mathrm{N}'_\eta(v) \cap v_o$
    Let $S_v = getWJC(\mathrm{N}(v), \mathrm{N}(v_o))$
end for
for Each element c in C_1 **do**
    Let $S_c = getWJC((\mathrm{N}(1) \setminus c), \mathrm{N}(2))$
    if $w_c = p(V_1, c) + |S_c - S_1| \geq \eta_1$ **then**
        Add c to N^e_1
    end if
end for
Repeat above for C_2
Return $\mathrm{N}^e(V_1), \mathrm{N}^e(V_2)$

a *Word Graph* and a corresponding *Positional Index*. The *Graph Reducer* then iterates over the graph, detects MWEs and merges constituent nodes. The following subsections present more details.

Before we describe the graph reduction algorithms, we introduce some notation as follows:

1. i^{th} vertex is denoted as V_i
2. The neighborhood of a vertex V, i.e. a set of vertices up to a depth of k, is denoted as $\mathrm{N}_k(V)$
3. Immediate neighborhood of a vertex V i.e. $\mathrm{N}_1(V)$ is denoted simply as $\mathrm{N}(V)$
4. $p(V)$ and $p(V_i, V_j)$ represent the prior and joint probabilities respectively

The process of graph reduction occurs in three phases : (a) Context determination (b) Local graph reduction and (c) Candidate pruning. Phases (a) and (b) operate on a neighborhood of a pair of vertices. The third phase however iterates over the graph and determines which vertex pairs to examine as we explain below.

3.2.1 Context Determination

We first determine a context (i.e. a set of vertices) for comparison. The basic idea of the algorithm is to define a context by using only *valuable* vertices in a given neighborhood. The inherent value is established in two ways : (a) the edge weight as compared to the maximum edge weight and (b) the contribution of the said vertex to the similarity / dissimilarity between the vertices being compared. We present Algorithm 1 that determines this context (or *"cross-enriched" neighborhood*).[2]

[2] getWJC() refers to weighted Jaccard coefficient

Algorithm 2 LOCAL GRAPH REDUCTION

Input: The sets : $\mathrm{C}(i,j)$, $\mathrm{U}(i)$ and $\mathrm{U}(j)$
Output: MWE candidates M
Initialize $M \leftarrow \emptyset$
Let the set $\mathrm{U} \coloneqq \mathrm{U}(i) \cup \mathrm{U}(j)$, $|\mathrm{U}| = k$
Let $A = zeros(k,k)$
Construct adjacency matrix where $A(x,y) = p(\mathrm{U}_x, \mathrm{U}_y) + \sum_c^{\mathrm{C}(i,j)} p(c, \mathrm{U}_y)$
for All x,y within the same partition **do**
    if $A(x,y) \gg A(y,x)$ **then**
        Delete U_y locally
    else if $A(x,y) \approx A(y,x)$ **then**
        Add pair $< \mathrm{U}_x, \mathrm{U}_y >$ to M
    end if
end for
Return M

Algorithm 3 GENERATING MWE CANDIDATES

Input: A word graph $G = (V, E, W)$, cross-enrichment parameter η, word rarity parameter ζ, co-occurrence parameter κ, positional index idx
Output: MWE candidates
Initialize $M_{op} \leftarrow \emptyset$
Let V_d be the vertices V sorted by descending order of degree
Initialize $\mathrm{M} \leftarrow \emptyset$
for $< V_i, V_j >$ in V_d **do**
    $\mathrm{N}^e(V_i), \mathrm{N}^e(V_j)$ = $crossEnrich(V_i, V_j, \eta)$
    compute $\mathrm{C}(i,j), \mathrm{U}_i, \mathrm{U}_j$
    $\mathrm{M} \leftarrow reduce(\mathrm{C}(i,j), \mathrm{U}_i, \mathrm{U}_j)$
end for
$\mathrm{M} \leftarrow filter(\mathrm{M}, \zeta, \kappa)$
for Group g in M **do**
    $M_{op} \leftarrow expandPhrase(g, idx)$
end for
Return M_{op}, G

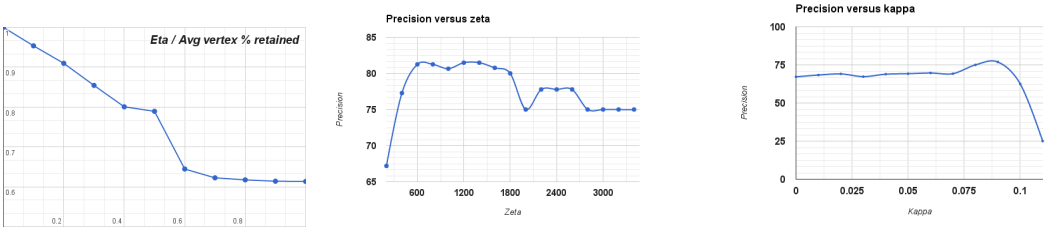
For a given vertices V_i and V_j, this algorithm effectively partitions their joint neighborhood into four disjoint sets:

1. Common vertices, $\mathrm{C}(i,j) \coloneqq \mathrm{N}^e(V_i) \cap \mathrm{N}^e(V_j)$
2. Uncommon vertices of i, $\mathrm{U}(i) \coloneqq \mathrm{N}^e(V_i) \setminus \mathrm{C}(i,j)$
3. Uncommon vertices of j, $\mathrm{U}(j) \coloneqq \mathrm{N}^e(V_j) \setminus \mathrm{C}(i,j)$
4. Ignored vertices, $\bigcup_k^{i,j} \mathrm{N}(V_k) \setminus \mathrm{N}^e(V_k)$

We only care about the common (C) and uncommon (U) vertices which act as inputs to the next phase.

3.2.2 Local Graph Reduction

In the next phase, we consider the sub-graph created by these three sets and perform local graph reductions as outlined in Algorithm 2. Essentially, we represent the local graph as a compressed adjacency matrix. For a given cell, $A(x,y)$, the weight in the matrix is set to the edge weight between vertices x and y plus the sum of weights from all common vertices to y. We then reduce the graph by either deleting



(a) Eta versus retained neighborhood

(b) Precision versus Zeta

(c) Precision versus kappa

Figure 2: Parameter tuning

a vertex if it is dominated by another node (i.e. this indicates that the dominated node never occurs independently) or by merging two vertices if their weights are equivalent (i.e. the given pair almost always co-occur - our original assumption). Note that these reductions are *local*.

3.2.3 Candidate Pruning & Phrase Expansion

In the final phase as illustrated in Algorithm 3, we output the final list of MWEs using a two step process. We first iterate over the graph that in turn calls Algorithm 1 and Algorithm 2. Next, we eliminate false positives based on two parameters : word rarity (ζ) and co-occurrence (κ). The former eliminates candidates that are composed of frequently occurring words, i.e. typical stopwords whilst the latter ensures a lower bound on the number of co-occurrences of the words that constitute the candidates. Finally, we use the positional index to reorder and expand the phrases as needed before outputting the final result.

A note about graph iteration is pertinent here. For a pair of vertices V_i and V_j input to the algorithms, it can be observed that the actual merge occurs on the vertices within the neighborhood of V_i, V_j and not on the vertices themselves. Thus, in order to cover as much graph as quickly as possible, the easiest strategy is to pick V_i, V_j in decreasing order of degree. Note that this also guarantees iteration in O(V) time.

Finally, although our method does seem similar to enumerating all n-grams and using some AM, we contend that this method can differentiate between nuances of usage due to the pairwise or cross-vertex iteration. In a typical n-gram approach, such contextual information is lost whereas in our method, it is equivalent to evaluating the n-grams in a limited context and is hence, more powerful. We now present details of parameter estimation and a short discussion on parameter sensitivity.

3.3 Parameter estimation

As we saw in Section 3.2, we use the following parameters:

- Enrichment parameter η : Determines which vertices in the current neighborhood will be considered
- Word rarity parameter ζ : Determines the level of rarity for a vertex to be considered
- Co-occurrence parameter κ : Determines the co-occurrence probability for an edge to be considered

We used the MH370 dataset (refer Table 2[3]) to find the optimum values of these parameters except for the *Enrichment parameter (η)* as described below. We obtained the value of η by evaluating the effect of varying η on a set of vertices and the neighboring vertices retained. We found a value of 0.6 to be a reasonable balance between over-pruning and retaining most vertices. Refer Figure 2 that demonstrates that a value of $\eta = 0.6$ does seem to have a large discriminatory power.

For estimating ζ and κ, we first used Algorithm 3 in a parameter-less mode (i.e. without filtering) and obtained all potential MWE candidates. For all such candidates, we established if the phrase indeed is a

[3]Collected when the MH370 flight had disappeared and investigation was underway

Sno	Dataset Name	# Tweets	Avg length	Vocabulary	Stopwords %	OOV (non English) %	Singleton %	Lang count
0	MH370	8,556	12.76	13,578	33.57	38.12	51.37	67
1	Brexit	18,488	10.96	23,734	30.74	40.40	34.9	58
2	DeleteYourAccount	2,244	6.85	2,445	44.45	24.39	42.33	45
3	Euro16	10,577	9.44	15,077	19.05	54.44	40.76	56
4	Giroud	6,697	9.35	8,576	17.91	58.10	45.98	51
5	PresidentObama	2,153	10.46	2,934	35.21	29.16	39.09	38
6	Pride	8,743	9.86	9,996	36.79	25.90	43.41	55
7	CalvinHarris	2,900	12.46	5,977	25.38	32.53	57.24	40
8	PokemonGO	7,019	10.81	15,068	32.14	44.22	64.32	60

Table 2: Dataset details

S.no	Phrase	DistScore	PhraseScore	HashtagScore	StopwordScore	Notes
1	clapback season	0.81	1	1	0	Ideal case : High scores for all three scores and no stopwords
2	Hillary Clinton	0.48	1	1	0	Named Entity but tokens can appear far apart
3	delete emails	0.37	1	0.04	0	Phrase query alone can be misleading
4	right now	0.45	1	1	1	Other measures compensate for lack of high distance score
5	if you	0.7112	1	1	2	High scores do not always mean MWEs

Table 3: Examples of need for four features

MWE using the Microsoft Web Language Model API [4] and the PMI metric [5]. We then measured system precision by varying each of the parameters independently as shown in Figure 2. We found the optimal values to be $\zeta = 1000$ and $\kappa = 0.01$.

4 Data and Experiments

Dataset	Dice	PMI	LogL	TwoT	T-Score	GRePE	# Candidates	Actual MWEs
Brexit	37.87	18.60	43.69	15.14	24.76	**62.40**	737	193
DeleteYourAccount	50.38	42.65	42.04	30.44	36.70	**66.14**	110	47
Euro16	32.48	14.40	**63.42**	39.28	61.17	42.36	328	67
Giroud	21.69	9.75	**78.38**	47.94	78.25	50.65	62	29
PresidentObama	**95.58**	86.26	92.72	2.04	92.72	59.93	36	15
Pride	54.37	40.18	**58.34**	15.30	39.41	51.55	137	56
CalvinHarris	**89.09**	41.31	66.52	73.62	77.58	84.17	33	21
PokemonGO	28.95	34.63	19.04	8.34	15.24	**58.46**	85	25
MAP / Total	51.30	35.97	58.02	29.01	53.23	**59.46**	791	453

Table 4: Experimental results

4.1 Datasets and data collection

As outlined in Section 1, our primary focus lies in extracting and analyzing MWEs from short text documents, namely tweets. Given the diverse nature of users, languages employed and topics discussed on Twitter[6], we wanted to achieve as broad coverage as possible. For over two weeks[7], we collected tweets for selected trending topics at different times of day. The choice of the selected topics was based on volumes as reported by Twitter plus the perceived global reach of the topic itself. However, for the final analysis we only used a subset of our crawled data as any sets with less than 2000 unique tweets were discarded. Although Twitter provides its own language identification, we used *langid* (Lui and Baldwin, 2012) for our use to allow generalization to other data sources (like Facebook) later.

A summary of the datasets is provided in Table 2 that captures the language and vocabulary spread for each hashtag. Note that the volume of tweets notwithstanding, each HashTag has tweets in at least 30 different languages, the average tweet length is only about 10 words and the word frequency distribution has a significant long-tail with about 40% of the words occuring just once.

4.2 Automatic Evaluation

As outlined in Section 2, we evaluate our system on precision as against recall and compare the system generated MWEs with those generated by standard AMs. Since we are computing Average Precision,

[4]https://www.microsoft.com/cognitive-services/en-us/web-language-model-api

[5]We tested different AMs and found PMI to be the most effective in this scenario

[6]https://about.twitter.com/company

[7]Roughly June 21 2016 - July 10 2016

the metric value is sensitive to the size of the result set considered. Given that our system produces limited number of MWEs, we restrict the output of compared AMs to be equal to the number of MWEs generated by our system.

In order to ascertain if a generated phrase is indeed a MWE, we performed two levels of evaluation. At the first level we use the Twitter Search API[8] as follows. For every candidate $W = (w_1, w_2)$, we execute three queries while restricting each query to top 25 unique results [9] : (a) w_1 w_2 (which is equivalent to w_1 AND w_2) (b) the phrase "w_1 w_2" (c) concatenation w_1w_2. Each result set is then converted to a corresponding numeric score as below

1. DistanceScore = Average normalized token distance between tokens w_1 and w_2
2. PhraseScore = Number of returned results / 25
3. HashtagScore = Number of returned results / 25

While the latter two scores approximate the probability of the phrase occurring either as separate words or concatenated together, the first score is a proxy for how frequently do the constituent words appear next to each other (as in a phrase) versus co-occurring in a tweet. Additionally, we add a fourth parameter, an integer stopword score $[0, 2]$ that acts as a regularization parameter to penalize phrases that contain stopwords which are bound to return a large number of results. We present some examples to illustrate the need for all four values in Table 3. We trained a simple multinomial logistic regression classifier on the MH370 dataset on manually evaluated MWE candidates with a 70% true label precision.

As second layer of screening, we assign one of the 15 POS labels as listed in Table 1 and double check that the extracted candidates are in fact MWEs. Note that we translate phrases from languages other than English into English before assigning the POS tags. We admit this is slightly lossy but we view at as a way to project all MWEs in the same token space for simplicity. Thus, for each dataset, we compute the Average Precision by using the true class labels obtained as explained above. We present the results in Table 4 along with Mean Average Precision (MAP).

We additionally compare the overlap between our method and the different AMs in Table 5a as well as splits by POS tag type in Table 5b. These tables show that although the different AMs do not necessarily generate the same candidate list (except LogLikelihood and T-score), the comparable POS split percentages indicate inherent bias within the dataset.

4.3 Discussion of results

We must take a moment to explain and examine the results. Although, it may not seem that our method is a vast improvement over other AMs when looking at the MAP, it must be noted that we do not produce "ranked" results as such and only candidates. We used a fixed ordering based upon the co-occurrence probability of phrases and a better ranking mechanism may exist but was not explored. The performance of the AMs is also bound to suffer when the full result sets are used. Further, except for the PresidentObama dataset, our method places within top 3 where it is not the best performing method. Comparing against Table 2, the method seems to suffer for predominantly English datasets (low OOV% - Pride, PresidentObama etc) but better for multilingual datasets (Brexit, PokemonGO). Thus, we could in principle augment our method with either AMs or existing POS based approaches for English to further improve performance. However, it can be concluded that overall the method returns a small and fairly precise set of MWEs as compared to AMs and enumerating all bigrams.

5 Future Work and Conclusions

In summary, we can enumerate our contributions as (a) we presented a language agnostic method for extracting MWEs from Twitter (b) we explored the performance of different AMs in a similar setting and (c) we showed a method for automatic evaluation of extracted MWEs. As an extension to this work, we would like to further analyze our results and study the effect of Twitter and social media specific features

[8] https://dev.twitter.com/rest/public/search

[9] With a page size of 10 tweets, this seemed a good choice for tweet depth without running too many queries

Measure	**Dice**	**PMI**	**LogL**	**TwoT**	**T-Score**	**GRePE**
Dice	NA	0.00	0.00	0.00	0.00	2.23
PMI	0.00	NA	0.00	0.00	0.00	0.00
LogL	0.00	0.00	NA	0.00	59.53	3.00
TwoT	0.00	0.00	0.00	NA	5.68	0.00
T-Score	0.00	0.00	59.53	0.00	NA	1.67
CDAM	2.23	0.00	3.00	0.00	1.67	NA

(a) AM Overlap

Measure	ADJP	NP	ADVP	VP
Dice	2.81	85.92	1.41	9.86
PMI	5.77	73.08	0.00	21.15
LogL	1.11	75.82	0.00	25.93
TwoT	0.05	52.50	0.00	42.50
T-Score	1.45	72.46	0.00	26.09
GRePE	1.64	83.61	0.00	14.75
Avg	**2.96**	**73.90**	**0.23**	**22.91**

(b) Split by POS tags

Table 5: Comparison between AMs

on MWE usage. Namely does internet language, hashtags and code switching impact how MWEs are used? We would also like to explore if the extracted MWEs can be utilized for other downstream tasks like generating summaries or automatic bilingual tweet alignment. We believe such work would help in developing resources for resource poor languages as well as aid in better understanding and modeling language usage on social media.

References

Muhammad Abdul-Mageed, Sandra Kübler, and Mona Diab. 2012. Samar: A system for subjectivity and sentiment analysis of arabic social media. In *Proceedings of the 3rd workshop in computational approaches to subjectivity and sentiment analysis*, pages 19–28. Association for Computational Linguistics.

Mirna Adriani and CJ Van Rijsbergen. 2000. Phrase identification in cross-language information retrieval. In *Content-Based Multimedia Information Access-Volume 1*, pages 520–528. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

Tetske Avontuur, Iris Balemans, Laura Elshof, Nanne Van Noord, and Menno Van Zaanen. 2012. Developing a part-of-speech tagger for dutch tweets. *Computational Linguistics in the Netherlands Journal*, 2:34–51.

Shane Bergsma, Paul McNamee, Mossaab Bagdouri, Clayton Fink, and Theresa Wilson. 2012. Language identification for creating language-specific twitter collections. In *Proceedings of the second workshop on language in social media*, pages 65–74. Association for Computational Linguistics.

Marine Carpuat and Mona Diab. 2010. Task-based evaluation of multiword expressions: a pilot study in statistical machine translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 242–245. Association for Computational Linguistics.

Joaquim Ferreira Da Silva, Gaël Dias, Sylvie Guilloré, and José Gabriel Pereira Lopes. 1999. Using localmaxs algorithm for the extraction of contiguous and non-contiguous multiword lexical units. In *Portuguese Conference on Artificial Intelligence*, pages 113–132. Springer.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *RANLP*, pages 198–206.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Spence Green, Marie-Catherine De Marneffe, John Bauer, and Christopher D Manning. 2011. Multiword expression identification with tree substitution grammars: A parsing tour de force with french. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 725–735. Association for Computational Linguistics.

Mengdie Hu, Shixia Liu, Furu Wei, Yingcai Wu, John Stasko, and Kwan-Liu Ma. 2012. Breaking news on twitter. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2751–2754. ACM.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A Smith. 2014. A dependency parser for tweets.

Anoop Kunchukuttan and Om Prakash Damani. 2008. A system for compound noun multiword expression extraction for hindi. In *6th International. Conference on Natural Language Processing*, pages 20–29.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM.

Nikhil Londhe, Vishrawas Gopalakrishnan, Aidong Zhang, Hung Q Ngo, and Rohini Srihari. 2014. Matching titles with cross title web-search enrichment and community detection. *Proceedings of the VLDB Endowment*, 7(12):1167–1178.

Marco Lui and Timothy Baldwin. 2012. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, pages 25–30. Association for Computational Linguistics.

Antonio Moreno-Ortiz, Chantal Pérez-Hernández, M Ángeles Del-Olmo, et al. 2013. Managing multiword expressions in a lexicon-based sentiment analysis system for spanish. *NAACL HLT 2013*, 13:1.

Joakim Nivre and Jens Nilsson. 2004. Multiword units in syntactic parsing. *Proceedings of Methodologies and Evaluation of Multiword Units in Real-World Applications (MEMURA)*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, pages 1320–1326.

Darren Pearce. 2002. A comparative evaluation of collocation extraction techniques. In *LREC*.

Ted Pedersen, Satanjeev Banerjee, Bridget T McInnes, Saiyam Kohli, Mahesh Joshi, and Ying Liu. 2011. The ngram statistics package (text:: nsp): A flexible tool for identifying ngrams, collocations, and word associations. In *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, pages 131–133. Association for Computational Linguistics.

NH Rais, MT Abdullah, and RA Kadir. 2011. Multiword phrases indexing for malay-english cross-language information retrieval. *Information Technology Journal*, 10(8):1554–1562.

Carlos Ramisch, Aline Villavicencio, and Christian Boitet. 2010. Multiword expressions in the wild?: the mwetoolkit comes in handy. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 57–60. Association for Computational Linguistics.

Carlos Ramisch, Vitor De Araujo, and Aline Villavicencio. 2012. A broad evaluation of techniques for automatic acquisition of multiword expressions. In *Proceedings of ACL 2012 Student Research Workshop*, pages 1–6. Association for Computational Linguistics.

Ines Rehbein. 2013. Fine-grained pos tagging of german tweets. In *Language Processing and Knowledge in the Web*, pages 162–175. Springer.

Zhixiang Ren, Yajuan Lü, Jie Cao, Qun Liu, and Yun Huang. 2009. Improving statistical machine translation using domain bilingual multiword expressions. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*, pages 47–54. Association for Computational Linguistics.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM.

Ivan A Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. In *Computational Linguistics and Intelligent Text Processing*, pages 1–15. Springer.

Nathan Schneider, Spencer Onuffer, Nora Kazour, Emily Danchik, Michael T Mordowanec, Henrietta Conrad, and Noah A Smith. 2014. Comprehensive annotation of multiword expressions in a social web corpus.

Yutaro Shigeto, Ai Azuma, Sorami Hisamoto, Shuhei Kondo, Tomoya Kouse, Keisuke Sakaguchi, Akifumi Yoshimoto, Frances Yung, and Yuji Matsumoto. 2013. Construction of english mwe dictionary and its application to pos tagging. In *Proc. of the 9th Workshop on Multiword Expressions*, pages 139–144.

R Mahesh K Sinha. 2011. Stepwise mining of multi-word expressions in hindi. In *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, pages 110–115. Association for Computational Linguistics.

Thamar Solorio and Yang Liu. 2008. Part-of-speech tagging for english-spanish code-switched text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1051–1060. Association for Computational Linguistics.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Gohneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, et al. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of The First Workshop on Computational Approaches to Code Switching*, pages 62–72. Citeseer.

Yulia Tsvetkov and Shuly Wintner. 2014. Identification of multiword expressions by combining multiple linguistic information sources. *Computational Linguistics*, 40(2):449–468.

Tim Van de Cruys and Begona Villada Moirón. 2007. Semantics-based multiword expression extraction. In *Proceedings of the Workshop on A Broader Perspective on Multiword Expressions*, pages 25–32. Association for Computational Linguistics.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. Pos tagging of english-hindi code-mixed social media content. In *EMNLP*, volume 14, pages 974–979.

Incremental Global Event Extraction

Alex Judea and **Michael Strube**
Heidelberg Institute for Theoretical Studies gGmbH
Schloss-Wolfsbrunnenweg 35
69118 Heidelberg, Germany
(alex.judea|michael.strube)@h-its.org

Abstract

Event extraction is a difficult information extraction task. Li et al. (2014) explore the benefits of modeling event extraction and two related tasks, entity mention and relation extraction, jointly. This joint system achieves state-of-the-art performance in all tasks. However, as a system operating only at the sentence level, it misses valuable information from other parts of the document. In this paper, we present an incremental approach to make the global context of the entire document available to the intra-sentential, state-of-the-art event extractor. We show that our method robustly increases performance on two datasets, namely ACE 2005 and TAC 2015.

1 Introduction

*But the **strikes** prove controversial.*

In many cases, it is not sufficient to look at one sentence when extracting events. In our example, *strikes* has no potential event arguments, and the context is not sufficient to disambiguate it correctly: Is it the trigger of an ATTACK event because it actually means *bomb strikes*? Or is it not a trigger at all because it refers to the industrial action?

*Soon after dawn on this fourth day, confirmation of the ship's first **strike** arrived . . . This is the first of an unknown number of **strikes** we'll conduct during our watch in "operation enduring freedom" . . . But the **strikes** prove controversial.*

Given the other sentences it is easier to infer that *strikes* is indeed the trigger of an ATTACK event. In this paper, we present a system that makes the global context of a document available to a state-of-the-art event extractor. Looking at a broader context also benefits argument detection. For example, within one document entities play coherent roles in different events. Consider the following text:

*Sam Waksal was **sentenced** to seven years and three months in federal prison . . . He's being released on his own cog in a sans before he's to **report** to jail.*

Looking only at individual sentences, it is hard to predict that *report* triggers an ARREST-JAIL event, which in turn makes it hard to predict that *he* is the Person of this event. If the system looks at the entire document and knows that *he* and *Sam Waksal* are coreferent, it can better infer that the Defendant of a SENTENCE event can be the Person of an ARREST-JAIL event, which in turn makes it easier to infer that *he* is this person.

In this paper, we present a method to incorporate the global, document-wide context into the decision process of a system that predicts entity mentions, events, and relations jointly. We use features that are based on the 'one sense per discourse' assumption (Gale et al., 1992), a concept widely used in Word Sense Disambiguation (e.g., Navigli and Lapata (2007)), and on the coherence of roles an entity plays in different events. We show that our method robustly increases performance on two datasets, namely ACE 2005 and TAC 2015.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2279–2289, Osaka, Japan, December 11-17 2016.

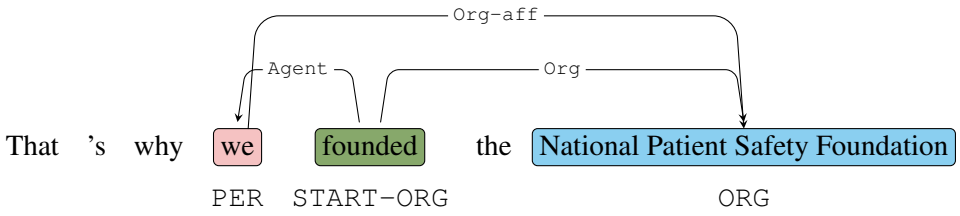


Figure 1: The configuration of a sentence. Depicted are two entities, a trigger, and the semantic relations and event argument relations between them.

The paper is structured as follows. Section 2 describes the task in more detail. Section 3 puts our work in context of other approaches to event extraction. Section 4 describes an intra-sentential, state-of-the-art system introduced by Li et al. (2014). In Section 5 we present *Incremental Global Inference*, a multi-pass procedure that makes the global context of a document accessible to the joint decoding of our intra-sentential event extractor. Section 6 reports evaluation results and Section 7 gives conclusions.

2 Task Description

Event extraction is an information extraction task where mentions of predefined event types are extracted from texts. We follow the task definition of the Automatic Content Extraction (ACE) program of 2005 which defines 33 event types, organized in eight categories. We also evaluate on another dataset, namely on the Event Nugget data of TAC 2015 (Mitamura et al., 2015).

In ACE, events are annotated only intra-sentential. Each event type has roles, e.g., `START-ORG`, an event indicating the founding of an organization, has the roles `Agent` and `Org`, whereas `DIE`, an event indicating the death of a person, has the roles `Agent`, `Victim`, and `Instrument`. The roles `Place` and `Time` are shared by all event types.

Roles are filled by zero or more *arguments*, that is, spans of text. The same span of text may be shared by multiple events as arguments, and may fill different roles in each of them. Finally, every event is indicated by a *trigger*.

Besides event annotations, ACE provides annotations of entity mentions and semantic relations. Detecting entity mentions is a task strongly related to event extraction because most arguments are mentions of persons, locations, organizations, etc.[1] Semantic relations and event arguments are also related because they coincide often with the start or end points of arguments.

Consider Figure 1. Depicted is the sentence *That's why we founded the National Patient Safety Foundation*. We can find two entity mentions, namely *we* as a `PER` and *National Patient Safety Foundation* as an `ORG` mention. Furthermore, there is one trigger of a `START-ORG` event, namely *founded*. This event has two arguments, namely *we* filling the role `Agent`, and *National Patient Safety Foundation* filling the role `Org`. Finally, there is an `Org-aff` relation between *we* and *National Patient Safety Foundation*.

While ACE provides annotations for all tasks involved (entity mentions, event triggers, event arguments), the TAC 2015 Event Nugget data provides only annotations for event triggers. The trigger annotation schemes are similar, the two data sets share many event types, and events occur only intra-sentential. However, some event types in TAC are more fine-grained, e.g., `TRANSPORT` in ACE was split into `TRANSPORT-PERSON` and `TRANSPORT-ARTIFACT` in TAC.

3 Related Work

The base system we use is the one described in Li et al. (2014)[2]. To our knowledge it is the only system to predict entity mentions, relations, event triggers, and event arguments jointly. It achieves state-of-the-art performance in all four tasks.

Many approaches to event extraction, including our base system, do not cross sentence boundaries (Grishman et al., 2005; Ahn, 2006; Lu and Roth, 2012; Li et al., 2013; Li et al., 2014). Only a few

[1] Some arguments are mentions of points in time, amounts of money, etc.

[2] This system is a combination of the systems described in Li et al. (2013) and Li and Ji (2014).

approaches go beyond sentences (Liao and Grishman, 2010; Hong et al., 2011) or beyond documents (Ji and Grishman, 2008) in order to exploit richer contexts for the extraction of events.

Recently, three deep learning systems were proposed. Nguyen and Grishman (2015) use a Convolutional Neural Network (CNN) to detect event triggers. They achieve good trigger detection performance, but they do not tackle the full event task. Chen et al. (2015) propose a more complicated version of a CNN which is able to detect multiple arguments in addition to triggers. Their system is a pipeline system (it predicts triggers and arguments separately) and suffers from error propagation. Nguyen et al. (2016) combine the advantages of joint models, explicit feature engineering and neural networks. They propose a joint, bi-directional recurrent network which additionally uses the features in Li et al. (2013). However, both Chen et al. (2015) and Nguyen et al. (2016) rely on gold entity mentions for trigger prediction.

The cross-sentential systems proposed in Liao and Grishman (2010) and Yang and Mitchell (2016) are closest to ours. We will describe them in the following.

Liao and Grishman (2010) propose a pipeline system that performs easy-first global inference. Local classifiers find triggers and arguments based solely on local information. Confident decisions are collected and used to inform global trigger and argument classifiers. In the local, intra-sentential phase the system performs pattern matching to align the entity mention context of a trigger to some known patterns. Similar to our approach, confident decisions are collected during the intra-sentential pass and used to infer harder cases.

Yang and Mitchell (2016) propose a pipeline system with global inference. In contrast to most other approaches, it also predicts entity mentions. Yang and Mitchell (2016) first train two Conditional Random Fields to generate mention and trigger candidates for a document. They only keep the 50 best mention candidates, and the 10 best trigger candidates. Then, they train three classifiers to capture entity mentions, within-event structures, and event-event relations. Finally, they formalize an Integer Linear Program that, given the local classifiers and the (global) event-event classifier, produces a globally optimal solution, instead of refining locally-optimal solutions with global information.

In terms of label inference (assigning types to candidates), our approach lies between Liao and Grishman (2010) and Yang and Mitchell (2016). The former apply global inference after local inference, the latter model both jointly. We let local and global inference inform each other and iteratively refine both.

A major difference between Liao and Grishman (2010) one the one hand, and Yang and Mitchell (2016) and our approach on the other is that the first approach uses gold entity mentions, while the other two use predicted entity mentions. Comparing full inference, we note that Yang and Mitchell (2016) use a pipeline approach: They first tag sentences for potential entity mention and event trigger candidates, and apply label inference afterwards. In contrast, we model both jointly.

4 Intra-Sentential Event Detection

In this section we describe our reimplementation of the system presented in Li et al. (2014), a state-of-the-art event detector. It employs joint decoding of entity mentions, events, and relations in order to make use of a rich feature set, including features which capture interdependencies of different subtasks. It uses a structured perceptron with beam search to explore different segmentations of a sentence and possible connections between segments.

4.1 Terminology

Following Li et al. (2014) we will call a specific segment of a sentence a *node* and a relation connecting two nodes an *arc*. The entire set of nodes and arcs for a sentence will be called *configuration*.

Formally, a node n is a tuple (b, e, n_t) where b and e are begin and end offsets, and n_t is an entity, event, or 'null' type. We encode offsets as token indices. An arc is a tuple (e_1, e_2, a_t) with $e_1 < e_2$, where e_1 and e_2 are the end offsets of the non-overlapping nodes of the connection, and a_t is a semantic relation or event argument type. Because semantic relations are directed and nodes are ordered by offset, we have to mark relation direction by defining an 'inverse' version of each semantic relation type.

4.2 Decoding and Training

Our decoding (producing configurations for a given sentence) is built around the idea of *Semi-Markov Chains* (Sarawagi and Cohen, 2004). Here, in contrast to more traditional token sequence labelers like CRFs, decoding produces segments of arbitrary length, possibly spanning multiple tokens and is therefore able to use features characterizing entire segments, and not only individual tokens. In the following we describe the decoding and training procedures in detail. Algorithm 1 formalizes the procedure.

We start our description with the method GENERATE NODES in Algorithm 1. Given a sentence s with tokens $t_1 \dots t_m$, the procedure generates nodes of different types and lengths ending at each t_i. It starts with t_1 and generates nodes of length one with different types, e.g., $(1, 1, \texttt{PER})$ or $(1, 1, \texttt{START-ORG})$. The procedure moves on to the next token, t_2. Now, it generates segments of length one and length two, e.g., $(2, 2, \texttt{PER})$ or $(1, 2, \texttt{PER})$, and adds them to the previously constructed configurations such that nodes do not overlap and there are no gaps.

Algorithm 1:
BEAM SEARCH$(s = t_1 \dots t_m, y, z, \textit{training})$

```
1  beam b ← ∅
2  for i = 1 ... m do
3      b(i) ← GENERATE NODES(b, i)
4      b(i) ← top_z(b(i))
5      if training ∧ y_i ∉ b(i) then
6          UPDATE(b(i, 1), y_i)
7          exit
8      for j = i − 1 ... 0 do
9          b(i) ← GENERATE ARCS(b, j, i)
10         b(i) ← top_z(b_i)
11         if training ∧ y_i ∉ b(i) then
12             UPDATE(b(i, 1), y_i)
13             exit
14 if training ∧ y ≠ b(m, 1) then
15     UPDATE(b(m, 1), y)
16     exit
17 return b(m, 0)
```

Some configurations may now contain two nodes which means that the procedure can predict arcs (GENERATE ARCS in Algorithm 1). It generates a compatible arc for each node pair, but does not overwrite configurations without arcs. This ensures that not predicting an arc between any two nodes is always a valid hypothesis. We collect type restrictions for arcs from the training data. For example, it is not possible that Organization-Affiliation relations hold between person mentions. Such restrictions keep the search space smaller and make learning easier.

Input to Algorithm 1 is a sentence s, a gold configuration y, and the beam size z. The beam b is a ranked list of hypotheses (that is, configurations) ending at any token position in the sentence. If the procedure is not in training mode, or if it did not make any prediction errors, it returns the top hypothesis ending at the last token position, $b(m, 1)$ for a sentence with m tokens.

Ideally, one would enumerate all possible configurations for a sentence and pick the best one. However, such an exhaustive enumeration is infeasible because the number of configurations grows exponentially with the number of tokens. Following Li et al. (2014) we approximate the enumeration by using *beam search*. We first expand configurations with new nodes and keep only the best z configurations. These are then expanded with arcs. During this process we again keep only the best z configurations. After arc generation, we move to the next position.

In training mode the procedure updates feature weights as soon as y_i, a prefix of the gold configuration, is not predictable anymore because it is no longer part of the beam. This is called *early update*. Huang et al. (2012) proved that, in structured perceptrons, standard updates may lead to bad performance with inexact search strategies such as beam search because there may be updates which actually lower the score of the gold solution, thus leading the model in a wrong direction. Early updates prevent this problem at the expense of higher training time. Note that we exit the procedure when early updates happen. There are two positions where early updates occur, namely after node generation and during arc generation (Lines 6 and 12).

4.3 Features

The feature set of our base system is complex because each subtask requires its own rich feature set. We can divide features into two broad categories: Static features, which do not depend on previous decisions, and dynamic features, which depend on previous decisions. Table 1 contains a feature summary struc-

Static	Common	lexical information* (segment tokens, context, dependencies)
		brown clusters, gazetteer entries*
	Mentions	character suffixes of length 3 and 4*
	Triggers	possible FrameNet frames in case of verbs*
	Arguments	trigger type, mention and mention context
		trigger-mention dependency/constituency paths
	Relations	entity types, contexts, extent overlaps
		mention-mention dependency/constituency paths
		syntactico-semantic structures (Chan and Roth, 2011)
Dynamic	Common	(event or entity) type bigrams
		consistencies (same text=same type, coordinations, pronouns)
	Triggers	dependency path between trigger pairs
	Arguments	characterize mentions filling the same role in the same event
		characterize triggers sharing a mention
	Relations	characterize triangles like A-C, B-C
		characterize constructions where the parts tend to have same relation types, e.g., coordinations
	Joint Argument-Relation	relation types and overlapping argument types

Table 1: A description of the base system's feature set. Static features marked with * apply to the entire segment and not to each token in the segment.

tured in this way. All features are concatenated with the node or arc type under consideration. Argument features additionally come without types in order to characterize properties of arguments in general.

In order to simulate the feature set of a more traditional sequence labeler, the base system adds a BILU-tag ("B" for first token of a segment, "I" for an intermediate token, "L" for the last token, and "U" in case of length-1 segments) to each token feature of the node under consideration. If this node is an entity and the last node in the configuration is also an entity, it additionally adds the entity type of the last segment to token features, otherwise it adds a 'null' label.

5 Incremental Global Inference

Our base system is limited in two ways. It operates intra-sententially and is thus limited to the information in a single sentence. In some cases this may be sufficient to predict all entity mentions, events, and relations, in many others it is not. An approach operating on the document level has access to a broader context and may be able to resolve more difficult cases.

Even within a sentence the base system is limited to the left context of the token under consideration. The entire sentence is only available when decoding reaches the last token. At that point many decisions are not reversible anymore because of the approximate search procedure.

In the following we present Incremental Global Inference, a method to cope with both limitations. It makes the global, document-wide context available to the local decoding of our base system.

Algorithm 2:
INCREMENTAL GLOBAL INFERENCE (d)

```
1  decision map r = ∅
2  for i = 1 ... k do
3      for Sentence s ∈ document do
4          r ∪ SEARCH(s, y, z, training, r)
           // no updates
5  for Sentence s ∈ document do
6      r_s = SEARCH(s, y, z, training, r)
7      if error(r_s) then
           // updates
8          UPDATE(x*, y*)
9      r ∪ r_s
10 return r
```

5.1 Procedure

Incremental Global Inference is a multi-pass approach. We iterate several times through the document. The decisions in each iteration are input to the next iteration. We first perform standard decoding as described in Section 4 for all sentences in a document, and feed back the decisions in this step for a second pass. In the second pass we extract global features, that is, features measuring the similarity of a new node or arc to decisions made in the last iteration. We again keep the decisions for the entire document and continue with a third decoding pass, etc. Algorithm 2 formalizes the procedure.

Feature type	Applies to	Condition	Description
Local	n is trigger, m is entity	dependency graph matching known	graph connecting event type and lemmas of n, and entity type of m
			indicator
	n is argument, m is trigger	dependency graph matching known	graph connecting role of n, event type, and lemmas of m
			indicator
Global	n and m are triggers	always	The event types of n and m
		lemmas equal	The event types of n and m
		coreferent arguments	The event types of n and m
	n and m are arguments	coreferent arguments	the roles and event types of n and m

Table 2: New local and global features for a node n. Global features are for node pairs (n, m), where m may come from the entire document.

Input to Algorithm 2 is a document d, output is a structure r holding the configuration for each sentence in the document. The algorithm can be divided in two parts, namely *collection* and *final decoding*. We will now describe the two parts.

In the collection phase (Lines 2 - 4), the procedure iterates k times through the document. In each iteration, and for each sentence, it performs SEARCH, a slightly different version of Algorithm 1, the only difference being that feature updates are omitted. We perform updates in *final decoding*. The decoding decisions are recorded per sentence in the decision map r. The decision map r is updated such that the decisions for a sentence in one iteration are overwritten with the decisions for the same sentence in the next iteration. r is input for the next iteration.[3] With each iteration, and with each update of r, SEARCH has a more reliable view on mentions, triggers, and arguments in the entire document, helping it to find new nodes and arcs which would be hard to get otherwise.

After collection comes final decoding (Lines 5 - 8). Here, we again perform SEARCH. Now, we also perform weight updates as soon as they are necessary. The final output of the algorithm is the final sate of r.

5.2 New Features

In this subsection we describe the new features we use. First, we will describe new local features. Then, we will describe the global features needed for our Incremental Global Inference. Table 2 summarizes the features we describe in the following. The first column reports feature types (local or global), the second column reports class requirements (triggers or arguments), the third column reports conditions under which features fire, and the last column gives short descriptions.

Local Features

The first local feature we introduce to event extraction is based on so-called *hidden units* (Das et al., 2014). The hidden units of an event type are the exclusive triggers it has in the training data. Hidden units define a semantic space for their event types. Measuring semantic relations of the node under consideration with this space helps to find triggers never seen during training. For example, let the candidate trigger be *meeting* and the candidate event type be `MEET`. We have several hidden units of `MEET` sharing semantic relations with the predominant sense of *meeting*: *convention* and *summit* as WordNet hypernyms (Fellbaum, 1998), and *meet* as a morphological variant. We can draw features from the hidden units themselves and from the semantic relations involved.

The second local feature we introduce to event extraction is based on dependency graphs between triggers and arguments. We collect all these graphs in the training data and index them by the involved argument and entity types. For example, we collect all graphs between `MEET` events and their `Place` arguments. One such graph is the following: `MEET` $\xrightarrow{\text{nsubjpass}}$ set $\xleftarrow{\text{nmod:in}}$ `LOC/Place` , coming from sentences like *The meeting was set in Beijing*, or *The conference was set in New York*. The graph encodes that a

[3] Initially, the decision map is empty.

MEET event trigger is the passive subject of *set*, and a LOC entity/a Place argument is connected via the nominal modifier *in* to the same word.

Note that the base system also uses dependencies to characterize syntactic connections of triggers and arguments. We extend this feature type in two important ways. First, we normalize dependency graphs by entity types, event types, or roles. Second, we can utilize dependency graphs during trigger prediction. The base system is always limited to the left context. When it predicts a trigger, it misses arguments to the right of it. Our Incremental Global Inference makes the entire sequence of entity mentions/event arguments available, regardless of the actual decoding position. This increases the chance to find the MEET trigger in our example, which may in turn influence other decisions in the next decoding pass.

eventmeet trigger. However, Incremental Global Inference makes the entire mention sequence available, regardless of the actual decoding position, increasing the chance to find the MEET trigger in our example, which may in turn influence other argument or entity mention decisions in the next decoding pass.

Global Features

We now describe the global features we use for our Incremental Global Inference. The global feature function takes two arguments (n, m), where n is the argument or trigger under consideration, and m is a trigger or argument coming from the entire document.

We have three types of global features. The first applies to triggers and catches all event types present in the document. As Liao and Grishman (2010) show, certain event types often co-occur, e.g. ATTACK and DIE often co-occur in news documents.

Another feature type fires if two triggers have equal event types and equal lemmas[4]. This is based on the observation that one word tends to trigger the same event type within one document (Ji and Grishman, 2008). This is strongly related to the 'one sense per discourse' assumption (Gale et al., 1992).

The third feature type applies to triggers and arguments. It catches event types of events with coreferent arguments, and the roles entities play in events. In ACE, an entity usually plays the same role for events with the same type (which is again realated to the 'one sense per discourse' assumption) and plays coherent roles for events with different types (Liao and Grishman, 2010). For example, if an entity is the Target of an ATTACK event, it rarely becomes the Attacker, at least not within one document. However, the same entity may be the Victim of a DIE event, but never the Agent. In order to increase recall of our coreference resolution system, we regard all non-pronoun string matches as coreferent, as well as partial string matches if the involved entities are persons.

6 Evaluation

In the following, we describe the data sets and the evaluation metrics we used. We report evaluation numbers and discuss them.

6.1 Data and Evaluation Metrics

We devise evaluation on two data sets, namely on ACE 2005 and on the Event Nugget data of TAC 2015. For the TAC evaluation, we use the official training and test sets. We manually split the training set into 132 documents for training and 26 for development. The test set consists of 202 documents. We removed the lemmas 'it' and 'say' from decoding because they proved to be highly ambiguous (Reimers and Gurevych, 2015).

For the ACE evaluation, we adopt the evaluation setting of Li et al. (2014). We train on English documents and remove the two smallest and most informal parts of the data, 'conversational telephone speech' and 'Usenet newsgroups'. From the remaining 511 documents, 351 are used for training, 80 for development, and 80 for testing. For direct comparison we use the same data split as Li et al. (2014). We set the beam size to 8. For Incremental Global Inference we perform 3 iterations. After each training epoch, we measure performance on the development set (in terms of trigger and argument F_1) and use the model with the best overall performance for evaluation on the test set.

[4]In case of multi-word triggers, we take the lemma of each token.

	Triggers						Arguments					
	Identification			Classification			Identification			Classification		
System	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Baseline	**67.2**	62.2	64.6	**64.7**	59.9	62.2	70.7	36.6	48.2	57.3	29.7	39.1
IGI	67.0	**65.9**	**66.5**	64.2	**63.1**	**63.7**	**76.1**	**40.8**	**53.1**	**59.9**	**32.1**	**41.8**

(a) Trigger and argument performance. 'Identification' reports performance without trigger type/argument role assignment.

	ATTACK(206)			TRANSPORT(98)			DIE(49)			MEET(42)		
System	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Baseline	**72.1**	71.4	**71.7**	44.0	49.0	46.4	**64.6**	85.7	73.7	66.7	76.2	71.1
IGI	70.1	**71.8**	71.0	**48.1**	**53.1**	**50.5**	63.8	**89.8**	**74.6**	**70.2**	**78.6**	**74.2**

(b) Trigger classification performance for the four most frequent event types.

	Triggers			Arguments		
System	P	R	F_1	P	R	F_1
Li et al. (2014)	67.9	62.8	65.3	64.7	35.3	45.6
Li et al. (2014) rerun	66.4	60.2	63.1	61.3	30.2	40.4
Baseline	64.7	59.9	62.2	57.3	29.7	39.1

(c) Trigger and argument classification performance. This is a comparison of results published by Li et al. (2014), a version retrained by us ('rerun'), and our reimplementation of the system.

Table 3: Micro-averaged precision, recall, and F_1 for the baseline and IGI on the ACE 2005 test set, and for our baseline compared to published results in Li et al. (2014) and our run using their code. Numbers in bold are the better numbers for the respective measure. Numbers in parentheses are frequencies in the test set.

	Triggers					
	Identification			Classification		
System	P	R	F_1	P	R	F_1
Baseline	**87.3**	47.0	61.1	**73.3**	39.5	51.3
IGI	77.1	**54.7**	**64.0**	64.1	**45.5**	**53.3**

(a) Trigger performance. 'Identification' reports performance without event type assignment.

	ATTACK(591)			CONTACT(587)			TRANSF.MNY(554)			BROADCAST(510)		
System	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Baseline	**69.0**	38.4	49.3	**43.5**	**11.4**	**18.1**	**71.9**	29.1	41.4	**44.0**	20.8	28.2
IGI	67.3	**43.5**	**52.8**	33.5	11.1	16.6	55.4	**34.3**	**42.4**	42.5	**30.4**	**35.4**

(b) Trigger classification performance for the four most frequent event types.

Table 4: Micro-averaged precision, recall and F_1 for the baseline and IGI on the TAC 2015 test set. In TAC 2015 only event trigger annotations are available. Numbers in bold are the better numbers for the respective measure. Numbers in parentheses are frequencies in the test set.

We follow standard evaluation criteria for triggers and events (Ji and Grishman, 2008). A trigger is correct, if its span and event type match a reference trigger. An argument is correct, if its span, event type, and role match a reference argument.

Our evaluations report *identification* and *classification* of event triggers and arguments. An event trigger or argument is correctly identified if its offsets match a reference trigger or argument, and correctly classified, if there is an additional match in event type or role. In all evaluations, we report micro-averaged precision (P), recall (R), and F_1. We devise two evaluation lines: We evaluate trigger and argument identification and classification on the one hand, and trigger classification performance for the four most frequent event types in ACE 2005 and TAC 2015 on the other hand.

6.2 Experiments and Results

Table 3 reports evaluation results for triggers and arguments on ACE 2005. Comparing the baseline with IGI in Table 3a we can see that IGI gives a considerably higher recall and F_1 for triggers, and a higher precision, recall, and F_1 for arguments, both, in terms of identification and classification. For identification, IGI increases trigger performance by 1.9 F_1 points and argument performance by 4.9 F_1 points. For classification, IGI improves 1.5 F_1 points for triggers, and 2.7 F_1 points for arguments.

The same trends can be observed for TAC 2015 (Table 4a). TAC 2015 offers trigger annotations only. For IGI, trigger identification improves by 2.9 F_1 points, and classification improves by 2 F_1 points compared to the baseline.

We now look at the classification performance of the four most frequent event types for both, ACE 2005 (Table 3b) and TAC 2015 (Table 4b). On both data sets, IGI improves F_1 for three of the four most frequent event types. For ACE 2005, `Transport` and `Meet` have higher precision and recall compared to baseline performance. `Attack` loses precision with IGI, resulting in a similar F_1 as the baseline. For TAC 2015, `Contact` loses precision and recall with IGI, resulting in a lower F_1. In all other cases, IGI considerably increases recall, at the expense of precision. For ACE 2005, the event type with most F_1 improvement using IGI is `Transport` (+4.1 points), a very difficult event type. For TAC 2015, the event type with most F_1 improvement using IGI is `Broadcast`, again a very difficult event type.

Table 3c is as a comparison of our reimplementation of Li et al. (2014) and the numbers they report. The first column ('Li et al. (2014)') reports published results from the respective paper. Column 'Li et al. (2014) rerun' reports performance of the source code we received from the authors. 'Baseline' is our baseline, a reimplementation of their system. We first compare published results with our rerun of the reference system (Lines 1 and 2). We can see that the published evaluation numbers and the numbers for our rerun of the reference system (using source code we received from the authors) do not match. We can only speculate about the reasons. The biggest difference we can spot is that the source code we received does not use the output of a frame-semantic parser for FrameNet features. It uses all possible frames for a given word instead, if the number of possible frames does not exceed 2. Li et al. (2014) report a gain of 0.8 F_1 points for triggers, and a gain of 2.2 F_1 points for arguments when using a frame-semantic parser. Since the source code we received already uses frames, we expect to gain only a fraction of this improvements using a full-fledged frame-semantic parser. We believe the feature set has changed slightly until we received the source code. It is not easy to reproduce published results (Fokkens et al., 2013). Comparing our reimplementation with our rerun of the reference system (Lines 2 and 3) we can see that the respective performances are close.

7 Conclusions

We presented a method to make the global, document-wide context available to a state-of-the-art, intra-sentential event extractor. The method is an incremental approach: With every iteration it collects new triggers and arguments from the entire document and makes them available to the intra-sentential base system for another decoding pass. Our method leads to considerable increases in recall and F_1 for triggers across different data sets and different event types, and to considerable increases in all measures for arguments.

Lemma matches and coreferent arguments as nuclei for global decoding are only a start. Future research has to investigate other similarity measures based on, e.g., WordNet and word vectors, and to explore the benefits of discourse structure for event extraction.

Acknowledgments

This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first author has been supported by a HITS PhD scholarship. We thank the anonymous reviewers for their helpful comments.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events,* Sydney, Australia, 23 July 2006, pages 1–8.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Portland, Oreg., 19–24 June 2011, pages 551–560.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Beijing, China, 26–31 July 2015, pages 167–176.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass.

Antske Fokkens, Marieke van Erp, Marten Postma, Ted Pedersen, Piek Vossen, and Nuno Freire. 2013. Offspring from reproduction problems: What replication failure teaches us. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Sofia, Bulgaria, 4–9 August 2013, pages 1691–1701.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. One sense per discourse. In *Proceedings of the DARPA Speech and Natural Language Workshop,* New York, N.Y., 23–26 February 1992, pages 233–237.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. NYU's English ACE 2005 system description. Technical report, Department of Computer Science, New York University, New York, N.Y.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Portland, Oreg., 19–24 June 2011, pages 1127–1136.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Montréal, Québec, Canada, 3–8 June 2012, pages 142–151.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies,* Columbus, Ohio, 15–20 June 2008, pages 254–262.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Baltimore, Md., 22–27 June 2014, pages 402–412.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Sofia, Bulgaria, 4–9 August 2013, pages 73–82.

Qi Li, Heng Ji, Yu Heng, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 1846–1851.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics,* Uppsala, Sweden, 11–16 July 2010, pages 789–797.

Wei Lu and Dan Roth. 2012. Automatic event extraction with structured preference modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Jeju Island, Korea, 8–14 July 2012, pages 835–844.

Teruko Mitamura, Liu Zhengzhong, and Eduard Hovy. 2015. Overview of TAC KBP 2015 Event Nugget Track. In *Proceedings of the Eighth Text Analysis Conference,* National Institute of Standards and Technology, Gaithersburg, Maryland, USA, 16–17 November 2015.

Roberto Navigli and Mirella Lapata. 2007. Graph connectivity measures for unsupervised word sense disambiguation. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence,* Hyderabad, India, 6–12 January 2007, pages 1683–1688.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* Beijing, China, 26–31 July 2015, pages 365–371.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* San Diego, Cal., 12–17 June 2016, pages 300–309.

Nils Reimers and Iryna Gurevych. 2015. Event Nugget Detection, Classification and Coreference Resolution using Deep Neural Networks and Gradient Boosted Decision Trees. In *Proceedings of the Eighth Text Analysis Conference,* National Institute of Standards and Technology, Gaithersburg, Maryland, USA, 16–17 November 2015.

Sunita Sarawagi and William W. Cohen. 2004. Semi-Markov conditional random fields for information extraction. In *Proceedings of Advances in Neural Information Processing Systems 17.* Vancouver, B.C., Canada, 13–18 December 2004, pages 1185–1192.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299. Association for Computational Linguistics.

Hierarchical Memory Networks for Answer Selection on Unknown Words

Jiaming Xu[a,*], **Jing Shi**[a,*], **Yiqun Yao**[a], **Suncong Zheng**[a], **Bo Xu**[a], **Bo Xu**[a,b]
[a]Institute of Automation, Chinese Academy of Sciences (CAS). Beijing, China
[b]Center for Excellence in Brain Science and Intelligence Technology, CAS. China
{jiaming.xu, shijing2014, yaoyiqun2014}@ia.ac.cn
{suncong.zheng, boxu, xubo}@ia.ac.cn

Abstract

Recently, end-to-end memory networks have shown promising results on Question Answering task, which encode the past facts into an explicit memory and perform reasoning ability by making multiple computational steps on the memory. However, memory networks conduct the reasoning on sentence-level memory to output coarse semantic vectors and do not further take any attention mechanism to focus on words, which may lead to the model lose some detail information, especially when the answers are rare or unknown words. In this paper, we propose a novel Hierarchical Memory Networks, dubbed HMN. First, we encode the past facts into sentence-level memory and word-level memory respectively. Then, k-max pooling is exploited following reasoning module on the sentence-level memory to sample the k most relevant sentences to a question and feed these sentences into attention mechanism on the word-level memory to focus the words in the selected sentences. Finally, the prediction is jointly learned over the outputs of the sentence-level reasoning module and the word-level attention mechanism. The experimental results demonstrate that our approach successfully conducts answer selection on unknown words and achieves a better performance than memory networks.

1 Introduction

With the recent resurgence of interest in Deep Neural Networks (DNN), many researchers have concentrated on using deep learning to solve natural language processing (NLP) tasks (Collobert et al., 2011; Sutskever et al., 2014; Zeng et al., 2014; Feng et al., 2015). The main merits of these representation learning based methods are that they do not rely on any linguistic tools and can be applied to different languages or domains. However, the memory of these methods, such as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014) compressing all the external sentences into a fixed-length vector, is typically too small to accurately remember facts from the past, and may lose important details for response generation (Shang et al., 2015). Due to the drawback, these traditional DNN models encounter great limitation on Question Answering (QA), as a complex NLP task, which requires deep understanding of semantic abstraction and reasoning over facts that are relevant to a question (Hermann et al., 2015; Yu et al., 2015).

Recently, lots of deep learning methods with explicit memory and attention mechanism are explored for Question Answering (QA) task, such as Memory Networks (MemNN) (Sukhbaatar et al., 2015), Neural Machine Translation (NMT) and Neural Turing Machine (NTM) (Yu et al., 2015). These methods exploit a external memory to store the past sentences with a continuous representation and utilize attention mechanism to automatically soft-search for parts of the memory for prediction. Compared with NMT and NTM, MemNN, making multiple computational steps (termed as "hops") on the memory before making an output, is better qualified for textual reasoning tasks. However, for QA task, MemNN only conducts the reasoning on sentence-level memory and does not further take any attention mechanism to focus on words in the retrieved facts. More recently, Yu et al. (2015) constructed a *Search-Response*

*The first two authors contributed equally.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2290–2299, Osaka, Japan, December 11-17 2016.

pipeline where *Search* component uses MemNN to search the supporting sentences and *Response* component uses NMT or NTM to generate answer on the selected sentences. However, that work needs the supervision of the supporting facts to guide the training of *Search* component and the combination of these two components through a separate training way may hurt the performance. Along the direction of that work, we believe that a joint learning model can achieve a better performance by designing a hierarchical architecture, with sentence-level and word-level components, which has shown promising results on document modeling (Lin et al., 2015) and document classification (Yang et al., 2016).

Besides, rare and unknown word problem as an important issue should be considered in NLP tasks, especially for QA task, where the words that we are mainly interested in are usually named entities which are mostly unknown or rare words (Marrero et al., 2013; Gulcehre et al., 2016). In order to control the computational complexity, many methods limit the trained vocabulary size, which further leads to lots of low-frequency words outside the trained vocabulary (Li et al., 2016). Traditional methods directly mask the rare or unknown words with meaningless *unk* which may lose the important information for answer selection task. For example, given a set of sentences as follows:

1. Miss, what is your name?
2. Uh, my name is *Wainwright*.
3. Please tell me your passport number.
4. Ok, it is *899917359*.

Assume that the words *Wainwright* and *899917359*[1] are rare words or outside the trained vocabulary. If these words are discarded or replaced with *unk* symbol, any models may not be able to select the correct answers for response during testing.

Based on the above observations, this paper proposes a Hierarchical Memory Networks[2] (dubbed to HMN) for answer selection. Our method first maps the sentences into a sentence-level memory and reasoning module takes multiple hops on the sentence-level memory to soft-search the related sentences. Meanwhile, all words in the sentences are encoded into a word-level memory with recurrent neural networks. Then, we exploit k-max pooling to sample the most relevant sentences and feed these selected sentences into attention mechanism on the word-level memory to focus the words. Finally, the prediction is jointly learned over the outputs of the sentence-level reasoning module and the word-level attention mechanism. Our main contributions are three-fold:

(1). We proposed a novel hierarchical memory networks for answer selection, where the reasoning module is performed on sentence-level memory to retrieve the relevant sentences and the attention mechanism is applied on word-level memory to focus the words. This hierarchical architecture allows the model to have explicit reasoning ability on sentences and also focus on more fine-grained words.

(2). k-max pooling is exploited to sample the most relevant facts based on the results of the sentence-level reasoning and then feed these facts into word-level attention mechanism, which can filter the noise information and also reduce the computational complexity on word-level attention.

(3). We release four synthetic domain dialogue datasets[3], two from air-ticket booking domain and two from hotel reservation domain, where the answers are mostly rare or unknown words, and lots of answers should be reasoned based on some supporting sentences. The experimental results show that our approach can successfully conduct answer selection on unknown words.

2 Background: Memory Networks

Here, we give a brief description of memory networks which have shown promising results on QA tasks (Weston et al., 2015; Bordes et al., 2015; Sukhbaatar et al., 2015). Memory network first introduced by Weston et al. (2015) is a new class of learning models which can easily read and write to part of a long-term memory component, and combine this seamlessly with inference for prediction. Formally, besides the explicit memory which is an array of cells to memorize the pre-trained vector representations

[1]Note that the personal information used in our examples and datasets throughout this paper is all synthetic and not real.

[2]It is worth noticing that the term "Hierarchical Memory Networks" has been mentioned in (Chandar et al., 2016) where the intention was to organize the memory into multi-level groups based on hashing, tree or clustering structures to make the reader efficiently access the memory, whereas in our paper the term has a different meaning.

[3]Our code and dataset are available: `https://github.com/jacoxu/HMN4QA`

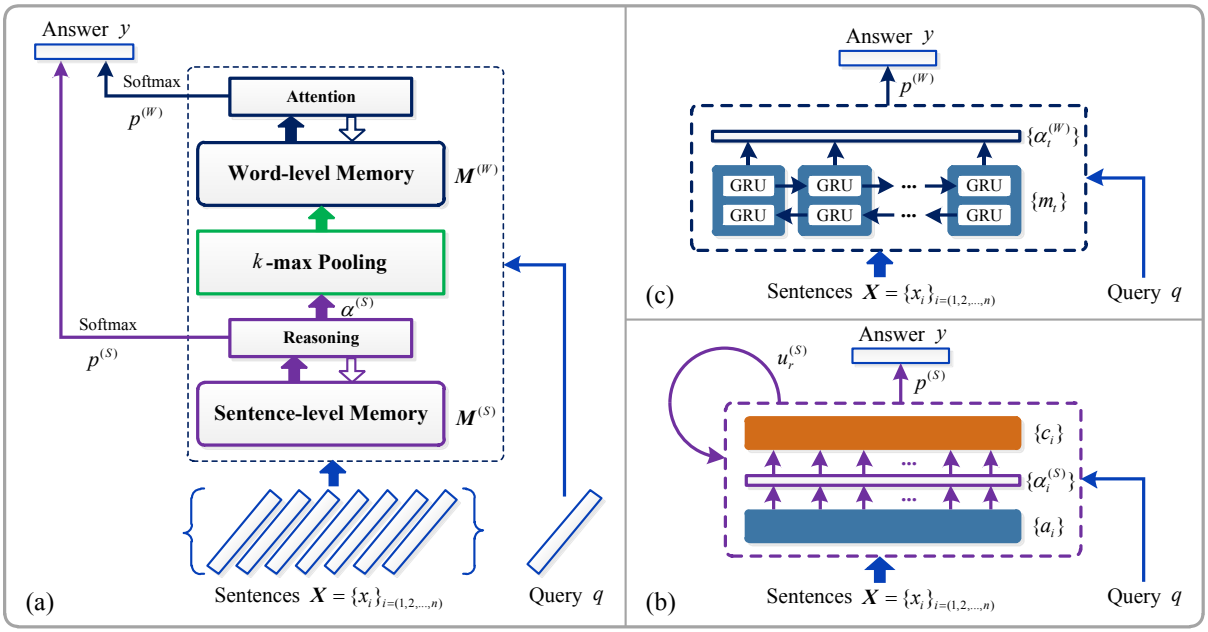


Figure 1: An illustration of our Hierarchical Memory Networks (HMN). (a): The overall architecture of the proposed HMN. (b): Reasoning module of our approach on sentence-level memory. (c): Attention module of our approach on word-level memory.

of the external data, a general memory network consists of four major components: (1). *Input feature map* which converts the incoming input to the internal feature representation. (2). *Generalization* which updates old memories given the new input. (3). *Output feature map* which produces a new output based on the new input and the current state. (4). *Response* which converts the output into the response format desired. Along the above framework, Sukhbaatar et al. (2015) put forward end-to-end memory networks which do not require the supervision of the supporting facts and are more generally applicable in realistic setting. Thus, we choose end-to-end memory networks, denoted as MemNN throughout our paper, as the foundation of our proposed approach.

3 Hierarchical Memory Networks for Answer Selection

3.1 Approach Overview

As described in Figure 1(a), we give an illustration of our HMN for answer selection. Given a set of n sentences denoted as: $\mathbf{X} = \{x_i\}_{i=(1,2,...,n)}$ and a query q, where i is the timestep of sentence x_i in the set. We first map these sentences $\mathbf{X}$ into the sentence-level memory $\mathbf{M}^{(S)}$ and the word-level memory $\mathbf{M}^{(W)}$ with low-dimensional distributed representations respectively. Then, reasoning on the sentence-level memory is utilized to soft-search the related sentences. We further exploit k-max pooling to sample the most relevant sentences based on the soft-searching results and take attention mechanism to focus on word-level memory of the selected sentences. The target answer y is used to guide the learning of the reasoning on sentence-level memory and the attention on word-level memory learning simultaneously.

3.2 Sentence-level Memory and Reasoning

In this section, we apply reasoning module to make multiple interaction on sentence-level memory based on the adjacent weight tying scheme of MemNN (Sukhbaatar et al., 2015), as shown in Figure 1(b). Given two word embedding matrices $\mathbf{A} \in \mathbb{R}^{|V| \times d}$ and $\mathbf{C} \in \mathbb{R}^{|V| \times d}$, where $|V|$ is the vocabulary size and d is the dimension of the word embedding, we first encode the word x_{ij} at timestep j in the sentence x_i into dual channels of word representation as $\mathbf{A}x_{ij} \in \mathbb{R}^d$ and $\mathbf{C}x_{ij} \in \mathbb{R}^d$.

In order to combine the order of the words into their representations, a positional encoding matrix $\mathbf{l}$ is applied to update the dual-channel word embeddings as $l_{gj} \cdot (\mathbf{A}x_{ij})$ and $l_{gj} \cdot (\mathbf{C}x_{ij})$, where

$$l_{gj} = (1 - j/J_i) - (g/d)(1 - 2j/J_i), \quad 1 \leq j \leq J_i,\ 1 \leq g \leq d, \tag{1}$$

and J_i is the length of the sentence x_i and g is the embedding index. This positional encoding scheme is also successfully applied in (Xiong et al., 2016).

Two temporal encoding matrices $\mathbf{T}_A \in \mathbb{R}^{n \times d}$ and $\mathbf{T}_C \in \mathbb{R}^{n \times d}$ are further utilized to encode the order of the sentences. Then, the sentence-level memory $\mathbf{M}^{(S)} = \{\{a_i\}, \{c_i\}\}_{i=(1,2,\ldots,n)}$ is reformed as:

$$a_i = \sum\nolimits_j l_j \cdot (\mathbf{A}x_{ij}) + \mathbf{T}_A(i), \quad c_i = \sum\nolimits_j l_j \cdot (\mathbf{C}x_{ij}) + \mathbf{T}_C(i), \tag{2}$$

where l_j is the j-th column vector of the position encoding matrix $\mathbf{l}$ according to the sentence x_i and the operation "$\cdot$" means the element-wise multiplication.

For the query q, the j-th word q_j is also embedded as $\mathbf{A}q_j \in \mathbb{R}^d$, where the $\mathbf{A}$ is the embedding matrix used in Eqn. (2). By encoding the word position j into the query representation, we get the probe representation of the query q as follows:

$$u_1^{(S)} = \sum\nolimits_j l_j \cdot (\mathbf{A}q_j), \tag{3}$$

where l_j is the j-th column vector of the position encoding matrix $\mathbf{l}$ according to the query q. Then the attention weights of the sentences according to the query can be calculated through the inner product of the two vectors as $\alpha_i^{(S)} = softmax(a_i^T u_1^{(S)})$, and the output of the sentence-level memory based on the activation of the query can be obtained as: $o_1 = \sum_i \alpha_i^{(S)} c_i$.

In order to perform reasoning on sentence-level memory to find the most relevant sentences, we make R hops to soft-search the sentences and output the final vector o_R. To be specific, during the $r+1$ hop of the reasoning operation, the process can be formalized as: $u_{r+1}^{(S)} = o_r + u_r^{(S)}$, $\alpha_i^{(S)} = softmax(a_i^T u_{r+1}^{(S)})$, $o_{r+1} = \sum_i \alpha_i^{(S)} c_i$, and the dual-channel memories are updated as follows:

$$a_i = \sum\nolimits_j l_j \cdot (\mathbf{A}^{r+1}x_{ij}) + \mathbf{T}_A^{r+1}(i), \quad c_i = \sum\nolimits_j l_j \cdot (\mathbf{C}^{r+1}x_{ij}) + \mathbf{T}_C^{r+1}(i), \tag{4}$$

where $1 \leq r \leq (R-1)$. Specifically, during the $r+1$ hop, the word embedding matrices $\mathbf{A}^{r+1}$ and $\mathbf{C}^{r+1}$ are mutually independent, so as the temporal encoding matrices $\mathbf{T}_A^{r+1}$ and $\mathbf{T}_C^{r+1}$. But during the adjacent two hops, $\mathbf{A}^{r+1} = \mathbf{C}^r$ and $\mathbf{T}_A^{r+1} = \mathbf{T}_C^r$. Finally, we can get the predicted word probability distribution by applying softmax on the output vector of the reasoning on the sentence-level memory as:

$$p^{(S)}(w) = softmax((\mathbf{C}^R)^T(o_R + u_R^{(S)})), \tag{5}$$

where $w = \{w_t\}_{t=(1,2,\ldots,|V|)}$ is the word set with a vocabulary size of $|V|$, the weight matrix is the same as the embedding matrix $\mathbf{C}^R \in \mathbb{R}^{|V| \times d}$ on the last hop, and T is the operation of matrix transposition.

3.3 k-max Pooling

Here, we exploit a pooling operation over the top attention weights $\alpha^{(S)}$ of the reasoning module on the sentence-level memory to sample the most relevant sentences. Given a value k and the top attention weights $\alpha^{(S)}$ of length $n \geq k$, we use the k-max pooling to select a subset of sentence sequences $\hat{\mathbf{X}} = \{\hat{x}_i\}_{i=(1,2,\ldots,k)}$, corresponds with their top-k maximum values of $\alpha^{(S)}$ on the sentences.

The k-max pooling operation makes it possible to pool the k most relevant sentences to the query and filter the noise information, which maybe more beneficial to select the correct answers. Moreover, this sampling module feeds a subset of sentences $\hat{\mathbf{X}}$ to the following attention mechanism on the word-level memory, which can reduce the computation complexity of the attention to focus on the relevant words.

3.4 Attention on Word-level Memory

For word-level memory, we first apply a Bi-directional GRU (BiGRU) to compute the hidden states of all the ordered words $\bar{w} = \{\bar{w}_t\}_{t=(1,2,\ldots,|t|)}$ in the sentence set $\mathbf{X}$, where $|t|$ is the time steps of the words in the sentences. In particular, for the t-th word $\bar{w}_t$, the forward GRU and the backward GRU encode

Domain (Lang)	Train/Dev/Test	Vocab	Unseen Answers (Dev/Test)	Max Len (P/S)
Air-Ticket (CH)	5,400/600/6,000	8,540	409 (68.2%)/4,020 (67.0%)	16/17
Hotel (CH)	5,400/600/6,000	7,586	367 (61.2%)/3,690 (61.5%)	16/16
Air-Ticket (EN)	5,400/600/6,000	7,537	342 (57.0%)/3,489 (58.2%)	16/18
Hotel (EN)	5,400/600/6,000	7,134	357 (59.5%)/3,452 (57.5%)	16/16
Total	21,600/2,400/24,000	29,092	1,406 (58.6%)/13,872 (57.8%)	16/18

Table 1: Statistics of the datasets, including the domain and the language of the dataset (CH: Chinese and EN: English), the number of train/dev/test set entries, the vocabulary size of datasets, the number and proportion of unseen answers on dev/test set, and the max paragraph length and sentence length of the datasets. The *Total* dataset consists all the samples of the above four datasets.

it as hidden states $\vec{h}_t = \vec{GRU}(\mathbf{C}^R \bar{w}_t)$ and $\overleftarrow{h}_t = \overleftarrow{GRU}(\mathbf{C}^R \bar{w}_t)$ respectively, where $\mathbf{C}^R$ is the word embedding matrix of the last hop on the sentence-level memory, and we set the dimension of $\vec{h}_t$ and $\overleftarrow{h}_t$ equals to the dimension of the word embedding. By summing the forward hidden states and the backward hidden states, we obtain the word-level memory as $\mathbf{M}^{(W)} = \{m_t\}_{t=(1,2,...,|t|)}$, where $m_t = \vec{h}_t + \overleftarrow{h}_t$. In this way, the memory m_t contains the context information of the t-th word $\bar{w}_t$ in the sentence set $\mathbf{X}$.

Then, we perform attention on the subset of the ordered words $\hat{w} = \{\hat{w}_t\}_{t=(1,2,...,|\hat{t}|)}$ in the selected sentences $\hat{\mathbf{X}}$ by using the probe vector $u_R^{(S)}$ of the last hop on the sentence-level memory and a subset of word-level memory $\{\hat{m}_t\}_{t=(1,2,...,|\hat{t}|)}$ selected form $\mathbf{M}^{(W)}$ according to the word subset $\hat{w}$. The normalized attention weights $\alpha^{(W)} = \{\alpha_t^{(W)}\}_{t=(1,2,...,|\hat{t}|)}$ on the word-level memory are calculated as:

$$\alpha_t^{(W)} = softmax(v^T tanh(\mathbf{W} u_R^{(S)} + \mathbf{U}\hat{m}_t)), \tag{6}$$

where $v \in \mathbb{R}^{d\times 1}$, $\mathbf{W} \in \mathbb{R}^{d\times d}$ and $\mathbf{U} \in \mathbb{R}^{d\times d}$ are all learning parameters updated during the training. Inspired by Pointer Networks (Vinyals et al., 2015), we adopt the normalized attention weights $\alpha^{(W)}$ on the word collection $\hat{w}$ as the probability distribution of the output words:

$$p^{(W)}(w) = trans(p^{(W)}(\hat{w})) = trans(\alpha^{(W)}), \tag{7}$$

where $trans(\cdot)$ means the operation to map the words probability distribution $p^{(W)}(\hat{w}) \in \mathbb{R}^{|\hat{t}|}$ into the probability distribution $p^{(W)}(w) \in \mathbb{R}^{|V|}$. To be specific, the map operation makes the probability distribution $p^{(W)}(\hat{w})$ of the word subset ($\hat{w} = \{\hat{w}_t\}_{t=(1,2,...,|\hat{t}|)}$) to be added into their corresponding positions in the vocabulary ($w = \{w_t\}_{t=(1,2,...,|V|)}$), and the probabilities of the words not in selected word subset $\hat{w}$ will be set to zero[4]. Finally, we get the new probability distribution $p^{(W)}(w) \in \mathbb{R}^{|V|}$.

3.5 Joint Learning

In this paper, we combine the probability distributions of the output words both on the sentence-level memory and the word-level memory to predict the joint probability distribution $p(w)$ as follows:

$$p(w) = p^{(S)}(w) + p^{(W)}(w). \tag{8}$$

Finally, we use the target answer y to guide the learning of the reasoning module on sentence-level memory and the attention module on word-level memory simultaneously. We choose the cross entropy as the cost function and apply Stochastic Gradient Descent (SGD) (Bottou, 1991) as the optimization method to train our joint model. The learned parameters include word embedding matrices $\mathbf{A}^1$ and $\{\mathbf{C}^r\}_{r=(1,2,...,R)}$, temporal encoding matrices $\mathbf{T}_A^1$ and $\{\mathbf{T}_C^r\}_{r=(1,2,...,R)}$ in Eqn. (2) and (4), the parameters $\{\theta_{BiGRU}\}$ of the BiGRU model and the attention parameters v, $\mathbf{W}$ and $\mathbf{U}$ in Eqn. (6).

[4]For example, if the word "airport", at two different timesteps in the ordered word subset $\hat{w}$, has two probabilities "0.1" and "0.3", the map operation would add up these probabilities and set "0.4" as the probability of the word "it" in the vocabulary.

	Air-Ticket (CH)	*Hotel (CH)*	*Air-Ticket (EN)*	*Hotel (EN)*	*Total*
MemNN-H1	4,727.8±79.2	3,680.8±48.3	4,051.8±53.8	3,067.4±71.3	14,492.8±282.8
MemNN-NT	3,424.0±106.6	2,816.2±35.9	2,984.2±93.0	2,304.0±70.3	10,250.4±152.1
MemNN	67.5±9.1	84.0±9.8	55.5±9.4	56.2±10.7	225.6±41.1
HMN-Sent	99.8±29.9	175.0±19.3	112.8±51.2	129.2±15.3	125.6±27.4
HMN-Word	22.2±7.5	31.8±7.8	24.8±9.6	12.8±4.9	27.4±4.8
HMN-Joint	**4.2±2.5**	**9.8±1.9**	**4.4±2.7**	**3.0±1.2**	**4.2±1.8**

Table 2: Comparison of predicted test error numbers of our HMN and MemNN with different components on four domain datasets (Test: 6,000 samples) and the *Total* dataset (Test: 24,000 samples). MemNN-H1: Memory network with 1 hop and temporal encoding, MemNN-NT: Memory network with 3 hops but without temporal encoding. MemNN: Memory network with 3 hops and temporal encoding. Note that HMN-Sent and HMN-Word, as the parts of HMN-Joint, are joint learning but give their predicted answers separately.

4 Experiments

4.1 Datasets and Setup

We conduct answer selection tasks on four synthetic domain dialogue datasets, two from air-ticket booking domain and two from hotel reservation domain. One complete dialogue history of each dataset has eight round responses. Besides greeting and ending sentences, one dialogue history consists six round responses to query and answer client's personal information, such as name, phone and passport number. The datasets contain hundreds of response patterns and thousands of entity information. More detailed descriptions can be found in our released datasets. The statistics of the datasets are summarized in Table 1. We use 45% of the data for training, 5% for validation and the remaining 50% for test. From the statistics, we can see that the proportions of the unseen answers on dev/test sets all overtake 57%.

In our experiments, the most of hyper parameters are set uniformly for the datasets as described in Table 3. The training gradients with an $l2$ norm larger than 40 are clipped to 40 and the learning rate is annealed every 15 epochs by $\lambda/2$ until 60 epochs are reached. The learned parameters are all initialized randomly from a Gaussian distribution with zero mean and 0.1 standard deviation. In order to make the comparison more intuitive, we use the number of predicted error samples on each dataset to evaluate the performance for answer selection and calculate the average result by repeating each experiment 5 times.

Hyperparameter	Hidden dim.	Hops	Max pooling	Learning rate	Batch size (*Total*)
Value	$d = 100$	$R = 3$	$k = 4$	$\lambda = 0.01$	30 (32)

Table 3: Hyperparameters used in our experiments. The dimension of word embeddings and the dimension of GRU hidden states are set equally to 100, batch sizes are 32 for *Total* and 30 for the others.

4.2 Comparison with Memory Networks

In order to evaluate the effect of multiple hops for reasoning module and temporal encoding on sentence-level memory, we design three MemNN (Sukhbaatar et al., 2015) based variants: MemNN-H1 (1 hop and temporal encoding), MemNN-NT (3 hops but not temporal encoding) and MemNN (3 hops and temporal encoding) on the datasets. We further evaluate the prediction performance of our HMN on different level memory components via HMN-Sent (prediction of reasoning module on sentence-level memory as Eqn. (5)), HMN-Word (prediction of attention module on word-level memory as Eqn. (7)) and HMN-Joint (joint prediction as Eqn. (8)). The comparison of these methods are reported in Table 2. From the results, we can see that MemNN-H1 without temporal encoding and MemNN-NT without multi-hop reasoning make the worst performances, which clearly demonstrate that multiple hops for reasoning module and temporal encoding on sentence-level memory play a very important role on our tasks. Despite that MemNN represents surprising results on this task, our HMN-Word and HMN-joint

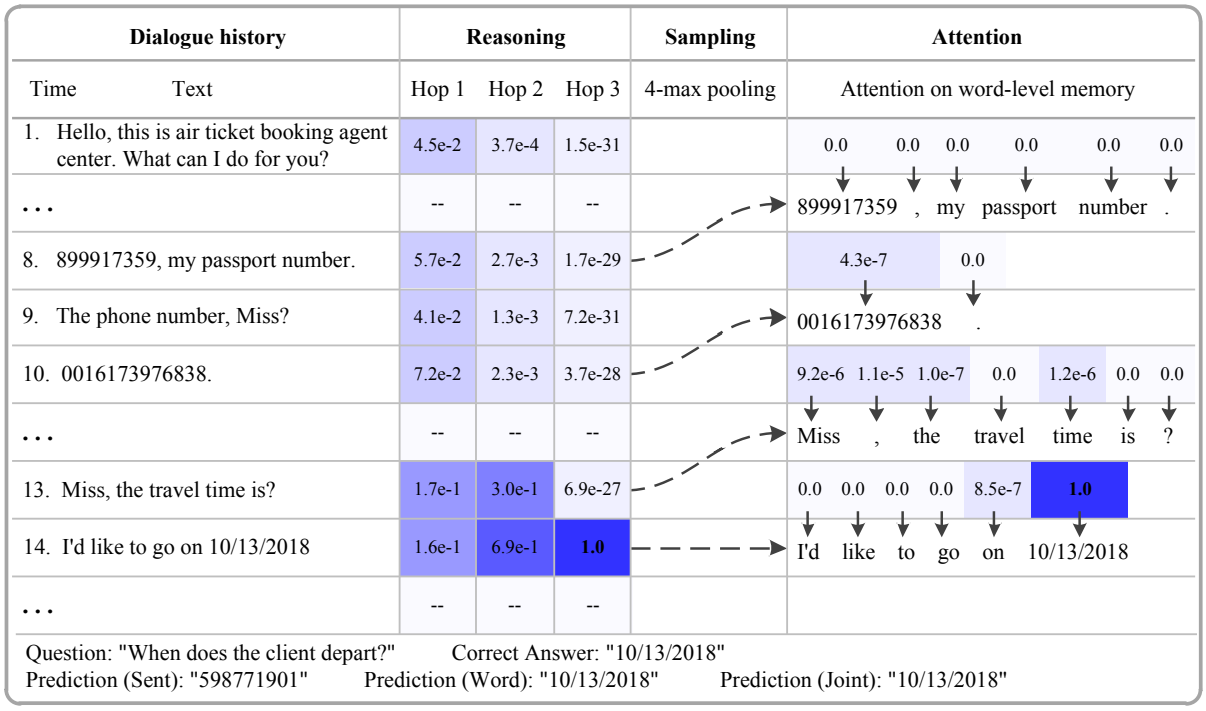


Figure 2: One example prediction by our HMN for answer selection. The hop columns in reasoning module show the probabilities of each hop, and the sampling module selects the 4-max relevant sentences and feeds these selected sentences to the attention module on word-level memory. In the experiments, all the sentences in the dialogue history are indexed in reverse order to reflect their relative distance.

further improve the answer selection performance on all the datasets. Compared with the results of HMN-Sent and HMN-Word, the results also show that the joint prediction can make a better performance.

4.3 How to Select the Correct Answers on Unknown Words

Here, we try to answer two questions: (1) How does the reasoning module focus on the related sentences and predict the rare and unknown words on sentence-level memory? (2) How does the attention module focus on the correct answers and distinguish multiple rare and unknown words on word-level memory? We give a visual example of our HMN over one dialogue history for answer selection in Figure 2 to get a better understanding of the effect of each module. The example is one dialogue history from air-ticket booking domain which contains 16 sentences associated with their temporal indexs.

From Figure 2, we can see that in the first hop, the reasoning module mainly focuses on the sentences 13 and 14, which have most semantic relevance to the question "When does the client depart?". As the effect of temporal encoding as Eqn. (4), the reasoning allocates more weight to the most related sentence 14 in the following hops. Another interesting result as shown in Table 2 is that MemNN and HMN-Sent represent surprising performance to predict the rare and unknown words on sentence-level memory. An explanation maybe that the way in which these methods use a simple way, rather than sophisticated LSTM or GRU, to encoding the sentence into memory as Eqn. (1). This simple sentence encoding strategy can remain the raw embedding representation of words, and MemNN and HMN-Sent utilize the transpose of the raw embedding matrix as the decoding weights to conduct answer match as Eqn. (5) in the raw embedding space. Nonetheless, sentence-level encoding may introduce other semantic information which may lead to predict an error answer, as the prediction (Sent) in the example.

After k-max pooling for sampling the most relevant sentences, a sophisticated attention mechanism is applied on the selected word-level memory. Two possible reasons make the attention successfully focus on the correct answers: One is that the probe vector $u_R^{(S)}$ as Eqn. (6), used in attention mechanism to interact with word-level memory, is generated from the reasoning module and has semantic relevance to the

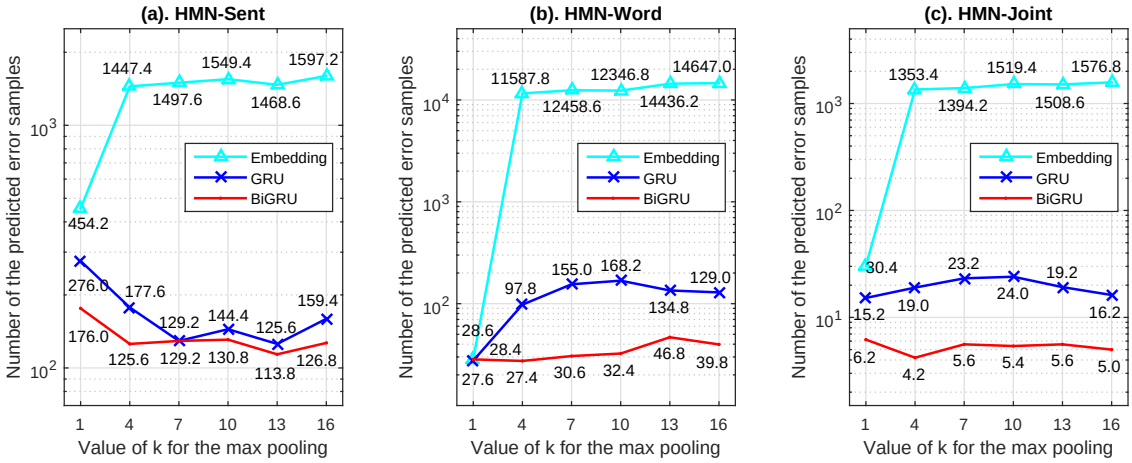


Figure 3: Answer selection evaluations of HMN-Sent, HMN-Word and HMN-Joint with different word-level memory encoding methods (BiGRU, GRU and Embedding) using various values of k on *Total*.

target answers. Another reason is that attention mechanism performs inhibitory effect on our task which can successfully filter the almost useless words, such as "time", "go" and "on" in the example. Besides the above reasons, we also investigate the influence of different word-level memory encoding methods, such as BiGRU ($\overrightarrow{GRU}(\mathbf{C}^R\bar{w}_t)+\overleftarrow{GRU}(\mathbf{C}^R\bar{w}_t)$), GRU ($GRU(\mathbf{C}^R\bar{w}_t)$) and Embedding ($\mathbf{C}^R\bar{w}_t$), by varying the values of k for max pooling, and the comparison of answer selection performance are present in Figure 3. The results show the expected effect that encoding the context information into word-level memory via BiGRU or GRU can help the attention module distinguish multiple rare and unknown words when we enlarge the value of k and introduce more unknown words to the attention mechanism.

From Figure 3, we further investigate the influence of k to the answer selection performance. We can see that the performances almost unchanged by using HMN-Joint with BiGRU when we vary the value of k. Considering that the more sentences k-max pooling samples from sentence-level memory, the more computational complexity the attention mechanism as Eqn. (6) costs on word-level memory, 4-max pooling used in our experiments is a good trade-off.

5 Related Works

Recently, lots of deep learning methods with explicit memory and attention mechanism have shown promising performance in Question Answering (QA) tasks. For example, Yu et al. (2015) applied Neural Machine Translation (NMT) (Bahdanau et al., 2015) with sophisticated attention mechanism and Neural Turing Machine (NTM) (Graves et al., 2014) with distributed external memory to solve QA tasks, and Sukhbaatar et al. (2015) designed end-to-end memory networks and introduced multi-hop reasoning component to solve various types of QA task. These representation learning based methods do not rely on any linguistic tools and can be applied to different languages or domains (Feng et al., 2015). However, most works of these deep learning based methods rarely focus on solving answer selection on unknown word problem. Recently, the unknown word problem has attracted more researchers' attention. Hermann et al. (2015) used NLP tools to recognize all the entity and establish co-references to replace all the rare entities by placeholders and trained an attention based model with softmax to predict the placeholder id. Li et al. (2016) replaced the rare words in a test sentence with similarity in-vocabulary words to solve machine translation task, where the representation of the rare words still can be learned from a large mono-lingual corpus. Gulcehre et al. (2016) utilized and extended the attention-based pointing mechanism (Vinyals et al., 2015) to point the unknown words for machine translation and text summarization. However, the sophisticated attention mechanism is applied on the all word-level representations which may result in high computational complexity, and lots of the fine-grained noise words should be filtered out by reasoning the relevant facts to the query in a high-level semantic space.

6 Conclusion

In this paper, we introduce hierarchical memory networks to solve answer selection problem on unknown words. We first encode the sentences into a sentence-level memory with temporal encoding. Then reasoning module conducts multi-hop interaction on the memory to retrieve the related sentences, and k-max pooling samples the k most related sentences. For word-level memory, BiGRU is utilized to encode the words and introduce context into the memory, then a sophisticated attention mechanism is applied on the selected word-level memory to focus the fine-grained words. We conduct answer selection experiments on four synthetic domain dialogue datasets which contain lots of unseen answers. The experimental results show that our hierarchical memory networks can achieve a satisfying performance.

Acknowledgements

We thank the anonymous reviewers for their insightful comments, and this work was supported by the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDB02070005), the National High Technology Research and Development Program of China (863 Program) (Grant No. 2015AA015402) and the National Natural Science Foundation (Grant No. 61602479 and 61403385).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Léon Bottou. 1991. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8).

Sarath Chandar, Sungjin Ahn, Hugo Larochelle, Pascal Vincent, Gerald Tesauro, and Yoshua Bengio. 2016. Hierarchical memory networks. *arXiv preprint arXiv:1605.07427*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)*, 12(Aug):2493–2537.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 813–820. IEEE.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1693–1701.

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

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. Towards zero unknown in neural machine translation. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2852–2858. AAAI Press.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 899–907.

Mónica Marrero, Julián Urbano, Sonia Sánchez-Cuadrado, Jorge Morato, and Juan Miguel Gómez-Berbís. 2013. Named entity recognition: fallacies, challenges and opportunities. *Computer Standards & Interfaces*, 35(5):482–489.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1577–1586.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2440–2448.

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

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2692–2700.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Yang Yu, Wei Zhang, Chung-Wei Hang, and Bowen Zhou. 2015. Empirical study on deep learning models for question answering. *arXiv preprint arXiv:1510.07526*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, pages 2335–2344.

Revisiting Taxonomy Induction over Wikipedia

Amit Gupta
EPFL
Lausanne, Switzerland
amit.gupta@epfl.ch

Francesco Piccinno
University of Pisa
Pisa, Italy
piccinno@di.unipi.it

Mikhail Kozhevnikov
Google Inc.
Zurich, Switzerland
qnan@google.com

Marius Paşca
Google Inc.
Mountain View, California
mars@google.com

Daniele Pighin
Google Inc.
Zurich, Switzerland
biondo@google.com

Abstract

Guided by multiple heuristics, a unified taxonomy of entities and categories is distilled from the Wikipedia category network. A comprehensive evaluation, based on the analysis of upward generalization paths, demonstrates that the taxonomy supports generalizations which are more than twice as accurate as the state of the art. The taxonomy is available at http://headstaxonomy.com.

1 Introduction

Motivation. As possibly the largest resource of publicly available, semi-structured knowledge, Wikipedia (Remy, 2002) serves as a stepping stone towards the construction of collections of structured data (Remy, 2002; Hoffart et al., 2013; Vrandečić and Krötzsch, 2014). Data within Wikipedia benefits from new additions and distributed curation by human editors, and has proven beneficial in text analysis tasks ranging from co-reference resolution (Ratinov and Roth, 2012), word sense (Mihalcea, 2007) and entity disambiguation (Ratinov et al., 2011), to information retrieval (Hu et al., 2009) and information extraction (Wu and Weld, 2010; Nastase and Strube, 2013; Hoffart et al., 2013; Dong et al., 2014).

Wikipedia links millions of entities (e.g. *Barack Obama*) to thousands of inter-connected categories of different granularity (e.g. *Presidents of the United States*, *Political office-holders, Politicians*) to form what is often referred to as the Wikipedia category network (WCN). However, obtaining a taxonomy of increasingly general categories from WCN is by no means trivial because upward edges in WCN, from entities to categories and also from child to parent categories, are not confined to *is-a* relations (Ponzetto and Strube, 2007). In fact, consistently discarding *not-is-a* edges such as *Japan⇝660 BC* or *Award winners⇝Awards*, while retaining as many true *is-a* edges as possible, has been the object of a steady body of research (Ponzetto and Strube, 2007; Hovy et al., 2013; Flati et al., 2014). Recent methods still produce taxonomies with glaring gaps in precision and coverage. More importantly, even if the methods correctly identify individual *is-a* edges with an accuracy as high as 85% (Flati et al., 2014), it is not uncommon for upward paths to traverse at least some incorrect edges. The resulting taxonomies transitively connect entities (e.g., *Natural language processing*) to many ancestor categories (e.g., *Physical body*, *Mass*)[1] that are not true generalizations, thus limiting their utility in practice.

Contributions. This paper proposes a novel method for taxonomy induction from WCN. As described in Section 3, the method exploits syntactic evidence in category titles to connect entities (i.e., pages) with increasingly more general categories. A novel, comprehensive framework for taxonomy evaluation is proposed, focusing on the accuracy and granularity of longer generalization paths, as opposed to individual edges. Section 4 describes the evaluation framework and carries out an in-depth comparison of the proposed taxonomy against the state of the art. It shows significant gains in accuracy relative to current state of the art, while maintaining similar coverage.

[1] Examples taken from http://wibitaxonomy.org .

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2300–2309, Osaka, Japan, December 11-17 2016.

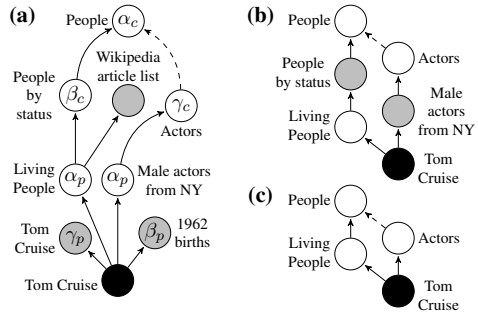


Figure 1: Taxonomy induction phases. Black circles denote entities. White circles denote categories. Dashed lines denote paths including possibly multiple edges. **(a)** Step 1: Page heuristics (α_p, β_p and γ_p) and category heuristics (α_c, β_c and γ_c) are applied sequentially to select candidate generalizations for each node (page or category), until one produces at least one candidate (white circles). Gray nodes show candidates that would have been produced by remaining heuristics. **(b)** Step 2: Nodes that encode redundant information are removed (grey). **(c)** Resulting taxonomy.

2 Related work

Thanks to continuous contributions and curation by many human editors, Wikipedia (Remy, 2002) recommends itself as a high quality resource of semi-structured knowledge. It enables multiple approaches to large scale knowledge acquisition and taxonomy induction (Hovy et al., 2013). One of the earliest attempts towards the latter is WikiTaxonomy (Ponzetto and Strube, 2007; Ponzetto and Strube, 2011). In WikiTaxonomy, relations are labeled as either *is-a* or *not-is-a*, using a cascade of heuristics based on the syntactic structure of category labels, the topology of the network and lexico-syntactic patterns for detecting subsumption and meronymy, similar to Hearst patterns (Hearst, 1992). WikiNet (Nastase et al., 2010) extends WikiTaxonomy by expanding *not-is-a* relations into fine-grained relations such as *part-of, located-in, etc*. YAGO, induces a taxonomy by employing heuristics linking Wikipedia categories to corresponding synsets in WordNet (Hoffart et al., 2013). YAGO's taxonomy forms the backbone of a variety of intelligent applications, including Watson (Ferrucci et al., 2010). DBPedia (Lehmann et al., 2015) aims to provide a fully-structured representation of semi-structured content of Wikipedia. It focuses on linking the extracted knowledge with existing resources such as YAGO, OpenCyc etc.

The Wikipedia Bitaxonomy project, or WIBI (Flati et al., 2014), the most recent effort towards large-scale taxonomy induction from Wikipedia, simultaneously induces a taxonomy for pages and a separate taxonomy for categories from WCN using the idea that information contained in pages can be useful in constructing a taxonomy of categories and vice-versa. First, an initial taxonomy over pages is constructed by extracting lemmas from their first sentences and resolving them to other pages in Wikipedia. Alternating between the two taxonomies, edges are added to each taxonomy based on the information available in the other. Finally, heuristics further enrich the category taxonomy by adding hypernym edges for nodes which are still orphans after the first two steps. In contrast to both WikiTaxonomy and our work, WIBI ignores the syntactic structure of category titles.

3 Taxonomy induction

A unified, high-accuracy taxonomy of pages and categories is induced from WCN through the application of a cascade of linguistically motivated heuristics, which exploit lexical and structural information (mainly the lexical head of categories) from Wikipedia to generate a set of candidate generalizations for pages (*page heuristics*) and categories (*category heuristics*). Subsequently, other heuristics are used to simplify the taxonomy by eliminating redundant nodes (see Figure 1). The heuristics, which are derived empirically or adapted from previous work, are described in this section using these notations:

- E: set of all WCN edges;
- h_c: lexical head of the title of category c;
- $C_a(n)$: set of *all* direct parents of node n (page or category) in WCN, $\{c \mid (n,c) \in E\}$[2];
- $C_{pl}(n) \subset C_a(n)$: subset of parent categories ($C_a(n)$) whose titles have a *plural* lexical head, such as *Administrative divisions*. Categories with plural heads have played an important role in earlier work on taxonomy induction from Wikipedia, as they are more likely to be genuine classes (e.g. *Countries*) as opposed to instances (e.g. *France*) (Suchanek et al., 2007; de Melo and Weikum, 2010);

[2]Wikipedia maintenance categories (e.g., *Sports award stubs*) are removed using a handful of blacklisted keywords such as "articles", "stubs" etc.

• L_p: set of *defining* lemmas attached to the root copular verb in the first sentence of the Wikipedia description of page p, e.g., *"William Shakespeare was an English poet, ..."* (Flati et al., 2014);

• $sup(h_{c_1}, h_{c_2})$: *global support* for an ordered pair of lexical heads h_{c_1} and h_{c_2}, defined as the number of edges in E, from a category with head h_{c_1} to a category with head h_{c_2};

• $\overrightarrow{v_h}$: vector of co-occurrence counts of plural head h with every unique plural head h' in WCN, where co-occurrence count is defined as the number of pairs of categories with heads h and h' which have at least one common child (page or category);

• $tsim\,(h_1, h_2)$: type similarity, defined as the cosine similarity between $\overrightarrow{v_{h_1}}$ and $\overrightarrow{v_{h_2}}$;

3.1 Category heuristics

Same head. Similarly to the head-matching heuristic in Ponzetto and Strube (2007), for any category c, pick all categories $c' \in C_a(c)$ as candidate generalizations, if they have the same lexical head as c. E.g. *Category:American actors* is picked as candidate generalization for *Category:American child actors*.

Global head support. For any category c, pick the category $c' \in C_{pl}(c)$ with the highest[3] global support $sup(h_c, h_{c'})$ as a candidate generalization, provided the support is above a fixed threshold T_{sup}. E.g. *Category:American entertainers* is picked as candidate generalization for *Category:American actors* because $sup(\textit{actors}, \textit{entertainers}) > T_{sup}$.

Type similarity. For any category c, pick the category $c' \in C_{pl}(c)$ which has head h' with the highest[3] type similarity $tsim(h, h')$ as a candidate generalization, if the similarity is above a fixed threshold T_{tsim}. E.g. *Category:People by occupation* is picked as candidate generalization for *Category:Entertainers* because $tsim(\textit{entertainers}, \textit{people}) > T_{tsim}$.

Only plural parent. For any category c, if $C_{pl}(c)$ contains only one category, pick it as a candidate generalization.

Only singular parent. For any category c with a non-plural head h_c, if $C_a(c)$ contains only one category, pick it as a candidate generalization.

Grouping child category. Categories whose titles match the pattern **X by Y** (e.g. *"Actors by nationality"*) usually indicate groupings of instances of *class* X by *attribute* Y (Nastase and Strube, 2008). Thus, for category c whose title matches the pattern **X by Y**, pick the category with title **X** (if one exists) as a candidate generalization.

Grouping parent category. For any category c, pick those categories in $C_{pl}(c)$ as candidate generalizations, whose titles match the pattern **X by Y**. E.g. *Category:Occupations by type* is picked as candidate generalization for *Category:Legal professions*.

Suffix head. For any category c, pick all categories $c' \in C_{pl}(c)$, whose lexical heads $h_{c'}$ are suffixes of h_c, as candidate generalizations. E.g. *Category:People by occupation* is picked as candidate generalization for *Category:Sportspeople*.

Lookahead candidates. For any category c, pick its grandparents (second-level ancestor categories) as candidate generalizations, if they satisfy the conditions in the SAME HEAD, GROUPING PARENT CATEGORY or SUFFIX HEAD heuristics. Higher-level ancestors are ignored as they are usually inaccurate.

Title head. For any category c, pick the category with the title h_c as a candidate generalization, if the lemma of h_c is in top T_l% most frequent lemmas among the defining lemmas L_p of the child pages of c. E.g. *Category:Writers* is picked as candidate generalization for *Category:Legal Writers*[4].

[3] If multiple categories satisfy the condition, all of them are picked.

[4] Key difference between **Same head** and **Title head** heuristic is that the latter does not require the candidate generalizations to be present in $C_a(c)$.

3.2 Page heuristics

Exact defining lemma. For page p, pick the category $c \in C_{pl}(p)$ as a candidate generalization if the lemma of its lexical head is in L_p. E.g. all parent categories of page *Johnny Depp* with lexical head *actors* are picked as candidate generalizations because *actor* is present in $L_{\text{Johnny Depp}}$.

Type-similar lemma. For page p, pick a category $c \in C_{pl}(p)$ as a candidate generalization, if the type similarity between the category's lexical head (h_c) and at least one of the defining lemmas in L_p is greater than a fixed threshold T'_{tsim}. E.g. all parent categories of page *Johnny Depp* with lexical head *people* are picked as candidate generalizations because *actor* is present in $L_{\text{Johnny Depp}}$ and $tsim\,(actors, people) > T'_{tsim}$.

Plural head. Similar to YAGO (Suchanek et al., 2007), for page p, pick all categories in $C_{pl}(p)$ as candidate generalizations.

Transfer. If a page p has an equivalent category[5], pick candidate generalizations generated by category heuristics for the equivalent category as candidate generalizations of p.

3.3 Construction of the HEADS taxonomy

The heuristics[6] are applied to individual pages or categories in order of decreasing edge-level precision, as measured on a manually annotated development set, which is the same order in which they have been presented above. For each node, the process stops when one of the heuristics produces at least one generalization, and the remaining heuristics for that node are ignored. For example, in Figure 1a, only the generalizations proposed by α_p for entity *Tom Cruise* are retained, namely *Living people* and *Male actors from NY*. Certain categories encode information that is orthogonal to types, and therefore superfluous as it may refer to time (*20th-century actors*), location (*Actors from Singapore*) or grouping by attributes (*Actors by nationality*). Such categories are detected using a few regular expressions and eliminated: their children are linked directly to their parents, and the redundant nodes are removed (Fig. 1b) producing a more compact taxonomy (Fig. 1c). This step is hereafter referred to as *simplification*.

The described process results in the HEADS taxonomy, which is evaluated in the next section. Taxonomy generation and evaluation in this submission is restricted to English Wikipedia. However, it can be easily adapted to other languages by porting the heuristics, a fairly straightforward task if a dependency parser is available in the target language. Adaptation to other languages is not explored in this study, and remains the object of future work.

4 Taxonomy evaluation

This section compares the HEADS taxonomy against the state of the art. It presents the standard edge-level evaluation (Ponzetto and Strube, 2011; Flati et al., 2014); demonstrates that, as popular as they might be, edge-level metrics do not reflect the real quality of a taxonomy; and proposes a more comprehensive evaluation, which takes into account the correctness of multi-edge generalization paths, overall probability of generalization errors, granularity of individual generalizations and accuracy of specializations. It is shown that performance along these newly-proposed dimensions is not necessarily correlated with edge-level metrics and cannot be estimated directly from them.

Experimental setup HEADS is constructed using a November 2015 snapshot of the English Wikipedia. To create a baseline for comparison, we initially attempted to re-implement the state-of-the-art taxonomy induction approach of Flati et al. (2014), but were unable to replicate the reported results. In particular, recall of the re-implementation was lower than expected. Since the source code for WIBI was not made public and was not available upon request, we instead compared HEADS directly against the entity and

[5] A page and category are considered equivalent if they have the same title after lemmatization of each token. If a disambiguation string is specified in the title (e.g., *biology* in *Family (biology)*), it should also match. e.g., *Families (biology)* ~ *Family (biology)* ≁ FAMILY.

[6] Threshold T_{sup} is set to 5, T_{tsim} and T'_{tsim} are set to 0.2 and T_l is set to 10.

Taxonomy	WIBI$_E$	WIBI$_C$	HEADS
Nodes	3,414,512	597,179	4,580,662
Entities (E)	3,414,512	-	4,239,486
Categories (C)	-	597,179	341,176
Leaves	3,308,755	465,682	4,359,178
Edges	3,859,717	594,917	11,648,975
$E \rightarrow E$	3,859,717	-	-
$E \rightarrow C$	-	-	11,077,992
$C \rightarrow C$	-	594,917	570,983
Avg. degree	1.13	0.996	2.54
WCCs	6,448	2,301	3,195
	Largest WCC		
Nodes	3,386,995 (99.2%)	469,453 (78.6%)	4,563,949 (99.6%)
Edges	3,838,286 (99.4%)	469,453 (78.9%)	11,634,161 (99.9%)

Table 1: Topological properties of HEADS and WIBI taxonomies. (WCC: weakly connected component)

Taxonomy	Edge type	P	R	C	A
WCN	$E \rightarrow C$	0.785	1.000	1.000	0.902
	$C \rightarrow C$	0.807	1.000	0.970	0.840
HEADS	$E \rightarrow C$	0.394	0.249	0.898	0.956
	$C \rightarrow C$	0.405	0.344	0.249	0.931
WIBI$_E$	$E \rightarrow E$	0.841†	0.794†	0.926†	0.789
WIBI$_C$	$C \rightarrow C$	0.852†	0.829†	0.973†	0.840

Table 2: Edge-level evaluation. $E \rightarrow C$ represents entity→category edges, $E \rightarrow E$ represents entity→entity edges and $C \rightarrow C$ represents category→category edges. †: results as reported in Flati et al. (2014). P: precision, R: recall, C: coverage, A: accuracy.

category taxonomies made available by Flati et al. (2014), referred to as WIBI$_E$ and WIBI$_C$, respectively. It is important to stress that WIBI taxonomies are generated using an older Wikipedia snapshot (October 2012). However, to the best of our knowledge, there is no evidence suggesting that taxonomy induction is easier or harder on more recent vs. older snapshots. Noisy edges between categories such as *Japan↝660 BC* can be found in both snapshots. Meanwhile, the network has grown significantly, with more than twice as many categories (1.37M vs. 619K) and 20% more entities (4.7M vs 3.8M), possibly adding to the complexity of the task.

4.1 Topological properties

The main topological properties of the HEADS and WIBI taxonomies are shown in Table 1. HEADS contains fewer categories and category→category edges than WIBI$_C$, due to the simplification step (cf. Section 3.3), which removes approximately 53% of parent categories from WCN. HEADS covers a larger number of entities than WIBI taxonomies, but a direct comparison of absolute sizes is not necessarily meaningful since the three taxonomies are defined in different spaces (WIBI$_E$ has entity→entity edges, WIBI$_C$ has category→category edges, while HEADS has entity→category and category→category edges). In addition, as already mentioned, WIBI taxonomies are generated using an older snapshot of Wikipedia. As shown in Table 1, the largest weakly connected component in HEADS and WIBI$_E$ covers over 99% of the nodes. HEADS has 50% fewer components, which is desirable, as each component is an enclave of isolated entities. WIBI$_C$, which is an order of magnitude smaller than WIBI$_E$ and HEADS, has even fewer connected components, but is overall less connected, with the largest connected component containing only 78% of the nodes. Lastly, HEADS contains about twice as many edges per node as the WIBI taxonomies (see avg. degree), which allows it to better account for multiple aspects of a concept or an entity, e.g., *Johnny Depp* being both an *Actor* and a *Film producer*.

4.2 Edge-level evaluation

The first comparison between HEADS and WIBI taxonomies follows the methodology introduced and consistently followed in prior literature, namely computing edge-level precision and recall scores against a gold standard (Ponzetto and Strube, 2011; Flati et al., 2014). To build the gold standard, 500 entities and 500 categories are randomly selected, and their parents in WCN are annotated by three human judges as correct or incorrect generalizations. [7] Table 2 shows *precision* and *recall* scores for HEADS and WIBI taxonomies by edge type. Precision and recall with respect to the golden edges are computed for each sampled node, and then averaged over all the nodes in the gold standard.

Compared to the WIBI taxonomies, HEADS shows significantly lower precision and recall scores in this evaluation. However, the losses can be largely attributed to the simplification procedure (cf. Section 3.3). For example, in Figure 1, the edge *Tom Cruise→Male actors from NY* would be missing from the final HEADS taxonomy as the node *Male actors from NY* would be removed by the simplification

[7] The inter-annotator agreement in terms of Fleiss' Kappa is 0.52. Annotations were harmonized by majority voting.

procedure, thus resulting in loss of precision and recall. Similarly, *Living People→People*, a correct edge, would be considered a precision loss, as it is absent from WCN (and hence, from the gold standard).

Table 2 also reports *coverage*, defined as the fraction of entities and categories in a taxonomy with at least one generalization, independent of its correctness. HEADS shows lower coverage on categories, because 65% of categories in WCN are removed from HEADS due to the simplification procedure.

As an additional metric, Table 2 reports edge-level *accuracy*, defined as the ratio of edges annotated as correct over the total number of edges sampled from a taxonomy. Accuracy scores are computed for each taxonomy by randomly sampling 450 edges of each type and annotating their correctness. HEADS is more accurate than WIBI$_E$ for entities, though a direct comparison is not meaningful, as WIBI$_E$ contains entity→entity edges while HEADS contains entity→category edges. For category→category edges, HEADS achieves a fairly significant > 10% improvement in accuracy compared to WIBI$_C$ taxonomy.

4.3 Beyond edge-level evaluation

Good performance at edge level, though widely used as an indicator of quality for a taxonomy (Ponzetto and Strube, 2007; Nastase and Strube, 2008; Flati et al., 2014), does not automatically translate into good performance at path level. For example, the generalization path *apples→fruits⇝vegetarians→people→organisms* is 75% edge-accurate (i.e., 3/4 edges are correct as indicated by the symbol →), but it can lead to the wrong inference that *apples* are *vegetarians* and, in turn, *people* and *organisms*. A single incorrect edge, namely *fruits⇝vegetarians*, causes a cascade of generalization errors for *fruits* and all its descendants, and a cascade of specialization errors for *vegetarians* and all its ancestors.

As an alternative to edge-level evaluation, the remainder of this section proposes a more structured scheme for evaluating a taxonomy. More specifically, it seeks to estimate the following: (1) What is the accuracy of multi-edge generalization paths ? (2) Are individual generalizations at the right level of granularity ? and (3) What is the accuracy of specializations of a concept.

4.3.1 Path-level evaluation

From the above example (*apples⇝organisms*), it is clear that during traversal of an upward generalization path, the correctness of individual edges is inconsequential to finding a good generalization for starting node (i.e., *apples*) once the first wrong edge (*fruits⇝vegetarians*) is encountered. Therefore, a good taxonomy should not only provide a large proportion of correct edges, but also provide correct generalization paths, i.e., paths which are correct in their entirety. However, since in practice it is common for relatively deep taxonomies to provide long generalization paths which pick at least one wrong edge, it would be still desirable to have a long *correct path prefix*, i.e., the maximal prefix of a path which is correct in its entirety.

This section evaluates HEADS and WIBI taxonomies on their ability to provide longer correct path prefixes and correct generalization paths. To avoid bias, it is desirable that paths sampled from different taxonomies start from the same node. Therefore, WIBI$_C$, which lacks the notion of entities, is first augmented with $E{\rightarrow}C$ edges from HEADS, resulting in a new hybrid taxonomy hereafter referred to as WIBI$_C$+H$_E$. For a sample of 250 entities present in HEADS, WIBI$_E$ and WIBI$_C$+H$_E$, one upward path is sampled per entity per taxonomy, for a total of 750 paths. Example paths are shown in Table 3, while Figure 2 shows the length distribution of the generalization paths sampled from each taxonomy. As expected, HEADS paths are generally shorter than WIBI taxonomies due to simplification.

To compare the three taxonomies, three human annotators[8] inspect each path starting from the entity and annotate the first incorrect generalization (e.g., *Film producer⇝Filmmaking* for the WIBI$_E$ example in Table 3). Figure 3 shows the average length of the correct path prefix in HEADS and WIBI taxonomies, along with 95% confidence interval bars[9], depending on the total length of a path. For a correct generalization path, the length of correct path prefix is the same as the path length, so an ideal taxonomy

[8] At least two annotators agreed for 93% of paths. All three annotators agreed for 53% of paths. Annotations are harmonized using majority voting.

[9] The confidence intervals reflect the distribution of the paths being sampled. A larger confidence bar indicates lower probability that a path of that length is chosen.

WIBI$_E$	WIBI$_C$+H$_E$	HEADS
Structure	Government	
↑Algebraic structure	... 23 more categories ...	**Apes**
↑Category (mathematics)	↑Cinema by region	**↑Humans**
↑Sequence	↑Cinema by continent	**↑People**
↑Process (science)	↑North American cinema	**↑Producers**
↑Filmmaking	↑Cinema of the United States	**↑American producers**
↑Film producer	**↑American film producers**	**↑American film producers**
Johnny Depp	**Johnny Depp**	**Johnny Depp**

Table 3: Upward generalization paths for *Johnny Depp* in three taxonomies. Correct path prefixes are shown in bold.

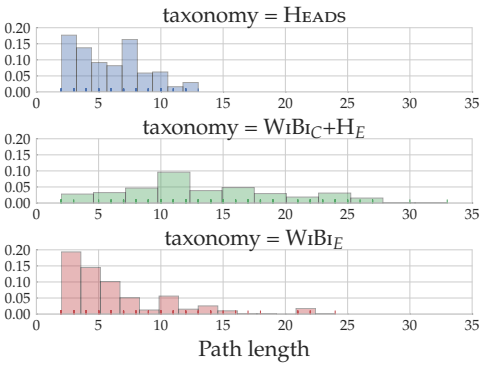


Figure 2: Length distribution of sampled generalization paths in different taxonomies.

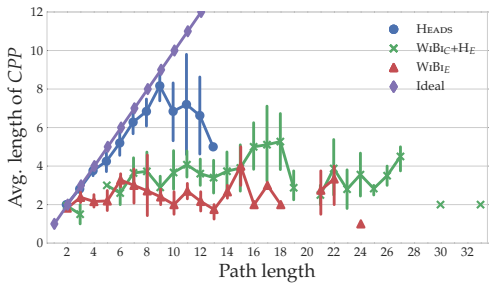


Figure 3: Average length of correct path prefix (*CPP*) in different taxonomies (computed using 750 annotated paths).

with only correct generalization paths would show up as the line $y = x$ in Figure 3. The behavior of HEADS is very close to an ideal taxonomy for the majority of path lengths, and outperforms WIBI$_E$ or WIBI$_C$+H$_E$ at all lengths, while WIBI$_C$+H$_E$ slightly outperforms WIBI$_E$. It is interesting to note that this difference does not translate into similar differences in edge-level evaluation, where all taxonomies consistently show relatively high accuracy (cf. Section 4.2). The superior performance of HEADS is further confirmed by Figure 4, which shows the probability of obtaining a correct generalization path of length $\leq k$. In contrast with WIBI taxonomies, HEADS generalization paths maintain high probability of correctness (> 0.7) at all lengths.

4.3.2 Path-granularity evaluation

A good taxonomy should not only provide correct generalization paths, but also ensure that each individual edge in the path provides generalization at the right level of granularity, i.e., neither too specific nor too general. To evaluate this aspect, 100 generalization paths originating from the same starting entities are sampled from different taxonomies. For each path, each individual edge is annotated by three human annotators with one of the following labels: 0 for wrong generalization (*fruits⇝vegetarians*); 1 for under-generalization (*fruits by country→fruits*); 2 for good-generalization (*edible fruits→fruits*); 3 for over-generalization (*edible fruits→physical bodies*). An edge under-generalizes if it adds or removes little information relative to the source node (e.g. *cricketers by team→cricketers*) or if it is a synonym or rephrasing of the original category (e.g. *coaches by sport→sport coaches*). An edge over-generalizes if it removes too much information. For example, for *bitstream→concept* one would expect the taxonomy to provide additional intermediate nodes (e.g. *binary sequences*) before generic node *concept*. Good-generalization label implies that edge is correct and neither over-generalizes nor under-generalizes. In order to ensure that the paths on which the comparison is performed are similar in length and complexity, we only consider pairs of shortest paths $\langle p_1, p_2 \rangle$ with the same final node, selected so as to minimize the difference in the length of the shortest paths $||p_1| - |p_2||$ in the two taxonomies, while ensuring that the paths are not identical ($p_1 \neq p_2$).

WIBI$_E$ is excluded from this experiment, since in contrast to HEADS and WIBI$_C$+H$_E$, WIBI$_E$ does not contain categories, hence the condition of same final node cannot be satisfied. Figure 5 graphically summarizes the results of this experiment. HEADS has fewer under-generalizations than WIBI$_C$+H$_E$

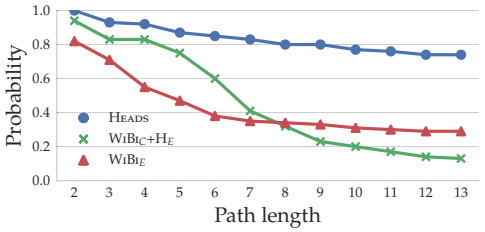


Figure 4: Probability of correct generalization paths vs. length (computed using 750 annotated paths). The probability at length k is the ratio of correct paths of length $\leq k$ to the total number of paths of length $\leq k$. Paths with length > 13 are omitted, as they are not present in HEADS samples, and always incorrect in WIBI$_E$ and WIBI$_C$+H$_E$ samples.

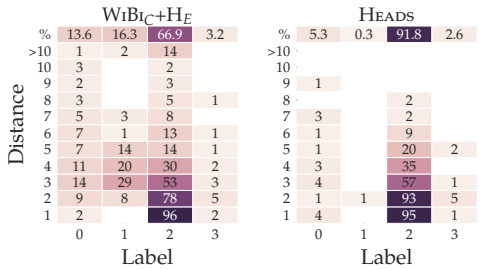


Figure 5: Generalization granularity evaluation for HEADS and WIBI$_C$+H$_E$ using 100 generalization paths. Labels on the horizontal axis indicate generalization granularity: 0 (wrong generalization), 1 (under-generalization), 2 (good-generalization) and 3 (over-generalization). Top row shows overall distribution of labels. Other rows represent number of sampled paths which have an edge with the corresponding label at the given distance from starting node.

Taxonomy	Overall accuracy	Per-node accuracy
WIBI$_E$	0.243	0.230
HEADS (entity)	**0.703**	**0.727**
WIBI$_C$	0.381	0.408
HEADS (category)	**0.670**	**0.725**

Table 4: Accuracy of specializations (computed using 100 (node, descendant) pairs). Overall accuracy is fraction of sampled (node, descendants) pairs which are correct, and per-node accuracy represents the average ratio of correct descendants per node. Results for entity and category descendants of HEADS are reported separately.

(0.3% vs 16.3%), which can be largely attributed to the simplification procedure (cf. Section 3.3). Despite the removal of 65% of categories through simplification, HEADS still does not suffer significantly from over-generalizations.

4.3.3 Specializations evaluation

A good taxonomy provides not only accurate generalizations going upwards in the taxonomy, but also accurate specializations going downwards. To evaluate this aspect, three human annotators annotate the correctness of a sample of descendants, for nodes in the taxonomies WIBI$_E$, WIBI$_C$ and HEADS. To avoid bias, nodes (entities for WIBI$_E$; categories for WIBI$_C$, HEADS) are sorted in decreasing order of the number of descendants in the respective taxonomies. 10 nodes at fixed ranks (5, 10, .., 50) from each list are selected for evaluation. To enable a comparison of WIBI$_E$ with WIBI$_C$ and HEADS, category nodes are manually mapped to equivalent entity nodes and vice-versa (e.g., *Category:Concepts* is mapped to the entity *Concept*). The annotators judge the correctness of 10 randomly sampled descendants for each selected node in each of the three taxonomies (see Table 4). HEADS is almost three times as accurate for entities as WIBI$_E$, and almost twice as accurate for categories as WIBI$_C$.

4.3.4 Extrinsic evaluation

This section compares HEADS, WIBI$_E$ and WIBI$_C$+H$_E$ on the task of selecting correct generalizations (e.g., *Countries*) for the variable slot in lexicalized templates such as *Passport of [X]*. These templates are mined by aggregation of Wikipedia page titles (e.g., *Passport of France*, *Passport of Canada*). The lexical fillers observed in the titles (e.g., *France*, *Canada*) are automatically disambiguated to a specific page (e.g., *France, European country* rather than *France, NY town*)[10], resulting in a set of filler entities for a template referred to as the template *support*.

To evaluate a taxonomy, for each template, the taxonomy is repeatedly traversed starting from subsamples of the support entities and equal-sized samples of random entities. Each non-leaf node in the taxonomy receives a score equal to the difference between the counts of support entities versus random

[10]Details of non-trivial problems of template mining and filler disambiguation are omitted due to space constraints, as they are not the main focus of this paper.

Template	Selected Generalizations		
	WIBI$_E$	WIBI$_C$+H$_E$	HEADS
railways in [X]	Tool, Entity, Publication, Operation (mathematics), Property (philosophy), Administrative division, Fine art, Material, Wealth, Combination	Geography, Countries, Statistics, Mathematical and quantitative methods (economics), Least developed countries, Capitals	Cities, Least developed countries, Administrative territorial entities
forestry in [X]	Entity, Wealth	Muslim-majority countries, Geography, Countries, Statistics, Mathematical and quantitative methods (economics), French-speaking countries and territories, Least developed countries	Muslim-majority countries, Least developed countries, Administrative territorial entities
[X] reader	Economic system, Entity, Document, Property (philosophy), Fine art, Material, Wealth	Philosophical concepts, Branches of philosophy, Concepts in metaphysics, Digital technology, Society, Psychology, Intelligence, Classification systems	Intellectual works, Concepts, Storage media, Literary characters
[X] ' day	Plurality (voting)	Public economics, Heavy metal subgenres, Intelligence, Economic policy, Heavy metal musical groups by nationality	Social groups, Occupations, Creative works
tomb of [X]	Tool, Value (mathematics), Publication, Proclamation, Official, Document, Instance (computer science), Fine art, Capital (economics), Material, Aesthetics, Electoral district, *[+2 more]*	People by nationality, Countries by continent, Hebrew Bible people, Ancient people, Religion, Genetics, Behavior, People by occupation, Fields of application of statistics, Jewish priests, Monarchy, Statistics, *[+16 more]*	People, Families, Ethnic groups, Noble titles

Table 5: Lists of selected generalizations for HEADS and WIBI taxonomies.

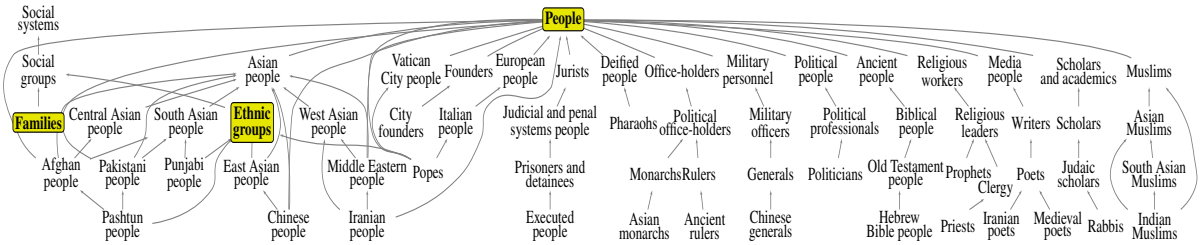


Figure 6: A view of the largest connected component of HEADS explored during the generalization of the template *Tomb of [X]*. The highlighted nodes are the selected generalizations.

entities from which the node can be reached in the given taxonomy. A node is selected as the generalization of fillers for the template, if its score is higher than both 1) the score of any of its parents, and 2) the sum of the scores of its children. For example, Figure 6 shows a subset of the largest connected component of HEADS explored while generalizing the fillers of the template *Tomb of [X]*. The selected generalizations are highlighted. The rationale behind this process is that in a good taxonomy, entities in the support (e.g., *France*, *Canada*) should consistently activate the same set of good generalizations (e.g., *Countries*).

Table 5 shows the lists of generalizations obtained with WIBI$_E$, WIBI$_C$+H$_E$ and HEADS for a few templates. A quantitative comparison of the results is inherently complex and outside the scope of this paper, yet it is immediately apparent that the generalizations obtained with WIBI$_C$+H$_E$ and WIBI$_E$ are generally quite noisy (e.g., *Genetics* for *Tomb of [X]*, *Wealth* for *railways in [X]*). On the contrary, HEADS shows a superior ability to select meaningful and compact generalizations that account for the polysemy of the templates without sacrificing precision (e.g., *People*, *Families*, *Ethnic groups* and *Noble titles* for *Tomb of [X]*).

5 Conclusion

Whether built from scratch or derived by filtering existing data, automatically-constructed taxonomies are accurate and useful only to the extent that they correctly assert not only short-range, but also longer-range generalizations among concepts or entities. The unified taxonomy introduced in this paper assembles entities and categories from Wikipedia that are in *is-a* relation relative to one another, primarily by detecting and analyzing lexical heads. A thorough evaluation framework is presented, and applied to the new taxonomy. In every respect, the taxonomy represents a significant improvement over the state of the art. It is more accurate along paths of arbitrary length and provides more accurate specializations.

References

G. de Melo and G. Weikum. 2010. MENTA: Inducing multilingual taxonomies from Wikipedia. In *Proceedings of the 19th International Conference on Information and Knowledge Management*, pages 1099–1108.

X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, and K. Murphy. 2014. Knowledge Vault: A Web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 601–610, New York, NY.

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

T. Flati, D. Vannella, T. Pasini, and R. Navigli. 2014. Two is bigger (and better) than one: the Wikipedia Bitaxonomy project. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 945–955, Baltimore, Maryland. Association for Computational Linguistics.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics*, pages 539–545, Nantes, France.

J. Hoffart, F. Suchanek, K. Berberich, and G. Weikum. 2013. YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence Journal. Special Issue on Artificial Intelligence, Wikipedia and Semi-Structured Resources*, 194:28–61.

E. Hovy, R. Navigli, and S. Ponzetto. 2013. Collaboratively built semi-structured content and Artificial Intelligence: The story so far. *Artificial Intelligence*, 194:2–27.

J. Hu, G. Wang, F. Lochovsky, J. Sun, and Z. Chen. 2009. Understanding user's query intent with Wikipedia. In *Proceedings of the 18th World Wide Web Conference*, pages 471–480, Madrid, Spain.

J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. Mendes, S. Hellmann, M. Morsey, P. van Kleef, and S. Auer et al. 2015. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6(2):167–195.

R. Mihalcea. 2007. Using Wikipedia for automatic word sense disambiguation. In *Proceedings of the 2007 Conference of the North American Association for Computational Linguistics*, pages 196–203, Rochester, New York.

V. Nastase and M. Strube. 2008. Decoding Wikipedia categories for knowledge acquisition. In *AAAI*, volume 8, pages 1219–1224.

V. Nastase and M. Strube. 2013. Transforming Wikipedia into a large scale multilingual concept network. *Artificial Intelligence*, 194:62–85.

V. Nastase, M. Strube, B. Börschinger, C. Zirn, and A. Elghafari. 2010. WikiNet: A very large scale multi-lingual concept network. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, pages 17–23, La Valetta, Malta.

S. Ponzetto and M. Strube. 2007. Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, pages 1440–1445, Vancouver, British Columbia.

S. Ponzetto and M. Strube. 2011. Taxonomy induction based on a collaboratively built knowledge repository. *Artificial Intelligence*, 175(9–10):1737–1756.

L. Ratinov and D. Roth. 2012. Learning-based multi-sieve co-reference resolution with knowledge. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1234–1244, Jeju Island, Korea.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 1375–1384, Portland, Oregon.

M. Remy. 2002. Wikipedia: The free encyclopedia. *Online Information Review*, 26(6):434.

F. Suchanek, G. Kasneci, and G. Weikum. 2007. Yago: A core of semantic knowledge unifying Wordnet and Wikipedia. In *Proceedings of the 16th International World Wide Web Conference*, pages 697–706, Banff, Canada.

D. Vrandečić and M. Krötzsch. 2014. Wikidata: A free collaborative knowledge base. *Communications of the ACM*, 57:78–85.

F. Wu and D. Weld. 2010. Open information extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127, Uppsala, Sweden.

Joint Learning of Local and Global Features for Entity Linking via Neural Networks

Thien Huu Nguyen[†], **Nicolas Fauceglia**[#], **Mariano Rodriguez Muro**[§], **Oktie Hassanzadeh**[§], **Alfio Massimiliano Gliozzo**[§] and **Mohammad Sadoghi**[‡]
[†] New York University, [#] Carnegie Mellon University, [‡] Purdue University
[§] IBM T.J. Watson Research Center, Yorktown Heights, New York, USA
thien@cs.nyu.edu,fauceglia@cs.cmu.edu
{mrodrig,hassanzadeh,gliozzo}.us.ibm.com,msadoghi@purdue.edu

Abstract

Previous studies have highlighted the necessity for entity linking systems to capture the local entity-mention similarities and the global topical coherence. We introduce a novel framework based on convolutional neural networks and recurrent neural networks to simultaneously model the local and global features for entity linking. The proposed model benefits from the capacity of convolutional neural networks to induce the underlying representations for local contexts and the advantage of recurrent neural networks to adaptively compress variable length sequences of predictions for global constraints. Our evaluation on multiple datasets demonstrates the effectiveness of the model and yields the state-of-the-art performance on such datasets. In addition, we examine the entity linking systems on the domain adaptation setting that further demonstrates the cross-domain robustness of the proposed model.

1 Introduction

We address the problem of entity linking (EL): mapping entity mentions in documents to their correct entries (called target entities) in some existing knowledge bases (KB) like Wikipedia. For instance, in the sentence "*Liverpool suffered an upset first home league defeat of the season.*", an entity linking system should be able to identify the entity mention "*Liverpool*" as a football club rather than a city in England in the knowledge bases. This is a challenging problem of natural language processing, as the same entity might be presented in various names, and the same entity mention string might refer to different entities in different contexts. Entity linking is a fundamental task for other applications such as information extraction, knowledge base construction etc.

In order to tackle the ambiguity in EL, previous studies have first generated a set of target entities in the knowledge bases as the referent candidates for each entity mention in the documents, and then solved a ranking problem to disambiguate the entity mention. The key challenge in this paradigm is the ranking model that computes the relevance of each target entity candidate to the corresponding entity mention using the available context information in both the documents and the knowledge bases.

The early approach for the ranking problem in EL has resolved the entity mentions in documents *independently* (*the local approach*), utilizing various *discrete* and *hand-designed* features/heuristics to measure the local mention-to-entity relatedness for ranking. These features are often specific to each entity mention and candidate entity, covering a wide range of linguistic and/or structured representations such as lexical and part-of-speech tags of context words, dependency paths, topical features, KB infoboxes (Bunescu and Pasca, 2006; Mendes et al., 2011; Cassidy et al., 2011; Ji and Grishman, 2011; Shen et al., 2014) etc. Although the local approach can exploit a rich set of discrete structures for EL, its limitation is twofold:

(i) The independent ranking mechanism in the local approach overlooks the topical coherence among the target entities referred by the entity mentions within the same document. This is undesirable as the topical coherence has been shown to be effective for EL in the previous work (Han et al.,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2310–2320, Osaka, Japan, December 11-17 2016.

2011; Hoffart et al., 2011; Ratinov et al., 2011; He et al., 2013b; Alhelbawy and Gaizauskas, 2014; Pershina et al., 2015).

(ii) The local approach might suffer from the data sparseness issue of unseen words/features, the difficulty of calibrating, and the failure to induce the underlying similarity structures at high levels of abstraction for EL (due to the extensive reliance on the hand-designed coarse features) (Sun et al., 2015; Francis-Landau et al., 2016).

The first drawback of the local approach has been overcome by *the global models* in which all entity mentions (or a group of entity mentions) within a document are disambiguated *simultaneously* to obtain a *coherent* set of target entities. The central idea is that the referent entities of some mentions in a document might in turn introduce useful information to link other mentions in that document due to the semantic relatedness among them. For example, the appearances of "*Manchester*" and "*Chelsea*" as the football clubs in a document would make it more likely that the entity mention "*Liverpool*" in the same document is also a football club. Unfortunately, the coherence assumption of the global approach does not hold in some situations, necessitating the discrete/coarse features in the local approach as a mechanism to compensate for the potential exceptions of the coherence assumption (Ratinov et al., 2011; Hoffart et al., 2011; Sil et al., 2012; Durrett and Klein, 2014; Pershina et al., 2015). Consequently, the global approach is still subject to the second limitation of data sparseness of the local approach due to their use of discrete features.

Recently, the surge of neural network (NN) models has presented an effective mechanism to mitigate the second limitation of the local approach. In such models, words are represented by *continuous* representations (Bengio et al., 2003; Turian et al., 2010; Mikolov et al., 2013) and features for the entity mentions and candidate entities are automatically learnt from data. This essentially alleviates the data sparseness problem of unseen words/features and helps to extract more effective features for EL in a given dataset (Kalchbrenner et al., 2014; Nguyen et al., 2016a). In practice, the features automatically induced by NN are combined with the discrete features in the local approach to extend their coverage for EL (Sun et al., 2015; Francis-Landau et al., 2016). However, as the previous NN models for EL are local, they cannot capture the global interdependence among the target entities in the same document (the first limitation of the local approach).

Guided by these analyses, in this paper, we propose to use neural networks to model both the local mention-to-entity similarities and the global relatedness among target entities in an unified architecture. This allows us to inherit all the benefits from the previous systems as well as overcome their inherent issues. Our work is an extension of (Francis-Landau et al., 2016) which only considers the local similarities.

Given a document, we simultaneously perform linking for every entity mention from the beginning to the end of the document. For each entity mention, we utilize convolutional neural networks (CNN) to obtain the distributed representations for the entity mention as well as its target candidates. These distributed representations are then used for two purposes: (i) computing the local similarities for the entity mention and target candidates, and (ii) functioning as the input for the recurrent neural networks (RNN) that runs over the entity mentions in the documents. The role of the RNNs is to accumulate information about the previous entity mentions and target entities, and provide them as the global constraints for the linking process of the current entity mention. We systematically evaluate the proposed model on multiple datasets in both the general setting and the domain adaptation setting. The experiment results show that the proposed model outperforms the current state-of-the-art models on the evaluated datasets. To our knowledge, this is also the first work investigating the EL problem in the domain adaptation setting.

2 Model

The entity linking problem in this work can be formalized as follows. Let D be the input document and $M = \{m_1, m_2, \ldots, m_k\}$ be the entity mentions in D. The goal is to map each entity mention m_i to its corresponding Wikipedia page (entity) or return "*NIL*" if m_i is not present in Wikipedia. For each entity

mention $m_i \in D$, let $P_i = \{p_{i1}, p_{i2}, \ldots p_{in_i}\}$ be its set of Wikipedia candidate pages (entities)[1] where n_i is the number of page candidates for m_i. Also, let $p_i^* \in P_i$ be the correct target entity for m_i.

Following Francis-Landau et al. (2016), we represent each entity mention m_i by the triple $m_i = (s_i, c_i, d_i)$, where s_i is the *surface string* of m_i, c_i is the *immediate context* (within some predefined window) of m_i and d_i is the *entire document* containing m_i. Essentially, s_i, c_i and d_i are the sequences of words to capture the contexts or topics of m_i at multiple granularities. For the target candidate pages p_{ij}, we use the *title* t_{ij} and *body content* b_{ij} to represent them ($p_{ij} = (t_{ij}, b_{ij})$). For convenience, we also denote $p_i^* = (t_i^*, b_i^*)$ for the correct entity pages. Again, t_{ij}, b_{ij}, t_i^* and b_i^* are also sequences of words.

In order to link the entity mentions, the strategy is to assign a relevance score $\phi(m_i, p_{ij})$ for each target candidate p_{ij} of m_i, and then use these scores to rank the candidates for each mention. In this work, we decompose $\phi(m_i, p_{ij})$ as the sum of the two following factors:

$$\phi(m_i, p_{ij}) = \phi_{local}(m_i, p_{ij}) + \phi_{global}(m_1, m_2, \ldots, m_i, P_1, P_2, \ldots, P_i)$$

In this formula, $\phi_{local}(m_i, p_{ij})$ represents the local similarities between m_i and p_{ij}, i.e, only using the information related to m_i and p_{ij}. $\phi_{global}(m_1, m_2, \ldots, m_i, P_1, P_2, \ldots, P_i)$, on the other hand, additionally considers the other mentions and candidates in the document, attempting to model the interdependence among these objects. The denotation $\phi_{global}(m_1, m_2, \ldots, m_i, P_1, P_2, \ldots, P_i)$ implies that we are computing the ranking scores for all the target candidates of all the entity mentions in each document simultaneously, preserving the order of the entity mentions from the beginning to the end of the document.

The model in this work consists of three main components: (i) the encoding component that applies convolutional neural networks to induce the distributed representations for the input sequences s_i, c_i, d_i, t_{ij}, and b_{ij}, (ii) the local component that computes the local similarities $\phi_{local}(m_i, p_{ij})$ for each entity mention m_i, and (iii) the global component that runs recurrent neural networks on the entity mentions $\{m_1, m_2, \ldots, m_k\}$ to generate the global features $\phi_{global}(m_1, m_2, \ldots, m_i, P_1, P_2, \ldots, P_i)$.

2.1 Encoding

Let x be some context word sequence of the entity mentions or target candidates (i.e, $x \in \{s_i, c_i, d_i\}_i \cup \{t_{ij}, p_{ij}\}_{i,j} \cup \{t_i^*, b_i^*\}_i$). In order to obtain the distributed representation for x, we first transform each word $x_i \in x$ into a real-valued, h-dimensional vector w_i using the word embedding table E (Mikolov et al., 2013): $w_i = E[x_i]$. This essentially converts the word sequence x into a sequence of vectors that is padded with zero vectors to form a fixed-length sequence of vectors $w = (w_1, w_2, \ldots, w_n)$ of length n.

In the next step, we apply the convolution operation over w to generate the hidden vector sequence, that is then transformed by a non-linear function G and pooled by the sum function (Francis-Landau et al., 2016). Following the previous work on CNN (Nguyen and Grishman, (2015a; 2015b)), we utilize the set L of multiple window sizes to parameterize the convolution operation. Each window size $l \in L$ corresponds to a convolution matrix $M_l \in \mathbb{R}^{v \times lh}$ of dimensionality v. Eventually, the concatenation vector $\bar{x}$ of the resulting vectors for each window size in L would be used as the distributed representation for x:

$$\bar{x} = \bigoplus_{l \in L} \sum_{i=1}^{n-l+1} G(M_l w_{i:(i+l-1)})$$

where $\bigoplus$ is the concatenation operation over the window set L and $w_{i:(i+l-1)}$ is the concatenation vector of the given word vectors.

For convenience, let $\bar{s}_i$, $\bar{c}_i$, $\bar{d}_i$, $\bar{t}_{ij}$, $\bar{b}_{ij}$, $\bar{t}_i^*$ and $\bar{b}_i^*$ be the distributed representations of s_i, c_i, d_i, t_{ij}, p_{ij}, t_i^* and b_i^* obtained by the convolution procedure above, respectively. Note that we apply the same set of convolution parameters for each type of text granularity in the source document D as well as in the

[1]For comparison purpose, we use the target candidates provided by Francis-Landau et al. (2016). Essentially, a query generation is executed for each entity mention, whose outputs are combined with link counts to retrieve the potential entities (including "*NIL*"). The query generation itself involves removing stop words, plural suffixes, punctuation, and leading or tailing words.

target entity side. The vector representations of the context would then be fed into the next components to compute the features for EL.

2.2 Local Similarities

We employ the local similarities $\phi_{local}(m_i, p_{ij})$ from (Francis-Landau et al., 2016), the state-of-the-art neural network model for EL. In particular:

$$\phi_{local}(m_i, p_{ij}) = \phi_{sparse}(m_i, p_{ij}) + \phi_{CNN}(m_i, p_{ij}) = W_{sparse}F_{sparse}(m_i, p_{ij}) + W_{CNN}F_{CNN}(m_i, p_{ij})$$

In this formula, W_{sparse} and W_{CNN} are the weights for the feature vectors F_{sparse} and W_{CNN} respectively. $F_{sparse}(m_i, p_{ij})$ is the sparse feature vector obtained from (Durrett and Klein, 2014). This vector captures various linguistic properties and statistics that have been discovered in the previous studies for EL. The representative features include the anchor text counts from Wikipedia, the string match indications with the title of the Wikipedia candidate pages, or the information about the shape of the queries for candidate generations (Francis-Landau et al., 2016).

$F_{CNN}(m_i, p_{ij})$, on the other hand, involves the cosine similarities between the representation vectors at multiple granularities of m_i and p_{ij}. In particular:

$$F_{CNN}(m_i, p_{ij}) = [cos(\bar{s}_i, \bar{t}_{ij}), cos(\bar{c}_i, \bar{t}_{ij}), cos(\bar{d}_i, \bar{t}_{ij}), cos(\bar{s}_i, \bar{b}_{ij}), cos(\bar{c}_i, \bar{b}_{ij}), cos(\bar{d}_i, \bar{b}_{ij})] \quad (1)$$

The intuition for this computation is that the similarities at different levels of contexts might help to enforce the potential topic compatibility between the contexts of the entity mentions and target candidates for EL (Francis-Landau et al., 2016).

2.3 Global Similarities

In order to encapsulate the coherence among the entity mentions and their target entities, we run recurrent neural networks over the sequences of the representation vectors for the entity mentions (i.e, the vector sequences for the surface strings $(\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_k)$ and for the immediate contexts $(\bar{c}_1, \bar{c}_2, \ldots, \bar{c}_k)$) and the target entities (i.e, the vector sequences for the page titles $(\bar{t}^*_1, \bar{t}^*_2, \ldots, \bar{t}^*_k)$ and for the body contents $(\bar{b}^*_1, \bar{b}^*_2, \ldots, \bar{b}^*_k)$)[2].

Let us take the representation vector sequence of the body contents of the target pages $(\bar{b}^*_1, \bar{b}^*_2, \ldots, \bar{b}^*_k)$[3] as an example. The recurrent neural network with the recurrent function Φ for this sequence will generate the hidden vector sequence $(h^b_1, h^b_2, \ldots, h^b_k)$ where: $h^b_i = \Phi(h^b_{i-1}, \bar{b}^*_i)$.

Each vector h^b_i in this sequence encodes or summarizes the information about the content of the previous target entities (i.e, before i) in the document due to the property of RNN.

Given the hidden vector sequence, when predicting the target entity for the entity mention m_i, we ensure that the target entity is consistent with the global information stored in h^b_{i-1}. This is achieved by using the cosine similarities between h^b_{i-1} and the representation vectors of each target candidate p_{ij} of m_i, (i.e, $cos(h^b_{i-1}, \bar{t}_{ij})$ and $cos(h^b_{i-1}, \bar{b}_{ij})$) as the global features for the ranking score.

We can repeat this process for the other representation vector sequences in both the entity mention side and the target entity side. The resulting global features would then be grouped into a single feature vector to compute the global similarity score $\phi_{global}(m_1, m_2, \ldots, m_i, P_1, P_2, \ldots, P_i)$ as in the local similarity section. An overview of the whole model is presented in Figure 1.

Regarding the reccurent function Φ, we employ the gated recurrent units (GRU) (Cho et al., 2014) to alleviate the "*vanishing gradient problem*" of RNN. GRU is a simplified version of long-short term memory units (LSTM) that has been shown to achieve comparable performance (Józefowicz et al., 2015).

Finally, for training, we jointly optimize the parameters for the CNNs, RNNs and weight vectors by maximizing the log-likelihood of a labeled training corpus. We utilize the stochastic gradient descent algorithm and the AdaDelta update rule (Zeiler, 2012). The gradients are computed via back-propagation. Following (Francis-Landau et al., 2016), we do not update the word embedding table during training.

[2]Note that we have different recurrent neural networks for different context vector sequences.

[3]In the training process, $(\bar{b}^*_1, \bar{b}^*_2, \ldots, \bar{b}^*_k)$ are obtained from the golden target entities while in the test time, they are retrieved from the predicted target entities.

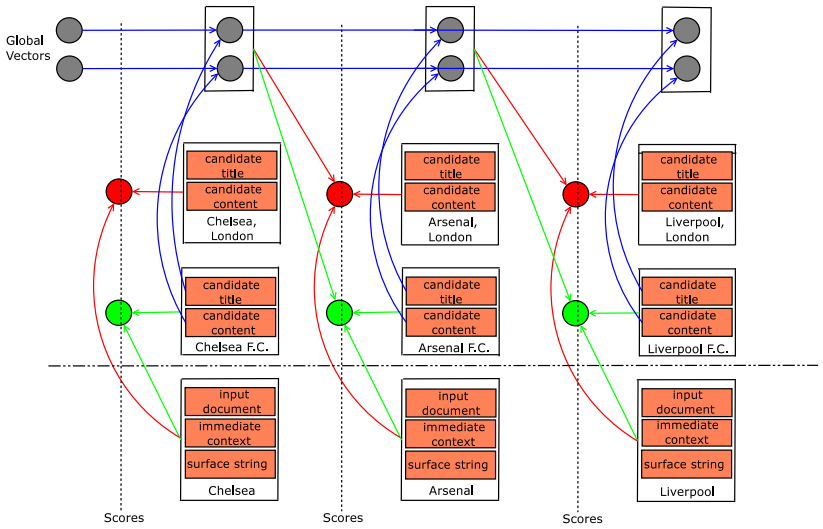


Figure 1: Joint model for learning local and global features for a document with 3 entity mentions: *Chelsea*, *Arsenal* and *Liverpool*. Each of the entity mentions has two entity candidate pages (either a football club or a city).The orange rectangles denote the CNN-induced representation vectors $\bar{s}_i$, $\bar{c}_i$, $\bar{d}_i$, $\bar{t}_{ij}$ and $\bar{b}_{ij}$. The circles in red and green are the ranking scores for the target candidates, in which the green circles correspond to the correct target entities. Finally, the circles in grey are the hidden vectors (i.e, the global vectors) of the RNNs running over the entity mentions. We only show the global entity vectors in this figure to improve the visualization.

3 Experiments

3.1 Datasets

Following (Francis-Landau et al., 2016), we evaluate the models on 4 different entity linking datasets:

i) ACE (Bentivogli et al., 2010): This corpus is from the 2005 evaluation of NIST. It is also used in (Fahrni and Strube, 2014) and (Durrett and Klein, 2014).

ii) CoNLL-YAGO (Hoffart et al., 2011): This corpus is originally from the CoNLL 2003 shared task of named entity recognition for English.

iii) WP (Heath and Bizer, 2011): This dataset consists of short snippets from Wikipedia.

iv) WIKI (Ratinov et al., 2011): This dataset contains 10,000 randomly sampled Wikipedia articles. The task is to disambiguate the links in each article[4].

For all the datasets, we use the standard data splits (for training data, test data and development data) as the previous works for comparable comparison (Francis-Landau et al., 2016).

3.2 Parameters and Resources

For all the experiments below, in the CNN models to learn the distributed representations for the inputs, we use window sizes in the set $L = \{2, 3, 4, 5\}$ for the convolution operation with the dimensionality $v = 200$ for each window size[5]. The non-linear function for transformation is $G = \tanh$.

We employ the English Wikipedia dump from June 2016 as our reference knowledge base.

Regarding the input contexts for the entity mentions and the target candidates, we utilize the window size of 10 for the immediate context c_i, and only extract the first 100 words in the documents for d_i and b_{ij}.

Finally, we pre-train the word embedings on the whole English Wikipedia dump using the word2vec toolkit (Mikolov et al., 2013). The training parameters are set to the default values in this toolkit. The dimensionality of the word embeddings is 300.

Note that every parameter and resource in this work is either taken from the previous work (Nguyen and Grishman, 2016b; Francis-Landau et al., 2016) or selected by the development data.

[4]As noted by Francis-Landau et al. (2016) and Nguyen et al. (2014b), the original Wikipedia dump in Ratinov et al. (2011) is no longer accessible, so we cannot duplicate the results or conduct comparable experiments with (Ratinov et al., 2011). We instead compare our performance with (Francis-Landau et al., 2016) that provides the access to their Wikipedia dump.

[5]As we need to compute the cosine similarities between the hidden vectors of the RNN models and the representation vectors of the target candidates, the number of hidden units for the RNN is set to $200|L| = 800$ naturally.

3.3 Evaluating the Global Features

In this section, we evaluate the effectiveness of the global features for EL. In particular, we differentiate two types of global features based on the side of information we expect to enforce the coherence. The first type of global features (***global-mention***) concerns the entity mention side and involves applying the global RNN models on the CNN-induced representation vectors of the entity mentions (i.e, the surface vectors $(\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_k)$ and the immediate context vectors $(\bar{c}_1, \bar{c}_2, \ldots, \bar{c}_k)$). The second type of global features (***global-entity***), on the other hand, focuses on the target entity side and models the coherence with the representation vectors of the target entities (i.e, the page title vectors $(\bar{t}^*_1, \bar{t}^*_2, \ldots, \bar{t}^*_k)$ and the body content vectors $(\bar{b}^*_1, \bar{b}^*_2, \ldots, \bar{b}^*_k)$). Table 1 reports the development performance (F1 scores) of the proposed model on different cases where the *global-mention* and *global-entity* features are included or excluded from the model.

Global Features	Dataset		
	ACE	CoNLL	WP
No	86.1	89.3	84.0
global-mention	86.8	90.2	84.2
global-entity	**86.9**	**90.7**	**84.2**
global-mention + *global-entity*	86.2	90.6	84.0

Table 1: Performance of the global features on the development set. *No* means not using the global features.

The most important observation from the table is that the global features, in general, help to improve the performance of the model on different datasets. This is substantial on the ACE and CoNLL datasets when only one type of the global features (either *global-mention* or *global-entity*) is integrated into the model. The combination of *global-mention* and *global-entity* is not very effective as it is actually worse than the performance of the individual global feature types. This suggests that *global-mention* and *global-entity* might cover overlapping information and their combination would inject redundancy into the model. The best performance is achieved by the *global-entity* features that would be used in all the evaluations below.

3.4 Comparing to the Previous Work

This section compares the proposed system (called *Global-RNN*) with the state-of-the-art models on our four datasets. These systems include the neural network model in (Francis-Landau et al., 2016), the joint model for entity analysis in (Durrett and Klein, 2014) and the AIDA-light system with two-stage mapping in (Nguyen et al., 2014b)[6]. Table 2 shows the performance of the systems on the test sets with the reference knowledge base of the June 2016 Wikipedia dump. We also include the performance of the systems on the December 2014 Wikipedia dump that was used and provided by (Francis-Landau et al., 2016) for further and compatible comparison.

Systems	Wikipedia 2014				Wikipedia 2016			
	ACE	CoNLL	WP	WIKI	ACE	CoNLL	WP	WIKI
DK2014 (Durrett and Klein, 2014)	79.6	-	-	-	-	-	-	-
AIDA-LIGHT (Nguyen et al., 2014b)	-	84.8	-	-	-	-	-	-
Local CNN (Francis-Landau et al., 2016)	**89.9**	85.5	90.7	82.2	86.1	84.5	90.4	81.4
Global-RNN	89.7	**87.2**†	**91.2**†	**83.7**†	**87.8**†	**86.5**†	**91.2**†	**81.7**

Table 2: Performance of the systems. Cells marked with †designate the *Global-RNN* models that significantly outperform the *Local CNN* model ($\rho < 0.05$).

First, we see that the performance of the systems drop significantly when we switch from Wikipedia 2014 to Wikipedia 2016 (especially for the datasets ACE and CoNLL). This is can be partly explained by the inclusion of new entities (pages) into Wikipedia from 2014 to 2016 that has made the entity mentions in the datasets more ambiguous[7]. Second and more importantly, *Global-RNN* significantly outperforms

[6]We note that (Alhelbawy and Gaizauskas, 2014) and (Pershina et al., 2015) also use the CoNLL-YAGO dataset for their experiments. However, since they evaluate the models on the whole dataset rather than the test set as the other works do, they are not comparable to the performance we report in this paper.

[7]The number of Wikipedia pages in 2014 is about 4.5 million while this number is 5 million in June 2016.

the all the compared models (except for the ACE dataset on Wikipedia 2014 and the WIKI dataset on Wikipedia 2016), thereby demonstrating the benefits of the joint modeling for local and global features via neural networks for EL in this work.

3.5 Domain Adaptation Experiments

The purpose of this section is to further evaluate the models in the domain adaptation setting to investigate their cross-domain robustness for EL.

It is often observed in many natural language processing tasks that the performance of a model trained on a source domain would degrade significantly when it is applied to a different target domain (Blitzer et al., 2006; Daume, 2007; McClosky et al., 2010; Plank and Moschitti, 2013; Nguyen and Grishman, 2014a). Such a performance loss originates from a variety of mismatches between the source and the target domains, including the differences in vocabulary, data distributions, styles etc. This has motivated the domain adaptation research that aims to improve the cross-domain performance of the models by adaptation techniques.

One of the key strategies of the domain adaptation techniques is the search for the domain-independent features that are discriminative across different domains (Blitzer et al., 2006; Jiang, 2009; Plank and Moschitti, 2013; Nguyen and Grishman, 2014a). These invariants serve as the connectors between different domains and help to transfer the knowledge from one domain to the others. For EL, we hypothesize that the global coherence is an effective domain-independent feature that would help to improve the cross-domain performance of the models. The intuition is that the entities mentioned in a document of any domains should be related to each other. Eventually, we expect that the proposed model with global coherence features would be more robust to domain shifts than the local approach (Francis-Landau et al., 2016).

3.5.1 Dataset

We use the ACE dataset to evaluate the cross-domain performance of the models. ACE involves documents in 6 different domains: broadcast conversation (bc), broadcast news (bn), telephone conversation (cts), newswire (nw), usenet (un) and webblogs (wl). Following the common practice of domain adaptation research on this dataset (Plank and Moschitti, 2013; Nguyen et al., 2015c; Gormley et al., 2015), we use **news** (the union of **bn** and **nw**) as the source domain and **bc**, **cts**, **wl**, **un** as four different target domains. We take half of **bc** as the development set and use the remaining data for testing. We note that **news** consists of formally written documents while a majority of the other domains is informal text, making the source and target domains very divergent in terms of vocabulary and styles (Plank and Moschitti, 2013).

3.5.2 Evaluation

Table 3 compares *Global-RNN* with the neural network EL model in (Francis-Landau et al., 2016), the best reported model on the ACE dataset in the literature[8]. In this table, the models are trained on the source domain **news**, and evaluated on **news** itself (in-domain performance) (via 5-fold cross validation) as well as on the 4 target domains **bc**, **cts**, **wl**, **un** (out-of-domain performance). The experiments in this section are done with the 2016 Wikipedia dump.

Models	Domain				
	in-domain	bc	cts	wl	un
Local CNN (Francis-Landau et al., 2016)	90.6	87.8	88.7	80.2	82.1
Global-RNN	**91.0**	**88.7**†	**88.9**	**81.3**†	**83.1**†

Table 3: Cross-domain performance. Cells marked with †designate the *Glob-RNN* models that significantly outperform the *Local CNN* model ($\rho < 0.05$).

The first observation from the table is that the performance of all the compared systems on the target domains is much worse than the corresponding in-domain performance. In particular, the performance gap between the in-domain performance and the the worst out-of-domain performance (on the domain

[8]The performance of the model from (Francis-Landau et al., 2016) reported in this work is obtained by running their actual released system.

wl) is up to 10%, thus indicating the mismatches between the source and the target domains for EL. Second and most importantly, *Global-RNN* is consistently better than the model with only local features in (Francis-Landau et al., 2016) over all the target domains (although it is less pronounced in the **cts** domain). This demonstrates the cross-domain robustness of the proposed model and confirms our hypothesis about the domain-independence of the global coherence features for EL.

3.5.3 Analysis

In order to better understand the performance gap in the domain adaptation experiments for EL, we visualize the representation vectors of the entity mentions in different domains. In particular, after *Global-RNN* is trained, we retrieve the representation vectors $\bar{c}_i$ for the immediate contexts of the entity mentions in the source and target domains, project them into the 2-dimension space via the t-SNE algorithm and plot them. Figure 2 shows the plot.

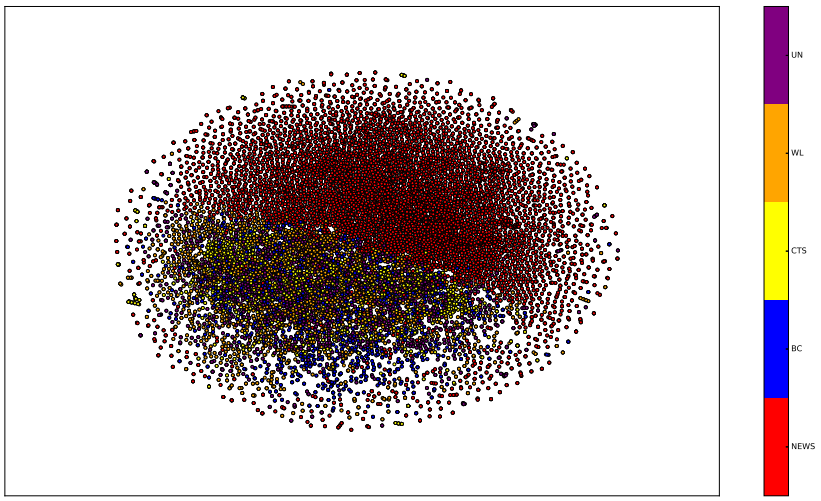


Figure 2: t-SNE visualization on the representation vectors c_i of different domains.

As we can see from the figure, the entity mentions in the target domains **bc**, **cts**, **wl** and **un** are quite separated from those of the source domain **news**, thereby explaining the performance loss in the domain adaption experiments.

It is not clear in Figure 2 why the models perform much worse on the target domains **wl** and **un** than the other domains (i.e, **bc** and **cts**). We further investigate this problem by computing the similarities between the target domains and the source domain. While there are several methods to estimate domain similarities (Plank and van Noord, 2011), in this work, we employ the mean of the cosine similarities of every mention pairs in the two domains of interest. Specifically, let E and F be the two domains of interest, and $E = \{e_1, e_2, \ldots, e_g\}$ and $F = \{f_1, f_2, \ldots, f_w\}$ be the sets of the representation vectors for the entity mentions in E and F respectively ($g = |E|, w = |F|$). The similarity between E and F is then given by:

$$Sim(E, F) = 100 \times \frac{\sum_{i=1}^{g} \sum_{j=1}^{w} cos(e_i, f_j)}{gw}$$

Table 4 shows the similarities between the source domain **news** and each target domains **bc**, **cts**, **wl** and **un** with respect to the representation vectors of the immediate context $\bar{c}_i$ (*context*) and the target entity titles $\bar{t}_i^*$ (*title*) for the entity mentions m_i. We also include the similarities in which the representation vectors are the local feature vectors $F_{CNN}(m_i, t_i^*)$ in Equation 1 (*interaction*). The goal of the local feature similarities is to characterize how the entity mentions in different domains interact with their target entities.

It is clear from the table that **wl** is the most dissimilar domain from the source domain. This is followed by **un** and partly explains the performance in Table 3.

Domain	*context*	*title*	*interaction*
bc	10.7	2.0	34.4
cts	11.4	2.0	32.6
wl	9.2	0.8	30.3
un	9.5	1.4	31.1

Table 4: Similarities to the source domain **news**.

4 Related Work

Entity linking or disambiguation has been studied extensively in NLP research, falling broadly into two major approaches: local and global disambiguation. Both approaches share the goal of measuring the similarities between the entity mentions and the target candidates in the reference KB. The local paradigm focuses on the internal structures of each separate mention-entity pair, covering the name string comparisons between the surfaces of the entity mentions and target candidates, entity popularity or entity type and so on (Bunescu and Pasca, 2006; Milne and Witten, 2008; Zheng et al., 2010; Ji and Grishman, 2011; Mendes et al., 2011; Cassidy et al., 2011; Shen et al., 2014). In contrast, the global approach jointly maps all the entity mentions within documents to model the topical coherence. Various techniques have been exploited for capturing such semantic consistency, including Wikipedia category agreement (Cucerzan, 2007), Wikipedia link-based measures (Kulkarni et al., 2009; Hoffart et al., 2011; Shen et al., 2012), Point-wise Mutual Information measures (Ratinov et al., 2011), integer linear programming (Cheng and Roth, 2013), PageRank (Alhelbawy and Gaizauskas, 2014; Pershina et al., 2015), stacked generalization (He et al., 2013a), to name a few. The entity linking techniques and systems have been actively evaluated at the NIST-organized Text Analysis Conference (Ji et al., 2014).

Neural networks are applied to entity linking very recently. He et al. (2013b) learn enttiy representation via Stacked Denoising Auto-encoders. Sun et al. (2015) employ convolutional neural networks and neural tensor networks to model mentions, entities and contexts while Francis-Landau et al. (2016) combine CNN-based representations with sparse features to improve the performance. However, none of these work utilize recurrent neural networks to capture the coherence features as we do in this work.

5 Conclusion

We present a joint model to learn the local context similarities and the global topical relatedness features for entity linking. CNNs are employed to capture the local similarities while RNNs are utilized to introduce the coherence. The model achieves the state-of-the-art performance on multiple datasets for entity linking. It is also shown to be more robust to domain shifts. Our future work is threefold: (i) integrating entity embedding models into the current work, (ii) exploring new neural models to jointly perform entity linking and entity extraction (Nguyen et al., 2016c), and (iii) further evaluating the models in the cross-lingual settings.

References

Ayman Alhelbawy and Robert Gaizauskas. 2014. Graph ranking for collective named entity disambiguation. In *ACL*.

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

Luisa Bentivogli, Pamela Forner, Claudio Giuliano, Alessandro Marchetti, Emanuele Pianta, and Kateryna Tymoshenko. 2010. Extending english ace 2005 corpus annotation with ground-truth links to wikipedia. In *Proceedings of the 2nd Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

R. Bunescu and M. Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *EACL*.

Taylor Cassidy, Zheng Chen, Javier Artiles, Heng Ji, Hongbo Deng, Lev-Arie Ratinov, Jing Zheng, Jiawei Han, and Dan Roth. 2011. Cuny-uiuc-sri tac-kbp2011 entity linking system description. In *TAC*.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *EMNLP*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP*.

Hal Daume. 2007. Frustratingly easy domain adaptation. In *ACL*.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. In *TACL*.

Angela Fahrni and Michael Strube. 2014. A latent variable model for discourse-aware concept and entity disambiguation. In *EACL*.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. In *NAACL*.

Matthew R. Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *EMNLP*.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: A graph-based method. In *SIGIR*.

Zhengyan He, Shujie Liu, Yang Song, Mu Li, Ming Zhou, and Houfeng Wang. 2013a. Efficient collective entity linking with stacking. In *EMNLP*.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013b. Learning entity representation for entity disambiguation. In *ACL*.

Tom Heath and Christian Bizer. 2011. Linked data: Evolving the web into a global data space. In *Morgan and Claypool, 1st edition*.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *ACL*.

Heng Ji, Joel Nothman, and Ben Hachey. 2014. Overview of tac-kbp2014 entity discovery and linking tasks. In *TAC*.

Jing Jiang. 2009. Multi-task transfer learning for weakly-supervised relation extraction. In *ACL-IJCNLP*.

Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *ICML*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *SIGKDD*.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *NAACL*.

Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*.

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

David Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *CIKM*.

Thien Huu Nguyen and Ralph Grishman. 2014a. Employing word representations and regularization for domain adaptation of relation extraction. In *ACL*.

Thien Huu Nguyen and Ralph Grishman. 2015a. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st NAACL Workshop on Vector Space Modeling for NLP (VSM)*.

Thien Huu Nguyen and Ralph Grishman. 2015b. Event detection and domain adaptation with convolutional neural networks. In *ACL-IJCNLP*.

Thien Huu Nguyen and Ralph Grishman. 2016b. Combining neural networks and log-linear models to improve relation extraction. In *Proceedings of IJCAI Workshop on Deep Learning for Artificial Intelligence (DLAI)*.

Dat Ba Nguyen, Johannes Hoffart, Martin Theobald, and Gerhard Weikum. 2014b. Aida-light: High-throughput named-entity disambiguation. In *WWW*.

Thien Huu Nguyen, Barbara Plank, and Ralph Grishman. 2015c. Semantic representations for domain adaptation: A case study on the tree kernel-based method for relation extraction. In *ACL-IJCNLP*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016a. Joint event extraction via recurrent neural networks. In *NAACL*.

Thien Huu Nguyen, Avirup Sil, Georgiana Dinu, and Radu Florian. 2016c. Toward mention detection robustness with recurrent neural networks. In *Proceedings of IJCAI Workshop on Deep Learning for Artificial Intelligence (DLAI)*.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *NAACL*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *ACL*.

Barbara Plank and Gertjan van Noord. 2011. Effective measures of domain similarity for parsing. In *ACL*.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *ACL*.

Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2012. Linden: Linking named entities with knowledge base via semantic knowledge. In *WWW*.

Wei Shen, Jianyong Wang, and Jiawei Han. 2014. Entity linking with a knowledge base: Issues, techniques, and solutions. In *TKDE*.

Avirup Sil, Ernest Cronin, Penghai Nie, Yinfei Yang, Ana-Maria Popescu, and Alexander Yates. 2012. Linking named entities to any database. In *EMNLP*.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *IJCAI*.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL*.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. In *CoRR, abs/1212.5701*.

Zhicheng Zheng, Fangtao Li, Minlie Huang, and Xiaoyan Zhu. 2010. Learning to link entities with knowledge base. In *NAACL*.

Structured Aspect Extraction

Omer Gunes[†]
[†] Department of Computer Science
University of Oxford
United Kingdom
first.last@cs.ox.ac.uk

Tim Furche[†‡]
[‡] Meltwater
Shoreditch, London
United Kingdom
first.last@meltwater.com

Giorgio Orsi[*‡]
[*]School of Computer Science
University of Birmingham
United Kingdom
g.orsi@cs.bham.ac.uk

Abstract

Aspect extraction identifies relevant features of an entity from a textual description and is typically targeted to product reviews, and other types of short text, as an enabling task for, e.g., opinion mining and information retrieval. Current aspect extraction methods mostly focus on aspect terms, often neglecting associated modifiers or embedding them in the aspect terms without proper distinction. Moreover, flat syntactic structures are often assumed, resulting in inaccurate extractions of complex aspects. This paper studies the problem of structured aspect extraction, a variant of traditional aspect extraction aiming at a fine-grained extraction of complex (i.e., hierarchical) aspects. We propose an unsupervised and scalable method for structured aspect extraction consisting of statistical noun phrase clustering, cPMI-based noun phrase segmentation, and hierarchical pattern induction. Our evaluation shows a substantial improvement over existing methods in terms of both quality and computational efficiency.

1 Introduction

The abundance of web data has made information extraction (IE) of strategic importance for data-driven companies such as, e.g., retailers, because of its applications to information retrieval (Kannan et al., 2011), knowledge base construction (Shin et al., 2015), media intelligence (Zeng et al., 2010), and conversational agents (Cassell, 2000). Among all IE-related tasks, *Aspect extraction* (AE) (Zhang and Liu, 2014) is certainly one of the most challenging. AE aims at identifying features of an entity, e.g., a phone, from a free text description ans is commonly associated with (aspect-based) sentiment analysis as a means of extracting *aspect terms* and associated *opinions* in, e.g., product reviews or other forms of opinionated and evaluative text (Hu and Liu, 2004; Popescu and Etzioni, 2005). Similar methods have also been used for extracting *attributes* from, e.g., product listings and other forms of *descriptive* text (Ghani et al., 2006; Kannan et al., 2011). An obvious challenge in aspect extraction is dealing with noisy, unstructured text (NUT). NUT has consistently challenged traditional NLP tools, in particular POS taggers and dependency parsers, due to ungrammatical sentences and incorrect orthography.

Example 1: Aspect extraction

Victorian two bedroom mid terrace property located in Cambridge and comprising of living room with ORIGINAL!!! cupboards, and ORIGINAL!!! picture rail.Stairway off living room leads to two bedrooms.

↓

{ bedroom mid terrace, picture rail.Stairway, cupboards, bedrooms, property} {Cambridge, ORIGINAL}

Example 1 shows the output of IBM's Alchemy Language service[1] when asked to retrieve aspect terms and named entities. Alchemy Language extracts picture rail.Stairway as an aspect due to a missing space after the full stop, and ORIGINAL as a named entity, due to incorrect orthography. Another challenge is distinguishing aspect terms from their *modifiers*. The terms Victorian, mid terrace, and the (nested) expression two bedroom are *qualifying* the term property, while the term two *quantifies* the term bedroom.

[1]https://alchemy-language-demo.mybluemix.net as of July 2016.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2321–2332, Osaka, Japan, December 11-17 2016.

Problem definition This paper studies the problem of recognizing and extracting *structured* aspects. *Structured Aspect Extraction* (SAE) generalizes aspect term extraction (ATE) with the extraction and *linking* of modifiers of the aspect term, identifying the possibly hierarchical structure of the aspects. Consider the text Victorian two bedroom mid terrace property from Example 1. An SAE system is expected to produce a single extraction with explicit annotations for *(i)* the aspect terms, i.e., property, bedroom *(ii)* the aspect modifiers, correctly typed as qualifiers, i.e., Victorian, mid terrace, and quantifiers, i.e., two, and *(iii)* the hierarchical structure of the aspect, i.e., $\langle\{\text{Victorian}, \langle\{\text{two}\}, \text{bedroom}\rangle, \text{mid terrace}\}, \text{property}\rangle$.

Contributions (Zhang and Liu, 2014) termed the recognition of *flat* $\langle\text{modifier}, \text{aspect}\rangle$ structures (i.e., a simple form of SAE) a very challenging task. In this paper we go beyond that, proposing (Section 2) a method for unsupervised SAE consisting of: **(1)** An effective normalization strategy specifically engineered for NUT. **(2)** A statistical noun phrase clustering method for unsupervised discovery of aspect terms and (raw) modifiers. **(3)** A cPMI-based noun-phrase segmentation, to identify multi-word expressions and nested structures. **(4)** A controlled induction of *structured aspect patterns*. We evaluate our method against state-of-the-art aspect extraction systems using data from the widely-accepted SemEval benchmarks and a SAE-specific dataset. The evaluation (Section 3) shows that our method improves on existing unsupervised systems for aspect-term extraction and solves the harder SAE task reliably.

Positioning Existing work on aspect extraction focuses mostly on three tasks: **(1)** Entity extraction (Zhang and Liu, 2014; Yahya et al., 2014), i.e., classifying entities in a free-text fragment (e.g., car, phone). **(2)** Aspect term extraction (Yu et al., 2011; Zhang and Liu, 2014; Chen et al., 2014), i.e., extracting features of the entity (e.g., battery, performance). **(3)** Opinion extraction (Pang and Lee, 2008; Yang and Cardie, 2014), i.e., extracting sentiments or opinions and link them to aspects terms.

(Probst et al., 2007) was the first method to go beyond simple aspect terms, proposing a semi-supervised method for extracting $\langle\text{modifier}, \text{aspect}\rangle$ pairs from product descriptions. Despite being supervised, the method has issues distinguishing genuine pairs representing aspects from spurious ones. (Kelly et al., 2012) makes extractions more precise by targeting $\langle\text{modifier}, \text{relation}, \text{aspect}\rangle$ triples in verbal phrases. Although precise, applying this method is unrealistic in practice since it severely impairs recall. In fact, empirical evidence suggests that most of the interesting aspects occur in proximity of named entities and in noun phrases. This shows the need for more flexibility in extracting aspects.

Leveraging a dependency parser is a possible way to capture more complex aspect structures, or at least to provide clues about the presence of entity modifiers (e.g., via AMOD relations). However, deep parsing is known to be inaccurate on NUT, and training suitable models difficult, since NUT lacks proper grammatical structure in the first place. This has been clear since the works by (Popescu and Etzioni, 2005) and (Raju et al., 2009) proposing, as a solution, ATE methods based on noun-phrase clustering. More recently, (Kim et al., 2012) used noun-phrase clustering to extract $\langle\text{modifier}, \text{aspect}\rangle$ pairs where modifiers are arbitrary n-grams, generically typed as either modifiers or quantifiers. Although limited to flat structures, this is currently the closest work to SAE we are aware of.

Similarly to (Popescu and Etzioni, 2005; Raju et al., 2009), we use *frequency*-based noun-phrase clustering to detect aspect terms in an unlabelled corpus. Besides making clustering more accurate by normalizing the descriptions, our method detects the underlying structure of the aspect via a cPMI-based segmentation of noun phrases that simulates an aspect-oriented parsing of the sentence. PMI values are also used by (Popescu and Etzioni, 2005) but only for aspect-term detection. Another option is to use topic modeling for clustering (Titov and McDonald, 2008; Sauper and Barzilay, 2013; Zhang and Liu, 2014). Topic modeling is effective in ATE but can under-cluster when applied to SAE, e.g., noun phrases headed by bedroom and kitchen are clustered together despite having different modifiers, producing incorrect associations of modifiers to aspect terms.

The ability to identify arbitrary hierarchical structures distinguishes our work from other aspect extraction methods based on distributional analysis of syntactic features such as, e.g., (Qiu et al., 2011; Zhuang et al., 2006; Wu et al., 2009; Yu et al., 2011; Zhou et al., 2013; Liu et al., 2013), and rule-based methods such as, e.g., (Poria et al., 2014). Our method can deal with a wide range of aspect structures without resorting to supervision as commonly done in methods based on sequence labelling, such as, e.g., (Li et

al., 2010; Choi and Cardie, 2010; Jakob and Gurevych, 2010; Yang and Cardie, 2014). Our hierarchical extraction should not be confused with the mapping of flat aspects to taxonomies and knowledge bases as done, e.g., by (Yahya et al., 2014). The generalization of these hierarchical structures into patterns enables us to transfer these structures to the extracted aspects, thus labelling the extractions as in (Kim et al., 2012), distinguishing the aspect term, from its qualifiers and quantifiers. Let us stress out that these coarse hierarchical structures cannot be trivially derived from fine-grained dependency relations.

2 Structured Aspect Extraction

We now formally define the task of *structured aspect extraction* (SAE). We introduce some non-standard notation but we assume readers are familiar with aspect extraction terminology.

Definitions A *structured aspect* is a tuple $\langle \mathsf{M}, \mathsf{t} \rangle$, where t is an *aspect term* (or *head* of the aspect) and $\mathsf{M}=\{\mathsf{m}_1, \ldots, \mathsf{m}_n\}$ is a sequence of modifiers of the aspect term (or *tail* of the aspect). A labelling function $\lambda : \mathsf{M} \cup \{\mathsf{t}\} \mapsto \mathsf{L}$ maps aspect terms and modifiers to a set L of labels. Aspect terms are labelled as the real-world entities they represent, e.g., BEDROOM. Each modifier $\mathsf{m} \in \mathsf{M}$ is labelled as either $\lambda(\mathsf{t})$-QUANTIFIER or $\lambda(\mathsf{t})$-QUALIFIER where $\lambda(\mathsf{t})$ is the label of the aspect term t. Consider the text fragment Victorian two bedroom mid terrace property from Example 1. The corresponding SAE is $\langle \{\text{Victorian}, \langle \{\text{two}\}, \text{bedroom} \rangle, \text{mid terrace}\}, \text{property} \rangle$ where the aspect terms property and bedroom are labelled as PROPERTY and BEDROOM respectively, Victorian, mid terrace, and $\langle \{\text{two}\}, \text{bedroom} \rangle$ are labelled as PROPERTY-QUALIFIER, and two is labelled as a BEDROOM-QUANTIFIER. Structured aspects are obtained by matching *structured aspect patterns* (hence SAP) against free text. An SAP mirrors the structure of a structured aspect and consists of a tuple $\langle \mathsf{P}, \mathsf{T} \rangle$, where T is either the surface form of an aspect term or a POS tag, and $\mathsf{P}=\{\mathsf{p}_1, \ldots, \mathsf{p}_m\}$ is a sequence of surface forms of a modifier (e.g., two, mid terrace), POS tags (e.g., CD, JJ), or (nested) SAPs. An SAP is matched against free text in the usual way. For symmetry w.r.t. structured aspects, T and P are called the *head* and *tail* of the SAP respectively. A possible SAP matching the structured aspect above is $\langle \{\text{Victorian}, \langle \{\text{CD}\}, \text{bedroom} \rangle, \text{JJ terrace}\}, \text{property} \rangle$.

Overview and architecture SAPs are *induced* from a homogeneous corpus of texts, i.e., with texts coming from the same domain, e.g., restaurants, products. The induction operates in four phases: (1) normalization, (2) clustering, (3) segmentation and typing, and (4) generalization. Texts are processed to obtain a corpus of orthographically normalized and POS tagged noun phrases (NP). The normalized NPs are clustered around their head nouns, being these likely candidates for aspect terms. A cPMI-based (Damani and Ghonge, 2013) segmentation of the modifiers of the NPs locates *(i)* multi-word expressions and *(ii)* nested aspects. The segmented NPs are then taken as a first approximation of an SAP (ground), where the head noun becomes the head of the SAP and the (segmented) modifier of the NP becomes its tail. Modifiers are then typed as quantifiers or qualifiers. Ground SAPs are then generalized by replacing the surface forms of the modifiers by their POS tags, thus obtaining the final SAPs.

Normalization and POS tagging The normalization tackles issues that affect subsequent phases of the induction process. In particular, it improves the accuracy of noun chunking on NUT, avoiding an expensive (and possibly pointless) NUT-specific training. The first problem is the high number of tokenization errors, leading to incorrect sentence splits. To address this, we have built a NUT-specific tokenizer, producing correct tokens in the presence of, e.g., abbreviations, units of measures, and incorrect sentence boundaries. These customizations are domain independent. Incorrect orthography is another major problem in NUT. It directly affects the performance of POS taggers and, in turn, of noun chunking. To address this, we normalize the orthography of each token to its most common orthography in the whole corpus. We consider five orthographic classes: uppercase, lowercase, upper initial, lower initial, and alphanumeric. An occurrence of a token is normalized if more than a certain percentage (experimentally set at 90% across all domains) of its other occurrences in the corpus have a different orthography.

In the first sentence of Example 2, orthographic errors lead to wrong tags for, e.g., Fantastic (CD), Beds (NNPS), and OXFORD (VBD), while the missing space after the first full stop leads to an incorrect sentence boundary. The POS tagger cannot therefore produce the correct tags for Don't. The second sentence

Example 2: Normalization

Fantastic 2 Beds OXFORD property.Don ' t miss it !
CD CD NNPS VBD NN POS NN VBP PRP PUNC

↓

{fantastic 2 beds Oxford property . } {Do n't miss it ! }
JJ CD NN NNP NN PUNC VB RB VB PRP PUNC

shows how POS tagging improves after our improved tokenization and orthographic normalization. We do not only produce more sensible POS tags, but we also recover the correct sentence boundaries.

Normalized texts are then split into sentences and POS tagged. We use a state-of-the-art rule-based sentence splitter and Hepple's POS tagger tagger (Hepple, 2000) trained on the Penn TreeBank Corpus. The tagged sentences are handed over to a rule-based NP chunker (Ramshaw and Mitchell, 1999) to produce a corpus of noun phrases.

Noun-phrase clustering NPs are clustered around their head nouns. Head nouns are stemmed and lemmatized, e.g., by normalizing plurals, to avoid over-segmentation of the clusters due to different surface forms of equivalent head nouns. The modifiers of the NPs are also normalized to prevent non-content prefixes and numerical expressions from fragmenting the clusters.

NPs are generalized to abstract forms where non-content words and numerical expressions are replaced by POS tags. A non-content word is a token with a POS tag in {CC, DT, EX, IN, PRP, PUNC}, while a numerical expression has POS tag CD. Prefixes consisting only of non-content words are then removed from the NPs. The process is illustrated in Example 3.

Example 3: Clustering

Two further double bedrooms	CD further double bedrooms	{CD further double bedrooms, [BEDROOM]
Three further double bedrooms	CD further double bedrooms	further double bedrooms,
A further double bedroom	DT further double bedroom	CD first floor bedrooms}
Two first floor bedrooms	CD first floor bedrooms	

We start with four NPs with 2 different surface forms for the head noun (i.e., bedroom and bedrooms). For three of the NPs, the modifier starts with a numerical expression. This remains part of the NP but as a POS tag (CD). The non-content word prefix A (POS tag DT), is removed from the NP. The result of this process is a single cluster headed by BEDROOM and consisting of three partially-generalized NPs. The clusters are filtered based on their cardinalities, i.e., the number of (possibly duplicated) NPs belonging to the cluster. The top-k clusters whose elements cover at least a certain percentage (experimentally set at 70% across all domains) of all the NPs are retained. Clusters can also be fragmented by spelling errors. The head nouns of discarded clusters are therefore checked for similarity against the head nouns of the retained ones. Two clusters are merged if the Dameraau-Levenshtein string edit distance is less than an experimentally set threshold (20% across all domains). If multiple merging options are possible, the one with highest similarity is chosen. If only equivalent options are available, we merge in all possible ways. Clearly, the normalization of the noun-phrase modifiers described above can affect the ranking of the noun-phrases, since clusters can be merged thus increasing their ranking.

Segmentation and typing The segmentation phase identifies multi-word expressions and hierarchical structures in NP modifiers, thus producing a first approximation of an SAP. The key tool used in the segmentation is corpus-level significant point-wise mutual information (cPMI) (Damani and Ghonge, 2013). Our definition of cPMI uses the corpus of NPs instead of arbitrary descriptions. Let $\mathcal{C}$ be the set of all clusters produced as described above. We denote by $f_{\mathcal{C}}(\mathrm{t})$ the frequency of the string t in all clusters of $\mathcal{C}$, i.e., obtained by summing up all of the occurrences of t in all clusters. Let $0 < \delta < 1$ be the normalization factor defined as in (Damani and Ghonge, 2013), and t‖w, the concatenation of two strings

t and w. We then define $\text{cPMI}_{\mathcal{C}}(\mathsf{t}, \mathsf{w})$ as follows:

$$\text{cPMI}_{\mathcal{C}}(\mathsf{t}, \mathsf{w}) = \log \frac{f_{\mathcal{C}}(\mathsf{t}\|\mathsf{w})}{f_{\mathcal{C}}(\mathsf{t}) \cdot \frac{f_{\mathcal{C}}(\mathsf{w})}{|\mathcal{C}|} + \sqrt{f_{\mathcal{C}}(\mathsf{t})} \cdot \sqrt{\frac{ln(\delta)}{(-2)}}}$$

The cPMI value is used to determine whether a token should be associated with *(i)* the head noun, *(ii)* a nested token representing the head of a different cluster, thus possibly inducing a nested structure, or *(iii)* an adjacent token, thus forming a multi-word expression.

We model the segmentation as the problem of finding a cPMI-optimal *parenthesization* (or, equivalently, a parse tree) of the NP. In order to achieve this, we first have to define the notions of *(i)* valid parenthesization and *(ii)* cPMI of a parenthesization. A valid parenthesization is a balanced parenthesization such that, each k-th level of the parenthesization: **(1)** consists of at least two elements (i.e., tokens or subpatterns), and **(2)** it either terminates with a head of cluster or it contains no heads of cluster. The cPMI of a (valid) parenthesization is the sum of the cPMI values computed between each element (token or parenthesized sub-expression) in the parenthesization and the first head of cluster following the element at the same level of nesting. If no heads of cluster are present at the same level of nesting, then the sum of the cPMI values between subsequent tokens is taken. The cPMI of the parenthesized sub-expressions is then recursively added to this value.

Example 4: Segmentation

p	:	Victorian two bedroom mid terrace property
p_i	:	(Victorian (two bedroom mid) (terrace) property)
p_v	:	(Victorian (two bedroom) (mid terrace) property)
SAP	:	$\langle\{$Victorian, $\langle\{$two$\}$, bedroom$\rangle$, mid terrace$\}$, property$\rangle$

Consider the NP p of Example 4. A balanced but invalid parenthesization is shown as p_i. The parenthesization violates both conditions given above, i.e., the level-2 parenthesization (two bedroom mid) contains a head of cluster (i.e., bedroom) but terminates with the token mid, and the level-2 parenthesization (terrace) contains a single token. A valid parenthesization is p_v and its cPMI is computed as:

$$\begin{aligned}\text{cPMI}_{\mathsf{np}} \;=\; & \text{cPMI}_{\mathcal{C}}(\text{Victorian, property}) \;+\; \text{cPMI}_{\mathcal{C}}(\text{two bedroom, property}) \;+ \\ & \text{cPMI}_{\mathcal{C}}(\text{mid terrace, property}) \;+\; \text{cPMI}_{\mathcal{C}}(\text{two, bedroom}) \;+\; \text{cPMI}_{\mathcal{C}}(\text{mid, terrace})\end{aligned}$$

Notice that $\text{cPMI}_{\mathsf{np}}$ now includes the cPMI value between mid and terrace (i.e., a multi-word expression).

Algorithm 1 formalizes the process described above. The pseudocode focuses on the most technical aspect of the procedure, i.e., the generation of the different parenthesizations. We also omit minor technicalities related to, e.g., handling of punctuation and non-content words. The algorithm receives as an input the clustered noun phrases, here represented as a map $\mathcal{C}$ having as keys the heads of clusters and as values (denoted as `val`$(\mathcal{C})$) sets of noun phrases. The output of the algorithm is again a map P, having as keys the heads of the patterns, and sets of SAPs as values. The algorithm iterates over all noun phrases np in all clusters and converts them into a sequence of tokens P (Line 4). The function `optiPar` then produces a cPMI optimal parenthesization of P (Line 5) that is then transformed into an SAP via the `par2SAP` function (Line 6). The SAP is stored in P (Line 7). The notation P`(`k`)` represents the cluster of SAPs headed by k, while `head(`P`)` returns the head of the pattern P. The function `optiPar` is an adaptation of standard combinatorial parenthesization algorithms, where we keep track of the parenthesization producing the highest noun-phrase-wide cPMI. The function takes a sequence of tokens P and recursively parenthesizes it (Lines 4-9). The $\text{cPMI}_{\mathsf{np}}$ of the parenthesization is then computed by the function `npcPMI` and tested against the current maximum (Lines 11-13). The function `isValid` in Line 10 checks that only valid parenthesizations are considered. For the sake of readability, Algorithm 1 gives a straightforward recursive implementation of the procedure but more efficient dynamic programming implementations can be provided.

In SAPs, nested patterns, e.g., $\langle\{$two$\}$, bedroom$\rangle$ are replaced with a reference to the cluster of SAPs headed by the head of the nested pattern, e.g., BEDROOM. The reason for this is to enable parallel matching

Algorithm 1: Segmentation

```
Function segmentation(C)
    input: C: the clusters
    P ← ∅;
    for C ∈ val(C)
        for distinct np ∈C
            P ← split(np);
            P_max ← optiPar(P,C);
            P_sap ← par2SAP(P_max);
        P(head(P_sap)) ← P(head(P_sap)) ∪ {P_sap};
    return P;
Function optiPar(P,C)
    input: C: the clusters
    input: P: the sequence of tokens from the noun phrase
    if |P| ≤ 2 then
        return P;
    ⟨max_pmi, P_max⟩ ← ⟨0, ∅⟩;
    for i ∈ [0, |P|-1]
        for j ∈ [|P|-1, i+1]
            if i > 0 ∨ j < |P|-1 then
                P′ ← optiPar(sub(P,0,i)) || '(' || optiPar(sub(P,i,j)) || ')' || optiPar(sub(P,j,|P|-1));
            else
                P′ ← '(' || P || ')';
            if isValid(P′) then
                cur_pmi ← npcPMI(P′,C);
                if cur_pmi > max_pmi then
                    ⟨max_pmi, P_max⟩ ← ⟨cur_pmi, P′⟩;
    return P_max;
```

of the SAP at runtime, thus avoiding matching multiple times the same fragment of text. The elements of a ground SAP are then *typed* according to their role in the pattern (Example 5). The head of the pattern is labelled as the head of the SAP cluster it belongs to and is used to match the aspect term. Modifiers with a POS tag CD and linked to an aspect term t are labelled as λ(t)-QUANTIFIER. All other modifiers linked to t, including nested SAPs, are labelled as λ(t)-QUALIFIER. Extractions produced by matching SAP against free text are labelled according to the typing of the SAP.

Example 5: Typing

⟨{Victorian, ⟨{two}, bedroom⟩, mid terrace}, property⟩

property	→	PROPERTY	bedroom	→	BEDROOM
Victorian	→	PROPERTY-QUALIFIER	two	→	BEDROOM-QUANTIFIER
mid terrace	→	PROPERTY-QUALIFIER			
two bedroom	→	PROPERTY-QUALIFIER			

Generalization Ground SAPs are limited to what has been directly observed in the corpus. As a consequence, the elements of an SAP are generalized to their POS tags to increase recall. The process must be controlled because excessive generalization may also lead to inaccurate extractions. The generalization rules are illustrated in Example 6 and operate as follows: (1) Numerical expressions, non-content words, and punctuation are always generalized to their POS tags. (2) Multi-word expressions are generalized as both n-grams and unigrams since they "behave" like a single token. In the latter case, the POS tag of the last token is used. (3) Aspect terms are not generalized unless the tail of the pattern contains a nested SAP with a ground head. The nature of the modifiers is often strictly related to the aspect term they refer to. This is obvious for, e.g., the modifiers of bedroom and property in Example 6. We therefore have to prevent undesired associations between aspect terms and incompatible modifiers. (4) Qualifiers are also semantically related to other qualifiers at the same level of nesting. As a consequence, we generalize at most one qualifier per level of nesting to maintain this connection. Clearly, this process may give rise to more than one generalization for the same SAP. The ground pattern is also preserved.

Example 6: Generalization

⟨{Victorian, ⟨{two}, bedroom⟩, mid terrace}, property⟩

↓

⟨{JJ, ⟨{CD}, bedroom⟩, mid terrace}, property⟩
⟨{Victorian, ⟨{CD}, bedroom⟩, JJ}, property⟩
⟨{Victorian, ⟨{CD}, bedroom⟩, JJ terrace}, property⟩
⟨{Victorian, ⟨{CD}, bedroom⟩, mid JJ}, property⟩
⟨{Victorian, ⟨{CD}, bedroom⟩, mid terrace}, NN⟩

Patterns are scored based on their ability to discriminate between correct and incorrect extractions. We adapt the scoring mechanism of (Gupta and Manning, 2014), where the quality of a pattern is estimated by matching the extracted patterns against a manually labelled validation set. In our unsupervised setting, no labelled dataset is available. We take the heads of the noun-phrase clusters as a surrogate of the set of valid aspects. The analysis is limited to aspect terms. Let T be the set of valid aspect terms as defined above, and E be the set of aspect terms produced by an SAP P. The score of P is computed as:

$$\nu(\mathsf{P}) = \frac{|T|}{\sum_{e \in E}(1 - \max_{t \in T} \mathbb{1}(\frac{dist(t,e)}{len(t)} < 0.2))} \cdot \log |T| \qquad \nu(\mathsf{P}) \in [0, \infty]$$

where $dist(t, e)$ denotes the Dameraau-Levenshtein edit distance between two strings t and e and $len(\cdot)$ denotes the length of the string. Patterns scoring less than an experimentally set threshold (50% across all domains) are eliminated.

3 Evaluation

Our method (Oxtractor) is implemented in Java. All experiments are run on a quad-core desktop machine at 3.40GHz and 32GB RAM, running Linux. All resources used in the evaluation are made available for replicability at http://bit.ly/2dewzdd and include: the SAED dataset and GS, our reimplementations of IIITH and ATL, a compiled version of Oxtractor, and all output files generated by all systems.

Datasets and metrics We use three groups of datasets in our evaluation (Table 1): The first two consist of the SemEval14 and SemEval15[2] datasets used for the aspect term extraction (ATE) and opinion target expression (OTE) subtasks of the aspect-based sentiment analysis (ABSA) task. The datasets provide laptops, restaurants and hotel reviews with associated gold standard (GS) annotations. The hotel domain is meant to be used in a completely unsupervised setting and therefore no training data is available. The size of SemEval14 in Table 1 is expressed in number of sentences instead of number of texts since this information is unavailable. We complement the SemEval15 datasets with some specifically designed for SAE (SAED). We provide texts from six domains (50% for testing, 50% for validation). Four of them, i.e., chairs, real estate, shoes, and watches, describe products. The Amazon texts come from the Stanford's Snap Lab's web data corpus (McAuley and Leskovec, 2013). The two remaining domains, i.e., hotels and restaurants, can be classified as services. These descriptions are still feature intensive but, differently from the products, the features are loosely connected to the main entity, i.e., locations, services/facilities offered. The dataset consists of both NUT and (semi-) formal English texts. We provide GS annotations for 150 texts equally distributed across the six domains. The GS provides and average of 355 aspect terms, 30 quantifiers, 430 qualifiers, and 45 nested aspects per domain. Annotations were produced by 6 independent annotators (λ=87%). We use standard recall, precision, and F_1 score metrics. However, due to the different granularity of the output produced by the systems and of the GS annotations, the definition of a correct extraction varies slightly with each evaluation task.

Comparative evaluation – Simplified SAE The method by (Kim et al., 2012), hence ATL, is currently the closest to SAE we are aware of. We have obtained from the authors the dataset used in their evaluation but not an implementation of the system. We have reimplemented the method and successfully reproduced the experimental results described in the original paper. Figure 1 shows a comparison between ATL and Oxtractor on the SAED dataset. An extraction is correct if modifiers and aspect terms match exactly the GS annotations, and if modifiers are correctly typed as qualifiers or quantifiers. This is a simplified

Table 1: Datasets

DATASET	DOMAIN	SIZE (#texts)	SOURCES	CATEGORY	FORMALITY	TYPE
SemEval14	restaurants	3k + 800 GS (*)	Citysearch	service	NUT	evaluative
	laptops	3k + 800 GS (*)	N/A	product	NUT	evaluative
SemEval15	restaurants	254 + 96 GS	Citysearch	service	NUT	evaluative
	hotels	N/A + 30 GS	Citysearch	service	NUT	evaluative
SAED	chairs	94k + 25 GS	Amazon, GumTree	product	NUT	descriptive
	hotels	20k + 25 GS	TripAdvisor	service	formal	descriptive
	real estate	87k + 25 GS	RightMove	product	semi-formal	descriptive
	restaurants	115k + 25 GS	TripAdvisor	service	formal	descriptive
	shoes	46k + 25 GS	Amazon, GumTree	product	NUT	descriptive
	watches	10k + 25 GS	Amazon, GumTree	product	NUT	descriptive

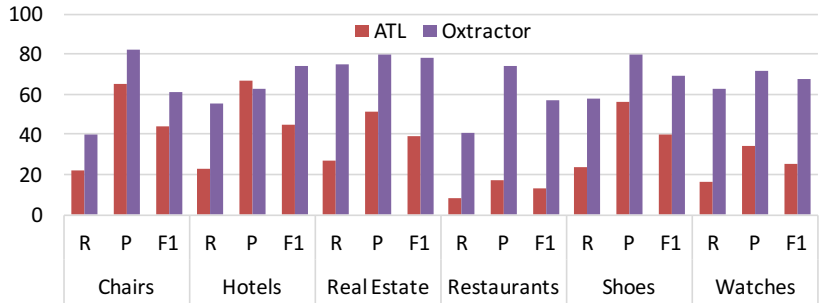
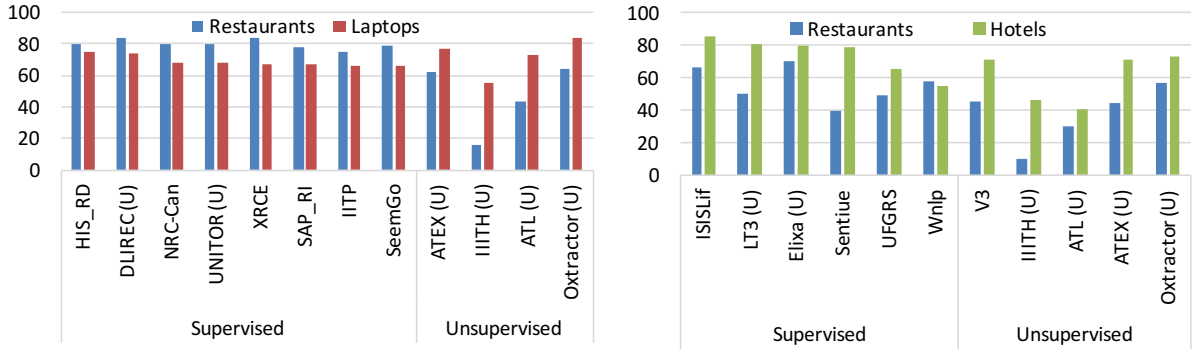


Figure 1: Oxtractor vs. ATL on simplified SAE (SAED dataset)

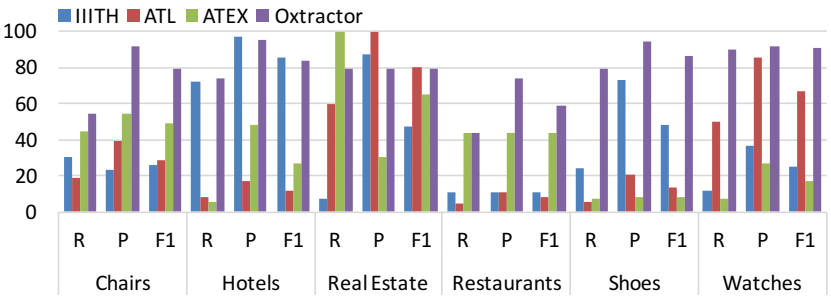
SAE setting where we do not require correct linking of modifiers to aspect terms. Oxtractor performs 33% better than ATL in average, outperforming it in all domains. Besides being unable to extract hierarchical structures, a visible issue in ATL is the inability to recognize semantic connections between modifiers and aspect terms. This leads to a number of incorrect extractions for both aspect terms and modifiers that could be avoided by leveraging, e.g., statistical co-occurrence or cPMI.

Comparative evaluation – ATE Restricting the evaluation to aspect terms makes it possible to compare Oxtractor against other ATE systems. We denote by IIITH and ATEX the methods proposed by (Raju et al., 2009) and (Zhang and Liu, 2014) respectively. IIITH uses unsupervised clustering of noun-phrases to derive aspect terms and is therefore similar to Oxtractor. We could not obtain the original IIITH system from the authors so the evaluation relies on our own implementation. ATEX, on the other end, is chosen because of its ATE method based on topic modeling. Moreover, it is freely available for testing. An aspect term is correctly extracted if it matches exactly a GS annotation. For Oxtractor, IIITH, and ATL we used the SAED corpora for training. Figures 2a and 2b show the results for the SemEval14 and SemEval15 datasets respectively. For all systems, except Oxtractor, IIITH, ATEX, and ATL, we report the scores provided in the corresponding SemEval papers. The symbol (U) denotes systems that have used additional data besides the training set provided by SemEval (i.e., *unconstrained* in SemEval terminology). Oxtractor outperforms all unsupervised systems and some of the supervised ones. Moreover, this is a lower bound for Oxtractor due to a difference between the granularity of the SemEval GS and the output produced by Oxtractor. E.g., Egyptian restaurant is considered a correct aspect term by SemEval but Oxtractor would only produce restaurant as the aspect term and Egyptian as its modifier (and thus a miss for SemEval). A striking result is the performance achieved by Oxtractor on the laptops domain in SemEval14, where also all supervised systems are outperformed. As for many other product-like domains, aspect terms in the laptops domain frequently fall within the scope of noun-phrases that are easily processed by our method. This is much less true for other service-like domains such as, e.g., restaurants and hotels. Figure 2c shows the performance of Oxtractor, IIITH, ATEX, and ATL on the SAED dataset. In this case, the GS differentiates between aspect terms and modifiers, thus explaining the lower performance of traditional ATE systems, e.g., IIITH and ATEX, and the higher accuracy of, e.g., ATL that is able to appreciate this difference. ATEX

[2]http://alt.qcri.org/semeval2014/task4/ and http://alt.qcri.org/semeval2015/task12/



(a) SemEval14 Dataset (b) SemEval15 Dataset



(c) SAED Dataset

Figure 2: Oxtractor vs. others in ATE

also struggles on restaurants due to long sentences.

Full SAE We evaluate the performance of Oxtractor in the full SAE setting using the SAED dataset. An extraction is correct if aspect terms and modifiers match the GS annotations, including the correct (and possibly hierarchical) associations between modifiers and entities. In average (Figure 3), Oxtractor achieves an F_1 of 58.6% in the full SAE setting, sensibly below the one obtained in the ATE (i.e., 79.6%) and simplified SAE (i.e., 67.8%) settings. Linking modifiers to aspect terms in the presence of hierarchical structures is indeed a much more challenging task than simply identifying them. Another interesting aspect is the impact of the generalization on the performance. Generalized SAPs produce 444 correct extractions against the 386 of the ground ones (+15%).

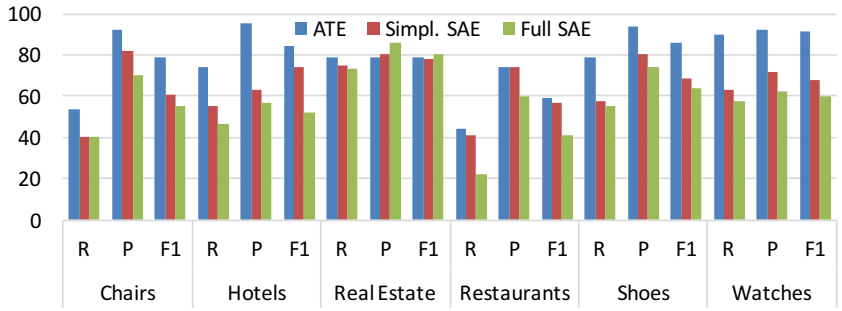
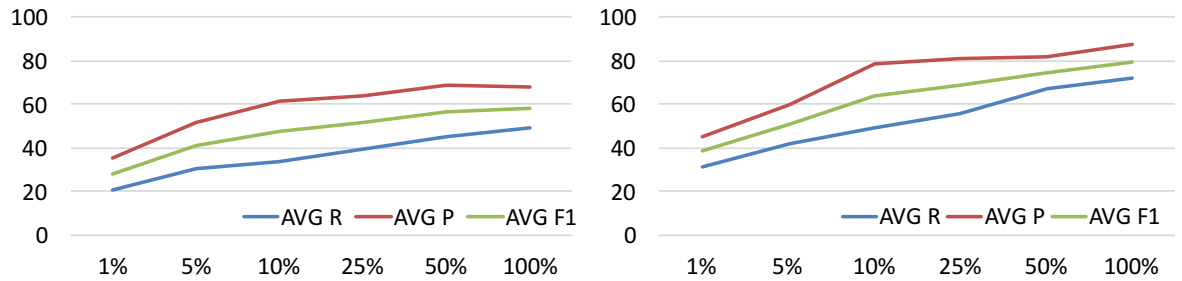


Figure 3: Oxtractor on full SAE

Corpus size One obvious question is how dependent our method is on the size of the training corpus. To measure this, we evaluated Oxtractor by inducing SAPs from increasing subsets of the original corpora corresponding to fractions of 1%, 5%, 10%, 25%, and 50% of their original size. Figure 4 shows the effect of the corpus size on the performance of Oxtractor in both the SAE and ATE settings. Clearly, larger corpora lead to better results in both settings. However, two interesting facts have been observed. Long-

tail aspects are only induced from sufficiently large corpora (thus the behavior between 10% and 50%). Larger corpora also have the disadvantages that sufficiently frequent but incorrect tokens can end up being extracted as modifiers. In other words, the increase in recall is not matched by a comparable increase of precision (Figure 4b). In the case of SAE we even observe a slight drop in precision (Figure 4a).



(a) SAE Task (b) ATE Task

Figure 4: Performance vs. corpus size (average – SAED dataset)

A breakdown of the data per domain allows us to draw further conclusions on the relationship between the size of the corpus and the performance of the SAPs. There is a relationship between the variety of features and the amount of data that is necessary to induce good quality SAPs. For domains such as, e.g., chairs, realestate, shoes, and watches, starting from 25% of the size of the corpus we do not notice substantial improvements in performance. This can be explained by the nature of the features in these domains that are intrinsically limited, e.g., make and models of the products, types of real estate properties, etc. In the restaurants and hotel domains the texts are much more variegated in features, e.g., restaurant and hotel names, dishes, locations, etc. Despite the large amount of texts available, it seems that our method would require even larger corpora before being able to converge to a stable set of aspects.

Efficiency Finally, we evaluate the efficiency of the SAP induction and matching phases. Oxtractor's efficiency mostly depends on the length of the sentences, due, e.g., to the morphological analysis, our cPMI-based segmentation, and pattern matching. Oxtractor induces SAPs at a rate of 14 $^{ms}/_{sent}$ and 6 $^{ms}/_{sent}$ for *long* (i.e., $\geq$10 tokens) and *short* (i.e., $<$10 tokens) sentences respectively. The matching time per sentence is almost negligible and ranges between $2ms$ and $3ms$ per text. We also notice a linear correlation between the size of the SAP and its matching time. This is achieved, despite the presence of hierarchical structures, by replacing nested patterns with references to the corresponding SAP clusters, enabling parallel matching of the nested SAPs. In terms of training time, Oxtractor induces patterns from $20k$ texts within $1hr$ on average. IIITH and ATL require more than $15hrs$ on the same dataset.

Discussion SAE is still in its infancy. A number of interesting problems can be studied in this area such as, e.g., semantic categorization of modifiers and aspect terms. Although the majority of structured aspects appear in noun phrases, a fair amount also appears in more complex syntactic structures. We extended our normalization to rewrite these structures into traditional noun-phrases with good results. Another issue is redundant SAPs, caused by aspect terms having the same "tail" and semantically belonging to a taxonomic hierarchy of concepts, e.g., two bedroom apartment, two bedroom house. We are currently investigating the use of knowledge bases such as, e.g., BabelNet (Navigli and Ponzetto, 2012) to reduce this redundancy. Finally, our evaluation shows that the gap between Oxtractor and supervised systems is still considerable. Oxtractor can be adapted to use supervision at different stages of the induction process, in particular during clustering and pattern scoring.

Acknowledgements

The research leading to these results has received funding from the EPSRC Programme Grant VADA, no. EP/M025268/1, and the ERC Grant ExtraLytics, no. ERC-2014-PoC. Oxtractor is not related to any of the commercial products currently offered by Meltwater.

References

Justine Cassell. 2000. Embodied conversational interface agents. *Commun. ACM*, 43(4):70–78.

Zhiyuan Chen, Arjun Mukherjee, and Bing Liu. 2014. Aspect extraction with automated prior knowledge learning. In *Proc. of ACL*, pages 347–358.

Yejin Choi and Claire Cardie. 2010. Hierarchical sequential learning for extracting opinions and their attributes. In *Proc. of ACL*, pages 269–274.

Om P Damani and Shweta Ghonge. 2013. Appropriately incorporating statistical significance in pmi. In *Proc. of EMNLP*, pages 163–169.

R. Ghani, K. Probst, Y. Liu, M. Krema, and A. Fano. 2006. Text mining for product attribute extraction. *SIGKDD Explorations*, 8(1):41–48.

Sonal Gupta and Christopher D Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In *CoNLL*, pages 98–108.

M. Hepple. 2000. Independence and commitment: Assumptions for rapid training and execution of rule-based pos taggers. In *Proc. of ACL*, pages 278–277.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proc. of SIGKDD*, pages 168–177.

Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single and cross-domain setting with conditional random fields. In *Proc. of EMNLP*, pages 1035–1045.

Anitha Kannan, Inmar E Givoni, Rakesh Agrawal, and Ariel Fuxman. 2011. Matching unstructured product offers to structured product specifications. In *Proc. of SIGKDD*, pages 404–412.

C. Kelly, B. Devereux, and A. Korhonen. 2012. Semi-supervised learning for automatic conceptual property extraction. In *Proc. of CMCL*, pages 11–20.

D. S. Kim, K. Verma, and P. Z. Yeh. 2012. Building a lightweight semantic model for unsupervised information extraction on short listings. In *Proc. of EMLNP*, pages 1081–1092.

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In *Proc. of ACL*, pages 653–661.

Kang Liu, Liheng Xu, and Jun Zhao. 2013. Syntactic patterns versus word alignment: Extracting opinion targets from online reviews. In *Proc. of ACL*, pages 1754–1763.

J. McAuley and J. Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proc. of RecSys*, pages 165–172.

R. Navigli and S. P. Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193:217–250.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proc. of HLT-EMNLP*, pages 339–346.

Soujanya Poria, Erik Cambria, Lun-Wei Ku, Chen Gui, and Alexander Gelbukh. 2014. A rule-based approach to aspect extraction from product reviews. In *Proc. of COLING*, pages 28–37.

K. Probst, R. Ghani, M. Krema, A. E. Fano, and Y. Liu. 2007. Semi-supervised learning of attribute-value pairs from product descriptions. In *Proc. of IJCAI*, pages 2838–2843.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27.

S. Raju, P. Pingali, and V. Varma. 2009. An unsupervised approach to product attribute extraction. In *Proc. of ECIR*, pages 796–800.

L. A. Ramshaw and M. P. Mitchell. 1999. Text chunking using transformation-based learning. In Armstrong S. et Al, editor, *Natural Language Processing Using Very Large Corpora*, volume 11 of *Text, Speech and Language Technology*, pages 157–176. Springer.

Christina Sauper and Regina Barzilay. 2013. Automatic aggregation by joint modeling of aspects and values. *JAIR*, 46(1):89–127.

Jaeho Shin, Sen Wu, Feiran Wang, Christopher De Sa, Ce Zhang, and Christopher Ré. 2015. Incremental knowledge base construction using DeepDive. *PVLDB*, 8(11):1310–1321.

Ivan Titov and Ryan T McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proc. of ACL*, volume 8, pages 308–316.

Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *Proc. of EMNLP*, pages 1533–1541.

Mohamed Yahya, Steven Whang, Rahul Gupta, and Alon Y Halevy. 2014. Renoun: Fact extraction for nominal attributes. In *Proc. of EMNLP*, pages 325–335.

Bishan Yang and Claire Cardie. 2014. Joint modeling of opinion expression extraction and attribute classification. *TACL*, 2:505–516.

Jianxing Yu, Zheng-Jun Zha, Meng Wang, and Tat-Seng Chua. 2011. Aspect ranking: identifying important product aspects from online consumer reviews. In *Proc. of HLT*, pages 1496–1505.

Daniel Zeng, Hsinchun Chen, Robert Lusch, and Shu-Hsing Li. 2010. Social media analytics and intelligence. *IEEE Intell. Syst.*, 25(6):13–16.

Lei Zhang and Bing Liu, 2014. *Aspect and Entity Extraction for Opinion Mining*, pages 1–40. Springer Berlin Heidelberg.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2013. Collective opinion target extraction in chinese microblogs. In *Proc. of EMNLP*, pages 1840–1850.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *Proc. of CIKM*, pages 43–50.

Robust Text Classification for Sparsely Labelled Data Using Multi-level Embeddings

Simon Baker [1,2] **Douwe Kiela** [1] **Anna Korhonen** [2]
[1]Computer Laboratory, 15 JJ Thomson Avenue
[2] Language Technology Lab, DTAL
University of Cambridge, UK
{sb895|dk427|alk23}@cam.ac.uk

Abstract

The conventional solution for handling sparsely labelled data is extensive feature engineering. This is time consuming and task and domain specific. We present a novel approach for learning embedded features that aims to alleviate this problem. Our approach jointly learns embeddings at different levels of granularity (word, sentence and document) along with the class labels. The intuition is that topic semantics represented by embeddings at multiple levels results in better classification. We evaluate this approach in unsupervised and semi-supervised settings on two sparsely labelled classification tasks, outperforming the handcrafted models and several embedding baselines.

1 Introduction

The objective of text classification is to label a scope of text according to predefined labels. While general domains tend to have sufficient amounts of labelled data, in specialised domains (e.g., scientific literature) such data are often scarce and labelled instances number in the hundreds, or low thousands at most. Such domains may also require highly specialised annotators, making labelled data expensive and difficult to obtain (Simpson and Demner-Fushman, 2012).

In order to mitigate the data sparsity problem, a lot of handcrafting is needed to engineer features specific to the task and domain. Typically this process involves a long NLP pipeline, e.g., POS-tagging, parsing, named entity recognition, semantic role labelling, feature selection, etc. Consequently, approaches based on handcrafting can be prohibitively time consuming, and since the resultant features are domain dependent, these systems are difficult to port to other domains (Sebastiani, 2002; Dai et al., 2007). While unsupervised and lightly-supervised methods can bypass the need for labelled data, they in turn tend to suffer from lower performance (Zhang and Elhadad, 2013; Quan et al., 2014; Aggarwal and Zhai, 2012).

In this paper, we present a novel approach to text classification that is especially beneficial in situations were labelled datasets are small. Our approach builds on the Distributed Memory (DM) model by Le and Mikolov (2014). The fast and simple unsupervised DM model acquires paragraph level embeddings. We improve on the model so that we jointly learn multi-level embeddings that encode class-label topical information in addition to text.

We jointly learn a model that captures embedding representation for the target class labels, as well as word-, sentence- and document-level representations in the same space. From these multi-level embeddings we derive a set of features. Our approach requires no manual feature engineering, can cope with small amounts of labelled data and produces features that are more robust to domain variation and portable across domains.

At the document-level, the overall "topic" is a mixture of the sub-topics of paragraphs in that document. The topics of the paragraphs are in turn mixtures of the sentence topics, all the way down to

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2333–2343, Osaka, Japan, December 11-17 2016.

word-level semantics. Our multi-level embeddings model captures this intuition elegantly; for example, an article about *cars* might have the first sentence discussing *car manufacturing*, followed by another discussing *car safety*, etc. Each of these topics can be represented by sentence-level embeddings, while a document-level embedding can capture the overall topic of the article.

We show that classifying text based on such multi-level semantics achieves superior performance both against very specialised handcrafted models and using word sentence or document embeddings alone. We demonstrate the effectiveness of our methodology on two real-world sparsely-labelled tasks: classification of biomedical text by (i) semantic categories, and (ii) rhetorical structure.

We apply our approach at two different levels of granularity: at document-level and at sentence-level. At the sentence-level, labelled data and contexts are even more sparse. In both cases, we compare our approach under a supervised setting against a handcrafted method and show that it rivals and in some cases clearly outperforms such methods. In addition, we compare against classifiers trained using standard embedding features and show that our approach outperforms them by a large margin. We also show that fast semi-supervised classification using our multi-level embedding features achieves promising results, even when compared against an SVM classifier using standard embeddings.

To our knowledge, this is the first work to introduce multi-level embeddings for text classification and to show their superior performance against handcrafted approaches and their robustness across domains which suffer from scarcity of labelled data.

2 Related Work

Embedded distributed representations have been used widely for document and sentence classification. For example, Huang et al. (2014) learn document-level embeddings using word-level embeddings as input. Yan et al. (2015) learn document-embeddings by combing a Deep Boltzmann Machine and a Deep Belief Network. Bhatia et al. (2015) learn embeddings for large multi-label classification in situations where the label set is extremely large. Liu et al. (2015) use latent topic models to learn a topic from each word, and then learn an embedding based on both the topic and the word. Yogatama and Smith (2014) use structured regularizers based on parse trees, topics, and hierarchical word clusters, as well as hierarchical sparse coding for regularization using stochastic proximal methods (Yogatama et al., 2015).

All of these works have been trained and evaluated on general domains such as newswire rather than on sparse domains with small labelled datasets.

There are works that target small labelled data text classification in sparse domains using techniques such as active learning (Guo et al., 2013; Figueroa et al., 2012; Nissim et al., 2015). The idea of active learning is to reduce annotation effort by iteratively selecting the most informative instances to be labelled by interactively querying an expert. Although good accuracy can be achieved, the approach relies on expert knowledge and interaction, and may still require feature engineering.

Other works tackle the sparsity of labelled data using distant supervision (Reschke et al., 2014; Vivaldi and Rodríguez, 2015). Here, a classifier is trained using data labelled automatically using approximate heuristics rather than annotators. However, due to the assumptions and bias that are inherent in such labelling heuristics, this may result in lower performance.

The work presented in this paper differs from the above as it focuses on learning embeddings for sparse domains with small labelled datasets; moreover, we focus on utilizing these embeddings specifically for text classification.

3 Approach

This section first describes the Distributed Memory model (Section 3.1), and then explains how we improved it for sparse domain text classification by introducing jointly learned multi-level representations (Section 3.2).

In Section 3.3 we describe three types of features that we extract from such representations, and in Section 3.4 we explain the fixed classification setup for our task-based evaluations.

3.1 The Distributed Memory model

The Distributed Memory model is an extension of the Continuous Bag of Words (CBoW) model of Mikolov et al. (2013). The DM model learns a representation of a paragraph that captures the semantics of a paragraph's "topic". In the model, every word is represented in a word embedding matrix, and every paragraph in a paragraph embedding matrix. Paragraph representations are averaged or concatenated to predict the next word in a context using a hierarchical softmax classifier.

DM introduces an additional component to the model that allows a representation of the paragraph (via paragraph ID), which is treated internally like any other word in the model's vocabulary. It acts as a memory that remembers what is missing from the current context. The model learns a vector representation of the paragraph that captures its overall topic semantics via stochastic gradient decent.

3.2 Joint learning of multi-level embeddings

We improve DM by learning distributed representations that capture the topical information at varying levels of granularity, that is, we learn embeddings at a word-, sentence- (or paragraph-), and document-level. We also learn a distributed representation of the class labels, since these can be viewed as another level of abstraction that is more abstract than the document-level.

Our intuition is that jointly learning representations at different levels of granularity (including that of class label) provides us with better embeddings for text classification than learning a representation at each level separately. Each level captures different topic semantics, ranging from word-level to the class label. Figure 1 illustrates our model.

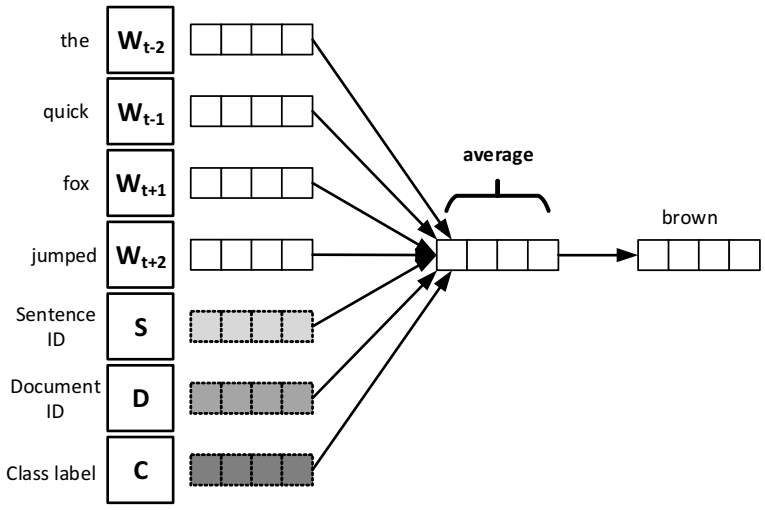


Figure 1: Illustration of distributed joint learning of different granularities of text contexts: words (W), sentences (S), documents (D) and classes (C). The model predicts the target word (w_t) based on the semantics captured by all these contexts. Shades represent level of abstraction/granularity.

In Figure 1, words from word embedding matrix **W**, sentences from sentence embedding matrix **S**, documents from document embedding matrix **D** and class-labels from class embedding matrix **C** are used as the context from which to predict the target word. That is, given a sequence of training words $w_1, w_2, w_3, ..., w_T$ that belongs to sentence s_t in document d_t, which has also a set of classification labels associated $c_1, ...c_m$. The objective of the model is to maximise the average log probability:

$$\frac{1}{T}\sum_{t=k}^{T-k} \log\ p(w_t|w_{t-k}, ...w_{t+k}, s_t, d_t, c_1, ...c_m) \tag{1}$$

We use a softmax output layer to obtain the probability of the target word given its context:

$$p(w_t|w_{t-k}, ...w_{t+k}, s_t, d_t, c_1, ...c_m) = \frac{e^{\vec{y}_{w_t}}}{\sum_i e^{\vec{y}_i}} \tag{2}$$

where each y_{w_t} is calculated as:

$$\vec{y}_{w_t} = \mathbf{U}\frac{\sum_{i=-k}^{k}\vec{w}_{t+i} + \vec{s}_t + \vec{d}_t + \sum_{i=1}^{m}\vec{c}_i}{2k+m+2} + b \tag{3}$$

where $k \neq 0$, $\mathbf{U}$ is the weight matrix, b is the bias, and we average the word vectors extracted from $\mathbf{W}$, the sentence vectors extracted from $\mathbf{S}$, similarly, the document vectors from $\mathbf{D}$ and class label vectors from $\mathbf{C}$.

3.3 Extracting features

We extract three types of features from the jointly-learned multi-level representations: the sentence or document embeddings (EMBED), the distances between word embeddings (WORD-DIST) and the similarities between classes (CLASS-SIM).

Embedding features: since embeddings are learned at different levels, when classifying at the document-level, we use the document-level embeddings. Likewise for sentence-level classification, we use only the sentence-level embeddings. Word-level embeddings are only used as part of extracting distance features.

Word distance features: We measure the cosine similarity between each unique non-stop word embedding occurring in the input sentence or document with the embedding representation for a given class label, i.e., $\delta_{w_i}^{c_i} = \cos(\vec{w}_i, \vec{c}_i)$, where $\vec{w}_i$ is embedding for word w_i in the input text, and $\vec{c}_i$ is the embedding representation of a class label that has been jointly learned from the training data. Since the input text has variable length, we represent these distances in sparse vector format using a dictionary of all non-stop words in the corpus labelled with the given class c_i; i.e., a "bag of word distances".

Class-similarity features: Word distance measures capture the similarity between words and class labels, but not between phrases or sentences. For this, we use word-level embeddings to measure the semantic similarity between a class and target text (sentence or document) using the so-called Earth Mover's Distance (EMD)[1], or the energy distance of moving a distribution.

EMD has been used successfully in image retrieval (Rubner et al., 2000), document topic similarity (Wan, 2007) and more recently in combination with word embeddings (Kusner et al., 2015). This method is useful for estimating the similarity between text with varying word count and overlap: the sentence *"sipping a cup of tea"*, for example, should have a relatively small EMD compared to *"wine tasting"*, despite them having no overlap and being of different length. Kusner et al. (2015) formulate the EMD problem as a linear program that can be expressed as the following optimisation:

$$\text{emd}(d, d') = \min_{\mathbf{T}\geq 0}\sum_{i,j=1}^{n}\mathbf{T}_{ij}||\vec{x}_i - \vec{x}_j||_2 \tag{4}$$

subject to the following flow constraints: $\sum_{j=1}^{n}\mathbf{T}_{ij} = \vec{d}_i$ and $\sum_{i=1}^{n}\mathbf{T}_{ij} = \vec{d'}_j$. Here, $\mathbf{T} \in \mathbb{R}^{n\times n}$ is a flow matrix, i.e., $\mathbf{T}_{ij}$ denotes how much of word i in the source document d travels to word j in the destination document d', and $\vec{x}_i$, $\vec{x}_j$ are embeddings for words i and j. Class-similarity features are obtained by finding the minimal distance between a given input (either document or sentence) and the given class, where only the most discriminatory word embeddings for the given class are combined, i.e., non-discriminatory words that occur in all classes are excluded[2]. We then use Equation 4 to measure the similarity between words occurring in the text and the class combined word list.

3.4 Supervised classification

We apply a fixed classification setup in order to compare our new method against several embedding baselines as well as handcrafted classification. We use Support Vector Machines with a linear kernel; implemented using scikit-learn (Pedregosa et al., 2011), and perform a standard grid search for kernel regularization parameter selection.

[1] Also known as the Wasserstein metric.

[2] We discarded all words that occur in more than 80% of all class contexts as non-discriminatory in the training set.

We use L1 and L2 normalization of input features, weighted equally according to the three aforementioned types (embedding, class-similarity and word-distance), i.e., features within each type are normalised separately and then combined.

We perform a 4-fold cross-validation setup and 5-fold nested cross-validation for kernel parameter tuning (using grid search); i.e., we do a 5-fold cross-validation grid search nested in each of the outer four folds.

3.4.1 Semi-supervised classification

In a semi-supervised setting, we use vast amounts of unlabelled data, i.e., documents/sentences unlabelled with any class information, and a much smaller amount of labelled documents/sentences. Instead of using a supervised classifier to learn the decision boundaries, we use the distance measurements and a tuned cut-off threshold for each class to determine class assignment.

We use WORD-DIST and CLASS-SIM (described in Section 3.3), and EMB-DIST: the cosine distance between the embedding of a sentence or document and an embedding of a class label. A cut-off threshold is used to determine positive or negative classification for each class. Under the WORD-DIST setup, we average all of the word distances. We perform a grid search for this threshold on 10% held-out data.

4 Task 1: Semantic text classification

We apply our methodology to a real-life biomedical text classification task. The aim of this task is to classify text at both document- and sentence-levels according to the Hallmarks of Cancer (HoC), a widely-employed framework in cancer research that was first introduced by Hanahan and Weinberg (2000). Motivated by the fact that cancer involves both genetic and epigenetic alterations (Marusyk et al., 2012), this framework provides an organizing principle to simplify the complexity of cancer biological processes (Baker et al., 2016).

4.1 Data

Baker et al. (2016) acquired a collection of PubMed abstracts using a set of search terms representative for each of the 10 hallmarks. The terms and their synonyms appearing in Hanahan and Weinberg (2000) and Hanahan and Weinberg (2011) were employed along with additional ones selected by a team of cancer researchers. Annotation was conducted by experts in cancer research, using the annotation tool described in Guo et al. (2012). Annotations are assigned at a sentence-level: a sentence is annotated if contains clear evidence relating to one or several hallmarks (Baker et al., 2016). Table 4.1 shows the distribution of 1,580 abstracts and sentences for each of the hallmark categories. The inter-annotator agreement is $k = 0.81$.

Hallmark	PS	GS	CD	RI	A	IM	GI	PI	CE	ID
# Abstracts	462	242	430	115	143	291	333	240	105	108
# Sentences	993	468	883	295	357	667	771	520	213	226

Table 1: Distribution of data for the ten hallmarks.

4.2 Handcrafted supervised model

We employ a fully supervised handcrafted baseline for this task, classifying using binary classifiers for each hallmark category. Sentences are first tokenised and part-of-speech tagged using the C&C tagger (Clark, 2002) trained on biomedical texts. The text is lemmatised using BioLemmatizer (Liu et al., 2012) and grammatical relations are extracted using the C&C Parser. The parser was trained using molecular biology annotations (Rimell and Clark, 2009). Finally, named entities are extracted from parsed data using ABNER (Settles, 2005), trained on the NLPBA and BioCreative corpora (Leitner et al., 2010).

We experimented with several types of handcrafted features for hallmark classification, chosen based on their inclusion in other state-of-the-art biomedical text classification systems. Only the first five are used for sentence-level classification, since the last two are only available at the document-level:

Lemmatised Bag of Words: the simplest feature employs all words occurring in input texts. We lemmatise the words in order to reduce sparsity.

Noun bigrams: Noun bigrams are used because they can be useful in capturing two word-concepts in texts (e.g., *Gene silencing*).

Grammatical relations: we use the *dobj* (direct object), *ncsubj* (non-clausal subject), and *iobj* (indirect object) relations, plus the head and dependent words in relations.

Verb classes: verb classes group semantically similar verbs together, abstracting away from individual words when faced with data sparsity. We used the hierarchical classification of 399 verbs by Sun and Korhonen (2009).

Named entities: domain-specific concepts, providing another way to group bags of words into meaningful categories. We use five types which are particularly relevant for cancer research: Proteins, DNA, RNA, Cell Line, and Cell Type.

Medical Subject Headings (MeSH): is a comprehensive controlled vocabulary for indexing journal articles and books in the life sciences. Most abstracts in our dataset contain an associated list of MeSH terms which we employ as features.

Chemicals list: a total of 3,021 associated chemicals (manually annotated). We use these as features, since processes involved with hallmarks might involve similar chemicals.

5 Task 2: Rhetorical text classification

Rhetorical text classification (also known as information structure analysis) segments scientific text into information categories. One such classification technique is argumentative zoning (Teufel and Moens, 2002) which captures the rhetorical progression of the scientific argument by segmenting a document into several zones, such as: "Objective", "Background", "Method", "Result", and "Conclusion".

This task differs from Task 1 in that the objective is to classify scientific text according to generic labels (i.e., unrelated to domain-specific knowledge) and the focus is on a different classification features, such as the position of the text in the document and the author's writing style. For example, the "Objective" zone of the argument generally appears very early in the article using an active voice.

5.1 Data

We evaluate using an expert-annotated dataset from (Guo et al., 2010) comprising of 1000 PubMed abstracts relevant to cancer biology. The dataset consists of 7985 labelled sentences, with an inter-annotator agreement of $k = 0.85$. There are five mutually non-exclusive classes, described together with their frequencies in Table 5.1.

Class	Description	# Abstracts	# Sentences
Objective (*OBJ*)	The background and the aim of the research	744	812
Background (*BKG*)	The circumstances pertaining to the current work	692	1517
Method (*METH*)	The way to achieve the goal	640	1617
Result (*RES*)	The principal findings	889	4028
Conclusion (*CON*)	Analysis, discussion and the main conclusions	859	1484

Table 2: Description of argumentative zones and their distribution in the annotated data.

5.2 Handcrafted supervised model

Many of the features used for this task are similar to those used in the Task 1, namely Bag-of-Words, Bigrams, Grammatical Relations. Here we also include Part-of-Speech tags, and the following task-specific features:

Location: categories tend to appear in typical positions in a document, e.g., *BKG* usually occurs at the beginning and *CON* at the end. The abstract is divided into ten equal parts and the location of a sentence is defined by the parts where the sentence begins and ends.

History: the category of the preceding sentence is used as a feature. This is because certain categories tend to appear before others. For example, *RES* tends to be followed by *CON* rather than other categories.

Voice: there is a correlation in scientific writing between the active and passive voice and certain categories, for example, passive voice is more frequent in *METH*.

6 Results

We now present the results of our experiments, where we compare our method to the handcrafted models, in addition to several baselines detailed below.

We train Skip-Gram with Negative Sampling (SGNS) representations on the corpus, and obtain sentence or document-level embedding using a composition function $f(w_i, ...w_n)$, where f is either addition (ADD), averaging (AVG) or the maximum (MAX). We do the same with Continuous Bag of Words (CBoW) representations. The resultant composed embeddings are used as input features for the classifier. For conciseness, we include only the best performing composite function here.

We also implemented using Keras (Chollet, 2015) a Convolutional Neural Network (ConvNet) for both sentence and document classification. We trained a binary classifier for each class, each consisting of the following layers: (i) input layer (domain trained embeddings using SGNS with $dim = 200$), (ii) 1-dimensional convolutional layer, (iii) max pooling layer with $dropout = 0.5$, (v) fully connected layer, and (vi) a binary softmax output layer. We use a binary cross-entropy loss function, and the Adam optimizer (Kingma and Ba, 2014). We also experimented with two key ConvNet parameters: the number of filters and the filter window size.

Finally, we compare against standard Bag of Words (BoW) classification, where each non-stop word in the corpus is a binary feature.

6.1 Task 1 results

The aim of Task 1 (semantic text classification) is to classify text into ten mutually non-exclusive classes, the Hallmarks of Cancer. The task has two sub-tasks: document-level and sentence-level classification. Table 6.1 shows the results for both levels. Sentence-level classification is more difficult, due to the smaller context information available. The table shows the results for the composed embedding baselines, the supervised BoW baseline and the handcrafted supervised model, as described in Section 4.2. This is followed by the three feature types (EMBED, CLASS-SIM, WORD-DIST) in all possible combinations and finally the full model, i.e., the one that uses all features.

With regards to the EMBED feature type, we distinguish between learning the representation independently (e.g., embeddings are learned without knowledge of the document) or jointly as described in Figure 1. We can see that the EMBED features by themselves perform better than any of the embedding baseline models. Jointly learning embeddings improves the F-score by approximately 4-5% for both document and sentence classification.

When considering the three features types, CLASS-SIM outperforms both EMBED and WORD-DIST, with an especially notable improvement in document classification.

When pairing the three feature types, the combination CLASS-SIM + WORD-DIST gives the best results, as is consistent with their individual results. Finally, when combining all three features, the full model outperforms all baselines with a significant margin, especially notable for sentence-level classification. Regarding the semi-supervised models, using class similarity CLASS-SIM alone significantly outperforms using word cosine distance WORD-DIST, and document-embedding to class-embedding distance EMB-DIST.

6.2 Domain variation

We also investigate the performance of our model and baselines when subjected to domain variation; that is, when we learn the embeddings from a different domain than that of the classification task. We experimented by learning the embeddings using the Wikipedia corpus. We seed the model with the labelled HoC training data and, then train on Wikipedia instead of domain specific literature acquired from PubMed.

	Document classification			Sentence classification		
Model	Precision	Recall	F-score	Precision	Recall	F-score
SGNS	43.7	25.9	32.5	13.6	23.3	17.2
CBoW	30.9	27.6	29.2	7.5	15.1	15.0
BoW	47.9	37.9	42.3	53.8	24.1	33.3
ConvNet	80.9	60.7	69.4	55.4	43.8	48.9
Handcrafted	82.8	69.4	75.5	59.2	46.4	51.4
EMB-DIST (semi-supervised)	24.3	28.5	26.3	25.4	25.4	21.9
WORD-DIST (semi-supervised)	30.9	36.5	33.5	43.7	24.6	31.5
CLASS-SIM (semi-supervised)	**44.1**	**38.8**	**41.3**	**26.7**	**42.1**	**32.6**
EMBED (independently)	44.0	37.4	40.4	26.5	48.0	34.2
EMBED (joint training)	54.2	46.4	49.9	37.6	39.6	38.6
CLASS-SIM	80.2	49.9	59.4	36.2	45.8	40.5
WORD-DIST	58.5	51.9	55.0	32.7	40.3	36.1
EMBED + CLASS-SIM	69.3	58.3	63.3	54.7	52.1	53.3
EMBED + WORD-DIST	60.9	60.4	60.7	54.6	56.3	55.4
CLASS-SIM + WORD-DIST	64.5	72.7	68.4	61.5	61.0	61.2
EMBED + CLASS-SIM + WORD-DIST	**85.5**	**69.8**	**76.4**	**77.7**	**60.1**	**67.6**

Table 3: Task 1 performance comparison. All figures are micro-averages (%).

Naturally, we expect all of the models to perform worse with Wikipedia-trained embeddings than with domain specific embeddings. This is indeed what happens (Table 6.2); however, some models prove more robust than others, i.e., their drop in F-score accuracy is smaller. By this measure, our full model and the semi-supervised models are less susceptible to domain variation with both document and sentence-level classification.

	Document		Sentence	
Model	Domain	Wikipedia	Domain	Wikipedia
SGNS	32.5	18.4	17.2	11.1
CBoW	29.2	14.3	15.0	10.4
ConvNet	69.4	38.3	48.9	29.7
Semi-supervised [3]	41.3	35.3	32.6	28.6
Full model	**76.4**	**61.5**	**67.6**	**54.6**

Table 4: Document and sentence classification micro-averaged F-score (%) using domain-specific and Wikipedia embeddings.

6.3 Task 2 results

The objective of Task 2 is to classify scientific text according to five argumentative zones. Table 6.3 summarises the results. Similar to Task 1, all three feature types perform significantly better than the embedding baseline models. When analysing the three feature types separately, WORD-DIST outperforms the other two. EMBED + WORD-DIST is the best-performing feature pair.

The full model significantly outperforms all baselines. However, it does not match the handcrafted approach. This is because the most influential feature in this task is the location of the text (Guo et al., 2011; Kiela et al., 2015). As our model does not take any word or sentence ordering into account, it would be difficult to compensate for the location feature. If, however, we include the location feature in addition to the three feature types in the SVM classification, our model outperforms the handcrafted

[3] Semi-supervised model uses CLASS-SIM.

baseline by a 1.4% difference. Admittedly, this would make the model slightly handcrafted by itself, but no additional work is necessary to get this feature and it does not vary across tasks or domains. This shows that our model including location information provides better features for this task than the handcrafted approach including location information. Looking at the semi-supervised models, the results suggest that CLASS-SIM outperforms the other feature types by an even larger margin than for Task 1.

Model	Precision	Recall	F-score
SGNS	45.6	31.5	37.3
CBoW	47.3	30.9	37.4
BoW	54.8	35.1	42.7
ConvNet	74.9	66.9	70.7
Handcrafted	88.9	85.0	86.9
EMB-DIST (semi-supervised)	23.7	38.9	29.4
WORD-DIST (semi-supervised)	36.6	28.8	32.2
CLASS-SIM (semi-supervised)	**57.1**	**40.9**	**47.7**
EMBED (sentences only)	43.3	41.9	42.6
EMBED (joint training)	57.6	37.7	45.6
CLASS-SIM	58.7	46.0	51.6
WORD-DIST	55.2	51.8	53.5
EMBED + CLASS-SIM	64.5	59.9	62.1
EMBED + WORD-DIST	78.1	57.5	66.3
CLASS-SIM + WORD-DIST	82.0	54.5	65.5
EMBED + CLASS-SIM + WORD-DIST	**81.2**	**72.7**	**76.7**
Full model + location	**89.6**	**86.9**	**88.3**

Table 5: Task 2 Micro-averaged performance comparison. All figures are percentages.

7 Discussion and conclusions

The aim of this paper has been to produce a robust approach to text classification for domains suffering from sparsity of labelled data, and to alleviate the necessity for handcrafting features. Our novel methodology jointly learns distributed semantic representations at the level of words, sentences, documents and class.

The intuition is that embeddings at each level capture slightly different topical semantics. We therefore employ these embeddings to produce three types of features that require no additional data or labour, that are efficient to extract and much easier to port than handcrafted features. We have shown how these feature types can be used with standard classification algorithms such as SVMs and with semi-supervised classification where the decision boundaries are not learned from labelled data.

In the first task (semantic text classification) our approach matched or outperformed a handcrafted fully-supervised approach. The model performed substantially better at sentence-level classification which had much less context than the document-level classification. We also showed that our features are less susceptible to domain variation.

In the second task (rhetorical text classification), the proposed model outperformed all baselines, as well as the handcrafted approach when including location information in the classification process.

Acknowledgments

The first author is funded by the Commonwealth Scholarship and the Cambridge Trust. This work is supported by Medical Research Council grant MR/M013049/1 and the Google Faculty Award.

References

Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text clustering algorithms. In *Mining Text Data*, pages 77–128. Springer.

Simon Baker, Ilona Silins, Yufan Guo, Imran Ali, Johan Högberg, Ulla Stenius, and Anna Korhonen. 2016. Automatic semantic classification of scientific literature according to the hallmarks of cancer. *Bioinformatics*, 32(3):432–440.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*, pages 730–738.

François Chollet. 2015. Keras: Deep learning library for theano and tensorflow.

Stephen Clark. 2002. Supertagging for combinatory categorial grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+ 6)*, pages 19–24.

Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. 2007. Transferring naive bayes classifiers for text classification. In *Proceedings of the national conference on artificial intelligence*, volume 22, page 540. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Rosa L Figueroa, Qing Zeng-Treitler, Long H Ngo, Sergey Goryachev, and Eduardo P Wiechmann. 2012. Active learning for clinical text classification: is it better than random sampling? *Journal of the American Medical Informatics Association*, 19(5):809–816.

Yufan Guo, Anna Korhonen, Maria Liakata, Ilona Silins Karolinska, Lin Sun, and Ulla Stenius. 2010. Identifying the information structure of scientific abstracts: an investigation of three different schemes. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, pages 99–107. Association for Computational Linguistics.

Yufan Guo, Anna Korhonen, Ilona Silins, and Ulla Stenius. 2011. Weakly supervised learning of information structure of scientific abstractsis it accurate enough to benefit real-world tasks in biomedicine? *Bioinformatics*, 27(22):3179–3185.

Yufan Guo, Ilona Silins, Roi Reichart, and Anna Korhonen. 2012. CRAB reader: A tool for analysis and visualization of argumentative zones in scientific literature. In *Proceedings of COLING 2012: Demonstration Papers*, pages 183–190.

Yufan Guo, Ilona Silins, Ulla Stenius, and Anna Korhonen. 2013. Active learning-based information structure analysis of full scientific articles and two applications for biomedical literature review. *Bioinformatics*, 29(11):1440–1447.

Douglas Hanahan and Robert A Weinberg. 2000. The hallmarks of cancer. *Cell*, 100(1):57–70.

Douglas Hanahan and Robert A Weinberg. 2011. Hallmarks of cancer: the next generation. *Cell*, 144(5):646–674.

Chaochao Huang, Xipeng Qiu, and Xuanjing Huang. 2014. Text classification with document embeddings. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 131–140. Springer.

Douwe Kiela, Yufan Guo, Ulla Stenius, and Anna Korhonen. 2015. Unsupervised discovery of information structure in biomedical documents. *Bioinformatics*, 31(7):1084–1092.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 957–966. JMLR Workshop and Conference Proceedings.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

Florian Leitner, Scott A Mardis, Martin Krallinger, Gianni Cesareni, Lynette A Hirschman, and Alfonso Valencia. 2010. An overview of biocreative ii. 5. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 7(3):385–399.

Haibin Liu, Tom Christiansen, William A Baumgartner Jr, and Karin Verspoor. 2012. Biolemmatizer: a lemmatization tool for morphological processing of biomedical text. *J. Biomedical Semantics*, 3:3.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *AAAI*, pages 2418–2424.

A. Marusyk, V. Almendro, and K. Polyak. 2012. Intra-tumour heterogeneity: a looking glass for cancer? *Nature Reviews Cancer*, 12(5):323–334.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Nir Nissim, Mary Regina Boland, Robert Moskovitch, Nicholas P Tatonetti, Yuval Elovici, Yuval Shahar, and George Hripcsak. 2015. An active learning framework for efficient condition severity classification. In *Artificial Intelligence in Medicine*, pages 13–24. Springer.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Changqin Quan, Meng Wang, and Fuji Ren. 2014. An unsupervised text mining method for relation extraction from biomedical literature.

Kevin Reschke, Martin Jankowiak, Mihai Surdeanu, Christopher D Manning, and Daniel Jurafsky. 2014. Event extraction using distant supervision. In *Language Resources and Evaluation Conference (LREC)*.

Laura Rimell and Stephen Clark. 2009. Porting a lexicalized-grammar parser to the biomedical domain. *Journal of biomedical informatics*, 42(5):852–865.

Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Burr Settles. 2005. Abner: an open source tool for automatically tagging genes, proteins and other entity names in text. *Bioinformatics*, 21(14):3191–3192.

Matthew S Simpson and Dina Demner-Fushman. 2012. Biomedical text mining: a survey of recent progress. In *Mining text data*, pages 465–517. Springer.

Lin Sun and Anna Korhonen. 2009. Improving verb clustering with automatically acquired selectional preferences. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 638–647. Association for Computational Linguistics.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: experiments with relevance and rhetorical status. *Computational linguistics*, 28(4):409–445.

Jorge Vivaldi and Horacio Rodríguez. 2015. Medical entities tagging using distant learning. In *Computational Linguistics and Intelligent Text Processing*, pages 631–642. Springer.

Xiaojun Wan. 2007. A novel document similarity measure based on earth movers distance. *Information Sciences*, 177(18):3718–3730.

Yan Yan, Xu-Cheng Yin, Sujian Li, Mingyuan Yang, and Hong-Wei Hao. 2015. Learning document semantic representation with hybrid deep belief network. *Computational intelligence and neuroscience*, 2015.

Dani Yogatama and Noah A Smith. 2014. Linguistic structured sparsity in text categorization.

Dani Yogatama, Manaal Faruqui, Chris Dyer, and Noah Smith. 2015. Learning word representations with hierarchical sparse coding. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 87–96.

Shaodian Zhang and Noémie Elhadad. 2013. Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts. *Journal of biomedical informatics*, 46(6):1088–1098.

Mathematical Information Retrieval based on Type Embeddings and Query Expansion

Yiannos A. Stathopoulos **Simone Teufel**
Computer Laboratory, University of Cambridge
15 JJ Thomson Avenue, Cambridge, UK
{yiannos.stathopoulos,simone.teufel}@cl.cam.ac.uk

Abstract

We present an approach to mathematical information retrieval (MIR) that exploits a special kind of technical terminology, referred to as a *mathematical type*. In this paper, we present and evaluate a type detection mechanism and show its positive effect on the retrieval of research-level mathematics. Our best model, which performs query expansion with a type-aware embedding space, strongly outperforms standard IR models with state-of-the-art query expansion (vector space-based and language modelling-based), on a relatively new corpus of research-level queries.

1 Introduction

Mathematical information retrieval (MIR) systems, such as MathWebSearch (Kohlhase and Prodescu, 2013; Hambasan et al., 2014), MIaS (Sojka and Líška, 2011) and Tangent (Pattaniyil and Zanibbi, 2014) have demonstrated that indexing and matching of formulae is beneficial to MIR. However, despite the sophistication of these methods, they leave elements of the natural language of mathematics largely unexploited. In contrast to general text, text in mathematics follows strong domain-specific conventions governing how content is presented (Ganesalingam, 2008). This is particularly so for research-level mathematics text (i.e., scientific articles), which is our main focus of interest. The conventionality of this text gives rise to many opportunities for the application of NLP to MIR. In this paper, we investigate the role of *mathematical types*, a special kind of technical terminology.

The term "type" refers to sequences of one or more words used to label mathematical objects (e.g., 'set", "smooth curve"), algebraic structures (e.g., "monoid", "group") and instantiable mathematical notions (e.g., "cardinality of a set"). Technical terms that are not used to refer to instances of these mathematical constructs are not types. Examples of non-types include references to the application of mathematical procedures (e.g., "proof by contradiction") and elements of the mathematical discourse (e.g., "theorem 4.1"). As a subclass of mathematical terminology, types are almost exclusively noun or prepositional phrases.

Types play a central role in communicating mathematical information by enabling mathematicians to name mathematical concepts, assign properties to objects and prove assertions about them. We consider types worthy of distinction from generic technical terms for two reasons: (a) types are used in the discourse to give sense to constituents of formulae and (b) types are used to consistently evoke mathematical concepts in textual argumentation and mathematical reasoning.

Our goal is to evaluate the usefulness of types as standalone lexical components in the retrieval of research-level mathematical information needs. Specifically, our hypothesis is twofold: (a) types are important discriminators in the mathematical discourse and (b) semantic relationships between types can be used to enrich queries and obtain significant improvements in retrieval efficiency.

A sub-task of our type-based approach is the construction of a type dictionary from a collection of documents. We address this in section 3 and describe a simple method for automatic identification of types. Furthermore, we evaluate our method using a gold standard set of type phrases which we have produced using judgements from 5 mathematicians.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2344–2355, Osaka, Japan, December 11-17 2016.

We focus on retrieving research-level mathematics because such material is rich in mathematical types. For instance, our system operates on queries such as the following (types are underlined):

> *Let P be a parabolic subgroup of $GL(n)$ with Levi decomposition $P = MN$, where N is the unipotent radical. Let p be an irreducible representation of $M(Z_p)$ inflated to $P(Z_p)$, how does $Ind_{P(Z_p)}^{GL_n(Z_p)}\pi$ decompose? It would be sufficient for me to know the result in the simplest case, where P is a Borel subgroup.*

Our experiments suggest that types are most effective when used to capture semantic relationships between mathematical concepts. Our top-performing type-based model, TypesExp, makes use of similarity in a type-aware word embedding space to identify semantically related types for query expansion. TypesExp outperforms state-of-the-art and traditional IR/query expansion models (described in section 4.3) demonstrating experimentally that types are valuable lexical components for IR in their own right (section 5).

2 Related Work

We use types to model mathematical concepts, but other constructs have been proposed in the literature for the same purpose. Grigore et al. (2009) take operators listed in OpenMath content dictionaries (CDs) to be mathematical concepts and use *term clusters* to model their semantics. A term cluster for a concept is composed of a label and a bag of nouns extracted from the operator description in the dictionary. This set is enriched manually using additional terms taken from online mathematical lexical resources. Grigore et al. assign the cluster that maximises the similarity (based on PMI and DICE) between the nouns in the local context of a target formula and those in the cluster, to represent the formula's semantics. Quoc et al. (2010) extract descriptions for formulae (phrases or sentences) from their surrounding context using a rule-based approach. Kristianto et al. (2012), on the other hand, used pattern matching on sentence parse trees and a "nearest noun" approach to extract descriptions, but these methods were later outperformed by SVMs (Kristianto et al., 2012).

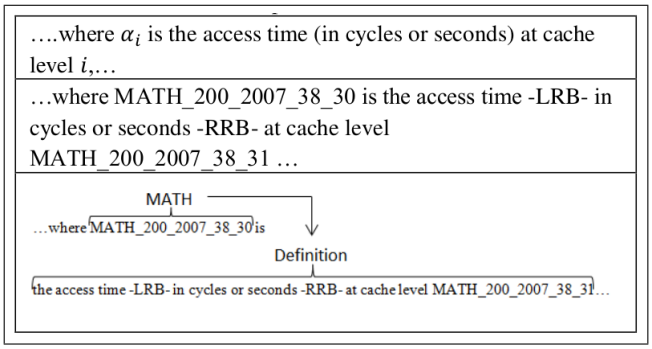


Figure 1: Example of extracting descriptions adapted from (Kristianto et al., 2012)

Our approach at modelling concepts with types incorporates the advantages of the described methods. Like term clusters, types label mathematical concepts and attach meaning in a distributional manner. However, rather than constructing type representations manually, we use types as concept labels and automatically compute distributional profiles for the concepts, based on word embeddings. Like formulae descriptions (Kristianto et al., 2012; Kristianto et al., 2014), our types are also extracted automatically from text. Although in some cases descriptions extracted by Kristianto et al. (2012, 2014) are noun phrases and resemble types, they often incorporate details that are particular to a specific context. For example, the extracted description for formula a_i presented in Figure 1 (adapted from (Kristianto et al., 2012)) contains information that is specific to the context (i.e., time at a particular cache level measured in cycles or seconds).

In our approach, we restrict types to relatively short technical terms which do not include context-specific information. This decision stems from our motivation to capture references to mathematical constructs in the form that they most consistently appear in scientific text.

3 Types for Math IR

Our definition of types is intended to model the perception of mathematical concepts shared between mathematicians and emerges from mathematical intuition. The notion of a type is intuitive to most mathematicians (as demonstrated in section 3.2). It is, however, hard to produce a concrete list of properties a technical term must adhere to in order to be considered a type.

Linguistically, types are a special kind of mathematical technical terminology. Technical terminology in general is well understood; for example, Justeson and Katz (1995) were among the first to define terminology as noun phrases with particular statistical properties. As lexical tokens, types are subject to a particularly high level of polysemy ("field" in Mathematics is a concept distinct to that in Physics) and synonymy (e.g., "karoubi envelope" is the same as "category of idempotent arrows"). Furthermore, many mathematical constructs are eponyms, i.e., named after their inventors, often additionally pre-modified by adjectives (e.g., "refined Noether normalization theorem", "abstract Hilbert space theorem"). New types can be formed through parameterisation (e.g.,"2-Group" and "G-Function" from "Group" and "Function" respectively) or by prepositional postmodification (e.g., "Ideal of a Ring", "Point on the Plane" and "Set of Matrices").

Conceptually, a type is any technical term that is (a) perceived by mathematicians to refer to mathematical objects, algebraic structures and mathematical notions and (b) can be instantiated in the discourse in the form of a variable. Here, we take mathematical objects to be anything that can be formally defined and manipulated in the discourse as part of formal deductive reasoning and/or proofs. This being said, it is important to highlight that some objects such as 'Number", "Matrix" and "Set" are considered basic and are never explicitly defined by mathematicians (Ganesalingam, 2008).

Like mathematical objects, algebraic structures also take part in mathematical manipulation but are defined as collections (or tuples) of other objects. Types can also refer to mathematical notions that can be instantiated as variables or can take the form of other objects. For example, an "envelope of elliptic trajectories" can be an "ellipse". Named axioms, theorems and conjectures are also considered to be types since they refer to universally accepted, formally defined constructs for the purpose of argumentation (e.g., to complete a proof).

Any technical term that does not fit the above description is not a type. Examples of non-types include properties of operators (e.g., "Associativity"), mathematical procedures (e.g., "Proof by contradiction"), processes (e.g., "Differentiation") and theories (such as "Chaos"). Note that mentions of mathematical theories are ambiguous: it is often unclear whether they refer to a branch of mathematics or to a formally defined construct. These examples are not types because either (a) they are not explicitly referring to mathematical constructions (i.e., they cannot assign meaning to variables), (b) their role in the discourse is indirect (e.g., properties capture relationships between concepts) and (c) they are used as discourse labels for anaphoric purposes.

Types can be organised hierarchically, with some types being specialised instances of other, more abstract constructions. This relationship is often mirrored linguistically: a type expressed by a longer string is often a subtype of the type corresponding to a sub-sequence string. For example, a "smooth curve" is a subtype of "curve". We consider sub-types to be distinct atomic units with discrete meaning despite the fact that they are constructed linguistically in a compositional manner with regards to their super-type. However, not all type/sub-type relationships are expressed on the surface. For example, it is not obvious that "Klein Bottle" is a sub-type of "Surface".

Types are relevant to both textual and mathematical contexts of queries and documents. As a result, types have potentially more impact than generic technical terms. Given that types communicate ideas shared between mathematicians, we anticipate that modelling the distributional profile of types (e.g., using an embedding space) will be beneficial to MIR.

3.1 Automatic Type Detection and Extraction

We address the problem of automatic detection and extraction of types so that we can perform large-scale experimentation. Our method proceeds as follows. First, we use the C-Value algorithm (Frantzi et al., 1998) on our corpus to extract technical terms (candidate types).

Given a collection of documents as input, the C-Value method identifies multi-word technical terms using both a linguistic and a statistical component. The linguistic component is employed primarily for eliminating multi-word strings that are unlikely to be technical terms. This is done by enforcing a stop-word list (high-frequency corpus terms) and through the application of linguistic filters (regular expressions) on sequences of part-of-speech tags. The statistical component assigns a "termhood" score to a candidate sequence based on its corpus-wide statistical characteristics and those of the sequences that contain it.

Each entry in the output of the C-Value algorithm corresponds to one technical term – an equivalence class of all variations of the term in the corpus. In the next step, our process selects technical terms that are likely to be types. We assume that technical terms that are types have an entry in the Encyclopedia of Mathematics[1] (8730 articles in total at the time of download) and/or an entry in Wikipedia (we used a 2014 Wikipedia dump) under the key categories "mathematical objects", "mathematical concepts" and "mathematical structures" and their sub-categories.

A dictionary of types is constructed by including technical terms that entirely match the title of one or more of these encyclopedia articles. We have opted to use this intersection of technical terms and article titles, as opposed to the titles alone, because not all titles are useful. For example, although the Wikipedia article[2] "The geometry and topology of three-manifolds" is filed under the categories of interest "Hyperbolic geometry", "3-manifolds" and "Kleinian groups", the title as a whole does not represent a single concept. In contrast, the technical term "Riemannian manifold" would completely match the title of the Wikipedia[3] (or Encyclopedia of Math) article for the concept and would thus be identified as a type by our method. The application of our method to the Mathematical REtrieval Corpus (MREC) (Líška et al., 2011)[4] has produced a dictionary of 10601 types.

3.2 Gold Standard Evaluation

Our definition of types is intricate and requires mathematical expertise. We also expect it to be subjective. We therefore evaluated the quality of accumulated types using 5 judges (third-year undergraduate and graduate mathematicians). Participants were shown a mixed list of types (as determined by our system) and non-types (technical terms that had been filtered out by our method as non-types). Without knowing what the source of each type was, they were asked to identify types using a 2-page definition of types (16 rules). The technical term list they were asked to judge consisted of 200 phrases and was constructed as follows: (1) two-thirds of the sample are sourced from the type list (10601 phrases). This set of 134 phrases is produced by sampling by observed distribution over phrase length. (2) The remainder of the sample (66 phrases) is sourced from the original list of C-Value technical terms not identified as types by our method. The termhood scores of these technical terms can range from very high to very low. As a result, we split the original technical term list (2.8 million phrases) into three equally-sized segments based on their C-Value score: (a) high score, (b) medium and (c) low score. From each segment, we removed any term already identified as a type and drew 22 phrases from the remainder. As before, segment samples are sampled by observed distribution based on phrase length. Sampling negatives from the three segments, as described above, enables us to compare the spread of positives and negatives across C-Value scores. (3) The two parts of the sample are concatenated and shuffled randomly. An HTML questionnaire is automatically produced and presented to annotators.

Precision and recall of our type identification method was $P = 73.9\%$ and $R = 81.8\%$ respectively, resulting in an F-score of 77.7%, with respect to the majority opinion. As expected, this is a subjective task, and without any specific training, annotator agreement, measured using Fleiss's Kappa (Fleiss, 1971), is in an intermediate range ($K = 0.65; N = 200, k = 2, n = 5$). We observed that judges often judged the following technical terms as types, although this contradicts our definition: author names, mathematical properties and non-sensical phrases.

[1] https://www.encyclopediaofmath.org

[2] https://en.wikipedia.org/wiki/The_geometry_and_topology_of_three-manifolds

[3] https://en.wikipedia.org/wiki/Riemannian_manifold

[4] The MREC is a subset of ArXiv and is composed of over 439,000 scientific papers which have had all LaTeX formulae converted into MathML (Líška et al., 2011). It is the document collection underlying the test collection we use (section 4).

4 IR Experiments

Traditional retrieval models operate under the assumption that each constituent term in a multi-word type is an independent source of information. Our intuition is that the words in multi-word types carry more information as a group and should therefore be treated as atomic lexical units by these models. We propose two type-aware models, based on the traditional Vector-space model (VSM). In our evaluation we compare our type-aware models to established traditional, term-based[5] retrieval models. The motivation behind this approach is to clearly identify the effects of type information. We keep the comparison at the lexical level so that the effects of types to retrieval performance can be isolated – models employing formulae indexing and matching are not considered.

4.1 Computing a Type Embedding Space

We use our dictionary of types (section 3.1) to construct a type embedding space – a word embedding space that includes embeddings for types as atomic lexical units. The type embedding space is used to assign meaning (or denotation) to types in the form of vector embeddings and to expand queries with new types (section 4.3.1).

The type embedding space is constructed as follows. First, we apply sentence tokenisation over the MREC using the Stanford CoreNLP toolkit (Manning et al., 2014). Subsequently, we apply longest sequence matching on the words of each sentence and identify all instances of types in the the corpus. Once detected, word sequences belonging to types are replaced by a single token (a concatenation of the constituent words of the type). As we are interested in modelling the relatedness of meaningful linguistic tokens, rather than mathematical artefacts, we replace MathML blocks representing formulae in the sentence by a single token ("@@@"). Finally, the re-written sentences are passed on to `word2vec` in skipgram mode (Mikolov et al., 2013a; Mikolov et al., 2013b) with negative sampling and window size=10 to produce the type embedding space.

4.2 Test Collection

Evaluation is carried out using the Cambridge University MathIR Test Collection (CUMTC) (Stathopoulos and Teufel, 2015), which is composed of 120 real-life MIR topics procured from the MathOverflow (MO) on-line community. As illustrated in Table 1, each MO thread in the CUMTC is sentence-tokenized, with sentences either being part of the "prelude" (introduction to the mathematical subject of interest) or part of a concrete sub-question. We produced 160 queries from the CUMTC by emitting one query per sub-question in the collection. The body of each query is obtained by concatenating the text of the sub-question to the text of the associated prelude.

Prelude	Let P be a parabolic subgroup of $GL(n)$ with Levi decomposition $P = MN$, where N is the unipotent radical. It would be sufficient for me to know the result in the simplest case, where P is a Borel subgroup.
SQ-1	Let p be an irreducible representation of $M(Z_p)$ inflated to $P(Z_p)$, how does $Ind_{P(Z_p)}^{GL_n(Z_p)}\pi$ decompose? (at least until $g = 3$).

Table 1: Topic 175 (MO post 90038), prelude and sub-question

We chose this test collection because its topics represent real-life, research-level information needs, expressed in the natural language of mathematics, and are rich in mathematical types. This is in contrast to the NTCIR (Aizawa et al., 2013; Aizawa et al., 2014) test collections which emphasise formulae search (Guidi and Coen, 2015) with queries primarily taking the form of bags of keywords and formulae. Although the NTCIR Open Information Retrieval (OIR) is composed of free-text queries like those in the CUMTC, these are not accompanied by pre-determined relevance judgements[6]. Furthermore, textual descriptions in the OIR evaluation set are not as linguistically rich as the text in mathematics papers. Documents and queries are processed using a single pipeline that performs case normalisation and employs the Tika framework to flatten MathML consistently.

[5] A bigram model has also been considered but excluded from the comparison due to extremely poor performance.

[6] Relevance is judged using interactive sessions with humans.

4.3 Experimental Design

Performance is measured using mean average precision (MAP). Queries derived from the CUMTC are fairly long (averaging 88 words) and describe highly specialised information needs. As a result, they have relatively few relevant documents (only 19.17% have more than 1 relevant documents). Thus, our experimental setup is closer to that of the TREC HARD and TREC Robust tasks (Voorhees and Harman, 2005), than to more general-purpose retrieval (such as NTCIR and TREC ad-hoc) and low MAP values are to be expected. In the case of the CUMTC, the sparsity of relevant documents is attributable to the difficulty of the queries, rather than to the inherent uniqueness of the answer (as is the case in homepage search). One of the effects of the nature of the CUMTC is that the MAP scores of IR models will be numerically low. However, note that the small number of relevant documents per query does not make the evaluation per se unstable: we use a large number of queries and adopt the paired permutation test for significance testing. The permutation test is a non-parametric test for mean difference and is known to be reliable with MAP and its derivatives (Smucker et al., 2007).

In order to investigate the usefulness of mathematical types for retrieving research-level mathematics we adopt a two-level comparison of retrieval models. On one level, we compare traditional, term-based retrieval models to type-aware derivatives (see Figure 2 for the derivational relationship between models). On the second level, we compare the effectiveness of term-based query expansion to that based on types.

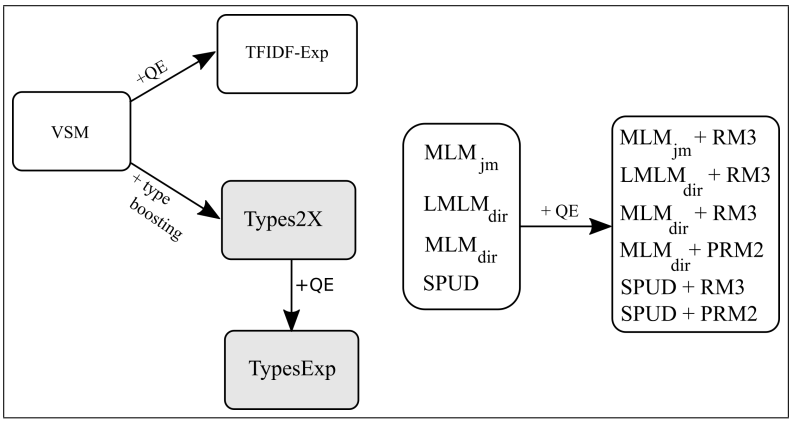


Figure 2: Relationships between basic models and their more sophisticated derivatives (grey boxes represent type-based models, white boxes represent term-based models).

The first level of comparison is performed across IR paradigms and includes term-based models that employ heuristic methods (e.g., VSM and BM25) as well as language modelling (e.g., classical multinomial language model (Zhai and Lafferty, 2001)). At this level, we wish to (a) determine the performance of traditional IR models on the CUMTC and (b) investigate if types are more useful than simple terms when retrieving research-level mathematics. A break-down of considered models based on this two-level comparison is presented in Table 2.

	No Expansion	**Query Expansion**
Terms	VSM	TFIDF-Exp
	BM25	-
	MLM_{jm}	MLM_{jm}+RM3
	MLM_{dir}	MLM_{dir}+RM3, MLM_{dir}+PRM2
	$LMLM_{dir}$	$LMLM_{dir}$+RM3
	SPUD	SPUD+RM3, SPUD+PRM2
Types	Types2X	TypesExp

Table 2: Overview of models.

4.3.1 Retrieval Models

We propose and evaluate two type-based heuristic models, based on VSM, that assign elevated significance to types through 2X boosting. One of our models makes use of inter-type similarity, encoded in a type embedding space, to expand queries.

Lucene VSM with 2X type boosting (Types2X). We apply the boosting pipeline described by Stathopoulos and Teufel (2015) to our type dictionary (section 3.1); i.e, we apply longest matching to a Lucene positional index and emit a type-aware "delta index". This type discovery and normalisation pre-processing step is also applied to queries. Our assumption is that types are a valuable source of information for an MIR system. Types2X assumes the role of a type-aware model that performs the simplest manipulation of those types that are physically present in the query (simple 2X boosting). Therefore, in our comparison, Types2X is used to measure the simplest possible way of incorporating types during retrieval (as opposed to just using terms). We expect Types2X to perform better than Lucene VSM (the term-based model it is based on) but no better than models incorporating type information in more sophisticated ways (such as TypesExp).

Lucene VSM with 2X type boosting and type-based query expansion (TypesExp). Unlike mathematical papers, queries are not always rich in types: on average each query contains around 13 type instances while documents in the MREC contain on average close to 548 type instances. TypesExp overcomes this problem by enriching queries with types. Queries are expanded using the types (as opposed to the terms) they contain. For each type in a query, the type-embedding space (using `word2vec` similarity as discussed in section 4.1) is used to discover n fresh related types. Semantic relatedness between types is modelled by the cosine similarity of their vector representations. The set of fresh types is appended to the original query and the new query is executed on a 2X type up-weighted VSM. The value for n is the only parameter of the model, which has been experimentally set to $n = 5$.

4.3.2 Baseline Models

Traditional, term-based retrieval and query expansion models [7] are used as baselines so that the effects of a-priori knowledge of types to retrieval performance can be isolated and quantified against uninformed approaches. Two query expansion (QE) methods based on pseudo-relevance feedback (PRF) are considered. The first method, known as RM3 (Abdul-jaleel et al., 2004; Lv and Zhai, 2009), assumes that documents and queries are generated by the same relevance model. The second, referred to as PRM2 (Lv and Zhai, 2011), makes use of proximity information in the feedback documents to expand queries.

Simple Baselines We use Lucene's default VSM (based on cosine similarity) and BM25 (Harter, 1975; Robertson et al., 1981; Robertson and Walker, 1994; Robertson et al., 1994) implementations. These models are considered not because of their strength per se, but because of their usefulness in identifying the effects of types in the performance of heuristic models: they are lingustically uninformed models. Furthermore, they are useful in quantifying the effects of boosting query types alone (Types2X).

Language Models (LM) Smoothed instances of the classical multinomial language model (Zhai and Lafferty, 2001) are also considered. The MLM_{jm} multinomial model employs Jelinek-Mercer (JM) smoothing. We rely on Lucene's implementation of the model and, in the absence of training data for the parameter λ, we use $\lambda = 0.7$ since there is strong evidence that this value is optimal for long queries (Zhai and Lafferty, 2001; Zhai and Lafferty, 2004). Two implementations of the multinomial model with Dirichlet smoothing are used in our comparison: the Lucene implementation ($LMLM_{dir}$) and the implementation by Cummins et al. (2015) (MLM_{dir})[8]. The smoothing parameter, μ, is set to $\mu = 2000$, which Zhai and Lafferty (2001, 2004) found to be near-optimal for large queries, such as those in our setup. These language models are included as more sophisticated, type-agnostic alternatives to the basic heuristic models.

[7] Technical issues have prevented us from incorporating the formula-aware Tangent and MiAS models in our evaluation.

[8] In correspondence with one of the authors of Cummins et al. (2015), it has come to our attention that there is suspicion in the community that the Lucene implementation of the classical multinomial language model may be incorrect for large queries, so we report two implementations for safety.

SPUD SPUD is a state-of-the-art unigram LM that models documents as draws from a multivariate Polya distribution (Cummins et al., 2015). Smoothing of the document model is performed using a linear combination of the unsmoothed document and background models and is controlled through a smoothing parameter ω. The SPUD ranking method estimates the probability that a query is generated from the expected multinomial drawn from each document model. We use the SPUD implementation by Cummins et al. (2015) with default parameters ($\omega = 0.8$), which have been shown to produce good results with long queries (Cummins et al., 2015). SPUD is included in our comparison as a state-of-the-art type-agnostic LM baseline against which we can benchmark the performance of our type-based models.

We also consider versions of the stated language models augmented with query expansion methods (RM3 and PRM2) implemented as part of the code accompanying (Cummins et al., 2015)[9]. Parameter values for these query expansion models are also taken from Cummins et al. (2015). LMs with QE are used to determine how type-based QE performs in comparison to state-of-the-art term-based QE.

Automatic QE using top TF-IDF terms (TFIDF-Exp). In this model, the query is expanded using the top s terms in the query as determined by document collection-wide TF-IDF scores. Stopwords and words with term frequency lower than 50 are excluded. Like before, each selected term is expanded using its n-nearest neighbours in a word embedding space (skip-gram, window size =10). The model has two parameters: *pool size* (n) and *seed size* (s). We found experimentally that the model performs best for $n = 1$ and $s = 1$. This baseline is used to determine whether any performance improvements obtained by type-based QE using an embedding space are due to types, rather than the use of embedding spaces in general.

5 Results and Discussion

Table 3 shows the results of all retrieval models considered[10]. The last two columns indicate the significance in MAP difference between each model and our proposed type-based models. As expected, absolute MAPs are low across the board, a phenomenon that can be attributed to the complexity of the underlying information needs and the resulting small number of relevant documents per query. But as long as the evaluation is stable, it is only the comparative performance of each model that we should be interested in.

	VSM	BM25	MLM_{jm}	$LMLM_{dir}$	MLM_{dir}	SPUD	TF-IDFExp	MLM_{jm} +RM3
MAP	.076	.079	.084	.072	.066	.090	.060	.063
TypesExp	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$
Types2X	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$
	$LMLM_{dir}$ +RM3	MLM_{dir} +RM3	MLM_{dir} +PRM2	SPUD +RM3	SPUD +PRM2	Types2X	TypesExp	
MAP	.051	.082	.061	.050	.072	.094	.150	
TypesExp	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$>$	-	
Types2X	$>$	$\approx$	$\approx$	$>$	$\approx$	-	$<$	

Table 3: Model MAP performance and comparison to TypesExp and Types2X models.

The model utilising type-based expansion (TypesExp) is our best model; it outperforms every other model. In some cases the differences are dramatic; TypesExp's MAP score is twice that of the Lucene VSM and, to the best of our knowledge, 0.15 MAP represents the state-of-the-art on the CUMTC. We now discuss which of TypesExp's components contributed most to this improvement over existing IR models. Although the up-weighting of types on its own (Types2X) improves MAP over the VSM, the difference in performance is not statistically significant. This is also the case when comparing type up-weighting to the majority of alternative models. The best performing model not employing query expansion is SPUD, followed closely by MLM with JM smoothing. The traditional BM25, VSM and

[9] https://github.com/ronancummins/spud

[10] In Tables 3, 4 and 5, $\ll$ indicates that "column" is significantly better than "row" $\alpha = 0.01$; $<$ at $\alpha = 0.05$. $\approx$ indicates that difference is not significant.

Lucene MLM_{dir} models performed comparably, with no significant differences in MAP between them (as observed in Table 4).

VSM	-						
BM25	$\approx$	-					
MLM_{jm}	$\approx$	$\approx$	-				
$LMLM_{dir}$	$\approx$	$\approx$	$\approx$	-			
MLM_{dir}	$\approx$	$\approx$	$\approx$	$\approx$	-		
TFIDF-Exp	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	-	
SPUD	$\approx$	$\approx$	$\approx$	$\approx$	$>$	$\approx$	-
	VSM	BM25	MLM_{jm}	$LMLM_{dir}$	MLM_{dir}	TFIDF-Exp	SPUD

Table 4: Comparison of models not employing query expansion.

	MLM_{jm} +RM3	$LMLM_{dir}$ +RM3	MLM_{dir} +RM3	MLM_{dir} +PRM2	SPUD +RM3	SPUD +PRM2
MLM_{jm}	$>$ **.021**	$>$ **.033**	.002	.023	.034	.013
MLM_{dir}	.003	.015	.016	.005	.016	.006
$LMLM_{dir}$	.009	$\gg$ **.021**	.01	.011	.022	.001
SPUD	$>$ **.028**	$>$ **.04**	.009	$>$ **.029**	$\gg$ **.04**	.019

Table 5: Absolute difference in MAP of unexpanded and query-expanded models.

General-purpose query expansion methods appear to be ineffective in retrieving research-level mathematics. From the results in Table 5 we observe that versions of the models augmented with state-of-the-art query expansion are either significantly outperformed or not significantly better than their corresponding basic versions. In other cases, the vanilla models significantly outperform their query expanding counterparts (e.g.,MLM_{dir} and $MLM_{dir} + RM3$, $SPUD$ and $SPUD + RM3$). This is in contrast to type-based query expansion, where the TypesExp model (2X type up-weighting+expansion using a type-aware embedding space) significantly outperforms both the VSM and Types2X models it is based on. The observations described above seem to point to the fact that, in the context of MIR, mathematical types encode more information than the sum of their individual, constituent terms.

On one hand, the performance of Types2X suggests that information coming from the types occurring in the queries alone may not be enough to produce significant improvements in retrieval efficiency. On the other hand, our experiments have shown that it is only the combination of query expansion with type information (rather than with simple terms) that yields significant performance gains on these difficult queries. Our intuition is that type-based expansion introduces semantically related concepts that elevate the score of topically relevant documents. Insight into why this method performs well can be obtained by looking into how types are expanded. The query in Table 6 is topic 175 (MO post 90038[11]). From an initial set of 6 types in the dictionary, our method expanded the query with 14 more types.

Query	Let P be a *parabolic subgroup* of $GL(n)$ with *Levi decomposition* $P = MN$, where N is the *unipotent radical*. Let p be an *irreducible representation* of $M(Z_p)$ inflated to $P(Z_p)$, how does $Ind_{P(Z_p)}^{GL_n(Z_p)}\pi$ decompose? It would be sufficient for me to know the result in the simplest case, where P is a *Borel subgroup*.
Query types	'levi decomposition', 'parabolic subgroup', 'unipotent', 'irreducible representation','borel subgroup'
Added Types	'reducible representation', 'regular element', 'conjugacy class', 'iwasawa decomposition', 'unipotent element', 'cartan decomposition', 'finite-dimensional representation', 'unitary representation','cartan subgroup', 'centralizer', 'subgroup', 'triangular decomposition', 'jordan decomposition','parabolic subalgebra'
TFIDF-Exp terms	'nilpotent', 'shredded', 'semi-simple', 'inflates', 'centralizer', 'pro-', 'puffed','engulfed', 'inflate', 'semisimple'

Table 6: Topic 175 (MO post 90038): Text and types in query, types in query and dictionary, expansion types and sample TFIDF-Exp expansion terms (n=5, $s = 2$).

Broadly speaking, all of the types expanded by our system are topically related to the types in the query. The types "iwasawa decomposition" and "cartan decomposition" in the expanded set strongly

[11] http://mathoverflow.net/questions/90038

relate to the types in the query (both are related to Levi decomposition of Lie algebras). However, there are also some instances of weak association between query and expanded types. For example, the type "subgroup" is almost certainly too general to do any good. On the other hand, the TFIDF-Exp expansion set for this query (bottom row of table 6) appears to be less likely to contain terms related to the topic subject. The "bag-of-types" model (Types2X) is inhibited by the fact that queries have a much smaller vocabulary of types compared to mathematical documents. Furthermore, we hypothesise that the model's limited performance and the effectiveness of topical relationships discovered through types and type embeddings are attributable to three characteristics of the mathematical discourse.

First, types can share algebraic structure, which permits mathematicians to perform mathematical reasoning by manipulating the shared components and properties. For example, both "monoid" and "semi group" have an underlying "set" and an associative binary operator. Thus, mathematics text may coerce either structure to its carrier set in formal argumentation and make statements that are true for both structures (Ganesalingam, 2008). Second, phenomena like polysemy and synonymy are frequent in mathematics. Lexically distinct terms can end up referring to the same concept: one name of the concept might come from its inventor, whereas another name is lexically descriptive (e.g., "Karoubi envelope" is synonymous to the type "category of idempotent arrows"). Third, concepts in mathematics can be abstracted using constructs from various mathematical frameworks. Often, a correspondence between concepts can be formed across different theories, each with its own palette of types. For example, a "magma" in group theory is a generalisation of a "groupoid", which can be defined as a special kind of "category" in category theory. Mathematicians have the flexibility to map mathematical concepts between theories, perform reasoning in one theory and then project back to another theory.

Methods that exploit semantic type relatedness, such as TypesExp, can be advantageous in cases where (a) lexical sparsity requires the use of some form of generalisation or similarity across concepts, of which QE is a simple variant, and (b) information about the similarity between types is more informative than similarity between raw words.

6 Conclusions

We have shown experimentally that types are a valuable, but currently underutilised aspect of the mathematical linguistic discourse. Our model improves MIR of research-level queries by automatically identifying types in text and building a similarity (embedding) space with which related types can be detected in documents more effectively than with existing methods. We find that type-based query expansion using this method of semantic proximity outperforms state-of-the-art IR/expansion models on a realistic, large-scale test collection. This strongly suggests that it is the identification of semantic relationships between types that can improve the quality of research-level MIR in the future even further. As an additional advantage, prior knowledge of types may also help improve parts of an NLP pipeline for mathematics texts, particularly in the absence of domain-specific training material.

References

Nasreen Abdul-jaleel, James Allan, W. Bruce Croft, O Diaz, Leah Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade. 2004. Umass at trec 2004: Novelty and hard. In *In Proceedings of TREC-13*.

Akiko Aizawa, Michael Kohlhase, and Iadh Ounis. 2013. Ntcir-10 math pilot task overview. In *Proceedings of the 10th NTCIR Conference*, June.

Akiko Aizawa, Michael Kohlhase, Iadh Ounis, and Moritz Schubotz. 2014. NTCIR-11 math-2 task overview. In *Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies, NTCIR-11, National Center of Sciences, Tokyo, Japan, December 9-12, 2014*.

Ronan Cummins, Jiaul H. Paik, and Yuanhua Lv. 2015. A pÓlya urn document language model for improved information retrieval. *ACM Trans. Inf. Syst.*, 33(4):21:1–21:34, May.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many rater. *Psychological Bulletin*, 76:378–382.

Katerina T. Frantzi, Sophia Ananiadou, and Jun-ichi Tsujii. 1998. The c-value/nc-value method of automatic recognition for multi-word terms. In *Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries*, ECDL '98, pages 585–604, London, UK, UK. Springer-Verlag.

Mohan Ganesalingam. 2008. *The Language of Mathematics*. Ph.D. thesis, Cambridge University Computer Laboratory.

Mihai Grigore, Magdalena Wolska, and Michael Kohlhase. 2009. Towards context-based disambiguation of mathematical expressions. In *The joint conference of ASCM 2009 and MACIS 2009. 9th international conference on Asian symposium on computer mathematics and 3rd international conference on mathematical aspects of computer and information sciences, Fukuoka, Japan, December 14–17, 2009. Selected papers.*, pages 262–271. Fukuoka: Kyushu University, Faculty of Mathematics.

Ferruccio Guidi and Claudio Sacerdoti Coen. 2015. A survey on retrieval of mathematical knowledge. *CoRR*, abs/1505.06646.

Radu Hambasan, Michael Kohlhase, and Corneliu-Claudiu Prodescu. 2014. Mathwebsearch at NTCIR-11. In *Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies, NTCIR-11, National Center of Sciences, Tokyo, Japan, December 9-12, 2014*.

Stephen P. Harter. 1975. A probabilistic approach to automatic keyword indexing. part i. on the distribution of specialty words in a technical literature. *Journal of the American Society for Information Science*, 26(4):197–206.

John S. Justeson and Slava M. Katz. 1995. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(1):9–27.

Michael Kohlhase and Corneliu-Claudiu Prodescu. 2013. Mathwebsearch at NTCIR-10. In *Proceedings of the 10th NTCIR Conference on Evaluation of Information Access Technologies, NTCIR-10, National Center of Sciences, Tokyo, Japan, June 18-21, 2013*.

Giovanni Yoko Kristianto, Minh quoc Nghiem, Yuichiroh Matsubayashi, and Akiko Aizawa. 2012. Extracting definitions of mathematical expressions in scientific papers. In *In JSAI*.

Giovanni Yoko Kristianto, Goran Topic, and Akiko Aizawa. 2014. Exploiting textual descriptions and dependency graph for searching mathematical expressions in scientific papers.

Martin Líška, Petr Sojka, Michal Růžička, and Petr Mravec. 2011. Web interface and collection for mathematical retrieval: Webmias and mrec. In Petr Sojka and Thierry Bouche, editors, *Towards a Digital Mathematics Library.*, pages 77–84, Bertinoro, Italy, Jul. Masaryk University.

Yuanhua Lv and ChengXiang Zhai. 2009. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1895–1898, New York, NY, USA. ACM.

Yuanhua Lv and ChengXiang Zhai. 2011. Lower-bounding term frequency normalization. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 7–16, New York, NY, USA. ACM.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Nidhin Pattaniyil and Richard Zanibbi. 2014. Combining TF-IDF text retrieval with an inverted index over symbol pairs in math expressions: The tangent math search engine at NTCIR 2014. In *Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies, NTCIR-11, National Center of Sciences, Tokyo, Japan, December 9-12, 2014*.

Minh Nghiem Quoc, Keisuke Yokoi, Yuichiroh Matsubayashi, and Akiko Aizawa. 2010. Mining coreference relations between formulas and text using wikipedia.

S. E. Robertson and S. Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 232–241, New York, NY, USA. Springer-Verlag New York, Inc.

S. E. Robertson, C. J. van Rijsbergen, and M. F. Porter. 1981. Probabilistic models of indexing and searching. In *Proceedings of the 3rd Annual ACM Conference on Research and Development in Information Retrieval*, SIGIR '80, pages 35–56, Kent, UK. Butterworth & Co. #38.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at trec-3. In Donna K. Harman, editor, *TREC*, volume Special Publication 500-225, pages 109–126. National Institute of Standards and Technology (NIST).

Mark D. Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 623–632, New York, NY, USA. ACM.

Petr Sojka and Martin Líška. 2011. The art of mathematics retrieval. In Frank Wm. Tompa Matthew R. B. Hardy, editor, *Proceedings of the 2011 ACM Symposium on Document Engineering*, pages 57–60, Mountain View, CA, USA. ACM.

Yiannos Stathopoulos and Simone Teufel. 2015. Retrieval of research-level mathematical information needs: A test collection and technical terminology experiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 334–340.

Ellen M. Voorhees and Donna K. Harman. 2005. *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*. The MIT Press.

Chengxiang Zhai and John Lafferty. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 334–342, New York, NY, USA. ACM.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, April.

Text Retrieval by Term Co-occurrences in a Query-based Vector Space

Eriks Sneiders
Department of Computer and Systems Sciences
Stockholm University
Postbox 7003, SE-164 07, Kista, Sweden
eriks@dsv.su.se

Abstract

Term co-occurrence in a sentence or paragraph is a powerful and often overlooked feature for text matching in document retrieval. In our experiments with matching email-style query messages to webpages, such term co-occurrence helped greatly to filter and rank documents, compared to matching document-size bags-of-words. The paper presents the results of the experiments as well as a text-matching model where the query shapes the vector space, a document is modelled by two or three vectors in this vector space, and the query-document similarity score depends on the length of the vectors and the relationships between them.

1 Introduction

Vector-space in text retrieval is old news. Cosine similarity by Salton and McGill (1986), along with the probabilistic retrieval model by Robertson and Spark Jones (1976), has dominated text retrieval for the last three decades. Much development has focused on improving the relevance scoring factors beyond term frequency and inverted document frequency, e.g., co-occurrence and proximity of query terms in the document help improve the relevance score (see Section 6).

This paper reports the results of an inductive research. We started with a practical task, then analyzed the prototype and spotted a text retrieval model, then twisted the model in order to improve it. The practical task was to increase automation of the "ask us" function on a municipal website: before the user submits his or her message, the system checks whether some answer-relevant pages are published on the website; if yes, the system delivers the pages to the user. Reuse of previously published information would save the user's time and the municipality's resources.

The municipality gave us 25 anonymized sample messages to experiment with, which was not enough for machine learning and text categorization methods. We knew that text-pattern matching yields superior results in automated email answering (Sneiders, 2016). Unfortunately, text patterns require learning, good text patterns are difficult to develop and maintain. We needed something new.

Our research problem is matching email-style query messages to webpages by a technique that does not require learning or development of a knowledge base. Our solution is a text-matching technique that relies on term co-occurrence in a limited text chunk (a sentence or a paragraph), yet without the complexity of good text patterns. The novelty and contribution of this paper are: (i) demonstration of the advantage of such term co-occurrence in text matching over matching of document-size bags-of-words, and (ii) a text-matching model where the query shapes the vector space, a document is modelled by two or three vectors in this vector space, and the query-document similarity score depends on the length of the vectors and the relationships between them. The model is easy to visualize.

Further in this paper, the next section presents the initial prototype, the test data, and the results of the initial tests. Sections 3 and 4 explain the text-matching model and show new test results. Section 5 discusses the advantages and limitations of the technique. Section 6 presents related and further research. Section 7 concludes the paper.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,
pages 2356–2365, Osaka, Japan, December 11-17 2016.

2 Initial Experiment

While web search is a well-established industry, matching email-style messages to web documents is a novel task. The main difference between the two tasks lies in the semantic strength of the terms in the query: while search queries consist of consciously selected keywords, a message is a short story without any indication which words embody the essence of the story.

2.1 Text Matching Method

The query – an email-style message – is a bag of words. The document is a webpage split into text chunks, where a chunk is a sentence or a block of text inside HTML block-level elements (Mozilla, 2016). Each text chunk is a bag of words. Thus, our "document collection" is a collection of text chunks. We do keep track of which chunk belongs to which document. Calculation of the query-document similarity score takes five steps; these steps were experimentally developed and refined.

Step 1. The system matches the query to each chunk separately and discovers term pairs – two terms that co-occur in the chunk and co-occur in the query. From now on, only these terms pairs are significant, single terms are ignored. We do not consider term triples, quadruples, etc. A triple is two term pairs.

Step 2. Each term t_i that belongs to a term pair T in a chunk *ch* obtains a term weight by formula (1). The term weight includes term frequency in the chunk, terms frequency in the query, inverted chunk frequency of the term, which is a counterpart of inverted document frequency in our chunk collection, as well as inverted chunk frequency of the term pair. The latter means that a term may have different weights in one chunk but in different term pairs.

$$w(t_i, T, ch) = tf(t_i, ch) \cdot tf(t_i, q) \cdot icf(t_i) \cdot icf(T) \quad (1)$$

We experimented with the document structure attribute – title, the main text, or H1-H3 heading – for term weights and discovered that term pairs appear in these structure elements somewhat equally for relevant and non-relevant documents. Therefore we do not use the document structure attribute.

Step 3. The weight of a term t_i in a term pair T for the entire document *d* is calculated by formula (2) as a sum across the chunks in the document where the term pair exists.

$$w(t_i, T, d) = \sum_{\{ch \in d | T \in ch\}} w(t_i, T, ch) \quad (2)$$

Step 4. The weight of a term pair $T(t_i, t_j)$ in the document *d* is the sum of the weights of both terms t_i and t_j calculated by formula (3).

$$w(T(t_i, t_j), d) = w(t_i, T, d) + w(t_j, T, d) \quad (3)$$

Step 5. The query-document similarity score is the sum of all term-pair weights of this document calculated by formula (4). The document score is largely a sum of selected term weights, but we did the summation following certain logic.

$$score_0(d) = \sum_{\{T \in d\}} w(T, d) \quad (4)$$

Alternative steps 4 and 5. We may want to skip weighing term pairs and continue with the weights of individual terms from the term pairs. Formula (5) calculates the weight of a term t_i in the entire document *d* across all the term pairs T where t_i exists. The query-document similarity score is the sum of the weights of the unique terms from all the term pairs in the document, calculated by formula (6). N is the number of these unique terms. While formulas (4) and (6) are equivalent, formulas (3) and (4) are easier to understand whereas formulas (5) and (6) will be used further in this paper.

$$w(t_i, d) = \sum_{\{T \in d | t_i \in T\}} w(t_i, T, d) = tf(t_i, q) \cdot icf(t_i) \cdot \sum_{\{T \in d | t_i \in T\}} \left(icf(T) \cdot \sum_{\{ch \in d | T \in ch\}} tf(t_i, ch) \right) \quad (5)$$

$$score_0(d) = \sum_{i=1}^{N} w(t_i, d) \quad (6)$$

2.2 Experiment Data

The source of the document collection was a municipal website in Swedish. The web pages were generated by a content management system and had quite uniform appearance. With a little bit of experimenting with learned to remove the headers, footers, side bars. After removing duplicate content pages, we obtained 2607 documents. These documents were split into chunks. If a chunk was a block of text, the collection of chunks got 26 064 chunks. If a chunk was a sentence, the collection got 37 420 chunks. We applied compound splitting (Sjöbergh and Kann, 2004) and lemmatization (Carlberger and Kann, 1999) to the words in a chunk before the chunk was turned into a bag of words.

We tested two options of stop-word removal. One option was using a standard list of Swedish stop-words (Text Tools, 2016). The other option was part-of-speech (POS) filtering. Analysis of 30 most frequent lemmas that matched text patterns in automated email answering (Sneiders at al., 2014) showed that only nouns and verbs were domain-specific words. Barr et al. (2008) explored Yahoo! search queries and found that 40.2% of the terms were proper nouns, 30.9% nouns, 7.1% adjectives, 2.4% verbs. Inspired by these findings, we grouped the terms in our document collection by their POS-tags and subjectively examined which parts-of-speech felt like good candidates for descriptive term pairs. Eventually, we decided that only nouns, verbs, proper nouns, adjectives, and unrecognized terms should be left in the documents after POS filtering.

The average size and standard deviation of the block-chunks were 8.9±8.45 unique lemmatized terms after removing standard stop-words, and 8.29±7.71 terms after POS filtering; both after compound splitting. For sentence-chunks, the respective sizes were 6.44±4.21 and 6.01±3.93.

We had 25 email-style messages in Swedish as test queries. Each message was spell-checked and pre-processed as a chunk. The average size of the messages was 12.68±6.25 unique lemmatized terms after removing standard stop-words, and 10.76±4.90 terms after POS filtering. Following are three sample messages – one short, one long, and one medium long. We Google-translated the Swedish originals, with minor manual corrections, and removed location names from the text.

- "Hello, I'm looking for a summer job as a gardener in <town> but I do not know which homepages I can go in and look for summer jobs. Which homepages I can go to and check?"
- "Hey, we're moving to <town> municipality in July. Today we live in <town> and have our children at <name> school. The new house is closest to <name> school but this is another municipality. When I spoke to the school they told me that our children were welcome there and they now have a number of students who attend the school but live across municipal borders. It requires, however, an approval of <town> municipality. My question now is how it works, if we want our children to go to <name> school because it is closest, and that several of the children's friends go to that school too. Thanks in advance."
- "Hello! On the pedestrian and bicycle path between <address> in <town> and <village> is a lamppost partially bent since a storm last fall. The post is standing but the light is directed in a wrong direction. Who is responsible for this to be corrected? Thanks for a quick response. If the lamppost number is needed I can find it out."

We used the website's search engine to find documents relevant to the test messages. We devised a number of search queries per test message and manually examined all the retrieved documents. We found two kinds of documents. *Closely relevant documents* could answer the message. *Related documents* had "good to know" information but did not answer the message. 3 of the test messages had no relevant information on the website. 4 messages had only closely relevant documents. 4 messages had only related documents. 14 messages had both closely relevant and related documents. Of those messages that had closely relevant documents, 8 messages had one such document and 10 had two such documents. Of those messages that had related documents, 15 messages had one to three such documents, 2 had six such documents, and 1 had eleven such documents.

2.3 Performance Measures

The system matches term pairs in the query and the document, therefore it inevitably removes relevant documents that do not have matching term pairs. Hence, we measured *recall* as the share of relevant documents that did have matching term pairs and therefore were included in the results. The average recall was calculated across the queries that had corresponding documents in the collection.

In order to quantify exposure of relevant documents to the user, we measured the *average rank* (the position in the result list) of relevant documents. We calculated the average rank for each query with non-zero recall, and then the average of these averages.

In order to simulate automated email answering, we measured *P@1* (precision among top n documents where n is 1) for closely relevant documents. The average P@1 was calculated across the queries with non-zero recall.

Our baseline method was *cosine similarity* between unigram-based document vectors. We tested cosine similarity in two settings. At first, we applied cosine similarity to all the 2607 documents and ranked them. The second option was to filter the 2607 documents by the term pairs first, which resulted in a much smaller set of documents with a higher density of relevant documents. Then we applied cosine similarity in this smaller set of documents. For cosine similarity, term frequency was counted in the entire document; inverted document frequency was calculated in the document collection. We realize that cosine similarity is a rather basic baseline. It did, however, serve the purpose. We leave a more thorough comparison of text-matching methods to further research.

Recall, the average rank of relevant documents, and cosine similarity were measured separately for closely relevant and related documents.

2.4 Test Results

The tests have three parameters – (i) two options of stop-word removal, (ii) two scopes of a text chunk, and (iii) two options of the minimum number of term pairs that the entire document (not one chunk) and the query must have in common for the document to be considered for relevance ranking. Table 1 shows the initial results according to these parameters. The average values have their standard deviation, except for P@1 where individual values are 0 or 1.

Recall and the number of selected documents. The average number of selected documents lies between 25 and 109 of the total 2607, i.e., on average the system selected 1-4% of the documents. Meanwhile, the average recall does not drop below 0.61 for closely relevant documents. The best average recall is 0.83, i.e., the selected 74 (average) of the 2607 documents contain 83% (average) of all the closely relevant documents. Recall for the related documents is lower. Because the semantic distance between the query and a related document is longer than that between the query and a closely relevant document, apparently there are fewer co-occurring terms and fewer selected messages.

Stop-words	**Avg num of selected docs**	**Avg recall**		**Avg P@1**	**Avg rank**	
		Close	**Related**		**Close**	**Related**
Documents selected by chunk-sentence, min 2 common term pairs. Ranked by $score_0$						
POS filter	41.44±98.31	0.75±0.34	0.56±0.40	0.50	3.94±4.96	4.54±4.83
Standard	**74.28±204.48**	**0.83±0.33**	**0.58±0.38**	**0.56**	**3.47±3.87**	**6.51±6.84**
Documents selected by chunk-block, min 2 common term pairs. Ranked by $score_0$						
POS filter	63.36±139.49	0.75±0.34	0.57±0.39	0.50	4.66±6.04	8.50±10.17
Standard	108.52±264.91	0.83±0.33	0.61±0.36	0.50	4.19±4.52	11.64±16.63
Documents selected by chunk-sentence, min 3 common term pairs. Ranked by $score_0$						
POS filter	25.16±64.2	0.61±0.36	0.32±0.41	0.53	2.73±3.47	4.83±5.56
Standard	47.84±147.54	0.69±0.34	0.31±0.41	0.56	2.16±1.50	5.22±5.91
Documents selected by chunk-block, min 3 common term pairs. Ranked by $score_0$						
POS filter	43.20±109.47	0.67±0.37	0.41±0.42	0.53	4.63±6.07	6.95±8.68
Standard	77.28±217.2	0.78±0.34	0.41±0.41	0.50	3.97±4.48	7.61±9.83
Cosine similarity among all the documents						
POS filter	All 2607	1.00	1.00	0.44	34.42±52.09	83.29±125.89
Standard	All 2607	1.00	1.00	0.44	27.58±47.45	74.41±107.60
Cosine similarity after the documents were selected by chunk-sentence, min 2 common term pairs						
POS filter	41.44±98.31	0.75±0.34	0.56±0.40	0.56	6.34±12.59	5.50±5.33
Standard	74.28±204.48	0.83±0.33	0.58±0.38	0.63	7.16±13.95	7.09±9.01

Table 1. Results of the initial tests.

Average rank. For term-pair matching, the average rank (i.e., average of the averages, see Section 2.3) of closely relevant documents lies between 2.16 and 4.66 for different document selection options. The average rank of related documents is lower than that of closely relevant documents, as expected.

Cosine similarity in the entire document collection demonstrates rather low average ranks – 27.58 and 34.42. If, however, we first filter the collection by term pairs and then apply cosine similarity to the documents that have survived the filtering, then the average ranks are comparable with those by $score_0$. For cosine similarity after document filtering, the average rank of related documents is higher than that of closely relevant documents, although it should be the other way around. One reason for that could be the low recall for related documents and fewer documents that drag the average rank down. Cosine similarity disregards the added value of term pairs that helps $score_0$ distinguish between closely relevant and related document.

P@1 lies around 0.5, which means that roughly each second top document is a correct answer. This is too little if we want to use our technique for fully automated email answering with a relevant webpage as the answer. Notably, cosine similarity yields the best P@1 value 0.63. Because the average ranks for cosine similarity have large standard deviation, cosine similarity offers higher gains (also better P@1) and bigger losses (also lower average rank) than $score_0$.

We selected the result typed boldfaced in Table 1 as the *best initial result and the baseline* for further experiments with term-pair matching: the best recall for closely relevant documents (0.83) and the second best recall for related documents (0.58), while having the third best and still good average ranks (3.47 and 6.51).

2.5 Amount of Matching Text

In order to better understand term-pair matching, we counted the number of term pairs and unique terms per query-document match for closely relevant documents for the best initial result in Table 1. Of the total 28 closely relevant documents, 24 were selected by the system during the test. Figure 1 shows the number of term pairs and unique terms from these term pairs for each of the 24 successful query-document matches. In most cases, a query-document match required no more than 6-7 term pairs; the largest number of term pairs was 26. The number of unique terms in these term pairs was also up to 6-7 per match, 13 at most. In eight matching cases, the number of terms in the matching term pairs was only 3.

On average, the terms in the term pairs accounted for about 40% of the unique query terms and about 3% of the unique document terms after compound splitting, lemmatization, and stop-word removal. That means that quite a large portion of the query but a small portion of the document participated in a successful query-document match.

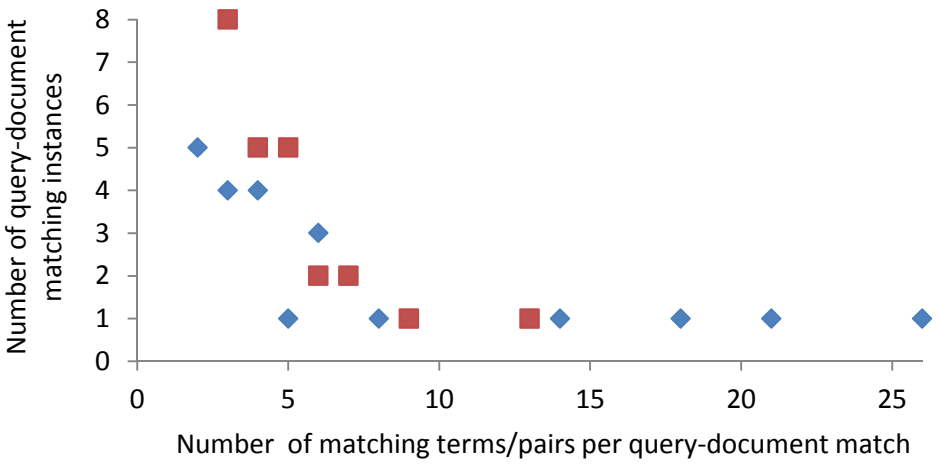


Figure 1. The number of term pairs (diamonds) and unique terms (squares) from these term pairs per query-document match, and the number of the corresponding matching instances. At (7; 2) a square hides a diamond, they overlap.

3 Text-Matching Model Based on Term Pairs in a Query-based Vector Space

While experimenting with the prototype, we designed formulas (1) to (6) according to our own perception of text relevance. Soon we realized that the formulas do have a representation in vector algebra. Let us imagine a term pair in the document *d* as a vector T_j; the term pair originates from one or several chunks of *d*. T_j has two coordinates $t_{ji} = w(t_i, T_j, d)$ each calculated by formula (2). The document vector P is the sum of all the term-pair vectors T_j as shown in Figure 2 (a). The document vector P has coordinates $p_i = w(t_i, d)$ each calculated by formula (5).

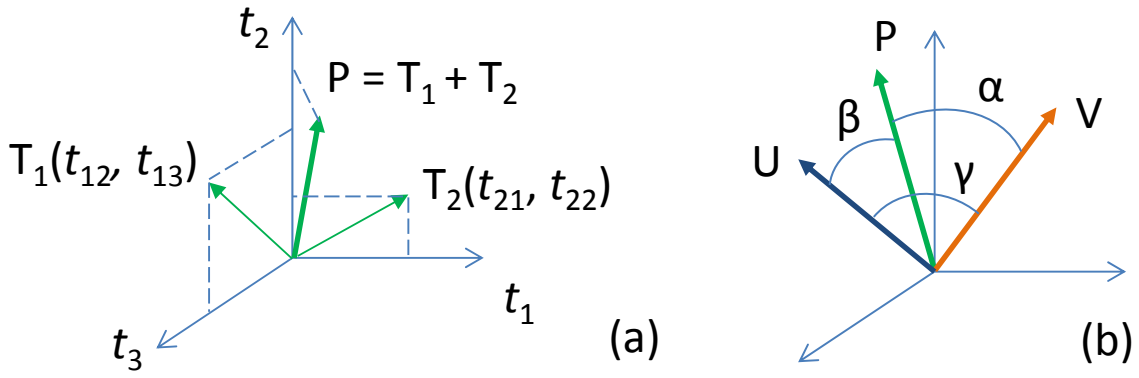


Figure 2. (a) Document vector P as the sum of term-pair vectors. (b) Document vector P, vocabulary vector V, unigram vector U (used in Section 4), and the angles between them.

Traditionally in a vector space, the query and the document are two independent vectors. In Figure 2 we do not have any query vector; the query is not clearly visible. Still, the query is in there. Together with each individual document, the query establishes term pairs whose terms define the dimensions of the vector space. Furthermore, the query participates in the term weight formula (1) and, thus, contributes to the coordinates of P. The query provides the building blocks for the document vector P. *The query is embedded in the vector space and shapes the vector space*: limited dimensions of the vector space and query-term frequency in the coordinates of the term-pair vectors are the only presence of the query in the text-matching process.

How do we calculate similarity between the query, which is embedded in the vector space, and the document, which is a vector? One measure could be the length of P: longer P means more query participates in the document, which means more similarity. Another measure could be the volume of the polygon created by P: the more the document "fills" the query the more similar they are. We tested both measures; the average ranks were inferior to those of the best initial result in Table 1.

Let us examine why simple summation of vector coordinates works better than the length and volume scores. We can rewrite $score_0$ in formulas (5) and (6) as shown in formulas (7) and 8:

$$score_0(d) = \sum_{i=1}^{N} p_i \cdot 1 = P \bullet V = |P||V| \cos\alpha \tag{7}$$

$$p_i = w(t_i, d) = tf(t_i, q) \cdot icf(t_i) \cdot \sum_{\{T \in d | t_i \in T\}} \left(icf(T) \cdot \sum_{\{ch \in d | T \in ch\}} tf(t_i, ch) \right) \tag{8}$$

N is the number of dimensions of P, which is the number of unique terms in the term pairs. That number is not particularly large, as we see in Figure 1. V is the vocabulary vector, it has the same dimensions as P but the coordinates are 1. The vectors P and V and cos α in formula (7) are illustrated by Figure 2 (b). $Score_0$ favors the following query-document match conditions:

- | P | grows if p_i grows, i.e., (i) the query and the document have more term pairs in common and (ii) each term pair is more document-unique with higher inverted chunk frequency, (iii) each term in the term pairs has higher frequency in the document and (iv) in the query, and (v) the term is also more document-unique with higher inverted chunk frequency.
- $|V| = \sqrt{N}$, it grows along with the diversity of the vocabulary in the term pairs.

- cos α between vectors P and V is highest if all p_i values are the same, because vector V has each coordinate 1. Equal p_i values suggest that all terms in the term pairs should be equally important, where the importance is maintained by the balance between the frequency of the term in the query and the document, its uniqueness in the document collection, and uniqueness of its co-occurrences with other terms (i.e., term pairs). Less frequent terms require more unique co-occurrences.

In formula (7), the vocabulary vector V states that all terms in the term pairs are equal. Let us change it so that each coordinate of V is inverted chunk frequency of the term, as in formulas (9) and (10). Now the p_i values are not expected to be equal but instead be bigger for more unique terms in order to maintain good alignment between vectors P and V.

$$score_1(d) = P \bullet V = \sum_{i=1}^{N} p_i v_i \tag{9}$$

$$v_i = icf(t_i) \tag{10}$$

We tested formula (9) with the same options that our best initial result in Table 1 had: removal of standard stop-words, a chunk is a sentence, the query and the document must have at least 2 term pairs in common. Table 2 shows that $score_1$ yields better average ranks than the best initial result.

Formula	**Avg P@1**	**Avg rank**	
		Close	**Related**
$score_1$ (9)	0.50	3.38±3.93	6.24±6.81

Table 2. Query-document similarity score based on term pairs in a query-based vector space. (The corresponding number of selected documents and the recall values are boldfaced in Table 1.)

4 The Model Enhanced by Unigrams from the Document

Roughly 3% of the document terms participated in a successful query-document match, as stated in Section 2.5. Arguably, term co-occurrences in limited text chunks identify relevant parts of the document and use only these parts to score the document. Imagine an FAQ list. There are many FAQs and their answers in the document, but only one FAQ and its answer contains the right term co-occurrences. The system uses that FAQ to score the document and ignores the rest of the list.

What happens if we include all the vocabulary of the document into our query-document similarity score? If the document is concise, it may help. If we have a multi-topic document, the non-co-occurring and possibly not so relevant terms from the entire document are likely to damage the score. Let us try and see what happens.

Figure 2 (b) shows the new arrangement. We have the old vectors P and V, as well as a new unigram vector U, which is the traditional document term vector. For the highest score, P and V are aligned (α is small), as well as U and P are aligned (β is small), or U and V are aligned (γ is small). Good alignment between U and P or V requires that most document terms end up in the term pairs.

$$score_2(d) = (P \bullet V)\cos\beta = \sum_{i=1}^{N} p_i v_i \frac{\sum_{i=1}^{N} p_i u_i}{\sqrt{\sum_{i=1}^{N} p_i^2}\sqrt{\sum_{i=1}^{N} u_i^2}} \tag{11}$$

$$score_3(d) = (P \bullet V)\cos\gamma = \sum_{i=1}^{N} p_i v_i \frac{\sum_{i=1}^{N} v_i u_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} u_i^2}} \tag{12}$$

$$u_i = tf(t_i, d) \cdot tf(t_i, q) \cdot idf(t_i) \tag{13}$$

Formulas (11), (12), (8), (10) and (13) implement this similarity score. Please observe that u_i – the coordinates of the unigram vector U – include the traditional term frequency in the entire document and inverted document frequency. No chunks. N here is the number of dimensions in the document, which means that many p_i and v_i values are 0; they are non-zero for the dimensions of the term pairs.

Formulas (8) and (13) include query-term frequency $tf(t_i, q)$ in the coordinates of vectors P and U. If we move $tf(t_i, q)$ from the coordinates of P and U to the coordinates of V, then U and V will start resembling the traditional document and query vectors. The dot product P ● V will not change, the angles β and γ will. Let us redefine the coordinates p_i, v_i, and u_i as shown in formulas (14), (15), and (16). The new $score_4$ and $score_5$ are calculated as $score_2$ and $score_3$ but with p'_i, v'_i, and u'_i instead of p_i, v_i, and u_i.

$$p_i^{'} = icf(t_i) \cdot \sum_{\{T \in d | t_i \in T\}} \left(icf(T) \cdot \sum_{\{ch \in d | T \in ch\}} tf(t_i, ch) \right) \tag{14}$$

$$v_i^{'} = tf(t_i, q) \cdot icf(t_i) \tag{15}$$

$$u_i^{'} = tf(t_i, d) \cdot idf(t_i) \tag{16}$$

$$score_4(d) = score_2(d, p^{'}, v^{'}, u^{'}) \tag{17}$$

$$score_5(d) = score_3(d, p^{'}, v^{'}, u^{'}) \tag{18}$$

We tested the formulas with the same text processing and document selection options as in Section 3. Table 3 shows the results. In comparison with $score_1$, the average P@1 has improved in 3 of 4 cases. Notably, cosine similarity yielded the best P@1 in Table 1; now cosine similarity is somewhat smuggled in by the use of cos β and cos γ. As to the average ranks of closely relevant documents, $score_1$ with only term pairs and without the unigram vector U had a marginally better value. For related documents, vector U was beneficial for $score_4$ and $score_5$ where query-term frequency was in the coordinates of vector V.

Formula	Avg P@1	Avg rank	
		Close	Related
$score_2$ (11)	0.56	3.44±4.17	6.18±6.84
$score_3$ (12)	0.44	3.66±4.04	6.06±7.51
$score_4$ (17)	0.56	4.16±4.40	4.61±4.30
$score_5$ (18)	0.56	3.56±3.26	4.60±3.98

Table 3. Query-document similarity based on term pairs and unigrams from the document.
(The corresponding number of selected documents and the recall values are boldfaced in Table 1.)

5 Advantages and Limitations

Our term-pair-based text-matching technique has three *advantages*. First of all, it has a good average rank for closely relevant messages; we measured 3.38±3.93 at average recall 0.83±0.33. The second advantage is low maintenance costs, because the technique does not have any knowledge base to develop and maintain. Finally, we may argue that term co-occurrence identifies relevant parts of the document and ignores the rest (only about 3% of the document terms are used), which means the technique works pretty well with multi-topic documents.

The first *limitation* is email-size queries. The query is big enough to shape a vector space and small enough to be one text chunk. We have not yet considered this technique for other query sizes. Furthermore, we know that a message sent to a contact center is keyword-rich text because it describes the context and requires a response (Sneiders, Sjöbergh and Alfalahi, 2016). We do not know how the technique would work with queries where keyword density is lower than that of email-style messages.

The design of the technique had a question-answering task in mind. In our experiments, term-pair matching worked best for closely relevant documents. Topic-related text retrieval and categorization, which most of text retrieval is, does not deal with the concept of answer.

Because a successful query-document match relies on rather few term pairs (see Section 2.5), the technique relies on good text pre-processing tools – spelling correction, compound splitting, lemmatization – which lay the groundwork for a larger number of term pairs.

The average P@1 value around 0.5 is nice but not sufficient for fully automated email answering.

6 Related and Future Research

Utilizing term co-occurrence in a text chunk is not a new phenomenon in email answering at contact centers. Malik at al. (2007) match sentences in a query message to tag-questions (like FAQs) attached to standard answers. Marom and Zukerman (2005) cluster answers given to similar inquiries, and then pick out most representative sentences from the answers in one cluster in order to build a model answer. Sneiders, Sjöbergh and Alfalahi (2016) match manual text patterns to email messages for assigning standard answers. A text pattern looks for co-occurring terms in a paragraph of the query; a phrase in the text pattern considers the sequence and distance between the terms in a sentence.

Corpus-based question answering (QA) relies on term co-occurrence in a limited piece of text. Let us consider one of the earlier works (Kwok et al., 2001) which pinpoints the main steps of QA. Retrieval of source documents for candidate answers requires good search queries, which means a good set of co-occurring terms. Answer extraction involves locating regions in the documents with high density of the search keywords. Finally, answer selection calculates the weight of each candidate answer, which is a text snippet, based on co-occurrence of many and relevant terms in the snippet.

Term co-occurrence in a document (not a chunk) has been used to build new text categorization features, along with the unigrams (Figueiredo et al, 2011). The logic there is the same as ours – term co-occurrence creates new meaning that is not visible if we consider each term separately.

Proximity of the query terms in a document has been used to enhance the probabilistic retrieval model in BM25 for web search (e.g., He et al., 2011; Svore et al., 2010). Song at al. (2008) identify text spans that contain query terms in the document, and calculate relevance contribution of a span with respect to each query term in there. The relevance contribution is used instead of term frequency in the BM25 document score.

As for our own text-matching model, it needs tests with more data and more baseline systems in order to see whether the perceived value of the model is real.

Synonyms and semantic distance between terms is something to test. They would increase the number of both relevant and non-relevant term pairs. Domain-specific synonyms are likely to help, but that would mean developing a "knowledge base", something we wanted to avoid in the beginning.

Sneiders, Sjöbergh and Alfalahi (2016) have observed that the essence of the message comes in one sentence in half of the email messages sent to a contact center. If we learn to identify that sentence, then we can greatly reduce the number of non-relevant term pairs coming from the rest of the message.

So far we did not consider co-occurrence of more than 2 terms in a combination – a triple is two pairs. Whether the added value of a new term in the combination of terms is linear (the sum of pairs) or not, we do not know it yet.

Previously we argued that term-pair matching identifies the most relevant parts of a multi-topic document and calculates query-document similarity with respect to those parts. Currently the entire document is retrieved to the user the way search engines do, whereas a better option might be extracted relevant parts of the multi-topic document.

7 Conclusion

Traditionally in text retrieval, presence of terms in a document is the main information carrier. We have shown that term co-occurrence in a limited text chunk (a sentence or paragraph) is an information carrier in its own right. In our experiments, only about 3% of unique document terms were used to successfully match the document to the query (Section 2.5). These 3% came from the term pairs that text chunks in the document had in common with the query.

Table 1 shows the success of our term pair matching for document *filtering*. For example, the selected 74 (on average) out of 2607 documents yielded 83% (on average) recall for the closely relevant documents. Ranking among a few selected documents with high density of relevant documents is more effective than ranking in the entire document collection.

Furthermore, we used the term pairs to score the documents for *relevance ranking*. At the 83% recall, the system could reach the average rank of closely relevant documents 3.38 (Table 2) while cosine similarity in the same settings yielded 7.16 (Table 1).

Our main academic contribution is a simple but effective *text-matching model* where the query shapes the vector space and a document is modelled by two or three vectors. (i) The document vector is the sum of individual term-pair vectors; it favors many and subject-specific term pairs with subject-relevant terms. (ii) The vocabulary vector favors a large diversity of vocabulary in the term pairs and equally important terms in the term pairs. (iii) The optional unigram vector favors matches where most of the document terms end up in the term pairs. The model is easy to visualize.

Reference

Cory Barr, Rosie Jones and Moira Regelson. 2008. The Linguistic Structure of English Web-Search Queries. Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics, 1021-1030.

Johan Carlberger and Viggo Kann. 1999. Implementing an efficient part-of-speech tagger. Software-Practice and Experience, 29(9):815-32.

Fábio Figueiredo, Leonardo Rocha, Thierson Couto, Thiago Salles, Marcos André Gonçalves and Wagner Meira Jr. 2011. Word co-occurrence features for text classification. Information Systems, 36(5):843-858.

Ben He, Jimmy Xiangji Huang and Xiaofeng Zhou. 2011. Modeling term proximity for probabilistic information retrieval models. Information Sciences, 181(14), pp.3017-3031.

Cody Kwok, Oren Etzioni and Daniel S. Weld. 2001. Scaling question answering to the web. ACM Transactions on Information Systems (TOIS), 19(3):242-262.

Rahul Malik, L. Venkata Subramaniam and Saroj Kaushik. 2007. Automatically Selecting Answer Templates to Respond to Customer Emails. IJCAI, Vol. 7, 1659-1664.

Yuval Marom and Ingrid Zukerman. 2005. Towards a framework for collating help-desk responses from multiple documents. Proceedings of the IJCAI05 Workshop on Knowledge and Reasoning for Answering Questions, 32-39.

Mozilla. 2016. Block-level elements. https://developer.mozilla.org/en/docs/Web/HTML/Block-level_elements Accessed in July 2016.

Stephen E. Robertson and K. Sparck Jones. 1976. Relevance weighting of search terms. Journal of the American Society for Information Science, 27(3):129-146.

Gerard Salton and Michael J. McGill. 1986. Introduction to Modern Information Retrieval. McGraw-Hill, Inc. New York, NY, USA.

Jonas Sjöbergh and Viggo Kann. 2004. Finding the Correct Interpretation of Swedish Compounds, a Statistical Approach. Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC), 899–902.

Eriks Sneiders, Gunnar Eriksson and Alyaa Alfalahi. 2014. Exploring the Traits of Manual E-Mail Categorization Text Patterns. Advances in Natural Language Processing. Springer International Publishing, 337-344.

Eriks Sneiders. 2016. Review of the Main Approaches to Automated Email Answering. Advances in Intelligent Systems and Computing. Springer International Publishing, 135-144.

Eriks Sneiders, Jonas Sjöbergh and Alyaa Alfalahi. 2016. Email Answering by Matching Question and Context-Specific Text Patterns: Performance and Error Analysis. Advances in Natural Language Pro-cessing. Springer International Publishing, 123-133.

Ruihua Song, Michael J. Taylor, Ji-Rong Wen, Hsiao-Wuen Hon and Yong Yu. 2008. Viewing term proximity from a different perspective. Advances in information retrieval (proceedings of ECIR 2008). Springer Berlin Heidelberg, 346-357.

Krysta M. Svore, Pallika H. Kanani and Nazan Khan. 2010. How Good is a Span of Terms? Exploiting Proximity to Improve Web Retrieval. Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval. ACM, 154-161.

Text Tools. 2016. Swedish stop words. http://countwordsfree.com/stopwords/swedish . Accessed in July 2016.

Pairwise Relation Classification with Mirror Instances and a Combined Convolutional Neural Network

Jianfei Yu
School of Information Systems
Singapore Management University
jfyu.2014@phdis.smu.edu.sg

Jing Jiang
School of Information Systems
Singapore Management University
jingjiang@smu.edu.sg

Abstract

Relation classification is the task of classifying the semantic relations between entity pairs in text. Observing that existing work has not fully explored using different representations for relation instances, especially in order to better handle the asymmetry of relation types, in this paper, we propose a neural network based method for relation classification that combines the raw sequence and the shortest dependency path representations of relation instances and uses mirror instances to perform pairwise relation classification. We evaluate our proposed models on two widely used datasets: SemEval-2010 Task 8 and ACE-2005. The empirical results show that our combined model together with mirror instances achieves the state-of-the-art results on both datasets.

1 Introduction

Relation classification is a very important task for many Natural Language Processing (NLP) applications including question answering (Yao and Van Durme, 2014), knowledge base population (Socher et al., 2013) and opinion mining (Kobayashi et al., 2007). The goal of relation classification is to automatically identify the semantic relation between a pair of entities in free text. For example, a relation classification system should be able to capture the *Cause-Effect* relation between the entities *pressure* and *burst* in the sentence "The *burst* has been caused by water hammer *pressure*."

Like any classification task, a key research question of relation classification is the identification of a good feature representation for each relation instance. Traditional approaches focus on either combining many manually designed features (Zhou et al., 2005; Jiang and Zhai, 2007; Li and Ji, 2014) or leveraging various kernels to implicitly explore a large feature space (Bunescu and Mooney, 2005; Zhang et al., 2006; Qian et al., 2008; Nguyen et al., 2009), but both approaches suffer from their poor generalization ability on unseen words, and fail to achieve very satisfactory performance (Nguyen et al., 2015). Recently, with the advances of deep learning in NLP, neural networks (NNs) have exhibited their advantages in dealing with unseen words through pre-trained word embeddings and capturing meaningful hidden representations. Different NN architectures, including Convolutional Neural Network (CNN) (Zeng et al., 2014), Recursive Neural Network (ReNN) (Socher et al., 2012) and Recurrent Neural Network (RNN) (Xu et al., 2015b), have been applied to relation classification.

However, most existing NN-based approaches only exploit one of the following structures to represent relation instances: raw word sequences (Zeng et al., 2014; dos Santos et al., 2015), constituency parse trees (Socher et al., 2012; Hashimoto et al., 2013) and dependency parse trees (Xu et al., 2015a; Xu et al., 2015b; Miwa and Bansal, 2016). For the models based on raw sequence, despite maintaining all the information in relation instances, they cannot well handle long-distance relations. For the models based on constituency parse trees, one of the bottlenecks is handling long-distance relations (Ebrahimi and Dou, 2015). For the dependency tree-based models, although they focus on the condensed information

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2366–2377, Osaka, Japan, December 11-17 2016.

captured by the shortest dependency path between the two entities and thus are good at capturing long-distance relations, they lose some supplementary information in the original instance (Liu et al., 2015). Observing that the raw sequence and the dependency path representations highly complement each other, we expect a combination of the two structures to be more effective in capturing long-distance relations without losing any information.

Moreover, another important issue with the feature representation of relation instances is regarding the asymmetry of relation types. Most relation types are asymmetric. Take the *Cause-Effect* relation in the SemEval dataset as an example. *Cause-Effect*(e_1, e_2) indicates that e_1 is the cause and e_2 is the effect. If their roles are reversed, we need to represent the relation as either *Cause-Effect*(e_2, e_1) or *Effect-Cause*(e_1, e_2). Suppose we have K different asymmetric relation types. The current common practice to handle the relation directions is to transform the K+1 class labels (where the +1 is for the *Other* relation, which is symmetric) into $2K$+1 class labels, where each of the K asymmetric relations is expanded into two labels to capture the two directions. For example, from *Cause-Effect*, another label *Effect-Cause* is created. Given any sentence containing two entities, we can always treat the first entity as e_1 and the second entity as e_2. We can then classify their relation into one of the $2K$+1 labels.

Although this approach has been shown to be effective, it neglects the fact that the two class labels corresponding to the same original asymmetric relation are correlated. Take the above-mentioned *burst-pressure* sentence as an example. Most previous methods will treat it as a positive instance for the *Effect-Cause* relation only (because the first entity *burst* in the sentence is the effect). They will not relate the sentence to the *Cause-Effect* relation, although if we treat the second entity *pressure* as e_1, its relation to the first entity *burst* is *Cause-Effect*. We believe that if we represent each relation instance in two ways by swapping the order of the two entities, we can not only implicitly link the pair of relation labels from the same relation but also make a better prediction on a relation instance based on its two representations.

Based on the two observations above regarding the complementary nature of the raw sequence and dependency path representations and the asymmetry of relation types, in this paper, we propose a mirror instance based pairwise relation classification (MI) method using a convolutional neural network that combines raw sequence and dependency path representations. Our MI method creates mirror instances from the original relation instances by swapping the order of the two entities and using the reversed relation label. The method also learns appropriate weights to combine the predictions made on the original instance and the mirror instance for the final prediction.

Evaluation on SemEval-2010 Task 8 and ACE-2005 shows that both mirror instances and combining raw sequence and dependency path representations help improve the performance of relation classification. Our results also show that: (1) by using only half of the negative training instances to generate mirror instances, we can push the F_1 score to 85.0 on SemEval-2010 Task 8 without using any additional manually-crafted, linguistic-driven features; (2) and with only one additional linguistic-driven feature (entity type), we can obtain results competitive with the state-of-the-art results on ACE-2005.

2 Our Proposed Model

In this section, we first formally formulate the task and introduce our notation. We then present our proposed mirror instance method, including the mirror instance generation strategy and our pairwise relation classification framework. Finally, we present our proposed combined CNN models.

2.1 Problem Formulation

A relation instance consists of a sentence with two entities inside tagged as e_1 and e_2. Here e_1 always precedes e_2 in the sentence. Let $\mathcal{R}$ be a set of pre-defined asymmetric relation types, and $\mathcal{S}$ be a set of pre-defined symmetric relations including no relation. A labeled relation instance has a relation label that indicates both the relation existing between the two entities and the direction of the relation. For example, a relation label can be in the form of either $r(e_1, e_2)$ or $r(e_2, e_1)$, where $r \in \mathcal{R} \cup \mathcal{S}$. To

Relation	
Cause-Effect	Effect-Cause
Component-Whole	Whole-Component
Content-Container	Container-Content
Entity-Destination	Destination-Entity
Entity-Origin	Origin-Entity
Instrument-Agency	Agency-Instrument
Member-Collection	Collection-Member
Message-Topic	Topic-Message
Product-Producer	Producer-Product
Other	

Table 1: Relations in SemEval-2010 Task 8.

Relation	
PART-WHOLE	WHOLE-PART
ORG-AFF	AFF-ORG
ART	ART^{-1}
PHYS	PHYS^{-1}
GEN-AFF	AFF-GEN
PER-SOC	
None	

Table 2: Relations in ACE-2005.

make our explanations simpler, we assume that we are always predicting the relation from e_1 to e_2, and therefore for each $r \in \mathcal{R}$, we introduce a reversed relation label $\text{rev}(r)$ to capture the cases when relation r is from e_2 to e_1. For example, if r is *Cause-Effect*, then $\text{rev}(r)$ is *Effect-Cause*. In total, we have $2K$+L class labels, where $K = |\mathcal{R}|$ and $L = |\mathcal{S}|$. For the SemEval-2010 Task 8 data, we list all the $2 \times 9 + 1$ class labels in Table 1, where the $+1$ is for the case when there is no relation, denoted by *Other*. For the ACE-2005 data, all the $2 \times 5 + 2$ class labels are listed in Table 2, where $+2$ indicates the symmetric person-social relation and no relation, denoted by *PER-SOC* and *None*.

We further assume that each relation instance has two kinds of word representations. The first is the raw sequence (RSeq) representation, which consists of the sequence of words in the original sentence. The second is the shorted dependency path (SDP) representation, which is the shortest path from e_1 to e_2 in the dependency parse tree of the original sentence.

Let us use $\mathcal{V}$ to denote the vocabulary that contains all unique words in our dataset and $\mathcal{E}$ the set of directed dependency relation labels such as $\xrightarrow{\text{pobj}}$. The RSeq representation of a relation instance contains a sequence of words $(w_1, w_2, \ldots)$ where $w_i \in \mathcal{V}$. In addition, inspired by the work by Zeng et al. (2014), to tag the positions of e_1 and e_2, we assume that each word w_i in the sequence is associated with two position indices p_i and q_i, which indicate the relative distances of w_i from e_1 and e_2, respectively. Take the token "caused" in the previous *burst-pressure* sentence as an example. Since its relative distance to the two entities "burst" and "pressure" are 3 and -4 respectively, its two position indices are 3 and -4. We use $\mathcal{P}$ to denote the set of all possible position indices in our dataset.

The SDP representation can also be regarded as a sequence of tokens $(t_1, t_2, \ldots)$, where each token is either a word or a directed dependency relation, that is, $t_j \in \mathcal{V} \cup \mathcal{E}$. Similar to the RSeq representation, we also use the relative distances of t_j to e_1 and e_2 to indicate the positions of e_1 and e_2, namely c_j and d_j. The left side of the bottom layer of Figure 2 shows the RSeq and the SDP representations of the relation instance "The $[\textit{burst}]_{e_1}$ has been caused by water hammer $[\textit{pressure}]_{e_2}$."

Formally, we assume that we are given a set of labeled relation instances $\{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$, where $y^{(n)}$ is a relation label and $x^{(n)}$ has two kinds of word representations: $\text{RSeq}(x^{(n)})$ and $\text{SDP}(x^{(n)})$.

2.2 Mirror Instance Method

Our first proposal is a new framework to model each relation instance by a pair of representations. The key idea is to first generate a *mirror* instance from each original relation instance, and then perform joint training and testing by making use of both the original and the mirror instances.

Our method is motivated by the observation that each relation instance can provide us with a pair of examples with opposite directions. For example, "The $[\textit{burst}]_{e_1}$ has been caused by water hammer $[\textit{pressure}]_{e_2}$." is an original relation instance and is labeled as *Effect-Cause*. If we swap the order of e_1 and e_2, then the resulting mirror instance "The $[\textit{burst}]_{e_2}$ has been caused by water hammer $[\textit{pressure}]_{e_1}$." should be labeled as *Cause-Effect*. Recall that in standard practice the relation labels *Cause-Effect* and *Effect-Cause* are treated as two unrelated relations. But intuitively these two relation labels are highly

Raw Sequence (RSeq)	Shortest Dependency Path (SDP)	Label
The [*burst*]$_{e_1}$ has been caused by water hammer [*pressure*]$_{e_2}$ -1 0 1 2 3 4 5 6 7 -8 -7 -6 -5 -4 -3 -2 -1 0	[*burst*]$_{e_1}$ $\xleftarrow{\text{nsubjpass}}$ caused $\xrightarrow{\text{prep}}$ by $\xrightarrow{\text{pobj}}$ [*pressure*]$_{e_2}$ 0 1 2 3 4 5 6 -6 -5 -4 -3 -2 -1 0	Effect-Cause
The [*burst*]$_{e_2}$ has been caused by water hammer [*pressure*]$_{e_1}$ -8 -7 -6 -5 -4 -3 -2 -1 0 -1 0 1 2 3 4 5 6 7	[*burst*]$_{e_2}$ $\xleftarrow{\text{nsubjpass}}$ caused $\xrightarrow{\text{prep}}$ by $\xrightarrow{\text{pobj}}$ [*pressure*]$_{e_1}$ -6 -5 -4 -3 -2 -1 0 0 1 2 3 4 5 6	Cause-Effect

Figure 1: An example of the representations of a mirror instance (in the bottom row) by our method.

related, and should not be independent of each other. By generating a mirror instance from each original instance, we can not only double the number of training data but also implicitly link the two labels r and $\mathrm{rev}(r)$. More importantly, for each testing instance, we can better identify its relation label based on its two representations.

For a relation instance x, let us use $\bar{x}$ to denote its mirror instance that we generate, and for a relation label y, let us use $\mathrm{rev}(y)$ to denote its *mirror label*, which is the reverse of y. Note that if y corresponds to a symmetric relation label, then $\mathrm{rev}(y)$ also corresponds to the same relation label.

Our mirror instance generation idea is inspired by the negative sampling method by Xu et al. (2015a) but our practice is fairly different. In their method, they only create a negative instance for each positive instance by reversing the original SDP, which will cause the expanded training set more biased to negative instances and thus largely reduce the recall of positive instances, whereas in our method, our generated mirror instances are not simply labeled as *Other* (or *None*) but labeled as a reversed relation from the original relation label. As a result, the class distribution of our generated *mirror* training set is almost the same as that of the original training set because of the *mirror* relationship between the original and *mirror* instances. More importantly, they simply expand the original training set with additional negative samples and their training process is the same as that in standard practice, while we propose a different pairwise relation classification framework in which the original and the mirror representations are jointly used for each relation instance.

Mirror Instance Generation

The next question is how we should construct RSeq($\bar{x}$) and SDP($\bar{x}$), the word representations of the mirror instance, such that we can use the same CNN architecture to learn the hidden sentence representations of these mirror instances.

For both SDP($\bar{x}$) and RSeq($\bar{x}$), although we could simply reverse the original representation as was done by Xu et al. (2015a), we feel that this would result in a completely reversed sentence or shortest dependency path that is unnatural. So we adopt the following way of constructing RSeq($\bar{x}$) and SDP($\bar{x}$). We leave the sequence of words untouched. For the position indices, since they are used to indicate the positions of the two entities, we simply swap the two position indices for each word such that the original e_1 now becomes e_2 and the original e_2 now becomes e_1. The bottom row of Figure 1 shows RSeq($\bar{x}$) and SDP($\bar{x}$) for the mirror instance "The [*burst*]$_{e_2}$ has been caused by water hammer [*pressure*]$_{e_1}$."

Pairwise Relation Classification

Training: Once it is clear how the RSeq and the SDP representations of a mirror instance are constructed, the next challenge is how to train with these pairs of original and mirror instances such that we can make a final prediction for each relation instance.

Essentially, in addition to the original training data $\{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$, we now have additional N training instances $\{(\bar{x}^{(n)}, \mathrm{rev}(y^{(n)})\}_{n=1}^{N}$, where $\bar{x}^{(n)}$ is the mirror instance of $x^{(n)}$ and $\mathrm{rev}(\cdot)$ is as defined previously. Moreover, these pairs of original and mirror instances have a one-to-one correspondence relationship, and therefore there should not be any disagreement between their labels.

We therefore design the following loss function to capture two components. The first component is to

maximize the log-likelihood of both the original and the mirror instances as follows:

$$J_c = -\sum_{n=1}^{N} \Big(\log p(y^{(n)}|x^{(n)}; \mathbf{\Theta}) + \log p(\mathrm{rev}(y^{(n)})|\bar{x}^{(n)}; \mathbf{\Theta}') \Big), \quad (1)$$

where $\mathbf{\Theta}$ and $\mathbf{\Theta}'$ are two sets of parameters respectively in the CNN model of the original instances and the mirror instances, which will be detailed in Section 2.3.

However, the Eqn. (1) above still treats each label separately and cannot link $y^{(n)}$ and $\mathrm{rev}(y^{(n)})$ to capture the relations between $x^{(n)}$ and $\bar{x}^{(n)}$. Consequently, we further construct a one-to-one correspondence relationship between $(x^{(n)}, y^{(n)})$ and $(\bar{x}^{(n)}, \mathrm{rev}(y^{(n)}))$. Intuitively, for $(x^{(n)}, \bar{x}^{(n)})$, the probability of the final label being $y^{(n)}$ should be a weighted combination of the probability of $x^{(n)}$ being $y^{(n)}$ and the probability of $\bar{x}^{(n)}$ being $\mathrm{rev}(y^{(n)})$. We therefore introduce another parameter $\omega \in \mathbb{R}^{2K+L}$ as a weight vector to combine the likelihood of the original and the mirror instances. The loss function is given as follows:

$$J_f = -\sum_{n=1}^{N} \log \Big(\sigma(\omega_{y^{(n)}}) p(y^{(n)}|x^{(n)}; \mathbf{\Theta}) + (1 - \sigma(\omega_{y^{(n)}})) p(\mathrm{rev}(y^{(n)})|\bar{x}^{(n)}; \mathbf{\Theta}') \Big),$$

where $\sigma(\omega_{y^{(n)}}) = \frac{1}{1+e^{-\omega_{y^{(n)}}}}$ is a tradeoff weight between the probability of $x^{(n)}$ being $y^{(n)}$ and the probability of $\bar{x}^{(n)}$ being $\mathrm{rev}(y^{(n)})$.

Finally, we minimize $J_c + J_f$ as our overall objective function. Since the overall objective function consists of two components and each component is related to the other, we propose to jointly optimize them via stochastic gradient descent with shuffled mini-batches, based on the practice by Kim (2014). In our implementation, the learning rate of each parameter is scheduled by Adadelta (Zeiler, 2012) (ϵ = 10^{-1}, ρ = 0.95 for ω, and ϵ = 10^{-6}, ρ = 0.95 for $\mathbf{\Theta}$ and $\mathbf{\Theta}'$).

Testing: After training with pairs of original and mirror instances, during the testing stage, how to predict the label of a relation instance becomes straightforward. For a test instance x^t, we should again generate its mirror instance $\bar{x}^t$. Thereafter, we can obtain two class distributions by using the trained model, one from x^t and the other from $\bar{x}^t$. Let us use $c(x^t)$ to denote the former and $c(\bar{x}^t)$ the latter. Finally, we can obtain the final class distribution $c(x^t, \bar{x}^t)$ based on $c(x^t)$ and $c(\bar{x}^t)$:

$$c_k(x^t, \bar{x}^t) = \sigma(\omega_k) c_k(x^t) + (1 - \sigma(\omega_k)) c_{\mathrm{rev}(k)}(\bar{x}^t), \quad 1 \le k \le 2K + L,$$

where $c_k(x^t)$ and $c_{\mathrm{rev}(k)}(\bar{x}^t)$ represent the probability of x^t having the relation k and $\bar{x}^t$ having the relation $\mathrm{rev}(k)$ respectively, and $c_k(x^t, \bar{x}^t)$ denotes the probability of the pair of relation instances having the relation k. The final predicted label is the relation with the highest probability among $c(x^t, \bar{x}^t)$.

2.3 Our Combined CNN Model

Under the mirror instance based pairwise relation classification (MI) framework, we further target at learning better representations for both the original and the mirror instances. Motivated by the observation that the raw sequence and the dependency path representations highly complement each other, we propose to combine the RSeq and the SDP representations of each relation instance (either the original or the mirror instance) together based on the multi-channel CNN architecture by Kim (2014).

Figure 2 illustrates the whole architecture of the MI framework, which contains two proposed combined CNN models. Each model obtains hidden representations of both the raw sequence and the shortest dependency path of a relation instance and then concatenates them for relation classification. It consists of a lookup layer, a convolution-pooling layer and an MLP layer.

Lookup: The lookup layer maps the input sequences to real-valued embedding vectors. Let $\mathbf{W}_e \in \mathbb{R}^{d_1 \times |\mathcal{V} \cup \mathcal{E}|}$ denote the lookup table for words and directed dependency relations, where each

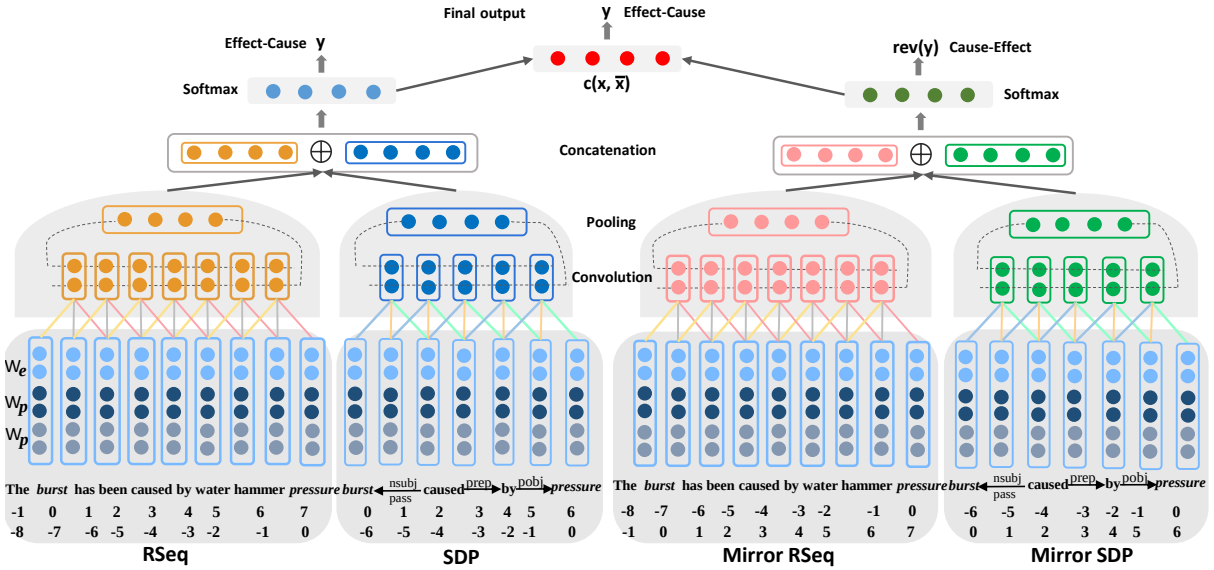


Figure 2: The architecture of the mirror instance based pairwise relation classification (MI) framework. The left and right components correspond to two combined CNN models (*Comb*) respectively for original instances and mirror instances.

column is a d_1-dimensional embedding vector for either a word in $\mathcal{V}$ or a dependency relation in $\mathcal{E}$. Let $\mathbf{W}_p \in \mathbb{R}^{d_2 \times |\mathcal{P}|}$ denote another lookup table for position indices, where each column is a d_2-dimensional embedding vector for a position index. Note that this position embedding idea is borrowed from the work by Zeng et al. (2014), and thus word representations of the raw sequence in our combined model is the same as theirs. After applying the lookup layer, both the RSeq and the SDP representations are transformed into a sequence of $(d_1 + 2d_2)$-dimensional vectors.

Convolution-Pooling: Two separate CNNs are used to process the RSeq representation and the SDP representation, and their mechanisms are the same. For each CNN, at position i of the original sequence, the embedding vectors inside a window of size n centered at i are concatenated into a new vector, which we refer to as $\mathbf{z}_i \in \mathbb{R}^d$. A convolution operation is then performed by applying a filter $\mathbf{F} \in \mathbb{R}^{h \times d}$ on $\mathbf{z}_i$ to produce a hidden vector $\mathbf{h}_i = g(\mathbf{F}\mathbf{z}_i + \mathbf{b})$, where $\mathbf{b} \in \mathbb{R}^h$ is a bias vector and g is a element-wise non-linear transformation function. Note that we pad the original sequence in front and at the back to ensure that at each position i we have n vectors to be combined into $\mathbf{h}_i$. After the convolution operation is applied to the whole sequence, we obtain $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots]$, and we apply a max-over-time pooling operator to take the maximum value of each row of $\mathbf{H}$ to obtain an overall hidden vector $\mathbf{h}^*$, which encodes the information from the entire sequence. Let $\mathbf{h}_r^*$ denote this hidden vector derived from RSeq and $\mathbf{h}_s^*$ the hidden vector derived from SDP.

MLP: The top layer of our model is a multilayer perceptron (MLP) with a softmax layer at the end to predict a $(2K + L)$-class distribution. This means the objective function for training our model is $J(\mathbf{\Theta}) = -\sum_{n=1}^{N} \log p(y^{(n)}|x^{(n)}; \mathbf{\Theta})$, where $\mathbf{\Theta}$ is the set of all model parameters including $\mathbf{W}_e$, $\mathbf{W}_p$, $\mathbf{F}$, $\mathbf{b}$ and the weights in the multilayer perceptron, $y^{(n)}$ is the true relation label for relation instance $x^{(n)}$, and $p(y^{(n)}|x^{(n)}; \mathbf{\Theta})$ is the probability of assigning $y^{(n)}$ to $x^{(n)}$ based on the softmax layer. As discussed in Section 2.2, in our implementation, the gradients are computed via back propagation.

3 Experiments

3.1 Dataset and Evaluation Metric

To evaluate our proposed method, we conduct our experiments on the SemEval-2010 Task 8 dataset and the English portion of the ACE-2005 dataset.

SemEval-2010 Task 8: This dataset contains 10,717 relation instances, including 8000 instances for training and 2717 for testing. Following Kim (2014), we randomly choose 10%, i.e., 800 of the training

Method	Prec	Rec	F_1
RSeq	81.11	84.72	82.78
RSeq+MI	**81.23**	**85.42**	**83.22***
SDP	80.57	83.54	82.01
SDP+MI	**80.98**	**83.89**	**82.36***

Table 3: Evaluation of our mirror instance method. * indicates that our method significantly improves the corresponding baseline with $p < 0.05$ based on McNemar's test.

Method	Prec	Rec	F_1
RSeq	81.11	84.72	82.78
SDP	80.57	83.54	82.01
Comb	**81.27**	**85.33**	**83.20***

Table 4: Comparison of our combined model with two baseline models using either *RSeq* or *SDP* representation.

Method	Prec	Rec	F_1
Comb	81.27	85.33	83.20
Comb+MI	**82.07**	**86.63**	**84.23***
Comb+RMI	***82.34***	***87.76***	***84.96****

Table 5: Evaluation of our combined CNN model together with the mirror instance method. RMI stands for reduced mirror instances.

instances as the development set. Following all previous work, we use the macro-averaged F_1 score to evaluate our model based on the SemEval-2010 Task 8 official scorer.

ACE-2005: This dataset consists of 6 domains: broadcast news (*bn*), newswire (*nw*), broadcast conversation (*bc*), telephone conversation (*cts*), weblogs (*wl*) and usenet (*un*). Following some previous work (Plank and Moschitti, 2013; Nguyen et al., 2015; Gormley et al., 2015), we consider a domain adaptation setting for coarse-grained relation extraction. Specifically, we take the union of *bn* and *nw* as the training set, half of *bc* as the development set, and the remainder (i.e., *cts* and *wl* as well as the other half of *bc*) as the test set. Following Plank and Moschitti (2013), we use the micro-averaged F_1 score to evaluate our model.

3.2 Experiment Settings

We use the pre-trained word embeddings from *word2vec*[1] to initialize the lookup table $\mathbf{W}_e$, and set the dimension d_1 to 300. For unknown words and directed dependency labels, we randomly initialize their 300-dimensional embedding vectors. We also randomly initialized the other lookup table of the position embeddings $\mathbf{W}_p$, and set the dimension d_2 to 50. Note that in our preliminary experiments for ACE-2005, we found that the performance without considering the entity types of the two entity mention heads is very limited. Hence, for ACE-2005, we also randomly initialize another lookup table of the entity type embeddings, whose dimension is set to 50, and represent each token by concatenating its word embedding, position embedding and entity embedding.

We want to compare our combined CNN model with models that use either RSeq or SDP alone, so we consider three experiment settings: *SDP* refers to a CNN model that uses only SDP representation of a relation instance, *RSeq* refers to a CNN model that uses only the RSeq, and *Comb* refers to our combined model. For each setting, we use the development set to tune the window size n and the dimension of the hidden states h. In a previous study by Nguyen and Grishman (2015), it was found that using multiple window sizes in CNN can bring significant improvements for the RSeq representation. We therefore also experiment with combining multiple window sizes for *RSeq* and *SDP*. In the end, we find that for *RSeq*, the optimal setting is to use a combination of windows with sizes 2, 3, 4 and 5 and to set h to 150. For *SDP*, the optimal setting is to use a single window of size 5 and to set h to 400. For *Comb*, we use the same window sizes and hidden sizes h as *RSeq* and *SDP*.

For the other parameters in $\mathbf{\Theta}$ and $\mathbf{\Theta}'$, we adopt the settings reported by Nguyen and Grishman (2015). That is, the non-linear transformation function g is $\tanh$, the mini-batch size is 50, the dropout rate α equals 0.5, and the hyperparameter for the l_2 norms is set to be 3.

3.3 Evaluation of our Proposed Approach

In this section, we evaluate the different components of our method.

Effect of the Mirror Instance Method

To evaluate the effect of the mirror instances, first, we apply the mirror instance method on top of the two baseline methods *RSeq* and *SDP*, and show the results on SemEval-2010 in Table 3 and the results

[1] https://code.google.com/p/word2vec/

Method	dev set			bc			cts			wl			avg
	Prec	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**	**F_1**
RSeq	73.5	52.7	61.4	70.3	52.4	60.1	65.8	45.9	54.1	57.7	44.1	50.0	54.7
RSeq+MI	69.1	59.1	**63.7***	65.7	59.2	**62.3***	67.7	44.9	54.0	59.5	46.9	**52.4***	**56.2**
SDP	67.6	49.3	57.0	59.4	45.3	51.4	55.4	37.7	44.9	48.8	36.6	41.8	46.0
SDP+MI	66.5	51.7	**58.2***	57.6	48.6	**52.7***	53.3	39.3	**45.3**	45.8	37.4	41.2	**46.4**

Table 6: Evaluation of our mirror instance method on ACE-2005.

Method	dev set			bc			cts			wl			avg
	Prec	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**	**F_1**
RSeq	73.5	52.7	61.4	70.3	52.4	60.1	65.8	45.9	54.1	57.7	44.1	50.0	54.7
SDP	67.6	49.3	57.0	59.4	45.3	51.4	55.4	37.7	44.9	48.8	36.6	41.8	46.0
Comb	72.4	57.8	**64.3***	70.6	57.5	**63.4***	62.5	49.7	**55.4** *	60.4	49.5	**54.4***	**57.7**

Table 7: Evaluation of our combined CNN model on ACE-2005.

on ACE-2005 in Table 6. We can see that with the help of the mirror instances, both *RSeq* and *SDP* can improve their performance in most cases, and the improvements are statistically significant. This indicates the usefulness of the mirror instances generated by our method.

The Combined CNN Model

To check the effect of combining RSeq and SDP representations, in Table 4 and Table 7, we compare *Comb* with *SDP* and *RSeq* on SemEval-2010 and ACE-2005 respectively. We can observe that *Comb* outperforms both *SDP* and *RSeq* on two datasets and the improvements are statistically significant. We can also see that the precision of *Comb* and that of the other two models are relatively close, especially on SemEval-2010, and the advantage of *Comb* is mainly from its recall. It suggests that the RSeq and the SDP representations complement each other and therefore can work better when combined.

The Combined CNN Model together with the Mirror Instance Method

We then apply our mirror instance method on top of *Comb*. In Table 5 and the top two rows of Table 9, we can observe that our mirror instance method (*Comb*+MI) can significantly improve the F_1 score of *Comb*, especially making high improvements in recall, which further verifies the usefulness of our mirror instance method.

Since the goal of our relation classification task is to improve the F_1 score for the positive relation types excluding the label *Other*, we further investigate the impact of reducing the number of mirror instances generated from the *Other* relation instances, i.e., the negative relation instances on SemEval-2010. By tuning the percentage of negative mirror instances to reduce, we achieve the best performance when reducing 50% of the negative mirror instances. We refer to this method as *Comb*+RMI and show its performance in Table 5 in the last row. We can see that it achieves a F_1 score of 84.96.

3.4 Comparison with the State of the Art

In this section, we compare our proposed method with all recently published results for SemEval-2010 Task 8 and ACE-2005.

SemEval-2010 Task 8: Since most existing studies have used additional hand-crafted linguistic features (AF) to help the classification task, we show two different F_1 scores, one with AF and one without in Table 8. It is easy to observe that without AF, Vu et al. (2016) obtained the best F_1 score of 84.9 by combining CNN and RNN models via a voting strategy; with AF, Xu et al. (2015b) achieved the best result with negative sampling and 8000 negative examples from the New York Times (NYT) dataset.

Method	Additional Features (AF)	F_1	
		with AF	without AF
SVM (Rink and Harabagiu, 2010)	POS, prefixes, morphological, WordNet, Levin classed, ProBank, FrameNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2	-
CNN (Zeng et al., 2014)	words around entities, WordNet	82.7	78.9
DepNN (Liu et al., 2015)	NER	83.6	82.8
Hybrid(FCM+Feat) (Gormley et al., 2015)	NER	83.7	-
SDP-LSTM (Xu et al., 2015b)	POS,Wordnet	83.7	82.4
DepLNN+NS (Xu et al., 2015a)	Samples from NYT, WordNet	85.6	84.0
CR-CNN (dos Santos et al., 2015)	-	-	84.1^{+}
ER-CNN + RNN (Vu et al., 2016)	-	-	84.9^{+}
SpTree (Miwa and Bansal, 2016)	Wordnet	85.5	84.5
Comb+RMI	Wordnet, NER	**85.7**	**85.0**

Table 8: Comparisons with state-of-the-arts results on SemEval-2010. $^{+}$ indicates using a special ranking-based objective function.

Method	**dev set**			**bc**			**cts**			**wl**		
	Prec	**Rec**	**F$_1$**	**Prec**	**Rec**	**F$_1$**	**Prec**	**Rec**	**F$_1$**	**Prec**	**Rec**	**F$_1$**
Comb	72.4	57.8	64.3	70.6	57.5	63.4	62.5	49.7	55.4	60.4	49.5	54.4
Comb+MI	70.9	60.4	**65.2***	66.2	63.6	**64.9***	65.1	51.5	***57.5****	57.1	52.4	**54.7**
The State-of-the-art Systems												
FCM	-	-	-	66.6	57.9	61.9	65.6	44.3	52.9	57.8	44.6	50.4
Hybrid(FCM+Feat)	-	-	-	74.4	55.3	63.5	74.5	45.0	56.1	65.6	47.6	55.2
Hybrid(CNN+RNN+Feat)	69.3	66.3	***67.8***	65.8	66.5	***66.1***	63.6	51.7	57.0	56.4	57.2	***56.8***

Table 9: Evaluation of our combined CNN model together with the mirror instance method on ACE-2005. The results of the state-of-the-art systems are taken from Nguyen and Grishman (2016).

We can also see that without utilizing any AF, our *Comb*+RMI method can push the F_1 score to the state-of-the-art, 85.0. Furthermore, we also consider adding two kinds of lexical features to our model, namely, Named Entity type (NER) and WordNet hypernyms. We first obtain the NER features of all words and Wordnet hypernyms of the two entities using the tool developed by Ciaramita and Altun (2006). Then, we represent each token by concatenating its word embedding, position embedding and entity embedding. Finally, following the practice by Zeng et al. (2014), we also concatenate the Wordnet hypernyms of the two entities with the combined hidden vector. As we can see from the last line of Table 8, our method can achieve the state-of-the-art F_1 score, 85.7.

ACE-2005: In Table 9, it is easy for us to observe that on all three test domains, our proposed *Comb*+MI method can outperform the state-of-the-art single system FCM with a large margin, which combines traditional linguistic features with learned word embeddings by a log-bilinear model. In addition, we can also find that even in comparison with a competitive hybrid model, which integrates FCM and a traditional feature-based method, *Comb*+MI can still achieve slightly better performance on the *bc* and *cts* domains, and similar performance on the *wl* domain.

Recently, Nguyen and Grishman (2016) proposed an ensemble method by first combining CNN and RNN via a stacking strategy and then integrating it with a traditional feature-based method in a hyrid model. Although our result is slightly lower than Hybrid(CNN+RNN+Feat) on average, we believe that our model can be further improved with such an ensemble strategy, which we leave to our future work.

4 Related Work

Traditional work on relation classification can be categorized into feature-based methods and kernel-based methods. The former relies on a large number of human-designed features (Zhou et al., 2005; Jiang and Zhai, 2007; Li and Ji, 2014) while the latter leverages various kernels to implicitly explore a much larger feature space (Bunescu and Mooney, 2005; Nguyen et al., 2009). However, both methods suffer from error propagation problems and poor generalization abilities on unseen words. The most popular method to solve the two limitations is based on neural networks (NNs), which have been shown successful in extracting meaningful features and generalizing on unseen words for many NLP tasks (Kim, 2014). For relation classification, Socher et al. (2012) proposed a recursive matrix-vector model based on constituency parse trees. Zeng et al. (2014) and dos Santos et al. (2015) respectively proposed a standard and a ranking-based CNN model based on the raw word sequences. More recently, Xu et al. (2015b) and Miwa and Bansal (2016) respectively proposed a multi-channel sequential LSTM model and a bidirectional tree-LSTM model on the shortest dependency path for relation classification.

Although all these models have been shown to be effective, all of them only focus on learning a single representation for each relation instance. Different from all previous methods, we first design a strategy to generate a mirror instance from each original relation instance and then propose a pairwise relation classification framework to learn a pair of representations for each relation instance.

On the other hand, most existing NN-based approaches for relation classification are either based on the shortest dependency path or the raw sequence, although these two representations may complement each other. In this work, we propose to combine them together based on the multi-channel CNN architecture (Kim, 2014), aiming to capture long-distance relations without losing any information.

5 Conclusions

In this paper, we first proposed a mirror instance method to learn a pair of representations for each instance, which basically includes a mirror instance generation strategy and a pairwise relation classification framework. Based on this, we further proposed a combined CNN model based on both the RSeq and the SDP representations of relation instances. Our experimental results demonstrate that our mirror instance method can improve the baseline models and our combined model without mirror instances, and our combined CNN model is more effective than models only using the RSeq or the SDP representation of relation instances. Finally, with the help of some lexical features, our combined CNN model together with the mirror instance method achieves the state-of-the-art result on SemEval-2010 and highly competitive results on ACE-2005.

Acknowledgment

We would like to thank the reviewers for their valuable comments. This research is supported by the Singapore National Research Foundation under its International Research Centre@Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA).

References

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 724–731. Association for Computational Linguistics, October.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602. Association for Computational Linguistics.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*

and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 626–634. Association for Computational Linguistics, July.

Javid Ebrahimi and Dejing Dou. 2015. Chain based rnn for relation classification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1244–1249. Association for Computational Linguistics, May–June.

Matthew R. Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1774–1784. Association for Computational Linguistics, September.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376. Association for Computational Linguistics, October.

Jing Jiang and ChengXiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 113–120. Association for Computational Linguistics, April.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics, October.

Nozomi Kobayashi, Kentaro Inui, and Yuji Matsumoto. 2007. Extracting aspect-evaluation and aspect-of relations in opinion mining. In *EMNLP-CoNLL*, volume 7, pages 1065–1074. Citeseer.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, Maryland, June. Association for Computational Linguistics.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng WANG. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 285–290. Association for Computational Linguistics, July.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48. Association for Computational Linguistics, June.

Thien Huu Nguyen and Ralph Grishman. 2016. Combining neural networks and log-linear models to improve relation extraction. *Proceedings of IJCAI Workshop on Deep Learning for Artificial Intelligence*.

Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1378–1387. Association for Computational Linguistics, August.

Thien Huu Nguyen, Barbara Plank, and Ralph Grishman. 2015. Semantic representations for domain adaptation: A case study on the tree kernel-based method for relation extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 635–644, Beijing, China, July. Association for Computational Linguistics.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1498–1507. Association for Computational Linguistics.

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 697–704. Association for Computational Linguistics.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259. Association for Computational Linguistics, July.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics, July.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning With Neural Tensor Networks For Knowledge Base Completion. In *Advances in Neural Information Processing Systems*.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 534–539, San Diego, California, June. Association for Computational Linguistics.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 536–540. Association for Computational Linguistics, September.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794. Association for Computational Linguistics, September.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966, Baltimore, Maryland, June. Association for Computational Linguistics.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344. Dublin City University and Association for Computational Linguistics, August.

Min Zhang, Jie Zhang, Jian Su, and GuoDong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 825–832. Association for Computational Linguistics, July.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 427–434. Association for Computational Linguistics, June.

FastHybrid: A Hybrid Model for Efficient Answer Selection

Lidan Wang
IBM Watson
T.J. Watson Research Ctr.
Yorktown Heights, NY
wangli@us.ibm.com

Ming Tan
IBM Watson
T.J. Watson Research Ctr.
Yorktown Heights, NY
mingtan@us.ibm.com

Jiawei Han
Computer Science Dept.
UIUC
Urbana, IL
hanj@cs.uiuc.edu

Abstract

Answer selection is a core component in any question-answering systems. It aims to select correct answer sentences for a given question from a pool of candidate sentences. In recent years, many deep learning methods have been proposed and shown excellent results for this task. However, these methods typically require extensive parameter (and hyper-parameter) tuning, which gives rise to efficiency issues for large-scale datasets, and potentially makes them less portable across new datasets and domains (as re-tuning is usually required). In this paper, we propose an extremely efficient hybrid model (*FastHybrid*) that tackles the problem from both an accuracy and scalability point of view. *FastHybrid* is a light-weight model that requires little tuning and adaptation across different domains. It combines a fast deep model (which will be introduced in the method section) with an initial information retrieval model to effectively and efficiently handle answer selection. We introduce a new efficient attention mechanism in the hybrid model and demonstrate its effectiveness on several QA datasets. Experimental results show that although the hybrid uses no training data, its accuracy is often on-par with supervised deep learning techniques, while significantly reducing training and tuning costs across different domains.

1 Introduction

Open-domain question answering (QA) aims to serve a user's information request by returning a list of direct answers. This problem has been receiving an increasingly amount of attention in the NLP and machine learning communities in the recent years (Ferrucci et al., 2012; Etzioni et al., 2011). Answer sentence selection (Yih et al., 2013; Tan et al., 2016; Yu et al., 2014; Severyn et al., 2013), which, given a user question, returns the correct sentences that contain the exact answer, is a core component in QA systems. The performance of QA systems critically depends on choosing the right candidate sentences which facilitate the extraction of final answers.

To be successful, in addition to accuracy, real-world systems must be scalable and select the most accurate answer sentences in a short amount of time. However, accuracy and speed/scalability are competing forces that often counteract each other. It is often the case that methods developed for improving accuracy incur moderate-to-large computational costs. For example, models based on neural networks have become very popular due to their strong accuracy for this task (Yu et al., 2014). However, they are typically slower at training and test time as compared to simple models, which may limit their use on very large datasets (Joulin et al., 2016). The speed issue is particularly important when working with new domains and datasets, as the model may have to be re-trained or adapted for the new dataset. Model re-training and adaptations, even with incremental techniques (Zhou et al., 2012; Chopra et al., 2013; Glorot et al., 2011) on state-of-the-art hardware (Chilimbi et al., 2014; Xing et al., 2015), could be prohibitively inefficient when working with time-critical applications or an impatient end-user – who may prefer a method with minimum (or no) training time spent, *and* getting similar accuracy as the expensively trained models.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2378–2388, Osaka, Japan, December 11-17 2016.

On the other hand, classical information retrieval (IR) models are extremely fast as compared to the deep models, but may not be as quite accurate. IR models use fast word matching features (e.g., unigram, bigram overlaps between question and answer) to quickly score candidate sentences. Typically no training is necessary, since these simple models are fairly robust and can be applied to different datasets without modification (Tao et al., 2007; Bendersky et al., 2010; Buttcher et al., 2006). This property makes them attractive for scaling to large number of new domains and datasets.

In this paper, we take a step to eliminate the virtual dichotomy between accuracy and scalability/speed by developing a hybrid model that is simultaneously effective (as the deep models) and extremely efficient at training and test time (as the simple models). We name our proposed structure *FastHybrid*. We introduce the concept of a fast (deep) model in Section 3, then describe the *FastHybrid* model which combines IR with the proposed fast (deep) model for the best possible accuracy and training speed.

Unlike many other deep learning models, the fast deep model skips the intermediate convolution steps altogether, and directly operates on the raw word vectors coming from the answer and a slightly modified question text, constructed from a simple attention approach we propose in Section 3.2, to perform standard pooling operations. The simple attention in the fast deep model (Section 3.2) works by focusing explicitly on the question's influence on the answer with respect to the answer's representation. While seemingly, important information from convolution may get lost, however, with the new simple attention, our model often beats its expectations – not only does it perform well for time, but for accuracy equally.

The fast deep model is combined with an IR model via a hybrid structure to handle answer selection accurately and efficiently. The IR model is used to create an initial ranking of candidate sentences, and for the questions that cannot be handled well by IR, the fast deep model is applied. The hybrid structure leverages complementary strengths by IR and the fast model (both in terms of accuracy and speed), as will be discussed in Section 3.4.

Our model is nearly hyper-parameter/parameter-free, so it is extremely efficient (no training) for different datasets, and as a result, it can scale very easily to a large number of new domains and users. The remainder of the paper is organized as follows: we start with a discussion of related work, Section 3 describes our fast model and the hybrid approach, and our methods are evaluated in Section 4, before discussing future work and concluding.

2 Related work

In recent years, the problem of answer selection, which is a sub-problem in question-answering, has been getting a lot of attention in the research community (Yih et al., 2013; Tan et al., 2016; Yu et al., 2014; Severyn et al., 2013). Moving away from the shallow word-level features, these deep learning approaches focus on extracting important features from low-level representations (word embeddings) by using various types of deep neural networks (Graves et al., 2013; Hochreiter et al., 1997). The resulting models can effectively work at a semantic-level (i.e., can match a correct answer with a question even if they do not have any words in common, however semantically related). This can be viewed as a big improvement in comparison to the standard information retrieval approaches for the same task, which typically use hand-crafted word matching features.

However, from an efficiency's point of view, the deep models are very time-consuming to train. In particular, to achieve good success, the models typically require a large number of parameters and weights, making parameter (and hyper-parameter) tuning an expensive process for large-scale applications. While many remedies have been proposed to address the efficiency issue, ranging from developing fast training hardwares (Chilimbi et al., 2014) to parameter sharing and simplifying model structures, in comparison to standard simple IR approaches (Tao et al., 2007; Bendersky et al., 2010; Buttcher et al., 2006), the training efficiency and scalability of these methods still significantly lack behind.

This issue is particularly significant when we have to adapt the model repeatedly for new domains. While domain-adaptation techniques have been proposed for many deep learning models (Chopra et al., 2013; Zhou et al., 2012; Ganin et al., 2015; Sun et al., 2016), the focus and starting point of these work have largely been on how to adjust the models for accuracy, rather than from an accuracy and scalability point of view.

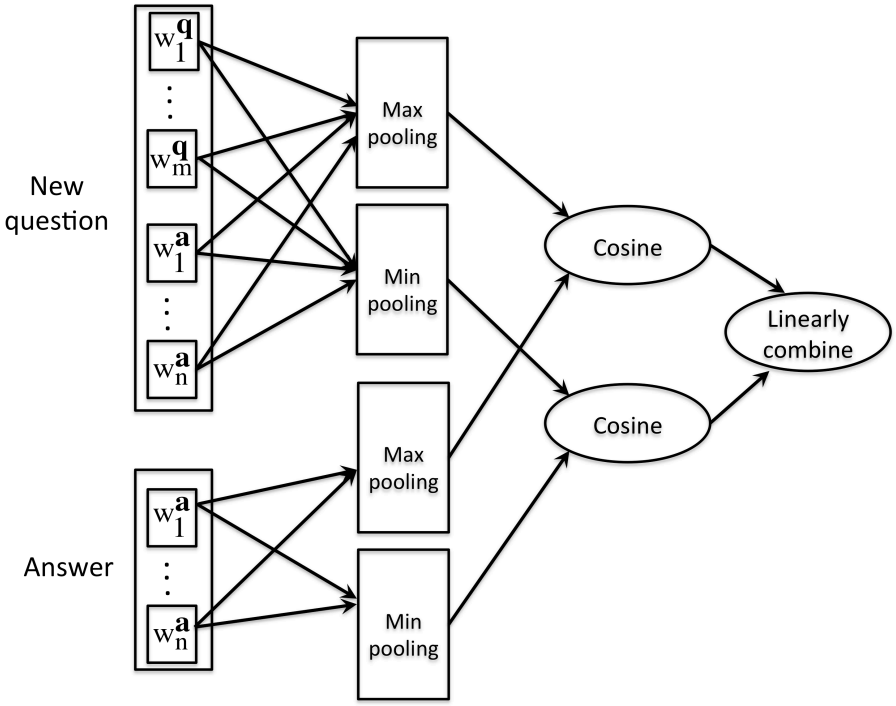


Figure 1: Model architecture for simplified deep learning model (a.k.a. fast model)

3 Methods

In this section, we describe our proposed hybrid model, *FastHybrid*, for effective and efficient answer selection. It combines a fast (deep) model[1] with an initial IR model to create an efficient and effective overall structure for this task.

A major area that distinguishes our work from the past deep learning models is that we directly operate on the raw word embedding vectors to perform standard aggregation operations (e.g., max, min-pooling) before computing similarity scores (Section 3.3). We completely discard the convolution operations used by many other deep models altogether. With the simple attention (Section 3.2) used by the fast model, our system is orders of magnitude faster in training than the standard deep models, while performing well for accuracy equally. We begin with a quick overview of the preliminary background on the Word2Vec embedding, then focus on the fast (deep) model in Section 3.2 and Section 3.3, before discussing the hybrid structure in Section 3.4.

3.1 Word2Vec Embedding

Recently (Mikolov et al., 2013) introduced *word2vec*, a word-embedding procedure. They use a shallow neural network language model to learn a vector representation for each word. More specifically, a neural network architecture (the skip-gram model) is proposed and consists of an input layer, a projection layer, and an output layer to predict words nearby. Each word vector is trained to maximize the log probability of neighboring words in a corpus. That is, given a sequence of words $w_1, \ldots, w_Z$,

$$\frac{1}{Z} = \sum_{z=1}^{Z} \sum_{i \in nb(z)} log\ p(w_i|w_z)$$

[1]We slightly abuse terminology and refer to it as a fast (deep) model. Note no convolution filters etc (representative of standard deep models) are used. We use this name mostly for comparison purpose with the state-of-the-arts deep learning models.

where $nb(z)$ denotes the set of neighboring words of w_z and $p(w_i|w_z)$ denotes the hierarchical softmax of the associated word vectors $\mathbf{v}_{w_z}$ and $\mathbf{v}_{w_t}$ (see (Mikolov et al., 2013) for more details). Due to the simple architecture and the use of hierarchical softmax, the skip-gram model can be trained on billions of words per hour using a conventional desktop computer. It is unsupervised to learn the word embeddings and it can be computed on the corpus of interest or pre-trained in advance. Note our framework is not limited to a particular type of word embeddings, any commonly-used embeddings for text can be substituted into our framework.

3.2 Efficient simple attention

Attention mechanisms (Mnih et al., 2014) have been shown to be useful in deep learning models. They help guide pooling to be more cognizant of the key answer tokens relative to the question. While useful, using attention brings another layer of complexity, more model parameters, hence longer training time.

In this section, we ask the question – can we design a more efficient attention method in the fast model that does not require additional parameters yet achieves good results? The answer is in the affirmative. Our new attention strategy in the fast model is extremely simple (only 1 line of code!), however, it is surprisingly effective. For a pair of question and answer, we replace the question text by an augmented version of the question, which is formed by a simple *concatenation* of the original question and the given answer. This exact process is illustrated by the left side in Figure 1, where each w_m^q denotes a word in the question q, and w_n^a is a word in the answer a. The new question consists of the combined tokens from the question and answer. A look-up on word-embeddings is then performed to get the corresponding word vectors for the new question tokens. Then max- and min-pooling will be performed (details in the next section) on these word vectors. The final cosine score will be computed between the new question and the answer. Note on the answer side, the representation of the answer text stays intact (i.e., a lookup on word vectors of the original answer tokens is performed, and the vectors are passed to the pooling stage).

Why do we call this simple strategy an *attention* mechanism? Adding question q's tokens into the answer will "corrupt" the answer's representation and the subsequent max- and min-pooling results of the answer, by *biasing* the answer's representation towards q's semantic meaning. Intuitively, if the answer is correct for q, then they are semantically identical, in which case the "corruption" of the answer a by q should not change the answer's semantic meaning, and the subsequent cosine similarity between the answer a and the combined text should be very close to one (i.e., strong similarity). One the other hand, if the answer is incorrect for q, they are semantically different, the corruption of a by q will make the answer's representation drastically different from its original, forcing the cosine score with its original representation far from one.

The simple "corruption" by q can be viewed as an *attention* placed on the answer, by *focusing explicitly* on q's influence on the answer with respect to the answer's original representation. This is exactly what is captured by Figure 1. The new question can be viewed as a corrupted version of a (by adding q's tokens into it), and after the subsequent pooling operations, cosine similarities between the original answer's representation and the new corrupted version are computed.

3.3 Pooling and similarity computation

The next step in the fast model is the max- and min-pooling as shown in Figure 1. They are directly performed on word vectors from the answer and the modified question to form one-dimensional vector representations. The pooling is performed along each dimension in the word vector, over the tokens in the input text.

We can think of max-pooling as a way for the model to ask whether a given semantic class is found anywhere in the input text, while the min-pooling captures the absence of it. Both pooling techniques have been used before in deep learning applications. In this work, each pooling is used to create a pair of representations for the inputs, from which a cosine similarity score is derived. The resulting two cosine similarity scores (based on two pooling strategies) are then combined via a weighted linear combination to form a final score for the answer:

$$Score(a) = w \cdot cosine(v_{max}^{q'}, v_{max}^{a}) + (1 - w) \cdot cosine(v_{min}^{q'}, v_{min}^{a})$$

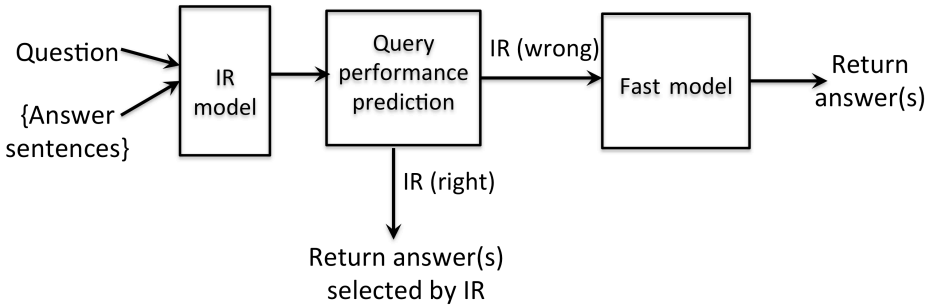


Figure 2: Hybrid approach combining IR with fast, simplified deep model (Figure 1)

where vectors $v_{max}^{q'}$ and v_{max}^{a} denote the outcome of max-pooling performed on the inputs (i.e., answer a and the augmented question q'). The vectors $v_{min}^{q'}$ and v_{min}^{a} are similarly defined. The single weight w in the linear combination is empirically set. We found in practice, this weight is fairly robust and insensitive to different datasets. In this work, it is simply set to a constant ($w = 0.7$) and does not change from datasets to datasets. Finally, $Score(a)$ is the score used to rank each candidate answer for the given question, and the one with the highest score is chosen as the best answer by this model.

3.4 Hybrid structure

A hybrid structure is used to combine the fast model with an IR retrieval model. In this work, we use the IR retrieval model in (Bendersky et al., 2010), which has shown dominant performance in a number of text and sentence retrieval tasks. The hybrid model is shown in Figure 2. It leverages complementary strengths of the IR model and the fast model. For accuracy, the IR retrieval model and the fast model each can cover a unique population of the questions. For example, the IR retrieval model excels at exact matching of the question with answer terms, if the answer contains sufficient word overlaps with the question. Predictably, a decent fraction of questions and answers may fall into this category, given the past wide usage and success of such retrieval models (Tao et al., 2007; Bendersky et al., 2010; Buttcher et al., 2006) for answer selection and retrieval. On the other hand, some correct answers may not share any words with the question, and in this situation, the fast model, which works from a semantic level, can be put into use.

A key component in the hybrid model is deciding when to return the IR results, and when to forward it to the fast model for scoring. This problem is formulated as "query performance prediction" (QPP), previously studied by the retrieval community (He et al., 2005; Hauff et al., 2009). A predictive model (usually of the form of a logistic regression) is used to predict the accuracy of the initial results for a given question, based on statistics extracted from IR answer scores for the question, such as the separation (ratio) between the maximum and minimum answer scores, etc. In this work, we follow the work by (He et al., 2005) to build the query performance prediction model, which is used to direct a question to either the fast model or return the IR results. It is beyond the scope of our paper to detail the QPP approach; we refer interested readers to (He et al., 2005) for more details about query performance prediction.

Finally, we would like to point out the entire pipeline is hyper-parameter free, relieving the need for expensive hyper-parameters tuning. This property allows our model to work with new domains and datasets more seamlessly – reducing the workload in model tuning.

4 Experiments

In this section we present a comprehensive set of experiments over three QA datasets: WikiQA, TrecQA, and InsuranceQA. WikiQA (Yang et al., 2015) is an open domain question-answering dataset. We use the subtask that assumes that there is at least one correct answer for a question. The TrecQA dataset was created based on TREC QA task (8-13) data (Voorhees et al., 2000). We follow the exact approach of train/dev/test question selections as in (Wang et al., 2015). InsuranceQA(v2)[2] is a recently released

[2]git clone https://github.com/shuzi/insuranceQA.git

	WikiQA	TrecQA	InsuranceQA(v2)
Train (# questions)	873	1162	12,889
Dev (# questions)	126	65	2000
Test (# questions)	243	68	2000
Avg # cand answers	9	38	500

Table 1: Dataset statistics: WikiQA, TrecQA, and InsuranceQA(v2)

large-scale non-factoid QA dataset from the insurance domain, which has drawn interests from the deep learning community for studying answer selections.

The statistics of these datasets are given in Table 1, including the number of questions, and the average number of candidate answers in the train/dev/test set. We study the following methods in our experiments:

- **IR retrieval model (Bendersky et al., 2010).** This IR model utilizes unigram and bigram overlaps between the question and answer, and represents the state-of-the-art in the IR field for effective answer selection and text retrieval (Bendersky et al., 2010). Similar to our approach, this model requires no training and it is applied the same way for all datasets.

- **FastHybrid**. The proposed hybrid model (Section 3) does not require any training. The tunable parameter (the weight w in the linear combination of cosine scores) is empirically set to 0.7 and stays the same for all datasets. Thus, the training and dev sets are not used by the hybrid model. In addition, the model uses standard word embedding vectors[3] trained from the Wikiepdia[4] and Gigaword [5] data collections. Although we do not re-train the word-embeddings and use the same pre-trained embeddings for all datasets, we could potentially re-train it on each dataset to achieve higher accuracy performance than reported here.

- **Supervised background models**. Since model scalability and efficiency are an issue, for supervised models, rather than re-training them for each new QA dataset, we could instead just train them on a big background dataset containing examples representative of the individual datasets, then apply the trained model to each QA data without re-training. This strategy refrains from repeated training, and scales better to large number of domains – thus, it serves as a direct comparison point to the hybrid model in the experiments. Here we use the Yahoo! answers dataset[6], which is a large Q&A corpus from which we extracted question-answer pairs as the training data. As the training method, we use two state-of-the-arts deep learning models (Feng et al., 2015; Santos et al., 2016), which apply convolutional neural networks to extract meaningful representations for each question and answer pair (Feng et al., 2015), and utilize bidirectional attentions on the questions and answers to enhance answer selection accuracy (Santos et al., 2016). We denote these two supervised background models as *Supervised background-1* and *Supervised background-2*, from applying (Feng et al., 2015) and (Santos et al., 2016) to the Yahoo background corpus, respectively.

- **Deep learning methods for answer selection**. As mentioned earlier, there is a recent surge on applying deep learning for answer selection. To be maximally effective, these techniques typically require in-domain training data for large-scale parameter tuning. Although our end goal is clearly different from that of standard in-domain supervised deep learning – we approach the problem from an accuracy *and* scalability point of view and develop light-weight models which require little tuning and adaptation across different domains – we present their results for completeness purpose whenever appropriate. As we will see, although the hybrid model uses no training data, its accuracy is often on-par (and sometimes slightly better) than supervised deep learning methods trained with in-domain data (Santos et al., 2016; Feng et al., 2015). This point echoes a similar observation made recently by related work on classification (Joulin et al., 2016).

[3]http://nlp.stanford.edu/projects/glove/
[4]https://dumps.wikimedia.org/enwiki/20140102/
[5]https://catalog.ldc.upenn.edu/LDC2011T07
[6]https://webscope.sandbox.yahoo.com/

Model	WikiQA top-1 acc	TrecQA top-1 acc	InsuranceQAv2 top-1 acc
IR (Bendersky et al., 2010)	40.9%	63.23%	18.20%
Supervised background-1	44.4%	61.7%	21.6%
Supervised background-2	44.4%	60.2%	19.4%
FastHybrid	48.2%	71.5%	22.7%

Table 2: Top-1 test accuracy on three QA datasets

Model	WikiQA training	TrecQA training	InsuranceQA(v2) training
IR model (Bendersky et al., 2010)	no training (0s)	no training (0s)	no training (0s)
Supervised background-1 model	6h/170K question-answer pairs		
Supervised background-2 model	6h/180K question-answer pairs		
FastHybrid	no training (0s)	no training (0s)	no training (0s)

Table 3: Training time on three QA datasets.

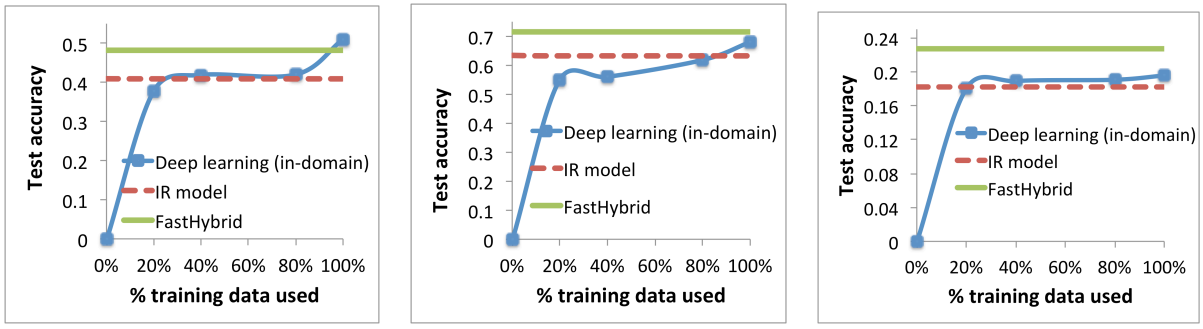
All experiments were run on a NVIDIA Tesla K20Xm GPU processor, with memory size per board (GDDR5) 5GB.

4.1 Model accuracy

Table 2 presents the top-1 accuracy for each QA domain test set[7]. Among the four comparison methods, IR and *FastHybrid* incur no training time (since no training is needed), while the supervised background models are trained on the large-scale background Yahoo Q&A corpus. We see that the hybrid model consistently achieves better accuracies for all QA domains in comparison to other techniques. An interesting observation is that while the hybrid model is simple, it is more robust than the supervised background models. For example, while in WikiQA and InsuranceQA, the supervised background models achieve decent performance relative to IR and *FastHybrid*, in TrecQA, their performance drops significantly (at 61.7%, and 60.2%, respectively) with respect to the hybrid model (71.5%). While utilizing the same training and tuning mechanism as the state-of-the-arts deep learning techniques (in terms of parameter tuning etc), it cannot make up for the domain gap between the TrecQA domain and the background Yahoo! Q&A corpus used as the training data. This points out that while the supervised background models can save some training time by doing a one-time offline training, it may potentially and significantly hurt the accuracy of new domains not well represented by the background corpus. This contrasts with the hybrid model, which aims to simultaneously achieve better efficiency, scalability, while not hurting accuracy.

Furthermore, while we try to build models that can scale to large number of domains more quickly and easily – a departure from standard in-domain supervised techniques (which can be used as an accuracy upper-bound), in many cases the hybrid model performs on-par with these methods for accuracy. As we will see in Section 4.3, the hybrid model (not using any training data) slightly outperforms supervised deep models (Feng et al., 2015) from in-domain training for two out of three datasets, while being significantly faster and more scalable – a highly desirable property for large-scale real-world applications. Note our original goal was to achieve no significant loss in accuracy while reducing costs. It is interesting

[7]Top-1 accuracy is one of the most commonly-used metrics to evaluate answer selection and question-answering. Other metrics include MRR (Radev et al., 2002) and MAP (Baeza-Yates et al., 1999). Since in this paper we focus on getting a correct answer rather than the entire ranking of answers, top-1 accuracy is reported.



(i) WikiQA (ii) TrecQA (iii) InsuranceQA(v2)

Figure 3: Top-1 test accuracy as the training dataset size is being varied from 0% to 100% of its original, in increments of 20%, for three QA datasets (i) WikiQA; (ii) TrecQA; and (iii) InsuranceQA(v2).

to see that the hybrid can surpass the expectation, and not only performs well for time, but for accuracy equally. This confirms our earlier observation that the simple attention mechanism used by the model works quite well in practice.

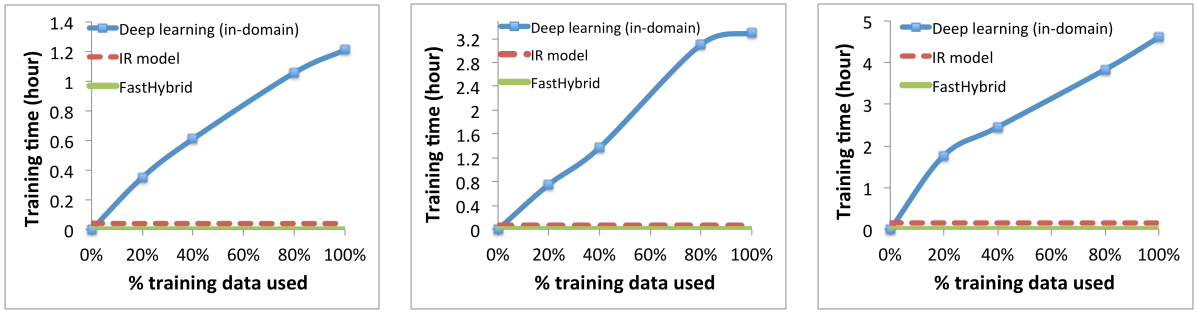
4.2 Training time

Table 3 presents the training time for each model. As mentioned earlier, all training and test experiments were carried out on a NVIDIA Tesla K20Xm GPU. As expected, both the IR model and *FastHybrid* are highly efficient. The supervised background models are more expensive, since they have to be trained on a large-scale background corpus. We note the training times of the supervised background models depend on the values of their hyper-parameters (e.g., # convolution filters, context window size etc). They are set in accordance to the settings used by the authors of these deep learning models. We also note that the supervised background models only have to be trained once. This is significant when we want to scale to a large number new domains (e.g., in the order of thousands). The saving in time across these many domains can add up to be quite noticeable. Nonetheless, when taking both accuracy and scalability/efficiency into account, we would like to have a model that has a higher accuracy yet not incurring too much costs (in training, tuning etc.), which is achieved by the hybrid model.

4.3 Test accuracy vs training set size

Next, we would like to ask the question – assume we follow the standard supervised setup where in-domain training data is used to train a model each time, can we improve deep learning model training efficiency by using a reduced set of the data and achieve similar test accuracy as using the full training set? Figure 3 and Figure 4 explore the effects of in-domain training set size on test accuracy and training time, respectively. Figure 3 shows the test accuracy as a function of the amount of training data used, and each reduced training set is a random sample from the full training data (i.e., sampled at 20%, ..., 80% of the full set). Figure 4 reports the corresponding training efficiency achieved at each training set size. Note it is clear for IR and the *FastHybrid*, they reside along the x-axis in Figure 4 which denotes their constant (0) training time.

Given the results shown earlier (Table 2) where the deep learning training method (Feng et al., 2015), when applied to the Yahoo background corpus (supervised background-1), slightly outperforms supervised background-2, we employ (Feng et al., 2015) for in-domain training, denoted by "Deep learning (in-domain)" in Figure 3 and Figure 4. Note at 100% training data, this represents a standard in-domain supervised deep learning model. As we can see from Figure 3, the hybrid model achieves equal/better accuracies for two out of three datasets (TrecQA and InsuranceQA), as compared to the supervised in-domain deep learning model from using *full* training data. Furthermore, for the deep learning model, it is clear that reduced training sets lead to much improved training efficiency. For example, at 20% of the original training set size (Figure 4), its training time drops to 0.35h, 0.75h, and 1.77h for WikiQA, TrecQA, and InsuranceQA, respectively – a decent improvement over its original time (1.21h, 3.3h, 4.6h, respectively). However, the enhanced training efficiency comes at a cost of reduced test accuracy. As



(i) WikiQA (ii) TrecQA (iii) InsuranceQA(v2)

Figure 4: Training time as the training dataset size is being varied from 0% to 100% of its original, in increments of 20%, for three QA datasets (i) WikiQA; (ii) TrecQA; and (iii) InsuranceQA(v2).

shown by Figure 3, when no training data is used, its test accuracy drops to 0%. In the future, it would be interesting to look into how to deal with very few training data for these techniques, a problem that has been drawing much interest recently (Socher et al., 2013; Romera-Paredes et al., 2015; Palatucci et al., 2014; Ba et al., 2015).

5 Conclusion

In this work, we have developed the *FastHybrid* model, a hybrid model which combines a fast model with an IR model to form an efficient and effective hybrid structure for the answer selection task. Unlike the previous deep learning models for this task, our hybrid model is nearly hyper-parameter/parameter-free, so it is extremely efficient (no training) for different datasets, and as a result, it can scale very easily to a large number of new domains and users. We performed a set of extensive experimental studies that demonstrate both the accuracy and training efficiency of our new method, as compared to several strong baselines noted for their accuracy and efficiency. In the future, we plan to explore applying this model to related tasks such as recommendation. In addition, we are interested in building and plugging in additional query performance prediction (QPP) models into our hybrid model.

6 Acknowledgements

We sincerely thank Bowen Zhou and Bing Xiang for comments that improved the manuscript, and Cicero dos Santos for assistance with the deep learning software used in this paper.

References

Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, and Ruslan Salakhutdinov. 2015. Predicting Deep Zero-Shot Convolutional Neural Networks using Textual Descriptions. In *IEEE International Conference on Computer Vision.*

Ricardo Baeza-Yates, and Berthier Ribeiro-Neto 1999. Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc.

Michael Bendersky, Donald Metzler, and Bruce Croft. 2010. Learning Concept Importance Using a Weighted Dependence Model. In *ACM International Conference on Web Search and Data Mining.*

Stefan Buttcher, Charles Clarke, and Brad Lushman. 2006. Term proximity scoring for ad-hoc retrieval on very large text collections. In *The 29th annual international ACM SIGIR conference on Research and development in information retrieval.*

Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. 2014. Building an efficient and scalable deep learning training system. In *The 11th USENIX conference on Operating Systems Design and Implementation.*

Sumit Chopra, Suhrid Balakrishnan, and Raghuraman Gopalan. 2013. DLID: Deep Learning for Domain Adaptation by Interpolating between Domains. In *ICML 2013 Workshop on Representation Learning.*

Oren Etzioni. 2011. Search needs a shake-up. In *Nature. 476(7358):25–26*

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: a study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU).*

David Ferrucci. 2012. Introduction to "This is Watson". In *IBM Journal of Research and Development. 56(3.4).*

Yaroslav Ganin, and Victor Lempitsky. 2015. Unsupervised Domain Adaptation by Backpropagation. In *International Conference on Machine Learning.*

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. In *International Conference on Machine Learning.*

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *International Conference on Acoustics, Speech and Signal Processing.*

Claudia Hauff, Leif Azzopardi, and Djoerd Hiemstra. 2009. The Combination and Evaluation of Query Performance Prediction Methods. In *European Conference on Information Retrieval.*

Ben He, and Iadh Ounis. 2005. Query Performance Prediction. In *Information Systems, 31(7).*

Sepp Hochreiter, and Jurgen Schmidhuber. 1997. Long short-term memory. In *Neural Computation.*

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *http://arxiv.org/pdf/1607.01759.pdf.*

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

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuogl. 2014. Recurrent Models of Visual Attention. In *Annual Conference on Neural Information Processing Systems.*

Mark Palatucci, Dean Pomerleau, Geoffrey E. Hinton, and Tom M. Mitchell. 2009. Zero-shot Learning with Semantic Output Codes. In *Annual Conference on Neural Information Processing Systems.*

Dragomir R. Radev, Hong Qi, Harris Wu, and Weiguo Fan. 2002. Evaluating Web-based Question Answering Systems. In *The International Conference on Language Resources and Evaluation.*

Bernardino Romera-Paredes, and Philip H. S. Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning.*

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive Pooling Networks. *http://arxiv.org/pdf/1602.03609.pdf.*

Aliaksei Severyn, and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Richard Socher, Milind Ganjoo, Christopher Manning, and Andrew Ng. 2013. Zero-Shot Learning Through Cross-Modal Transfer. In *Annual Conference on Neural Information Processing Systems.*

Baochen Sun, Jiashi Feng, and Kate Saenko. 2016. Return of Frustratingly Easy Domain Adaptation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16).*

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved Representation Learning for Question Answer Matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.*

Tao Tao, and ChengXiang Zhai. 2007. An exploration of proximity measures in information retrieval. In *The 30th annual international ACM SIGIR conference on Research and development in information retrieval.*

Ellen Voorhees. 2000. Overview of the Trec-9 Question Answering Track. In *TREC conference.*

Di Wang, and Eric Nyberg. 2015. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Eric P. Xing, Qirong Ho, Wei Dai, Jin Kyu Kim, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, Abhimanu Kumar, and Yaoliang Yu. 2015. Petuum: A new platform for distributed machine learning on big data. In *CM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question Answering Using Enhanced Lexical Semantic Models. In *The 51st Annual Meeting of the Association for Computational Linguistics.*

Lei Yu, Karl Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. In *NIPS Deep Learning Workshop.*

Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. 2012. Online Incremental Feature Learning with Denoising Autoencoders. In *The 15th International Conference on Artificial Intelligence and Statistics (AISTATS).*

Extracting Spatial Entities and Relations in Korean Text

Bogyum Kim and Jae Sung Lee
Dept. of Computer Science
Chungbuk National University
Chungdae-ro 1, Seowon-gu, Cheongju, 28644, Korea
bogyum@cbnu.ac.kr, jasonlee@cbnu.ac.kr

Abstract

A spatial information extraction system retrieves spatial entities and their relationships for geological searches and reasoning. Spatial information systems have been developed mainly for English text, e.g., through the SpaceEval competition. Some of the techniques are useful but not directly applicable to Korean text, because of linguistic differences and the lack of language resources. In this paper, we propose a Korean spatial entity extraction model and a spatial relation extraction model; the spatial entity extraction model uses word vectors to alleviate the over generation and the spatial relation extraction model uses dependency parse labels to find the proper arguments in relations. Experiments with Korean text show that the two models are effective for spatial information extraction.

1 Introduction

A spatial information extraction system retrieves spatially related lexical items and their relationships and then provides the information in a normalized form. This information is used for geological searches and reasoning, and ultimately, for understanding natural language text. For example, from the spatial relations that A is on B and B is on C, a human can simply infer the fact that A is on C. A spatial information extraction system retrieves the relations 'on (A, B)' and 'on (B, C)' from the text; then, a reasoning program can infer the relation 'on (A, C)' from the relations. This enables the system to build a knowledge base with a compact size from text for many intelligent systems such as robot navigation and question-answering systems.

A spatial information extraction task is usually carried out by two sub tasks: spatial entity extraction and spatial relation extraction. Because spatial entity extraction retrieves the entities to be used for spatial relationships, it is different from a place extraction task in named entity recognition systems (Lee et al. 2011), which only retrieves place-related entities. This implies that spatial entity extraction deals with all of the entities involved in spatial relations, such as trajectors, landmarks, and spatial signals. Moreover, spatial signals are usually articles or particles that do not have explicit arguments for spatial relations. Whether some entities or relations are extracted or not depends on the semantic roles in a sentence, which makes the task more complicated and challenging.

Many spatial information extraction systems have been developed for English text, especially those developed through the competition at the SpaceEval conference (Pustejovsky et al. 2015). The application of the techniques directly to Korean text is not simple because of its different linguistic features. The Korean language is a morphologically rich and agglutinative language, which is very different from English (Kim et al. 2016). Moreover, because Korean has a relatively free word order and words are frequently omitted, the order of neighboring words does not always have a significant meaning as in English, where it plays an important role in spatial word classification.

In this paper, we propose two models to extract spatial entities and spatial relations in Korean text. For entity extraction, an ensemble model is used to boost recall, and word vectors are used to tune the results for precision. For relation extraction, a sequence of the dependency labels from the trigger to the argument is used to calculate the argument probability. All of these extraction tasks are based on the ISO-Space mark-up scheme (Pustejovsky et al. 2015). In section 2, related works are briefly reviewed.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2389–2396, Osaka, Japan, December 11-17 2016.

The entity extraction method using GloVe word vectors (Pennington et al. 2014) and the relation extraction method using a Bayesian probability follow in sections 3 and 4, respectively. A discussion of the experiments and the conclusions follow in sections 5 and 6, respectively.

2 Related Works

There are generally two approaches to spatial entity extraction, similar to other natural language processing tasks: rule-based and data-driven approaches. The performance of a rule-based approach heavily depends on how much the dictionary covers the open words and how much the rule reflects the linguistic features. This approach usually needs a considerable amount of human labor to encode the rules and fill the dictionary entries manually. Moreover, it is language and domain dependent.

A data-driven approach uses machine learning tools such as CRFs and SVM. A spatial entity extraction task is considered to be a task of sequence labeling, which is solved by using CRFs (Kordjamshidi et al. 2010, Pustejovsky et al. 2015, Roberts and Harabagium 2012, Nichols and Botros 2015) and SVM (Bastianelli et al 2013). Data-driven approaches performed better than rule-based approaches in general and are easily portable to other domains.

The common features for spatial entity extraction based on machine learning are morphemes, named entities, word dependencies, semantic roles, and semantic information such as a WordNet category. Semantic information contributes to the performance. As resources such as WordNet are not easily available to many languages yet, word vectors were used by Bastianelli et al. (2013) and Nichols and Botros (2015), which are generated from a large raw corpus. The word vectors were used to provide semantic information as fine-grained lexical representations and clustered numbers. A spatial entity extraction system for Korean text has been developed by Kim et al. (2015) using a CRFs model, where morphemes, named entities, and parsing results are used as the features. It is based on the SpRL scheme corpus, and preliminary results were provided: with the test using 1,753 annotated sentences from Wikipedia, it was reported that the average F1 score of the entities is 0.610, whereas that of spatial relations is 0.318. As the entities and relations of the annotation scheme in ISO space are different, they are not compared directly to this paper's result.

Dependency parsing results are used for relation extraction. Cross et al. (2011) and Bastianelli et al. (2013) used a parse tree to construct a GRCT (Grammatical Relation Centered Tree) graph for an SVM tree kernel. Jeong et al. (2011) and Kwak et al. (2013) used dependency structures for the relation extraction of Korean sentences; the former used a composite kernel to extract general relations, and the latter built rules to extract spatial relations.

3 Entity Extraction

3.1 Base model

We define a base model called the E1 model, which incorporates the useful features used in prior systems that are applicable to the Korean language. Moreover, we have added more features to improve the performance, such as language-specific features, word phrase spacing, and morpheme-POS (part of speech) tag vectors. For describing the features for the base model, we define the acronyms for the feature description in Table 1. All of the CRFs features for the base model are defined in Table 2 using the acronyms, where a letter means an acronym defined in Table 1, and the attached number is the size of window. For example, MT3 means 'morpheme and POS tag pairs within a 3-morpheme window.' (We use a morpheme window here instead of word window in English text.)

Table 1. Acronyms for the feature elements.

M: morpheme	D: dependency label
T: part of speech tag	H: head's dependency label
B: BI tag of a word phrase spacing	W: main morpheme-POS tag of head
S: sense number of a morpheme	V: cluster number of a morpheme-POS tag vector
N: named entity tag	C: cluster number of the head's morpheme-POS tag vector

Table 2. Features of the E1 model for the CRFs.

All entities	M3, T3, MT3, B3, MS3, MM3, TT3, N3, D1, H1, W1, V3, C1

M	T	B	S	N	D	H	W	V	C
…	…	…	…	…	…	…	…	…	…
충북	NNP	B	00	B-OGG_EDU	B-NP	NP	대학교/NNG	211	181
대학교	NNG	B	00	I-OGG_EDU	B-NP	NP_AJT	안/NNG	181	298
안	**NNG**	**B**	**01**	**NONE**	**B-NP_AJT**	**VP**	**위치하/VV**	**298**	**103**
에	JKB	I	00	NONE	I-NP_AJT	VP	위치하/VV	185	103
위치하	VV	B	00	NONE	B-VP	VP	있/VX	103	185
…	…	…	…	…	…	…	…	…	…

M3: 대학교 **안** 에 TT3: NNG-**NNG** **NNG**-JKB

T3: NNG **NNG** JKB W1: **있/VX**

MT3: 대학교-NNG **안-NNG** 에-JKB V3: 181 **298** 185

Fig 1. Examples of element features shown in vertical forms and composite features.

3.2 Ensemble model

As the Korean spatial tagged corpus is not large and not well-balanced, machine learning programs such as CRFs are not learned properly. Therefore, we use multiple sub models to overcome the skewness in the data distribution. We assigned the respective features to each entity type, as summarized in Table 3. After testing a candidate with multiple sub models of each entity type, the results are simply accumulated. This is called the E2 model, and this is an interim model for the following final model.

Table 3. Features of each sub model for the CRFs in the E2 model.

Entity type	Feature list
PLACE, PATH	M3, T3, MT3, MM3, TT3, MS3, B3, N3, V3
SPATIAL_ENTITY	M3, T3, MT3, MM3, TT3, MS3, B3
MOTION	M3, T3, MM3, TT3, MS3, B3, V3, D1
MOTION_SIGNAL	M3, T3, MM3, TT3, H1, W1, C1
SPATIAL_SIGNAL	M3, T3, MM3, TT3, N3, H1, W1, C1
MEASURE	M5, T5, MM3, TT3, N3, V3

3.3 Ensemble model using word vectors

As the multiple sub models still produce many false entities, a word vector is used to filter them. The idea is that the common characteristics of entities can be represented in entity tag vectors by summing all of the word vectors learned from the training corpus. The entity tag vectors are used later during testing to check the validity of the candidate entity vectors. Eq. 1 expresses a formula used for the entity tag vector calculation, where the function f converts w_i into a vector representation, and eq. 2 expresses an equation used for the validation method during testing, where θ is the minimum cosine similarity between the entity tag vector (centroid) and a tagged word vector (instance), which is determined during training.

The vectors for the spatial entities are the word vectors of the entities themselves in the training data, as expressed in eq. 3. The function *w2v* converts an argument word into a vector representation using deep learning programs such as GloVe (Pennington et al. 2014). However, the vectors of the signal entities are the context word vectors of the signal words, as expressed in eq. 4, because the Korean

signals are usually particles, which are too general to be characterized for tag vectors. We propose this model for spatial entity extraction and call it the E3 model.

$$\overrightarrow{STag_j} = \frac{1}{N}\sum_{i=1}^{N} f(w_i) \quad (1)$$

$$STag(w_i) = argmax_j(cos(\overrightarrow{STag_j}, f(w_i)) > \theta_j) \quad (2)$$

$$f(w_i) = w2v(w_i) \quad (3)$$

for STag(w_i) ∈ {PLACE, PATH, SPATIAL_ENTITY, MOTION} or candidates

$$f(w_i) = \frac{1}{2L}\sum_{l=1}^{L}(w2v(context_l(w_i)) + w2v(context_{-l}(w_i))) \quad (4)$$

for STag(w_i) ∈ {MOTION_SIGNAL, SPATIAL_SIGNAL} or candidates

4 Relation Extraction

In ISO-Space, a spatial relation consists of two static relations and one dynamic relation. The static relations are the topological relation (QSLink) and orientational link (OLink), which are triggered by SPATIAL_SIGNAL. The extracted relations are represented in a triple format: <trajector, trigger, landmark>. The dynamic relation is the move relation (MoveLink) triggered by a MOTION event. The extracted relation is represented in octuple format: <mover, trigger, source, goal, landmark, mid-point, path, motion signal>. However, it is not easy to extract all the octuplet arguments in most sentences. For a relaxed implementation, the octuple is converted into many triples; then, one or all of the triples are extracted (Nichols and Botros 2015, D'Souza and Ng 2015). We chose to extract one of those triples, <mover, trigger, goal>, because its arguments are filled in most cases, and the triple is also chosen for extraction target in (Nichols and Botros 2015).

4.1 Rule-based model

A rule-based method is straight-forward to implement, if linguistic regularities for spatial relation extraction are easily found. For a performance comparison, we also define a rule-based relation extraction model as a base model and call it the R1 model. Because of the free word order and frequent omission of words in the Korean language, regularities are not easily found. Therefore, the rules do not pose many restrictions, as summarized in Table 4.

Table 4. Relation extraction rules.

Rule 1	Static relations are triggered by SPATIAL_SIGNAL and dynamic relations by MOTION.
Rule 2	SPATIAL_SIGNAL with the type 'TOPOLOGICAL' generates QSLink, and that with the type 'DIRECTIONAL' generates both OLink and DIR_TOP.
Rule 3	The arguments for the relations are 'PLACE,' 'PATH,' and 'SPATIAL_ENTITY.'
Rule 4	All spatial entities for a relation are within the same dependency head (VP, VNP).
Rule 5	When there is more than one argument under a dependency head, the argument closest to the trigger in the dependency relation is classified as a landmark, and the other arguments are classified as trajectors, resulting in multiple relations.
Rule 6	If one and more triggers exist, the arguments cannot cross the other triggers at a sentence position.

4.2 Bayesian model

Dependency parsing is quite effective for free word order languages such as the Korean language and provides useful information for long-distance relations (Lim et al. 2014). We utilize the parsing result to find valid arguments for given triggers such as the SPATIAL_SIGNAL or MOTION tag. In this model, all of the possible argument candidates are searched and verified with a Bayesian probability, which is learned with the training corpus. The argument with the highest probability is chosen, as shown in eq. 5. The probability is calculated as the product of the prior probability of an argument and the conditional probability of the sequence of dependency labels (DPL). The DPL includes the labels from a trigger to the argument, and its conditional probability is approximated in eq. 6.

$$\mathrm{DPL} = (dpl_1, dpl_2, \ldots, dpl_n)$$
$$\mathrm{A} = \{\text{trajector}, \text{landmark}\}$$
$$argmax_{A_i} P(A_i, DPL) = argmax_{A_i} P(DPL|A_i) \cdot P(A_i) \quad (5)$$
$$\begin{aligned} \mathrm{P}(DPL|A_i) &= P(dpl_n, dpl_{n-1}, \ldots, dpl_1|A_i) \\ &= \mathrm{P}(dpl_n|dpl_{n-1}, \ldots, dpl_1, A_i) \cdot P(dpl_{n-1}|dpl_{n-2}, \ldots, dpl_1, A_i) \cdots P(dpl_1|A_i) \\ &\cong \mathrm{P}(dpl_1|A_i) \cdot \prod_{j=2}^{n} P\left(dpl_j \middle| dpl_{j-1}\right) \end{aligned} \quad (6)$$

5 Experiment

5.1 Experimental setup

As pre-processing steps, a morphological analysis and POS tagging, named entity recognition, and dependency parsing are carried out, and their sources and performance are summarized in Table 5. We used CRFsuite (Okazaki 2007) and the GloVe word vector (Pennington et al. 2014). The word vectors are trained with the morpheme-tagged data in the Sejong corpus (NIKL 2011) to build 300 vector clusters for a feature set.

For the test data, we used the Korean spatial annotation corpus (Kim et al. 2016), which is constructed from 175 documents (1593 sentences) from the Wikitravel web-site[1] following the SpaceEval annotation scheme (Pustejovsky et al. 2015). The testing corpus statistics are listed in Table 6.

The experiment was performed with 5-fold cross validation test. The experiment for entity extraction was directly carried out with the raw corpus data, and relation extraction was performed with the corpus annotated with spatial entities beforehand. (Each experiment corresponds to tasks 1.b and 3.a of SpaceEval)

Table 5. Performance of the pre-processing modules.

Modules	performance	source
Morph. analysis and POS tagging.	99.03% (pre)	(Lee et al. 2016)
Named entity recognition	86.86% (f1)	(Lee et al. 2011)
Dependency parsing	87.63% (LAS)	(Lim et al. 2014)

Table 6. Number of tags in the testing corpus.

Entity						relation		
name	num	ratio	name	num	ratio	name	num	ratio
PLACE	5,636	67.6%	M._SIGNAL	266	3.2%	QSLink	3,548	65.9%
PATH	320	3.8%	S._SIGNAL	1,299	15.6%	OLink	970	18.0%
S._ENTITY	270	3.2%	MEASURE	248	3.0%	MoveLink	868	16.1%
MOTION	294	3.5%						
			entity total	8,333	100.0%	relation total	5,386	100.0%

5.2 Results

Table 7 summarized the results of spatial entity extraction. The performance of MEASURE is the second-best because its typical surface form, e.g., the "number + unit" form, is very easily recognized by a program. PLACE is the best performer, which can also be easily found by a named entity recognizer. Moreover, as the ratio of PLACE tag is the largest, 67.6%, in the distribution as presented in Table 6, the prior probability contributes to find more PLACE tags. On the other hand, SPATIAL_ENTITY exhibits the worst performance. We conjecture that the first reason is that the size of the training corpus is too small; the number of Korean spatial entity tags is 270 (3.2%), as presented in Table 6, whereas that of English spatial entity tags is 1670 (23.6%) in the corpus used for SpaceEval 2015 (Pustejovsky et al. 2015). The second reason is that its part of speech tag is a general noun, which is not easy to distinguished from other spatial tags. Moreover, the same word can be either a SPATIAL_ENTITY tag or

[1] http://www.wikitravel.com/ko/

none tag depending on the context. In the following example, 'car' in the first sentence is tagged SPATIAL_ENTITY, but that in the second sentence is not. This is same for English, but it is more difficult for a free-word-order language to be disambiguated.

철수가 **자동차/spatial_entity** 에 탔다. (Cheolsu got in a **car/spatial_entity**.)

철수가 **자동차/none** 를 샀다. (Cheolsu bought a **car/none**.)

The E2 model increased the recall but decreased the precision, as indicated in Table 7. However, the E3 model using spatial tag vectors greatly increased the precision by 13.3% point compared to that of the E2 model. Consequently, the F1 measure performance of the E3 model increased by 2.1% point compared to that of the E1 model, which means that use of the tag vector is effective for selecting valid spatial entities.

Table 7 summarizes a comparison of the E3 model with SpRL-CWW (Nichols and Botros 2015), which was the best model at SpaceEval 2015 (Pustejovsky et al. 2015). The overall performance of the E3 model is better than that of SpRL-CWW for all precision, recall, and F1 measure criteria. The performance of SPATIAL_ENTITY and PATH for the E3 model, however, is relatively much lower than that of SpRL-CWW. This means that the ambiguity of general nouns in a semantic role is still problematic and the size of the training corpus is relatively smaller as fore-mentioned.

The performance of relation extraction is summarized in Table 8. All of the values of the precision, recall, and F1 measure for the R2 model are better than those for the R1 model. This implied that the use of the Bayesian probability for selecting arguments is effective. For a general comparison, we have listed the results of the two best approaches in the table, where CWW (Nichols and Botros 2015) is the best machine learning approach, and Pust (Pustejovsky et al. 2015) is the best rule-based approach. Unfortunately, the R2 model exhibits a very low performance compared with both of them.

The spatial relations in Korean text are relatively hard to be retrieved, because the related entities are relatively separated and their appearing order is not consistent. While all related entities are usually located closely to the spatial signal in English text, the entities are sometimes far from the spatial signal in a Korean sentence. Sentence 1 in Fig. 2 shows an example in which the trajector is far from the trigger. In addition, the appearing order is not consistent as shown in Korean sentence 2 in Fig. 2. Both landmark and trajectory appear before the trigger in the first OLINK, whereas landmark appears before and trajectory appears after the trigger in the second OLINK. We used the dependency relations of words to alleviate this problem and thus improved the performance. However, we still need to find more effective methods to overcome Korean linguistic barriers such as the free word order and the lack of language resources; and this problem will be studied in future research.

Table 7. Performance of spatial entity extraction.

Label	Precision				Recall				F1			
	E1	E2	**E3**	CWW	E1	E2	**E3**	CWW	E1	E2	**E3**	CWW
PLACE	0.919	0.917	**0.961**	0.802	0.928	0.930	**0.958**	0.777	0.923	0.924	**0.960**	0.789
PATH	0.848	0.441	**0.552**	0.815	0.397	0.543	**0.539**	0.614	0.541	0.487	**0.545**	0.701
S. ENTITY	0.463	0.210	**0.326**	0.793	0.213	0.444	**0.444**	0.653	0.292	0.285	**0.376**	0.716
MOTION	0.801	0.354	**0.544**	0.823	0.479	0.713	**0.709**	0.7	0.600	0.473	**0.616**	0.756
M. SIGNAL	0.800	0.236	**0.556**	0.766	0.392	0.698	**0.694**	0.6	0.536	0.353	**0.617**	0.673
S. SIGNAL	0.851	0.770	**0.892**	0.75	0.729	0.794	**0.836**	0.603	0.786	0.782	**0.863**	0.668
MEASURE	0.990	0.951	**0.951**	0.889	0.881	0.906	**0.906**	0.707	0.936	0.928	**0.928**	0.788
Overall	0.894	0.728	**0.861**	0.795	0.849	0.855	**0.880**	0.674	0.849	0.786	**0.870**	0.73

E1: base model, E2: ensemble model, E3: proposed ensemble model using word vector,
CWW: 5-fold cross validation (Nichols and Botros 2015)

Table 8. Performance of spatial relation extraction.

Relation	Precision				Recall				F1			
	R1	**R2**	CWW	Pust	R1	**R2**	CWW	Pust	R1	**R2**	CWW	Pust
QSLink	0.40	**0.49**	0.66	-	0.51	**0.55**	0.54	-	0.45	**0.52**	0.59	-
OLink	0.12	**0.24**	0.69	-	0.19	**0.48**	0.52	-	0.15	**0.32**	0.59	-
MoveLink	0.18	**0.24**	0.57	-	0.35	**0.65**	0.45	-	0.24	**0.35**	0.5	-
Overall	0.30	**0.37**	0.64	0.86	0.42	**0.54**	0.50	0.84	0.35	**0.44**	0.56	0.85

R1: base model, R2: proposed model using dependency label, CWW: (Nichols and Botros 2015), Pust: Baseline 3.a (Pustejovsky et al. 2015)

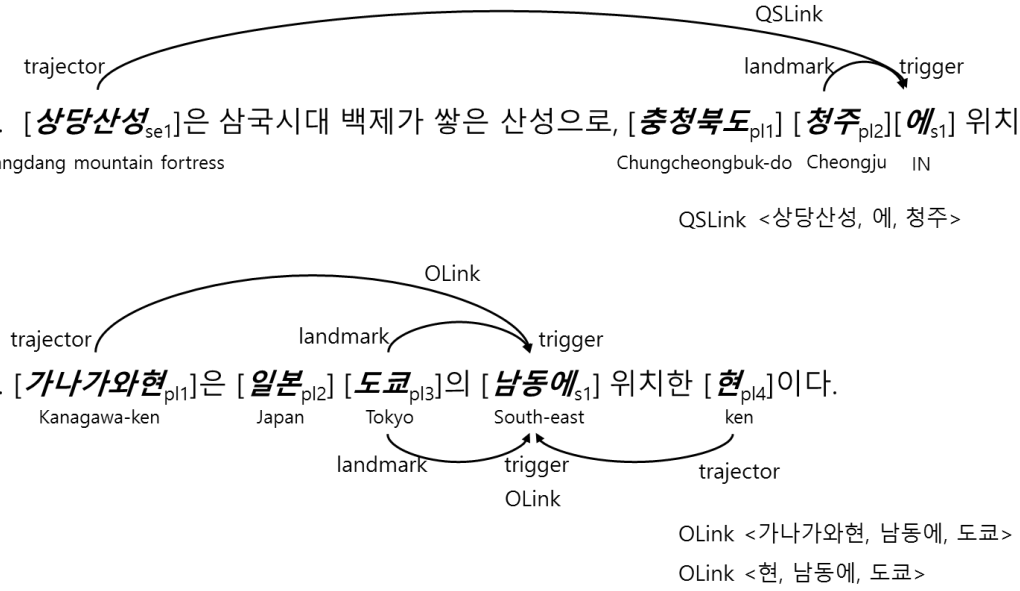


Fig 2. Various types of spatial relation caused by free word order.

6 Conclusion

We have proposed two models for Korean spatial entity extraction and spatial relation extraction. For entity extraction, we utilized the features of prior systems with an adaptation to Korean linguistic features. Moreover, we proposed a new approach to filter false entities using spatial tag vectors, which can be learned automatically from a raw corpus. The experiment showed that the spatial tag vectors are effective for spatial entity extraction and showed better performance than English state-of-the-art performance.

For relation extraction, we proposed a model that uses the dependency label probability to select proper arguments, which is effective and better than a simple rule-based model but much lower than the state-of-the-art performance of an English one. We conjecture that this mainly originates from linguistic differences, especially syntactic structures such as the free word order and word omission, which still require further investigation.

Acknowledgement

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP). (No. R0101-16-0062, Development of Knowledge Evolutionary WiseQA Platform Technology for Human Knowledge Augmented Services)

Reference

Bogyum Kim, Yongmin Park, and Jae Sung Lee. 2015. Automatic space information extraction from Korean text. *Journal of Information*. 18(7):2953-2962.

Bogyum Kim, Myung Yun Kang, and Jae Sung Lee. 2016. Issues in spatial information annotation in Korean texts. *The First International Workshop on Spatial/Temporal Information Extraction from Unstructured Texts*, pages 458-461.

Changki Lee, Pum-Mo Ryu, and HyunKi Kim. 2011. Named entity recognition using a modified Pegasos algorithm. *In Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 2337-2340. ACM.

Chang-Hoo Jeong, Sung-Pil Choi, Yun-Soo Choi, Sa-Kwang Song, and Hong-Woo Chun. 2011. Relation extraction based on composite kernel combining pattern similarity of predicate-argument structure. *Journal of Internet Computing and Services*, 12(5):73-85.

Chung-Hee Lee, Joon-Ho Lim, Soojong Lim, and HyunKi Kim. 2016. Syllable-based Korean POS tagging based on combining a pre-analyzed dictionary with machine learning. *Journal of Korean Institute of Information Scientists and Engineers*, 43(3):362-369.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1034-1046, Association for Computational Linguistics.

Emanuele Bastianelli, Danilo Croce, and Roberto Basili. 2013. UNITO-HMM-TK: Structured kernel-based learning for spatial role labelling. *In Second Joint Conference on Lexical and Computational Semantics*, SemEval 2013, pages 573-579.

Eric Nichols and Fadi Botros. 2015. SpRL-CWW: Spatial relation classification with independent multi-class models. *In Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 895-901.

Haritz Salaberri, Olatz Arregi, and Beñat Zapirain. 2015. IXAGroupEHUSpaceEval: (X-Space) A WordNet-based approach towards the automatic recognition of spatial information following the ISO-Space annotation scheme. *In Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 856-591.

James Pustejovsky, Parisa Kordjamshidi, Marie-Francine Moens, Aaron Levine, Seth Dworman, and Zachary Yocum. 2015. SemEval-2015 task 8: SpaceEval. *In Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 884-894.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532-1543.

Jennifer D'Souza and Vincent Ng. 2015. UTD: Ensemble-based spatial relation extraction. *In Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 862-869.

Joon-Ho Lim, Yeo-Chan Yoon, Yongjin Bae, Su-Jong Im, Hyunki Kim, and Kyu-Chul Lee. 2014. Korean dependency parsing model based on transition system using head final constraint. *In Proceedings of the 27th Annual Conference on Human & Cognitive Language Technology*, pages 81-86.

Kirk Roberts and Sanda M. Harabagium. 2012. UTD-SpRL: A joint approach to spatial role labeling. *In Proceedings of the First Joint Conference on Lexical and Computational Semantics Volume 2: Proceedings of the 6th International Workshop on Semantic Evaluation*, pages 419-424. Association for Computational Linguistics.

Naoaki Okazaki. 2007. CRFsuite: A first implementation of conditional random fields (CRFs). *http://www.chokkan.org/software/crfsuite/*.

NIKL (National Institute of Korean Language). 2011. 21st century Sejong project final result, revised edition.

Parisa Kordjamshidi, Martijn Van Otterlo, and Marie-Francine Moens. 2010. Spatial role labeling: Task definition and annotation scheme. *In Proceedings of the 7th Conference on International Language Resources and Evaluation*, pages 413-420, European Language Resources Association.

Sujeong Kwak, Bogyum Kim, and Jae Sung Lee. 2013. Triplet extraction using Korean dependency parsing result. *In Proceedings of the 25th Annual Conference on Human & Cognitive Language Technology*, pages 86-89.

Hybrid Question Answering over Knowledge Base and Free Text

Kun Xu[1], **Yansong Feng**[1,*], **Songfang Huang**[2] and **Dongyan Zhao**[1]
[1]Institute of Computer Science & Technology, Peking University, Beijing, China
[2]IBM China Research Lab, Beijing, China
{xukun, fengyansong, zhaody}@pku.edu.cn
huangsf@cn.ibm.com

Abstract

Recent trend in question answering (QA) systems focuses on using structured knowledge bases (KBs) to find answers. While these systems are able to provide more precise answers than information retrieval (IR) based QA systems, the natural incompleteness of KB inevitably limits the question scope that the system can answer. In this paper, we present a hybrid question answering (hybrid-QA) system which exploits both structured knowledge base and free text to answer a question. The main challenge is to recognize the meaning of a question using these two resources, i.e., structured KB and free text. To address this, we map relational phrases to KB predicates and textual relations simultaneously, and further develop an integer linear program (ILP) model to infer on these candidates and provide a globally optimal solution. Experiments on benchmark datasets show that our system can benefit from both structured KB and free text, outperforming the state-of-the-art systems.

1 Introduction

Recently, with the emergence of large structured knowledge bases (KBs) like DBpedia (Auer et al., 2007), Freebase (Bollacker et al., 2008) and Yago (Suchanek et al., 2007), increasing research efforts on automatically answering natural language questions has shifted from using text corpora only to large scale structured KBs like DBpedia, Freebase (known as KB-QA). Compared to pure text resources used in IR-based QA systems, structured knowledge bases may help to provide users with more accurate and concise answers, especially for factoid questions.

Generally, the traditional KB-QA paradigm assumes that world knowledge can be encoded using a closed vocabulary of formal predicates. In this paradigm, the system is given a knowledge base as input, and the question answering problem reduces to semantic parsing, i.e., mapping from text to logical forms containing the predicates from the given knowledge base. However, the closed predicate vocabulary assumed by the traditional KB-QA paradigm has inherent limitations. First, a closed predicate vocabulary has limited coverage, as such vocabularies are typically powered by community efforts. Second, a closed predicate vocabulary may abstract away potentially relevant semantic differences. Third, even a logical form was produced, the answers may be incomplete due to the imperfection of the KB, which has been addressed by (Riedel et al., 2013; Chen et al., 2014). For example, no logical form could be produced for the question *who is the front man of the band that wrote Coffee & TV*. Because the semantics of *front man* cannot be adequately encoded using Freebase or DBpedia predicates.

On the other hand, knowledge bases like DBpedia capture real world facts, and web resources like Wikipedia may provide a large repository of sentences that complement those facts. For instance, we can find in Wikipedia a sentence *In August 2009, Debelle performed at Africa Express in Paris, an event set up by Blur and Gorillaz front-man Damon Albarn*, which indicates the front man of the band in the example question is *Damon Albarn*[1]. Moreover, text corpora is also shown effective in refining the answers retrieved from the KBs (Xu et al., 2016). Motivated by these observations, we tackle the

[1]The Blur band wrote the Coffee & TV song.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2397–2407, Osaka, Japan, December 11-17 2016.

question answering task by integrating these two types of heterogeneous data, i.e., structured knowledge bases and free text, while is rarely investigated before.

This task involves three main challenges. The first is how to represent the meaning of a question by the clues from two types of heterogeneous resource. Secondly, for each phrase, there exist multiple grounded candidates over the KB and text corpora, how to perform inference on these candidates itself is a problem. The third challenge is how to properly incorporate the coherence of two types of heterogeneous resource, KB predicates and textual relations, into the inference model.

In this paper, we propose a joint inference approach to simultaneously solve these disambiguations. Specifically, our method consists of two main steps as outlined in (§2). In the first step, we employ preliminary models to perform the entity linking and relation extraction (§3). Next, we develop an integer linear program (ILP) model, where the candidate mapping of phrases to KB items and textual relations are the variables restricted by several designed constraints, and they could be determined simultaneously through joint inference (§4). The main contributions of this paper are two folds:

- We introduce a new task paradigm of the question answering community, and present a novel hybrid-QA framework to accommodate the structured KB and free text.
- We propose a joint inference model to solve the disambiguation among entities and relations across text and KBs.

Our evaluation results on benchmark datasets show that our system benefits from the integration of the KB and free text outperforming the state-of-the-art systems.

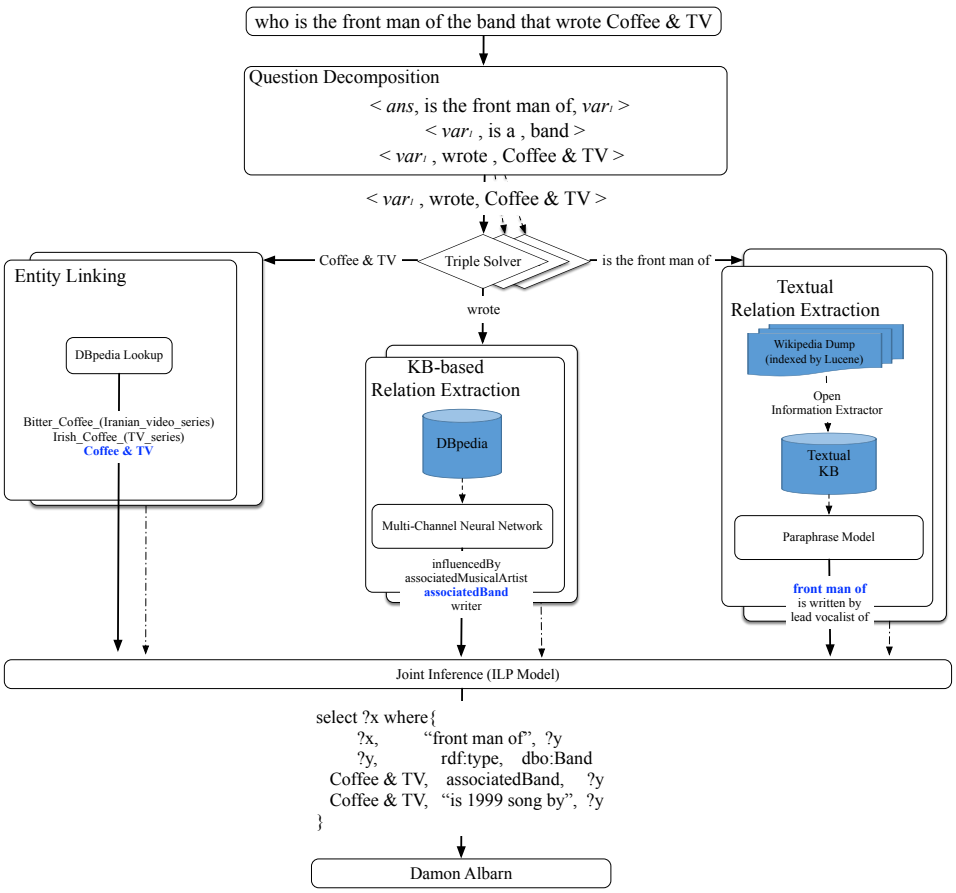


Figure 1: A running example of our hybrid-QA system for the question *who is the front man of the band that wrote Coffee & TV*, where the blue annotations are correct.

2 Our Method

Figure 1 gives an overview of our method for the aforementioned question "*who is the front man of the band that wrote Coffee & TV*". We have two main steps: (1) perform the local predictions, i.e., Entity Linking (EL) and Relation Extraction (RE); and (2) further infer on the retained candidate entities, KB predicates and textual relations to find an optimal assignment under certain constraints.

Let us take a close look into step 1. Here we first perform entity linking to identify possible KB entities in the question. Then we employ two types of relation extractors to predict both KB predicates and textual relations existing between two entities or question word and entities in the question. Specifically, we propose a neural network based method to map relational phrases to KB predicates, and apply a paraphrase model to find most likely textual relations that describe the phrases. In Step 2, we perform a joint inference over the local predictions of EL and RE models to find a best configuration through an ILP model.

As shown in Figure 1, it is often the case that a question may involve multiple relations. Consider the example question, the answers of this question should satisfy the following two constraints: (1) the person is the front man of a band (textual relation); and (2) the band wrote the song *Coffee & TV* (KB predicate). We use the 6 syntax-based rules as introduced in (Xu et al., 2016) to preprocess such *multi-relational* questions, i.e., decomposing them into a set of simple questions formulated as *ungrounded* triples. For instance, the example question can be decomposed into three ungrounded triples: $<$*ans*, *is the front man of*, $var_1>$, $<var_1$, *is a*, *band* $>$ and $<var_1$, *wrote*, *Coffee & TV*$>$[2].

3 Prelimiary Models

Since we represent the meaning of a question using clues from two types of heterogeneous resources, we tackle the QA problem in an IE-based fashion involving entity linking and relation extraction. In particular, we simultaneously map relational phrases to KB predicates and textual relations.

3.1 Entity Linking

The preliminary entity linking model can be any approach which outputs a score for each entity candidate. Note that a recall-oriented model will be more than welcome, since we expect to introduce more potentially correct local predictions into the inference step. In this paper, we adopt DBpedia Lookup[3] and S-MART (Yang and Chang, 2015) to retrieve top 10 entities from DBpedia and Freebase, respectively. These entities are treated as candidate entities that will be eventually disambiguated in the joint inference step.

3.2 KB-based Relation Extraction

The choice of KB-based relation extraction model is also broad. In this paper, we employ the **M**ulti-**C**hannel **C**onvolutional **N**eural **N**etworks (MCCNNs) model presented in (Xu et al., 2016) to learn a compact and robust relation representation. This is crucial since there exist thousands of relations in a KB, using lexicalized features inevitably suffers from the sparsity problem and their poor generalization ability on unseen words (Gormley et al., 2015).

The MCCNN model treats the conjunction of three parts in a ungrounded triple as a sentence (**subject** relational phrase **object**). The first channel takes the shortest path between the subject and object in the dependency tree[4] as input, while the other channel takes the relational phrase itself as input. Each channel uses the network structure described in (Collobert et al., 2011), which uses a convolutional layer to project the word-trigram vectors of words within a context window of 3 words to a local contextual feature vector, followed by a max pooling layer that extracts the most salient local features to form a fixed-length. The global feature vector is then fed to feed-forward neural network layers to output the final non-linear semantic features, as the vector representation of the relational phrase.

[2]Here ans denotes the answer and var_1 denotes an intermediate variable.

[3]http://wiki.dbpedia.org/projects/dbpedia-lookup.

[4]We use Stanford CoreNLP dependency parser.

Learning The model is learned using pairs of relational phrase and its corresponding KB predicate. Given an input phrase, the network outputs a distribution vector over the predicates o. We denote t as the target distribution vector, in which the value for gold relation is set 1, others are set 0. We compute the cross entropy error between t and o as the loss function. The model parameters can be efficiently computed via back-propagation through network structures. In experiment, we train two distinct relation extractors over DBpedia and Freebase, respectively. For DBpedia, we use the PATTY dataset (Nakashole et al., 2012) which consists of 127,811 pairs of relational phrases and DBpedia predicates involving 225 DBpedia predicates. For Freebase, we use 3,022 phrase-predicate pairs of WEBQUESTIONS used in (Xu et al., 2016), which involves 461 Freebase predicates.

3.3 Open Relation Extraction

Despite huge amounts of precise knowledge facts, structured KBs still have natural limitation in the coverage of knowledge domains compared to the vast information on the web. For example, out of 500,000 relations extracted by the ReVerb Open IE system (Fader et al., 2011), only about 10,000 can be aligned to Freebase (Berant et al., 2013). To alleviate this problem, we propose a paraphrase based method that can map relational phrases to proper textual relations. Specifically, we first apply an open information extractor (Angeli et al., 2015) on the English Wikipedia to construct a repository of $<$*argument*$_1$, *relation*, *argument*$_2>$ triples, where the *arguments* are entity phrases found in the input sentence and the *relation* represents certain relationship between the arguments. By linking these arguments to KB entities, we can obtain a textual knowledge repository.

Paraphrasing Once the candidate set of textual relations $TR = \{tr_1, tr_3, ..., tr_{|TR|}\}$ are constructed, given a relational phrase rp, our goal is to find the tr that has the same meaning as rp, which can be treated as a paraphrase task. Our framework accommodates any paraphrasing method, such as the method based on dynamic pooling and recursive autoencoders (RAE) (Socher et al., 2011), which we adopt in our framework. Generally, the RAEs are based on a novel unfolding objective and learn feature vectors for phrases in syntactic trees. These features are used to measure the word-wise and phrase-wise similarities between two sentences. Since sentences may be of arbitrary length, the resulting matrix of similarity measures is of variable size. Then a dynamic pooling layer is introduced to compute a fixed-sized representation from the variable-sized matrices. Finally the pooled representation is used as input to a classifier $\mathcal{C}_p$.

Learning In our experiment, we directly used the pre-trained RAE which is trained on a subset of 150,000 sentences from the NYT and AP sections of the Gigaword corpus. To train the classifier $\mathcal{C}_p$, we use the PARALEX corpus (Fader et al., 2013), which is a large monolingual parallel corpora, containing 18 million pairs of question paraphrases from `wikianswers.com`, which were tagged as having the same meaning by the users of the website.

4 Joint Inference

The goal of the inference step is to find a global optimal configuration of entity phrases and relational phrases with semantic components. As the result of disambiguating one phrase can influence the mapping of other phrases, we consider all phrases jointly in one disambiguation task. Now, we will first describe three key criteria that are used to evaluate the configuration in details.

KB Predicate and Entity's Coherence If the relational phrase rp is grounded to a KB predicate kr, we should examine whether the semantic types of the entities fulfill the expectations of KB predicates. Particularly, we first obtain the type of subject entity e, which is collected from the KB's schema, and examine whether there exists another entity with the same type taking the subject position of this predicate in the KB. If such an entity exists, it indicates this entity is compatible with the KB predicate, $Coh_{e,kr} = 1$, otherwise 0.

Textual Relation and Entity's Coherence Similarity, we also need to capture the coherence, $Coh_{e,tr}$, between a textual relation tr and entity e. Since the textual relation does not have well-defined schemas

like the KB, we practically treat the types of collected entities that take the subject and object position of tr as the type expectations of tr. For instance, *written by* takes `Coffee&TV` (a song) and `Blur` (an English band), which indicates the type expectations of *written by* should include `Song` and `Band`. We then determine whether e is compatible with tr by examining whether the type of e fulfills the type expectations of tr. If e is compatible with tr, $Coh_{e,tr} = 1$, otherwise 0.

KB Predicate and Textual Relation's Coherence Notice that, we allow a relational phrase to be simultaneously mapped to a KB predicate and a textual relation. In this case, the KB predicate kr and textual relation tr should be compatible with each other. For this purpose, we first determine if kr and tr have the same argument expectations. If so, we use the trained MCCNN to capture the coherence of a KB predicate kr and textual relation tr, $Coh_{kr,tr}$. In practice, we treat this problem as a variant of relation classification, i.e., the coherence score is the probability of mapping word sequence tr to KB predicate kr. Otherwise, $Coh_{kr,tr}$ is set to -1.

Integer Linear Program Formulation Now we describe how we aggregate the above components, and formulate the joint inference problem into an ILP framework. Given the above definitions, our objective function is to maximize the score of entity linking, relation extraction and their coherence among them:

$$\max \quad \alpha \times conf^e + \beta \times conf^r + \delta \times conf^{er} \tag{1}$$

where α, β and δ are weighting parameters tuned on development set. $conf^e$ is the overall score of entity linking:

$$conf^e = \sum_d \sum_{ep \in d, e \in C_e(ep)} w_{ep,e} Y_{ep,e} \tag{2}$$

where d is the ungrounded triple, $C_e(ep)$ is the candidate entity set of the entity phrase ep, $w_{ep,e}$ is the entity linking score, and $Y_{ep,e}$ is a boolean decision variable that indicates if entity phrase ep maps to entity e. $conf^r$ represents the overall score of relation extraction:

$$conf^r = \sum_d \sum_{rp \in d, kr \in C_{kr}(rp)} q_{rp,kr} Z_{rp,kr} + \sum_d \sum_{rp \in d, tr \in C_{tr}(rp)} v_{rp,tr} W_{rp,tr} \tag{3}$$

where $C_{kr}(rp)$ is the set of candidate KB predicates of relation phrase rp, $C_{tr}(rp)$ is the set of candidate textual relations corresponding to rp, $q_{rp,kr}$ and $v_{rp,tr}$ are the scores of relational phrase rp mapped to KB relation kr and textual relation tr. We define two boolean decision variables $Z_{rp,kr}$ and $W_{rp,tr}$ to denote whether rp is mapped to kr and tr. coh^{er} evaluates the coherence between the candidate entities and relations in the framework:

$$conf^{er} = \sum_d \sum_e \sum_{kr} o_{e,kr} Coh_{e,kr} + \sum_d \sum_e \sum_{tr} o_{e,tr} Coh_{e,tr} + \sum_d \sum_{kr} \sum_{tr} o_{kr,tr} Coh_{kr,tr} \tag{4}$$

where $o_{e,kr}$, $o_{e,tr}$ and $o_{kr,tr}$ are the coherence scores among entities, KB predicates and textual relations. We introduce three boolean decision variables $Coh_{e,kr}, Coh_{e,tr}, Coh_{kr,tr}$ to denote whether two semantic components are both selected.

Constraints Now we describe the constraints used in our ILP problem. The first kind of constraints is introduced to ensure that each entity phrase should be disambiguated to only one entity:

$$\forall d, \forall e \in C_e(ep), \sum_{ep \in d, e \in C_e(ep)} Y_{ep,e} \leq 1 \tag{5}$$

The second type of constraints ensure that each relational phrase should be disambiguated to only one KB relation or one textual relation *at most*:

$$\forall d, \forall kr \in C_{kr}(rp), \sum_{rp \in d, kr \in C_{kr}(rp)} Z_{rp,kr} \leq 1 \tag{6}$$

$$\forall d, \forall tr \in C_{tr}(rp), \sum_{rp \in d, tr \in C_{tr}(rp)} W_{rp,tr} \leq 1 \tag{7}$$

The third constraint ensures the decision variable $Coh_{e,kr}$ equals 1 if and only if both the corresponding variables $Y_{ep,e}$ and $Z_{rp,kr}$ equal 1.

$$\forall d, \forall e \in C_e(ep), \forall kr \in C_{kr}(rp), \forall tr \in C_{tr}(rp) \tag{8}$$

$$Coh_{e,kr} \leq Y_{ep,e} \quad Coh_{e,kr} \leq Z_{rp,kr} \quad Y_{ep,e} + Z_{rp,kr} \leq 1 + Coh_{e,kr} \tag{9}$$

Similarly, we further add the following constraints for $Coh_{e,tr}$ and $Coh_{kr,tr}$:

$$Coh_{e,tr} \leq Y_{ep,e} \quad Coh_{e,tr} \leq W_{rp,tr} \quad Y_{ep,e} + W_{rp,tr} \leq 1 + Coh_{e,tr} \tag{10}$$

$$Coh_{kr,tr} \leq Z_{rp,kr} \quad Coh_{kr,tr} \leq W_{rp,tr} \quad Z_{rp,kr} + W_{rp,tr} \leq 1 + Coh_{kr,tr} \tag{11}$$

We use Gurobi[5] to solve the above ILP problem.

Method	WebQ	QALD-6
Bordes et al. (2014)	39.2	-
Dong et al. (2015)	40.8	-
Yao (2015)	44.3	-
Bast (2015)	49.4	-
Berant (2015)	49.7	-
Reddy et al. (2016)	50.3	-
Yih et al. (2015)	52.5	-
Xu et al. (2016)	53.3	-
This work		
KB	44.1	10.1
KB + Joint	47.1	14.3
Text	40.3	28.7
Text + Joint	45.5	37.4
KB + Text + Joint	**53.8**	**40.9**

Table 1: Results on the test set of QALD-6 and WEBQUESTIONS.

QALD-6
What is the most common language in norway
What currency do they use in switzerland
When olympic games 2012 opening ceremony
What countries does queen elizabeth ii reign
What is the best sandals resort in st lucia
What did the islamic people believe in
WEBQUESTIONS
What is the largest city in the county in which Faulkner spent most of his life
Under which pseudonym did Charles Dickens write some of his books
Where was the Father of Singapore born
Which German mathematicians were members of the von Braun rocket group
Who is the architect of the tallest building in Japan

Table 2: Example questions from WEBQUESTIONS and QALD-6.

5 Experiment

In this section we evaluate our system on two benchmark datasets, QALD-6 and WEBQUESTIONS. After describing the setup, we present our main empirical results and analyze the components of our system.

The QALD-6 task[6] includes a hybrid QA dataset which contains 50 training questions and 25 test questions. We select 15 questions from the training set as the development set and use the remaining 60 ones to evaluate our system.

We also use the WEBQUESTIONS dataset (Berant et al., 2013), which contains 5,810 question-answers pairs. We further split this dataset into the same training and test sets as other baselines, which contain 3,778 questions (65%) and 2,032 questions (35%), to evaluate the system.

As shown in Table 2, these two datasets vary significantly in both syntactic and semantic complexity. For example, 85% questions of WEBQUESTIONS can be directly answered via a single Freebase predicate. However all questions of QALD-6 involve at least one DBpedia predicate and one textual relation, thus can not be accurately answered using DBpedia only.

5.1 Experimental Settings

We have 6 dependency tree patterns based on Bao et al. (2014) to decompose a question into sub-questions. We initialize the word embeddings with Turian et al. (2010)'s word representations with dimensions set to 50. The hyper parameters in our model are tuned using the development set. The window size of MCCNN is set to 3. The sizes of the hidden layer 1 and the hidden layer 2 of the two MCCNN channels are set to 200 and 100, respectively. For each relational phrase, we retain 20 candidate KB predicates and textual relations to the ILP model. The hyper parameters of the ILP objective function (i.e., α, β and δ) are set to 1, 3 and 4, respectively.

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

[6]http://qald.sebastianwalter.org/index.php?q=6

5.2 Results and Discussion

We use the average question-wise F_1 as our evaluation metric. To give an idea of the impact of different configurations of our method, we consider the following variations with existing methods.

KB. This method involves prediction relying on the KB only in a pipelined fashion. First the entity linking system is run to predict the entity. Then we run the KB-based relation extraction system (described in §3.2) and select the best relation that can cooccur with the entity. We choose this entity-relation pair to predict the answer.

KB + Joint. In addition to selecting local optimal results, we further perform the joint inference over entity and KB predicates.

Text. Instead of applying a KB-based RE method, we map the relation phrase to textual relations as described in §3.3 and find a local optimal solution.

Text + Joint. This method augments the above method with a joint inference step.

KB + Text + Joint. This is our main model. We perform the entity linking, map the relation phrase to KB predicates and textual relations simultaneously, and then infer on the local predications to find a global optimal assignments of the phrases.

Table 1 summarizes the results on the test data along with the results from the literature[7]. We can see that the joint inference gives a performance boost of at least 3% (from 44.1% to 47.1%) regardless of using which type of relation extractor. In addition, text corpora can significantly improve the system performance when using the KB only, and vice versa. The combination of structured KB and free text along with the joint inference outperforms the default model by at least 3.5% (from 37.4% to 40.9%). On the WEBQUESTIONS, our method achieves a new state-of-the-art result beating the previous reported best result of Xu et al. (2016) (with one-tailed t-test significance of $p < 0.05$). And our results on QALD-6 also establishes a new baseline.

5.3 Impact of Textual Relations and KB Predicates

As shown in Table 1, KB-based relation extractor performs better than textual relation extractor on WEBQUESTIONS, but worse on QALD-6. This is due to the fact that WEBQUESTIONS is designed to evaluate the KB-QA systems, therefore the involved relations are guaranteed to be explicitly mapped to KB predicates. In contrast, QALD-6 is proposed to evaluate hybrid-QA systems, and almost no question can be answered using a KB only. Although different datasets have different appetites for the relation extractors, we find the combination of them significantly improves the overall performance.

We also compared our paraphrase model (RAE) with two baselines: EDIT-based and VECTOR-based paraphrase models. Specifically, the former computes the token edit distance between the textual relation tr and relation phrase rp as the similarity score, obtaining 43.6% and 35.4% F_1 on the development set of WEBQUESTIONS and QALD-6, respectively.

The latter obtains the vector representations of tr and rp by summing the word vectors (Turian et al., 2010), and compute the cosine similarity as the similarity score, obtaining 45.7% and 39.3% F_1 on the development set of WEBQUESTIONS and QALD-6, respectively. We find the RAE paraphrase model boosts the performance at least by 6% on QALD-6 and 2% on WEBQUESTIONS.

5.4 Impact of ILP's Constraints

One question of interest is when the ILP model prefers to mapping relational phrases to KB predicates and textual relations simultaneously. We mainly rely on the coherence score between KB predicates and textual relations, i.e., $Coh_{kr,tr}$, to guide the inference model to find a proper assignments. Specifically, if kr and tr have the same argument type expectations, we compute the $Coh_{kr,tr}$ as the probability of mapping tr to kr using the neural network as described in §3.2. Otherwise, $Coh_{kr,tr}$ is set to -1. The

[7]We list several recent results on WEBQUESTIONS. We use development data for all our ablation experiments. Similar trends are observed on both development and test results.

intuition behind is that the selected pair of KB predicates and textual relations should first be coherent, and then semantically similar. If there does not exist such a coherent pair, the model prefers to choosing the one which has higher overall score and neglects the other.

5.5 Error analysis

We analyze the errors of *KB + Text + Joint* model. Around 2% of the errors are caused by incorrect entity linking, and around 5% of the errors are due to incorrect question decomposition. The remaining errors are due to the relation extraction: (i) unbalanced distribution of KB predicates heavily influences the performance of MCCNN model towards frequently seen relations as observed in (Xu et al., 2016); (ii) the RAE model can hardly find proper assignments of textual relations for short-length relational phrases.

5.6 Limitations

While our inference on the structured KB and free text allows the system to answer more open questions to some extent, we still fail at answering some semantically complex questions such as *what is the second longest river in USA* involving aggregation operations. Our current assumption that free text could provide useful textual relations may work only for frequently typed queries or for popular domains like movies, politics and geography. We note these limitations and hope our result will foster further research in this area.

6 Related Work

Over time, the QA task has evolved into two main streams – QA on unstructured data, and QA on structured data. TREC QA evaluations (Voorhees and Tice, 1999) have been explored as a platform for advancing the state of the art in unstructured QA (Wang et al., 2007; Heilman and Smith, 2010; Yao et al., 2013; Yih et al., 2013; Yu et al., 2014; Yang et al., 2015; Hermann et al., 2015). While initial progress on structured QA started with small toy domains like GeoQuery (Zelle and Mooney, 1996), recent trend in QA has shifted to large scale structured KBs like DBPedia, Freebase (Unger et al., 2012; Cai and Yates, 2013; Berant et al., 2013; Kwiatkowski et al., 2013), and on text repository (Banko et al., 2007; Carlson et al., 2010; Krishnamurthy and Mitchell, 2012; Fader et al., 2013; Parikh et al., 2015). An exciting development in structured QA is to exploit multiple KBs (with different schemas) at the same time to answer questions jointly (Yahya et al., 2012; Fader et al., 2014; Zhang et al., 2016).

Our model combines the best of both worlds by inferring over the structured KB and unstructured text. Our work is closely related to Joshi et al. (2014) who aim to answer noisy telegraphic queries using both structured and unstructured data. Their work is limited in answering single relation queries. Our work also has similarities to Sun et al. (2015) who does question answering on unstructured data but enrich it with Freebase.

Joint inference methods over multiple local models has been applied to KB-QA systems (Yahya et al., 2012). In contrast to this prior work concentrating on the structured KB, our constraints are more complex, as we address the joint mapping of relational phrases onto KB predicates and textual relations.

7 Conclusion and Future Work

We have presented a hybrid-QA framework that could infer both on structured KBs and unstructured text to answer natural language questions. Our experiments reveal that integrating structured KB and unstructured text along with a joint inference method improves the overall performance. Our main model achieves the state-of-the-art results on benchmark datasets. A potential application of our method is to improve open domain question answering using the documents retrieved by a search engine.

Since we recognize the query intention inherent in the question using shallow methods, our method is less expressive than the deep meaning representation methods like semantic parsing. Our future work involves developing a shallow semantic parser based on relation extraction in order to better understand the meaning of the questions.

Acknowledgments

We would like to thank Weiwei Sun, Liwei Chen, and the anonymous reviewers for their helpful feedback. This work is supported by National High Technology R&D Program of China (Grant No. 2015AA015403, 2014AA015102), Natural Science Foundation of China (Grant No. 61202233, 61272344, 61370055) and the joint project with IBM Research. For any correspondence, please contact Yansong Feng.

References

Gabor Angeli, Melvin Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Association for Computational Linguistics (ACL)*.

Sren Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. Dbpedia: A nucleus for a web of open data. In *ISWC/ASWC*.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction for the web. In *IJCAI*.

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *ACL*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *EMNLP*.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*.

Liwei Chen, Yansong Feng, Songfang Huang, Yong Qin, and Dongyan Zhao. 2014. Encoding relation requirements for relation extraction via joint inference. In *ACL*, pages 818–827.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *ACL-IJCNLP*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*.

Anthony Fader, Luke S. Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *SIGKDD*.

Matthew R. Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *MENLP*, pages 1774–1784, Lisbon, Portugal, September. Association for Computational Linguistics.

Michael Heilman and Noah A Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *NAACL*.

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

Mandar Joshi, Uma Sawant, and Soumen Chakrabarti. 2014. Knowledge graph and corpus driven segmentation and answer inference for telegraphic entity-seeking queries. In *EMNLP*.

Jayant Krishnamurthy and Tom M Mitchell. 2012. Weakly supervised training of semantic parsers. In *EMNLP-CoNLL*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP*.

Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek. 2012. PATTY: A taxonomy of relational patterns with semantic types. In *EMNLP*, pages 1135–1145.

Ankur P. Parikh, Hoifung Poon, and Kristina Toutanova. 2015. Grounded semantic parsing for complex knowledge extraction. In *NAACL*.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics*, 4.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS.*, pages 801–809.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*.

Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. 2015. Open domain question answering via semantic enrichment. In *WWW*.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL*, pages 384–394.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over rdf data. In *WWW*.

Ellen M Voorhees and Dawn M. Tice. 1999. The trec-8 question answering track report. In *TREC*.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question Answering on Freebase via Relation Extraction and Textual Evidence. In *Proceedings of the Association for Computational Linguistics (ACL 2016)*, Berlin, Germany, August. Association for Computational Linguistics.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural language questions for the web of data. In *EMNLP*.

Yi Yang and Ming-Wei Chang. 2015. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. In *ACL-IJNLP*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*.

Xuchen Yao, Benjamin Van Durme, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *NAACL*.

Xuchen Yao. 2015. Lean question answering over freebase from scratch. In *NAACL*.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *ACL*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL-IJCNLP*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI*.

Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. 2016. A joint model for question answering over multiple knowledge bases. In *AAAI*.

Improved Word Embeddings with Implicit Structure Information

Jie Shen **Cong Liu***
School of Data and Computer Science, Sun Yat-sen University
shenjie5@mail2.sysu.edu.cn, liucong3@mail.sysu.edu.cn

Abstract

Distributed word representation is an efficient method for capturing semantic and syntactic word relations. In this work, we introduce an extension to the *continuous bag-of-words* model for learning word representations efficiently by using implicit structure information. Instead of relying on a syntactic parser which might be noisy and slow to build, we compute weights representing probabilities of syntactic relations based on the Huffman softmax tree in an efficient heuristic. The constructed "implicit graphs" from these weights show that these weights contain useful implicit structure information. Extensive experiments performed on several word similarity and word analogy tasks show gains compared to the basic *continuous bag-of-words* model.

1 Introduction

Unsupervised word embeddings have been shown to improve many downstream NLP tasks, such as dependency parsing (Chen and Manning, 2014; Kong et al., 2014), part-of-speech tagging (Collobert et al., 2011), sentiment analysis (Socher et al., 2013) and machine translation (Devlin et al., 2014). Low-dimensional embeddings are generally learnt in a language model by maximizing the likelihood of a large corpus of raw text data. Word vectors that are close to each other are semantically related based on the *distributional hypothesis* (Harris, 1954), which states that words in similar contexts have similar meanings.

Based on the *distributional hypothesis*, many methods of building word vectors were explored. Examples of these are *SENNA* (Collobert and Weston, 2008), the *hierarchical log-bilinear* model (Mnih and Hinton, 2009), `Word2Vec` (Mikolov et al., 2013a; Mikolov et al., 2013b) and *GloVe* (Pennington et al., 2014). In this work, we introduce an extension to the *continuous bag-of-words* (CBOW) model (Mikolov et al., 2013b). The CBOW model is widely used for learning word embeddings from raw textual data, popularized via the `Word2Vec` tool. Not only does it build useful word embeddings, but it is also efficient for training and scales well to huge corpora.

In (Levy and Goldberg, 2014), based on the fact that nearby words are not necessarily syntactically related, word context is derived from dependency parse-trees relying on a syntactic parser instead of simply using the surrounding words. Only the words that have dependency relations with the center word are used as the context words, as illustrated in Figure 1.

However, syntactic parsing is a more difficult and time-consuming task than finding word embeddings. The challenge is to absorb the advantage of using syntactic information while avoiding the complexity of parsing. In this paper, we use a simple method to attach different weights to the context words to approximate the context obtained from explicit syntactic trees, such as dependency parse-trees. The method of obtaining contextual weights is based on the Huffman softmax tree in softmax period. Our method is as efficient as CBOW since we do not explicitly construct syntactic trees but only use the weights representing implicit syntactic structures.

*Cong Liu is the corresponding author.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2408–2417, Osaka, Japan, December 11-17 2016.

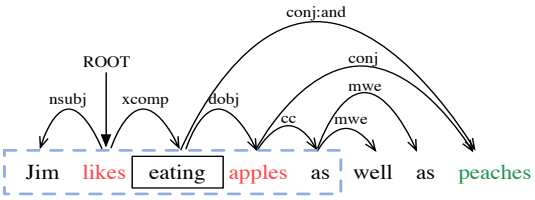


Figure 1: The enhanced dependency parse-graph for sentence "*Jim likes eating apples as well as peaches*". The blue dashed rectangle means the context window when predicting *eating* in the CBOW model.

In order to qualitatively inspect our implicit structure weights, we construct "implicit graphs" using our computed weights for different sentences. By comparing with their dependency parse-trees, these "implicit graphs" show that these weights contain useful implicit structure information.

To quantitatively evaluate the quality of the word embeddings generated from our proposed model, we experiment on several word similarity and word analogy tasks. Experiment results show gains using our method compared to the CBOW model.

The rest of the paper is organized as follows. Section 2 introduces related work on word representations. The CBOW model is briefly reviewed in Section 3. In Section 4, we present the proposed method. Evaluation and results are discussed in Section 5. Finally, Section 6 concludes the paper with future work.

2 Related Work

Word embedding is a key component in many downstream NLP tasks. Prior works explore effective and efficient methods to learn word embeddings. Bengio et al. (2006) proposed a *Neural Network Language Model* (NNLM) which predicts the distribution of the center word through several previous words. Mnih and Hinton (2007) proposed the *Log-Bilinear Language* (LBL) model which has been later accelerated by using hierarchical softmax (Mnih and Hinton, 2009) to exponentially reduce the computational complexity. Pennington et al. (2014) introduced *GloVe* which combines global matrix factorization and local context window together. Mikolov et al. (2013a; 2013b) proposed `Word2Vec` which contains two models: CBOW and Skip-Gram.

While `Word2Vec` is not sensitive to word order, many models have been proposed to deal with this problem. Ling et al. (2015) present two simple modified models: "Structured Skip-n-gram" and "Continuous Window", solving syntax-based problems. From another perspective, Levy and Goldberg (2014) use another type of context which uses word contexts derived from dependency parse-trees.

3 Continuous Bag-of-Words (CBOW)

Our departure point is the *continuous bag-of-words* (CBOW) model introduced in (Mikolov et al., 2013b). The CBOW model predicts the center word w_t given the representations of the surrounding words $w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c}$, where c is a hyper-parameter defining the window size of context. The objective function is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\log p(w_t|w_{t-c}^{t+c}),$$

where T is the size of a sequence of words. The basic CBOW defines the probability of predicting the center word w_t using the softmax function:

$$p(w_t|w_{t-c}^{t+c}) = \frac{\exp({\mathbf{v}_t'}^{\top}\mathbf{v}_{t-c}^{t+c})}{\sum_{i=1}^{V}\exp({\mathbf{v}_i'}^{\top}\mathbf{v}_{t-c}^{t+c})},$$

where $\mathbf{v}_t^{'}$ is the word embeddings of w_t and $\mathbf{v}_{t-c}^{t+c}$ is the context representation. V is vocabulary size. $\mathbf{v}_{t-c}^{t+c}$ is calculated by summing the word representations of context words:

$$\mathbf{v}_{t-c}^{t+c} = \sum_{-c \leq j \leq c, j \neq 0} \mathbf{v}_{t+j},$$

where $\mathbf{v}_{t+j}$ is the word representation of word w_{t+j}.

In order to improve the computation speed of the full softmax layer, several efficient extensions are proposed including hierarchical softmax and negative sampling (Mikolov et al., 2013b).

4 CBOW with Implicit Structure Information

We will discuss the proposed CBOW-CW model in three parts. Section 4.1 explains the advantages of adding implicit structure information to CBOW. In Section 4.2, we introduce a more advanced model using implicit structure information to offer theoretical soundness of the later CBOW-CW model. Then we derive a simplified method CBOW-CW what we actually have done in this paper in Section 4.3.

4.1 Motivation

While the window size c is a hyper-parameter of the CBOW model which is fixed before the training starts, the model samples the actual window size between 1 and c uniformly for each token in actual implementation. This scheme is equivalent to weighting according to distances from the center word divided by the window size (Levy et al., 2015).

However, this might not be reasonable, since a word from a large distance away can also be informative. For instance, in Figure 1, the words *apples* and *peaches* are equally important for predicting the word *eating*, while their relative distances to *eating* are different.

To solve this problem, Levy and Goldberg (2014) use word contexts derived from the dependency parse-trees. Figure 1 shows an enhanced dependency parse-graph which is generated using the Stanford parser (Chen and Manning, 2014). In the CBOW model, when predicting the word *eating*, the context words are *Jim*, *likes*, *apples* and *as* for a size-2 window. However, in the model of Levy and Goldberg (2014), the context words are words that have dependency relations with the center word *eating*: *likes*, *apples* and *peaches*. Note that the word *peaches*, one of the modifiers of *eating* that is not a context word in CBOW model, is now taken into the set of context words.

However, this method relies on syntactic parsing which is time-consuming. The challenge is how to absorb the advantages of a syntax-based context while avoiding the complexity of parsing. In the following, we first introduce a more advanced model using implicit structure information. Then we derive a simplified method CBOW-CW to attach different weights to the context words to approximate the effect of different context words obtained from syntactic trees.

4.2 Advanced Implicit Syntax-based Model

In this section, we present a more advanced implicit-syntax-based model. The implicit-syntax-based model assigns a weight α_{t+j} to each context word w_{t+j} given a center word w_t. If we could rely on a syntactic parser, we could compute α_{t+j} by summing up the number of syntactic relations between w_{t+j} and w_t on syntactic trees. Alternatively, in the implicit-syntax-based model, we assume a neural network to predict the weights $\boldsymbol{\alpha}$ using the center word and its context words, without relying on explicit syntactic trees.

To summarize, the implicit-syntax-based model contains two components as illustrated in Figure 2. The first component sums up the word vectors of the context words with weights, and then passes the sum to the softmax layer to predict the probability of the center word. The second component is a neural network that predicts these weights. To train them, we can use the alternative optimization method which optimizes one component at a time assuming the other component is optimal. To simplify discussion, we temporarily assume that a full softmax model is used. The first CBOW-with-weights component predicts the center word by:

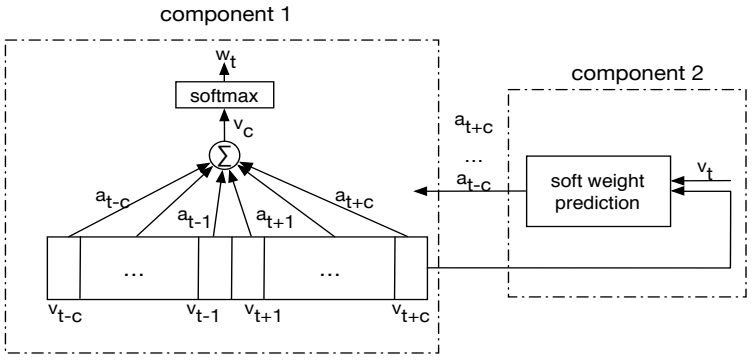


Figure 2: The two components in the implicit syntax-based model.

$$p(w_t|w_{t-c},\cdots,w_{t-1},w_{t+1}\cdots,w_{t+c}) = softmax(U \sum_{-c\leq j\leq c, j\neq 0} \alpha_{t+j}\mathbf{v}_{t+j}) = softmax(U\boldsymbol{\alpha}_t V_t^\intercal),$$

where U is the parameter in the full softmax layer, and the i-th column in U represents the expected context vector $\mathbf{u}_i$ of each word w_i in the vocabulary, $\boldsymbol{\alpha}_t$ is a vector consists of weights $\alpha_{t-c},\cdots,\alpha_{t-1},\alpha_{t+1},\cdots,\alpha_{t+c}$ and $V_t^\intercal$ is the transpose of the matrix concatenating the word embeddings of the context words, i.e. $\mathbf{v}_{t-c},\cdots,\mathbf{v}_{t-1},\mathbf{v}_{t+1}\cdots,\mathbf{v}_{t+c}$. During an iteration in the alternative optimization, the "soft weights" predictor component is optimized in order to predict a weight α_t for each center word w_t, such that the weighted sum $\mathbf{v}_c$ of the context word vectors approximates the expected context vector $\mathbf{u}_t$ of the center word. However, we can use a simple method to approximate the "soft weights" predictor component, since the optimal $\boldsymbol{\alpha}_t$ such that $\boldsymbol{\alpha}_t V_t^\intercal = \mathbf{u}_t$ can be solved analytically, assuming that the CBOW-with-weights is optimal, by $\boldsymbol{\alpha}_t = \mathbf{u}_t(V_t^\intercal)^{-1}$.

However, the actual CBOW model uses a Huffman softmax tree instead of a full softmax tree for improving performance, where context vectors $\mathbf{u}_t$ cannot be obtained directly. Therefore, we design a simple heuristic to calculate $\boldsymbol{\alpha}_t$ as described in Section 4.3. To sum up, our simple CBOW-CW model is an efficient approximation to the more advanced implicit-syntax-based model described above.

4.3 CBOW with Context Weights (CBOW-CW)

Now that the advanced implicit-syntax-based model offers the theoretical soundness of our idea, we can design our model to be simple, so that it is efficient to run and incremental to implement based on CBOW and, hopefully, other word embedding algorithms. We introduce the CBOW-CW model which is actually what we have done in this paper.

This work generalizes the CBOW model by attaching different weights α's to different context words. We denote $\mathbf{v}_{t-c}^{t+c}$ as the context representation of a center word w_t. It is the weighted sum of the word vectors $\mathbf{v}_{t+j}$ of context words of w_t, and is defined below:

$$\mathbf{v}_{t-c}^{t+c} = \sum_{-c\leq j\leq c, j\neq 0} \alpha_{t+j}\mathbf{v}_{t+j}.$$

In Levy and Goldberg (2014), a dependency parse-tree is used to determine the context of each word. For a center word, its neighbor nodes, including its head word and its modifier words in the dependency parse-tree are assigned a weight 1, and the other words are assigned a weight 0 in a sense. Instead of using a single best dependency parse-tree and assigning context words with "hard weights", i.e. 0's and 1's, a natural alternative is to ask a dependency parser to return a number of best dependency parse-trees and their probabilities, and then assign context words with "soft weights". In CBOW-CW, we define "soft weights" in the same sense, but we compute them using a simple and efficient method that does not explicitly rely on dependency parse-trees. The concrete method we use to compute the "soft weights" is described in the following part.

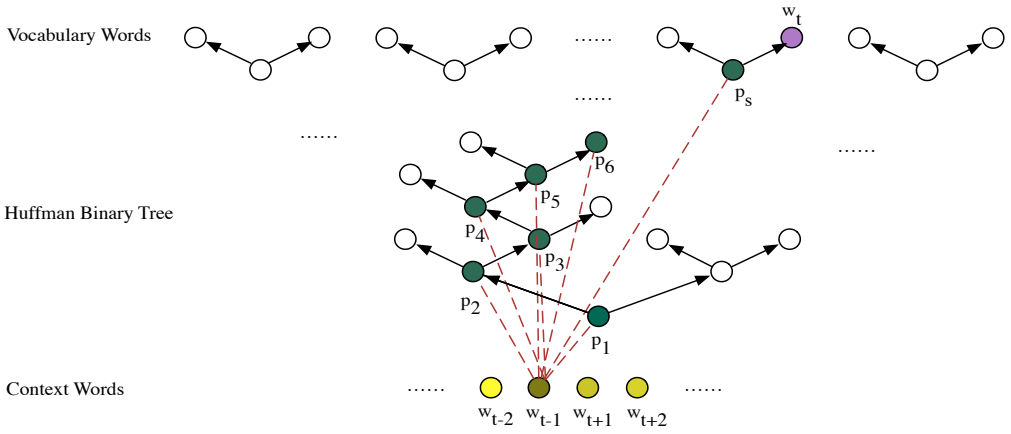


Figure 3: Computing the "soft weight" for each word w_{t+j} in the Huffman softmax tree of CBOW. Nodes on the top are the words in the vocabulary, which are leaf nodes of the binary softmax tree. The $P_t = \{p_1, p_2, \ldots, p_s\}_t, w_t$ is a path from the root node to the center word w_t in the Huffman softmax tree, where p_1 is the root node. The red dashed lines indicate the dot product of a context word vector and an vector associated with the nodes in the softmax tree.

Computation of the "Soft Weights" Before discussing the details of CBOW-CW, we need to talk about the Huffman softmax tree in CBOW. CBOW uses a Huffman softmax tree to accelerate its softmax layer. As shown in Figure 3, the leaf nodes of the Huffman softmax tree are the words to predict in the vocabulary, and each internal node is associated with a vector. To predict a word given its context vector $\mathbf{v}_c$, a path $P = \{p_1, p_2, \cdots, p_s, w_t\}$ is found from the root node p_1 of the tree to the predicted leaf node w_t. This path is determined hop-by-hop: the next node p_{k+1} of the partially computed path $\{p_1, p_2, \cdots, p_k\}$ is determined by the sign of the inner-product $\mathbf{v}_c \cdot \mathbf{v}_k$, where the vector $\mathbf{v}_k$ is associated with p_k. If the inner product is positive, p_{k+1} is the left child of p_k, otherwise the right child of p_k.

In CBOW-CW, we compute the "soft weight" α_{t+j} of context word w_{t+j} given the center word w_t as follows. Let $P_t = \{p_1, p_2, \cdots, p_s, w_t\}$ be the path from the root p_1 to the center word w_t in the Huffman softmax tree. We know that for a particular context word, once a "hop" is wrong, the rest of the path will be wrong. The weight α_{t+j} is vaguely defined as its "contribution" in finding the path P_t. This "contribution" is here defined as the number of correct next hops on path P_t that are computed, if the context vector $\mathbf{v}_c$ was replaced by $\mathbf{v}_{t+j}$. That is, the number of correct hops on the path that can be predicted, if we use context word w_{t+j} alone instead of the complete set of context words of w_t. The weight is a measure of similarity between a context word vector and the sum of context vectors. Concretely, the weight α_{t+j} for the context word w_{t+j} is defined as:

$$\alpha_{t+j} = \frac{\sum_{1 \le k \le s} 1\{(\mathbf{v}_k \cdot \mathbf{v}_{t+j}) \cdot (\mathbf{v}_k \cdot \mathbf{v}_c) > 0\}}{Z},$$

where s is the length of the path P_t. The indicator function $1\{x\}$ returns 1 iff x is $true$. $\mathbf{v}_k$ is the vector associated with node p_k on path P_t. $\mathbf{v}_{t+j}$ is the word vector of context word w_{t+j}. Z is the normalization factor: $Z = \sum_{-c \le j \le c, j \ne 0} \alpha_{t+j}$.

5 Experiments

To qualitatively analysis our method of attaching implicit syntactic weights, we make comparisons of CBOW-CW and CBOW, and we also visualize "implicit graph" for several sentences to compare with dependency parse-trees. To quantitatively evaluate the quality of the word embeddings generated from the proposed model, we experiment on several word similarity and word analogy tasks.

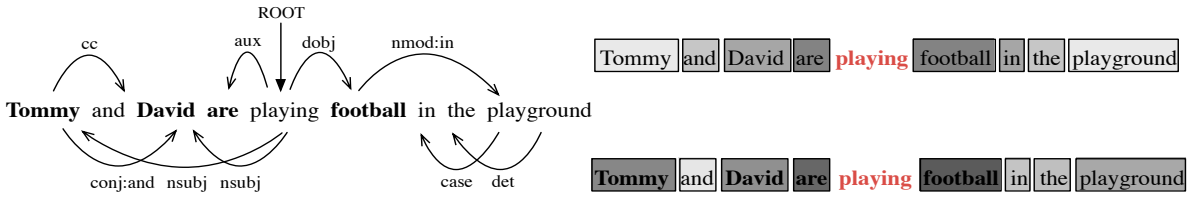


Figure 4: A sentence "*Tommy and David are playing football in the playground* " and its extended dependency parse-tree (top). The context weights in CBOW model (middle). The context weights in CBOW-CW (bottom). The center word is "playing". Darker colors indicate larger weights.

5.1 Comparison of CBOW-CW and CBOW

As illustrated in Figure 4, we show the difference between CBOW-CW and the CBOW using the sentence "*Tommy and David are playing football in the playground* " with the center word *playing* and the context window size 4. The dependency parse-graph is generated using the Stanford parser (Chen and Manning, 2014). In the CBOW model, the context words are *Tommy*, *and*, *David*, *are*, *football*, *in*, *the* and *playground*. The weights of the context words become smaller as their distances to the center word increase. However, one can see that *Tommy*, *David*, *are* and *football* are modifiers of *playing* so that they should be more important for predicting *playing*. In contrast, the proposed CBOW-CW model is able to attach more reasonable weights to the context words: the weights of *Tommy*, *David*, *are* and *football* are higher than most of the other words in the sentence.

5.2 Visualization of Implicit Syntactic Graph

In order to inspect our implicit syntactic weights, we visualize the "implicit graph" constructed using the "soft weights" for several sentences as illustrated in Figure 5. These sentences have different grammatical structures and they are chosen randomly. We draw undirected edges between two words if the inner-product of their word vectors exceed a threshold. The constructed graphs are not identical to the dependency parse-graphs. The dependency parse-graphs are built on human defined syntactic relations. There are other relations between words in reality, and therefore our "soft weights" are less restricted. Nevertheless, from the comparison of the "implicit graphs" and the enhanced dependency parse-graphs, we can find that our efficiently computed "implicit graphs" implicitly contain some dependency relations. This shows that our "soft weights" contain useful implicit structure information.

5.3 Word embeddings

We experiment with a large number of hyper-parameters. The space of hyper-parameters explored in this work is shown in Table 1. `win`, `neg`, `dim` and `ite` denote the window size, the number of negative examples, the dimension of word vectors and training iterations, respectively. The model will discard words that appear less than `min` times.

Training Corpora We built word embeddings using the original CBOW implementation[1] and our modified model on an English Wikipedia dump[2] containing about 1,989 million words, pre-processed by removing non-textual elements, sentence splitting and tokenization. The default value of the hyper-parameter `min` is 5, which means filtering out words with less than 5 instances and resulting in a vocabulary of 1,953,057 words. We also use the text8 corpus provided in `Word2Vec` package as a smaller dataset. The text8 corpus is the first 10^8 bytes of fil9, which is a 715 MB file filtered from the 1 GB file enwiki9.

Training Details For both the original CBOW and our CBOW-CW, we set the learning rate to the default value 0.05 and decrease it as the training process (Mikolov et al., 2013a). The settings of other hyper-parameters are showed in Table 1.

[1]https://code.google.com/p/word2vec/
[2]Collected in November of 2015.

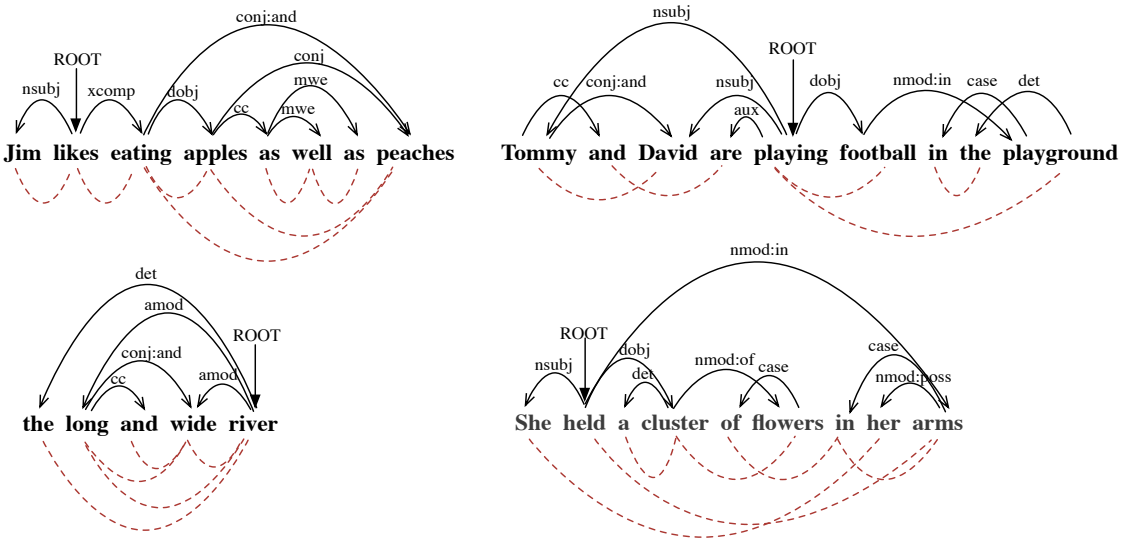


Figure 5: Constructed "implicit graphs" for several sentences. The enhanced dependency parse-graphs (shown in dark solid curves) above the sentences and the visualization of "implicit graphs" using our implicit syntactic weights (shown in red dashed curves) below the sentences.

Hyper-params	values	enwiki	text8
`win`	3, 5, 10, 20	10	10
`neg`	0, 8	8	8
`dim`	50, 100, 200, 300	300	200
`min`	5, 20, 50, 400	5	5
`ite`	5, 10	10	10

Table 1: The space of hyper-parameters explored in this work (the second column). The setting of hyper-parameters at which we report the results when using the enwiki corpus (the third column). The setting of hyper-parameters at which we report the results when using the text8 corpus (the forth column).

5.4 Test Datasets

We evaluate the word representations on several word similarity and word analogy tasks.

Word Similarity The word similarity computation is a traditional task for evaluating word embeddings. We used four datasets to evaluate word similarity including *WordSim353* (ws) (Finkelstein et al., 2001) partitioned into two datasets, *WordSim Similarity* (wss)and *WordSim Relatedness* (wsr) (Zesch et al., 2008; Agirre et al., 2009); Bruni et al.'s (2012) *MEN* dataset; Radinsky et al.'s (2011) *Mechanical Turk* (MTurk) dataset; the *TOEFL* dataset (Landauer and Dumais, 1997).

The first three datasets contain word pairs together with human-annotated similarity scores. The word embeddings are evaluated by ranking according to their cosine similarities and calculate the Spearman's rank correlation(Spearman's ρ) with the human-assigned scores. For the *TOEFL* set, we choose the nearest neighbor of the question word from the 4 candidates based on the cosine distance and use the accuracy to measure the performance.

Word Analogy The analogy dataset presents questions in the form of "a is to a^* as b is to b^*" in which b^* needs to be guessed from the entire vocabulary. Mikolov et al. (2013c) found that the learned word representations capture meaningful syntactic and semantic regularities referred to as *linguistic regularities*. We apply the Google's analogy dataset (Mikolov et al., 2013a) to evaluate word analogy performance. It contains 19,544 questions divided into two categories: 8,869 semantic questions and 10,675 syntactic questions. Such questions are answered through finding the word vector v_{b^*} which has the maximum cosine distance to the vector "$v_{a^\star} - v_a + v_b$" (Levy et al., 2014): $\arg\max_{b^\star \in V}(\cos(b^\star, b + a^\star - a)$. The

Results on the enwiki corpus (%)									
Method	similarity						analogy		
	ws	wss	wsr	MEN	MTurk	TOEFL	tot	sem	syn
CBOW	67.64	73.72	61.91	71.55	64.19	**80.00**	**68.41**	77.10	**61.20**
CBOW-CW	**69.00**	**73.77**	**65.06**	**72.84**	**67.17**	**80.00**	67.98	**79.03**	58.80
Results on the text8 corpus (%)									
Method	similarity						analogy		
	ws	wss	wsr	MEN	MTurk	TOEFL	tot	sem	syn
CBOW	69.02	70.22	66.40	64.06	60.35	70.00	30.86	30.10	31.40
CBOW-CW	**72.03**	**73.36**	**70.84**	**65.59**	**63.55**	**73.75**	**37.28**	**42.04**	**33.89**

Table 2: Performance of CBOW and CBOW-CW (our work) on different word similarity and word analogy tasks. The best result for each dataset is highlighted in **bold**.

evaluation metric for word analogy is the percentage of questions for which the $\arg\max$ result is the correct answer.

5.5 Results and Analysis

We compare the proposed CBOW-CW model with the original CBOW model and report the results using the settings in Table 1 for the text8 and enwiki corpus, respectively. Other settings of hyper-parameters can also obtain similar results.

Word Similarity The left of Table 2 shows the results for several word similarity tasks. The best result for each dataset is highlighted in **bold**. We found that in almost all of these datasets, except for the *TOEFL* dataset when training on enwiki corpus, the proposed model performs better than the original CBOW model. The results indicate that the method of appending different weights to different context words is effective and the heuristic for computing the "implicit weights" is helpful for building word embeddings. When training on the enwiki corpus, the CBOW-CW model can obtain similar results compared to the CBOW model on the *TOEFL* dataset. We suspect this maybe due to the fact that the *TOEFL* dataset is composed of low-frequency words and our model favors only high-frequency words since the Huffman softmax tree is built according to word frequency.

Word Analogy Table 2 also shows the results of the word analogy tasks. The CBOW-CW model does not perform well as it does on the word similarity tasks compared with the CBOW. When training on the large enwiki corpus, the proposed model performs better in terms of semantic accuracy but not so well in terms of syntactic accuracy. When training on the text8 corpus, CBOW-CW achieves pretty well improvement, especially on semantic accuracy where it increases nearly 12 percent compared with CBOW. Those results show that CBOW-CW is better in catching semantic information than syntactic information.

By comparing the results training on enwiki and text8, we interestingly find that our model provides a large accuracy gain over CBOW with small training data. This provides an advantage for small training corpora and indicates that our model has higher convergence rate than the original CBOW model. We are still exploring the reason of this, and we suspect that maybe due to that the method of computing weights is biased by the depth of the word in the Huffman tree. The shallow of the word in the Huffman tree, the higher accuracy of computing the weights.

Regarding the computation speed, the CBOW-CW model runs at a rate around a quarter of the rate of CBOW due to the additional computation for context weights. Even so, our model is capable of training word embeddings on the largest corpora in one day.

6 Conclusions

We proposed an extension to the CBOW model by adding implicit structure information. Our method absorbed the advantage of using a dependency tree-based context while avoiding the complexity of it. In future work, we will conduct more evaluations with other weighting schemes.

Acknowledgements

This work was funded in part by the National Science Foundation of China (grant 61472459).

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *ACL (1)*, pages 1370–1380. Citeseer.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A Smith. 2014. A dependency parser for tweets. In *In Proc. of EMNLP*. Citeseer.

Thomas K Landauer and Susan T Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *ACL (2)*, pages 302–308.

Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. 2014. Linguistic regularities in sparse and explicit word representations. In *CoNLL*, pages 171–180.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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

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

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Using wiktionary for computing semantic relatedness. In *AAAI*, volume 8, pages 861–866.

Word Embeddings and Convolutional Neural Network for Arabic Sentiment Classification

Abdelghani Dahou, Shengwu Xiong, Junwei Zhou*, Mohamed Houcine Haddoud and Pengfei Duan
Hubei Key Laboratory of Transportation Internet of Things,
School of Computer Science and Technology, Wuhan University of Technology
Wuhan 430070, China
dahou@whut.edu.cn xiongsw@whut.edu.cn junweizhou@msn.com
haddoud.medhoucine@gmail.com duanpf@whut.edu.cn

Abstract

With the development and the advancement of social networks, forums, blogs and online sales, a growing number of Arabs are expressing their opinions on the web. In this paper, a scheme of Arabic sentiment classification, which evaluates and detects the sentiment polarity from Arabic reviews and Arabic social media, is studied. We investigated in several architectures to build a quality neural word embeddings using a 3.4 billion words corpus from a collected 10 billion words web-crawled corpus. Moreover, a convolutional neural network trained on top of pre-trained Arabic word embeddings is used for sentiment classification to evaluate the quality of these word embeddings. The simulation results show that the proposed scheme outperforms the existed methods on 4 out of 5 balanced and unbalanced datasets.

1 Introduction

A growing number of people get used to give their opinions on social network websites, forums, video sharing websites, blogs and e-commerce websites, leading to a most rising research fields caused by the important opinionated web contents. The opinions could be used for many applications such as consumer modeling, sales prediction, opinion survey or user intent understanding. Sentiment analysis which is used to identify, extract and classify subjective information in the opinions, has attracted a lot of attention. Sentiment analysis can be divided into several levels: document level (Yessenalina et al., 2010), sentence level (Farra et al., 2010), word/term level (Engonopoulos et al., 2011) or aspect level (Chifu et al., 2015).

Currently, sentiment analysis is commonly used for English, while sentiment analysis on the Arabic language is still recognized at its early stages (Nabil et al., 2015; ElSahar and El-Beltagy, 2015), since sentiment analysis on Arabic is considered as a more challenging work. Firstly, Arabic has a very complex morphology and structure. Inflectional and derivational nature of Arabic language makes the monophonically analysis on Arabic more difficult (Hammo et al., 2002). Secondly, Arabic Internet users mostly use dialectal Arabic rather than Modern Standard Arabic (MSA), while MSA is the formal written language but dialectal Arabic is used in informal daily communication. Moreover, dialectal Arabic is not included in education systems or standardized (Habash, 2010). The diversity of different writings and the language cultural creates more challenges to learn and build language models for representations. Nowadays, more than 267 million people speak Arabic as the first language, and more than 250 million as the second language covering 58 countries[1]. There are around 168.1 million Arabic Internet users with a user growth rate of 6,592.5% (November 2015 by Internetworldstats[2]) , making research of sentiment analysis on the Arabic language important.

In this paper, we try to solve the problems of word embeddings and sentiment classification for Arabic text. We firstly use a web-crawler to build a 10 billion Arabic words corpus, and we realize an word embeddings model to produce Arabic word representations using this corpus. Finally, a convolutional

* Corresponding author. Tel.: +86 027 87298267.

[1]http://www.ethnologue.com/statistics/size.

[2]http://www.internetworldstats.com/stats7.htm

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2418–2427, Osaka, Japan, December 11-17 2016.

neural network (CNN) trained on top of pre-trained Arabic word embeddings is used for sentiment classification. The simulation results show that the proposed scheme has a better performance than existed approaches for the Arabic sentiment classification on different datasets. For the convenience of the evaluation of our scheme, we distribute freely the source code and the generated word embeddings on the web[3].

The rest of the paper is organized as follows. The related work will be introduced in Section 2. Section 3 refers to the construction of word embeddings model for Arabic, and Section 4 refers to the CNN model for sentiment classification of Arabic text. Experiments and analyses will be given in Section 5. The conclusion is drawn in Section 6.

2 Related Works

In this Section, we review existing works related to the proposed scheme. We start with the works on word embeddings, which represent individual words of a language as vectors onto a lower dimensional vector space. Then we introduce the works related to sentiment classification. These methods and techniques can be used to generate semantic representations of texts and perform classification for various tasks in NLP, which are the interesting and related works to our study.

Via neural language models, various deep learning methods have been proposed to learn word vector representations. WORD2VEC (Mikolov et al., 2013) has been proposed for building word representations in vector space, which consists of two models, including continuous bag of word (CBOW) and Skipgram (Skip-Gram). Global vectors for word representations are also used to build word representations (Pennington et al., 2014), where training is based on statistics of word to word co-occurrence from a corpus.

Recently, sentiment classification becomes one of the most motivating research area among natural language processing (NLP) community. Many tools and applications have been applied to Arabic sentiment classification. (Abdul-Mageed et al., 2011) reported efforts for classifying MSA news data at the sentence level for both subjectivity and sentiment using support vector machine (SVM) classifier. (Abdul-Mageed et al., 2014) presented an SVM-based system for subjectivity and sentiment analysis for Arabic social media named SAMAR. (Farra et al., 2010) proposed an Arabic sentence level classification based on syntactic and semantic approaches. (El-Halees, 2011) proposed a combined classification approach for document level sentiment classification using different classifiers, including lexicon based classifier, maximum entropy classifier and k-nearest neighbors (KNN) classifier. More recently, CNN have achieved remarkably strong performance tackling NLP tasks and gotten some interesting results (Kalchbrenner et al., 2014; Kim, 2014). (Kalchbrenner et al., 2014) introduced a dynamic CNN for modeling sentences, and (Kim, 2014) proposed an improved scheme which employs dynamic-updated and static word embeddings simultaneously for sentence classification based on CNN.

3 Arabic Word Embeddings

Machine learning offers significant benefits for representations of a word from text and understanding natural language. The WORD2VEC tool[4] (Mikolov et al., 2013) remains a popular choice benefited from its fast training and good results. CBOW and Skip-Gram in WORD2VEC use a probabilistic prediction approach which captures syntactic and semantic word relationships from very large data sets. In this work, we explore several architectures to build neural word embeddings for MSA and dialectal Arabic. In particular, we perform a comprehensive analysis to train and evaluate word representations using CBOW and Skip-Gram models, with the goal of generating a better quality representations for Arabic sentiment classification. Fig. 1 shows the steps from collecting the corpus to building word representations.

[3] http://pan.baidu.com/s/1eS2mxCe

[4] https://code.google.com/archive/p/word2vec/.

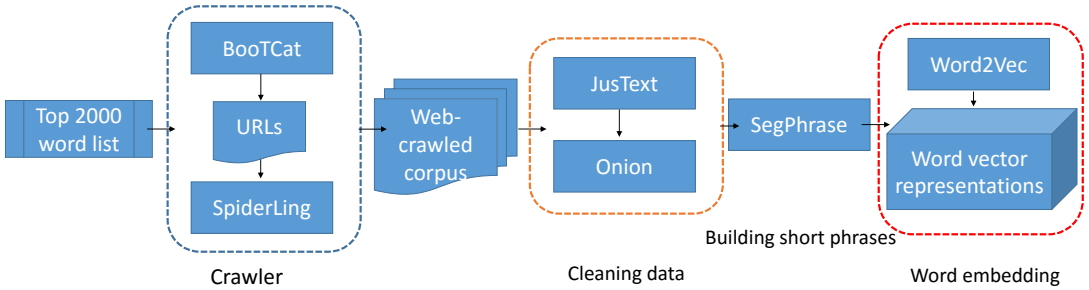


Figure 1: The processes of building Arabic word embeddings

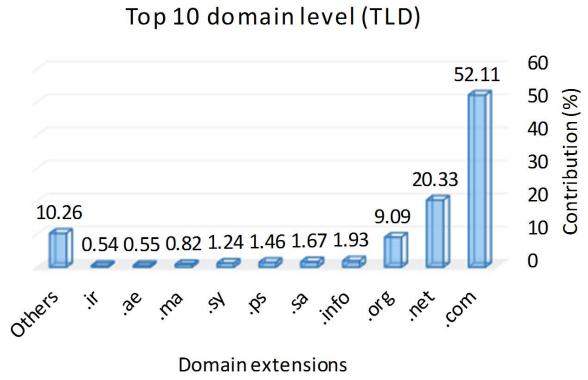


Figure 2: Top 10 Internet top-level domains (TLD).

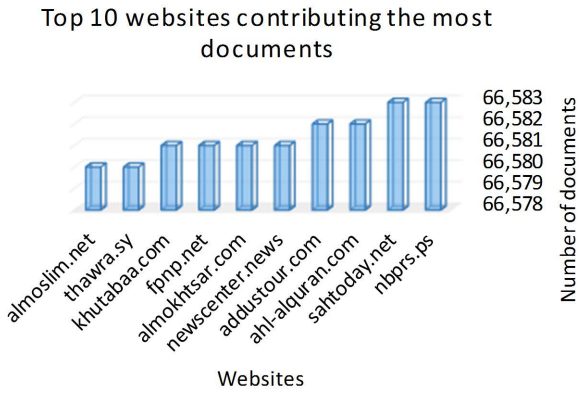


Figure 3: Top 10 websites exploited from the crawler seeding list which contribute the most documents.

3.1 Crawling and Preparing the Corpus

The existing corpora are rare freely accessible for download and are typically not large enough for CNN based Arabic sentiment classification. In order to generate Arabic word representations, we create a huge web-crawled corpus for MSA and dialectal Arabic text.

The acquisition of the corpus is performed by means of SpiderLing[5] (Suchomel et al., 2012), which is an open-source and free crawler for effective creation and annotation of linguistic corpora.

The steps for creating the corpus are described as follows:

1) The seeds for the crawler consist of 10,421 URLs, which are generated using Bing queries, after feeding BootCAT [6] with a word-list of top 2000 words based on their frequency. It should be noticed that the word-list has to be filtered from stop words. Then, the seeds are filtered from duplicated URLs, while the filtered seeds are fed to SpiderLing as a start point for Arabic web pages crawling. The top-10 level web domains used by SpiderLing to produce our corpus are shown in Fig. 2, while Fig. 3 presents the list of top-10 domains that contributed the most of documents.

2) For data normalization, we used jusText[7] (Pomikálek, 2011) which is a heuristic based boilerplate removal tool, which is used to exclude contents such as navigation links, advertisements and headers from downloaded web pages. It only keeps paragraphs containing full sentences and removes contents which are not in the desired language. Also, we used Onion[8] (Pomikálek, 2011) for near-duplicate detection and removing. The deduplication is performed on paragraph level and threshold is set to 0.5.

3.2 Pre-processing and Normalization

An intrinsic limitation of word representations for the Arabic language is that sometimes we need two words to represent one meaning. For example, the word “acceptable” and its antonym “unacceptable”.

[5]http://corpus.tools/browser/spiderling.
[6]http://bootcat.sslmit.unibo.it/.
[7]http://code.google.com/p/justext/.
[8]https://code.google.com/p/onion.

However, the antonym word has no direct (word-to-word) Arabic translation, غير and مقبول can't be easily combined to obtain "غير_مقبول" which is the closest translation to the antonym word. Motivated by this kind of limitation, we thus use SegPhrase[9] (Liu et al., 2015) to learn these phrases.

Based on a data-driven approach presented in (Mikolov et al., 2013) which relies on the count of unigram and bigram to form phrases, we run an experiment using the following equation:

$$score(w_i, w_j) = \frac{count(w_i w_j) - \delta}{count(w_i) \times count(w_j)}, \quad (1)$$

where δ is a discounting coefficient to prevent the formation of phrases with infrequent words. Bigrams with a score over the chosen threshold are then formed as phrases.

We use an Arabic text consisted of 1.3 billion words and a vocabulary[10] of 3.5 million words to form short phrases based on equation (1). From this experiment, we notice that the average amount of formed phrases occupy 47.42 % of the new vocabulary when we set the threshold to 100. These short phrases occupy 16.18 % from the total number of tokens in the Arabic text. We also notice that raw frequency methods could not reflect the quality of these phrases, since both good and bad phrases can possess high frequency.

To learn these semantical and meaningful phrases and to generate better word embeddings for Arabic, we choose SegPhrase rather than the previous approach. SegPhrase is a framework that aims to extract and mine quality phrases from a large text, combined with a module for phrase segmentation. Thus, text data is transformed from word granularity to phrase granularity. To use SegPhrase for Arabic text, we also need to build two knowledge bases, where the smaller one consists of 94,544 labels and contains high-quality phrases for positive labels. Moreover, the larger one contains 121,127 labels, which is used to filter medium and low-quality phrases for negative labels.

Some examples are سان فرانسيسكو (San Francisco), المملكه العربيه السعوديه (Kingdom of Saudi Arabia) and هواري بومدين (Houari Boumediene) labeled to be positive. In contrast, phrases like تسويق دولي (International marketing), تسلسل زمني (Chronology) and تحت المجهر (Under the microscope) were labeled as negative. A threshold of raw frequency with value 10 is specified for frequent phrase mining, which will generate a candidate set. Another parameter is used to parse our corpus using the generated model which is the ratio of top ranked phrases with value 0.5.

The processing steps for cleaning and normalizing the corpus are as follows : 1) Remove punctuation, diacritics, non letters , and non Arabic. 2) Normalization of the different writings of the latter (Alef) إ , أ , آ with ا. 3) Convert the letter (Teh Marbuta) ة to ه.

3.2.1 Training Parameters

By using Word2Vec tool, we build several models based on different architectures and word vector dimensionality. Table 1 shows the training parameters used for various models based on CBOW and Skip-Gram architectures.

Model	dimensionality	Window size	Sample	Negative	Freq. thresh.	Max iterations
CBOW	100,200,300	5	$1 \times e^{-5}$	10	10	3
Skip-Gram	100,200,300	10	$1 \times e^{-5}$	10	10	3

Table 1: Word vector representations training parameters

For the quality evaluation of the generated Arabic word embeddings, we use word analogy questions task. We compare the vectors in (Zahran et al., 2015) with our vectors.

[9]https://github.com/shangjingbo1226/SegPhrase.

[10]We did not set a frequency threshold because we want to use all the words even the rare words.

4 CNN for Sentiment Classification

In this Section, we tackle our downstream semantic task which is Arabic sentiment classification relying on the Arabic word embeddings model from the previous section. It is a binary classification task between positive and negative and covers two domains: reviews and tweets. Here, the standard 10-fold cross validation is applied to the balanced and unbalanced datasets to report accuracy on all different used datasets.

4.1 CNN Architecture

We consider a CNN architecture similar to that described in (Kim, 2014), with one channel that allows the adaptation of pre-trained vectors for each task. The hyper-parameters used for the training of all models[11] will be discussed in later paragraphs.

Due to the absence of a large supervised training set and as described in (Socher et al., 2011; Iyyer et al., 2014), the training word vectors are initialized by a pre-trained model, in our case the CBOW58[12] model. Where each vector has a dimensionality of $k = 300$ and is trained using CBOW architecture. An initialization of word vectors for all words which are out from CBOW58 model's vocabulary has been performed by sampling from a uniform distribution in the range of [-0.25,0.25] following (Kim, 2014) work.

Here, we define S as a sentence of n words, and let m_i be the ith word in the sentence S[13], where each $m_i \in S$ is represented by x_i a k-dimension vector such as $x_i \in \mathbb{R}^k$.

To perform convolution operation via linear filters over S, we convert S to a sentence matrix of shape $(n \times k)$. Where each row corresponds to a vector x_i [14]. Considering our filter $w \in \mathbb{R}^{hk}$ we set 3 different window widths (of h words) from $D \subset \mathbb{N}^*$, where $D = \{3, 4, 5\}$, with a fixed length k for each filter dimensionality. The application of a convolution operation using one filter window size $h \in D$ over the sentence matrix produces new features.

To capture most relevant global features, and deal with variable sentence lengths. We use a *max-over-time* pooling operation (Collobert et al., 2011) to downsample the feature maps.

After concatenating all feature maps in one single vector with a fixed length, we feed this vector through a fully-connected layer with $Dropout$ (Hinton et al., 2012). Here, the $Dropout$ rate is set to 0.5, and a sigmoid function to generate the final classification. A gradient based optimization named Adagrad (Duchi et al., 2011) with Mini-batch size of 32 is used. The output is the probability distribution over labels. Fig.4 provides a schematic illustrating of the model architecture.

5 Experiments and Analyses

In this section we explore the process of analyzing the quality and the impact of word embeddings using two major evaluations. We start by the intrinsic evaluation using word analogy questions task in Section 5.1. Secondly, we do an extrinsic evaluation by performing Arabic sentiment classification task after defining a CNN model trained on top of the generated word embeddings model in Section 5.2.

5.1 Vector Quality Evaluation

The goal of word analogy questions is to correctly identify the relationship between *C* and *D*, given a relationship between words *A* and *B*. The questions will be in the form of $A : B : C : D$, where the pairs of word (*A*, *B*) and (*C*, *D*) are sharing the same relation (e.g. "man:woman", "king:queen"). We hide the identity of the fourth word D and we predict it based on the other three words, using similarity measure functions like cosine similarity, or Euclidean distance. To find a word that is closest to *D* measured by cosine distance, an algebraic computation can be performed on word embeddings. Here, we simply compute vector $vector(D) = vector(C) - vector(A) + vector(B)$

[11] All experiments run with Keras and Theano on an NVIDIA GeForce GTX 780 Ti GPU.

[12] CBOW58 is the best model of Arabic word embeddings built in Section 3 and the results are listed in Section 5.1.

[13] We use the same zero-padding strategy as in (Kim, 2014).

[14] x_i refers to word vector extracted from $CBOW58$ model with $k = 300$.

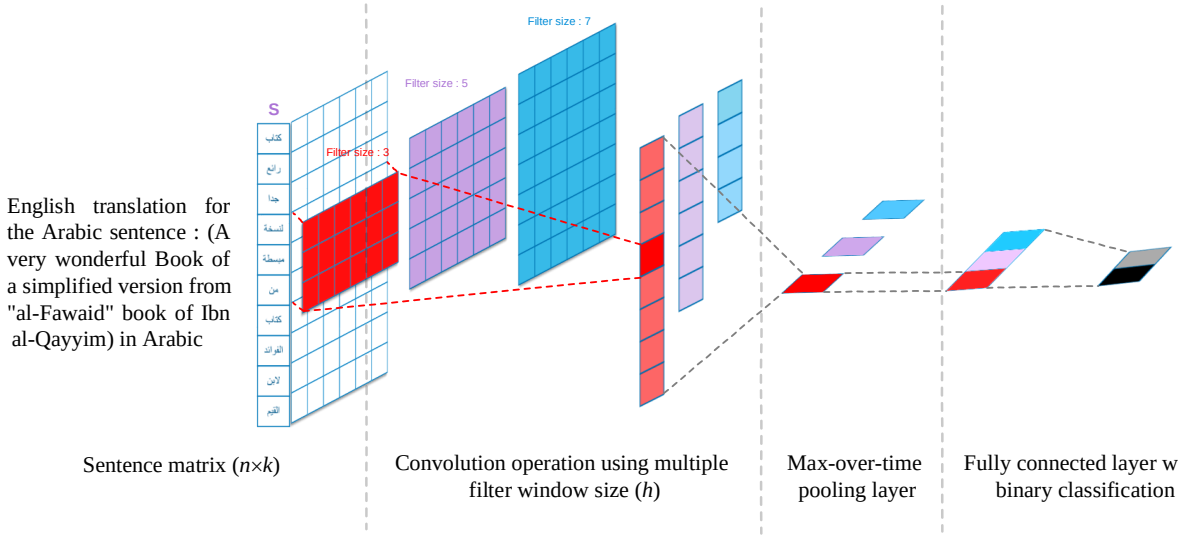


Figure 4: Model architecture for an Arabic example sentence.

To test the quality of the vectors, we used the test cases from (Zahran et al., 2015) with the correction of few Arabic spelling errors and the adding of new analogy questions for opposite words such as opposite 2-gram. The generated new test cases for Arabic contains 24,294 questions. Overall, there are 10,463 semantic and 13,831 syntactic questions. It covers 15 analogy questions, 5 types of semantic questions, and 10 types of syntactic related questions. Some examples are shown in Table 2. The cosine similarity measure we used is defined as:

$$\cos(u, v) = \frac{u \cdot v}{\|u\|\|v\|} \tag{2}$$

Analogy question	Word pair 1				Word pair 2			
Common-capital-countries	Rome	روما	Italy	ايطاليا	Bagdad	بغداد	Iraq	العراق
Oppisite 1-gram	Short	قصير	Tall	طويل	Sad	حزين	Happy	سعيد
Opposite 2-gram	Certain	مؤكد	Uncertain	غير مؤكد	Acceptable	مقبول	Unacceptable	غير مقبول

Table 2: Examples of word analogy questions test set

Equation (3) is used to calculate the cosine similarity of the analogy questions to predict the closest word. Where V is the used vocabulary ignoring the three question words *B*, *A* and *C*. An answer on one of the question analogy is counted correct only if one of the top five predicted words matches the fourth word *D*.

$$\arg\max_{D \in V}(\cos(D, C)\cos(D, A) + \cos(D, B)) \tag{3}$$

Here we use Equation (3) for the evaluation. The results for all different word vector models are shown in Table 3, where "Cov" is an abbreviation for coverage and "Acc" for accuracy, and we represent different dimension as $i-$d where i denotes the dimension size (100-D describe a dimensionality of 100).

In Section 3, we have shown the processing steps to build the corpus and the training parameters for our word embeddings. Collecting a big corpus for these tasks is a major step, but the most important is the evaluation part for the quality of the collected data and the pre-processing operation. The used data to build Arabic word embeddings models consist of 3.4 billion words and a vocabulary of 2.2 million words, In (Zahran et al., 2015), they used 5.8 billion words in their corpus with a vocabulary of 6.3 million words. More iterations improved the accuracy significantly during the training process of our models. Pre-processing the crawled corpus by mining quality phrases to avoid forming incorrect or low-quality phrases could help to enhance the quality of short phrases. As shown in opposite-2gram analogy

Model	(Zahran et al., 2015)				Our models											
Arch	CBOW		Skip-G		CBOW						Skip-Gram					
Dimention size	300-D		300-D		100-D		200-D		300-D		100-D		200-D		300-D	
Accuracy & Coverage	Acc	Cov	Acc	Cov	Acc	Cov	Acc	Cov	Acc	Cov	Acc	Cov	Acc	Cov	Acc	Cov
Capital-common-countries	**95.89**	100	94.16	100	85.93	100	90.26	100	90.91	100	85.50	100	90.26	100	92.21	100
Capital-world	69.81	100	69.38	100	68.81	98.16	74.51	98.16	**78.15**	98.16	66.20	98.16	74	98.16	75.92	98.16
Currency	14.95	98	13.54	98	16.37	96	**20.45**	96.00	17.86	96	9.65	96	10.25	96	9.48	96
City-in-state	19.04	96.88	22.09	96.88	29.58	96.88	39.29	96.88	**46.19**	96.88	25.52	96.88	38.58	96.88	41.63	96.88
Family	36.19	100	33.33	100	27.14	100	33.81	100	**39.05**	100	24.52	100	31.43	100	37.14	100
Adjective-toadverb	30.91	100	19.70	100	27.83	100	32.02	100	**34.36**	100	12.44	100	18.97	100	21.67	100
Opposite-1gram	**22.73**	100	13.64	100	11.82	100	12.73	100	17.27	100	13.64	100	17.27	100	20	100
Opposite-2gram	12.80	28.57	9.55	28.57	40.12	69.05	**46.17**	69.05	43.15	69.05	21.19	69.05	27	69.05	29.18	69.05
Comparative	**76.59**	100	68.02	100	59.44	100	66.98	100	69.84	100	40.40	100	55.48	100	56.59	100
Superlative	**72.25**	100	62.22	100	55.02	100	64.20	100	68.28	100	36.65	100	49.15	100	51.42	100
Present-tense	58.18	100	55.56	100	50.45	100	56.26	100	**60.45**	100	40.30	100	49.75	100	51.72	100
Nationality-adjective	**84.18**	100	84.12	100	79.66	98.33	81.76	98.33	83.02	98.33	78.71	98.33	83.22	98.33	83.68	98.33
Past-tense	**73.01**	100	70.13	100	57.18	100	67.56	100	71.41	100	47.95	100	59.94	100	62.37	100
Plural	53.90	100	47.84	100	42.64	100	46.97	100	**54.33**	100	35.71	100	36.80	100	37.45	100
Plural-verbs	95.56	100	94.86	100	94.86	100	96.07	100	**96.47**	100	86.39	100	88.51	100	88.41	100
TOTAL	54.40	94.90	50.54	94.90	49.79	97.23	46.97	97.23	**58.05**	97.23	41.65	97.23	48.71	97.23	50.59	97.23

Table 3: Total accuracy of (Zahran et al., 2015) word embeddings model and our Arabic word embeddings models on word analogy questions test set. All numbers in the table are percentage.

questions type of Table (3), the coverage rate is 69.05 % compared to 28.57 % in a trained model based on Equation (1). According to comparisons based on the word embeddings dimensionality, it could be found that using low dimensionality does not help much to improve the quality of these vectors when trained on a large corpus. Therefore, a major drawback in these neural based language models is that training on a higher dimensionality is time consuming.

These analogy evaluation scores for different models and architectures are not a great estimator of real-world tasks performance. The use of these word embeddings in a downstream task such as Arabic sentiment classification in our case will be determined by whether these vectors are of good quality or not. These scores provide us with a general idea of what our data looks like.

5.2 Arabic Sentiment Classification

We apply a binary sentiment classification for different corpora from two different domains: reviews and tweets. We run the experiments on the LABR book reviews dataset (Aly and Atiya, 2013) which consists of over 63,000 reviews downloaded from Goodreads[15] in 2013, Arabic Sentiment Tweets Dataset (ASTD) (Nabil et al., 2015) which consists of over 10,000 Arabic tweets, Arabic Gold-Standard Twitter Sentiment Corpus (Refaee and Rieser, 2014) (GS-dataset) collected in 2014, Twitter data set for Arabic sentiment analysis collected by (Abdulla et al., 2013) that consists of 2000 labeled tweets, and also datasets collected by (ElSahar and El-Beltagy, 2015) that covers five domains:

1) *Hotel Reviews (HTL)*: For the hotels domain 15K Arabic reviews are scrapped from TripAdvisor[16].
2) *Attraction Reviews (ATT)*: Attraction reviews are scrapped from TripAdvisor.
3) *Restaurant Reviews (RES)*: Two sources are scrapped to cover restaurants reviews: Qaym[17] and TripAdvisor.
4) *Movie Reviews (MOV)*: collected from 1k movies in Elcinemas.com website, and consists of around 1.5K movies reviews.
5) *Product Reviews (PROD)*: For the Products domain, a dataset of 15K reviews is scraped from the Souq[18] website. The dataset includes reviews from Egypt, Saudi Arabia, and the United Arab Emirates.

We pre-processed all datasets by extracting positive and negative sentences, and splitting them to two main classes. The first one is the balanced class where the numbers of reviews and tweets are equal,

[15] www.goodreads.com.
[16] www.tripadvisor.com.
[17] www.Qaym.com
[18] www.souq.com

and the number is set to the minimum number of positive or negative sentences. The second one is the unbalanced class, where the numbers of reviews and tweets are unequal, and the numbers are set to the maximum numbers of positive and negative sentences, respectively. Datasets preparation statistics are shown in Table 4.

Datasets	Polarity	LABR	ASTD	Gold-Standard	Twitter Data set	ATT	HTL	MOV	PROD	RES
Balanced	Positive	8012	665	858	972	204	6192	384	807	9092
	Negative	8012	665	858	972	204	6192	384	807	9092
Unbalanced	Positive	42689	665	858	978	5242	27428	969	2882	25858
	Negative	8012	1496	1897	972	204	6192	384	807	9092

Table 4: Dataset preparation statistics

Models	LABR	ASTD	Gold-Standard	Twitter Data set	ATT	HTL	MOV	PROD	RES
CNN-balanced	**86.7**	75.9	73.8	86.3	74.2	**88.6**	**83.2**	**83.3**	77.1
CNN-unbalanced	**89.6**	79.07	75.8	85.01	96.2	**91.7**	**80.7**	**87.3**	78.5
(ElSahar and El-Beltagy, 2015)-Linear SVM	78.3	-	-	-	-	87.6	74.3	75.8	83.6
(Abdulla et al., 2013)-SVM	-	-	-	87.2	-	-	-	-	-
(Refaee and Rieser, 2014b)-SVM-BOW	-	-	87.74	-	-	-	-	-	-

Table 5: Comparison of existing methods with our CNN models on the same datasets. SVM (Abdulla et al., 2013), SVM+BOW SVM and Bag of Word from (Refaee and Rieser, 2014), Linear SVM (ElSahar and El-Beltagy, 2015).

Concerning Section 4 and Section 5.2 which evaluates the ability of a CNN based model to perform Arabic sentiment analysis. Table 5 presents results of our CNN models against other methods listing their best classification experimental results. Each cell has numbers that represent the accuracy of the evaluation performed on each dataset. There is a huge gap between different datasets in their sizes. A bigger dataset has better performance in terms of model accuracy. LABR dataset reaches 89.6 % when it has been trained in unbalanced form. The existence of sarcastic and dialectal Arabic (reviews/tweets) really could have a severe impact on the model accuracy. One problem is that there is not that much consecutive semantic/syntactic content in a single tweet. These datasets are examined in their balanced and unbalanced form. The results show that a CNN architecture with one non-static channel and one convolutional layer outperform the listed techniques on a scale 4 of 5 when using balanced datasets. By using all the data in an unbalanced form for the training and validation, the model is more accurately, showing a significant good performance compared to other models.

Additionally, the accuracy of the proposed model for Gold-Standard dataset is lower than that listed in (Refaee and Rieser, 2014), the reason is the used data set in our work is smaller than the original one, and the full dataset of (Refaee and Rieser, 2014) is not available for free. The CNN model provides a remarkable accuracy improvement over the Linear SVM approach used by (ElSahar and El-Beltagy, 2015), which is the previous best-reported result for their collected datasets. Initializing word vectors using our pre-trained vectors (CBOW58) gives a significant accuracy increase. Random initialization of words out of CBOW58 vocabulary can affect the vectors quality during the training. Although, exploring the effect of hyper-parameters for the CNN model still to be investigated in a detailed way, to observe the impact of these parameters on the network performance tackling different tasks.

6 Conclusions

In this study, we have introduced a crawling scheme for a large multi-domain corpus in order to build an Arabic word embeddings model. We have provided short practical and empirically informed procedures for exploring Arabic word embeddings and convolutional neural network (CNN) for sentiment classification.

The experiment results of building Arabic word embeddings from a web-crawled corpus illustrate that the performance increases along with the quality of the data. The results also show that high dimensionality vectors perform well on a large corpus. The results of CNN for sentiment datasets show that

initializing word vectors using a pre-trained word embeddings gain a remarkable performance. They also indicate that a bigger dataset usually has better performance regarding model accuracy.

Acknowledgments

The work described in this paper was supported in part by the National High-tech R&D Program of China (Grant No. 2015AA015403), by the National Natural Science Foundation of China (Grant No. 61601337), by the Fundamental Research Funds for the Central Universities (Grant Nos. 2015III015-B04, 2015IVA034 and 2016III011), by Nature Science Foundation of Hubei Province (Grant No. 2015CFA059), by Science & Technology Pillar Program of Hubei Province (Grant No. 2014BAA146), by Science and Technology Open Cooperation Program of Hannan Province (Grant No. 152106000048).

References

Muhammad Abdul-Mageed, Mona T Diab, and Mohammed Korayem. 2011. Subjectivity and sentiment analysis of modern standard arabic. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 587–591.

Muhammad Abdul-Mageed, Mona Diab, and Sandra Kübler. 2014. Samar: Subjectivity and sentiment analysis for arabic social media. *Computer Speech & Language*, 28(1):20–37.

Nawaf A Abdulla, Nizar A Ahmed, Mohammad A Shehab, and Mahmoud Al-Ayyoub. 2013. Arabic sentiment analysis: Lexicon-based and corpus-based. In *2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, pages 1–6.

Mohamed A Aly and Amir F Atiya. 2013. Labr: A large scale arabic book reviews dataset. In *ACL (2)*, pages 494–498.

Emil St. Chifu, Tiberiu St. Letia, and Viorica R. Chifu. 2015. Unsupervised aspect level sentiment analysis using self-organizing maps. In *2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 468–475.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Alaa El-Halees. 2011. Arabic opinion mining using combined classification approach. pages 1–8.

Hady ElSahar and Samhaa R El-Beltagy. 2015. Building large arabic multi-domain resources for sentiment analysis. In *Computational Linguistics and Intelligent Text Processing*, pages 23–34.

Nikos Engonopoulos, Angeliki Lazaridou, Georgios Paliouras, and Konstantinos Chandrinos. 2011. Els: a word-level method for entity-level sentiment analysis. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, pages 1–9.

Noura Farra, Elie Challita, Rawad Abou Assi, and Hazem Hajj. 2010. Sentence-level and document-level sentiment mining for arabic texts. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 1114–1119.

Nizar Y Habash. 2010. Introduction to arabic natural language processing. *Synthesis Lectures on Human Language Technologies*, 3(1):1–187.

Bassam Hammo, Hani Abu-Salem, and Steven Lytinen. 2002. Qarab: A question answering system to support the arabic language. In *Proceedings of the ACL-02 workshop on Computational approaches to semitic languages*, pages 1–11.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, pages 1–18.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Association for Computational Linguistics*, pages 1113–1122.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, pages 1–6.

Jialu Liu, Jingbo Shang, Chi Wang, Xiang Ren, and Jiawei Han. 2015. Mining quality phrases from massive text corpora. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1729–1744.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mahmoud Nabil, Mohamed Aly, and Amir F Atiya. 2015. Astd: Arabic sentiment tweets dataset. In *EMNLP*, pages 2515–2519.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Jan Pomikálek. 2011. Removing boilerplate and duplicate content from web corpora. *Disertacnı práce, Masarykova univerzita, Fakulta informatiky*.

Eshrag Refaee and Verena Rieser. 2014. An arabic twitter corpus for subjectivity and sentiment analysis. In *LREC*, pages 2268–2273.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, pages 151–161.

Vít Suchomel, Jan Pomikálek, et al. 2012. Efficient web crawling for large text corpora. In *Proceedings of the seventh Web as Corpus Workshop (WAC7)*, pages 39–43.

Ainur Yessenalina, Yisong Yue, and Claire Cardie. 2010. Multi-level structured models for document-level sentiment classification. In *EMNLP*, pages 1046–1056.

Mohamed A Zahran, Ahmed Magooda, Ashraf Y Mahgoub, Hazem Raafat, Mohsen Rashwan, and Amir Atyia. 2015. Word representations in vector space and their applications for arabic. In *Computational Linguistics and Intelligent Text Processing*, pages 430–443.

Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts

Xingyou Wang[1], **Weijie Jiang**[2], **Zhiyong Luo**[3],
[1]Beijing Language and Culture University, Beijing, China {ultimate010@gmail.com}
[2]Tsinghua University, Beijing, China {jwj14@mails.tsinghua.edu.cn}
[3]Beijing Language and Culture University, Beijing, China {luo_zy@blcu.edu.cn}

Abstract

Sentiment analysis of short texts is challenging because of the limited contextual information they usually contain. In recent years, deep learning models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have been applied to text sentiment analysis with comparatively remarkable results. In this paper, we describe a jointed CNN and RNN architecture, taking advantage of the coarse-grained local features generated by CNN and long-distance dependencies learned via RNN for sentiment analysis of short texts. Experimental results show an obvious improvement upon the state-of-the-art on three benchmark corpora, MR, SST1 and SST2, with 82.28%, 51.50% and 89.95% accuracy, respectively. [1]

1 Introduction

The rapid development of the Internet, e-commerce and social networks brings about a large amount of user-generated short texts on the Internet, such as online reviews for products, services and blogs. Such short texts as online reviews are usually subjective and semantic oriented. To discriminate and classify the semantic orientation of such short texts properly is of great research and practical value.

Sentiment analysis of short texts is challenging because of the limited contextual information and the sparse semantic information they normally contain. The existing research on sentiment analysis of short texts basically include emotional knowledge-based methods and feature-based classification methods. The former mainly focuses on the extraction and the sentiment classification based on opinion-bearing words and opinion sentences (Hu and Liu, 2004; Kim and Hovy, 2005). The latter focuses on the sentiment classification based on features. Turney (2002) presented the unsupervised PMI-IR (Pointwise Mutual Information and Information Retrieval) algorithm to measure the similarity of words or phrases. Pang et al. (2002) and Cui et al. (2006) used n-grams and POS tags and applied them to NB, ME and SVM classifiers. Based on these studies, Kim and Hovy (2006) introduced the positional features and opinion-bearing word features. Mullen and Collier (2004) combined various features and used SVM for classification.

With the development of deep learning, typical deep learning models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have achieved remarkable results in computer vision and speech recognition. Word embeddings, CNNs and RNNs have been applied to text sentiment analysis and gotten remarkable results. Kim (2014) applied CNN on top of pre-trained word vectors for sentence-level classification. Some studies utilized recursive neural networks to construct the sentence-level representation vector in sentiment analysis (Socher et al., 2011; Socher et al., 2012b; Socher et al., 2013b). Le and Mikolov (2014) presented the paragraph vector in sentiment analysis. Tai et al. (2015) put forward the tree-structured long short-term memory (LSTM) networks to improve the semantic representations. The improved performance of these algorithms mainly benefits from the following aspects: (1) The high-dimensional distributional vectors endow similar semantic-oriented words with high similarity. Word embeddings can better solve the semantic sparsity of short texts compared with the one-hot representation. (2) Similar to the translation, rotation and scale invariance of images in CNN, CNN is able to learn the local features from words or phrases in different places of texts. (3) RNN takes words in

[1]Code and data are available at `https://github.com/ultimate010/crnn`

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2428–2437, Osaka, Japan, December 11-17 2016.

a sentence in a sequential order and is able to learn the long-term dependencies of texts rather than local features.

In this paper, we present a jointed CNN and RNN architecture that takes the local features extracted by CNN as input to RNN for sentiment analysis of short texts. We develop an end-to-end and bottom-up algorithm to effectively model sentence representation. We take the word embeddings as the input of our CNN model in which windows of different length and various weight matrices are applied to generate a number of feature maps. After convolution and pooling operations, the encoded feature maps are taken as the input to the RNN model. The long-term dependencies learned by RNN can be viewed as the sentence-level representation. The sentence-level representation is taken to the fully connected network and the softmax output reveals the classification result. The deep learning algorithm we put forward differs from the existing methods in that: (1) We apply windows of different length and various weight matrices in convolutional operation. The max pooling operates on the adjacent features and moves from left to right instead of on the entire sentence. In this case, the feature maps generated in our CNN model retain the sequential information in the sentence context. (2) The deep learning architecture of our model takes advantage of the encoded local features extracted from the CNN model and the long-term dependencies captured by the RNN model. We experiment on three benchmarks for sentiment classification, MR, SST1 and SST2 and achieve the state-of-the-art results.

This work is organized as follows. In section 2, we discuss some background knowledge about word embeddings, sentence-level representation, convolutional neural network and recurrent neural network. In section 3, we describe our jointed CNN model and RNN model in detail. Section 4 presents our experiment results and some discussion. Finally, in section 5, we conclude and remark on our work.

2 Background

In this section, we discuss some background knowledge on word embeddings, sentence-level representation, convolutional neural network (CNN) and recurrent neural network (RNN).

2.1 Word Embeddings and Sentence-Level Representation

When applying deep learning methods to a text classification task, we normally need to transform words into high-dimensional distributional vectors that capture morphological, syntactic and semantic information about the words. Let d be the length of word embeddings and l be the length of a sentence. The sentence-level representation is encoded by an embedding matrix $C \in \mathbb{R}^{d \times l}$, where $C_i \in \mathbb{R}^d$ corresponds to the word embeddings of the i-th word in the sentence.

2.2 Convolution and Pooling

Convolution is widely used in sentence modeling (Kim, 2014; Kalchbrenner et al., 2014; dos Santos and Gatti, 2014). The structure of convolution varies slightly in different research fields, among which the structure used in natural language processing is shown in Figure 1.

Generally, let l and d be the length of sentence and word vector, respectively. Let $C \in \mathbb{R}^{d \times l}$ be the sentence matrix. A convolution operation involves a convolutional kernel $H \in \mathbb{R}^{d \times w}$ which is applied to a window of w words to produce a new feature. For instance, a feature c_i is generated from a window of words $C[*, i : i + w]$ by

$$c_i = \sigma(\sum(C[*, i : i + w] \circ H) + b) \quad (1)$$

Here $b \in \mathbb{R}$ is a bias term and σ is a non-linear function, normally tanh or ReLu. $\circ$ is the Hadamard product between two matrices. The convolutional kernel is applied to each possible window of words in the sentence to produce a feature map. $c = [c_1, c_2, \ldots, c_{l-w+1}]$, with $c \in \mathbb{R}^{l-w+1}$.

Next, we apply pairwise max pooling operation over the feature map to capture the most important feature. The pooling operation can be considered as feature selection in natural language processing. The max pooling operation is shown in Figure 2.

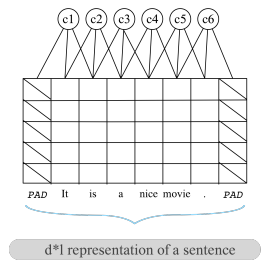


Figure 1: (1) A sentence matrix with padding. (2) Convolution with a window of three words.

Specifically, the output of convolution, the feature map $c = [c_1, c_2, \ldots, c_{l-w+1}]$ is the input of the pooling operation. Let the input be downscaled by 2, namely, the adjacent two features in the feature map be caculated as follows:

$$p_i = \max(c_{2\times i-1}, c_{2\times i}) \tag{2}$$

The output of the max pooling operation is $p = [p_1, p_2, \ldots, p_{\lfloor \frac{l-w+1}{2} \rfloor}]$, $p \in \mathbb{R}^{\lfloor \frac{l-w+1}{2} \rfloor}$.

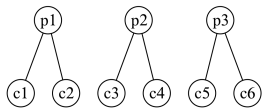


Figure 2: Pairwise max pooling operation on a scaling of 2

2.3 Recurrent neural network

Recurrent Neural Networks (RNNs) have shown great promise in machine translation tasks(Liu et al., 2014; Sutskever et al., 2014; Auli et al., 2013). Unlike feedforward neural networks, RNNs are able to handle a variable-length sequence input by having a recurrent hidden state whose activation at each time is dependent on that of the previous time. Figure 3 shows what a typical RNN looks like.

The diagram shows a RNN being unrolled into a full network. For example, if the input sequence is a four-word sentence, the network would be unrolled into a 4-layer neural network, one layer for each word. The formulas that govern the calculations in a RNN are as follows.

- x_t is the input at time step t. For example, x_i could be a one-hot vector or word embeddings corresponding to the i-th word of a sentence.
- h_t is the hidden state and the "memory" of the network at time step t. h_t is calculated according to the previous hidden state and the input at the current step:

$$h_t = \sigma(Ux_t + Wh_{t-1}) \tag{3}$$

  Here h_0 is typically initialized to a zero vector in order to calculate the first hidden state.
- o_t is the output at time step t. For sentiment classification of short texts, it would be a vector of probabilities across all the sentiment categories. o_t is calculated as follows:

$$o_t = softmax(Vh_t) \tag{4}$$

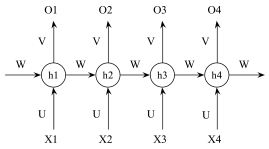


Figure 3: An unrolled recurrent neural network

Similar to a traditional neural network, we can use a twisted backpropagation algorithm Backpropagation Through Time (BPTT) to train a RNN (Mozer, 1989). Unfortunately, it is difficult to train RNN to capture long-term dependencies because the gradients tend to either vanish or explode (Bengio et al., 1994). Hochreiter and Schmidhuber (1997) proposed a long short-term memory (LSTM) unit and Cho et al. (2014) proposed a gated recurrent unit (GRU) to deal with the problem effectively.

2.4 LSTM and GRU

2.4.1 LSTM

The Long Short-Term Memory (LSTM) was first proposed by Hochreiter and Schmidhuber (1997) that can learn long-term dependencies. See Figure 4 for the graphical illustration.

Different from traditional recurrent unit, LSTM unit keeps the existing memory $c_t \in \mathbb{R}^n$ at time t. The input at time t is x_t, h_{t-1}, c_{t-1}, the output is h_t, c_t, they can be updated by the following equaions:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{5}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{6}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{7}$$
$$g_t = tanh(W_g x_t + U_g h_{t-1} + b_g) \tag{8}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{9}$$
$$h_t = o_t \odot tanh(c_t) \tag{10}$$

where $\sigma(\cdot)$ denotes the logistic sigmoid fuction. The opration $\odot$ denotes the element-wise vector product. At each time step t, there are an input gate i_t, a forget gate f_t, an output gate o_t, a memory cell c_t and a hidden unit h_t. h_0 and c_0 can be initialized to 0 and the parameters of the LSTM is W, U, b.

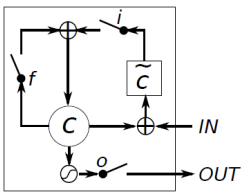


Figure 4: LSTM unit (Chung et al., 2014)

2.4.2 GRU

A gated recurrent unit (GRU) was initially proposed by Cho et al. (2014) to make each recurrent unit to adaptively capture dependencies of different time scales. See Figure 5 for the graphical illustration of GRU.

The parameters can be updated by the following equations

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{11}$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{12}$$

$$\hat{h}_t = \tanh(W x_t + U(r_t \odot h_{t-1})) \tag{13}$$

$$h_t = (1 - z_t) h_{t-1} + z_t \hat{h}_t \tag{14}$$

Where σ denotes the logistic sigmoid function, $\odot$ denotes the element-wise multiplication, r_t denotes the reset gate, z_t denotes the update gate and $\hat{h}_t$ denotes the candidate hidden layer.

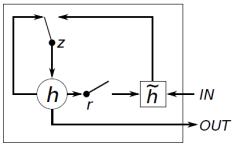


Figure 5: GRU unit (Chung et al., 2014)

3 Model

Convolutional neural networks (CNNs) are likely to extract local and deep features from natural language. It has been shown that CNN has gotten improved results in sentence classification (Kim, 2014). Recurrent neural networks (RNNs) are various kinds of time-recursive neural network that is able to learn the long-term dependencies in sequential data. Seeing that we can view the words in a sentence as a sequence from left to right, RNNs can be modeled in accordance with people's reading and understanding behavior of a sentence. Socher et al. (2012a) presented a convolutional-recursive deep model for 3D object classification that combined the convolutional and recursive neural networks together. The CNN layer learns the low-level translation invariant features which are inputs to multiple, fixed-tree RNNs (recursive neural networks) in order to compose higher order features. Kim et al. (2015) described a model that employed a convolutional neural network (CNN) and a highway network over characters, whose output is given to a long short-term memory (LSTM) recurrent neural network language model (RNN-LM). These two models both get better results than prior methods. However, the recursive neural networks need to build a tree structure that is usually based on the parser result of sentence. The recurrent neural network is particularly suited for modeling the sequential pattern. Inspired by those works and the fact that CNN can extract local features of input and RNN (recurrent neural network) can process sequence input and learn the long-term dependencies, we combine both of them in sentiment analysis of short texts. The model architecture is shown in Figure 6.

Our model consists of the following parts: word embeddings and sentence-level representation, convolutional and pooling layers, concatenation layer, RNN layer, fully connected layer with softmax output.

3.1 Word Embeddings and Sentence-Level Representation

Word embeddings play an important role in word representation. The commonly used are random initialization and unsupervised pre-training of word embeddings. In our experiment, we perform unsupervised learning of word-level embeddings using the $word2vec$ method and also test random initialization. Let ν be the size of bag-of-words and d be the length of a word embedding, then the word embeddings of all the words in the vocabulary are encoded by column vectors in an embedding matrix $Q \in \mathbb{R}^{d \times \nu}$. A sentence can be represented by:

$$S = [w_1, w_2, \ldots, w_l] \tag{15}$$

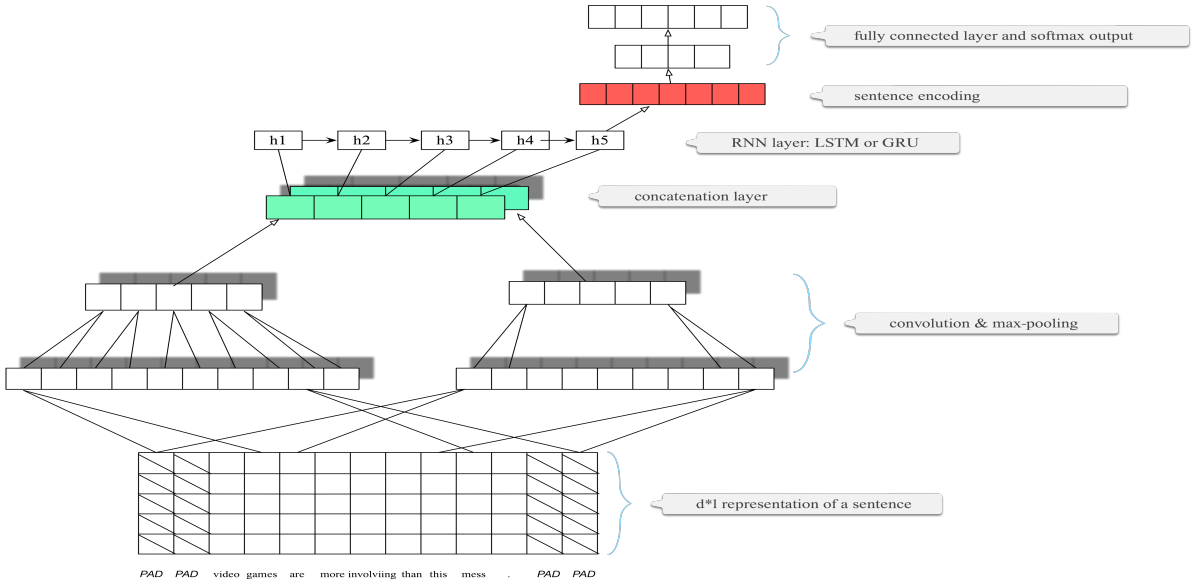


Figure 6: Model architecture for an example sentence

$$w_i \in [1, \nu], i \in [1, l]$$

Here is the sentence-level representation:

$$C_i = Q[w_i], C_i \in \mathbb{R}^d \tag{16}$$
$$C = [C_1, C_2, \ldots, C_l], C \in \mathbb{R}^{d \times l} \tag{17}$$

The column vector C_i corresponds to the word embedding of the i-th word in the sentence.

3.2 Convolution and Pooling

The convolutional layer applies a matrix-vector operation to each window of size w of successive windows in the sentence-level representation sequence. Let $H \in \mathbb{R}^{d \times w}$ be the weight matrix of the convolutional layer, we then add a bias item b to the result of the matrix-vector operation and get a feature mapping $c \in \mathbb{R}^{l-w+1}$. The i-th element of the feature map is:

$$c_i = \sigma(\sum(C[*, i : i + w] \circ H) + b) \tag{18}$$

where $C[*, i : i + w]$ is the i-th to the $i + w$-th column vectors of the sentence-level representation.

The same weight matrix is used to extract local features for each window of the given sentence. Using the matrix over all word windows of the sentence, we extract the n-grams feature vector of size $l - w + 1$. We also apply various kinds of weight matrices and multiple filter lengths to get various and sufficient features.

We apply the max and average pooling operations and find that the former performs better with less computational complexity. Thus, we apply the max pooling operation to the output of convolutional layer which transform the feature map of size $l - w + 1$ to $\lfloor \frac{l-w+1}{2} \rfloor$,

$$p = [p_1, p_2, \ldots, p_{\lfloor \frac{l-w+1}{2} \rfloor}] \tag{19}$$

We apply m kinds of matrix weight to get m feature maps.

$$P = [p^1, p^2, \ldots, p^m] \tag{20}$$

where $P \in \mathbb{R}^{\lfloor \frac{l-w+1}{2} \rfloor \times m}$.

We borrow the experience from the literature (Zhang and Wallace, 2015; Kim, 2014) and choose window size 4 and 5 to get matrix P_4, P_5. After trunking longer result, we concatenate P_4 and P_5 together to get Z, where

$$Z = \oplus(P_4, P_5), Z \in \mathbb{R}^{\lfloor \frac{l-w+1}{2} \rfloor \times (m \times 2)} \tag{21}$$

where $\oplus$ is the concatenation operator, Z can be viewed as the convolutional coding of a sentence.

3.3 Recurrent Neural Network

The features generated from convolution and pooling operation can be viewed as advanced features like n-grams. Since recurrent neural network (RNN) can process sequential input and learn the long-term dependencies, we take these features as the input of the recurrent neural network. We apply LSTM and GRU that are mentioned in previous chapter and both get good results. The output of RNN $T \in \mathbb{R}^n$ is deemed as the encoding of the whole sentence.

3.4 Fully Connected Network with Softmax Output

The features generated from RNN form the penultimate layer and are passed to a fully connected softmax layer whose output is the probability distribution over all the categories. The softmax operation over the scores of all the categories is calculated as follows:

$$\hat{P}_i = \frac{exp(o_i)}{\sum_{j=1}^{C} exp(o_j)} \tag{22}$$

We take cross entropy as the loss function that measures the discrepancy between the real sentiment distribution $\hat{P}^t(C)$ and the model output distribution $\hat{P}(C)$ of sentences in the corpora.

$$loss = -\sum_{s \in T} \sum_{i=1}^{V} \hat{P}_i^t(C) log(\hat{P}_i(C)) \tag{23}$$

Here T is the training corpora, V is the number of the sentiment categories. $\hat{P}^t(C)$ is the V-dimension one-hot coding vector where the elements corresponding to the sentence's real sentiment category is 1 and other elements 0. The entire model is trained end-to-end with stochastic gradient descent.

4 Experimental Setup and Results

We conduct experiments to empirically evaluate our method by applying it to three benchmarks as follows.

- MR: Movie reviews with one sentence per review. Classification involves detecting positive/negative reviews (Pang and Lee, 2005).[2]
- SST1: Stanford Sentiment Treebank - an extension of MR but provided five kinds of labels, very negative, negative, neutral, positive and very positive (Socher et al., 2013a). [3]
- SST2: Same as SST1 but with neutral reviews removed and binary labels.

The experiment runs on Tesla K40c GPU. Summary statistics of the datasets are in Table 1.

[2] https://www.cs.cornell.edu/people/pabo/movie-review-data/
[3] http://nlp.stanford.edu/sentiment/

Data	c	l	$\lvert V_{train}\rvert$	$\lvert V_{val}\rvert$	$\lvert V_{test}\rvert$
MR	2	20	8655	961	1046
SST1	5	18	151525	0	2200
SST2	2	19	76836	0	1811

Table 1: Summary statistics for the datasets after tokenization. c: Number of target classes. l: Average sentence length. $|V_{train}|$: Training set size. $|V_{val}|$: Validation set size. $|V_{test}|$: Test set size. The training set of SST1 and SST2 includes phrases extracted from sentences and sentences themselves, and test set only includes sentences.

4.1 Model Variations

We experiment with several variants of the model.

- CNN-GRU-word2vec: A model with pre-trained vectors from word2vec, max pooling and GRU recurrent unit.
- CNN-LSTM-word2vec: A model with pre-trained vectors from word2vec, max pooling and LSTM recurrent unit.
- AGV-GRU-word2vec: A model with pre-trained vectors from word2vec, average pooling and GRU recurrent unit.
- CNN-GRU-rand: A model with randomly initialized vectors, max pooling and GRU recurrent unit.
- CNN-LSTM-rand: A model with randomly initialized vectors, max pooling and LSTM recurrent unit.

4.2 Results and Dicussion

Results of our models against other methods are listed in tabel 2. Specially, our models with pre-trained vectors from word2vec[4] and max pooling perform best among all the models, of which the one with the GRU recurrent unit performs better on MR and SST2 while the one with LSTM performs better on SST1. The classification accuracy is raised by 0.7% on MR and 1.8% on SST2, when implementing CNN-GRU-word2vec model compared with the existing models. At the same time, the CNN-LSTM-word2vec model raises the classification accuracy by 0.1%. Furthermore, LSTM reveals good performance on SST1 while GRU performs better on MR and SST2.

In the meantime, we find that our models with pre-trained vectors all perform better than the others with randomly initialized vectors on all three corpora. Thus, we infer that the pre-trained vectors on large-scale corpora can solve the semantic sparsity problem to some degree.

Compared with the existing methods and experiment results, we find that our jointed architecture of CNN and RNN model performs better than the CNN and RNN models alone in sentiment classification of short texts. We take advantage of both the CNN model and the RNN model thus get higher classification accuracy than the existing models. CNN extracts the local features of input and RNN processes sequence input while learning the long-term dependencies and get sentence-level feature representation. The experiments substantiate the validity of our idea.

5 Conclusion

In this work we present a deep neural network architecture that takes advantage of the construction of convolutional neural network (CNN) and recurrent neural network (RNN) and joint them together for sentimental analysis of short texts. In particular, our pooling operation on adjacent words is able to retain the local features and their sequential relations in a sentence. Besides, RNN can learn the long-term dependencies and the positional relation of features as well as the global features of the whole sentence.

[4]Pre-trained vectors `https://code.google.com/archive/p/word2vec/`

Group	Model	MR	SST1	SST2
Other	NB(Socher et al., 2013b)	–	41.0	81.8
	SVM(Socher et al., 2013b)	–	40.7	79.4
CNN	1-layer convolution(Kalchbrenner et al., 2014)	–	37.4	77.1
	Deep CNN(Kalchbrenner et al., 2014)	–	48.5	86.8
	Non-static(Kim, 2014)	81.5	48.0	87.2
	Multichannel(Kim, 2014)	81.1	47.4	88.1
Recursive	Basic(Socher et al., 2013b)	–	43.2	82.4
	Matrix-vector (Socher et al., 2013b)	–	44.4	82.9
	Tensor (Socher et al., 2013b)	–	45.7	85.4
	Tree LSTM1 (Zhu et al., 2015)	–	48.0	-
	Tree LSTM2 (Tai et al., 2015)	–	51.0	88.0
	Tree LSTM3 (Le and Zuidema, 2015)	–	49.9	88.0
	Tree bi-LSTM (Li et al., 2015)	0.79	–	–
Recurrent	LSTM(Tai et al., 2015)	–	46.4	84.9
	bi-LSTM(Tai et al., 2015)	–	49.1	87.5
Vector	Word vector avg(Socher et al., 2013b)	–	32.7	80.1
	Paragraph vector(Le and Mikolov, 2014)	–	48.7	87.8
TBCNNs	c-TBCNN(Mou et al., 2015)	–	50.4	86.8
	d-TBCNN(Mou et al., 2015)	–	51.4	87.9
CNN-RNN	CNN-GRU-word2vec	**82.28**	50.68	**89.95**
	CNN-LSTM-word2vec	81.52	**51.50**	89.56
	AVG-GRU-word2vec	81.44	50.36	89.61
	CNN-GRU-rand	76.34	48.27	86.64
	CNN-LSTM-rand	77.04	49.50	86.80

Table 2: Results of our jointed architecture of CNN and RNN against other methods.

Our models perform well on three benchmark datasets and achieve higher classification accuracy than the existing models.

Our jointed neural network architecture can be applied to sentence modeling as well as other natural language processing tasks. For future work, we will extend our models to long texts classification tasks.

Acknowledgments

We would like to thank Zhiyong Luo for his meticulous guidance and Jinsong Zhang for providing servers for us to run our code. We would also like to thank Ju Lin for his insightful comments.

References

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *EMNLP*, volume 3, page 0.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Hang Cui, Vibhu Mittal, and Mayur Datar. 2006. Comparative experiments on sentiment classification for online product reviews. In *AAAI*, volume 6, pages 1265–1270.

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78.

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Soo-Min Kim and Eduard Hovy. 2005. Automatic detection of opinion bearing words and sentences. In *Companion Volume to the Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, pages 61–66.

Soo-Min Kim and Eduard Hovy. 2006. Automatic identification of pro and con reasons in online reviews. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 483–490. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. *arXiv preprint arXiv:1503.02510*.

Jiwei Li, Dan Jurafsky, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*.

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. 2014. A recursive recurrent neural network for statistical machine translation. In *ACL (1)*, pages 1491–1500.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. *Unpublished manuscript: http://arxiv. org/abs/1504. 01106v5. Version*, 5.

Michael C Mozer. 1989. A focused back-propagation algorithm for temporal pattern recognition. *Complex systems*, 3(4):349–381.

Tony Mullen and Nigel Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In *EMNLP*, volume 4, pages 412–418.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Y Ng. 2012a. Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems*, pages 665–673.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012b. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. 2013a. Parsing With Compositional Vector Grammars. In *EMNLP*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over tree structures. *arXiv preprint arXiv:1503.04881*.

Stance Classification in Rumours as a Sequential Task Exploiting the Tree Structure of Social Media Conversations

Arkaitz Zubiaga[1], **Elena Kochkina**[1], **Maria Liakata**[1], **Rob Procter**[1], **Michal Lukasik**[2]
[1] University of Warwick, Coventry, UK
[2] University of Sheffield, Sheffield, UK
{a.zubiaga,e.kochkina,m.liakata,rob.procter}@warwick.ac.uk
m.lukasik@sheffield.ac.uk

Abstract

Rumour stance classification, the task that determines if each tweet in a collection discussing a rumour is supporting, denying, questioning or simply commenting on the rumour, has been attracting substantial interest. Here we introduce a novel approach that makes use of the sequence of transitions observed in tree-structured conversation threads in Twitter. The conversation threads are formed by harvesting users' replies to one another, which results in a nested tree-like structure. Previous work addressing the stance classification task has treated each tweet as a separate unit. Here we analyse tweets by virtue of their position in a sequence and test two sequential classifiers, Linear-Chain CRF and Tree CRF, each of which makes different assumptions about the conversational structure. We experiment with eight Twitter datasets, collected during breaking news, and show that exploiting the sequential structure of Twitter conversations achieves significant improvements over the non-sequential methods. Our work is the first to model Twitter conversations as a tree structure in this manner, introducing a novel way of tackling NLP tasks on Twitter conversations.

1 Introduction

Rumour stance classification is a task that is increasingly gaining popularity in its application to tweets. While Twitter is a generous source of reports of breaking news, outpacing even news outlets (Kwak et al., 2010), it also comes with the caveat that some of those reports are still rumours at the time of posting and so are yet to be verified and corroborated (Mendoza et al., 2010; Procter et al., 2013b; Procter et al., 2013a). The rumour stance classification task intends to assist in this verification process by determining the type of support expressed in different tweets discussing the same rumour (Qazvinian et al., 2011). Aggregation of the stance of multiple tweets discussing a rumour can then be of help to determine its likely veracity, enabling – among other benefits – the flagging of highly disputed rumours that are likely to be false.

Previous research on rumour stance classification for tweets has been limited to the tweet as the unit to be classified. However, such approaches ignore the additional context and knowledge that can be gained from the structure of Twitter interactions within conversational threads (Zubiaga et al., 2016; Procter et al., 2013b; Tolmie et al., 2015). The latter are formed as Twitter users reply to one another's posts and ultimately users build on each others' stance towards the rumour, leading to a potential consensus. For example, a tweet may report a rumour (source tweet) and others may reply to it by further supporting it or providing counter-evidence. Our objective here is to mine the sequence of stance types encountered in conversational threads collected from Twitter. The ultimate goal could be to aggregate such views to help determine the veracity of a rumour, which we hypothesise our task could be helpful for.

In order to make use of the sequence of stance types, we analyse conversations arising from tweets posted by users who are replying to one another. These replies result in tree-structured conversations, often nested, where replies are triggered by a source tweet that initiated the conversation. We make the following contributions:

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2438–2448, Osaka, Japan, December 11-17 2016.

- We hypothesise that making use of the sequential structure of conversational threads can improve stance classification in relation to a classifier that determines a tweet's stance from the tweet alone. To the best of our knowledge, the structure of Twitter conversations has not been studied before for classifying each of the underlying tweets and our work is the first to evaluate it for stance classification.
- We introduce a novel way of analysing tweets by mining the context from conversational threads. To do this, we propose two different models for capturing the sequential structure of conversational threads, viewing them as a) separate linear branches and b) as a tree structure.
- We evaluate the effectiveness of two flavours of Conditional Random Fields (CRF) in addressing stance classification on rumourous Twitter conversations. We compare the performance of these two CRF settings with other non-sequential baselines, including the non-sequential equivalent of CRF, a Maximum Entropy classifier. Our results show that while there is no significant difference when performance is measured based on micro-averaged F1 score (equivalent to accuracy and influenced by the majority class), sequential approaches do perform substantially better in terms of macro-averaged F1 score, proving that exploiting the conversational structure improves the classification performance.
- We also show that the use of tree CRF leads to an improvement over the linear-chain CRF, suggesting that in stance classification for conversational threads it is important to consider the whole tree structure rather individual linear branches. Our results advocate the merit of further exploring the use of sequential approaches to exploit conversational structures mined from Twitter posts for a wider range of NLP tasks.

2 Related Work

Following early work by Qazvinian et al. (2011) introducing the task of rumour stance classification for tweets, interest in this problem has increased substantially. However, the line of research initiated by Qazvinian et al. (2011) is significantly different to the one tackled in this paper. They perform 2-way classification of each tweet as *supporting* or *denying* a long-standing rumour, such as disputed beliefs that *Barack Obama is reportedly Muslim*. The authors use tweets observed in the past to train a classifier, which is then applied to new tweets discussing the same rumour. In recent work, rule-based methods have been put forward as a way to improve on the performance of the Qazvinian et al. (2011) baseline. This is the approach followed by Liu et al. (2015), who introduced a simple rule-based method that looks for the presence of positive or negative words in a tweet. One draw back of such rule-based approaches is that they may not generalise to new, unseen rumours. Similarly, Hamidian and Diab (2016) have recently studied the extent to which a model trained from historical tweets can be used for classifying new tweets discussing the same rumour. While Zhao et al. (2015) did not study stance classification, they showed that tweets that trigger questioning responses from others are likely to report disputed rumours, which reinforces the motivation of our work of determining the stance of tweets to then deal with rumours.

Classification of stance towards a target on Twitter has been addressed in SemEval-2016 task 6 (Mohammad et al., 2016). Task A had to determine the stance of tweets towards five targets as 'favor', 'against' or 'none'. Task B tested stance detection towards an unlabelled target, which required a weakly supervised or unsupervised approach. The dataset of this competition was not related to rumours or breaking news, it only considered a 3-way classification and did not provide any relations between tweets, which were treated as individual instances.

Our work presents different objectives in three aspects. First, we aim to classify the stance of tweets towards rumours that emerge while breaking news unfold; these rumours are unlikely to have been observed before, and hence rumours from previously observed events, which are likely to diverge, need to be leveraged for training. As far as we know, only Lukasik et al. (2015; 2016a; 2016b) have tackled stance classification in the context of breaking news applied to new rumours. Lukasik et al. (2015; 2016a) used Gaussian Processes to perform 3-way stance classification into supporting, denying or questioning, while comments where not considered as part of the task. Lukasik et al. (2016b) did include comments to perform 4-way stance classification; they used Hawkes Processes to exploit the temporal sequence

of stances towards rumours to classify new tweets discussing rumours. Work by Zeng et al. (Zeng et al., 2016) has also performed stance classification for rumours around breaking news, but overlapping rumours were used for training and testing.

Second, recent research has posited that a 4-way classification is needed to capture responses seen in the unfolding of breaking news (Procter et al., 2013b; Zubiaga et al., 2016). Moving away from the 2-way classification above, which is somewhat limited for our purposes, we adopt this expanded scheme including tweets that are *supporting*, *denying*, *querying* or *commenting* rumours. This adds two more categories to the scheme used in early work, where tweets would only support or deny a rumour. Moreover, our approach takes into account the interaction between users on social media, whether it is about appealing for more information in order to corroborate a rumourous post (*querying*) or to say something that does not contribute to the resolution of the rumour's veracity (*commenting*). Finally, instead of dealing with tweets as single units in isolation, we exploit the conversational structure of Twitter replies, building a classifier that learns the dynamics of stance in tree-structured conversational threads. The closest work when it comes to exploiting conversational structure in tweets is that of Ritter et al. (2010) who modelled linear sequences of replies in Twitter conversations with Hidden Markov Models for dialogue act tagging, but the structure of the tree as a whole was not exploited.

As far as we know, no work has leveraged the conversational structure of Twitter postings for stance classification, and hence its utility remains unexplored. A work that is related is that of Lukasik et al. (2016b), who exploited the temporal sequence of tweets, although the conversational structure was ignored and each tweet was treated as a separate unit. In other domains where debates or conversations are involved, the sequence of responses has been exploited to make the most of the evolving discourse and perform an improved classification of each individual post after learning the structure and dynamics of the conversation as a whole. For instance, Qu and Liu (2011) found Hidden Markov Models to be an effective approach to classify threads in on-line fora as successfully solving or not the question raised in the initial post. This was later further studied in a SemEval shared task, where each post in a forum thread had to also be classified as good, potential or bad (Màrquez et al., 2015). FitzGerald et al. (2011) used a linear-chain CRF to identify high-quality comments in threads responding to blog posts.

In a task that is related to stance classification, researchers have also studied the identification of agreement and disagreement in on-line conversations. To classify agreement between question-answer (Q-A) message pairs in fora, Abbott et al. (2011) used Naive Bayes as the classifier, and Rosenthal and McKeown (2015) used a logistic regression classifier. However, in both cases only pairs of messages were considered, and the entire sequence of responses in the tree was not used. CRF has also been used to detect agreement and disagreement between speakers in broadcast debates (Wang et al., 2011), which our task differs from in that it solely focuses on text. It is also worthwhile to emphasise that stance classification is different to agreement/disagreement detection, given that in stance classification one has to determine the orientation of a user towards a rumour. Instead, in agreement/disagreement detection, one has to determine if a pair of posts share the same view. In stance classification, one might agree with another user who is denying a rumour, and hence they are denying the rumour as well, irrespective of the pairwise agreement. To the best of our knowledge Twitter conversational thread structure has not been explored in the stance classification problem.

3 Stance classification using the conversational structure of Twitter threads

The rumour stance classification task consists in determining the type of support that each individual post expresses towards the disputed veracity of a rumour. The task is especially interesting in the context of Twitter, where unverified reports about breaking news are continually being posted and discussed as they unfold. This problem was originally tackled as a 2-way classification task, where each tweet was classified as supporting or denying a rumour. However, recent research (Procter et al., 2013b) found this categorisation to be insufficient to encompass all the different kinds of reactions to rumours, and a broader, 4-way classification task has been suggested instead. The argument behind this is that users in social media will not necessarily express a clear inclination towards supporting or denying a rumour, but can also be skeptical by posing questions about it or can make comments about the rumour that are

unrelated to its disputed veracity. The four categories in the extended classification scheme thus include *supporting*, *denying*, *querying* and *commenting*. In this work we set out the rumour stance classification that adopts this broader scheme. We define the rumour stance classification task as follows: we assume we have a set D of rumours R_i, each of which is composed of a collection of rumourous conversational threads. For simplicity here we refer to a rumour as the aggregate of its Twitter conversational threads, which is ultimately a collection of tweets. Each rumour has a variably sized set of tweets t_i discussing it so that $R_i = \{t_1, ..., t_{|R_i|}\}$; the task consists in determining the stance of each of the tweets t_j pertaining to a new, unseen rumour R_i as one of $Y = \{supporting, denying, querying, commenting\}$.

Moreover, within this task we propose leveraging conversation structure as one of the main features that characterise social media (Tolmie et al., 2015). So the task becomes one of classifying each tweet in a conversational thread, in the context of the thread. Twitter conversations consist of replies to each other, together forming a tree structure, as shown in the example in Figure 1. Replies can be nested in each other, so that the depth of the tree can vary. Hence, in the stance classification task applied to Twitter conversations we have rumours containing a variably sized set of conversations $R_i = \{C_1, ..., C_{|R_i|}\}$. Each of these conversations, C_j, has a varying number of tweets in it. By definition, a conversation has a source tweet (the root of the tree), $t_{j,1}$, that initiates it. The source tweet $t_{j,1}$ can receive replies by a varying number k of tweets $Replies_{t_{j,1}} = \{t_{j,1,1}, ..., t_{j,1,k}\}$, each of which can in turn receive replies by a varying number k of tweets, e.g., $Replies_{t_{j,1,1}} = \{t_{j,1,1,1}, ..., t_{j,1,1,k}\}$. Thus, we encode the tweet index as a sequence of ids of consecutive children of a preceding node, while traversing the conversation structure.

4 Dataset

We use the PHEME rumour dataset associated with eight events corresponding to breaking news stories (Zubiaga et al., 2016), which provide tweet-level annotations for stance[1]. Tweets in this dataset include tree-structured conversations, which are initiated by a tweet about a rumour (source tweet) and nested replies that further discuss the rumour circulated by the source tweet (replying tweets). Details on how the annotation was conducted through crowdsourcing can be found in Zubiaga et al. (2015).

The annotation scheme employed by the authors differs slightly from the one we need for our purposes, so we adapt it to our needs as follows. The source tweet of a conversation is originally annotated as *supporting* or *denying*, and each subsequent tweet is annotated as *agreed*, *disagreed*, *appeal for more information* (*querying*) or *commenting* as a pairwise annotation with respect to the source tweet. Instead, the labels needed for our task are *supporting*, *denying*, *querying* and *commenting*. To convert the labels, we keep the labels as *supporting* or *denying* in the case of source tweets. For the reply tweets, we keep their label as is for the tweets that are *querying* or *commenting*. To convert those tweets that agree or disagree into *supporting* or *denying*, we apply the following set of rules: (1) if a tweet agrees with a supporting source tweet, we label it *supporting*, (2) if a tweet agrees with a denying source tweet, we label it *denying*, (3) if a tweet disagrees with a supporting source tweet, we label it *denying* and (4) if a tweet disagrees with a denying tweet, we label it *supporting*. The latter enables us to infer stance with respect to the overarching rumour rather than refer to agreement with respect to the source. The resulting dataset includes 4,519 tweets, and the transformations of annotations described above only affect 24 tweets (0.53%), i.e., those where the source tweet denies a rumour, which is rare. The example in Figure 1 shows a rumour thread taken from the dataset along with our inferred annotations, as well as how we establish the depth value of each tweet in the thread.

One notable characteristic of the dataset is that the distribution of categories is skewed towards *commenting* tweets, and that this imbalance varies slightly across the eight events in the dataset (see Table 1). Given that we consider each event as a separate fold that is left out for testing, this varying imbalance makes the task more realistic and challenging. The striking imbalance towards *commenting* tweets is also indicative of the increased difficulty with respect to previous work on stance classification. Most of which performed binary classification of tweets as either supporting or denying, which as shown in our

[1]While the dataset includes data for nine events, here we use the eight events whose tweets are in English, excluding the ninth with tweets in German.

[depth=0] **u1:** These are not timid colours; soldiers back guarding Tomb of Unknown Soldier after today's shooting #StandforCanada –PICTURE– **[support]**

[depth=1] **u2:** @u1 Apparently a hoax. Best to take Tweet down. **[deny]**

[depth=1] **u3:** @u1 This photo was taken this morning, before the shooting. **[deny]**

[depth=1] **u4:** @u1 I don't believe there are soldiers guarding this area right now. **[deny]**

[depth=2] **u5:** @u4 wondered as well. I've reached out to someone who would know just to confirm that. Hopefully get response soon. **[comment]**

[depth=3] **u4:** @u5 ok, thanks. **[comment]**

Figure 1: Example of a tree-structured thread discussing the veracity of a rumour, where the label associated with each tweet is the target of the rumour stance classification task.

experiments only account for less than 28% of the tweets.

Event	Supporting	Denying	Querying	Commenting	Total
charliehebdo	239 (22.0%)	58 (5.0%)	53 (4.0%)	721 (67.0%)	1,071
ebola-essien	6 (17.0%)	6 (17.0%)	1 (2.0%)	21 (61.0%)	34
ferguson	176 (16.0%)	91 (8.0%)	99 (9.0%)	718 (66.0%)	1,084
germanwings-crash	69 (24.0%)	11 (3.0%)	28 (9.0%)	173 (61.0%)	281
ottawashooting	161 (20.0%)	76 (9.0%)	63 (8.0%)	477 (61.0%)	777
prince-toronto	21 (20.0%)	7 (6.0%)	11 (10.0%)	64 (62.0%)	103
putinmissing	18 (29.0%)	6 (9.0%)	5 (8.0%)	33 (53.0%)	62
sydneysiege	220 (19.0%)	89 (8.0%)	98 (8.0%)	700 (63.0%)	1,107
Total	910 (20.1%)	344 (7.6%)	358 (7.9%)	2,907 (64.3%)	4,519

Table 1: Distribution of categories for the eight events in the dataset.

5 Experiment Design

In this section we describe the classifiers, features and evaluation measures we used in our experiments.

5.1 Classifiers

Conditional Random Fields (CRF). We use CRF as a structured classifier to model sequences observed in Twitter conversations. With CRF, we can model the conversation as a graph that will be treated as a sequence of stances, which also enables us to assess the utility of harnessing the conversational structure for stance classification. In contrast to traditionally used classifiers for this task, which choose a label for each input unit (e.g. a tweet), CRF also consider the neighbours of each unit, learning the probabilities of transitions of label pairs to be followed by each other. The input for CRF is a graph $G = (V, E)$, where in our case each of the vertices V is a tweet, and the edges E are relations of tweets replying to each other. Hence, having a data sequence X as input, CRF outputs a sequence of labels Y (Lafferty et al., 2001), where the output of each element y_i will not only depend on its features, but also on the probabilities of other labels surrounding it. The generalisable conditional distribution of CRF is shown in Equation 1 (Sutton and McCallum, 2011).

$$p(y|x) = \frac{1}{Z(x)} \prod_{a=1}^{A} \Psi_a(y_a, x_a) \tag{1}$$

where Z(x) is the normalisation constant, and Ψ_a is the set of factors in the graph G.

We use CRFs in two different settings.[2] First, we use a linear-chain CRFs (Linear CRF) to model each branch as a sequence to be input to the classifier. We also use Tree-Structured CRFs (Tree CRF) or General CRFs to model the whole, tree-structured conversation as the sequence input to the classifier. So in the first case the sequence unit is a branch and our input is a collection of branches and in the second case our sequence unit is an entire conversation, and our input is a collection of trees. An example of the distinction of dealing with branches or trees is shown in Figure 2. With this distinction we also want to experiment whether it is worthwhile building the whole tree as a more complex graph, given that users replying in one branch might not have necessarily seen and be conditioned by tweets in other branches. However, we believe that the tendency of types of replies observed in a branch might also be indicative of the distribution of types of replies in other branches, and hence useful to boost the performance of the classifier when using the tree as a whole. An important caveat of modelling a tree in branches is also that there is a need to repeat parts of the tree across branches, e.g., the source tweet will repeatedly occur as the first tweet in every branch extracted from a tree.[3]

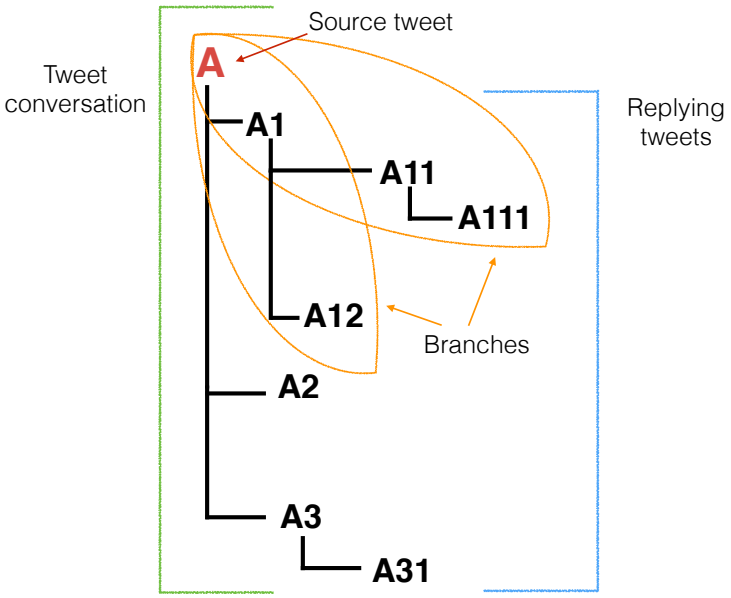


Figure 2: Example of a tree-structured conversation, with two overlapping branches highlighted.

Maximum Entropy classifier (MaxEnt). As the non-sequential equivalent of CRF, we use a Maximum Entropy (or logistic regression) classifier, which is also a conditional classifier but which operate at the tweet level, ignoring the conversational strucsture. This enables us to compare directly the extent to which treating conversations as sequences instead of having each tweet as a separate unit can boost the performance of the classifier.

Additional baselines. We also compare four more non-sequential classifiers[4]: Naive Bayes (NB), Support Vector Machines (SVM), Random Forests (RF), and Majority (i.e., a dummy classifier always labelling the most frequent class).

We experiment in an 8-fold cross-validation setting. Seven events are used for training and the remainder event is used for testing. With this, we simulate a realistic scenario where we need to use knowledge from past events to train a model that will be used to classify tweets in new events. For evaluation purposes, we aggregate the output of all eight runs as the micro-averaged evaluation across runs.

[2]We use the PyStruct to implement both variants of CRF (Müller and Behnke, 2014).

[3]Despite this also leading to having tweets repeated across branches in the test set and hence producing an output repeatedly for the same tweet with Linear CRF, this output is consistent and there is no need to aggregate different outputs.

[4]We use their implementation in the scikit-learn Python package

5.2 Features

We use four types of features to represent the tweets. Note that all of them are local features extracted from the tweet itself and independent of the rest of the conversation, hence enabling us to focus our comparison on how using the sequential structure can impact on the results.

Feature type #1: Lexicon.

- *Word Embeddings:* a vector with 300 dimensions averaging vector representations of the words in the tweet using Word2Vec (Mikolov et al., 2013). The Word2Vec model for each of the eight folds is trained from the collection of tweets pertaining to the seven events in the training set, so that the event (and the vocabulary) in the test set is unknown.
- *Part of speech (POS) tags:* a vector where each feature represents the number of occurrences of a type of POS tag in the tweet. The vector is then composed of the numbers of occurrences of different POS tags in the tweet, parsed using Twitie (Bontcheva et al., 2013).
- *Use of negation:* binary feature determining if a tweet has a negation word or not. We use a list of negation words, including: not, no, nobody, nothing, none, never, neither, nor, nowhere, hardly, scarcely, barely, don't, isn't, wasn't, shouldn't, wouldn't, couldn't, doesn't.
- *Use of swear words:* binary feature determining if 'bad' words are present in a tweet. We use a list of 458 bad words[5].

Feature type #2: Content formatting.

- *Tweet length:* the length of the tweet in number of characters.
- *Capital ratio:* the ratio of capital letters among all alphabetic characters in the tweet.
- *Word count:* the number of words in the tweet, counted as the number of space-separated tokens.

Feature type #3: Punctuation.

- *Use of question mark:* binary feature for the presence or not of question marks in the tweet.
- *Use of exclamation mark:* binary feature for the presence or not of exclamation marks in the tweet.
- *Use of period:* binary feature for the presence or not of periods in the tweet.

Feature type #4: Tweet formatting.

- *Attachment of URL:* binary feature, capturing the use or not of URLs in the tweet.
- *Attachment of picture:* binary feature that determines if the tweet has a picture attached.
- *Is source tweet:* binary feature determining if the tweet is a source tweet or is instead replying to someone else. Note that this feature can also be extracted from the tweet itself, checking if the tweet content begins with a Twitter user handle or not; there is no need to make use of the conversational structure to extract this feature.

5.3 Evaluation Measures

Given that the classes are clearly imbalanced in our case, evaluation solely based on accuracy can arguably suffice to capture competitive performance beyond the majority class. To account for the imbalance of the categories, we use both micro-averaged and macro-averaged F1 scores. Note that the micro-averaged F1 score is equivalent to the accuracy measure, while the macro-averaged F1 score complements it by measuring performance assigning the same weight to each category.

6 Results

Table 2 shows the results comparing performance of the different classifiers, both in terms of micro- and macro-F1 scores, and F1 scores by class. Due to the fact that the dataset is clearly imbalanced with a skew towards *commenting* tweets, we observe that even the majority classifier performs very well in terms of micro-averaged F1 score. In fact, the majority classifier is only slightly outperformed by other classifiers if we look at this evaluation measure. This is why we argue for an evaluation based on macro-averaged F1 score, which accounts for the ability of classifiers to produce an output that better fits to the distribution of classes. Interestingly, we observe that the conditional classifiers (i.e., MaxEnt, Linear CRF and Tree CRF) perform substantially better than the rest in terms of macro-averaged F1 score, which are the only ones to achieve a score of at least 0.4. Comparison of macro-averaged F1

[5] http://urbanoalvarez.es/blog/2008/04/04/bad-words-list/

scores of these three classifiers shows that the Tree CRF slightly outperforms the Linear CRF, while both perform significantly better than the non-sequential Maximum Entropy classifier. These results therefore do suggest that exploiting the sequential structure of conversations can lead to improvements on stance classification in rumourous Twitter conversations using the same set of local features.

When we look at the performance by class, we can observe that classifiers performing well only in terms of micro-averaged F1 have the tendency to perform well for the majority class (comments). Interestingly, CRF classifiers using conversational structure show remarkable improvements for other classes, especially supporting and querying tweets, where Tree CRF performs the best. However, all classifiers struggle to classify denials, with performance scores comparable to the other categories. We believe that one of the main reasons for this is that denials are one of the minority classes in the dataset. While querying tweets are also rare, some of the features like question marks are highly indicative of a tweet being a query, and hence they are easier to classify. Denials may in turn have significant commonalities with comments, given that the latter may also use negating words which may seem like denials. As shown in the confusion matrix for the Tree CRF in Table 3, the majority class *commenting* is being chosen in as many as 75.8% of the cases by the classifier for those tweets that are actually denials. Collection of additional denying tweets may be of help to improve performance in this class.

Classifier	Micro-F1	Macro-F1	S	D	Q	C
Majority	0.643	0.196	0.000	0.000	0.000	0.783
SVM	**0.676**	0.292	0.372	0.000	0.000	**0.796**
Random Forest	0.666	0.357	0.360	0.022	0.260	0.787
Naive Bayes	0.175	0.203	0.435	**0.147**	0.169	0.060
MaxEnt	0.666	0.400	0.352	0.062	0.396	0.789
Linear CRF	0.646	0.433	0.454	0.105	0.405	0.767
Tree CRF	0.655	**0.440**	**0.462**	0.088	**0.435**	0.773

Table 2: Micro- and Macro-F1 performance results, and F1 scores by class (S: supporting, D: denying, Q: querying, C: commenting)

	S	D	Q	C
S	**366 (40.4%)**	32 (3.5%)	22 (2.4%)	487 (53.7%)
D	38 (11.1%)	**22 (6.4%)**	23 (6.7%)	260 (75.8%)
Q	11 (3.1%)	10 (2.8%)	**149 (41.6%)**	188 (52.5%)
C	261 (9.0%)	91 (3.1%)	133 (4.6%)	**2,421 (83.3%)**

Table 3: Confusion matrix for Tree CRF (S: supporting, D: denying, Q: querying, C: commenting).

For comparison with the state-of-the-art stance classification approach by Lukasik et al. (2016b), we present results broken down by event in Table 4, both for their approach based on Hawkes Processes as well as our Tree CRF approach. Note that Lukasik et al. (2016b) only tested their approach on four of the events, and therefore performance scores for the rest of the events are not shown. As can be observed from the four events for which we have comparable results, the Hawkes Process performs better in terms of micro-F1, and therefore accurately classifying more instances. However, the Tree CRF performs substantially better in terms of macro-F1, which shows Tree CRF's ability to better estimate the distribution of labels in what is a highly imbalanced task and hence favouring the use of conversational structure in the classification process. We deem this a strong factor in this case as even a simple majority classifier achieves high micro-F1 scores, and the challenge lies in boosting macro-F1 scores to better balance the classification.

To better understand the effect of exploiting sequential structure, we break down performance scores by the depth of tweets. By this we want to see if the sequential classifiers are consistently performing

	Tree CRF		HP (Lukasik et al., 2016b)	
Event	Micro-F1	Macro-F1	Micro-F1	Macro-F1
ottawashooting	0.629	**0.457**	**0.678**	0.323
ferguson	0.559	**0.390**	**0.684**	0.260
charliehebdo	0.686	**0.427**	**0.729**	0.326
sydneysiege	0.677	**0.495**	**0.686**	0.325
germanwings-crash	0.694	0.523	–	–
putinmissing	0.660	0.446	–	–
prince-toronto	0.670	0.518	–	–
ebola-essien	0.629	0.384	–	–

Table 4: Micro- and Macro-F1 performance results broken down by event, along with a comparison with the results obtained by Lukasik et al. (2016b)'s state-of-the-art approach based on Hawkes Processes, where available.

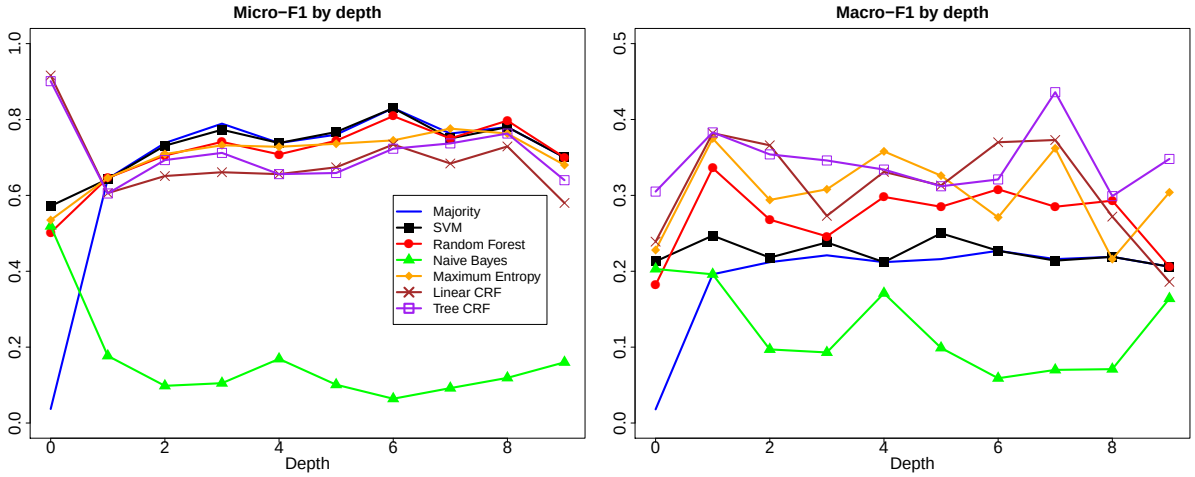


Figure 3: Micro- and macro-F1 scores by depth of tweet.

well across tweets of different depth within conversations. Figure 3 shows these results for tweets from depth 0 (source tweet) to depth 9. Further depths are omitted due to the small number of instances available. When we look at micro-averaged scores, we do not see a big performance difference across classifiers, except for the CRF classifiers performing better for source tweets; this is due to the fact that most of the source tweets tend to support a rumour, and hence sequential classifiers can learn this.

What is more interesting is to look again at the macro-averaged scores, where we see that the sequential approaches, especially the Tree CRF, consistently performs well for different levels of depth. More specifically, Tree CRF performs best in 7 out of 10 levels of depth analysed, with Linear CRF being better once (depth = 2) and Maximum Entropy being better twice (depth = 4 and 5).

7 Conclusions

We have introduced a novel way of tackling the rumour stance classification task, where a classifier has to determine if each tweet is supporting, denying, querying or commenting on a rumour's truth value. We mine the sequential structure of Twitter conversations in the form of users' replies to one another, extending existing approaches that treat each tweet as a separate unit. We have used two different sequential classifiers: a linear-chain CRF modelling tree-structured conversations broken down into branches, and a tree CRF modelling them as a graph that includes the whole tree. These classifiers have been compared with the non-sequential equivalent Maximum Entropy classifier, as well as other baseline classifiers, on eight Twitter datasets associated with breaking news.

While previous work had looked at the tweet as a single unit, we have shown that exploiting the discursive characteristics of interactions on Twitter, by considering probabilities of transitions within tree-structured conversational threads, can lead to significant improvements. Not only do we see that the linear sequence in a branch can be useful for the classifier to learn transitions, but also that having the whole picture of the tree showing the overall tendency of a conversation can further boost the performance of the classifier. Our results suggest that a tree CRF classifier outperforms all non-sequential classifiers, proving the utility of mining the conversational structure for stance classification, even when only local features are used.

To the best of our knowledge, this is the first attempt at aggregating the conversational structure of Twitter threads to produce classifications at the tweet level. Besides the utility of mining sequences from conversations for stance classification, we believe that our results will, in turn, encourage the study of sequential classifiers applied to other NLP tasks where the output for each tweet can benefit from the structure of the entire conversation, e.g., sentiment analysis and language identification.

Our plans for future work include testing additional sequential classifiers (e.g. LSTM). Moreover, while we have only tested local features for the purposes of making the experiments comparable, we also plan to test contextual features. This may also alleviate the effect of the class imbalance, producing results that are more satisfactory for minority classes, especially denials. Our approach assumes that rumours have been already identified or input by a human. An ambitious avenue for future work includes developing a rumour detection system whose output would be fed to the stance classification system.

Acknowledgments

This work has been supported by the PHEME FP7 project (grant No. 611233). This research utilised Queen Mary's MidPlus computational facilities, supported by QMUL Research-IT and funded by EPSRC grant EP/K000128/1.

References

Rob Abbott, Marilyn Walker, Pranav Anand, Jean E Fox Tree, Robeson Bowmani, and Joseph King. 2011. How can you say such things?!?: Recognizing disagreement in informal political argument. In *Proceedings of the Workshop on Languages in Social Media*, pages 2–11.

Kalina Bontcheva, Leon Derczynski, Adam Funk, Mark A. Greenwood, Diana Maynard, and Niraj Aswani. 2013. TwitIE: An open-source information extraction pipeline for microblog text. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*. Association for Computational Linguistics.

Nicholas FitzGerald, Giuseppe Carenini, Gabriel Murray, and Shafiq Joty. 2011. Exploiting conversational features to detect high-quality blog comments. In *Advances in Artificial Intelligence*, pages 122–127. Springer.

Sardar Hamidian and Mona T Diab. 2016. Rumor identification and belief investigation on twitter. In *Proceedings of NAACL-HLT*, pages 3–8.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. 2015. Real-time rumor debunking on twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1867–1870, New York, NY, USA. ACM.

Michal Lukasik, Trevor Cohn, and Kalina Bontcheva. 2015. Classifying tweet level judgements of rumours in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2590–2595.

Michal Lukasik, Kalina Bontcheva, Trevor Cohn, Arkaitz Zubiaga, Maria Liakata, and Rob Procter. 2016a. Using gaussian processes for rumour stance classification in social media. *arXiv preprint arXiv:1609.01962*.

Michal Lukasik, P. K. Srijith, Duy Vu, Kalina Bontcheva, Arkaitz Zubiaga, and Trevor Cohn. 2016b. Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter. In *Proceedings of the 54th Meeting of the Association for Computational Linguistics*, pages 393–398. Association for Computer Linguistics.

Lluís Màrquez, James Glass, Walid Magdy, Alessandro Moschitti, Preslav Nakov, and Bilal Randeree. 2015. Semeval-2015 task 3: Answer selection in community question answering. In *Proceedings of SemEval*.

Marcelo Mendoza, Barbara Poblete, and Carlos Castillo. 2010. Twitter under crisis: can we trust what we rt? In *Proceedings of the first workshop on social media analytics*, pages 71–79. ACM.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Saif M Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the International Workshop on Semantic Evaluation, SemEval*, volume 16.

Andreas C Müller and Sven Behnke. 2014. Pystruct: learning structured prediction in python. *The Journal of Machine Learning Research*, 15(1):2055–2060.

Rob Procter, Jeremy Crump, Susanne Karstedt, Alex Voss, and Marta Cantijoch. 2013a. Reading the riots: What were the police doing on twitter? *Policing and society*, 23(4):413–436.

Rob Procter, Farida Vis, and Alex Voss. 2013b. Reading the riots on twitter: methodological innovation for the analysis of big data. *International journal of social research methodology*, 16(3):197–214.

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599. Association for Computational Linguistics.

Zhonghua Qu and Yang Liu. 2011. Finding problem solving threads in online forum. In *Proceedings of IJCNLP*, pages 1413–1417.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180. Association for Computational Linguistics.

Sara Rosenthal and Kathleen McKeown. 2015. I couldnt agree more: The role of conversational structure in agreement and disagreement detection in online discussions. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 168.

Charles Sutton and Andrew McCallum. 2011. An introduction to conditional random fields. *Machine Learning*, 4(4):267–373.

Peter Tolmie, Rob Procter, Mark Rouncefield, Maria Liakata, and Arkaitz Zubiaga. 2015. Microblog analysis as a programme of work. *arXiv preprint arXiv:1511.03193*.

Wen Wang, Sibel Yaman, Kristin Precoda, Colleen Richey, and Geoffrey Raymond. 2011. Detection of agreement and disagreement in broadcast conversations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 374–378. Association for Computational Linguistics.

Li Zeng, Kate Starbird, and Emma S Spiro. 2016. #unconfirmed: Classifying rumor stance in crisis-related social media messages. In *Tenth International AAAI Conference on Web and Social Media*.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1395–1405. ACM.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Kalina Bontcheva, and Peter Tolmie. 2015. Crowdsourcing the annotation of rumourous conversations in social media. In *Proceedings of the 24th International Conference on World Wide Web*, pages 347–353. ACM.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PLoS ONE*, 11(3):1–29, 03.

Tweet Sarcasm Detection Using Deep Neural Network

Meishan Zhang[1], **Yue Zhang**[2] and **Guohong Fu**[1]
1. School of Computer Science and Technology, Heilongjiang University, China
2. Singapore University of Technology and Design
mason.zms@gmail.com,
yue_zhang@sutd.edu.sg,
ghfu@hotmail.com

Abstract

Sarcasm detection has been modeled as a binary document classification task, with rich features being defined manually over input documents. Traditional models employ discrete manual features to address the task, with much research effect being devoted to the design of effective feature templates. We investigate the use of neural network for tweet sarcasm detection, and compare the effects of the continuous automatic features with discrete manual features. In particular, we use a bi-directional gated recurrent neural network to capture syntactic and semantic information over tweets locally, and a pooling neural network to extract contextual features automatically from history tweets. Results show that neural features give improved accuracies for sarcasm detection, with different error distributions compared with discrete manual features.

1 Introduction

Sarcasm has received much research attention in linguistics, psychology and cognitive science (Gibbs, 1986; Kreuz and Glucksberg, 1989; Utsumi, 2000; Gibbs and Colston, 2007). Detecting sarcasm automatically is useful for opinion mining and reputation management, and hence has received growing interest from the natural language processing community (Joshi et al., 2016a). Social media such as Twitter exhibit rich sarcasm phenomena, and recent work on automatic sarcasm detection has focused on tweet data.

Tweet sarcasm detection can be modeled as a binary document classification task. Two main sources of features have been used. First, most previous work extracts rich discrete features according to the *tweet content* itself (Davidov et al., 2010; Tsur et al., 2010; González-Ibánez et al., 2011; Reyes et al., 2012; Reyes et al., 2013; Riloff et al., 2013; Ptáček et al., 2014), including lexical unigrams, bigrams, tweet sentiment, word sentiment, punctuation marks, emoticons, quotes, character ngrams and pronunciations. Some of these work uses more sophisticated features, including POS tags, dependency-based tree structures, Brown clusters and sentiment indicators, which depend on external resources. Overall, ngrams have been among the most useful features.

Second, recent work has exploited contextual tweet features for sarcasm detection (Rajadesingan et al., 2015; Bamman and Smith, 2015). Intuitively, the *history behaviors* for a tweet author can be a good indicator for sarcasm. Rajadesingan et al. (2015) exploit a behavioral approach to model sarcasm, using a set of statistical indicators extracted from both the target tweet and relevant history tweets. Bamman and Smith (2015) study the influences of tweet content features, author features, audience features and environment features, finding that contextual features are very useful for tweet sarcasm detection.

So far, most existing sarcasm detection methods in the literature leverage discrete models. While on the other hand, neural network models have gained much attention for related tasks such as sentiment analysis and opinion extraction, achieving the best results (Socher et al., 2013; dos Santos and Gatti, 2014; Vo and Zhang, 2015; Zhang et al., 2016). Success on these tasks shows potentials of neural network on sarcasm detection (Amir et al., 2016; Ghosh and Veale, 2016; Joshi et al., 2016b). There are two main advantages of using neural models. First, neural layers are used to induce features automatically,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2449–2460, Osaka, Japan, December 11-17 2016.

making manual feature engineering unnecessary. Such neural features can capture long-range and subtle semantic patterns, which are difficult to express using discrete feature templates. Second, neural models use real-valued word embedding inputs, which are trained from large scale raw texts, and are capable of avoiding the feature sparsity problem of discrete models.

In this paper, we exploit a deep neural network for sarcasm detection, comparing its automatic features with traditional discrete models. First, we construct a baseline discrete model that exploits the most typical features in the literature, including the features on the target tweet content and the features on historical tweets of the author, achieving competitive results as compared to the previous best systems.

Second, we build a neural model, with two sub neural networks to capture tweet content and contextual information, respectively. The two-component structure closely corresponds to the two feature sources of the baseline discrete model. We model the tweet content with a gated recurrent neural network (GRNN) (Cho et al., 2014b; Cho et al., 2014a), and use a gated pooling function for feature extraction. To model the salient words from the contextual tweets, we use pooling to extract features directly.

Results on a tweet datasets show that the neural model achieves significantly better accuracies compared to the discrete baseline, demonstrating the advantage of the automatically extracted neural features in capturing global semantic information. Further analysis shows that features from history tweets are as useful to the neural model as to the discrete model. We make our source code publicly available under GPL at `https://github.com/zhangmeishan/SarcasmDetection`.

2 Related Work

Features. Sarcasm detection is typically regarded as a classification problem. Discrete models have been used and most existing research efforts have focused on finding effective features. Kreuz and Caucci (2007) studied lexical features for sarcasm detection, finding that words, such as interjections and punctuation, are effective for the task. Carvalho et al. (2009) demonstrated that oral or gestural expressions represented by emoticons and special keyboard characters are useful indicators of sarcasm. Both Kreuz and Caucci (2007) and Carvalho et al. (2009) rely on unigram lexical features for sarcasm detection. More recently, Lukin and Walker (2013) extended the idea by using n-gram features as well as lexicon-syntactic patterns.

External sources of information have been exploited to enhance sarcasm detection. Tsur et al. (2010) applied features based on semi-supervised syntactic patterns extracted from sarcastic sentences of Amazon product reviews. Davidov et al. (2010) further extracted these features from sarcastic tweets. Riloff et al. (2013) identified a main type of sarcasm, namely contrast between a positive and negative sentiment, which can be regarded as detecting sarcasm using sentiment information. There has been work that comprehensively studies the effect of various features (González-Ibánez et al., 2011; González-Ibánez et al., 2011; Joshi et al., 2015).

Recently, contextual information has been exploited for sarcasm detection (Wallace et al., 2015; Karoui et al., 2015). In particular, contextual features extracted from history tweets by the same author has shown great effectiveness for tweet sarcasm detection (Rajadesingan et al., 2015; Bamman and Smith, 2015). We consider both traditional lexical features and the contextual features from history tweets under a unified neural network framework. Our observation is consistent with prior work: both sources of features are highly effective for sarcasm detection (Rajadesingan et al., 2015; Bamman and Smith, 2015).. To our knowledge, we are among the first to investigate the effect of neural networks on this task (Amir et al., 2016; Ghosh and Veale, 2016; Joshi et al., 2016b).

Corpora. With respect to sarcasm corpora, early work relied on small-scale manual annotation. Filatova (2012) constructed a sarcasm corpus from Amazon product reviews using crowdsourcing. Davidov et al. (2010) discussed the strong influence of hashtags on sarcasm detection. Inspired by this, González-Ibánez et al. (2011) used sarcasm-related hashtags as gold labels for sarcasm, creating a tweet corpus by treating tweets without such hashtags as negative examples. Their work is similar in spirit to the work of Go et al. (2009), who constructed a tweet sentiment automatically by taking emoticons as gold sentiment labels.

The method of González-Ibánez et al. (2011) was adopted by Ptáček et al. (2014), who created a

sarcasm dataset for Czech. More recently, both Rajadesingan et al. (2015) and Bamman and Smith (2015) followed the method for building a sarcasm corpus. We take the corpus of Rajadesingan et al. (2015) for our experiments.

Neural network models. Although only very limited work has been done on using neural networks for sarcasm detection, neural models have seen increasing applications in sentiment analysis, which is a closely-related task. Different neural network architectures have been applied for sentiment analysis, including recursive auto-encoders (Socher et al., 2013), dynamic pooling networks (Kalchbrenner et al., 2014), deep belief networks (Zhou et al., 2014), deep convolutional networks (dos Santos and Gatti, 2014; Tang et al., 2015) and neural CRF (Zhang et al., 2015). This line of work gives highly competitive results, demonstrating large potentials for neural networks on sentiment analysis. One important reason is the power of neural networks in automatic feature induction, which can potentially discover subtle semantic patterns that are difficult to capture by using manual features. Sarcasm detection can benefit from such induction, and several work has already attempted for it (Amir et al., 2016; Ghosh and Veale, 2016; Joshi et al., 2016b). This motivates our work.

3 Baseline Discrete Model

We follow previous work in the literature, building a strong discrete baseline model using features from both the *target tweet* itself and its *contextual tweets*. The structure of the model is shown in Figure 1(a), which consists of two main components, modeling the target tweet and its contextual tweets, respectively. In particular, the **local component** (the left of Figure 1(a)) is used to extract features $\mathbf{f}$ from the target tweet content, and the **contextual component** (the right of Figure 1(a)) is used to extract contextual features $\mathbf{f}'$ from the history tweets of the author. Based on $\mathbf{f}$ and $\mathbf{f}'$, a logistic regression is used to obtain the output:

$$\mathbf{o} = \mathrm{softmax}\big(W_o \cdot (\mathbf{f} \oplus \mathbf{f}')\big), \tag{1}$$

where the matrix W_o is the model parameter matrix, $\mathbf{o}$ is the output two-bit sarcasm/non-sarcasm vector, and $\oplus$ denotes vector concatenation.

3.1 The Local Component

Given an input tweet $w_1, w_2 \cdots w_n$, we extract a set of sparse discrete feature vectors $\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_n$ by instantiating a set of feature templates over each word, respectively. In particular, we follow Rajadesingan et al. (2015) and use three feature templates, including the current word w_i, the word bigram $w_{i-1}w_i$ and the word trigram $w_{i-2}w_{i-1}w_i$. The final local feature vector $\mathbf{f}$ is the sum of $\mathbf{f}_i$ from all words: $\mathbf{f} = \sum_{i=1}^{n} \mathbf{f}_i$.

3.2 The Contextual Component

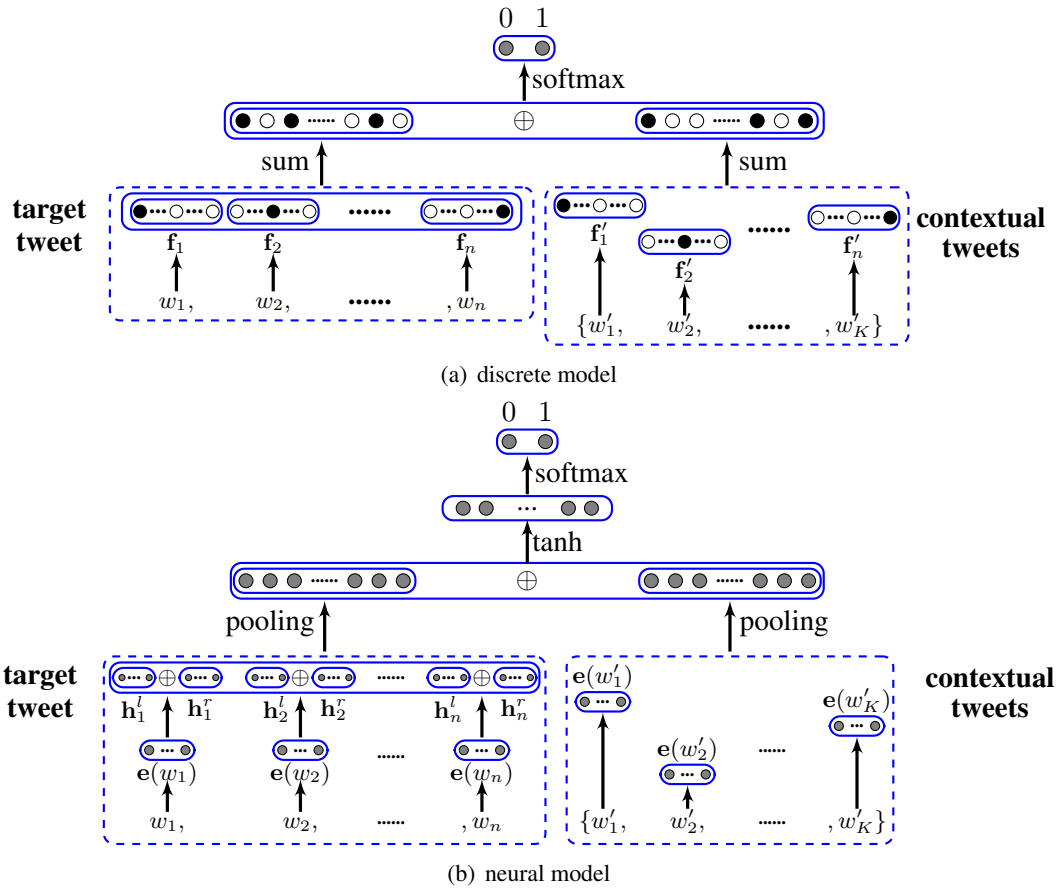
We follow Bamman and Smith (2015) for defining the features of the contextual tweets. In particular, a set of salient words are extracted from history tweets of the target tweet author, which can reflect the tendency of the author in using irony or sarcasm towards certain subjects.

First, we extract a number of most recent history tweets by using Twitter API[1], setting the maximum number of history tweets to 80.[2] The words in the history tweets are sorted by their *tf-idf* values. To estimate *tf* and *idf*, we regard the set of history tweets for a given tweet as one document, and use all the tweets in the training corpus to generate a number of additional documents. We choose a fixed-number of contextual tweet words with the highest *tf-idf* values for contextual features.

Denoting the set of words extracted from contexts as $\{\mathbf{w}'_1, \mathbf{w}'_2, \cdots, \mathbf{w}'_K\}$, where K is a hyper-parameter set manually, we use a single feature template w_i the extract a sparse feature vector $\mathbf{f}'_i$ for each word. The final contextual feature is the sum of all unigram features: $\mathbf{f}' = \sum_{i=1}^{K} \mathbf{w}'_i$. The set of baseline features, adopted from Rajadesingan et al. (2015) and Bamman and Smith (2015), are simple yet effective, giving highly competitive accuracies in our experiments.

[1] https://dev.twitter.com

[2] We use the data shared by Rajadesingan et al. (2015) directly, following their setting of history tweets.



(a) discrete model

(b) neural model

Figure 1: Discrete and neural models for tweet sarcasm detection (• denotes 1, ○ denotes 0, and • denotes a real-value feature).

4 Proposed Neural Model

In contrast to the discrete model, the neural model explores low-dimensional dense vectors as input. Figure 1(b) shows the overall structure of our proposed neural model, which has two components, corresponding to the **local** and the **contextual** components of the discrete baseline model, respectively. The two components use neural network structures to extract dense real-valued features $\mathbf{h}$ and $\mathbf{h}'$ from the local and history tweets, respectively, and we add a non-linear hidden layer to combine the neural features from the two components for classification. The output nodes can computed by:

$$\begin{aligned} \mathbf{c} &= \tanh(W_c(\mathbf{h} \oplus \mathbf{h}') + \mathbf{b}_c), \\ \mathbf{o} &= \text{softmax}(W_o\mathbf{c}), \end{aligned} \tag{2}$$

where the matrices W_c and W_o, and the vector $\mathbf{b}_c$ are model parameters.

As Figure 1 shows, the neural model is designed in such a way so that the correspondence between the model and the discrete baseline is maximized at the level of feature sources, for the convenience of direct comparison.

4.1 The Local Component

As shown on the left of Figure 1(b), we use a bi-directional gated recurrent neural network (GRNN) to model a tweet. The input layer of the network, represented by $\mathbf{x}_i$ at each position of the input tweet, is the concatenation of three consecutive word vectors, with the current word $\mathbf{w}_i$ in the center. With respect to the source of information, it is similar to the trigram feature templates of the baseline discrete model. Formally, at each word location i, the input vector is $\mathbf{x}_i = [\mathbf{e}(w_{i-1}), \mathbf{e}(w_i), \mathbf{e}(w_{i+1})]$, where $\mathbf{e}$ is a function to obtain dense embeddings for words based on a matrix E, which is a model parameter.

A recurrent neural network is used to capture sequential features automatically, hence giving semantic information over the whole input tweets. Compared with the vanilla recurrent neural network structure, gated recurrent neural networks such as long-short-term-memory (Hochreiter and Schmidhuber, 1997) apply gate structures to effectively reduce the issues of exploding and diminishing gradients (Pascanu et al., 2013; Yao et al., 2015), and therefore have been widely used as a more effective form of recurrent neural networks.

We exploit two efficient GRNNs to obtain a left-to-right ($\mathbf{h}_1^l \mathbf{h}_2^l \cdots \mathbf{h}_n^l$), and a right-to-left hidden node sequence ($\mathbf{h}_n^r \mathbf{h}_{n-1}^r \cdots \mathbf{h}_1^r$), respectively. Taking the left-to-right GRNN as an example, the hidden node vectors $\mathbf{h}_i^l$ are computed by:

$$\mathbf{h}_i^l = (\mathbf{1} - \mathbf{z}_i^l) \odot \mathbf{h}_{i-1}^l + \mathbf{z}_i^l \odot \tilde{\mathbf{h}}_i^l$$
$$\tilde{\mathbf{h}}_i^l = \tanh(W_1^l \mathbf{x}_i + U_1^l(\mathbf{r}_i^l \odot \mathbf{h}_{i-1}^l) + \mathbf{b}_1^l)$$
$$\mathbf{z}_i^l = \text{sigmoid}(W_2^l \mathbf{x}_i + U_2^l \mathbf{h}_{i-1} + \mathbf{b}_2^l)$$
$$\mathbf{r}_i^l = \text{sigmoid}(W_3^l \mathbf{x}_i + U_3^l \mathbf{h}_{i-1}^l + \mathbf{b}_3^l)$$

where the $\mathbf{z}_i^l$ and $\mathbf{r}_i^l$ are two gates, and $\odot$ denotes Hadamard product. W_1^l, U_1^l, W_2^l, U_2^l, W_3^l, U_3^l, $\mathbf{b}_1^l$, $\mathbf{b}_2^l$ and $\mathbf{b}_3^l$ are model parameters.

We use the same method to obtain $\mathbf{h}_i^r$ in the reverse direction, with the corresponding model parameters $W_1^l, U_1^l, \cdots\cdots, \mathbf{b}_3^l$ being W_1^r, U_1^r, W_2^r, U_2^r, W_3^r, U_3^r, $\mathbf{b}_1^r$, $\mathbf{b}_2^r$ and $\mathbf{b}_3^r$, respectively. After both hidden node sequences are computed, we concatenate the bi-directional hidden nodes at each position, obtaining $\mathbf{h}_1 \mathbf{h}_2 \cdots \mathbf{h}_n$ ($\mathbf{h}_i = \mathbf{h}_i^l \oplus \mathbf{h}_i^r$).

We apply a gated pooling function over the variable-length sequence $\mathbf{h}_1 \mathbf{h}_2 \cdots \mathbf{h}_n$ to project these GRNN hidden node features into a global feature vector $\mathbf{h}$. Formally, the pooling function is defined by $\mathbf{h} = \sum_{i=1}^n \alpha_i \odot \mathbf{h}_i$, where the α_i values are computed automatically by $\alpha_i \infty \exp\big(\tanh(W_g \mathbf{h}_i + \mathbf{b}_g)\big)$ with the constrain of $\sum_1^n \alpha_i = 1$, W_g and $\mathbf{b}_g$ are model parameters. Here α_is are vectors that control the bit-wise combination between the hidden vectors $h_i, i \in [1 \cdots n]$.

The gated pooling add to the degree of flexibility in the interpolation compared with max, min and average pooling techniques, which are commonly used to extract features from variable length vector sequences. For example, when all α_is are equal, the resulting pooling effect is the same as the average pooling function. This gated pooling mechanism is similar in spirit to the attention method of Bahdanau et al. (2014), but is used for each element in the operated vectors rather than the full vectors.

Note that our baseline discrete model and neural model have highly similar structures, differing mainly in the use of manual discrete features and automatic neural features. In particular, the discrete feature vector $\mathbf{f} = \sum_{i=1}^n \mathbf{f}_i$ can be regarded as being obtained by using a *sum* pooling function $\mathbf{f} = \sum_{i=1}^n \alpha_i \odot \mathbf{f}_i$, where $\alpha_i = \mathbf{1}$ ($i \in [1, n]$). Here $\mathbf{f}_i$ is a discrete feature vector with manual feature engineering. The neural model obtains $\mathbf{f}$ also by pooling, with α_i being trained automatically. Different from $\{\mathbf{f}_i\}$, the features $\{\mathbf{h}_i\}$ are obtained through automatical feature extraction via GRNN, rather than manual combination of one-hot features.

4.2 The Contextual Component

We follow the discrete baseline, using the same contextual tweet words extracted from history tweets for contextual features. Different from target tweet words, contextual tweet words are separate words without structures, and therefore it is unnecessary to use structured neural networks such as GRNNs to model them. As a result, we directly apply the gated pooling function to project their embedding vectors into a fixed-dimensional feature vector $\mathbf{h}'$.

5 Training

We use supervised learning with a training objective to minimize the cross-entropy loss over a set of training examples $(x_i, y_i)|_{i=1}^N$, plus with a l_2-regularization term,

$$L(\theta) = -\sum_{i=1}^{N} \log p_{y_i} + \frac{\lambda}{2} \parallel \theta \parallel^2,$$

where θ is the set of model parameters, and p_{y_i} is the model probability of the gold-standard output y_i, which is computed by using logistic regression over the output vector $\mathbf{o}$ in Eq (1) and (2) for the discrete and neural models, respectively.

Online AdaGrad (Duchi et al., 2011) is used to minimize the objective function for both discrete and neural models. All the matrix and vector parameters are initialized by uniform sampling in $(-0.01, 0.01)$. The initial values of the embedding matrix E can be assigned either by using the same random initialization as the other parameters, or by using word embeddings pre-trained over a large-scale tweet corpus. We obtain pre-trained tweet word embeddings using *GloVe* (Pennington et al., 2014)[3]. Embeddings are fine-tuned during training, with E belonging to model parameters.

6 Experiments

6.1 Experimental Settings

6.1.1 Data

We use the dataset of Rajadesingan et al. (2015) to conduct our experiments, collected by querying the Twitter API using the keywords #sarcasm and #not, and filtering retweets and non-English tweets automatically. In total, Rajadesingan et al. (2015) collected 9,104 tweets that are self-described as sarcasm by the authors. We stream the tweet corpus using the tweet IDs they provide.[4]

We remove the #sarcasm and #not hashtags from the tweets, assigning to them the *sarcasm* output tags for training and evaluation. General tweets that are *non-sarcastic* are also obtained using the tweet IDs shared by Rajadesingan et al. (2015). For each tweet, a set of history tweets are extracted using Twitter API, for obtaining of contextual tweet words. We remove the #sarcasm and #not hashtags of the history tweets also, in order to avoid predicting sarcasm by using explicit clues.

Our models are evaluated on a balanced and an imbalanced dataset, respectively, where the balanced dataset includes equal sarcastic and non-sarcastic tweets, and the imbalanced dataset has a sarcasm:non-sarcasm ratio of 1:4.

6.1.2 Evaluation

We perform ten-fold cross-validation experiments and exploit the overall accuracies of sarcasm detection as the major evaluation metric. We report the macro F-measures as well, considering the data imbalance. Concretely, for both sarcasm and non-sarcasm, we compute their precisions, recalls and F-measures, respectively, and then we report the averaged F-measure. To tune the model hyper-parameters, we choose 10% of the training dataset as the development corpus.

6.2 Hyper-parameters

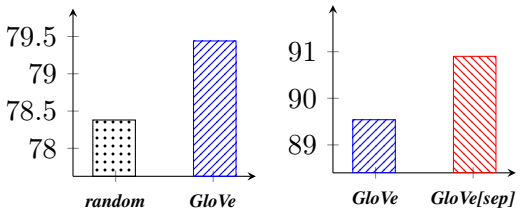
There are several important hyper-parameters in our models, and we tune their values using the development corpus. For both the discrete and neural models, we set the regularization weight $\lambda = 10^{-8}$ and the initial learning rate $\alpha = 0.01$. For the neural models, we set the size of word vectors to 100, the size of hidden vectors in GRNNs to 100, and the size of the non-linear combination layer to 50. One exception is the maximum number of history tweet words, which is 100 as default in align with Bamman and Smith (2015). We will show that the performance is still increasing when the number becomes larger in the next subsection.

6.3 Development Results

We conduct development experiments to study the effect of pre-trained word embeddings for the neural models, as well as the effect of the contextual information for both sparse and neural models. These experiments are performed on the balanced dataset.

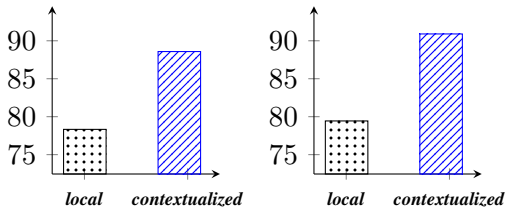
[3] http://nlp.stanford.edu/projects/glove/

[4] We are unable to obtain all the sarcastic tweets, due to modified authorization status of some tweets.



(a) local (b) contextualized

Figure 2: Influence of word representations.



(a) discrete (b) neural

Figure 3: Influence of contextual features.

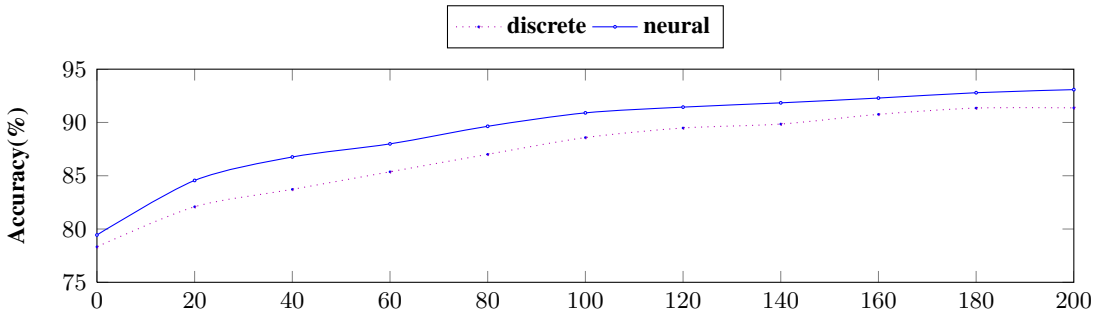


Figure 4: Developmental results with respect to different number of history tweet words.

6.3.1 Initialization of Word Embeddings

We use the neural model with only local features to evaluate the effect of different word embedding initialization methods. As shown in Figure 2(a), a better accuracy is obtained by using *GloVe* embeddings for initialization compared with random initialization. The finding is consistent with previous results in the literature on other NLP tasks, which show that pre-trained word embeddings can bring better accuracies (Collobert et al., 2011; Chen and Manning, 2014).

6.3.2 Differentiating Local and Contextual Word Embeddings

We can obtain embeddings of contextual tweet words using the same looking-up function as target tweet words, thereby giving each word a unique embedding regardless whether it comes from the target tweet to classify or its history tweets. However, the behavior of contextual tweet words should intuitively be different, because they are used as different features. An interesting research question is that whether separate embeddings lead to improved results. We investigate the question by using two embedding look-up matrices E and E', for target tweet words and contextual tweet words, respectively. The result in Figure 2(b) confirms our assumption, showing an improved accuracy by separating the two types of embeddings for each word.

Based on the above observation, we use *GloVe* embeddings for initialization in our final neural models, and use separate embedding matrices for target and contextual tweet words.

6.3.3 Effect of Contextual Features

Previous work has shown the effectiveness of contextual features for discrete sarcasm detection models (Rajadesingan et al., 2015; Bamman and Smith, 2015). Here, we study their effectiveness under both discrete and neural settings. The results are shown in Figure 3. It can be seen that contextual tweet information is highly useful under the neural setting also, which is consistent with previous work for the discrete models.

In more detail, we look at the performance with different maximum number of contextual words, in order to see the potential of contextual features. Figure 4 shows the development results, with the number range from 0 to 200, where 0 denotes the local model. As shown, the performance is consistently increasing with the increase of maximum contextual word number, although in this work we choose this

Model	Balanced		Imbalanced	
	Accuracy	F-measure	Accuracy	F-measure
Local				
baseline	78.55	78.53	86.45	75.14
neural	**79.29**‡	**79.36**‡	**87.25**‡	**77.37**‡
Riloff et al. (2013)	77.26	–	78.40	–
baseline l_1	78.56	–	81.63	–
Contextualized				
baseline	88.10	88.11	91.15	85.92
neural	**90.74**‡	**90.74**‡	**94.10**‡	**90.26**‡
SCUBA++	86.08	–	89.81	–

Table 1: Final results of our proposed models, where ‡ denotes a p-value below 10^{-3} compared to the baseline by pairwise t-test.

value by 100 to align with (Bamman and Smith, 2015).

6.4 Final Results

Table 1 shows the final results of our proposed models on both the balanced and the imbalanced datasets. The neural models show significantly better accuracies compared to the corresponding discrete baselines. Take the balanced data for example. Using only local tweet features, the neural model achieves an accuracy of 78.55%, significantly higher compared to the accuracy of 79.29% by the discrete model. Using also context tweet features, the accuracy of the neural model goes up to 90.74%, showing the strength of the history information. The F-measure values are consistent with the accuracies. These results demonstrate large advantages for the neural models on the task.

One interesting finding is that although the accuracies of imbalanced dataset are higher than those of balanced one, the micro F-measure values are on the contrary. The most possible reason could be the label bias of the imbalanced dataset, because detailed results show that the F-measures of sarcasm decrease significantly on the imbalanced dataset. According the final results, we can find both neural and contextual features can make up the F-measure gaps between balanced and imbalanced datasets, which further demonstrates the advantages of our final model.

We also compare the neural model with other sarcasm detection models in the literature. Shown in Table 1, Riloff et al. (2013) is lexicon-based model based on target tweet words only, identifying sarcasm by checking whether both positive and negative sentiment exist. *SCUBA++* shows the best results of Rajadesingan et al. (2015), using a contextualized model. Our baseline model gives higher accuracies compared to this state-of-the-art model, despite using similar features. One reason can the use of different optimization. For example, *baseline-l_1* shows the accuracy of our baseline using l_1 regularization instead of l_2, which yields variations of up to 1%. Nevertheless, the main purpose of the comparison is to show that our baseline is comparable to the best systems. Bamman and Smith (2015) report an accuracy of 75.4% on a balanced dataset, which is lower than our result. However, they performed evaluation on a different set of data, thus the results are not directly comparable.

6.5 Analysis

In order to better understand the differences between neural and manual features, we compare the discrete and neural models in more details on the test dataset. We focus on the balanced setting, and compare the contextualized models.

6.5.1 Error Characteristics

Figure 5 shows the output sarcasm probabilities of both models on each test tweet, where the x-axis represents the discrete model and the y-axis represents the neural model. The shapes + and • represent the gold-standard sarcasm and non-sarcasm labels, respectively. A probability value above 0.5 corresponds

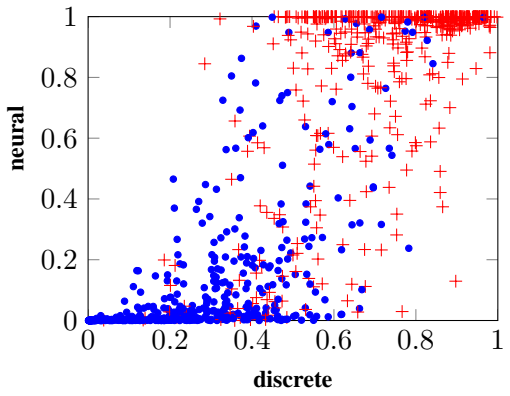


Figure 5: Error characteristic comparison, where + denotes sarcasm and • denotes non-sarcasm.

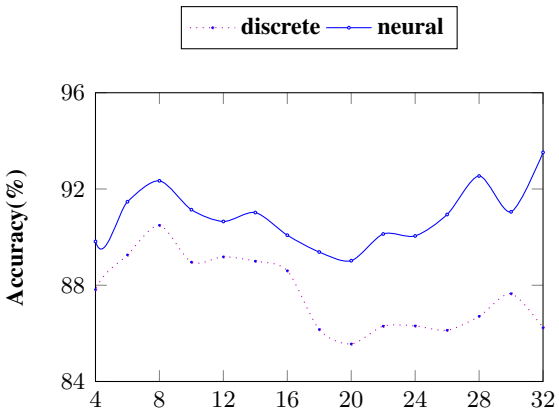


Figure 6: Accuracies against tweet length.

sarcasm	non-sarcasm
I guess finally knowing what it could have been makes me better .	so happy my brother has so many good people to help him with his move next weekend.
trying to fix my car is exactly what i wanna be doing on a saturday night.	Never go a day without telling your parents you love them

Table 2: Examples which the neural model predicted correctly but the discrete model incorrectly.

to the *sarcastic* output label. Intuitively, "+"s on the right half and "•"s on the left half of the figure show the examples that the discrete model predicts correctly, and "+"s on the top half and "•"s on the bottom half of the figure show the examples that the neural model predicts correctly.

As shown in the figure, most "+"s are in the top-right area, and most "•"s are in the bottom-left area, which indicates that the accuracies of both models are reasonably high. On the other hand, the samples are more scattered along the x-axis. This shows that the neural model is more confident in its predictions for most examples, demonstrating the discriminate power of the automatic neural features as compared with the manual discrete features.

6.5.2 Impact of Tweet Length

The GRNN neural model can potentially capture non-local syntactical and semantic information. We verify this assumption by comparing the accuracies of the neural and discrete models with respect to the tweet length. As shown in Figure 6, the neural model consistently outperforms the discrete model with respect to the tweet length. For longer tweets, the accuracies of the discrete model drops significantly, but those of the neural model remains stable.

6.5.3 Example Outputs

Table 2 shows some example sentences that the neural model predicted correctly, but the discrete model predicted incorrectly. Understanding sarcasm in the positive case requires global semantic information, which is better captured by non-local features from the recurrent neural network model. For the two cases with *non-sarcasm* gold labels, there are surface features such as "so happy", "so many" and "never", which are useful indicators of sarcasm for the discrete model. These features are local and can occur in both sarcasm and non-sarcasm tweets, thereby reducing the confidence of the discrete model (as shown in Figure 5) and can cause relatively more mistakes.

7 Conclusion

We constructed a deep neural network model for tweet sarcasm detection. Compared with traditional models with manual discrete features, the neural network model has two main advantages. First, it is free

from manual feature engineering and external resources such as POS taggers and sentiment lexicons. Second, it leverages distributed embedding inputs and recurrent neural networks to induce semantic features. The neural network model gave improved results over a state-of-the-art discrete model. In addition, we found that under the neural setting, contextual tweet features are as effective for sarcasm detection as with discrete models.

Acknowledgments

We thank the anonymous reviewers from COLING 2016, ACL 2016, NAACL 2016, AAAI 2016 and EMNLP 2015 for their constructive comments, which helped to improve the final paper. This work is supported by National Natural Science Foundation of China (NSFC) grants 61602160 and 61672211, Natural Science Foundation of Heilongjiang Province (China) grant F2016036, the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301 and SRG ISTD 2012 038 from Singapore University of Technology and Design.

References

Silvio Amir, Byron C. Wallace, Hao Lyu, Paula Carvalho, and Mario J. Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. In *CONLL 2016*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

David Bamman and Noah A Smith. 2015. Contextualized sarcasm detection on twitter. In *Ninth International AAAI Conference on Web and Social Media*.

Paula Carvalho, Luís Sarmento, Mário J Silva, and Eugénio De Oliveira. 2009. Clues for detecting irony in user-generated contents: oh...!! it's so easy;-). In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 53–56.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

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

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 107–116.

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014*, pages 69–78.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Elena Filatova. 2012. Irony and sarcasm: Corpus generation and analysis using crowdsourcing. In *LREC*, pages 392–398.

Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network. In *Proceedings of the 7th WASSA*, pages 161–169.

Raymond W Gibbs and Herbert L Colston. 2007. *Irony in language and thought: A cognitive science reader*. Psychology Press.

Raymond W Gibbs. 1986. On the psycholinguistics of sarcasm. *Journal of Experimental Psychology: General*, 115(1):3.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th ACL*, pages 581–586.

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

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd ACL*, pages 757–762.

Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. 2016a. Automatic sarcasm detection: A survey. *arXiv preprint arXiv:1602.03426*.

Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. 2016b. Are word embedding-based features useful for sarcasm detection? In *EMNLP*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd ACL*, pages 655–665. Association for Computational Linguistics.

Jihen Karoui, Benamara Farah, Véronique MORICEAU, Nathalie Aussenac-Gilles, and Lamia Hadrich-Belguith. 2015. Towards a contextual pragmatic model to detect irony in tweets. In *Proceedings of the 53rd ACL*, pages 644–650.

Roger J Kreuz and Gina M Caucci. 2007. Lexical influences on the perception of sarcasm. In *Proceedings of the Workshop on computational approaches to Figurative Language*, pages 1–4.

Roger J Kreuz and Sam Glucksberg. 1989. How to be sarcastic: The echoic reminder theory of verbal irony. *Journal of Experimental Psychology: General*, 118(4):374.

Stephanie Lukin and Marilyn Walker. 2013. Really? well. apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue. In *Proceedings of the Workshop on Language Analysis in Social Media*, pages 30–40.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of ICML*, pages 1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 EMNLP*, pages 1532–1543.

Tomáš Ptáček, Ivan Habernal, and Jun Hong. 2014. Sarcasm detection on czech and english twitter. In *Proceedings of COLING 2014*, pages 213–223, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 97–106.

Antonio Reyes, Paolo Rosso, and Davide Buscaldi. 2012. From humor recognition to irony detection: The figurative language of social media. *Data & Knowledge Engineering*, 74:1–12.

Antonio Reyes, Paolo Rosso, and Tony Veale. 2013. A multidimensional approach for detecting irony in twitter. *Language resources and evaluation*, 47(1):239–268.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *EMNLP*, pages 704–714.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the EMNLP*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1014–1023, July.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. Icwsm-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *ICWSM*.

Akira Utsumi. 2000. Verbal irony as implicit display of ironic environment: Distinguishing ironic utterances from nonirony. *Journal of Pragmatics*, 32(12):1777–1806.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *Proceedings of IJCAI*, BueNos Aires, Argentina, August.

Byron C. Wallace, Do Kook Choe, and Eugene Charniak. 2015. Sparse, contextually informed models for irony detection: Exploiting user communities, entities and sentiment. In *Proceedings of the 53rd ACL*, pages 1035–1044.

Kaisheng Yao, Trevor Cohn, Katerina Vylomova, Kevin Duh, and Chris Dyer. 2015. Depth-gated recurrent neural networks. *arXiv preprint arXiv:1508.03790*.

Meishan Zhang, Yue Zhang, and Duy Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *Proceedings of the EMNLP*, pages 612–621.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Shusen Zhou, Qingcai Chen, Xiaolong Wang, and Xiaoling Li. 2014. Hybrid deep belief networks for semi-supervised sentiment classification. In *Proceedings of COLING 2014*, pages 1341–1349. Dublin City University and Association for Computational Linguistics.

Agreement and Disagreement: Comparison of Points of View in the Political Domain

Stefano Menini
University of Trento
Fondazione Bruno Kessler
Trento, Italy
menini@fbk.eu

Sara Tonelli
Fondazione Bruno Kessler
Trento, Italy
satonelli@fbk.eu

Abstract

The automated comparison of points of view between two politicians is a very challenging task, due not only to the lack of annotated resources, but also to the different dimensions participating to the definition of agreement and disagreement. In order to shed light on this complex task, we first carry out a pilot study to manually annotate the components involved in detecting agreement and disagreement. Then, based on these findings, we implement different features to capture them automatically via supervised classification. We do not focus on debates in dialogical form, but we rather consider sets of documents, in which politicians may express their position with respect to different topics in an implicit or explicit way, like during an electoral campaign. We create and make available three different datasets.

1 Introduction

When it comes to evaluate whether the statements of two persons are in agreement or not about a topic, several past works approach the problem by classifying the single statements as supporting or opposing the topic, considering the task as a variant of sentiment analysis (Somasundaran and Wiebe, 2010).

These approaches proved to be reliable in specific settings, where the goal of the statements was to express support or opposition w.r.t. the topic. However, when applied to the political domain, they often result into an oversimplified representation of the dynamics involved in the comparison of two positions. In our view, several aspects contribute to the assessment of agreement and disagreement in the political domain, requiring to be properly addressed. As an example, let us consider two excerpts uttered by Kennedy and Nixon in 1960 about the situation in *Cuba* under the *Castro regime*:

> **Kennedy:** *"There is not any doubt we had great influence in Cuba, and I think it is unfortunate that we did not use that influence more vigorously to persuade Castro to hold free, open elections, so that the people of Cuba could have made the choice."*

> **Nixon:** *"What we must remember too is that the United States has the military power - and Mr. Castro knows this - to throw him out of office tomorrow or the next day or any day that we choose."*

The two examples show that neither Nixon nor Kennedy support the Castro regime and that they both share the same negative sentiment about it in their speeches. But beside being on the same side in contrasting the regime, their points of view on it are, from a pragmatic perspective, very different: while Kennedy supports free elections, Nixon does not hesitate to remark United States military power and the possibility to remove Castro in any moment. Overall, the two positions are in disagreement. This example shows that the sentiment and the proposed solution are two relevant aspects to define a politician's attitude w.r.t. a topic, and that the contribution of these two aspects need to be better studied when comparing different points of view: while one could argue that disagreement is expressed by

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2461–2470, Osaka, Japan, December 11-17 2016.

different sentiment and different semantic content, our pilot analysis shows that the boundaries are not so clear-cut.

The main contribution of this work lies in the presentation of a first feasibility study on manually annotated data from the political domain (Section 4), where we decompose the notion of agreement / disagreement, and in the presentation of a novel system (Section 5), which takes into account the insight gained during the feasibility study. We evaluate our approach on three datasets created for this task, which we make freely available[1], and show that our approach is effective on each of them: the one created for the feasibility study presented in Section 4.1, an extended version of the same dataset, and a larger one extracted from Debatepedia presented in Section 5.2.

2 Related Work

Given the highly polarized nature of political debates, we can find in the literature many works focused on classifying political statements as supporting or opposing a debated topic. This classification can be approached in different ways, for instance by emphasizing the role of sentiment polarity in a statement, or using topic modeling to define each position.

In (Somasundaran and Wiebe, 2009; Somasundaran and Wiebe, 2010) the authors propose a way to classify stances. They gather posts on different topics from online forums, and classify the statements from these debates as in favor or against the debated issue. They use the MPQA corpus to automatically generate a lexicon of entries indicative of a positive and negative argument and add information about the use of modal verbs and sentiment-based features. A similar task is proposed in Abbott et al. (2011) to recognize disagreement in online political forums between quoted text and a given response. The goal was achieved by using word-based features (for example discourse markers) as well as meta-information. Another work investigating stance classification of online posts was presented in Anand et al. (2011): there, the authors attempt to improve the unigram baseline by adding more cognitive-motivated features such as contextual information and opinion dependencies to define the target of opinion words. The results show that the use of these features improves classification results for many topics.

Another work from Gottipati et al. (2013) proposes to learn topics and support/opposition from discussions in Debatepedia by using a model based on the probabilistic distribution of the terms over the topics and the sides of the debate. A similar work on Debatepedia has been proposed by Awadallah et al. (2012), to classify quotes as belonging to a topic and supporting or opposing it.

Other approaches to classification rely on corpus-specific features, as in Thomas et al. (2006), who detect support and opposition to legislation in congressional debates by using speech transcriptions as well as records on voting, information about the speakers and the relations among them. Other works focus on the identification of agreement and disagreement in dialogues, such as (Galley et al., 2004; Hillard et al., 2003). They classify consecutive speech transcription segments produced by different speakers as positive or negative with respect to the discussed topic by using lexical, structural, and prosodic features.

Compared to previous works, our task is different in that we perform pairwise agreement/disagreement detection between two points of view: our focus is on the relation between the two rather than the single stance. Another difference lies in the types of textual units we want to classify: we do not work on single statements but rather on longer snippets including several sentences, based on the assumption that in the political domain a person's position with respect to a topic may not be overtly expressed. Our goal is to generalise over single statements and detect agreement and disagreement based on a broader, but not necessarily explicit, textual context.

3 Task Description

In the political domain, public debates in which two opponents discuss their point of view on specific topics, usually suggested by a moderator, are just one of several occasions in which political agendas are described. If we consider an electoral campaign, for instance, candidates issue declarations, mostly in the form of speeches, in which several topics are more or less explicitly discussed, and only towards

[1] Available at `https://dh.fbk.eu/resources/agreement-disagreement`

the end of the campaign a direct confrontation between opponents takes place. While past approaches to detect direct support or opposition in dialogues (Galley et al., 2004; Hillard et al., 2003) could be appropriate to analyse such direct confrontations, they may fail to capture the complexity of other types of statements. For example, the limited length of turns in dialogues, the presence of specific emphatic expressions and the need to focus on one topic at a time all contribute to the automated detection of agreement or disagreement in direct confrontations, but this information is not necessarily present when we consider larger documents collections, containing the public declarations of a politician. This second scenario is the focus of the present work. Specifically, given two document collections containing the public declarations and speech transcriptions of two politicians, our goal is to assess whether the two agree or not on a topic. In this scenario, we cannot assume that, each time a topic is mentioned, a position is explicitly expressed, but rather that it can be understood given a set of statements related to the topic. Besides, we cannot assume that two datasets representing two politicians contain direct references or replies between the opponents on a given topic. In this complex scenario, we first define a methodology to process the document collections and extract the text passages to be compared and classified. Then, we propose an approach to classify such pairs based on supervised learning, that takes into account features capturing the relevant dimensions analysed in the pilot study.

4 Pilot Study

To investigate which dimensions are involved in the perception of agreement and disagreement in the political domain, we first perform an exploratory study by manually annotating a dataset created for the task.

4.1 1960 Presidential Campaign Dataset

We collect the transcription of discourses and official declarations issued by Nixon and Kennedy during 1960 presidential campaign from The American Presidency Project[2]. The corpus includes 881 documents[3] and more than 1,6 million tokens (around 830,000 tokens for Nixon and 815,000 tokens for Kennedy).

We define 38 topics relevant to the electoral campaign with the help of a history scholar and we represent them via a set of manually defined keywords (e.g. a topic about the Agricultural program defined as *[agricultural program, agricultural policy, farmer, farm]*, about Education *[education, school]* or about Atomic energy *[atomic energy, nuclear energy, atomic power, nuclear power]*).

For each topic, for example *Education*, we extract from the two sets of documents all sentences containing at least one of the keywords defining the topic, plus the previous and the following sentence. The decision of extending the selection to three sentences is taken in order to have a more complete portion of text about the topic. These three sentences correspond to a *text excerpt*.

For each topic, we finally pair five random excerpts from Kennedy with five random excerpts from Nixon to create our *snippets*. We used this approach because the annotation task focused on general questions that needed enough context to be answered, and single excerpts may not provide enough information for the task.

4.2 Corpus Annotation

We build 350 snippets across all the topics and ask two independent annotators to annotate them using the CrowdFlower web interface[4]. We did not open the task to the public, but we relied on trusted annotators. For each pair, the following questions are asked:

1. *Are Nixon's and Kennedy's statements about the topic in agreement, disagreement or neutral?*

2. *What is Kennedy's sentiment with respect to the topic?*

[2]The American Presidency Project, by John T. Woolley and Gerhard Peters (`http://www.presidency.ucsb.edu/1960_election.php`)

[3]These documents are freely released under the NARA public domain license.

[4]`https://www.crowdflower.com/`

3. *What is Nixon's sentiment with respect to the topic?*

4. *Are the solutions or initiatives proposed by Nixon and Kennedy similar or different?*

For each pair, we collect *i)* a judgment about the *agreement/disagreement* relation between Nixon and Kennedy on the topic (*Question 1*), *ii)* a judgment on Kennedy's and Nixon's *sentiment* w.r.t. the topic (*Question 2 and 3*) being either *Positive*, *Neutral* or *Negative* and *iii)* a judgment on the *solutions* proposed by the two candidates as *similar*, *different* or *neutral*, i.e. no solution proposed (*Question 4*). The inter-annotator agreement as provided by Crowdflower is 73% for Question 1, 69% for Question 2, 70% for Question 3, and 78% for Question 4. Crowdflower re-assigns then each snippet for which there is no agreement to one of the three categories based on a 'confidence score' (we could not obtain the exact formula used to compute this value but we assume it takes into account annotators' reliability).

4.3 Data Analysis

Based on the answers to Question 1, the dataset is composed by 203 pairs of snippets where Nixon and Kennedy agree on the topic, 97 pairs in which they disagree and 50 pairs where they neither agree nor disagree. We focus on the pairs in agreement or disagreement and further analyse the contribution of *sentiment* and *proposed solution* to the perception of agreement. Results are reported in Table 1. The analysis shows that being in agreement does not necessarily imply having the same sentiment or suggesting the same solution (no solution is suggested in most of the cases). In cases of disagreement, instead, a different sentiment and a different solution prevail.

Overall, if we cast the agreement / disagreement prediction on the basis of the sentiment, we can correctly guess 221 pairs over 350 (63.1%). Similarly, if we cast the agreement / disagreement prediction based on the fact that same or different solutions are proposed, we can correctly guess 173 pairs over 350 (49.4%). These values show that sharing the same *sentiment*, and proposing the same *solution*, *per se* are not reliable indicators of agreement.

	Same Sentiment	Same Solution
Agreement Nixon-Kennedy (203 pairs)	134/203 (66.0%)	89/203 (43.8%)
	Different Sentiment	Different Solution
Disagreement Nixon-Kennedy (97 pairs)	87/97 (89.7%)	84/97 (86.5%)

Table 1: Correspondence between agreement/disagreement and similar/different sentiment or solution.

We further analyse the direct relation between the sentiment and the semantic content (i.e. solution proposed) of the snippet pairs in the dataset with the help of Figure 1. The graph highlights that the pairs in which the two politicians share the same sentiment (on the left side) are split in half between similar and different solutions (on the right side). Instead, most of the pairs with a different sentiment (left side of the graph) end to different solutions, but only half of the pairs with different solutions (right side of the graph) derive from statements with a different sentiment.

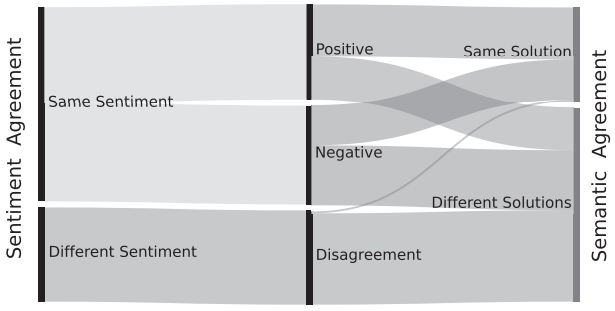


Figure 1: Relation between sentiment (annotations from Questions 2 and 3) and semantic content (annotations from Question 4) in Nixon's and Kennedy's statements.

The main findings of this pilot study can be summarised as follows: *i)* there is not a direct correspondence between the fact that Nixon and Kennedy agree on a topic and the fact that they share the same sentiment, *ii)* there is not a direct correspondence between the fact that Nixon and Kennedy agree on a topic and the fact that they propose the same initiative or solution, and *iii)* the sentiment and semantic content of the statements contribute both to defining agreement/disagreement between the two politicians. This implies that automatically classifying agreement and disagreement in this scenario needs to be addressed with appropriate features that can capture these different dimensions.

5 Experimental Setup

Based on the insight gained in the pilot study, we develop a system to automatically classify agreement and disagreement between two politicians (see Question 1 in the annotation task) trying to integrate information related to sentiment and semantic content of the statements (Questions 2-4). We classify pairs of *snippets*, one for each politician. For the task we adopt a supervised machine learning approach using LIBSVM (Chang and Lin, 2011) to train a Support Vector Machine (SVM). We evaluate our approach on the three datasets described in Section 5.2.

5.1 Features

We rely on three main categories of features. The contribution of speakers' sentiment to their agreement/disagreement is represented by a set of *sentiment-based features* (e.g. the sentiment of the statements and sentiment of the topic). To capture the differences and similarities in the proposed solutions, we use a set of *semantic features* (e.g word embeddings, cosine similarity and entailment), based on the intuition that two texts proposing the same solution are more likely to be semantically similar than two texts proposing different solutions. Finally, we add a set of *surface features* (e.g. the lexical overlap and the use of negations) that we expect to bear some information on the relation between the two snippets.

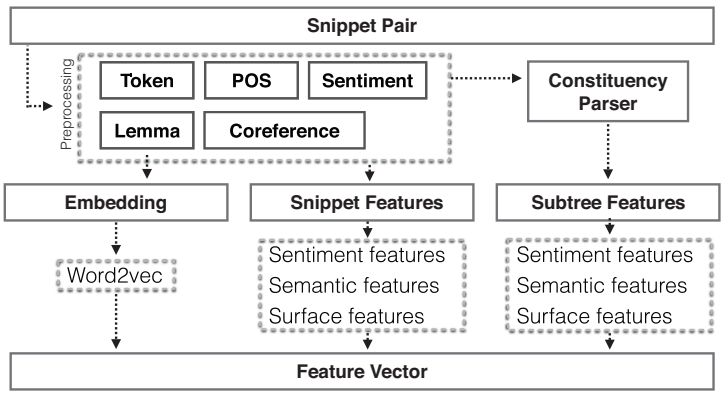


Figure 2: Features Extraction Pipeline

Figure 2 shows the pipeline we implemented for feature extraction. After preprocessing, all the features, except for word embeddings, are extracted at two different levels of granularity. In the first one, we extract the features at snippet level, which provide more information about the context in which the topic is used. In the second one, we focus only on the portion of text directly related to the topic. We use the Stanford Parser (De Marneffe et al., 2006) to extract from the sentences in the snippets all the subtrees containing the keywords representing the topic, and then we extract the features from them. With this pruning, we focus less on the context and more on the information which is directly related to the topic.

For each pair of snippets, we extract the following features:

Sentiment information: These features are inspired by previous works using sentiment information to predict a speaker's opinion on a topic (Pang and Lee, 2008; Abbasi et al., 2008). Each pair is represented by four sentiment scores, two scores for each snippet in the pair. We rely on the sentiment analysis module in the Stanford CoreNLP (Socher et al., 2013). We use a global sentiment score for each snippet (obtained by the average of the sentiment score of each sentence in it), and a score for the sentiment in

the subtrees related to the topic (obtained by the average of the sentiment score of each content word in the subtrees).

Word embeddings: Past works showed that word embeddings are an effective tool to define ideological positions in political documents (Iyyer et al., 2014). In our case, we do not focus on the sentence level, but rather on the keywords defining the topics debated in our pairs. We treat each snippet separately and we obtain two vectors for each pair: one vector representing the keywords of the topic in the first snippet and one vector for the topic in the second snippet (e.g. a vector for *Castro Regime* in Kennedy and a vector for *Castro Regime* in Nixon). The vectors are extracted using Word2vec (Mikolov et al., 2013) on each snippet (425 words on average with the topic occurring multiple times), with continuous bag-of-word algorithm, a windows size of 8 and a vector dimensionality of 50. We use this feature assuming that, when two people agree (i.e. have a similar point of view) on a topic, their respective vectors are more similar than when they are in disagreement.

Cosine similarity: In addition to the representation of the topics based on word embeddings, we use a set of features to quantify the relatedness between the way the two speakers talk about the topic. From each snippet in the pair, we extract two types of semantic vectors based on the co-occurrences (Turney et al., 2010) of topic keywords: one is computed over the entire snippet, while the other considering only the topic subtree. Co-occurrences are extracted from a window of 8 tokens and weighed using local pointwise mutual information. We then compute the cosine similarity between the vectors of the two sides of the snippet.

Entailment: The presence of entailment between the two snippets can be relevant to define if the position expressed by a speaker is accepted by the other (Cabrio and Villata, 2012). For this feature, we use the Excitement Open Platform (Magnini et al., 2014). For each pair, we use information about the entailment between the two snippets (in both the directions) and between the text in the subtrees related to the topic (in both the directions).

Lemma overlap: Past works showed that lexical overlap contributes to determining topical alignment between two texts (Somasundaran et al., 2009). Therefore, we compute lemma overlap of nouns, verbs and adjectives between two snippets. Although lexical overlap is already integrated in the textual entailment features, we believe that this information can provide useful information also in isolation.

Negation: For each snippet, we extract two features related to explicit negation cues (e.g. *not*, *don't*, *never*), adopting the list used in Councill et al. (2010). Using the parse tree of the snippets, we identify the words under the scope of a negation, and then consider as features *i)* the number of negated words in each snippet (normalized to its length) and *ii)* the percentage of the overlapping lemmas that in one snippet are under a negation. We expect that, if the same words are negated in a snippet and not in the other, this information can shed light on the relation between them.

5.2 Datasets

We evaluate our approach on three different datasets. The first is the 1960 Presidential Campaign dataset presented in the pilot study (300 snippet pairs). Given the limited number of pairs in this dataset, we create an extended version of it as follows: given a snippet pair related to a topic, we randomly replaced in each snippet two of the five excerpts belonging to it with others from the same politician, on the same topic, and with the same agreement/disagreement. By swapping random pairs of excerpts between snippets, we generated new ones and we were also able to better balance the agreement/disagreement proportion. Overall, the final dataset contains 1,400 pairs, balanced between the agreement/disagreement classes.

We further wanted to measure the impact of our approach on a larger corpus, more suitable for machine learning tasks. Therefore, we extract from Debatepedia[5] pairs of snippets compliant with the structure and the content of the two other datasets. We choose Debatepedia, an online encyclopedia of debates, because it provides statements from two opposing sides debating on well-defined, controversial topics. In particular, for each topic, Debatepedia gathers a set of relevant evidences and statements, mainly from news, that are framed as being in favour or against a specific debate question, e.g. "Is the $700 billion

[5] http://www.debatepedia.org/

bailout for the 2008 US financial crisis a good idea?" (Figure 3). In this way we have a large amount of snippets clustered into topics (e.g. individual rights, public safety, clean energy) and already structured as in agreement or not.

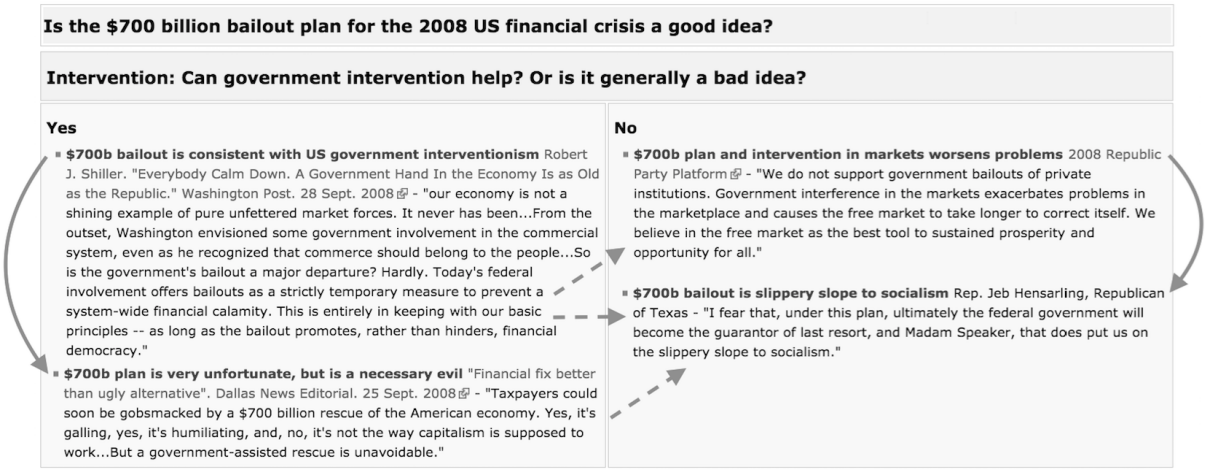


Figure 3: The Debatepedia Structure: the list of statements supporting the topic are on the left side and the list of statements opposing the topic on the right side. The solid arrows connect a pair in agreement, while the dashed lines link pairs in disagreement.

We collect from Debatepedia 646 debates for a total of 17,055 unique statements. To simulate our task, we need to organize the statements into pairs and associate each pair to a topic expressed by a keyword appearing in both statements. The pairs were created as indicated in Figure 3, by coupling two snippets being on the same side (for the agreement cases) or being on two opposite sides of the debate (for the disagreement cases).

In order to identify a keyword shared between two paired snippets, we first select only the pairs sharing at least one noun (74%). Then, we manually revised the 500 nouns most frequently overlapping, keeping only those that are not too generic and are related to the given topic of the debate. This led to a final list of 164 nouns, used as topics, for a total of 29,354 pairs. This final set of pairs, used in the classification task, is still balanced, with 14,042 pairs marked as in agreement and 15,312 in disagreement.

6 Evaluation

We report in Table 2 the results obtained on the three datasets using SVM with radial kernel (10-fold cross validation). A random baseline corresponding to the majority class is also reported. Beside evaluating the classifier performance with all features, we also analysed the contribution of single features (i.e. *negation/overlap*, *entailment*, *sentiment*, *cosine*, *word embeddings*) and their combinations to the task. Finally, we perform another evaluation, adding to the features mentioned before the outcome of the coreference resolution system in StanfordCoreNLP (Lee et al., 2011). Our intuition was that, since our snippets usually include more than one sentence, resolving pronouns and coreferential expressions may make the content more explicit, thus enabling a better agreement detection. However, results show that this information causes a slight performance drop. This can be due to coreference resolution errors rather than the feature itself, and we plan to further investigate this issue in the future.

Evaluation results confirm the findings suggested by the feasibility study: considering only sentiment-based features, or those related to semantic content is not effective as combining the two information layers (*Sent+Entailment+Embeddings+Cosine*). Surface features still have an impact on the outcome of every run, although limited.

If we compare the results obtained on the three datasets, we observe that the limited size of the 1960 Elections dataset affects classification, because some configurations yield the same performance as the random baseline. On the two other datasets, instead, the selected features are effective for classification, with a better performance achieved on the Extended 1960 dataset. This is due to the fact that all snippets

	1960 Elections	Extended 1960 Elections	Debatepedia
Used Features	Accuracy	Accuracy	Accuracy
Random Baseline	67.6%	50.0%	52.2%
Negation+Overlap	67.6%	52.7%	54.5%
Entailment	68.7%	55.7%	55.6%
Sentiment	67.6%	56.2%	54.7%
Cosine	67.6%	67.5%	53.0%
Word Embeddings	76.3%	77.5%	67.3%
Sentiment+Entailment	68.6%	60.0%	57.6%
Sent+Entailment+Embeddings	81.4%	79.7%	72.9%
Sent+Entailment+Embeddings+Cosine	81.7%	79.8%	73.1%
All features	**83.0%**	**80.1%**	**74.0%**
All features + coreference	81.6%	79.0%	73.7%

Table 2: Classification results on the 1960 Elections dataset (300 pairs), Extended 1960 Elections dataset (1,400 pairs) and Debatepedia Dataset (29,354 pairs)

are extracted from documents by Nixon and Kennedy, so that language and style are consistent across all pairs. Debatepedia, instead, relies on a wide range of sources, thus language variation is much higher.

We finally run a last experiment to test our approach on a different task, i.e. the classification of single statements as supporting or opposing a topic. We argue that, even if we chose our features with a focus on *pairwise* comparison, some of them may be effective also for single snippets. We rely again on Debatepedia, with the goal of classifying single arguments as supporting or opposing a topic. To this purpose, we remove the features strictly related to the pairwise comparison (e.g. the lemma overlap or the cosine similarity), and then classify each snippet belonging to one of the 29,354 pairs of our Debatepedia dataset via 10-fold cross-validation. Using SVM, we yield an accuracy of 87.2%. This result shows that the core set of features used to capture the point of view at snippet level are effective both to perform comparisons and to detect support or opposition given a single statement and a topic. Gottipati et al. (2013) perform the same task on Debatepedia data using a topic model-based approach, and achieve 86.0% accuracy. Although the two results are not directly comparable, since their dataset comprising 3,000 snippets is not available, we can conclude that our approach is a reliable solution also for single argument classification and can generalise well over different tasks.

7 Conclusion

In this paper, we introduced a study on the different dimensions which contribute to define agreement and disagreement between points of view in the political domain. We presented a pilot study that highlights how agreement w.r.t. a topic is derived both from the sentiment about it and the solution proposed. We used then these two dimensions, complemented by other lexical features, to train a classifier that was tested on data from the 1960 U.S Presidential Election and from Debatepedia. With this approach, we were able to correctly classify agreement and disagreement with good accuracy.

In addition to SVM, we experimented also with Convolutional Neural Networks using the TensorFlow implementation (Abadi et al., 2015), configured with 10 layers, 100 nodes and 100 iterations. So far, the performance achieved with CNN is around 20% lower than with SVM on all datasets, therefore we did not report the details in the experimental description. We plan to further investigate the motivations behind this gap, and to continue experimenting with other TensorFlow configurations.

Another research direction that we plan to pursue is the role of neutral judgments when analysing agreement and disagreement: we collected 50 neutral judgments during the pilot annotation, but we discarded them because we wanted to focus on pairwise agreement or opposition. However, in a real application scenario, it would be very important to add also this class. This extension is currently ongoing. In the future, we also plan to include the module for agreement and disagreement detection in the platform for the analysis of political speeches presented in Moretti et al. (2016).

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*, 1.

Ahmed Abbasi, Hsinchun Chen, and Arab Salem. 2008. Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums. *ACM Transactions on Information Systems (TOIS)*, 26(3):12.

Rob Abbott, Marilyn Walker, Pranav Anand, Jean E Fox Tree, Robeson Bowmani, and Joseph King. 2011. How can you say such things?!?: Recognizing disagreement in informal political argument. In *Proceedings of the Workshop on Languages in Social Media*, pages 2–11. Association for Computational Linguistics.

Pranav Anand, Marilyn Walker, Rob Abbott, Jean E Fox Tree, Robeson Bowmani, and Michael Minor. 2011. Cats rule and dogs drool!: Classifying stance in online debate. In *Proceedings of the 2nd workshop on computational approaches to subjectivity and sentiment analysis*, pages 1–9. Association for Computational Linguistics.

Rawia Awadallah, Maya Ramanath, and Gerhard Weikum. 2012. Polaricq: Polarity classification of political quotations. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1945–1949. ACM.

Elena Cabrio and Serena Villata. 2012. Combining textual entailment and argumentation theory for supporting online debates interactions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 208–212. Association for Computational Linguistics.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Isaac G Councill, Ryan McDonald, and Leonid Velikovich. 2010. What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the workshop on negation and speculation in natural language processing*, pages 51–59. Association for Computational Linguistics.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Michel Galley, Kathleen McKeown, Julia Hirschberg, and Elizabeth Shriberg. 2004. Identifying agreement and disagreement in conversational speech: Use of bayesian networks to model pragmatic dependencies. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 669. Association for Computational Linguistics.

Swapna Gottipati, Minghui Qiu, Yanchuan Sim, Jing Jiang, and Noah A. Smith. 2013. Learning topics and positions from Debatepedia. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1858–1868, Seattle, Washington, USA, October. Association for Computational Linguistics.

Dustin Hillard, Mari Ostendorf, and Elizabeth Shriberg. 2003. Detection of agreement vs. disagreement in meetings: Training with unlabeled data. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*, pages 34–36. Association for Computational Linguistics.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proceedings of the Association for Computational Linguistics*.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34. Association for Computational Linguistics.

Bernardo Magnini, Roberto Zanoli, Ido Dagan, Kathrin Eichler, Günter Neumann, Tae-Gil Noh, Sebastian Pado, Asher Stern, and Omer Levy. 2014. The excitement open platform for textual inferences. In *ACL (System Demonstrations)*, pages 43–48.

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

Giovanni Moretti, Rachele Sprugnoli, Stefano Menini, and Sara Tonelli. 2016. ALCIDE: Extracting and visualising content from large document collections to support Humanities studies. *Knowledge-Based Systems*, 111:100–112.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 226–234. Association for Computational Linguistics.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124. Association for Computational Linguistics.

Swapna Somasundaran, Galileo Namata, Lise Getoor, and Janyce Wiebe. 2009. Opinion graphs for polarity and discourse classification. In *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing*, pages 66–74. Association for Computational Linguistics.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 327–335. Association for Computational Linguistics.

Peter D Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

Targeted Sentiment to Understand Student Comments

Charles Welch and **Rada Mihalcea**
University of Michigan
2260 Hayward Street
Ann Arbor, MI 48109, USA
{cfwelch,mihalcea}@umich.edu

Abstract

We address the task of targeted sentiment as a means of understanding the sentiment that students hold toward courses and instructors, as expressed by students in their comments. We introduce a new dataset consisting of student comments annotated for targeted sentiment and describe a system that can both identify the courses and instructors mentioned in student comments, as well as label the students' sentiment toward those entities. Through several comparative evaluations, we show that our system outperforms previous work on a similar task.

1 Introduction

Sentiment analysis is the computational study of people's opinions or emotions; it is a challenging problem that is increasingly being used for decision making by individuals and organizations (Pang and Lee, 2008). There is a significant body of research on sentiment analysis, addressing entire documents (Agarwal and Bhattacharyya, 2005), including blogs (Godbole et al., 2007; Annett and Kondrak, 2008) and reviews (Yi et al., 2003; Cabral and Hortacsu, 2010); sentences (Yu and Hatzivassiloglou, 2003; Nigam and Hurst, 2004) or otherwise short spans of texts such as tweets (Pak and Paroubek, 2010; Kouloumpis et al., 2011); and phrases (Wilson et al., 2005; Turney, 2002). More recent work has also addressed the task of aspect sentiment (Pontiki et al., 2015; Thet et al., 2010; Lakkaraju et al., 2014), which aims to address the sentiment toward attributes of the target entity, such as the service in a restaurant (Sauper and Barzilay, 2013), or the camera of a mobile phone (Chamlertwat et al., 2012).

In this paper we address the task of *targeted sentiment*, defined as the task of identifying the sentiment (*positive, negative*) or lack thereof (*neutral*) that a writer holds *toward* entities mentioned in a statement. Targeted sentiment has been only recently introduced as a task, to our knowledge with contributions from only two research groups that focused primarily on settings with scarce resources (Mitchell et al., 2013; Zhang et al., 2015). While previous work on data sets such as product reviews can give an accurate measure of sentiment toward products (as explicit targets of the opinions being expressed in the reviews), some corpora include additional challenges. Targeted sentiment addresses the challenge of identifying entities in running text (e.g., Twitter, student comments), and attributing separate sentiment to each mentioned entity.

In our work, we focus on an application-driven task, namely that of understanding students' sentiment towards courses and instructors as expressed in their comments. As an example, consider the statement:

(1) *I thought that natural language processing with professor Doe was a great class.*

We want to recognize the targets "natural language processing" (a course) and "Doe" (an instructor), as well as a positive sentiment toward the course, and a neutral sentiment toward the instructor. We approach targeted sentiment as a pipeline of two tasks: (1) entity extraction, which aims to identify the entities of interest (in our case, courses and instructors); and (2) entity-centered sentiment analysis, which classifies the sentiment (positive, negative, neutral) held by the student writer toward those entities.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2471–2481, Osaka, Japan, December 11-17 2016.

Section 2 overviews previous work, and shows how our work fits into the bigger picture of sentiment analysis research. Section 3 describes the data used for our experiments. Section 4.1 shows how entities are extracted from text for use in targeted sentiment analysis, and Section 4.2 describes how the sentiment held toward these entities is classified. An overall evaluation of our system and comparison with previous work are presented in Section 5, followed by a discussion and conclusions in Section 6.

2 Previous Work

Most work in sentiment analysis is done at one of three levels: document level, sentence level, and aspect level. These three levels of granularity are ordered from coarsest to finest, with the finer granularity tasks being less well studied. In general, an opinion can be represented by the following quintuple, $(e_i, a_{ij}, oo_{ijkl}, h_k, t_l)$ (Zhang and Liu, 2014). The value e_i here represents the ith entity and a_{ij} represents the aspect j of this entity. The kth holder of the opinion is represented by h_k and the time, l, that the opinion is expressed is given by t_l. Given the entity, aspect, holder, and time, one can reason about an opinion orientation oo_{ijkl}. This is usually a positive, negative, or neutral value, although occasionally a larger number of sentiment values are used (e.g., very positive, very negative).

The work most similar to ours is the open domain targeted sentiment task (Mitchell et al., 2013). Unlike Mitchell et al, we do not use an artificially balanced data set. Instead we collected all the utterances from students who talked about whichever entities they chose. While we do limit the types of entities to only classes or instructors, we do not limit the specific entities themselves and students can talk about any entities that are relevant to their previous educational experience. Our method is also somewhat different in that we do not evaluate subjectivity: all the entities are assigned a positive, negative, or neutral sentiment, and there are no entities without sentiment.

There were two follow-up papers to Mitchell et al. (Zhang et al., 2015; Zhang et al., 2016) (both from the same research group). The first of these papers worked on improving the three models used in Mitchell et al. including the pipeline, joint, and collapsed models. They show some improvements but the pipeline mode, which is most similar to ours, does not greatly differ in performance. The latter paper used different neural network models on a combination of three data sets. Two of these data sets are derived from Twitter (including Mitchell's) and the last is derived from MPQA. We do not attempt to compare to that work, but we show comparable F1 measures using simple linguistic features.

The next most closely related work to ours are the tasks of sentiment slot filling, target dependent sentiment analysis, and aspect-based sentiment analysis. Slot filling is the task of discovering information about a named entity and storing it in a knowledge source (Surdeanu and Ji, 2014). The 2013 Text Analysis Conference (TAC) had two similar tasks, which were slot filling and temporal slot filling (Surdeanu, 2013). For the slot filling task, systems had to determine the correct value for a set of slots for people and organizations. People contained slots such as "date of birth", "age", or "spouse", while organizations contained slots such as "website", "founded by", and "country of headquarters". For the task of temporal slot filling, a system must determine two time ranges. The first time range is a range in which the expressed slot-value was known to begin being a true statement and the second time is when the expressed fact is known to have ceased being true. Sentiment slot filling is the task of taking a query opinion holder and orientation and returning the set of entities that satisfy this condition. In terms of the quintuple we use to represent sentiment, these are related tasks because although slot filling and temporal slot filling are not exactly sentiment tasks, they are concerned with the entity, aspect, and time values. The sentiment slot filling task is concerned with the entity, orientation, and opinion holder.

In the task of target dependent sentiment analysis, the goal is to take a query entity and find the sentiment toward this entity in a set of tweets (Jiang et al., 2011). This is usually done with a small set of entities where the corpus is constructed by querying Twitter for tweets that contain the substring matching the entity. Jiang et al. perform this task in three steps. They first identify whether subjectivity exists, then the polarity of the sentiment toward the target, and then use a graph based method to improve classification accuracy using retweets, i.e., tweets from the same users that mention the same entities. In our task, we use a larger set of entities that have many ways of being mentioned; this makes the entity identification part of the task more difficult. We also do not have a social network structure to leverage

to improve performance.

Aspect-based sentiment analysis has been the focus of recent SemEval tasks as well as a TAC task (Ellis et al., 2014; Pontiki et al., 2014; Pontiki et al., 2015). The 2014 sentiment task was continued in 2015, and again in 2016. Researchers submitted a variety of models to evaluate the sentiment of aspects on sets of reviews for laptops, restaurants, and hotels. The highest scoring systems in the SemEval 2015 Task 12 used maximum entropy and support vector machine (SVM) models with bag of words (BoW), verb and adjective lemmas, bigrams after verbs, negation terms, punctuation, point-wise mutual information scores, part of speech tags, and other features (Li et al., 2013; Zhang and Lan, 2015). The results presented were marked as either constrained or unconstrained systems. Unconstrained systems were allowed to use data outside the training data provided, while constrained systems could not. The top two scoring models were unconstrained but the top scoring constrained system used Brown clusters in addition to other features. These are counts of how many words in the sentence belong to semantic clusters of words derived in previous work (Hamdan et al., 2015). Other entries used similar features with several entries using SVM models and a single entry that relied on an unsupervised model.

3 Data

As we are not aware of any dataset consisting of statements describing courses and instructors, and the sentiment that the writers (students) have toward them, we collected our own dataset. We extracted sentences from a Facebook student group where students describe their experience with classes in the Computer Science department at the University of Michigan, as well as from a survey run with students in the same department. The final data set consists of 1,042 utterances written by both graduates and undergraduates, describing both classes and instructors that the students had/interacted with. Table 1 shows three statement examples drawn from our dataset.

Student utterance	Annotation
I thought that introductory programming concepts was a difficult class and I did not like it.	⟨class name=introductory programming concepts, sentiment=negative⟩
Professor Williams is my favorite teacher that I've had so far.	⟨instructor name=Williams, sentiment=positive⟩
I took CS 203 last Winter. Davis was teaching and I thought the class was excellent.	⟨class name = CS 203, sentiment=positive⟩ ⟨instructor name=Davis, sentiment=neutral⟩

Table 1: Sample student utterances from our dataset along with annotations.

All the utterances were first manually annotated by one of the authors to identify courses and instructors. As often done in entity extraction methods, we identify entities using an I(nside) O(utside) B(eginning) model. For instance, given the text "I am enrolled in CS 445.", and assuming the entity to be extracted is a course name, the annotation would include the following labels "I_O am_O enrolled_O in_O CS_B 445_I.", indicating that CS is at the beginning of the course name, 445 is inside a course name, and all the other tokens are outside the course name.

Classes can be mentioned by department and class ID as in "CS 484," by ID alone as in "484," or by name as in "introduction to artificial intelligence" or "intro to AI." Instructors are mentioned by name, but could be mentioned by first, last, or first and last names. In total, the 1,042 utterances include 976 class mentions and 256 instructor mentions, for a total of 1,232 entities.

The perceived sentiment toward each entity was also manually labeled by one of the authors as either positive, negative, or neutral. When no explicit sentiment is expressed toward an entity, it is assumed to be neutral. If no sentiment is evident from a given utterance, it is assumed to be neutral. Table 1 shows the annotations for the three sample utterances from our dataset.

To calculate inter-annotator agreement for the identification of entities, a second annotator labeled 100 utterances from the data set, containing 1,263 tokens. Of these, 1,067 were mutually labeled as not being part of any entity. Of the remaining 196 tokens, 2% were not in agreement. Including all tokens, agreement was measured as 0.987 using Cohen's kappa. These two percent were two instances where

the human judges disagreed on whether or not a sequence of tokens was a course name (i.e., an entity that needed to be annotated) or simply a course description. For example, in the sentence "I believe that databases are a crucial part of computer science and 520 was interesting," while "databases" is part of the class name, one annotator decided that the word was simply a description of the content of the course and not an entity.

To calculate inter-annotator agreement for sentiment annotations, a second annotator individually labeled all the 1,232 entities. The agreement between the two annotators was measured at 77.7%, which gives a Cohen's kappa of 0.661 considered to be good agreement. Agreement was calculated as the percentage of entities for which both annotators assigned the same label. Of the annotator disagreements, 10.7% were neutral-negative disagreements, 11.2% neutral-positive disagreements, and 0.2% positive-negative.

4 Targeted Sentiment Analysis

We address this task as a pipeline of two steps. We first identify the target entities (i.e., courses and instructors), followed by a classification held by the student writer toward those entities. In the following, we describe and evaluate the method used for each step, and compare the results obtained against the state-of-the-art.

4.1 Entity Extraction

As mentioned before, we use an IOB model to identify entities in the text. We therefore apply a classification process to every token in the input text. For each token, we build a feature vector, using the following features:

Core features. These include the current word, the case and part-of-speech of the current word, the previous two words; features are also derived from the two words neighboring the current word, which are computed the same way as for the current word.

Lexicons. We record the presence/absence of words in two custom lexicons: one consisting of the professor names gathered from the University of Michigan; the second one including all the words used in the names of the classes offered in the Computer Science department at the same university. The lexicon features are generated for the current word as well as each neighboring word.

Professor titles. We use a list of titles, such as "Dr." or "Prof." to assist with the identification of professor names. The list was compiled manually, and consists of 15 tokens. A feature is generated to indicate whether a token belongs to this list or not. 38% of utterances in the corpus contain professor titles.

Sequence. Students often use a subset of the words in a class name to refer to it. The sequence feature is a binary feature that indicates whether the current word is inside a course or an instructor name sequence, where the courses and instructor names are drawn from the two lexicons described above.

Acronym. The acronym feature is another binary feature that indicates if the input token is an acronym of any class or instructor names in the lexicons. It takes the first letters of each of the words in a name and checks to see if the token matches the concatenated string of first letters for an entry in the lexicon. It subsequently checks if the removal of any number of letters, while retaining order, matches the given token. For instance, "AI" and "ITAI" both match "Introduction to Artificial Intelligence".

Nearest entity. Sometimes class or instructor names are misspelled, and for such cases lexicon features may not be effective. We create a feature that checks if the current token has an edit distance less than three to a word in a class or instructor name in the lexicons. If a match is found, the feature is set to a value of "C" (course) or "I" (instructor) respectively. If no token exists, the feature is set to "N".

As a machine learning algorithm, we use a conditional random field, as it has been previously shown to be highly effective for such entity extraction tasks (Zhang and Liu, 2014). We run a set of 67-33 train-test splits using stratified sampling. Table 2 shows the F-measure results obtained by our system, which makes use of all the features described above, for each of the four token types (B and I for courses and instructors). For comparison, we also show the results obtained with a basic setting, when only the core features are used, as well as the results obtained with a state-of-the-art entity extraction system available from the Stanford NLP group (Finkel et al., 2005), which we have retrained using our corpus.

Using our system, we see a statistically significant improvement over the core baseline for all four tokens ($p < 0.01$). We also find a statistically significant improvement over the Stanford system for I_C and B_I ($p < 0.01$) but no significant difference for the other two token types.[1]

System	B_C	I_C	B_I	I_I
Our system	0.945	0.881	0.922	0.901
Baseline (core features)	0.940*	0.849*	0.863*	0.841*
Stanford (Finkel et al., 2005)	0.944	0.848*	0.896*	0.908

Table 2: F-score figures for the identification of I and B tokens, for course (C) and instructors (I), where * indicates a that our system has a statistically significant improvement for the given token ($p < 0.01$)

To gain a better understanding of the role played by each of the features considered, we also perform feature ablation, with results for the individual feature sets shown in Table 3. We also show the base feature set for comparison.

Interestingly, while lexicon features show the greatest improvement, the titles feature does not show any improvement over the base features. It is possible that this feature ends up being subsumed by the neighboring words, included in the base features. The sequence, acronym, and nearest entity features are all based on the provided lexicons so it is not surprising that sequence and nearest entity features work well. Among them, the acronym feature appears to be less useful simply because many class names are not commonly abbreviated. The most frequently abbreviated name is "AI" for "artificial intelligence". Classes are more often referred to by a subset of the words in the class name, which is a case covered by the sequence feature. This is why we see an improvement in I tokens for classes, whereas the instructor I tokens do not show an improvement for these features. There are also fewer I instructor tokens overall in the corpus, which could make it harder to learn the importance of these features.

Since classes can be identified by an ID number (e.g., "490") or by a name (e.g. "Machine Learning") we can examine the B_C token in more detail. If we separate the B_C token into a token for class IDs and a token for class name words, we find that the improvement using the lexicon, sequence, and nearest entity features is statistically significant only for the class name words ($p < 0.01$). There is no statistically significant improvement for the ID tokens by themselves, which is not surprising given that the identification of such IDs (most of the times consisting of numbers) is an easy task.

Features	B_C	I_C	B_I	I_I
Baseline (core features)	0.940	0.849	0.863	0.841
Lexicons	0.944*	0.875*	0.915*	0.896*
Titles	0.940	0.851	0.861	0.839
Sequence	0.945*	0.871*	0.858	0.832
Acronym	0.940	0.848	0.860	0.835
Nearest entity	0.944*	0.865*	0.910*	0.895*

Table 3: Feature ablation for the identification of I and B tokens, for courses (C) and instructors (I). A feature that provides results significantly better than the base feature set is indicated with * ($p < 0.01$)

We also run an entity-based evaluation, where we use the IOB tokens to construct full class and instructor names. This is done by finding the B tokens that have the correct following sequence of I tokens. If any of the B or I tokens are missing, or are of the wrong type, the entity is not counted as correct. Table 4 shows the precision, recall, and F-score obtained by our system for the extraction of instructor and class entities, and compares our results with those obtained with the Stanford entity extraction system.

[1]Throughout this paper, we measure the statistical significance of our results by using a paired t-test with Bonferroni correction using the same 67-33 train-test splits.

	Our system			Stanford (Finkel et al., 2005)		
Set	Precision	Recall	F-score	Precision	Recall	F-score
Instructors	0.833	0.888	0.859	0.711	0.802	0.754
Classes	0.920	0.899	0.910	0.886	0.867	0.876
Both	0.900	0.897	0.900	0.845	0.853	0.849

Table 4: Precision, Recall and F-score measures for the identification of class and instructor entities

4.2 Entity-Centric Sentiment Analysis

Once the entities of interest are identified, the next step is to determine the sentiment held by the writer (student) toward those entities. This is performed as a classification task using three classes: positive, negative, and neutral. For each candidate entity, we build a feature vector using one of the following configurations:

Weighted bag-of-word. The default model is constructed using unigram counts. The first step is to extract a set of the words that exist in the training set. Using this vocabulary set, counts are constructed for every utterance. These counts are weighted based on their distance, in number of tokens, to the target entity in the statement. For each occurrence of each word, the feature is computed by $\sum_{i \in I} 1/d_{ie}$, where I is the set of occurrences of that word and d is the distance (in words) to the target entity e.

Tree weighted n-grams. A sentence is not linear in nature. A sentence contains clauses and phrases that can be grouped into a tree structure. Consider the sentence "I thought that CS 203 was going to be good, but it was awful". In this sentence "CS 203" is the target entity and we find that a positive sentiment word "good" is closer (using linear distance in number of tokens) to the entity than the negative sentiment word "awful," which represents the actual sentiment toward the entity. If we construct a constituency parse tree from this sentence, and calculate the distance as the number of hops between nodes in the tree, then the negative sentiment word is actually closer to the entity word. For each word in an utterance, we calculate this feature as the number of edges in the parse tree between that word and the target entity. For instance, for the example shown in Figure 1, the distance between "awful" and the target entity "203" is six, while the distance between "good" and "203" is eight.

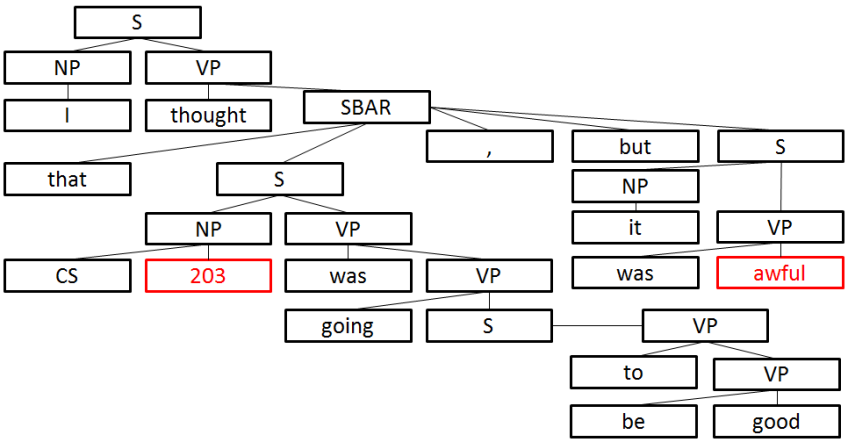


Figure 1: Example sentence, "I thought that CS 203 was going to be good, but it was awful", showing the parse tree weighting for counts using the number of node hops between a given word and the target entity.

Weighted sentiment lexicons. We also implement a feature based on the presence/absence of words from two sentiment lexicons: Bing Liu's lexicon (Hu and Liu, 2004), and the MPQA lexicon (Riloff and Wiebe, 2003) (Wilson et al., 2005). These are two of the most commonly used lexicons in recent sentiment work, and contain 6,789 and 8,222 words respectively, labeled as positive, negative, or neutral. For each word in the utterance, we now generate four features: one simply reflecting the weight of the word (calculated as described before, as a distance to the target entity), and the other three reflecting

whether the word appears as a positive, negative, or neutral word in any of the lexicons; these three latter features are again represented as weighted distance scores.

We use an SVM classifier, with a grid search for the SVM cost and gamma parameters performed using three-fold cross validation on the training set. Training and test splits contained approximately 67% and 33% of the data respectively, stratified as mentioned in Section 4.1, such that the two entity types (instructor or class) and each of the three sentiment class labels are roughly evenly spread across the train, development, and test sets.

Table 5 shows the results obtained with our sentiment analysis system. Note that all the experiments are run on the gold standard set of entities (i.e., manually annotated entities). For comparison, we also report a majority baseline, calculated as the percentage of instances in the entire data set that are neutral, as well as the inter-annotator agreement, as described in Section 3.

We also include the result obtained by using the Stanford sentiment analysis tool (Socher et al., 2013). We do not retrain this model on our own data, as this would require additional node level annotation for the parse tree of each utterance; instead, we use their sentence level sentiment analysis, which assigns an integer score of 0-4 to each sentence, ranging from "very negative" to "very positive". We assign the sentence level scores to each entity contained within that sentences. The five values can be mapped to the three values used in our data set in a number of ways, but the way that maximizes the accuracy over our entire data set maps 0 to our "negative," 1 and 2 to our "neutral," and 3 and 4 to our "positive."

Feature	Accuracy
Our system	69.5%
Majority baseline	52.8%
Stanford (Socher et al., 2013)	62.3%
Annotator agreement	77.7%

Table 5: Sentiment accuracies for our system compared to a majority baseline, the Stanford sentiment analysis tool using recursive neural tensor networks, and the inter-annotator agreement.

For a deeper analysis, Table 6 shows the results obtained by our various features.

Feature	Accuracy
Weighted bag-of-words	67.9%
Tree weighted n-grams	65.6%
Weighted sentiment lexicon	69.5%*

Table 6: Sentiment accuracies of different feature models where * indicates a feature whose difference from the default linear weighted bag-of-words is a statistically significant improvement ($p < 0.01$).

5 Overall Evaluation and Discussion

In the previous section, we described the methods used for each of the two stages of targeted sentiment analysis, along with results obtained at each stage. We now perform an overall evaluation of this task, and compare our system with previous methods for targeted sentiment analysis.

First, we evaluate the correctness of the sentiment at entity level, where an entity is marked as correct only if both the entity and the writer's sentiment toward that entity are correct. Table 7 shows the precision, recall, and F-score obtained for instructor and classes individually, and for all the entities together, assuming: (1) ground truth identification of the entities (i.e., manual annotations); and (2) automatic annotation of the entities using our system from Section 4.1.

Second, we compare the results of our system with previous work by (Mitchell et al., 2013). In that work, the authors use a dataset consisting of 2,350 English tweets containing 3,577 volitional entities, which include PERSON and ORGANIZATION entities. They evaluate the performance of the sentiment on entities by checking only the "B" token from the IOB annotation to see if the associated sentiment is

Entities	Precision	Recall	F-score
Ground Truth Instructors	0.643	0.643	0.643
Ground Truth Classes	0.710	0.710	0.710
Ground Truth Both	0.695	0.695	0.695
Extracted Instructors	0.581	0.578	0.580
Extracted Class	0.571	0.599	0.585
Extracted Both	0.573	0.600	0.586

Table 7: Micro-averaged Precision, Recall, and F-score for full targeted sentiment analysis, for both courses and instructors, using ground truth or automatically identified entities.

correct. If so, it is counted as a true positive. Note that this is less constrained than our evaluation, which also requires that the subsequent I tokens be correct.

In order to allow for a comparison between our system and theirs, we train our pipeline model on their data, by using the same ten-fold cross validation that the previous authors provided. Note that for this comparison, in the entity extraction step of our system we do not use the lexicon, professor title, acronym, sequence, or nearest entity features because of their domain specificity (these features are specifically aimed at finding sentiment toward courses and instructors, and are not expected to be useful on a dataset of general Twitter data). The results of this comparison are shown in Table 8. Mitchell et al. examine targeted sentiment with only volitional entities and do not use "neutral" as a class for targeted sentiment. For these reasons we include the second and third rows in Table 8.

Additionally, because previous work had purposefully not used certain features so that their method could be applied to low resource languages, we also show the performance of the system when we remove the part-of-speech features from our entity extraction step. Note that some of the previous work used accuracy, while other work used F-score; we therefore report both.

We also compare our system to that of (Zhang et al., 2015). Zhang et al. 2015 use a neural network model and report their F-score performance on the same corpus. They perform two evaluations, one that uses only positive/negative sentiment, and one that includes the neutral class. We find that our model is comparable when part-of-speech tags are excluded, but outperform the neural models when they are included.

Method	Accuracy	F-score
Our system	68.3%	0.687
Our system, positive/negative sentiment only	68.6%	0.664
Our system, volitional entities, positive/negative sentiment only	70.8%	0.703
Our system, no part-of-speech features	28.9%	0.393
(Mitchell et al., 2013)	30.8%	NA
(Zhang et al., 2015)	NA	0.401
(Zhang et al., 2015) positive/negative sentiment only	NA	0.279

Table 8: Accuracy and F-score for different versions of our system, as compared to previous work.

Discussion. There are a number of errors that are made by our system. Some of the errors come simply from fully or partially missing entities in the beginning of the pipeline. For instance, we found that the named entity recognition fails on some professor names, mainly because some professors use names other than those listed in the online resources that we used to generate our lexicons. A few other less common errors included recognizing first and last names as separate people, and combining class names listed after each other, e.g. "natural language processing and compilers."

Another batch of errors have correctly recognized entities, but incorrectly classified sentiment. The most common of these cases is incorrectly assigning the neutral class to an entity; the classifier may be somewhat bias toward this class given that it is assigned to 52% of entities in the corpus. Another error

involves having multiple entities in a sentence and assigning the sentiment expressed to the wrong entity. For example, in the sentence "I think that John Smith was an interesting teacher in natural language processing", positive sentiment is incorrectly assigned to "natural language processing." Another type of error comes from unresolved pronouns. In the utterance, "John Smith taught the data mining class that I took. He was an amazing teacher and I wish that he would teach machine learning," "John Smith" is classified as having neutral sentiment, rather than positive; coreference resolution could help if we reweighted the features taking into account the correct pronoun set as entity words.

6 Conclusions

In this paper, we addressed the task of targeted sentiment analysis in the context of understanding the sentiment that students hold toward courses and instructors. We introduced a new annotated dataset, collected from students at the University of Michigan, and proposed new features for the extraction of entities and the classification of the sentiment toward these entities. We performed evaluations of each of the two stages in our pipeline model, and showed that both our entity extraction method and the entity-centric sentiment analysis have performance that is competitive with the state-of-the-art. Moreover, in an overall evaluation of our pipeline, we showed that our system exceeds the performance of the two previously proposed systems for targeted sentiment analysis (Mitchell et al., 2013; Zhang et al., 2015).

Through several feature ablation analyses, we found that lexicon features play an important role in this task, and we plan to further investigate the use of such lexicons in the future, as well as that of more advanced representations of domain-specific knowledge such as knowledge-graphs.

Acknowledgements

This material is based in part upon work supported by IBM under contract 4915012629 and by the Michigan Institute for Data Science. Any opinions, findings, conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of IBM or the Michigan Institute for Data Science.

References

Alekh Agarwal and Pushpak Bhattacharyya. 2005. Sentiment analysis: A new approach for effective use of linguistic knowledge and exploiting similarities in a set of documents to be classified. In *Proceedings of the International Conference on Natural Language Processing (ICON)*.

Michelle Annett and Grzegorz Kondrak. 2008. A comparison of sentiment analysis techniques: Polarizing movie blogs. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 25–35. Springer.

Luis Cabral and Ali Hortacsu. 2010. The dynamics of seller reputation: Evidence from ebay. *The Journal of Industrial Economics*, 58(1):54–78.

Wilas Chamlertwat, Pattarasinee Bhattarakosol, Tippakorn Rungkasiri, and Choochart Haruechaiyasak. 2012. Discovering consumer insight from twitter via sentiment analysis. *J. UCS*, 18(8):973–992.

Joe Ellis, Jeremy Getman, and Stephanie Strassel. 2014. Overview of linguistic resource for the tac kbp 2014 evaluations: Planning, execution, and results. In *Proc. Text Analysis Conference (TAC2014)*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Namrata Godbole, Manja Srinivasaiah, and Steven Skiena. 2007. Large-scale sentiment analysis for news and blogs. *ICWSM*, 7(21):219–222.

Hussam Hamdan, Patrice Bellot, and Frederic Bechet. 2015. Lsislif: Crf and logistic regression for opinion target extraction and sentiment polarity analysis. *SemEval-2015*, pages 753–758.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 151–160. Association for Computational Linguistics.

Efthymios Kouloumpis, Theresa Wilson, and Johanna D Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! *ICWSM*, 11:538–541.

Himabindu Lakkaraju, Richard Socher, and Chris Manning. 2014. Aspect specific sentiment analysis using hierarchical deep learning. *Advances in Neural Information Processing Systems*.

Yan Li, Yichang Zhang, Xin Tong Doyu Li, Jianlong Wang, Naiche Zuo, Ying Wang, Weiran Xu, Guang Chen, and Jun Guo. 2013. Pris at knowledge base population 2013. In *Proc. TAC 2013 Workshop*.

Margaret Mitchell, Jacqueline Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *Conference on Empirical Methods in Natural Language Processing*, pages 1643–1654.

Kamal Nigam and Matthew Hurst. 2004. Towards a robust metric of opinion. In *AAAI spring symposium on exploring attitude and affect in text*, pages 598–603.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, pages 1320–1326.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Maria Pontiki, Haris Papageorgiou, Dimitrios Galanis, Ion Androutsopoulos, John Pavlopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 27–35.

Maria Pontiki, Dimitrios Galanis, Haris Papageogiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Denver, Colorado*.

Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 105–112. Association for Computational Linguistics.

Christina Sauper and Regina Barzilay. 2013. Automatic aggregation by joint modeling of aspects and values. *Journal of Artificial Intelligence Research*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642.

Mihai Surdeanu and Heng Ji. 2014. Overview of the english slot filling track at the tac2014 knowledge base population evaluation. In *Proc. Text Analysis Conference (TAC2014)*.

Mihai Surdeanu. 2013. Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *Proceedings of the Sixth Text Analysis Conference (TAC 2013)*.

Tun Thura Thet, Jin-Cheon Na, and Christopher SG Khoo. 2010. Aspect-based sentiment analysis of movie reviews on discussion boards. *Journal of information science*.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

Jeonghee Yi, Tetsuya Nasukawa, Razvan Bunescu, and Wayne Niblack. 2003. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 427–434. IEEE.

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 129–136. Association for Computational Linguistics.

Zhihua Zhang and Man Lan. 2015. Ecnu: Extracting effective features from multiple sequential sentences for target-dependent sentiment analysis in reviews. *SemEval-2015*, page 736.

Lei Zhang and Bing Liu. 2014. Aspect and entity extraction for opinion mining. In *Data mining and knowledge discovery for big data*, pages 1–40. Springer.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, Arizona, USA. Association for the Advancement of Artificial Intelligence*.

Towards Sub-Word Level Compositions for Sentiment Analysis of Hindi-English Code Mixed Text

Aditya Joshi[1]*, Ameya Prabhu†[2]*, Manish Shrivastava[3] and Vasudeva Varma[1]
[1]Search and Information Extraction Lab
[2]Centre for Visual Information Technology
[3]Language Technologies Research Center
International Institute of Information Technology, Hyderabad (India)
{aditya.joshi, ameya.prabhu} @research.iiit.ac.in
{m.shrivastava, vv} @.iiit.ac.in

Abstract

Sentiment analysis (SA) using code-mixed data from social media has several applications in opinion mining ranging from customer satisfaction to social campaign analysis in multilingual societies. Advances in this area are impeded by the lack of a suitable annotated dataset. We introduce a Hindi-English (Hi-En) code-mixed dataset for sentiment analysis and perform empirical analysis comparing the suitability and performance of various state-of-the-art SA methods in social media.

In this paper, we introduce learning sub-word level representations in LSTM (Subword-LSTM) architecture instead of character-level or word-level representations. This linguistic prior in our architecture enables us to learn the information about sentiment value of important morphemes. This also seems to work well in highly noisy text containing misspellings as shown in our experiments which is demonstrated in morpheme-level feature maps learned by our model. Also, we hypothesize that encoding this linguistic prior in the Subword-LSTM architecture leads to the superior performance. Our system attains accuracy 4-5% greater than traditional approaches on our dataset, and also outperforms the available system for sentiment analysis in Hi-En code-mixed text by 18%.

1 Introduction

Code Mixing is a natural phenomenon of embedding linguistic units such as phrases, words or morphemes of one language into an utterance of another (Muysken, 2000; Duran, 1994; Gysels, 1992). Code-mixing is widely observed in multilingual societies like India, which has 22 official languages most popular of which are Hindi and English. With over 375 million Indian population online, usage of Hindi has been steadily increasing on the internet.

This opens up tremendous potential for research in sentiment and opinion analysis community for studying trends, reviews, events, human behaviour as well as linguistic analysis. Most of the current research works have involved sentiment polarity detection (Feldman, 2013; Liu, 2012; Pang and Lee, 2008) where the aim is to identify whether a given sentence or document is (usually) positive, negative or neutral. Due to availability of large-scale monolingual corpora, resources and widespread use of the language, English has attracted the most attention.

Seminal work in sentiment analysis of Hindi text was done by Joshi et al. (2010) in which the authors built three step fallback model based on classification, machine translation and sentiment lexicons. They also observed that their system performed best with unigram features without stemming. Bakliwal et al. (2012) generated a sentiment lexicon for Hindi and validated the results on translated form of Amazon Product Dataset Blitzer et al. (2007). Das and Bandyopadhyay (2010) created Hindi SentiWordNet, a sentiment lexicon for Hindi.

* indicates these authors contributed equally to this work.
† Corresponding Author

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2482–2491, Osaka, Japan, December 11-17 2016.

Sentence variations
Trailer dhannnsu hai bhai
Dhannnsu trailer hai bhai
Bhai trailer dhannnsu hai
Bhai dhannnsu trailer hai

Table 1: Illustration of free structure present in code mixed text. All sentences convey the same meaning.

Word	Meaning	Appearing Variations
बहुत (bahut)	very	bahout bohut bhout bauhat bohot bahut bhaut bahot bhot
मुबारक (mubaarak)	wishes	mobarak mubarak mubark
प्यार (pyaar)	love	pyaar peyar pyara piyar pyr piyaar pyar

Table 2: Spelling variations of *romanized* words in our Hi-En code-mix dataset.

Sentiment Analysis in Code-mixed languages has recently started gaining interest owing to the rising amount of non-English speaking users. Sharma et al. (2015) segregated Hindi and English words and calculated final sentiment score by lexicon lookup in respective sentient dictionaries.

Hindi-English (Hi-En) code mixing allows ease-of-communication among speakers by providing a much wider variety of phrases and expressions. A common form of code mixing is called as *romanization* [1], which refers to the conversion of writing from a different writing system to the Roman script. But this freedom makes the task for developing NLP tools more difficult, highlighted by (Chittaranjan et al., 2014; Vyas et al., 2014; Barman et al., 2014). Initiatives have been taken by shared tasks (Sequiera et al., 2015; Solorio et al., 2014), however they do not cover the requirements for a sentiment analysis system.

Deep learning based approaches (Zhang and LeCun, 2015; Socher et al., 2013) have been demonstrated to solve various NLP tasks. We believe these can provide solution to code-mixed and romanized text from various demographics in India, as similar trends are followed in many other Indian languages too. dos Santos and Zadrozny (2014) demonstrated applicability of character models for NLP tasks like POS tagging and Named Entity Recognition (dos Santos and Guimarães, 2015). LSTMs have been observed to outperform baselines for language modelling (Kim et al., 2015) and classification (Zhou et al., 2015). In a recent work, (Bojanowski et al., 2016) proposed a skip-gram based model in which each word is represented as a bag of character n-grams. The method produced improved results for languages with large vocabularies and rare words.

The *romanized* code mixed data on social media presents additional inherent challenges such as contractions like "between" → "btwn", non-standard spellings such as "cooolll" or "bhut bdiya" and non-grammatical constructions like "sir hlp plzz naa". Hindi is phonetically typed while English (Roman script) doesn't preserve phonetics in text. Thus, along with diverse sentence construction, words in Hindi can have diverse variations when written online, which leads to large amount of tokens, as illustrated in Table 2. Meanwhile there is a lack of a suitable dataset.

Our contributions in this paper are (i) Creation, annotation and analysis of a Hi-En code-mixed dataset for the sentiment analysis, (ii) Sub-word level representations that lead to better performance of LSTM networks compared to Character level LSTMs (iii) Experimental evaluation for suitability and evaluation of performance of various state-of-the-art techniques for the SA task, (iv) A preliminary investigation of embedding linguistic priors might be encoded for SA task by char-RNN architecture and the relation of architecture with linguistic priors, leading to the superior performance on this task.

Our paper is divided into the following sections:

We begin with an introduction to Code Mixing and romanization in Section 1. We mention the issues with code-mixed data in context of Sentiment Analysis and provides an overview of existing solutions. We then discusses the process of creation of the dataset and its features in Section 2. In Section 3, we introduce Sub-word level representation and explains how they are able to model morphemes along with propagating meaningful information, thus capturing sentiment in a sentence. Then in Section 4, we explain our experimental setup, describe the performance of proposed system and compare it with baselines and other methods, proceeded by a discussion on our results.

[1] https://en.wikipedia.org/wiki/Romanization

2 Dataset

We collected user comments from public Facebook pages popular in India. We chose pages of Salman Khan, a popular Indian actor with massive fan following, and Narendra Modi, the current Prime Minister of India. The pages have 31 million and 34 million facebook user likes respectively. These pages attract large variety of users from all across India and contain lot of comments to the original posts in code-mixed representations in varied sentiment polarities. We manually pre-processed the collected data to remove the comments that were not written in roman script, were longer than 50 words, or were complete English sentences. We also removed the comments that contained more than one sentence, as each sentence might have different sentiment polarity. Then, we proceeded to manual annotation of our dataset. The comments were annotated by two annotators in a 3-level polarity scale - positive, negative or neutral. Only the comments with same polarity marked by both the annotators are considered for the experiments. They agreed on the polarity of 3879 of 4981 (77%) sentences. The Cohen's Kappa coefficient (Cohen, 1960) was found to be 0.64. We studied the reasons for misalignment and found that causes typically were due to difference in perception of sentiments by individuals, different interpretations by them and sarcastic nature of some comments which is common in social media data. The dataset contains 15% negative, 50% neutral and 35% positive comments owing to the nature of conversations in the selected pages.

The dataset exhibits some of the major issues while dealing with code-mixed data like short sentences with unclear grammatical structure. Further, *romanization* of Hindi presents an additional set of complexities due to loss of phonetics and free ordering in sentence constructions as shown in Table 1. This leads to a number of variations of how words can be written. Table 2 contains some of the words with multiple spelling variations in our dataset, which is one of the major challenges to tackle in Hi-En code-mixed data.

Dataset	Size	# Vocab	Social	CM	Sentiment
STS-Test	498	2375	✓		✓
OMD	3238	6211	✓		✓
SemEval'13	13975	35709	✓		✓
IMDB	50000	5000			✓
(Vyas et al., 2014)	381	-	✓	✓	
Ours	3879	7549	✓	✓	✓

Table 3: Comparison with other datasets.

Popular related datasets are listed in Table 3. STS, SemEval, IMDB etc. have been explored for SA tasks but they contain text in English. The dataset used by Vyas et al. (2014) contains Hi-En Code Mixed text but doesn't contain sentiment polarity. We constructed a code mixed dataset with sentiment polarity annotations, and the size is comparable with several datasets. Table 4 shows some examples of sentences from our dataset. Here, we have phrases in Hindi (source language) written in English (target) language.

Example	Approximate Meaning	Sentiment Polarity
Aisa PM naa hua hai aur naa hee hoga	Neither there has been a PM like him, nor there will be	Positive
abe kutte tere se kon baat karega	Who would talk to you, dog?	Negative
Trailer dhannnsu hai bhai	Trailer is awesome, brother.	Positive

Table 4: Examples of Hi-En Code Mixed Comments from the dataset.

Our dataset and code is freely available for download [2] to encourage further exploration in this domain.

[2] https://github.com/DrImpossible/Sub-word-LSTM

3 Learning Compositionality

Our target is to perform sentiment analysis on the above presented dataset. Most commonly used statistical approaches learn word-level feature representations. We start our exploration for suitable algorithms from models having word-based representations.

3.1 Word-level models

Word2Vec(Mikolov et al., 2013) and Word-level RNNs (Word-RNNs) (thang Luong et al., 2013) have substantially contributed to development of new representations and their applications in NLP such as in Summarization (Cao et al., 2015) and Machine Translation (Cho et al., 2014). They are theoretically sound since language consists of inherently arbitrary mappings between ideas and words. Eg: The words person(English) and insaan(Hindi) do not share any priors in their construction and neither do their constructions have any relationship with the semantic concept of a person. Hence, popular approaches consider lexical units to be independent entities. However, operating on the lexical domain draws criticism since the finite vocabulary assumption; which states that models assume language has finite vocabulary but in contrast, people actively learn & understand new words all the time.

Excitingly, our dataset seems suited to validate some of these assumptions. In our dataset, vocabulary sizes are greater than the size of the dataset as shown in Table 3. Studies on similar datasets have shown strong correlation between number of comments and size of vocabulary (Saif et al., 2013). This rules out methods like Word2Vec, N-grams or Word-RNNs which inherently assume a small vocabulary in comparison to the data size. The finite vocabulary generally used to be a good approximation for English, but is no longer valid in our scenario. Due to the high sparsity of words themselves, it is not possible to learn useful word representations. This opens avenues to learn non-lexical representations, the most widely studied being character-level representations, which is discussed in the next section.

3.2 Character-level models

Character-level RNNs (Char-RNNs) have recently become popular, contributing to various tasks like (Kim et al., 2015). They do not have the limitation of vocabulary, hence can freely learn to generate new words. This freedom, in fact, is an issue: Language is composed of lexical units made by combining letters in some specific combinations, i.e. most of the combinations of letters do not make sense. The complexity arises because the mappings between meaning and its construction from characters is arbitrary. Character models may be apriori inappropriate models of language as characters individually do not usually provide semantic information. For example, while " $King - Man + Women = Queen$" is semantically interpretable by a human, "$Cat - C + B = Bat$" lacks any linguistic basis.

But, groups of characters may serve semantic functions. This is illustrated by $Un + Holy = Unholy$ or $Cat + s = Cats$ which is semantically interpretable by a human. Since sub-word level representations can generate meaningful lexical representations and individually carry semantic weight, we believe that sub-word level representations consisting composition of characters might allow generation of new lexical structures and serve as better linguistic units than characters.

3.3 Sub-word level representations

Lexicon based approaches for the SA task (Taboada et al., 2011; Sharma et al., 2015) perform a dictionary look up to obtain an individual score for words in a given sentence and combine these scores to get the sentiment polarity of a sentence. We however want to use intermediate sub-word feature representations learned by the filters during convolution operation. Unlike traditional approaches that add sentiment scores of individual words, we propagate relevant information with LSTM and compute final sentiment of the sentence as illustrated in Figure 1.

Hypothesis: We propose that incorporating sub-word level representations into the design of our models should result in better performance. This would also serve as a test scenario for the broader hypothesis proposed by Dyer et. al. in his impressive ICLR keynote [3] - Incorporating linguistic priors in network architectures lead to better performance of models.

[3] Available at: http://videolectures.net/iclr2016_dyer_model_architecture/

Methodology: We propose a method of generating sub-word level representations through 1-D convolutions on character inputs for a given sentence. Formally, let C be the set of characters and T be an set of input sentences. The sentence $s \in T$ is made up of a sequence of characters $[c_1, ..., c_l]$ where l is length of the input.

Hence, the representation of the input s is given by the matrix $Q \in R^{d \times l}$ where d is the dimensionality of character embedding that corresponding to $[c_1, ..., c_l]$. We perform convolution of Q with a filter $H \in R^{d \times m}$ of length m after which we add a bias and apply a non-linearity to obtain a feature map $f \in R^{l-m+1}$. Thus we can get sub-word level (morpheme-like) feature map. Specifically, the i^{th} element of f is given by:

$$f[i] = g((Q[:, i : i + m - 1] * H) + b) \tag{1}$$

where $Q[:, i : i + m - 1]$ is the matrix of $(i)^{th}$ to $(i + m - 1)^{th}$ character embedding and g corresponds to ReLU non-linearity.
Finally, we pool the maximal responses from p feature representations corresponding to selecting sub-word representations as:

$$y_i = max(f[p * (i : i + p - 1)]) \tag{2}$$

Next, we need to model the relationships between these features $y^i[:]$ in order to find the overall sentiment of the sentence. This is achieved by LSTM(Graves, 2013) which is suited to learning to propagate and 'remember' useful information, finally arriving at a sentiment vector representation from the inputs. We provide f_t as an input to the memory cell at time t. We then compute values of I_t - the input gate, $\tilde{C}_t$ - the candidate value for the state of the memory cell at time t and f_t - the activation of the forget gate, which can be used to compute the information stored in memory cell at time t. With the new state of memory cell C_t, we can compute the output feature representation by:

$$O_t = \sigma(W y_t + U h_(t - 1) + V(C_t + b) \tag{3}$$

$$h_t = O_t tanh(C_t) \tag{4}$$

where W,U and V are weight matrices and b_i are biases. After l steps, h_l represents the relevant information retained from the history. That is then passed to a fully connected layer which calculates the final sentiment polarity as illustrated in the Figure 1.

Figure 2 gives schematic overview of the architecture. We perform extensive experiments to qualitatively and quantitatively validate the above claims as explained in the next section.

4 Experiments

We perform extensive evaluation of various approaches, starting with a suitability study for the nature of approaches that would be able to generalize to this data. We compare our approaches with the state-of-the-art methods which are feasible to generalize on code-mixed data and (Sharma et al., 2015), the current state-of-the-art in Hi-En code-mixed SA task.

4.1 Method Suitability

Following approaches have been used for performing SA tasks in English but do not suit mix code setting:

- Approaches involving NLP tools: RNTN (Socher et al., 2013) etc which involve generation of parse trees which are not available for code mixed text;
- Word Embedding Based Approaches: Word2Vec, Word-RNN may not provide reliable embedding in situations with small amount of highly sparse dataset.
- Surface Feature engineering based approaches: Hashtags, User Mentions, Emoticons etc. may not exist in the data.

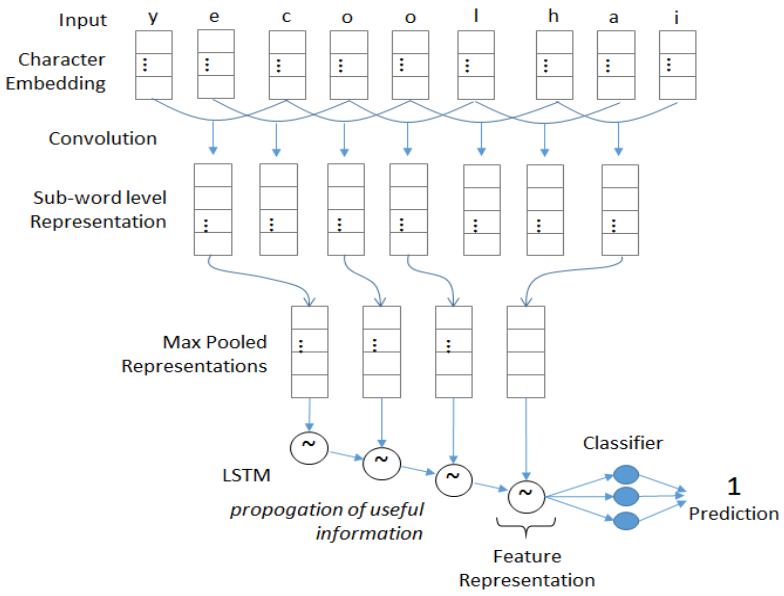


Figure 1: Illustration of the proposed methodology

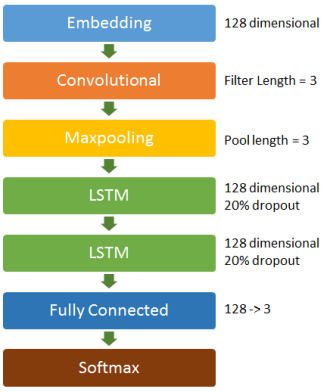


Figure 2: Schematic overview of the architecture.

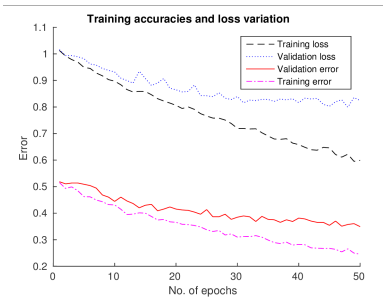


Figure 3: Training accuracy and loss variation.

4.2 Experimental Setup

Our dataset is divided into 3 splits- Training, validation and testing. We first divide the data into randomized 80-20 train test split, then further randomly divide the training data into 80-20 split to get the final training, validation and testing data.

As the problem is relatively new, we compare state of the art sentiment analysis techniques (Wang and Manning, 2012; Pang and Lee, 2008) which are generalizable to our dataset. We also compare the results with system proposed by Sharma et al. (2015) on our dataset. As their system is not available publicly, we implemented it using language identification and transliteration using the tools provided by Bhat et al. (2015) for Hi-En Code Mixed data. The polarity of thus obtained tokens is computed from SentiWordNet (Esuli and Sebastiani, 2006) and Hindi SentiWordNet (Das and Bandyopadhyay, 2010) to obtain the polarity of words, which are then voted to get final polarity of the sentence.

The architecture of the proposed system (Subword-LSTM) is described in Figure 2. We compare it with a character-level LSTM (Char-LSTM) following the same architecture without the convolutional and maxpooling layers. We use Adamax (Kingma and Ba, 2014) (a variant of Adam based on infinity norm) optimizer to train this setup in an end-to-end fashion using batch size of 128. We use very simplistic architectures because of the constraint on the size of the dataset. As the datasets in this domain expand, we would like to scale up our approach to bigger architectures. The stability of training using this architecture is illustrated in Figure 3.

Method	Reported In	Our dataset		SemEval' 13	
		Accuracy	**F1-Score**	**Accuracy**	**F1-Score**
NBSVM (Unigram)	(Wang and Manning, 2012)	59.15%	0.5335	57.89%	0.5369
NBSVM (Uni+Bigram)	(Wang and Manning, 2012)	62.5%	0.5375	51.33%	0.5566
MNB (Unigram)	(Wang and Manning, 2012)	66.75%	0.6143	58.41%	0.4689
MNB (Uni+Bigram)	(Wang and Manning, 2012)	66.36%	0.6046	58.4%	0.469
MNB (Tf-Idf)	(Wang and Manning, 2012)	63.53%	0.4783	57.82%	0.4196
SVM (Unigram)	(Pang and Lee, 2008)	57.6%	0.5232	57.6%	0.5232
SVM (Uni+Bigram)	(Pang and Lee, 2008)	52.96%	0.3773	52.9%	0.3773
Lexicon Lookup	(Sharma et al., 2015)	51.15%	0.252	N/A	N/A
Char-LSTM	Proposed	59.8%	0.511	46.6%	0.332
Subword-LSTM	Proposed	**69.7%**	**0.658**	**60.57%**	0.537

Table 5: Classification results show that the proposed system provides significant improvement over traditional and state of art method for Sentiment Analysis in Code Mixed Text

Comment/*Meaning*	**Transliterated Comment**	**Observation**
Bhai ied mubaraq *Wishes for Eid, brother*	भाई आईद मुबरक	The words ied and mubaraq are spelt differently from usual form which caused incorrect transliteration.
Aatma aur mun ki pavitrata ki jhalak hai is beti ki aawaz mei, khush raho beti *There's a glimpse of piousness of soul and mind in this girl's voice, stay happy, girl!*	आत्मा अर मुन की पवितराता की झालक है इस बेती कि आवाज़ में, खुश रहो बेती	The words aur, pavitrata and beti are written as expected, but still incorrectly transliterated.
love u sir love u soo much I'ts beautyful vedio *Love you so much sir. It's a beautiful video.*	लव उ sir लव उ सू much उर्स इ'ट beautyful vedio	love should have been identified as an English word, which expresses sentiment of the sentence.

Table 6: Output produced a by Hi-En Transliteration Tool

4.3 Observations

In the comparative study performed on our dataset, we observe that Multinomial Naive Bayes performs better than SVM(Pang and Lee, 2008) for snippets providing additional validation to this hypothesis given by Wang and Manning (2012).

We also observe that unigrams perform better than bigrams and Bag of words performs better than tf-idf in contrast to trends in English, as the approaches inducing more sparsity would yield to poorer results because our dataset is inherently very sparse. The lexicon lookup approach (Sharma et al., 2015) didn't perform well owing to the heavily misspelt words in the text, which led to incorrect transliterations as shown in Table 6.

4.4 Validation of proposed hypothesis

We obtain preliminary validation for our hypothesis that incorporating sub-word level features instead of characters would lead to better performance. Our Subword-LSTM system provides an F-score of 0.658 for our dataset, which is significantly better than Char-LSTM which provides F-score of 0.511.

Since we do not have any other dataset in Hi-En code-mixed setting of comparable to other settings, we performed cross-validation of our hypothesis on SemEval'13 Twitter Sentiment Analysis dataset. We took the raw tweets character-by-character as an input for our model from the training set of 7800 tweets and test on the SemEval'13 development set provided containing 1368 tweets. The results are summarized in Table 5. In all the cases, the text was converted to lowercase and tokenized. No extra features or heuristics were used.

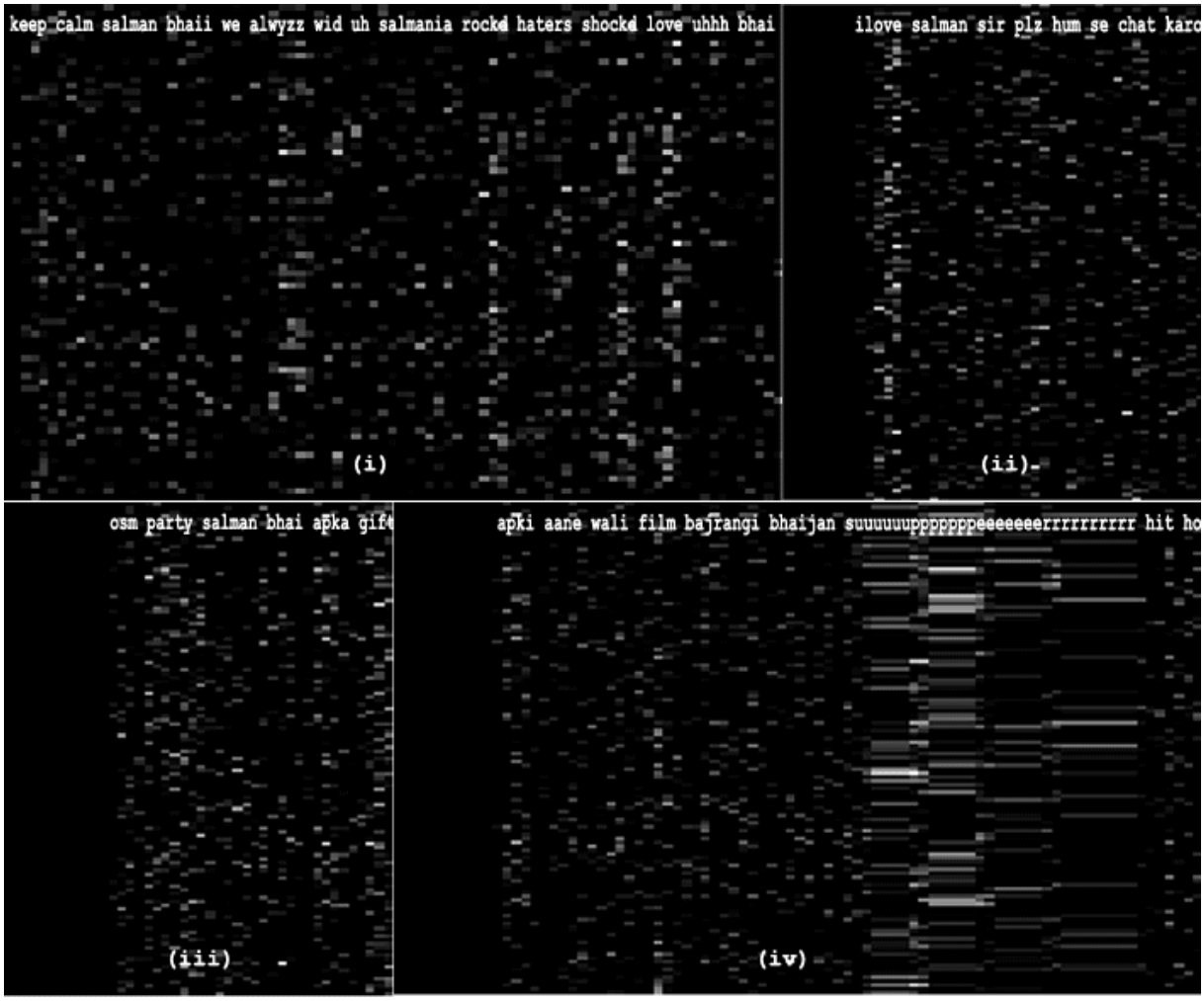


Figure 4: Visualization of the convolution layer for examples comments from the dataset show that word segments convey sentiment information despite being severely misspelt.

4.5 Visualizing character responses

Visualizations in Figure 4 shows how the proposed model is learning to identify sentiment lexicons. We see that different filters generally tend to learn mappings from different parts, interestingly showing shifting trends to the right which maybe due to LSTM picking their feature representation in future time steps. The words sections that convey sentiment polarity information are captured despite misspelling in example (i) and (ii). In example (iii), starting and ending phrases show high response which correspond to the sentiment conveying words (party and gift). The severe morpheme stretching in example (iv) also affects the sentiment polarity.

5 Conclusion

We introduce Sub-Word Long Short Term Memory model to learn sentiments in a noisy Hindi-English Code Mixed dataset. We discuss that due to the unavailability of NLP tools for Hi-En Code Mixed text and noisy nature of such data, several popular methods for Sentiment Analysis are not applicable. The solutions that involve unsupervised word representations would again fail due to sparsity in the dataset. Sub-Word LSTM interprets sentiment based on morpheme-like structures and the results thus produced are significantly better than baselines.

Further work should explore the effect of scaling of RNN and working with larger datasets on the results. In the new system, we would like to explore more deep neural network architectures that are able to capture sentiment in Code Mixed and other varieties of noisy data from the social web.

References

Akshat Bakliwal, Piyush Arora, and Vasudeva Varma. 2012. Hindi subjective lexicon: A lexical resource for hindi polarity classification. In *Proceedings of International Conference on Language Resources and Evaluation*.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code-mixing: A challenge for language identification in the language of social media. In *In Proceedings of the First Workshop on Computational Approaches to Code-Switching*.

Irshad Ahmad Bhat, Vandan Mujadia, Aniruddha Tammewar, Riyaz Ahmad Bhat, and Manish Shrivastava. 2015. Iiit-h system submission for fire2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation*, FIRE '14, pages 48–53, New York, NY, USA. ACM.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *In ACL*, pages 187–205.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015. Ranking with recursive neural networks and its application to multi-document summarization. In *AAAI*, pages 2153–2159.

Gokul Chittaranjan, Yogarshi Vyas, Kalika Bali, and Monojit Choudhury. 2014. Word-level language identification using crf: Code-switching shared task report of msr india system. In *Proceedings of The First Workshop on Computational Approaches to Code Switching*, pages 73–79.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46, April.

Amitava Das and Sivaji Bandyopadhyay. 2010. Sentiwordnet for indian languages. In *Proceedings of the Eighth Workshop on Asian Language Resouces*.

Cícero Nogueira dos Santos and Victor Guimarães. 2015. Boosting named entity recognition with neural character embeddings. *CoRR*, abs/1505.05008.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1818–1826.

Luisia Duran. 1994. Toward a better understanding of code switching and interlanguage in bilinguality: Implications for bilingual instruction. *Journal of Educational Issues of Language Minority Students*, 14:69–87.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC06*, pages 417–422.

Ronen Feldman. 2013. Techniques and applications for sentiment analysis. *Commun. ACM*, 56(4):82–89, April.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Marjolein Gysels. 1992. French in urban lubumbashi swahili: Codeswitching, borrowing, or both. *Journal of Multilingual and Multicultural Development*, 13:41–55.

Aditya Joshi, Balamurali R., and Bhattacharyya Pushpak. 2010. A fall-back strategy for sentiment analysis in hindi: a case study. In *Proceedings of the 8th ICON*, Stroudsburg, PA. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models. *CoRR*, abs/1508.06615.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

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

Pieter Muysken. 2000. *Bilingual Speech: A Typology of Code-mixing*. Cambridge University Press.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, January.

Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. 2013. Evaluation datasets for twitter sentiment analysis: A survey and a new dataset, the sts-gold. *1st Interantional Workshop on Emotion and Sentiment in Social and Expressive Media: Approaches and Perspectives from AI (ESSEM 2013)*.

Royal Sequiera, Monojit Choudhury, Parth Gupta, Paolo Rosso, Shubham Kumar, Somnath Banerjee, Sudip Kumar Naskar, Sivaji Bandyopadhyay, Gokul Chittaranjan, Amitava Das, and Kunal Chakma. 2015. Overview of FIRE-2015 shared task on mixed script information retrieval. In Prasenjit Majumder, Mandar Mitra, Madhulika Agrawal, and Parth Mehta, editors, *Post Proceedings of the Workshops at the 7th Forum for Information Retrieval Evaluation, Gandhinagar, India, December 4-6, 2015.*, volume 1587 of *CEUR Workshop Proceedings*, pages 19–25. CEUR-WS.org.

Shashank Sharma, PYKL Srinivas, and Rakesh Chandra Balabantaray. 2015. Text normalization of code mix and sentiment analysis. In *2015 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2015, Kochi, India, August 10-13, 2015*, pages 1468–1473.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Stroudsburg, PA, October. Association for Computational Linguistics.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Gohneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, and Pascale Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of The First Workshop on Computational Approaches to Code Switching, held in conjunction with EMNLP 2014.*, pages 62–72, Doha, Qatar. ACL.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Comput. Linguist.*, 37(2):267–307, June.

Minh thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*, pages 104–113.

Yogarshi Vyas, Spandana Gella, Jatin, Kalika Bali, and Monojit Choudhury. 2014. Pos tagging of english-hindi code-mixed social media content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 974–979.

Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 90–94, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *CoRR*, abs/1502.01710.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. 2015. A C-LSTM neural network for text classification. *CoRR*, abs/1511.08630.

Distance Metric Learning for Aspect Phrase Grouping

Shufeng Xiong[1,3], **Yue Zhang**[2], **Donghong Ji**[1]*, **Yinxia Lou**[1]
[1]Computer School, Wuhan University, Wuhan, China
[2]Singapore University of Technology and Design, Singapore
[3]Pingdingshan University, Pingdingshan, China
{xsf, dhji}@whu.edu.cn
yue_zhang@sutd.edu.sg

Abstract

Aspect phrase grouping is an important task in aspect-level sentiment analysis. It is a challenging problem due to polysemy and context dependency. We propose an Attention-based Deep Distance Metric Learning (ADDML) method, by considering aspect phrase representation as well as context representation. First, leveraging the characteristics of the review text, we automatically generate aspect phrase sample pairs for distant supervision. Second, we feed word embeddings of aspect phrases and their contexts into an attention-based neural network to learn feature representation of contexts. Both aspect phrase embedding and context embedding are used to learn a deep feature subspace for measure the distances between aspect phrases for K-means clustering. Experiments on four review datasets show that the proposed method outperforms state-of-the-art strong baseline methods.

1 Introduction

For aspect-level sentiment analysis (Hu and Liu, 2004; Pang and Lee, 2008), aspect identification from the corpus is a necessary step. Here *aspect* is the name of a feature of the product, while an *aspect phrase* is a word or phrase that actually appears in a sentence to indicate the aspect. Different aspect phrases can be used to describe the same aspect. For example, "picture quality" could be referred to "photo", "image" and "picture". All aspect phrases in the same group indicate the same aspect. In this paper, we assume that all aspect phrases have been identified by using existing methods (Jin et al., 2009; Kobayashi et al., 2007; Kim and Hovy, 2006), and focus on grouping domain synonymous aspect phrases.

Most existing work employed unsupervised methods, exploiting lexical similarity from semantic dictionary as well as context environments (Zhao et al., 2014; Zhai et al., 2011a; Guo et al., 2009). The context for an aspect phrase is formed by aggregating related sentences that mention the same aspect phrase. Thereafter, aspect phrase and context environment are represented using bag-of-word (BoW) models separately, and integrated into a unified learning framework.

One limitation of the existing methods is that they do not model the interaction between aspect phrases and their contexts explicitly. For example, in the review "the picture is clear, bright and sharp and the sound is good", the words "clear", "bright" and "sharp" are related to the aspect phrase "picture", while the word "good" is related to the aspect phrase "sound". By the traditional model, these words are not differentiated when they are taken for the context, thereby causing noise in the grouping of the two aspect phrases.

To address this issue, we propose a novel neural network structure that automatically learns the relative importance of each context word with respect to a target/aspect phrase, by leveraging an attention model (Luong et al., 2015; Rush et al., 2015; Ling et al., 2015). As shown in Figure 1, given a sentence that contains an aspect phrase, we use a neural network to find a vector representation of the aspect phrase and its context. For the grouping of a certain aspect phrase, we concatenate all the occurrences of the aspect phrase in a corpus to find its vector form. Thus, the problem of aspect phrase grouping is transformed into a clustering problem in the resulting vector space. Different from traditional methods, which leverage

*corresponding author

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2492–2502, Osaka, Japan, December 11-17 2016.

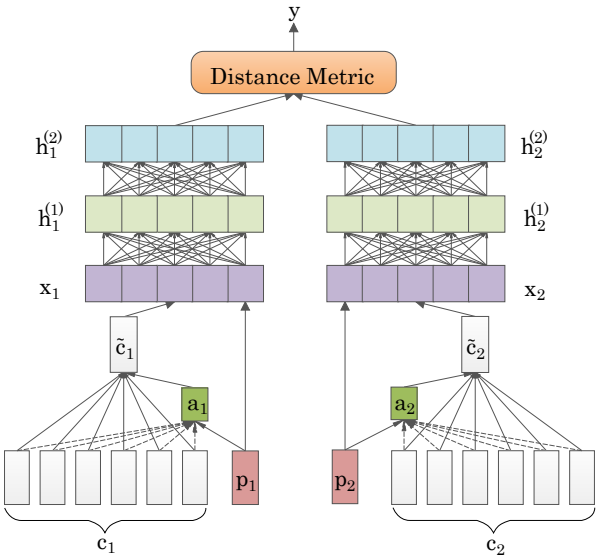


Figure 1: Architecture of the proposed method. For a given pair of aspect phrases p_1 and p_2, with their contexts c_1 and c_2 respectively, two vectors x_1 and x_2 are obtained via attention-based semantic combination, and then mapped into the same feature subspace as $h_1^{(2)}$ and $h_2^{(2)}$.

a bag-of-word feature space, our vector space considers not only words, but also semantic similarities between aspect phrases and contexts (Xu et al., 2015).

One challenge to the success of our method is the finding of a proper training algorithm for the neural network model. Inspired by word embedding training methods (Collobert et al., 2011; Mikolov et al., 2013), we take a negative sampling approach. In particular, we take pairs of sentences that contain the same aspect phrase as positive training examples, and pairs of sentences that contain incompatible aspect phrase as negative training examples, maximizing a score margin between positive and negative examples. Here two aspect phrases are incompatible if the distance based on a semantic lexicon is large (Faruqui et al., 2015; Yu and Dredze, 2014).

To find a better vector space representation, we add two nonlinear transformation layers, as shown in $h^{(1)}$ and $h^{(2)}$ in Figure 1. This method is similar to the Mahalanobis distance metric learning for face verification (Hu et al., 2014). Model training is performed by back-propagation over all neural nodes. With such vector space being learned, direct K-means clustering can be used to group aspect phrases.

Results on a standard benchmark show that our neural network significantly outperforms traditional models. The average results on 4 domains reached 0.51 (Purity) and 1.74 (Entropy), better than the previous best result (0.43 in Purity and 2.02 in Entropy).

2 Method

Our proposed model addresses two main problems: (1) to express fine-grained semantic information with a fixed length vector, which can naturally combine aspect phrases and their contexts, and (2) to provide nonlinear transformations to learn a feature subspace, under which the distance between each intra-group aspect phrase pair is smaller than that between each inter-group pair. We discuss an attention-based semantic composition model in Section 2.1, and then describe a Multi-Layer Perceptron (MLP)-based nonlinear transformation model in Section 2.2, which are designed for (1) and (2), respectively.

2.1 Attention-Based Semantic Composition

The goal of the model is to learn the semantic representation of the context of each aspect phrase. For our task, the same context is frequently shared by more than one aspect phrases. For example, in the sentence: "the **picture** is clear and sharp and the **sound** is good.", two different aspect phrases "picture" and "sound" are mentioned. We use an attention-based neural semantic composition model to consider

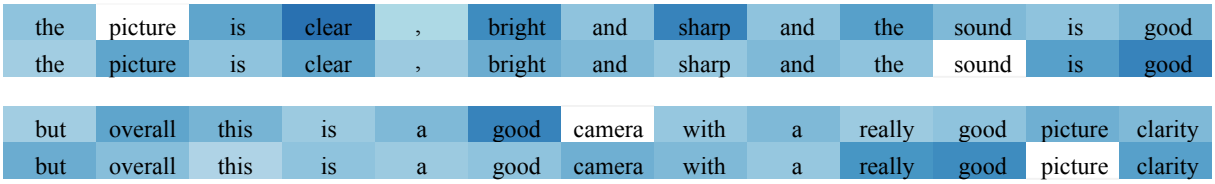


Figure 2: Visualization of weighted context by our attention model. There is a context which contains two aspect phrases (white background), where the weight of each word is taken from the developing dataset. For different aspect phrases, the words in the same context have different weights. Different background represent different weights, bigger weights corresponding to deeper background.

contextual words based on their weight scores with respect to each aspect phrase. In particular, given each word vector e_i, which is projected into a word embedding matrix $L_w \in \mathbb{R}^{d \times |V|}$, where d is the dimension of word vector and $|V|$ is the size of word vocabulary. All of e_i can be randomly initialized from a uniform distribution, and then updated during the back propagation training procedure. Alternatively, another way is using pre-trained vectors as initialization, which is learnt from text corpus with embedding learning algorithms. In our experiment, we adopt the latter strategy. Let $c = \{e_i | e_i \in R^{l \times 1}\}_{i=1,2,...,n}$ denote the set of n input words in context, where l is the dimension of the original context segment. We employ a linear layer to combine the original context vector c and attentional weight a to produce an attentional context representation as:

$$\tilde{c} = f_w(c, a), \tag{1}$$

where f_w is a weighted average function. The idea is to give different weights for different words in the context when deriving the context vector $\tilde{c}$. The weight $a \in \mathbb{R}^{n \times 1}$ is a variable-length attention vector, whose size is equal to the number of words in the context. Its value is computed as follows:

$$a(e_i) = \frac{\exp(score(e_i, p))}{\sum_{i'} \exp(score(e_{i'}, p))}, \tag{2}$$

where $score(e_i, p) = W_a^T [e_i; p]$ and $W_a \in \mathbb{R}^{(2*d) \times 1}$ is a model parameter to learn. Although the length of context is variable, our model uses a fixed-length W_a parameter to weight the importance of each word e_i for its corresponding aspect phrase p. This results in a fixed length vector form for each aspect phrase in a variable-size context.

2.2 MLP-Based Nonlinear Transformation

After obtaining the attention-based context $\tilde{c}$, we employ a MLP-based nonlinear transformation to learn a feature subspace for final aspect phrase grouping. Although $\tilde{c}$ is a weighted context according to the aspect phrase, the aspect phrase p itself is still a necessary source of information for grouping. Therefore, we concatenate $\tilde{c}$ and p to produce a vector x as the input to MLP.

Our model is based on a variant of the Mahalanobis distance metric learning method (Hu et al., 2014). The problem is formulated as follows. Given a training set $X = \{x_i | x_i \in R^{(2*d) \times 1}\}_{i=1,2,...,m}$, where x_i is the ith training sample and m is the size of training set. The method aims to seek a linear transformation W, under which the distance between any two samples x_i and x_j can be computed as:

$$d_w(x_i, x_j) = \|Wx_i - Wx_j\|_2 \tag{3}$$

where W is an alternative of the covariance matrix M in Mahalanobis distance. $\|A\|_2$ represents the $L2$ norm of the matrix A. M can be decomposed by:

$$M = W^T W, \tag{4}$$

Further, Equation (3) can be rewritten as

$$\begin{aligned} d_w(x_i, x_j) &= \|Wx_i - Wx_j\|_2 \\ &= \sqrt{(x_i - x_j)^T W^T W (x_i - x_j)} \\ &= \sqrt{(x_i - x_j)^T M (x_i - x_j)} \end{aligned} \tag{5}$$

Equation (5) is the typical form of Mahalanobis distance between x_i and x_j. Therefore, Equation (3) is both the Euclidean distance of two samples in the linear transformed space, and the Mahalanobis distance in the original space. The transformation Wx can be replaced with a generalized function g. When g is a nonlinear function, we obtain the nonlinear transformation form of Mahalanobis distance. Following Hu et al. (2014), we use the squared Euclidean distance in our model:

$$d_g^2(x_i, x_j) = \|g(x_i) - g(x_j)\|_2^2 \tag{6}$$

As shown in Figure 1, we use hierarchical nonlinear mappings to project the samples to a feature subspace. Assume that there are M layers in the designed network, and $k^{(m)}$ units in the mth layer, where $m = 1, 2, ..., M$. For a given aspect phrase sample x, the output of the first layer is computed as

$$h^{(1)} = f_a(W^{(1)}x + b^{(1)}) \in \mathbb{R}^{k^{(2)}}, \tag{7}$$

where the weight matrix $W^{(1)} \in \mathbb{R}^{k^{(2)} \times k^{(1)}}$ can be seen as a linear projection transformation, $b^{(1)} \in \mathbb{R}^{k^{(2)}}$ is a bias vector, and $f_a : \mathbb{R} \mapsto \mathbb{R}$ is a nonlinear activation function.

Subsequently, the output of the first layer $h^{(1)}$ is used as the input of the second layer. In the same way, the output of the second layer is

$$h^{(2)} = f_a(W^{(2)}h^{(1)} + b^{(2)}) \in \mathbb{R}^{k^{(3)}}, \tag{8}$$

where $W^{(2)} \in \mathbb{R}^{k^{(3)} \times k^{(2)}}$, $b^{(2)} \in \mathbb{R}^{k^{(3)}}$ and f_a are the projection matrix, a bias and a nonlinear activation function of the second layer, respectively.

Finally, the output of the topmost layer is calculated as follows:

$$h^{(M)} = f_a(W^{(M)}h^{(M-1)} + b^{(M)}) \in \mathbb{R}^L, \tag{9}$$

where L is the dimension of the output vector.

Given a pair of aspect phrase samples x_i and x_j, let $g(x_i) = h_i^{(M)}$ and $g(x_j) = h_j^{(M)}$. The function g represents a hierarchical nonlinear transformation, in which the aspect phrase sample pair is passed through the M-layer deep network and mapped into a feature subspace. By using Equation (6), we can measure the distance between the sample pair in the new feature subspace.

2.3 K-means Clustering

With a given corpus, we first employ the parallel deep neural network to learn the semantic representation $h^{(M)}$, and then utilize the K-means algorithm to perform clustering of $h^{(M)}$. During training, we sample aspect phrase pairs using the sentence as context. During testing, we concatenate all the sentences that mention the aspect phrase for clustering the aspect phrase.

2.4 Model Training

The ultimate goal of our model is to make the distance metric effective for grouping aspect phrase samples. To achieve this, we use a large-margin framework to restrict the distance, as proposed by Mignon and Jurie (2012). In particular, sample pairs containing the same aspect phrase are used as positive instances and sample pairs with incompatible aspect phrase are used as negative instances.

We exploit lexical similarity to obtain incompatible aspect phrases, which have low similarity in semantic lexicon. In particular, we choose WordNet as the semantic lexicon. Two aspect phrase are incompatible when the WordNet similarity between them is smaller than a threshold η. And the WordNet similarity is calculated by Equation (12)

$$Res(w_1, w_2) = IC(LCS(w_1, w_2)) \tag{10}$$

$$IC(w) = -logP(w) \tag{11}$$

$$Jcn(w_1, w_2) = \frac{1}{IC(w_1) + IC(w_2) - 2 \times Res(w_1, w_2)} \tag{12}$$

where LCS (lease common subsumer) is the most specific concept that is a shared ancestor of the two concepts represented by the words (Pedersen, 2010). $P(w)$ is the probability of the concept word w. In our experiments, the threshold is set to 0.85.

Traditional methods (Zhai et al., 2010; Zhai et al., 2011b) exploit lexical knowledge to provide soft constraint for clustering aspect phrases. They assume that the aspect phrases that have high similarity in semantic lexicon, are likely to belong to the same group. In this cause, our method uses a similar assumption.

For obtaining the training data, we apply an extra sample pair generation process. The generated sample aspect phrase pairs are fed into left and right sub neural network of Figure 1, respectively. Specifically, each training sentence is utilized with its labelled aspect phrase as a gold sample. Then, we combine each aspect phrase and its related sentences to form training sample set. For example, given an aspect phrase p_1 and a sentence set $S^1 = \{s_1^1, s_2^1, ..., s_m^1\}$, in which there are m sentences that mention p_1, we can construct m samples $\{p_1 \cup s_1^1, p_1 \cup s_2^1, ..., p_1 \cup s_m^1\}$. The group label of the sample is the same as its original aspect phrase, e.g. when p_1 belongs to group 1, then all of $p_1 \cup s_i^1$ have the group label 1.

Assuming that there are n selected aspect phrases, the number of positive sample pairs candidates is $\binom{n}{2}$. A negative sample pair is formed by randomly selecting incompatible aspect phrases and their contexts. For balancing the training set, we obtain the same number of negative sample pairs.

In the training objective, the distance $d_g^2(x_i, x_j)$ of positive instances ($l_{ij} = 1$) is less than a small threshold t_1 and that of negative instance ($l_{ij} = -1$) is higher than a large threshold t_2, where the label l_{ij} denotes the similarity or dissimilarity between a sample pair x_i and x_j, and $t_2 > t_1$. Let $t > 1$, $t_1 = t - 1$ and $t_2 = t + 1$. This constraint can be formulated as follows:

$$l_{ij}(t - d_g^2(x_i, x_j)) > 1, \tag{13}$$

Equation (13) enforces the margin between $d_g^2(x_i, x_j)$.

During the training phase, each aspect phrase pair must satisfy the constraint in Equation (13). Let $\omega = 1 - l_{ij}(t - d_g^2(x_i, x_j))$, we minimize the objective function:

$$J = \frac{1}{2} \sum_{i,j} \sigma(\omega) + \frac{\lambda}{2} \sum_{m=1}^{M} (\left\| W^{(m)} \right\|_F^2 + \left\| b^{(m)} \right\|_2^2) \tag{14}$$

where $\sigma(\omega) = \frac{1}{\beta} \log(1 + \exp(\beta\omega))$ is the generalized logistic loss function (Mignon and Jurie, 2012), which is a smooth approximation of the hinge loss $E(z) = \max(0, z)$. β is the sharpness parameter, λ is a regularization parameter and $\|W\|_F^2$ represents the Frobenius norm of matrix W.

The minimization problem in Equation (14) is solved using a stochastic sub-gradient descent scheme. We train the network using back-propagation. We set the dimension of word vectors as 200, the output length of MLP as 50. The parameters of the linear layer are initialized using normalized initialization (Glorot and Yoshua, 2010). We train a three-layer MLP and employ dropout with 50% rate to the hidden layer. We choose $tanh$ as the activation function. The threshold t, the regularization parameter λ and learning rate μ are empirical set as 3, 0.002 and 0.03 for all experiments, respectively.

3 Experiments

3.1 Data Preparation

We employ the datasets of Xiong and Ji (2015) to evaluate our proposed approach. The datasets are based on *Customer Review Datasets* (CRD) (Hu and Liu, 2004), including four different domains: digital camera (DC), DVD player, MP3 player (MP3) and cell phone (PHONE). We take 3 different random splits of datasets (30% train, 50% test, and 20% development). The statistics are described in Table 1.

	DC	DVD	MP3	PHONE
#Sentences	330	247	581	231
#Aspect Phrases	141	109	183	102
#Aspects	14	10	10	12
#Pairs	19163	11211	64945	8855

Table 1: Statistics of the review corpus. # donotes the size

3.2 Pre-trained Word Vectors

We use *Glove*[1] tools to train word embeddings, and the training parameters are set by following Pennington et al. (2014). Because of the review corpus is too small for learning word embeddings, we use *Amazon Product Review Data* (Jindal and Liu, 2008) as one auxiliary training corpus.

3.3 Evaluation Metrics

Since the problem of aspect phrase grouping is a clustering task, two common measures for clustering are used to its evaluate performance (Zhai et al., 2010) : *Purity* and *Entropy*.

Purity is the percentage of correct clusters that contain only data from the gold-standard partition. A large Purity reflects a better model.

Entropy looks at how various groups of data are distributed within each cluster. A smaller Entropy reflects a better model.

3.4 Baseline Methods and Settings

The proposed ADDML method is compared with a number of methods, which include (1) the state-of-the-art methods, (2) baseline neural methods.

In the first category[2], all methods except Kmeans and CC-Kmeans exploit labelled data, which is generated using sharing word constraint and lexical similarities based on WordNet[3]:

Kmeans. The most popular clustering algorithm based on distributional similarity with *cosine* as similarity measure and BoW as feature representation.

DF-LDA. A combination of Dirichlet Forest Prior and LDA model, in which it can encode domain knowledge (the label) into the prior on topic-word multinomials (Andrzejewski et al., 2009)[4]. The code is available in author's website[5].

L-EM. A state-of-the-art semi-supervised method for clustering aspect phrases (Zhai et al., 2011a). It employed lexical knowledge to provide a initialization for EM. We implemented this method ourselves.

CC-Kmeans. It is proposed by Xiong and Ji (2015), it encodes the capacity limitation as constraint and proposes a capacity constrained K-means to cluster aspect phrases. We use the code from the author[6].

[1] http://nlp.stanford.edu/projects/glove/

[2] Because we use sample pairs but not cluster label for training, it is not possible to train a supervised classifier for testing.

[3] The generation of labelled data follows Zhai et al. (2011a).

[4] There are other LDA based methods for this task, such Constrained LDA (Zhai et al., 2010). Although Constrained LDA used two domains data form CRD dataset as core corpus, they actually crawled many other camera and phone reviews. So we are unable to compare with them by directly using their published results. Since DF-LDA is more effectiveness and suitable for our smaller datasets, we use DF-LDA as the representative of the LDA-based methods.

[5] http://pages.cs.wisc.edu/~andrzeje/research/df_lda.html

[6] https://github.com/pdsujnow/cc-kmeans

The word embedding composite methods employ different composite strategies to form the sample vector, respectively. The clustering method is Kmeans with cosine distance in which word embedding is used as feature vector.

AVG/MIN/MAX+MLP use the average/minimum/maximum value of all the context word vectors in each dimension as the context vector $\tilde{c}$, respectively, and then, concatenates aspect phrase p and $\tilde{c}$ to form the sample vector.

AP only uses aspect phrase (AP) vector to cluster aspect phrases.

Since all the methods based on Kmeans depend on the random initiation, we use the average results of 10 runs as the final result. For L-EM, we use the same parameter settings with the original paper.

3.5 Results

	Purity					Entropy				
	DC	DVD	MP3	PHONE	**avg**	DC	DVD	MP3	PHONE	**avg**
Kmeans	0.4079	0.3922	0.3509	0.3333	0.3711	2.2627	2.0056	2.2862	2.5894	2.2860
DF-LDA	0.4365	0.4362	0.3467	0.4329	0.4132	2.1355	1.9705	2.2054	2.3875	2.1747
L-EM	0.4605	0.4706	0.3333	0.4561	0.4301	2.0451	1.9145	2.2427	1.8952	2.0244
CC-Kmeans	0.4554	0.4483	0.3333	0.4353	0.4181	1.9604	1.9841	2.2897	1.8794	2.0284
AVG	0.5089	0.4483	0.3667	0.4941	0.4545	1.7203	2.1759	2.2030	1.7039	1.9508
MIN	0.4554	0.3218	0.3600	0.4118	0.3872	2.1055	2.5479	2.6598	2.2158	2.3822
MAX	0.4554	0.3563	0.3667	0.4353	0.4034	2.1230	2.4440	2.6036	2.1744	2.3363
AP	0.4196	0.4253	0.3600	0.4588	0.4159	2.1816	2.2074	2.3087	1.9946	2.1731
ADDML	**0.5658**	**0.5098**	**0.3684**	**0.6143**	**0.5146**	**1.7119**	**1.8043**	**2.1274**	**1.3282**	**1.7429**

Table 2: Comparison of Purity and Entropy with baselines. Our model is ADDML.

We present and compare the results of ADDML and the 8 baseline methods based on 4 domains. The results are shown in Table 2, where **avg** represents the average result of the 4 domains. The results are separated into two groups according to categories of the baseline methods. Our approach outperforms baseline methods on the average result of all domains. In addition, we make the following observations:

- From the first group, we can see that L-EM and CC-Kmeans perform better than the other methods. The methods that exploit external knowledge and constraint can achieve better performance. However, the proposed ADDML method outperforms all baselines by using weighted contexts as well as distance metric learning.

- From the second group, all methods employ word embeddings to represent word semantic and text composition semantic. Yet these methods achieve uneven results due to different semantic composition strategies. The neural bag-of-word AVG method performs better than the others in the overall result, in which it averages the semantics of each word in the context. The average operation is a commonly used approach in many neural methods, such as CNN (Convolution Neural Network), and achieves better performance. However, it still falls behind our ADDML method according to its task-independent characteristic.

3.6 Discussion

Case study We manually examined a number of samples, which can be successfully grouped by ADDML but not the baselines. For example, two aspect phrases "*photo quality*" and "*quality*" belong to group "*picture*" and "*build quality*", respectively. Most of the baselines incorrectly clustered them into the same group, while ADDML correctly grouped them. There are two main reasons: (1) the two aspect phrase themselves have similar semantics characteristic and share words, (2) reviewers commonly used similar words to express their opinion. ADDML can learn an exact vector representation that are context sensitive, while the baseline methods can not distinguish similar contexts. Figure 2 shows some example results of attention values.

	Purity						Entropy					
	DC	DVD	MP3	PHONE	**avg**	↑	DC	DVD	MP3	PHONE	**avg**	↑
AP	0.4196	0.4253	0.3600	0.4588	0.4159		2.1816	2.2074	2.3087	1.9946	2.1731	
$cnn + ml$	0.4673	0.4609	0.3003	0.4947	0.4308	3.6%	1.8984	1.9515	2.2908	1.6294	1.9433	10.6%
$atn + ml$	0.4605	0.4706	0.3070	0.5088	0.4367	5.0%	1.8477	1.8158	2.2662	1.5179	1.8619	14.3%
$cnn + mlp + ml$	0.5526	0.4118	0.3246	0.6140	0.4757	14.4%	1.8494	1.8980	2.1626	1.3336	1.8109	16.7%
$atn + mlp + ml$(ADDML)	**0.5658**	**0.5098**	**0.3684**	**0.6143**	**0.5146**	23.7%	**1.7119**	**1.8043**	**2.1274**	**1.3282**	**1.7429**	19.8%

Table 3: The result of different module combinations.

Module Analysis ADDML has three modules: attention-based semantic composition module (atn), MLP-based nonlinear transformation module (mlp) and metric learning (ml). For studying the contribution of each module, we introduce a general convolution neural network (cnn) as an alternative to atn. cnn is a state-of-the-art neural network method for modelling semantic representation of sentence (Kalchbrenner et al., 2014; Tang et al., 2015), which extracts N-gram features by convolution operations.

Table 3 reports the results of different module combinations. We use AP, which only uses aspect phrase vector for clustering, as a reference. The symbol ↑ denotes average percentage improvement than AP in 4 domains. By considering context, $cnn + ml$ and $atn + ml$ achieved better results than AP, which only uses aspect phrase embeddings. After adding nonlinear transformation module, $cnn + mlp + ml$ and $atn + mlp + ml$ further improve the performance. Under the same condition, atn is superior to cnn for our task.

Naturally, aspect phrases in some domains, such as MP3, may have fixed meanings. As a result, an aspect phrase and its context have less correlation under the grouping task in these domains. Therefore, AP achieves a little better result than the other methods except for ADDML. Overall, atn solves the context representation and $mlp + ml$ provides a better metric learning ability for our model.

Similarity Threshold Different similarity thresholds η results in different negative sample pairs, and have a certain impact on performance of our model. For obtaining a better threshold, we performed experiments on developing data with different similarity value. Figure 3 presents the result on DC dataset. The performance slowly decrease with the growth of threshold, which is in line with intuitively understanding. For obtaining enough negative samples, we chosen 0.3 as the similarity threshold.

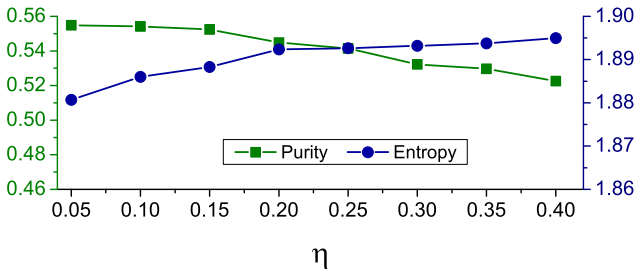


Figure 3: Influence of the similarity threshold η

4 Related Work

Our work is related to aspect-level sentiment analysis, metric learning, and deep learning.

For **aspect-level sentiment analysis**, there are many methods on clustering aspect phrases. Some topic-model-based approaches jointly extract aspect phrases and group them at the same time (Chen et al., 2013; Moghaddam and Ester, 2012; Lu et al., 2011; Jo and Oh, 2011; Zhao et al., 2010; Lin and He, 2009). Those methods tend to discover coarse-grained and grouped aspect phrases, but not specific opinionated aspect phrase themselves. In addition, Zhai et al. (2011a) showed that they did not perform well even considering pre-existing knowledge. Some other work focuses on grouping aspect phrases. Guo et al. (2009) grouped aspect phrases using multi-level LaSA, which exploits the virtual

context documents and semantic structure of aspect phrase. Zhai et al. (2010) used an EM-based semi-supervised learning method for clustering aspect phrases, in which the lexical knowledge is used to provide better initialization for EM. Zhao et al. (2014) proposed a framework of Posterior Regularization to cluster aspect phrases, which formalizes sentiment distribution consistency as a soft constraint. This method requests special semi-structured reviews to estimate the sentiment distribution. In contrast to these methods, we provide a Siamese neural network to learning feature representation through distance supervising.

Metric learning algorithms have been successfully applied to address the problem of face verification (Ding et al., 2015; Yi et al., 2014; Cai et al., 2012; Hu et al., 2014). A common objective of these methods is to learn a better distance metric so that the distance between a positive pair is smaller than the distance between a negative pair. However, these methods not perform nonlinear transformation. Hu et al. (2014) employed a MLP-based nonlinear transformation, but its input is the given image descriptor, which can be directly concatenated to form feature vectors. In this paper, we adapt this method to the aspect phrase grouping task, and provide an extra attention-based semantic composite model to obtain feature vectors based on word vectors of aspect phrase and its context.

Our work is related to **word embedding** and **deep learning**. Prior research (Collobert and Weston, 2008; Mnih and Hinton, 2007; Mikolov et al., 2013; Tang et al., 2014; Ren et al., 2016b) presented different models to improve the performance of word embedding training, and our training is inspried by negative sampling. Deep learning methods (Kalchbrenner et al., 2014; Kim, 2014; Socher et al., 2013; Vo and Zhang, 2015; Zhang et al., 2015; Zhang et al., 2016; Ren et al., 2016a) have been applied to many tasks related to sentiment analysis. In this paper, we explore attention (Luong et al., 2015; Rush et al., 2015; Ling et al., 2015) with a MLP network to tackle the aspect phrase grouping problem.

5 Conclusion

We studied distance metric learning for aspect phrase grouping, exploring a novel deep neural network framework. By leveraging semantic relations between aspect phrase and their contexts, our approach give better performance to strong baselines which achieve the best results in standard benchmark. Our method can be applied to other NLP applications, such as short text clustering and sentence similarity measures.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (No. 61173062, 61373108, 61133012), the major program of the National Social Science Foundation of China (No. 11&ZD189), the key project of Natural Science Foundation of Hubei Province, China (No. 2012FFA088), the Educational Commission of Henan Province, China (No. 17A520050), the High Performance Computing Center of Computer School, Wuhan University and T2MOE201301 from Singapore Ministry of Education.

References

D. Andrzejewski, X. Zhu, and M. Craven. 2009. Incorporating domain knowledge into topic modeling via dirichlet forest priors. *Proc Int Conf Mach Learn*, 382(26):25–32.

Xinyuan Cai, Chunheng Wang, Baihua Xiao, Xue Chen, and Ji Zhou. 2012. Deep nonlinear metric learning with independent subspace analysis for face verification. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, pages 749–752, New York, NY, USA.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Mal Castellanos, and Riddhiman Ghosh. 2013. Exploiting domain knowledge in aspect extraction. In *EMNLP*, EMNLP, pages 1655–1667. ACL.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, ICML, pages 160–167, New York, NY, USA.

Ronan Collobert, Jason Weston, L. E. On Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Shengyong Ding, Liang Lin, Guangrun Wang, and Hongyang Chao. 2015. Deep feature learning with relative distance comparison for person re-identification. *Pattern Recognition*.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*, NAACL.

Xavier Glorot and Bengio Yoshua. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, International conference on artificial intelligence and statistics, pages 249–256.

Honglei Guo, Huijia Zhu, Zhili Guo, XiaoXun Zhang, and Zhong Su. 2009. Product feature categorization with multilevel latent semantic association. In *CIKM*, CIKM, pages 1087–1096. ACM.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. KDD, pages 168–177. ACM.

Junlin Hu, Jiwen Lu, and Yap-Peng Tan. 2014. Discriminative deep metric learning for face verification in the wild. In *cvpr*, cvpr, pages 1875 – 1882. IEEE.

Wei Jin, Hung Hay Ho, and Rohini K. Srihari. 2009. Opinionminer: A novel machine learning system for web opinion mining and extraction. In *KDD '09*, KDD '09, pages 1195–1204, New York, NY, USA. ACM.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *WSDM*, WSDM, pages 219–230. ACM.

Yohan Jo and Alice H. Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of WSDM*, WSDM '11, pages 815–824, New York, NY, USA. ACM.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*, ACL.

Soo-Min Kim and Eduard Hovy. 2006. Extracting opinions, opinion holders, and topics expressed in online news media text. In *Proc. ACL Workshop on Sentiment and Subjectivity in Text*, Proc. ACL Workshop on Sentiment and Subjectivity in Text, pages 1–8, Sydney, Australia. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, EMNLP, pages 1746–1751, Doha, Qatar.

Nozomi Kobayashi, Kentaro Inui, and Yuji Matsumoto. 2007. Extracting aspect-evaluation and aspect-of relations in opinion mining. pages 1065–1074, Prague, Czech Republic. Association for Computational Linguistics.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of CIKM*, CIKM, pages 375–384, Hong Kong, China. ACM. 1646003.

Wang Ling, Yulia Tsvetkov, Silvio Amir, Ramon Fermandez, Chris Dyer, Alan W. Black, Isabel Trancoso, and Chu-Cheng Lin. 2015. Not all contexts are created equal: Better word representations with variable attention. In *Proceedings of EMNLP*, EMNLP, pages 1367–1372, Lisbon, Portugal.

Bin Lu, M. Ott, C. Cardie, and B. K. Tsou. 2011. Multi-aspect sentiment analysis with topic models. In *Proc. ICDMW*, Proc. ICDMW, pages 81 – 88. IEEE.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 1412–1421, Lisbon, Portugal.

A. Mignon and F. Jurie. 2012. Pcca: A new approach for distance learning from sparse pairwise constraints. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2666–2672.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. NIPS*, Proc. NIPS, pages 3111–3119. Curran Associates, Inc.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *ICML*, ICML, pages 641–648.

Samaneh Moghaddam and Martin Ester. 2012. On the design of lda models for aspect-based opinion mining. In *Proceedings of CIKM*, CIKM, pages 803–812. ACM. 2396863.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Ted Pedersen. 2010. Information content measures of semantic similarity perform better without sense-tagged text. In *HLT '10*, HLT '10, pages 329–332, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. EMNLP*, Proc. EMNLP, pages 1532–1543. Association for Computational Linguistics.

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016a. Context-sensitive twitter sentiment classification using neural network.

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016b. Improving twitter sentiment classification using topic-enriched multi-prototype word embeddings.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 379–389, Lisbon, Portugal.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, page 1642. Citeseer.

Duyu Tang, Furu Wei, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*, ACL.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of ACL*, pages 1014–1023, Beijing, China.

Duy-Tin Vo and Yue Zhang. 2015. Deep learning for event-driven stock prediction. In *Proceedings of IJCAI*, IJCAI, BueNos Aires, Argentina.

Shufeng Xiong and Donghong Ji. 2015. Exploiting capacity-constrained k-means clustering for aspect-phrase grouping. In Songmao Zhang, Martin Wirsing, and Zili Zhang, editors, *Knowledge Science, Engineering and Management*, volume 9403 of *Knowledge Science, Engineering and Management*, pages 370–381. Springer International Publishing.

Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. 2015. Short text clustering via convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 62–69, Denver, Colorado.

Dong Yi, Zhen Lei, and Stan Z. Li. 2014. Deep metric learning for practical person re-identification. *arXiv preprint arXiv:1407.4979*.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of ACL*, pages 545–550, Baltimore, Maryland.

Zhongwu Zhai, Bing Liu, Hua Xu, and Peifa Jia. 2010. Grouping product features using semi-supervised learning with soft-constraints. In *Proc. COLING*, Proc. COLING, pages 1272–1280.

Zhongwu Zhai, Bing Liu, Hua Xu, and Peifa Jia. 2011a. Clustering product features for opinion mining. In *Proc. WSDM*, Proc. WSDM, pages 347–354. ACM. 1935884.

Zhongwu Zhai, Bing Liu, Hua Xu, and Peifa Jia. 2011b. Constrained lda for grouping product features in opinion mining. In *Proc. PAKDD*, Proc. PAKDD, pages 448–459.

Meishan Zhang, Yue Zhang, and Duy Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 612–621, Lisbon, Portugal.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *AAAI*, AAAI.

Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a maxent-lda hybrid. In *Proceedings of EMNLP*, EMNLP, pages 56–65. Association for Computational Linguistics. 1870664.

Li Zhao, Minlie Huang, Haiqiang Chen, Junjun Cheng, and Xiaoyan Zhu. 2014. Clustering aspect-related phrases by leveraging sentiment distribution consistency. In *Proc. EMNLP*, Proc. EMNLP, pages 1614–1623. Association for Computational Linguistics.

Constraint-Based Question Answering with Knowledge Graph

Junwei Bao[†*], Nan Duan[‡] , Zhao Yan[♯] , Ming Zhou[‡] , Tiejun Zhao[†]
[†]Harbin Institute of Technology, Harbin, China
[‡]Microsoft Research, Beijing, China
[♯]Beihang University, Beijing, China
[†]baojunwei001@gmail.com
[‡]{nanduan, mingzhou}@microsoft.com
[♯]yanzhao@buaa.edu.cn
[†]tjzhao@hit.edu.cn

Abstract

WebQuestions and *SimpleQuestions* are two benchmark data-sets commonly used in recent knowledge-based question answering (KBQA) work. Most questions in them are 'simple' questions which can be answered based on a single relation in the knowledge base. Such data-sets lack the capability of evaluating KBQA systems on complicated questions. Motivated by this issue, we release a new data-set, namely *ComplexQuestions*[1], aiming to measure the quality of KBQA systems on 'multi-constraint' questions which require multiple knowledge base relations to get the answer. Beside, we propose a novel systematic KBQA approach to solve multi-constraint questions. Compared to state-of-the-art methods, our approach not only obtains comparable results on the two existing benchmark data-sets, but also achieves significant improvements on the *ComplexQuestions*.

1 Introduction

Knowledge-based question answering is a task that aims to answer natural language questions based on existing knowledge bases (KB). In the last decades, large scale knowledge bases, such as Freebase (Bollacker et al., 2008), have been constructed. Based on Freebase, two benchmark data-sets, *WebQuestions* (Berant et al., 2013) and *SimpleQuestions* (Bordes et al., 2015) are constructed and used in most of KBQA work (Berant and Liang, 2014; Bordes et al., 2014a; Fader et al., 2014; Yang et al., 2014; Bao et al., 2014; Reddy et al., 2014; Dong et al., 2015; Yih et al., 2015).

However, about 85% of questions (Yao, 2015) of *WebQuestions* and all questions in *SimpleQuestions* are 'simple' questions, where a 'simple' question denotes that it can be answered based on a single KB relation. For example in Figure 1, "Which films star by Forest Whitaker" is a simple question that can be answered by the KB triples like ⟨Forest Whitaker,acted_films,?⟩ with a single KB relation acted_films. This leads to the fact that such data-sets cannot measure the capability of KBQA systems on 'multi-constraint' questions, where 'multi-constraint' means a question containing multiple semantic constraints expressed with different expressions to restrict the answer set. To answer a multi-constraint question, we have to base on multiple KB relations. For example in Figure 1, "Which films star by Forest Whitaker and are directed by Mark Rydell" is a multi-constraint question with a constraint "directed by Mark Rydell", which requires multiple

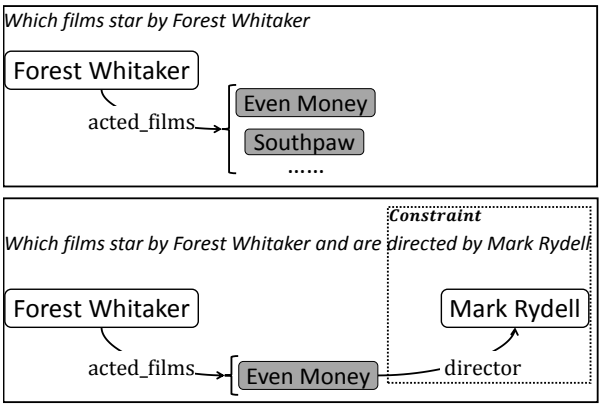


Figure 1: Simple and multi-constraint questions.

This work was finished while the author was visiting Microsoft Research Asia.

[1]https://github.com/JunweiBao/MulCQA/tree/ComplexQuestions

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2503–2514, Osaka, Japan, December 11-17 2016.

Constraint Category	Example	Percentage
Multi-Entity	which films star by **Forest Whitaker** and are directed by **Mark Rydell**?	30.6%
Type	which **city** did Bill Clinton born?	38.8%
Explicit Temporal	who is the governor of Kentucky **2012**?	10.4%
Implicit Temporal	who is the us president **when the Civil War started**?	3.5%
Ordinal	what is the **second longest** river in China?	5.1%
Aggregation	**how many** children does bill gates have?	1.2%

Table 1: Constraint categories, examples, and distributions over a set of web queries. Note, a multi-constraint question may belong to not only one constraint category. And there are still other relatively low frequency type of complex questions existing which we don't take consideration in this work. So the sum of the Percentage for these constraints are not guaranteed to be 1.

KB triples ⟨`Forest Whitaker,acted_films,Even Money`⟩ and ⟨`Even Money,director,Mark Rydell`⟩ with two KB relations `acted_films` and `director` to get the exact answer set.

Motivated by this issue, this work contributes to QA research in the following two aspects: (1) We propose a novel systematic KBQA approach to solve multi-constraint questions by translating a **mul**ti-**c**onstraint **q**uestion (MulCQ) to a **mul**ti-**c**onstraint query **g**raph (MulCG); (2) A new QA data-set, namely *ComplexQuestions*, is released, aiming to measure the quality of KBQA systems on multi-constraint questions. Compared to state-of-the-art approaches, our method obtains comparable results on the existing benchmark data-sets *WebQuestions* and *SimpleQuestions*. Furthermore, we achieve significant improvement on the newly created *ComplexQuestions* data-set.

2 Multi-Constraint Question

2.1 Constraint Classification

A MulCQ is defined as a question which requires multiple KB relations or special operations to get the answer. Based on web query analysis, we classify constraints into 6 categories as follows:

(1) *Multi-entity constraint*. A question in this category denotes that multiple entities occur in it, which restrict the answer. For example, "Forest Whitaker" and "Mark Rydell" are two entity constraints in the first question in Table 1.

(2) *Type constraint*. A question in this category denotes that its answer should follow a type, which is explicitly mentioned by the question. For example, the answer to the second question in Table 1 should be a "city" name, instead of locations with other types such as country, town, etc.

(3) *Explicit temporal constraint*. A question in this category denotes that it contains explicit temporal expressions, such as "2012" in the third question in Table 1. Such questions are very common in web queries, which means handling them well will bring about significant improvements.

(4) *Implicit temporal constraint*. A question in this category denotes that it contains implicit temporal expressions. For example, "when the Civil War started" denotes an implicit temporal constraint in the fourth question in Table 1. We should transform it into an explicit temporal constraint before answering the question. Such constraints are usually expressed by subordinate clauses.

(5) *Ordinal constraint*. A question in this category denotes that its answer should be selected from a ranked set, based on ordinal numbers or superlative phrases as ranking criteria. For example, "second longest" in the fifth question in Table 1 denotes that the answer should be the second item in the ranked Chinese river set, based on their lengths.

(6) *Aggregation constraint*. A question in this category denotes that it asks for the number of a set, which often starts with phrases "how many" or contains "number of", "count of", etc.

2.2 Question Selection

We perform the following steps to select suitable multi-constraint question candidates for human annotators to label, based on Freebase.

Firstly, a three month (2015.1.1-2015.4.1) query log from a practical search engine is used as the raw

Operation	Trigger	Description
$Equal(y_0, y_1)$	N/A	Return `True` if y_0 is equal to y_1, otherwise `False`
$<(y_0, y_1)$	"after" or "later then"	Return `True` if y_0 is smaller than y_0, otherwise `False`
$>(y_0, y_1)$	"before" or "earlier then"	Return `True` if y_0 is larger than y_0, otherwise `False`
`MaxAtN`(x, n)	Maximize superlatives in WordNet	Rank items of x in descending order, return the n^{th} one
`MinAtN`(x, n)	Minimize superlatives in WordNet	Rank items of x in ascending order, return the n^{th} one
`Count`(x)	"how many", "count of" or "number of"	Return the number of entity set x.

Table 2: Functional predicates defined in this work.

query set, which contains 20,999,951 distinct 5W1H questions[2] that satisfy the following two rules: (i) each query should not contain pronouns (e.g., 'you', 'my', etc.), as questions with such words are usually non-factual questions, and (ii) each query's length is between 7 and 20, as short queries seldom contain multi-constraints, and long queries are usually difficult to answer. Then, we further sample 10 percent of questions, and use an entity linking method (Yang and Chang, 2015) to detect entities. If no entity can be detected from a query, we simply remove it. Next, both *WebQuestions* and *SimpleQuestions* are used to extract a set of words, without considering stop words and entity words. If a query does not contain any word in this word set, we simply remove it. This is intuitive, as *WebQuestions* and *SimpleQuestions* are our training data, and we only consider queries that can be covered by the training data as query candidates. Last, we classify the remaining queries based on the following rules:

(1) If a question contains at least two non-overlap entities, then it belongs in the Multi-Entity category;

(2) If a question contains a type phrase that comes from Freebase, then it belongs in the Type category;

(3) If a question contains a time expression detected by an Named Entity Recognizer (NER) (Finkel et al., 2005), then it belongs in the Explicit Temporal category;

(4) If a question contains keywords "when", "before", "after" and "during" in the middle, then it belongs in the Implicit Temporal category;

(5) If a question contains ordinal number or superlative phrase from WordNet (Miller, 1995), then it belongs in the Ordinal category;

(6) If a question starts with "how many", or includes "number of" or "count of", then it belongs in the Aggregation category.

Note, a multi-constraint question may contain multiple types of constraints. We show constraint types, examples, and distributions in Table 1. Ten thousand questions from the above 6 categories are selected, according to their distributions. By manually labeling these questions according to Freebase, we obtain 878 multi-constraint question answer pairs.

2.3 Question Annotation

We release the *ComplexQuestions* data-set, which consists of 2100 multi-constraint question answer pairs coming from 3 sources:

(1) 596 QA pairs selected from *WebQuestions* training set, and 326 from the test set,

(2) 300 QA pairs released by (Yin et al., 2015),

(3) 878 manually labeled QA pairs based on Section 2.2.

We then split it into two parts: a training set containing 1300 QA pairs and a test set including 800 QA pairs[3].

3 Definition

3.1 Knowledge base

$\mathcal{K}$ denotes a knowledge base[4] (KB) that stores a set of facts. Each fact $t \in \mathcal{K}$ is a triple $\langle s, p, o \rangle$, where p represents a predicate (e.g., `birthday`), and s, o (e.g., `BarackObama`, `1961`) represent an entity or a value,

[2] 5W1H questions are ones the start with "what", "where", "when", "who", "which" or "how".

[3] We put QA pairs from the training (testing) set of *WebQuestions* still in the training (testing) set of *ComplexQuestions*, and the same for the test part.

[4] In this work, we use Freebase, which is a large knowledge base with more than 46 million entities and 2.6 billion facts. In Freebase setting, CVT, namely *compound value type* is a special entity category, which is not a real world entity, but is used to collect multiple fields of an event.

which are the subject and object of t.

3.2 Multi-Constraint Query Graph

Before introducing multi-constraint query graph (MulCG), we first define four basic elements:

Vertex There are two types of vertices: constant vertex (rectangle) and variable vertex (circle). A constant vertex represents a grounded KB entity or a value, such as `Barack Obama` or `1961`. A variable vertex represents ungrounded entities or unknown values.

Edge There are two types of edges: relational edge and functional edge. A relational edge represents a predicate in the KB, such as `birthday`. A functional edge represents a functional predicate of a truth, such as $<$ in the truth $\langle$`2000`, $<$, `2001`$\rangle$. Functional predicates are defined in Table 2.

Basic Query Graph A basic query graph is defined as a triple $\langle v_s, p, v_o \rangle$, where v_s denotes a constant vertex as the subject that occurs in a given question, v_o (shaded circle) denotes a variable vertex as hidden answers of the question, p denotes the 'path' that links v_s and v_o by one or two edges [5] (e.g., `officials-holder`).

Constraint A constraint is defined as a triple $\langle v_s, r, v_o \rangle$, where v_s is a constant vertex, v_o is a variable vertex, r is a functional edge, and after instantiation based on a knowledge base, all instantiated entities from v_o should satisfy the predicate of r with regard to v_s.

MulCG A MulCG is constructed based on a basic query graph $\mathcal{B}$ of a question and an ordered constraint sequence $\mathcal{C} = \{\mathcal{C}_1, ..., \mathcal{C}_N\}$ by the following operations: (1) Treat the basic query graph $\mathcal{B}$ of the given question as $\mathcal{G}_0$; (2) Iteratively add $\mathcal{C}_i$ to $\mathcal{G}_{i-1}$ to generate $\mathcal{G}_i$, by linking the variable vertex of $\mathcal{C}_i$ to a vertex of $\mathcal{G}_{i-1}$ with some possible path, or directly merge them as one variable vertex. (3) Output $\mathcal{G}_N$.

Given a MulCG of a question, we can execute it based on the KB by instantiating all variable vertices according to the constraints in order. Specifically, we start from the constant vertex in the basic query graph and instantiate all variable vertices according to the constraints in order. During this procedure, each instantiated paired entities connected by an edge should satisfy the predicate of the edge based on $\mathcal{K}$ and commonsense knowledge.

Figure 2 shows one possible MulCG for the given question. $\mathcal{B}$ is a basic query graph with a constant vertex `United States`, variable vertices y_0 and x, and two edges `officials` and `holder`. $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$ is an ordered constraint sequence detected based on the question, where $\mathcal{C}_1 = \langle$`President`, `Equal`, $y_1\rangle$, $\mathcal{C}_2 = \langle$`2000`, $<$, $y_2\rangle$, $\mathcal{C}_3 = \langle$`1`, `MaxAtN`, $y_2\rangle$. By adding $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ in order, we can construct the MulCG in Figure 2. Note, different constraint order can result in different MulCGs. We will introduce how to generate a MulCG in Section 4.

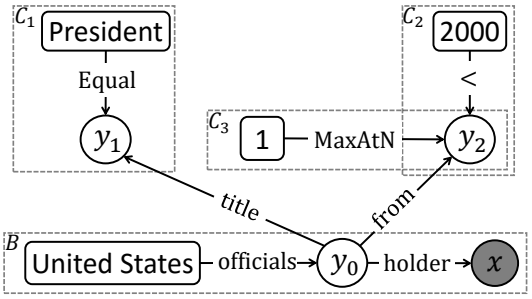


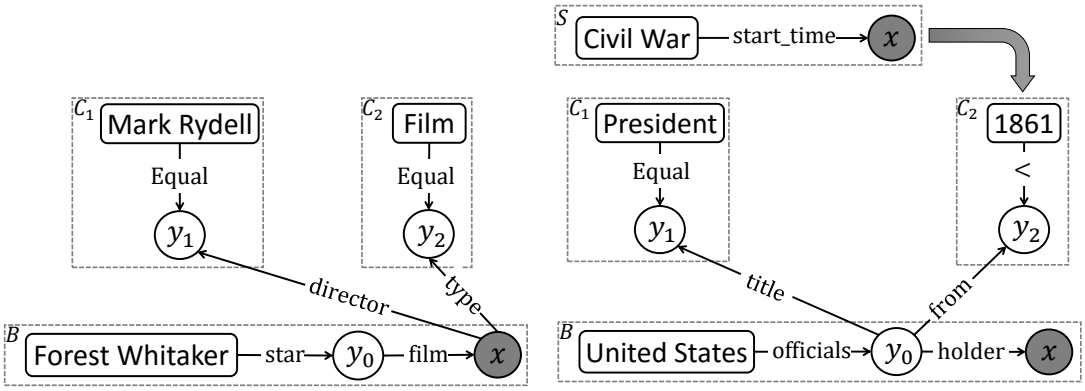
Figure 2: MulCG for question "Who was the first president of United States after 2000?"

Compared to the stage graph in Yih et al. (2015), our MulCG has the following two differences: (1) Entity constraints can be added beyond single KB fact, while stage graph only considers entities that connect to the CVT node of a single KB fact as constraints. (2) Non-entity constraints are defined and handled in a systematic way, while stage graph only considers limited non-entity constraints, i.e., type and gender.

4 Our Approach

Problem Formalization Given a MulCQ $\mathcal{Q}$ and a KB $\mathcal{K}$, the question is parsed into a set of MulCGs $\mathcal{H}(\mathcal{Q})$. For each MulCG $\mathcal{G} \in \mathcal{H}(\mathcal{Q})$, a feature vector $\mathcal{F}(\mathcal{Q}, \mathcal{G})$ is extracted and the one with the highest

[5] If p contains two edges, then the vertex between must represent a CVT entity in KB. We call an edge or two edges with a CVT variable vertex 'path' in this work.



(a) MulCG with entity and type constraint for question "Which films star by Forest Whitaker and are directed by "Who was U.S. president after the Civil War started". Mark Rydell".

Figure 3: MulCGs examples for different type of constraints.

ranking score is selected. Finally, by executing the MulCG, we get the answers $\mathcal{A}$.

4.1 Basic Query Graph Generation

We use the entity linking approach proposed by (Yang and Chang, 2015) to detect entities mentioned by the given question. For each detected entity s, we treat it as a subject constant vertex. Based on the KB, for each unique KB 'path' from s, where a KB 'path' means one hop predicate p_0 or two hop predicates p_1-p_2[6], we construct a basic query graph $\langle s, p_0, x\rangle$ or $\langle s, p_1\text{-}y_{cvt}\text{-}p_2, x\rangle$. y_{cvt} and x are variable vertices, and x denotes the answer. For example, the basic query graph $\mathcal{B}$ in Figure 2 can be represented as $\langle$`United States`, `officials`-y_0-`holder`, $x\rangle$.

To measure the quality of each basic query graph constructed, we leverage a convolutional neural network (CNN)-based model that is similar to (Gao et al., 2015; Shen et al., 2014b; Shen et al., 2014a; Yih et al., 2015) to calculate the similarity between question and the path of the basic query graph. We will describe the training resource in Section 4.4.

4.2 Constraint Detection and Binding

Basic query graph is fit for single relation questions (Yih et al., 2014; Bordes et al., 2015), but not suffices to express a question with multiple constraints, such as the question in Figure 2. Hence, we propose to use constraints to restrict the answer set by adding them into the basic query graph.Adding a constraint contains two steps: *Constraint Detection* and *Constraint Binding*. We explain how to add each of the six kinds of constraints respectively in the following parts.

Entity Constraint Entity constraint is designed to understand entities and relations which are often expressed by noun phrases and verb phrases. A constraint with an entity as its constant vertex is an entity constraint. For instance, Figure 3(a) is a question with multiple entities such as "Forest Whitaker" and "Mark Rydell". After the basic query graph $\mathcal{G}_0 = \mathcal{B}$ is generated, we detect a constraint $\mathcal{C}_1 = \langle$`Mark Rydell`, `Equal`, $y_1\rangle$ and bind it to $\mathcal{G}_0$ by an edge `director`. Generally, the two steps to add an entity constraint are as follows: (1) *Constraint Detection*: For a detected entity $e \in \mathcal{E}$ (e.g., `Mark Rydell`), we construct a constraint $\mathcal{C}_i = \langle e$, `Equal`, $y_i\rangle$ (e.g., $\langle$`Mark Rydell`, `Equal`, $y_1\rangle$); (2) *Constraint Binding*: Given a MulCG $\mathcal{G}_{i-1}$ (e.g., $\mathcal{B}$), and a detected constraint $\mathcal{C}_i$ (e.g., $\langle$`Mark Rydell`, `Equal`, $y_1\rangle$), we try to bind $\mathcal{C}_i$ to $\mathcal{G}_{i-1}$ by linking the variable vertex of $\mathcal{C}_i$ (e.g., y_1 of $\mathcal{C}_1$) to a vertex of $\mathcal{G}_{i-1}$ (e.g., x of $\mathcal{B}$) by a possible path p (e.g., `director`). To measure the similarity between the path p (e.g., `director`) and the 'context pattern'[7] (e.g., "directed by e1") of constraint $\mathcal{C}_i$, we adopt a convolutional neural network (CNN) model which is described in Section 4.4.

[6] A KB 'path' containing two hop predicates means the entity between the two predicates is a CVT entity in the KB.
[7] A 'context pattern' is a 2-word context with the entity mention replaced by a slot "*e1*"

Note, a constraint can be linked to any vertex in the basic query graph. For example, the constraint $\mathcal{C}_1$ in Figure 2 is binding to a variable vertex y_0. When a constraint is binding to a constant vertex, then it is usually used for entity disambiguation.

Type Constraint Answer type is often explicitly expressed by nouns in questions. For instance, "film" is the answer type of the question in Figure 3(a), which denotes a type constraint. (1) *Constraint Detection*: Different from an entity constraint, we detect an answer type with simple but efficient rules as Yao and Durme (2014) for the question. For each KB type[8], we construct a type constraint (e.g., $\mathcal{C}_2 = \langle \texttt{Film}, \texttt{Equal}, y_2 \rangle$); (2) *Constraint Binding*: A type constraint is only added to the variable vertex which denotes the answers with a specific edge `type`.

Explicit Temporal Constraint Temporal constraint is designed to understand temporal expressions, which are often expressed by numericals, prepositional phrases or clauses. An explicit time expression, such as "after 2000" in the question in Figure 2 indicates a temporal constraint $\mathcal{C}_2$. Through linking the vertex y_0 to y_2 with a predicate `from`, functional constraint $\mathcal{C}_2 = \langle \texttt{2000}, <, y_2 \rangle$ selects the subset of the grounded entities for y_0 whose taking office time is later than `2000`. Generally, the two steps to add an explicit temporal constraint are as follows: (1) *Constraint Detection*: We use Stanford NER (Finkel et al., 2005) to detect a time phrase. If a time t (e.g., `2000`) is detected, and a functional operation r (e.g., $<$) is triggered by a lexicon which is partially listed in Table 2, we then construct a constraint $\mathcal{C}_i = \langle t, r, y_i \rangle$ (e.g., $\mathcal{C}_2 = \langle \texttt{2000}, <, y_2 \rangle$); (2) *Constraint Binding*: If an explicit temporal constraint $\mathcal{C}_i$ (e.g., $\mathcal{C}_2$) is detected, then we execute $\mathcal{G}_{i-1}$ (e.g., $\mathcal{B}$ with $\mathcal{C}_1$) on the KB. If there is a KB path p from grounded entities of the linking vertex (e.g., y_0) in $\mathcal{G}_{i-1}$ satisfying the restriction that p's object KB type is `Date_Time`, then the temporal constraint $\mathcal{C}_i$ is bound to $\mathcal{G}_{i-1}$ by an edge denoting p.

Implicit Temporal Constraint Time expressions such as the clause "after the Civil War started" in the question in Figure 3(b) can also trigger a temporal constraint, but it is expressed with an implicit temporal adverbial clause. (1) *Constraint Detection*: A NER can not detect this kind of implicit temporal expressions, so the dependency information is adopted to detect temporal clause starting with predefined keywords[9]. We first use our system to answer the clause to get an explicit time (e.g., by a MulCG $\mathcal{S}$ to get `1861`), then the detection falls into the same procedure as an explicit temporal constraint; (2) *Constraint Binding*: It is same as an explicit temporal constraint.

Ordinal Constraint An ordinal constraint aims to understand the numerals and superlative forms of adjectives or adverbs. For example, the question in Figure 2 contains an expression "first" which denotes that an ordinal constraint $\mathcal{C}_3$ should be added to graph $\mathcal{G}_2$ ($\mathcal{B}$ with constraints $\mathcal{C}_1$ and $\mathcal{C}_2$). After executing $\mathcal{G}_2$, we rank the grounded values of vertex y_2 and pick up the 1^{st} one by functional operation `MaxAtN`. Generally, two steps are adopted to add an ordinal constraint: (1) *Constraint Detection*: We use a manually collected ordinal number list and superlative vocabulary from WordNet (Miller, 1995) to detect an ordinal number n (e.g., 1) and a functional operation *op* (e.g., `MaxAtN` or `MinAtN`) to construct an ordinal constraint $\mathcal{C}_i = \langle n, op, y_i \rangle$; (2) *Constraint Binding*: If an edge p (e.g., from) linking a vertex of $\mathcal{G}_{i-1}$ (e.g., y_0 of $\mathcal{G}_2$) to the variable vertex of a constraint $\mathcal{C}_i$ (e.g., y_2[10] of $\mathcal{C}_3$) satisfies the restriction that the object entity of the predicate of p is a numerical or time value, and the word embedding similarity between the superlative word and the binding path's last word is the largest, then $\mathcal{C}_i$ is bound to $\mathcal{G}_{i-1}$ with edge p (e.g., $\mathcal{C}_3$ is bound to $\mathcal{G}_2$ with `from`).

Aggregation Constraint An aggregation constraint is added when the question starts with phrase "how many", or contains "number of", "count of". For instance, the phrase "how many" in the question "how many children does bill gates have" trigger an aggregation constraint. We treat it specially by counting the number of the grounded entities of the answer vertex.

4.3 Search Space Generation

[8] A KB type is the type of an entity in the KB, such as `People`, `Film`. We extract entire KB types from Freebase.

[9] Such as "when","before","after","during",etc.

[10] Note, since the linking edges for $\mathcal{C}_2$ and $\mathcal{C}_3$ are both `from`, we bind $\mathcal{C}_3$ with $\mathcal{G}_2$ by merging y_3 and y_2 as y_2.

Given a MulCG $\mathcal{Q}$, Algorithm 1 explains how to generate the search space $\mathcal{H}(\mathcal{Q})$. Firstly, we set $\mathcal{H}(\mathcal{Q})$ and a temp set $\mathcal{T}$ empty. A set of entities $\mathcal{E}$ are detected by entity linking component which takes $\mathcal{Q}$ as input. Secondly, for each entity $e \in \mathcal{E}$, we generate all possible basic query graphs $\mathcal{G}_b$ based on the knowledge base $\mathcal{K}$. Each basic query graph $g_b \in G_b$ is added into both $\mathcal{T}$ and $\mathcal{H}(\mathcal{Q})$. Then, for each basic query graph $g_b \in \mathcal{T}$, based on $\mathcal{Q}, \mathcal{E}, \mathcal{K}$, a set of constraints $\mathcal{C}$ are detected through constraint detection component. Function $Permutation(\mathcal{C})$ returns all possible index sequences of permutations of $\mathcal{C}$. For each index sequence $\mathcal{I} \in Permutation(\mathcal{C})$, constraints are bound recurrently. For each constraint $\mathcal{C}_{\mathcal{I}_i}$, we try to bind $\mathcal{C}_{\mathcal{I}_i}$ into a temporary MulCG g_c. Finally, a set of MulCG candidates $\mathcal{H}(\mathcal{Q})$ is generated.

Algorithm 1: MulCG Generation

$\mathcal{H}(\mathcal{Q}) = \emptyset$;
$\mathcal{T} = \emptyset$;
$\mathcal{E} = EntityLinking(\mathcal{Q})$;
foreach $s \in \mathcal{E}$ **do**
  $\mathcal{G}_b = BasicQueryGraphGenration(s, \mathcal{K})$;
  foreach $g_b \in \mathcal{G}_b$ **do**
    insert g_b to $\mathcal{T}$;
    insert g_b to $\mathcal{H}(\mathcal{Q})$;
  end
end
foreach $g_b \in \mathcal{T}$ **do**
  $\mathcal{C} = ConstraintDetection(g_b, \mathcal{E}, \mathcal{Q}, \mathcal{K})$;
  foreach $\mathcal{I} \in Permutation(\mathcal{C})$ **do**
    $g_c = g_b$;
    for $i = 0$ *to* $|\mathcal{I}| - 1$ **do**
      $g_c = ConstraintBinding(g_c, \mathcal{C}_{\mathcal{I}_i})$;
    end
    insert g_c to $\mathcal{H}(\mathcal{Q})$;
  end
end
return $\mathcal{H}(\mathcal{Q})$.

4.4 Features and Ranking

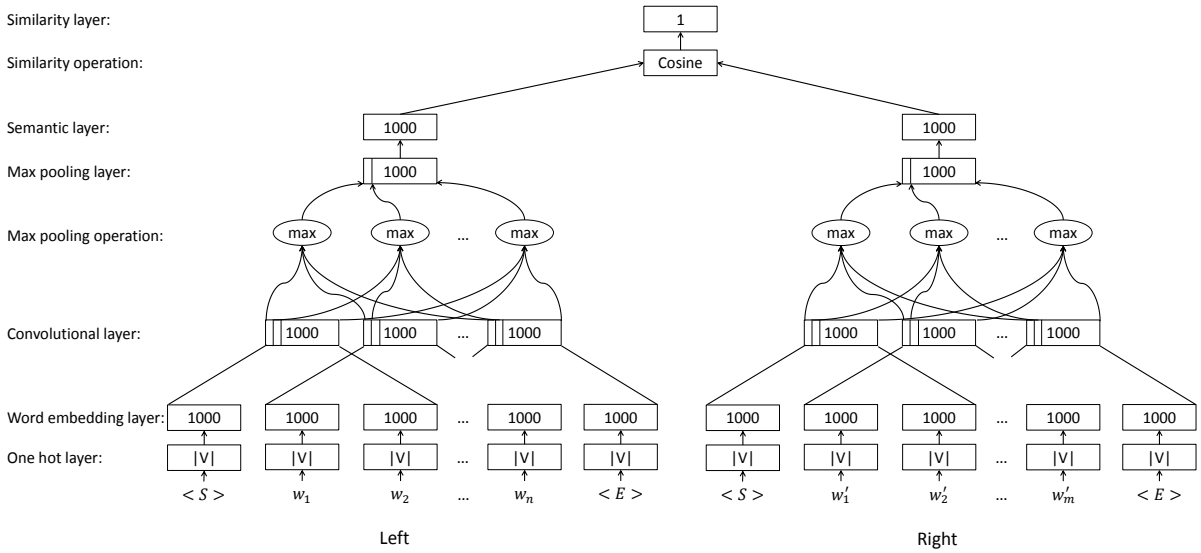


Figure 4: Siamese Convolutional Neural Network

In this work, we propose using Siamese convolutional neural networks (CNN) in Figure 4 to calculate the similarity of two sequences. The model consists of two neural networks taking two sequences as input and maps both of them to k-dimensional vectors. Similar models, such as CDSSM (Shen et al., 2014b; Gao et al., 2015) has been proved for web search. Besides, Yih et al. (2015) use similar frameworks for semantic parsing and question answering. This continuous space representation approach has shown better results compared to lexical matching approaches (e.g., word-alignment models).

Specifically, for two sequences $\mathcal{S}_l = (w_1, w_2, ..., w_n)$ and $\mathcal{S}_r = (w_1^{'}, w_2^{'}, ..., w_m^{'})$, we add "$\langle S \rangle$" and "$\langle E \rangle$" to the head and tail of them respectively to form an input $\mathcal{S}_l^{'}$ and $\mathcal{S}_r^{'}$ of the network. Firstly, a word hashing layer is adopted to hash a word $w \in \mathcal{S}_l$ (or $\mathcal{S}_r$) to a one-hot vector $\mathcal{H}_h = OneHot(w, \mathcal{V})$ based on the vocabulary $\mathcal{V}$. Then by looking up the word embedding table $\mathcal{W}_e$, each word w is embedded into a k-dimensional vector $\mathcal{H}_e = LookUp(\mathcal{H}_h, \mathcal{W}_e)$. A convolutional matrix $\mathcal{W}_c$ is used to project the embedding of words within a context window of 3 words to a local contextual feature vector, and a max pooling layer follows which extracts the most salient local features to get a fix length global feature

vector. With a multi-layer perceptron (MLP) to map the max pooling layer to a semantic layer, we get $\mathcal{H}_l$ and $\mathcal{H}_r$ as the distributed representation of the left and right side. Finally, the semantic similarity is computed as $cosine(\mathcal{H}_l, \mathcal{H}_r)$. To conclude, this model can be defined as $Sim(\mathcal{S}_l, \mathcal{S}_r)$.

Based on the CNN model described above, we design four features in Table 3 for the basic query graph, and four types of features for each kind of constraints, namely indicating features $\mathcal{I}_{s/t/e/i/o/a}$, count features $\mathcal{N}_{s/t/e/i/o/a}$, constraint detection features $\mathcal{V}_{s/t/e/i/o/a}$ and constraint binding features $\mathcal{P}_{s/t/e/i/o/a}$.

Given a MulCQ $\mathcal{Q}$ and a MulCG candidate $\mathcal{G}_i \in \mathcal{H}(\mathcal{Q})$, $\mathcal{F}_k(\mathcal{Q}, \mathcal{G}_i)$ represents the k^{th} feature of $\langle \mathcal{Q}, \mathcal{G}_i \rangle$. A liner scoring function is adopted to calculate the reward score of each $\langle \mathcal{Q}, \mathcal{G}_i \rangle$ as $Score(\mathcal{Q}, \mathcal{G}_i) = \sum_k \lambda_k \cdot \mathcal{F}_k(\mathcal{Q}, \mathcal{G}_i)$. A learning to rank method lambda-rank (Burges, 2010) is used to learn the each feature weight λ_k.

Features	Description
Basic Query Graph	$\mathcal{S}_{ent}$: Entity linking score ('EntityLinking') $\mathcal{S}_{pc}$: CNN score between question pattern and path ('PatChain') $\mathcal{S}_{qep}$: CNN score between question and entity+path ('QuesEP') $\mathcal{S}_{cw}$: CNN score between question and path, where the model is trained on ClueWeb ('ClueWeb')
Constraint	$\mathcal{I}_{s/t/e/i/o/a}$: **I**ndicating features for each kind of constraints, each value is 1 or 0 $\mathcal{N}_{s/t/e/i/o/a}$: **N**umber of each kind of constraints, each value is a positive integer $\mathcal{V}_s$: Sum of entity linking scores for constant **V**ertex in each entity constraint. $\mathcal{V}_t$: Sum of CNN scores between answer type and KB type of the constant **V**ertex in each type constraint $\mathcal{V}_i$: Sum of reward scores of temporal clause for constant **V**ertex in each implicit temporal constraint $\mathcal{P}_s$: Sum of CNN scores between context pattern and binding **P**ath for each entity constraint $\mathcal{P}_o$: Sum of embedding similarities between superlative phrase and binding **P**ath for each ordinal constraint

Table 3: Features and their description.

5 Experiment

We introduce experiment part on these aspects. We first introduce the settings of our experiments, especially the three data sets containing question/answer (QA) pairs. On these data sets, the results of our method are given, and based on the results we analyze drawbacks.

5.1 Set Up

System Components We use the entire Freebase dump which is same as Berant et al. (2013) and host it with Virtuoso engine[11]. Besides, an entity linker (Yang and Chang, 2015), the Stanford NER (Finkel et al., 2005), and an in-house implementation of shift-reduce dependency parser (Zhang and Nivre, 2011) with Stanford dependency (De Marneffe et al., 2006) which is used in detecting temporal clause are adopted in this work.

Data Sets We evaluate our approach on three data sets.

(i) *ComplexQuestions* (CompQ): It is a new data set which includes 2100 QA pairs released by this work with the details in Section 2.

(ii) *WebQuestions* (WebQ): It contains 3778 QA pairs on training set and 2032 on test set which is released by Berant et al. (2013). The questions are collected from query log and the answers are manually labeled based on Freebase.

(iii) *SimpleQuestions* (SimpQ): Each question in *SimpleQuestions* is written by a human with reference to a knowledge base triple.

CNN Training Data To train a CNN model, we first use our system S_0 to enumerate all possible basic query graphs for each question, and pick up the ones with the F_1 score larger then 0. We then get a set of question-path pairs to train the initial CNN models. Then we use these CNN models to train a system S_1. Given S_1, we use it to answer each question to get all MulCGs and pick up the ones with the F_1 score larger then 0.5. Finally we get a set of question-path pairs as our CNN training data.

METHOD	SETTING	CNN TRAINING SOURCE			TEST (Average F_1)	
	Constraint	**CompQ-Train**	**WebQ-Train**	**SimpQ**	**CompQ-Test**	**WebQ-Test**
STAGG	√	√	-	-	36.89	-
	√	-	√	-	-	52.36
	√	√	√	-	37.42	52.35
	√	√	√	√	37.69	54.30
This Work	-	√	-	-	30.31	-
	-	-	√	-	-	50.98
	-	√	√	-	31.58	51.69
	-	√	√	√	31.42	54.20
	√	√	-	-	40.94	-
	√	-	√	-	-	52.43
	√	√	√	-	41.75	52.49
	√	√	√	√	**42.33**	**54.36**

Table 4: Average F_1 score on **CompQ-Test** and **WebQ-Test** which stand for the test sets of *ComplexQuestions* and *WebQuestions* respectively. **CompQ-Train**, **WebQ-Train** stand for the training sets of *ComplexQuestions* and *WebQuestions* respectively, and **SimpQ** represents the *SimpleQuestions*. **Constraint** means whether to add constraints or not.

5.2 Results and Analysis

We re-implement STAGG method (Yih et al., 2015) as our baseline. STAGG method considers some constraints, such as entity constraint on CVT vertex, type constraint and ordinal constraint triggered by "first" and "oldest". To evaluate our method compared to the baseline on different settings, we design experiments shown in Table 4. The results show that our method outperforms the baseline on the test set of *ComplexQuestions* and have comparable result on the test set of *WebQuestions*.

Specifically, the **Constraint** column in Table 4 indicates using constraints or not. Through these settings, we can see how important the constraints are for answering multi-constraint questions. Besides, $\mathcal{S}_{pc}$ and $\mathcal{S}_{qep}$ are in-domain features that the CNN models they relay on vary from the training data. So we train different CNN models for $\mathcal{S}_{pc}$ and $\mathcal{S}_{qep}$ on different combinations of training resource. By these settings, we can know how does the amount of CNN training data effect on the results.

5.2.1 Results on *ComplexQuestions*

Table 4 shows that, when using the same CNN training sources, our method outperforms the STAGG method about 4.35 ± 0.30 points (40.94-36.89, 41.75-37.42 and 42.33-37.69) on *ComplexQuestions*. This result indicates that our systematic constraint solving method is more suitable for answering questions with multiple constraints than the baseline. Besides, adding constraints can bring about 10.54 ± 0.37 points' (40.94-30.31, 41.75-31.58 and 42.33-31.42) gain on *ComplexQuestions* which tells that constraints as an important feature can help to bring a significant improvement for multi-constraint questions. Since the training set of *ComplexQuestions* contains a relative small size of QA pairs (1300), we add *WebQuestions* and *SimpleQuestions* to enlarge the CNN training data. So another improvement is gained from adding new CNN training resource. STAGG achieves a 0.8 points' gain, our method improves with 1.39 and 1.11 points with or without constraint respectively.

5.2.2 Results on *WebQuestions* and *SimpleQuestions*

We also evaluate our method on the test set *WebQuestions*. From Table 4 we can see that, our method has comparable results with STAGG by comparing 52.43 to 52.36[12], 52.49 to 52.35, and 54.36 to 54.30. This indicates that adding constraints can get a small improvement on the *WebQuestions* because most of the questions in the *WebQuestions* are simple questions, each of which can be solved by a single KB relation. So adding more CNN training resource bring about 1.93 (54.36-52.43) point improvement for our method. Table 5 also shows the results of recent work on *WebQuestions*, and this work outperforms the others.

[11] http://virtuoso.openlinksw.com/

[12] We get a similar result with Yih (2015)'s 52.50 F_1 score.

Besides, the result also shows that our system performs stable, which not only works well on multi-constraint questions, but also simple questions. To further prove the stability of our system, we also test on the test set of *SimpleQuestions*, by using a dictionary based entity linker, our method achieves 72.78 on accuracy. This is because all the *SimpleQuestions* are single relation questions.

6 Related Work and Discussion

Knowledge-based question answering (KBQA) works with lexical features (Yao, 2015) or convolutional neural network features (Yih et al., 2015) already achieve good results on single relation questions. *WebQuestions* (Berant et al., 2013) and *SimpleQuestions* (Bordes et al., 2015) are two data sets for KBQA task. Based on our analysis, more than 84% and almost all questions in *WebQuestions* and *SimpleQuestions* are single relation questions.

Method	**Average F_1**
(Berant et al., 2013)	35.70
(Bordes et al., 2014b)	29.70
(Yao and Durme, 2014)	33.00
(Bao et al., 2014)	37.50
(Berant and Liang, 2014)	39.90
(Yang et al., 2014)	41.30
(Wang et al., 2014)	45.30
(Bordes et al., 2015)	39.90
(Yao, 2015)	44.30
(Berant and Liang, 2015)	49.70
(Yih et al., 2015)	52.50
(Reddy et al., 2016)	50.30
(Xu et al., 2016)	53.30
This work	**54.36**

Table 5: QA result on **WebQ**.

Previous KBQA work (Yao and Durme, 2014; Bordes et al., 2014a; Yang et al., 2014; Fader et al., 2014; Reddy et al., 2014; Dong et al., 2015; Bordes et al., 2015) testing on these two data sets do not solve multiple constraints systematically. Different methods such as specific *bridging* operator (Berant et al., 2013), semantic template $p.(p1.e1 \sqcap p2.e2)$ (Berant and Liang, 2014) are used to treat questions with multiple entities specially. Yih et al. (2015) have already done some work to handle questions with constraints, such as considering entity constraints on CVT vertice or ordinal constraints triggered by "first" or "oldest". But these methods haven't specifically evaluated their KBQA systems or presented solutions to multi-constraint questions in a systematic manner.

We propose a novel method to answer questions with multi-constraints by multiple-constraint query graphs. By evaluating it on the *ComplexQuestions* released by this work, we find our method works well on multi-constraint questions.

7 Conclusion

We release a QA data-set *ComplexQuestions* which contains multi-constraint questions, and propose a novel systematic KBQA method using multi-constraint query graph to answer multi-constraint questions. Experiments show that, compared to state-of-the-art approaches, our method obtains comparable results on the existing benchmark data-sets *WebQuestions* and *SimpleQuestions*. Furthermore, we achieve significant improvement on the newly created *ComplexQuestions* data-set. Besides, we put learning the matching between constraint expressions and semantic constraints from massive data in future work.

References

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 967–976, Baltimore, Maryland, June. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1415–1425.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 615–620.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. In *Machine Learning and Knowledge Discovery in Databases*, pages 165–180. Springer.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11:23–581.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 260–269, Beijing, China, July. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1156–1165. ACM.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Jianfeng Gao, Li Deng, Michael Gamon, Xiaodong He, and Patrick Pantel. 2015. Modeling interestingness with deep neural networks, December 17. US Patent 20,150,363,688.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014a. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014b. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 373–374. International World Wide Web Conferences Steering Committee.

Zhenghao Wang, Shengquan Yan, Huaming Wang, and Xuedong Huang. 2014. An overview of microsoft deep qa system on stanford webquestions benchmark. Technical report, Technical report, Microsoft Research.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2326–2336, Berlin, Germany, August. Association for Computational Linguistics.

Yi Yang and Ming-Wei Chang. 2015. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *EMNLP*, pages 645–650.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 956–966.

Xuchen Yao. 2015. Lean question answering over freebase from scratch. In *Proceedings of NAACL-HLT*, pages 66–70.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *ACL (2)*, pages 643–648. Citeseer.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China, July. Association for Computational Linguistics.

Pengcheng Yin, Nan Duan, Ben Kao, Junwei Bao, and Ming Zhou. 2015. Answering questions with complex semantic constraints on open knowledge bases. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1301–1310. ACM.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA - Short Papers*, pages 188–193.

Selecting Sentences versus Selecting Tree Constituents for Automatic Question Ranking

Alberto Barrón-Cedeño and **Giovanni Da San Martino** and
Salvatore Romeo and **Alessandro Moschitti**

Qatar Computing Research Institute,
Hamad bin Khalifa University
Doha, Qatar
{albarron, gmartino, sromeo, amoschitti}@qf.org.qa

Abstract

Community question answering (cQA) websites are focused on users who query questions onto an online forum, expecting for other users to provide them answers or suggestions. Unlike other social media, the length of the posted queries has no limits and queries tend to be multi-sentence elaborations combining context, actual questions, and irrelevant information. We approach the problem of question ranking: given a user's new question, to retrieve those previously-posted questions which could be equivalent, or highly relevant. This could prevent the posting of nearly-duplicate questions and provide the user with instantaneous answers. For the first time in cQA, we address the selection of relevant text —both at sentence- and at constituent-level— for parse-tree-based representations. Our supervised models for text selection boost the performance of a tree kernel-based machine learning model, allowing it to overtake the current state of the art on a recently released cQA evaluation framework.

1 Introduction

Community-driven question-and-answering (cQA) sites are popular forums in which users ask and answer questions on diverse topics. The freedom in these websites, and the diversity of their users, promote massive participation, resulting in large amounts of texts in the form of both questions and answers.

All the steps in the assembly line of these highly-collaborative sites —from the question posting up to the seek for sensitive answers among those triggered by the question—, pose interesting natural language processing (NLP) challenges. When a user posts a query question, a model can search for previously-posted equivalent or relevant questions which might address the user's information need at once. Once a number of comments have been posted intending to answer a question, another model can select the most appropriate one, or at least rank them on the basis of their quality. The same technology can be applied to discard inappropriate or diverting answers. When a website has accumulated a significant amount of posts, a model can be implemented to look for near-duplicates or related questions and answers.

Different approaches have been proposed to address these tasks. Here we focus on the first of them: question re-ranking in cQA. That is, given a query question q and a pool of previously-posted questions D, rank the questions in D according to their relevance against q. This problem has attracted the attention of a manifold of research works which rely on the use of standard lexical similarity metrics, semantic representations, machine-translation models, tree kernels, or neural networks, to mention just some examples of representations and models. Nevertheless, those approaches neglect one of the inherent facets of cQA sites: opposed to other social media, users are free to post potentially ill-formed texts of anarchic lengths. They may include multiple sentences with courtesy chunks, potentially-redundant elaborations, or off-topic fragments. In this paper, we address the problem of selecting the most relevant text chunks in the questions in order to build a better representation of the texts to be fed into the machine learning machinery. We propose supervised and unsupervised models that operate both at sentence and at chunk level (using constituency parse trees) and apply them on top of a state-of-the-art tree-kernel-based classification model. To the best of our knowledge, this is the first time that this problem is addressed. Our

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,
pages 2515–2525, Osaka, Japan, December 11-17 2016.

results on a recently-released cQA corpus show that, by carefully selecting both sentences and words, the performance of the ranking model can be significantly improved (in the range of more than two MAP points), also improving the current best model on the evaluation framework we used.

The rest of our contribution is distributed as follows. Section 2 overviews the literature addressing different problems in cQA, paying special attention to question ranking. Section 3 describes the machine learning strategy to rank questions upon which we build our main contribution: the proposal of two models to select the most relevant —noise-less— text fragments to represent the questions, described in Section 4. Section 5 describes our experimental settings and discusses the obtained results. Finally, Section 6 draws conclusions and sketches some of our current efforts.

2 Related Work

Community question answering poses various challenges: answer and question ranking, and question de-duplication are three examples. We now review the related literature with focus on question ranking.

One of the first approaches to answer ranking relied completely on the website's metadata (Jeon et al., 2006), such as an author's reputation and click counts. Agichtein et al. (2008) explored a graph-based model of contributors relationships together with both content- and usage-based features. These approaches depend heavily on the forum's meta-data and social features. Still, as Surdeanu et al. (2008) stress, relying on this kind of data causes the model portability to be difficult; a drawback that disappears when focusing on the content of the questions and answers only. Therefore, our model is based on textual content only. Some of the most recent proposals aim at classifying whole threads of answers (Joty et al., 2015; Zhou et al., 2015) rather than each answer in isolation.

Question ranking can be approached from different fronts. Cao et al. (2008) approached it as a recommendation task: given a query question, recommend questions that could be interesting or relevant, regardless of whether they convey the same information request. They tackle this problem by comparing representations based on topic terms graphs; i.e., by judging topic similarity. In a follow up paper, Duan et al. (2008) searched for equivalent questions by considering the question's focus as well. Zhou et al. (2011) dodged the lexical gap between two questions by assessing their similarity on the basis of a (monolingual) phrase-based translation model (Koehn et al., 2003). They considered the (pre-filtered) contents of the question–answer pairs as their "parallel" corpus to learn the translation model from. Jeon et al. (2005b) had used monolingual translation as well. Given a large repository of question and answer threads, they looked for highly-similar threads (Jeon et al., 2005a). Similar answers are likely to address similar questions! The questions in the so-generated pairs compose their "parallel" corpus. Wang et al. (2009) computed their similarity function on the syntactic-tree representations of the questions. The more substructures the trees have in common, the more similar their associated questions are. A different approach using topic modeling for question retrieval was introduced by Ji et al. (2012) and Zhang et al. (2014). Here, the authors use LDA topic modeling to learn the latent semantic topics that generate question/answer pairs and use the learned topics distribution to retrieve similar historical questions.

The recent boom in neural network approaches has also impacted question retrieval. dos Santos et al. (2015) applied convolutional neural networks to retrieve semantically-equivalent questions' subjects. When dealing with whole questions —subject and (generally long) body—, they had to aggregate a bag-of-words neural network to boost the model's performance. They suggested that the performance of state-of-the-art models for semantic paraphrasing assessment on short texts (e.g., (Filice et al., 2015b)) cannot be applied straightforwardly to whole questions in cQA. For the first time, we address such problem in this paper.

The two editions of the SemEval Task 3 on cQA (Nakov et al., 2015; Nakov et al., 2016) have triggered a manifold of approaches. The datasets they released include manual crowd-sourced annotation rather than forum-inferred judgments. The 2015 edition focused on answer retrieval. The first performing system (Tran et al., 2015) applied machine translation in a similar fashion as Jeon et al. (2005b) and Zhou et al. (2011), together with topic models, embeddings, and similarities. Both the first and the second runners (Hou et al., 2015; Nicosia et al., 2015) applied supervised models with lexical, syntactic and meta-data features. The 2016 edition included a question retrieval challenge as well. We take ad-

vantage of the evaluation framework developed for this task. The top-three participants opted for SVMs as learning models. The top-ranked (Franco-Salvador et al., 2016) used SVM^{rank} (Joachims, 2006), the first (Barrón-Cedeño et al., 2016) and second (Filice et al., 2016) runners up used KeLP (Filice et al., 2015a) to combine various kernels. Another difference between these models is in the amount of knowledge they use. Franco-Salvador et al. (2016) rely heavily on distributed representations and semantic information sources, such as Babelnet and Framenet. Both Barrón-Cedeño et al. (2016) and Filice et al. (2016) use lexical similarities and tree kernels on parse trees. No statistically-significant differences were observed in the performance of these three systems.

To the best of our knowledge, so far the only other work exploring text selection to improve cQA systems is that of Romeo et al. (2016). In that paper, we do sentence selection and pruning on the basis of attention weights, computed with neural networks.

3 Base Model to Rank Questions

In this section we describe a state-of-the-art technique to assess question–question similarity in question ranking. This is the core of our ranking approach and the base on which we test our text selection models (cf. Section 4). We apply a learning-to-rank approach (Liu, 2009) to produce the ranking of forum questions. Related questions in cQA sites are usually spotted and labeled as such by the users themselves. This can be directly projected into a training dataset with binary annotations: *Relevant* vs. *Irrelevant* and a perfect ranking puts all the *Relevant* questions on top of all the *Irrelevant* ones, regardless of the order within both subsets (Cao et al., 2008; dos Santos et al., 2015; Jeon et al., 2005b). We adopt the same architecture as the recently-proposed, most successful, models on this kind of setting (Franco-Salvador et al., 2016; Barrón-Cedeño et al., 2016; Filice et al., 2016). We apply a kernel approach to solve a binary classification problem, $f : Q \times D \rightarrow \{Relevant, Irrelevant\}$, and sort the forum questions $d \in D$ according to their classification score against q: $f(q, d)$.

We opt for a tree kernel applied to parse-tree representations (Moschitti, 2006b; Sun et al., 2011), as it performs well in ranking both passages (Severyn and Moschitti, 2012) and questions (Barrón-Cedeño et al., 2016; Da San Martino et al., 2016; Filice et al., 2016). Additionally, the nodes of the parse trees of the pairs (q, d) are marked with a REL tag when there is at least a lexical match between the phrases of the questions (c.f. (Filice et al., 2015b) for details). The approach for dealing with a pair of trees, is to compose kernels on single trees $K^T(x_1, x_2)$:

$$K((q_i, d_i), (q_j, d_j)) = K^T(t(q_i, d_i)), t(q_j, d_j))) + K^T(t(d_i, q_i)), t(d_j, q_j))), \quad (1)$$

where d_i and d_j are the i^{th} and j^{th} retrieved questions and $t(x, y)$ extracts the syntactic tree from the text x, enriching it with REL tags computed with respect to y. Note that $t(x, y)$ is not symmetric: $t(y, x)$ would return the tree related to y enriched with REL tags. Specifically we apply a partial tree kernel (PTK) as the base kernel K^T, which counts the number of shared subtrees between x_1 and x_2 (Moschitti, 2006a).

4 Text Selection for Parse-Tree Representations

We complement our classifier with twenty-two similarities $sim(q, d)$ computed on lemmatized versions of the texts, which are described in Table 1. The inverse of the Google-generated position of d (included in the corpus) is included as well. We plug the similarities on an RBF kernel and combine it linearly with the tree kernel as they boost the performance of the tree kernels (Da San Martino et al., 2016).

Questions in cQA websites tend to be composed of multiple sentences (c.f. Figure 1) and to include noisy or irrelevant fragments. We apply the same trick as Filice et al. (2016) to feed our tree kernel with pairs of single trees: we hang together the constituency parse trees of all the sentences in q (d) from an additional root node. Still, tree kernels are expensive and sensitive to noise, thus it is difficult for them to deal with multi-sentence noisy text. In order to tackle these issues in the questions' texts, we designed two general strategies which operate either at sentence or at word level. Our objective is twofold: we want to represent our questions with the shortest —most informative— text fragments and at the same time discard noise, such as acknowledgments or unnecessary elaborations.

Metric		Details
String similarity		
Greedy string tiling	(Wise, 1996)	Considering a minimum matching length of 3.
Longest common subsequence	(Allison and Dix, 1986)	Both standard and normalized by the first string.
Longest common substring	(Gusfield, 1997)	Based on generalized suffix trees.
Lexical similarity		
Jaccard coefficient	(Jaccard, 1901)	Over stopworded $[1, \ldots, 4]$-grams.
Word containment	(Lyon et al., 2001)	Over stopworded $[1, \ldots, 2]$-grams.
Cosine		Over stopworded $[1, \ldots, 4]$-grams.
		Over $[1, \ldots, 4]$-grams.
		Over $[1, \ldots, 3]$-grams of part of speech.
Syntactic similarity		
PTK	(Moschitti, 2006a)	Similarity between shallow syntactic trees.

Table 1: Overview of similarity metrics.

4.1 Learning to Select Sentences

This selection strategy occurs before the parse trees of the questions are built. The output of this process is a selected number of sentences from q and d, composed on the basis of sentence-pair rankings. Algorithm 1 sketches our strategy. Firstly, we combine all the possible pairs of sentences between questions q and d. Secondly, we compute a similarity function φ for each pair and rank the pairs accordingly. Thirdly, we identify the k pairs of sentences with the highest similarities. Finally, we discard those sentences from q and d which have not been included within the top-k pairs. Our rationale implies a number of constraints. The sentences that result from the process are in the same order as in the original questions; not in the order of the sentence ranking. This is because we want to preserve the discourse structure of the text. Sentences are promiscuous: they can be linked to different sentences from the other question. If a sentence is part of more than one of the top-k pairs, it is not duplicated in the output. Finally, the input question may go unaltered on either side. The similarity function φ is the key factor for this model and we experimented with two learning strategies:

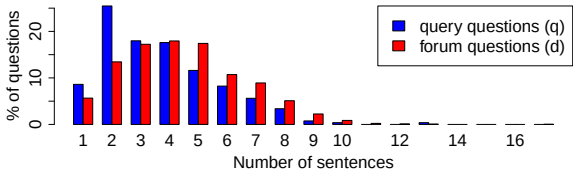


Figure 1: Distribution of question lengths in terms of sentences for query (q) and forum (d) questions.

Our unsupervised model is based on the cosine similarity over TF×IDF-weighted vector representations of the sentences in q and d.

Our supervised model is based on a binary SVM classifier, using the same similarity functions as in Section 3, over and RBF kernel. We add information in terms of features on the position of the sentence within the question: the inverse of the position, whether the sentence appears in position 1, between positions 2 and 4 (inclusive), or after position 4. We take 4 as threshold because it is the mode of the number of sentences in the questions of the corpus (mean=4.38). Our prediction function is $c(p_q, p_d) \in \{Relevant, Irrelevant\}$. The class labels are borrowed from those at question level.

We took good care of not misusing the development and test data. In the unsupervised model, we compute the document-frequency statistics on sentences from the training set only. In the supervised model, the scores for the training set are estimated by 5-fold cross validation. On dev and test, the used document frequencies are those from the training set and the scores are computed with the model trained on the training partition.

4.2 Learning to Select Constituents in Tree Kernel Spaces

Our strategy for selecting text at token level is to operate directly on the parse trees and prune those branches which are associated with less important or noisy text fragments. We use a supervised approach based on kernel methods to determine such fragments. A few facts on kernel methods need to be recalled to better understand our approach. After training a kernel method, using a kernel function $K()$, on the

Input :
q the query question
d the forum question
k the maximum number of pairs to return
Output:
q' q, after selecting the best sentences; $len(q') \leq len(q)$
d' d, after selecting the best sentences; $len(d') \leq len(d)$

$P \leftarrow sent(q) \times sent(d)$ // All the possible sentence pairs between q and d
for $p \in P$ **do**
  $score_p = \varphi(p_q, p_d)$ // Compute the similarity between the sentences
end
$P \leftarrow rank(P, score_P)$ // Rank the pairs in P in terms of the similarities
$P \leftarrow trim(P, k)$ // Select (at most) the top-k pairs in P
$q' \leftarrow q \cap P_q$ // Remove from q all the sentences not in P
$d' \leftarrow d \cap P_d$ // Remove from d all the sentences not in P
return q', d'

Algorithm 1: Sentence-level selection.

problem sketched in Section 3, the solution of the dual optimization problem is expressed as a linear combination of a subset of the training examples: $M = \{(\alpha_i, (q_i, d_i))\}$, where the (q_i, d_i) are training examples and α_i are the coefficients of the combination. The classification of a new example is obtained as the sign of the score function $f()$:

$$f\left(q, d\right) = \sum_{1 \leq i \leq |M|} \alpha_i K\left((q, d), (q_i, d_i)\right), \tag{2}$$

where $|M|$ is the number of support vectors, i.e., the number of elements of the set M. The higher the absolute value of the score of an example, the more confident the learning algorithm is in classifying it. We exploit such property of the kernel methods to devise a strategy to determine the importance $w(n)$ of a node. Let n be a node of a tree t, $\overset{n}{\triangle}$ is the proper sub-tree rooted at n, i.e., the tree composed of n and all its descendants in t. We use the score of $\overset{n}{\triangle}$ with respect to M to assess the importance of n:

$$w(n) = \begin{cases} \sum_{1 \leq i \leq |M|} \alpha_i K^T(\overset{n}{\triangle}, q_i) & \text{if } n \in q, q \in Q \\ \sum_{1 \leq i \leq |M|} \alpha_i K^T(\overset{n}{\triangle}, d_i) & \text{if } n \in d, d \in D. \end{cases} \tag{3}$$

In order to be consistent, only the parse trees of $q_i \in Q$ will be used to compute $w(n)$, if n belongs to a question in Q for each pair $(q_i, d_i) \in M$. Conversely if n belongs to a question in D only the parse trees of $d_i \in D$ will be used. Note that, by grouping and caching the scores computations as in (Aiolli et al., 2011, eq. (4)), the worst-case complexity of Eq. (3) for all the nodes is the same as the complexity of predicting the class of the pair (q, d).

Now we can proceed to prune a tree on the basis of the $w(n)$ importance estimated by model M for each of its nodes and a user-defined threshold. We prune a leaf node n if $-h < w(n) < h$. If n is not a leaf, then it is removed if all its children are going to be removed. Note that the threshold h determines the number of pruned nodes. Our algorithm has a constraint: REL-tagged nodes are never pruned, regardless of their estimated importance. This is because a REL tag indicates that q and d share a common leaf in $\overset{n}{\triangle}$, which conveys useful information for paraphrasing (Filice et al., 2015b).

5 Experiments

We perform three different experiments. Firstly, we evaluate the performance of our supervised sentence-level classifier and select the best of them for the next experiment. Secondly, we analyze the impact of the supervised and unsupervised sentence selectors in the performance of our overall question ranking model. Thirdly, we evaluate the impact of our tree-pruning strategy in the performance of our overall

Class	train	dev	test	overall
Relevant	1,083	214	233	1,530
Irrelevant	1,586	286	467	2,339
Total	2,669	500	700	3,869

Table 2: Class distribution in the training, development, and test partitions of the cQA-2016 corpus.

Model	Acc	P	R	F_1	MAP	AvgRec	MRR
sim	65.88	44.44	14.29	21.62	60.15	85.39	64.22
$sim + pos_{lin}$	68.24	53.85	25.00	34.15	61.13	85.79	64.22
$sim + pos_{RBF}$	71.76	59.09	46.43	52.00	62.84	86.95	66.67

Table 3: Performance of the sentence classifier on a subset of manually annotated sentences from the development set. Similarity features used on a linear kernel. Positional (pos) features are either used on a linear or an RBF kernel.

question ranking model. As discussed in Section 3, we use binary SVMs to generate the rankings, combining a tree kernel for the parse trees and an RBF kernel for the rest of features.

5.1 Setup

We run our experiments on the SemEval 2016 Task 3 on Community Question Answering evaluation framework (Nakov et al., 2016), which uses the cQA-2016 corpus in which each item includes a query question linked to ten potentially-relevant candidate questions. Table 2 shows the class distribution. We stick to the retrieval problem formulation of the task: a perfect model should rank relevant documents on top of the irrelevant ones. Our evaluation is based on mean average precision, average recall, and mean-reciprocal rank. This allows for a direct comparison against Task 3's official ones.

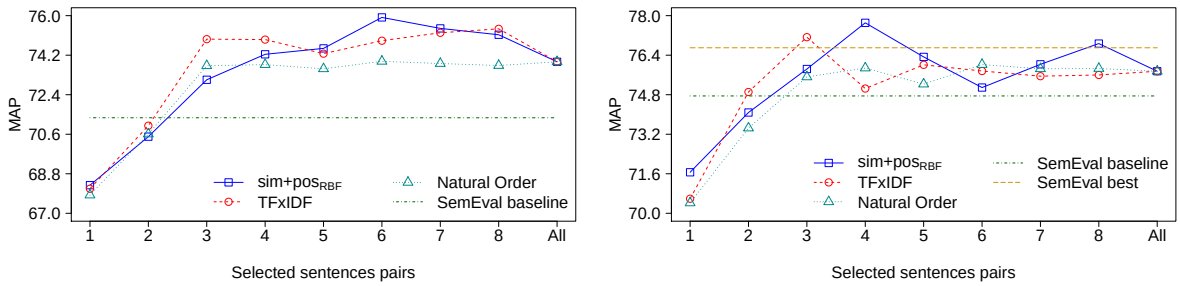
5.2 Impact of Sentence Selection in Question Ranking

We first evaluate our sentence ranking model in isolation, before assessing its impact on the question ranker. We trained a sentence level classifier, on the training partition of the corpus, as defined in Section 4.1 —considering the question-level annotations as gold standard. However, the latter choice introduces a lot of false positives as sentences in relevant questions may be unrelated. Thus, for correctly evaluating the sentence classifier, we used labels obtained by crowdsourcing. We selected sentence pairs to be annotated from the development set according to a few constraints. The (q, d) pairs we extract the sentences from must include at least five sentences in d. Each selected sentence must include at least 3 content words and 3 stop-words. This resulted in 125 sentences for this small control experiment. We submitted HIT's composed of 15 sentence pairs to CrowdFlower.[1] Annotators had to decide if two sentences "expressed the same information". Each item was annotated by 3 contributors, with an agreement of 0.83. 21% of the sentence pairs resulted as positive.

Table 3 shows the performance of different configurations of the sentence-level classifier, trained on the training partition and tested on the Crowdflower-annotated data. The task is the same as at question level: ranking, but we include accuracy, precision, recall, and F_1-measure to get a clear picture. Similarity measures constantly perform better on RBF kernels and, as observed, positional features do as well. The performance boost caused by the positional features is more evident in terms of F_1 and recall. These values might be perceived as relatively low, but let us keep in mind that many (unrelated) sentences in the training set inherit incorrect labels from the label of their question. Given these figures, we select the model with both similarities and position features on an RBF kernel to select sentences for the question ranker. Our unsupervised model does not require any evaluation nor tuning.

Now we look at the impact of our sentence selection model in the question ranker. Figure 2 shows the performance of the ranker as a function of the number of sentences. In our supervised model sim+pos$_{RBF}$, the x-axis refers to the number of pairs considered after the SVM-generated ranking. In our unsupervised model TF$\times$IDF, the x-axis stands for the number of sentences that represent q and d in the ranking model. We also display the behavior when the sentences are added in the natural order, which could be considered as a sentence-selection baseline. The constant line "SemEval baseline" corresponds to the competition baseline (Nakov et al., 2016). The upper constant line in Figure 2b corresponds to the winner of the SemEval competition (Franco-Salvador et al., 2016).

[1] https://www.crowdflower.com



(a) On the development set.

(b) On the test set.

Figure 2: Evolution of MAP for various sentence-selection strategies. The point marked as *All* on the x axis —where the curves converge— corresponds to the values obtained when using all the sentences.

It is worth observing that all our models depart relatively low performance values when considering one single sentence: MAP $\simeq$ 68.12 on development and 70.40 on test. Nevertheless, the natural order one stops improving at 3 sentences and it never goes over the base system, which considers the full texts. This is not the case for our selection models: already from 3 (and 4) sentences they improve over the full-text system. On the development set the supervised model reaches a MAP of 76.01 with five sentences and the unsupervised reaches 75.40 with eight sentences. The SVM supervised model is also the best on test, even when considering less sentences (Figure 2b). Again, the best performance is obtained by the supervised model. Although, TF$\times$IDF has a peak with 3 sentences, adding more pairs results in a quick performance decrease. Therefore, the supervised sentence classifier is more stable and thus preferable. In summary, our sentence pre-selection manages to identify those sentences which are more relevant to assess the similarity between two questions and produce a better ranking. Overall, the supervised model performs better than the unsupervised one and the best configurations improve over the SemEval best system. We select the best performing model on development —the SVM on sim+pos$_{RBF}$— and list its corresponding performance in Table 4, for comparison with the best pruning model and the rest of systems.

5.3 Impact of Constituency Tree Selection on Question Reranking

We experimented with two models for generating the weights $w()$ to prune the nodes: (*i*) an SVM with the kernels described in Section 3 (*model 1*); (*ii*) the same SVM trained on the training set after applying the sentence selection described in section 4.1 (*model 2*).

We first performed a 5-fold cross validation on the training set to select a pruning threshold. For each cross validation split into train/test, we first perform learning on train to get weigths using *model 1*, then we prune both train and test partitions and finally we apply the "Base Model" described in section 3 to assess its performance. In order to have more clues on the behavior of our technique, once learning has been performed on a training partition, we predict on the development and test sets as well. Figure 3 reports the results. Considering the curve related to the prediction on the split of the training set, pruning more than 35% of the nodes also increases MAP, with a peak of 69.41 when 75% of the nodes are

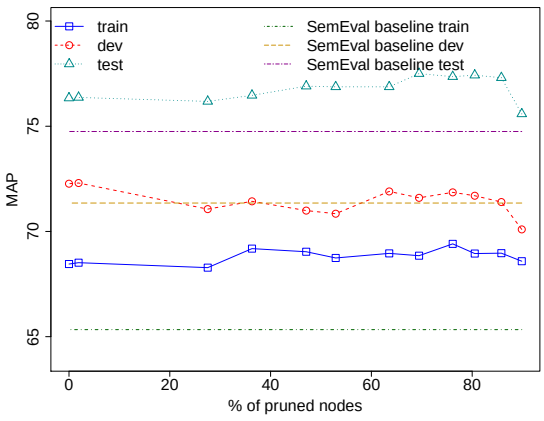
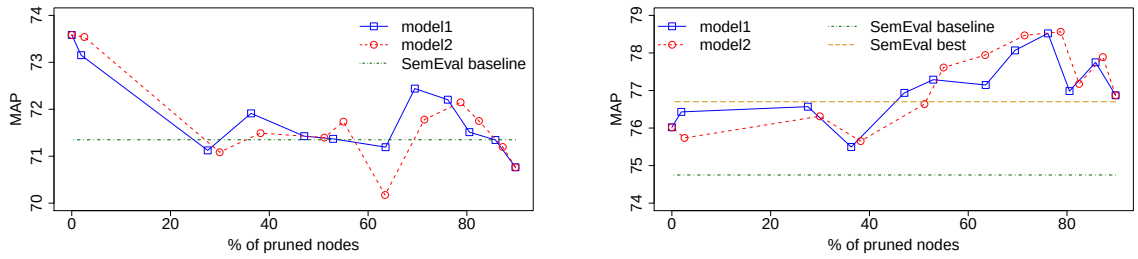


Figure 3: Cross validation for the pruning *model 1*. Each training fold is tested on the remaining part of the training set, the development and test sets.

Model	selection	Acc	P	R	F_1	MAP	AvgRec	MRR
Sentence *sim+pos*	6 sent	78.43	67.98	66.52	67.25	75.09	90.35	82.7
Pruning *model 2*	75% nodes pruned	80.57	69.96	72.96	71.43	**78.56**	91.39	85.12
Full text	-	78.71	68.58	66.52	67.54	76.02	90.70	84.64
SemEval Baseline	-	-	-	-	-	74.75	88.30	83.79
SemEval Best	-	76.57	63.53	69.53	66.39	76.70	90.31	83.02

Table 4: Performance of selected models on test.



(a) On the development set.

(b) On the test set.

Figure 4: MAP of base model with respect to the percentage of nodes pruned. Pruning performed with *model 1* and *model 2*.

pruned ($+0.958$ with respect to not applying pruning). The results on the development set do not show any increase in MAP. This may be due to annotation differences between the training/test and development partitions. At the same time the decrease is not significant. The behavior on the test set is similar to the one on the training set, although the peak is around 70% of nodes pruned this time (MAP=77.5, $+1.16$ w.r.t not applying pruning).

We performed further experiments by using all the training set for computing the weights $w()$ and, after pruning, for learning the "Base Model". This time we used the two models, *model 1* and *model 2*. Figure 4 shows the results on the development and test sets. Note that both on development and test sets there is not much difference between performing the pruning with *model 1* or *model 2*. The two plots show a similar pattern to the corresponding curves in Figure 3: on development there is no improvement, although there is a peak around 70% of nodes pruned, while on the test there is a notable improvement, the peak being MAP=78.56 ($+2.5$). Once again, our models manage to improve over SemEval's best (Franco-Salvador et al., 2016), even if it does not rely on any external knowledge. Note that the peak coincides with the threshold that we would select on cross validation on the training set, i.e. 75% of nodes pruned. We performed a student paired t-test (significance level 0.05) to check whether the MAP differences are statistically significant. It turns out that the best value on test allows us to have a statistically significant improvement with respect to the baseline.

The reduction of the size of the trees due to pruning, reduces significantly the running time, both in the learning and prediction phases. For example, the best model prunes 75% of the nodes, which translates into a reduction of the learning time of 12.7 times, from 134.51 to 10.53 minutes; the time for computing the predictions decreases from 15.14 to 9.22 minutes (1.64 times less).

Table 4 summarizes the results obtained with the different models. The tree kernel model itself, denoted as "Full text" in the table, improves over the SemEval baseline. Our best sentence selection model on development results in a worst performance on test. As aforementioned, it is not robust enough. Still it allows the pruning *model 2* system to improve by 1.86 MAP points the SemEval's best presented system.

6 Conclusions

Establishing question–question similarity for question ranking is of great importance in real-world community question answering tasks. While tree kernels are a key component of many state-of-the-art systems for question–question similarity, they are negatively affected by noisy and uninformative texts,

which are common for forum questions posted by web users. In this paper we studied the problem of selecting the most significant sentences and parse tree fragments from the question's text. For this purpose, we used unsupervised and supervised methods, exploiting standard cosine similarity models and we modeled supervised classifiers for learning effective sentence selection and tree fragments.

Our results on the recently-released SemEval 2016 cQA corpus show that supervised models can greatly improve the quality of text selection, thus reducing the size of the parse trees. An immediate consequence of this fact is that the prediction is faster (in our experiments prediction up to 50% and training more than 12 times), without any significant loss in MAP performance. In fact, it turns out that in most cases the structures removed might contain misleading information for the learning algorithm, therefore the MAP increased on the test set. Our proposed model outperforms the top systems submitted to the SemEval 2016 task on community Question Answering.

In the future, we would like to experiment with more advanced selection techniques, which have been shown successful in traditional text summarization; for instance, by using discourse structure.

Acknowledgements

This research was performed by the Arabic Language Technologies (ALT) group at HBKU's Qatar Computing Research Institute (QCRI), part of Qatar Foundation, within the Interactive sYstems for Answer Search project (Iyas).

References

Eugene Agichtein, Aristides Gionis, Carlos Castillo, Gilad Mishne, and Debora Donato. 2008. Finding high-quality content in social media with an application to community-based question answering. In *In Proceedings of WSDM*.

Fabio Aiolli, Giovanni Da San Martino, and Alessandro Sperduti. 2011. Extending Tree Kernels with Topological Information. volume 6791 of *Lecture Notes in Computer Science*, pages 142–149. Springer.

Lloyd Allison and Trevor Dix. 1986. A bit-string longest-common-subsequence algorithm. *Inf. Process. Lett.*, 23(6):305–310, December.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Shafiq Joty, Alessandro Moschitti, Fahad Al-Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. Convkn at semeval-2016 task 3: Answer and question selection for question answering on arabic and english fora. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 896–903, San Diego, California, June. Association for Computational Linguistics.

Yunbo Cao, Huizhong Duan, Chin-Yew Lin, Yong Yu, and Hsiao-Wuen Hon. 2008. Recommending questions using the mdl-based tree cut model. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 81–90, New York, NY, USA. ACM.

Giovanni Da San Martino, Alberto Barrón-Cedeño, Salvatore Romeo, and Alessandro Moschitti. 2016. Learning to re-rank questions in community question answering using advanced features. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, CIKM '16, Indianapolis, IN, October. Association for Computational Linguistics.

Cicero dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 694–699, Beijing, China, July. Association for Computational Linguistics.

Huizhong Duan, Yunbo Cao, Chin-Yew Lin, and Yong Yu. 2008. Searching questions by identifying question topic and question focus. In *ACL*, pages 156–164.

Simone Filice, Giuseppe Castellucci, Danilo Croce, Giovanni Da San Martino, Alessandro Moschitti, and Roberto Basili. 2015a. KeLP: a Kernel-based Learning Platform in java. In *The workshop on Machine Learning Open Source Software (MLOSS): Open Ecosystems*, Lille, France, July. International Conference of Machine Learning.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2015b. Structural representations for learning relations between pairs of texts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1003–1013, Beijing, China, July. Association for Computational Linguistics.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. Kelp at semeval-2016 task 3: Learning semantic relations between questions and answers. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1116–1123, San Diego, California, June. Association for Computational Linguistics.

Marc Franco-Salvador, Sudipta Kar, Thamar Solorio, and Paolo Rosso. 2016. Uh-prhlt at semeval-2016 task 3: Combining lexical and semantic-based features for community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 814–821, San Diego, California, June. Association for Computational Linguistics.

Dan Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences Computer Science and Computational Biology*. Cambridge University Press.

Yongshuai Hou, Cong Tan, Xiaolong Wang, Yaoyun Zhang, Jun Xu, and Qingcai Chen. 2015. Hitsz-icrc: Exploiting classification approach for answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 196–202, Denver, Colorado, June. Association for Computational Linguistics.

Paul Jaccard. 1901. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37(142):547–579.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005a. Finding semantically similar questions based on their answers. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '05*, page 617, New York, New York, USA, aug. ACM Press.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005b. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 84–90, Bremen, Germany.

Jiwoon Jeon, W. Bruce Croft, Joon Ho Lee, and Soyeon Park. 2006. A framework to predict the quality of answers with non-textual features. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '06*, page 228, New York, New York, USA, aug. ACM Press.

Zongcheng Ji, Fei Xu, Bin Wang, and Ben He. 2012. Question-answer topic model for question retrieval in community question answering. In *Proceedings of the 21st ACM international conference on Information and Knowledge Management*, pages 2471–2474. ACM.

Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 217–226, New York, NY, USA. ACM.

Shafiq Joty, Alberto Barrón-Cedeño, Giovanni Da San Martino, Simone Filice, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2015. Global thread-level inference for comment classification in community question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 573–578, Lisbon, Portugal, September. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

Caroline Lyon, James Malcolm, and Bob Dickerson. 2001. Detecting short passages of similar text in large document collections. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, EMNLP '01, pages 118–125, Pittsburgh, PA, USA.

Alessandro Moschitti. 2006a. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Computer Science*, pages 318–329. Springer Berlin Heidelberg.

Alessandro Moschitti. 2006b. Making tree kernels practical for natural language learning. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 113–120.

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, James Glass, and Bilal Randeree. 2015. SemEval-2015 Task 3: Answer Selection in Community Question Answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval '15, Denver, CO, USA.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. Semeval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 525–545, San Diego, California, June. Association for Computational Linguistics.

Massimo Nicosia, Simone Filice, Alberto Barrón-Cedeño, Iman Saleh, Hamdy Mubarak, Wei Gao, Preslav Nakov, Giovanni Da San Martino, Alessandro Moschitti, Kareem Darwish, Lluís Màrquez, Shafiq Joty, and Walid Magdy. 2015. QCRI: Answer selection for community question answering - experiments for Arabic and English. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval '15, Denver, Colorado, USA.

Salvatore Romeo, Giovanni Da San Martino, Alberto Barrón-Cedeño, Alessandro Moschitti, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Mitra Mohtarami, and James Glass. 2016. Neural attention for learning to rank questions in community question answering. In *Proceedings of the 26th International Conference on Computational Linguistics*, Osaka, Japan.

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer reranking. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 741–750, Portland, Oregon, USA.

Jun Sun, Min Zhang, and Chew Lim Tan. 2011. Tree sequence kernel for natural language. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, pages 921–926. AAAI Press.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2008. Learning to rank answers on large online QA collections. In *Proceedings of ACL-08: HLT*, pages 719–727, Columbus, Ohio, June. Association for Computational Linguistics.

Quan Hung Tran, Vu Tran, Tu Vu, Minh Nguyen, and Son Bao Pham. 2015. Jaist: Combining multiple features for answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 215–219, Denver, Colorado, June. Association for Computational Linguistics.

Kai Wang, Zhaoyan Ming, and Tat-Seng Chua. 2009. A syntactic tree matching approach to finding similar questions in community-based qa services. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 187–194. ACM.

Michael Wise. 1996. Yap3: Improved detection of similarities in computer program and other texts. In *Proceedings of the Twenty-seventh SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '96, pages 130–134, New York, NY, USA.

Kai Zhang, Wei Wu, Haocheng Wu, Zhoujun Li, and Ming Zhou. 2014. Question retrieval with high quality answers in community question answering. In *CIKM*.

Guangyou Zhou, Li Cai, Jun Zhao, and Kang Liu. 2011. Phrase-based translation model for question retrieval in community question answer archives. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 653–662. Association for Computational Linguistics.

Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. 2015. Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 250–259, Beijing, China, July. Association for Computational Linguistics.

Attention-Based Convolutional Neural Network for Semantic Relation Extraction

Yatian Shen, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, P.R.China
{10110240031,xjhuang}@fudan.edu.cn

Abstract

Nowadays, neural networks play an important role in the task of relation classification. In this paper, we propose a novel attention-based convolutional neural network architecture for this task. Our model makes full use of word embedding, part-of-speech tag embedding and position embedding information. Word level attention mechanism is able to better determine which parts of the sentence are most influential with respect to the two entities of interest. This architecture enables learning some important features from task-specific labeled data, forgoing the need for external knowledge such as explicit dependency structures. Experiments on the SemEval-2010 Task 8 benchmark dataset show that our model achieves better performances than several state-of-the-art neural network models and can achieve a competitive performance just with minimal feature engineering.

1 Introduction

Classifying the relation between two entities in a given context is an important task in natural language processing (NLP). Take the following sentence as an example:

Jewelry and other smaller $\langle e_1 \rangle$ *valuables* $\langle /e_1 \rangle$ *were locked in a* $\langle e_2 \rangle$ *safe* $\langle /e_2 \rangle$ *or a closet with a dead-bolt.*

Here, the marked entities "valuables" and "safe" are of the relation "Content-Container(e1; e2)".

Relation classification plays a key role in various NLP applications, and has become a hot research topic in recent years. Various machine learning based relation classification methods have been proposed for the task, based on either human-designed features (Kambhatla, 2004; Suchanek et al., 2006), or kernels (Kambhatla, 2004; Suchanek et al., 2006). Some researchers also employed the existing known facts to label the text corpora via distant supervision (Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2011; Takamatsu et al., 2012).

All of these approaches are effective because they leverage a large body of linguistic knowledge. However, these methods may suffer from two limitations. First, the extracted features or elaborately designed kernels are often derived from the output of pre-existing NLP systems, which leads to the propagation of the errors in the existing tools and hinders the performance of such systems (Bach and Badaskar, 2007). Second, the methods mentioned above do not scale well during relation extraction, which makes it very hard to engineer effective task-specific features and learn parameters.

Recently, neural network models have been increasingly focused on for their ability to minimize the effort in feature engineering of NLP tasks (Collobert et al., 2011; Zheng et al., 2013; Pei et al., 2014). Moreover, some researchers have also paid attention to feature learning of neural networks in the field of relation extraction. (Socher et al., 2012) introduced a recursive neural network model to learn compositional vector representations for phrases and sentences of arbitrary syntactic types and length. (Zeng et al., 2014; Xu et al., 2015b) utilized convolutional neural networks (CNNs) for relation classification. (Xu et al., 2015c) applied long short term memory (LSTM)-based recurrent neural networks (RNNs) along the shortest dependency path.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2526–2536, Osaka, Japan, December 11-17 2016.

We have noticed that these neural models are all designed as the way that all words are equally important in the sentence, and contribute equally to the representation of the sentence meaning. However, various situations have shown that it is not always the case. For example,

"The $\langle e_1 \rangle$ *women* $\langle /e_1 \rangle$ *that caused the* $\langle e_2 \rangle$ *accident* $\langle /e_2 \rangle$ *was on the cell phone and ran thru the intersection without pausing on the median."*, where the type of relation is "Cause-Effect(e2,e1)".

Obviously, not all words contribute equally to the representation of the semantic relation. In this sentence, "caused" is of particular significance in determining the relation "Cause-Effect", but "phone" is less correlated with the semantic of the relation of "Cause-Effect". So how to identify critical cues which determine the primary semantic information is an important task.

If the relevance of words with respect to the target entities is effectively captured, we can find critical words which determine the semantic information. Hence, we propose to introduce the attention mechanism into a convolution neural network (CNN) to extract the words that are important to the meaning of the sentence and aggregate the representation of those informative words to form a sentence vector. The key contributions of our approach are as follows:

1. We propose a novel convolution neural network architecture that encodes the text segment to its semantic representation. Compared to existing neural relation extraction models, our model can make full use of the word embedding, part-of-speech tag embedding and position embedding.

2. Our convolution neural network architecture relies on the word level attention mechanism to choose important information for the semantic representation of the relation. This makes it possible to detect more subtle cues despite the heterogeneous structure of the input sentences, enabling it to automatically learn which parts are relevant to the given class.

3. Experiments on the SemEval-2010 Task 8 benchmark dataset show that our model achieves better performance with an F1 score of 85.9% than previous neural network models, and can achieve a competitive performance with an F1 score of 84.3% just with minimal feature engineering.

2 Related Works

A variety of learning paradigms have been applied to relation extraction. As mentioned earlier, supervised methods have shown to perform well in this task. In the supervised paradigm, relation classification is considered as a multi-classification problem, and researchers concentrate on extracting complex features, either feature-based or kernel-based. (Kambhatla, 2004; Suchanek et al., 2006) converted the classification clues (such as sequences and parse trees) into feature vectors. Various kernels, such as the convolution tree kernel (Qian et al., 2008), subsequence kernel (Mooney and Bunescu, 2005) and dependency tree kernel (Bunescu and Mooney, 2005), have been proposed to solve the relation classification problem. (Plank and Moschitti, 2013) introduced semantic information into kernel methods in addition to considering structural information only. However, the reliance on manual annotation, which is expensive to produce and thus limited in quantity has provided the impetus for distant-supervision (Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2011; Takamatsu et al., 2012).

With the recent revival of interest in deep neural networks, many researchers have concentrated on using deep networks to learn features. In NLP, such methods are primarily based on learning a distributed representation for each word, which is also called a word embedding (Turian et al., 2010). (Socher et al., 2012) presented a recursive neural network (RNN) for relation classification to learn vectors in the syntactic tree path connecting two nominals to determine their semantic relationship. (Hashimoto et al., 2013) also employed a neural relation extraction model allowing for the explicit weighting of important phrases for the target task. (Zeng et al., 2014) exploited a convolutional deep neural network to extract lexical and sentence level features. These two levels of features were concatenated to form the final feature vector. (Ebrahimi and Dou, 2015) rebuilt an RNN on the dependency path between two marked entities. (Xu et al., 2015b) used the convolutional network and proposed a ranking loss function with data cleaning. (Xu et al., 2015c) leveraged heterogeneous information along the shortest dependency path between two entities. (Xu et al., 2016) proposed a data augmentation method by leveraging the directionality of relations.

Another line of research is the attention mechanism for deep learning. (Bahdanau et al., 2014) pro-

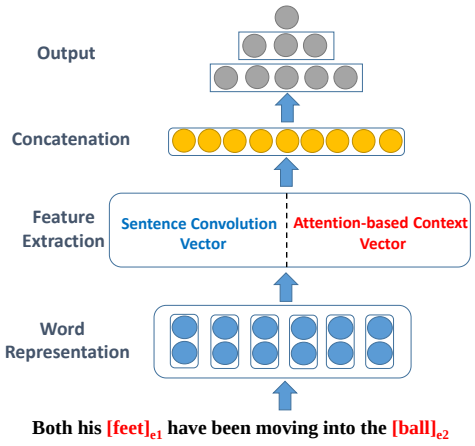


Figure 1: Architecture of the attention-based convolution neural network.

posed the attention mechanism in machine translation task, which is also the first use of it in natural language processing. This attention mechanism is used to select the reference words in the original language for words in the foreign language before translation. (Xu et al., 2015a) used the attention mechanism in image caption generation to select the relevant image regions when generating words in the captions. Further uses of the attention mechanism included paraphrase identification (Yin et al., 2015), document classification (Yang et al., 2016), parsing (Vinyals et al., 2015), natural language question answering (Sukhbaatar et al., 2015; Kumar et al., 2015; Hermann et al., 2015) and image question answering (Lin et al., 2015). (Wang et al., 2016) introduced attention mechanism into relation classification which relied on two levels of attention for pattern extraction. In this paper, we will explore the word level attention mechanism in order to discover better patterns in heterogeneous contexts for the relation classification task.

3 Methodology

Given a set of sentences $x_1, x_2, ...x_n$ and two corresponding entities, our model measures the probability of each relation r. The architecture of our proposed method is shown in Figure 1. Here, feature extraction is the main component, which is composed of sentence convolution and attention-based context selection. After feature extraction, two kinds of vectors – the sentence convolution vector and the attention-based context vector, are generated for semantic relation classification.

- **Sentence Convolution**: Given a sentence and two target entities, a convolutional neutral network (CNN) is used to construct a distributed representation of the sentence.
- **Attention-based Context Selection**: We use word-level attention to select relevant words with respect to the target entities.

3.1 Sentence Convolution

3.1.1 Input of Model

Word Embeddings. Figure 2 shows the architecture of our convolution neural network. In the word representation layer, each input word token is transformed into a vector by looking up word embeddings. (Collobert et al., 2011) reported that word embeddings learned from significant amounts of unlabeled data are far more satisfactory than the randomly initialized embeddings. Although it usually takes a long time to train the word embeddings, there are many freely available trained word embeddings. A comparison of the available word embeddings is beyond the scope of this paper. Our experiments directly utilize the embeddings trained by the CBOW model on 100 billion words of Google News (Mikolov et al., 2013).

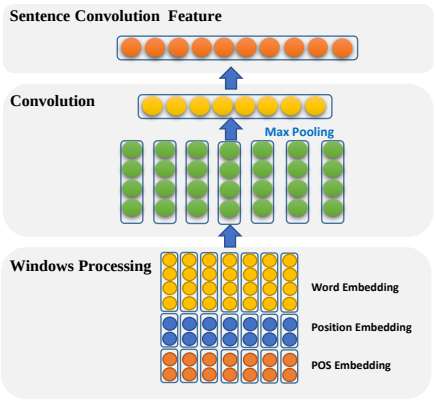


Figure 2: Architecture of convolution neural network.

Position Embeddings. In the task of relation extraction, the words close to the target entities are usually more informative in determining the relation between entities. Similar to (Zeng et al., 2014), we use position embeddings specified by entity pairs. It can help the CNN to keep track of how close each word is to the head or the tail entity, which is defined as the combination of the relative distances from the current word to the head or the tail entity. For example,

"The $\langle e_1 \rangle$ *game* $\langle /e_1 \rangle$ *was sealed in the original* $\langle e_2 \rangle$ *packing* $\langle /e_2 \rangle$ *unopened and untouched."*

In this sentence, the relative distance from the word "sealed" to the head entity "game" is 2 and the tail entity "packing" is -4. According to the above rule, we can obtain the relative distance from every word in the above sentence to each entity. We first create two relative distance files of entity e_1 and entity e_2. Then, we use the CBOW model to pretrain position embeddings on two relative distance files respectively (Mikolov et al., 2013). The dimension of position embedding is set 5.

Part-of-speech tag Embeddings. Our word embeddings are obtained from the Google News corpus, which is slightly different to the relation classification corpus. We deal with this problem by allying each input word with its POS tag to improve the robustness. In our experiment, we only take into use a coarse-grained POS category, containing 15 different tags. We use the Stanford CoreNLP Toolkit to obtain the part-of-speech tagging (Manning et al., 2014) . Then we pretrain the embeddings by the CBOW model on the taggings, and the dimension of part-of-speech tag embedding is set 10.

Finally, we concatenate the word embedding, position embedding, and part-of-speech tag embedding of each word and denote it as a vector of sequence $w = [WF, pF, POSF]$.

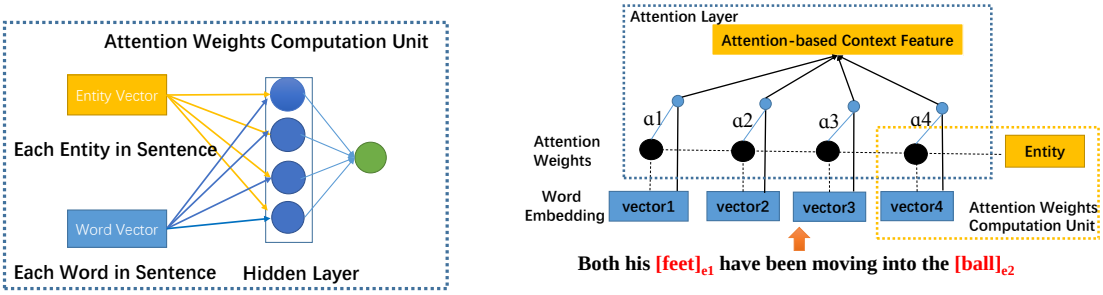
3.1.2 Convolution, Max-pooling and Non-linear Layers

In relation extraction, one of the main challenges is that, the length of the sentences is variable and important information can appear anywhere. Hence, we should merge all local features and perform relation prediction globally. Here, we use a convolutional layer to merge all these features. The convolutional layer first extracts local features with a sliding window of length l over the sentence. We assume that the length of the sliding window l is 3. Then, it combines all local features via a max-pooling operation to obtain a fixed-sized vector for the input sentence. Since the window may be outside of the sentence boundaries when it slides near the boundary, we set special padding tokens for the sentence. It means that we regard all out-of-range input vectors w_i ($i < 1$ or $i > m$) as zero vector.

Let $x_i \in R^k$ be the k-dimensional input vector corresponding to the ith word in the sentence. A sentence of length n (padded where necessary) is represented as:

$$x_{1:n} = x_1 \oplus x_2 \oplus x_3 \oplus ... \oplus x_n \tag{1}$$

where $\oplus$ is the concatenation operator. Let $x_{i:i+j}$ refer to the concatenation of words $x_i, x_{i+1}, ..., x_{i+j}$. A convolution operation involves a filter $w \in R^{hk}$, which is applied to a window of h words to produce a new feature. For example, a feature c_i is generated from a window of words $x_{i:i+h-1}$ by



(a) Attention weights computation unit.

(b) Attention layer network.

Figure 3: Architecture of attention layer network.

$$c_i = f(w \cdot x_{i:i+h-1}) \tag{2}$$

Here f is a non-linear function such as the hyperbolic tangent. This filter is applied to each possible window of words in the sentence $\{x_{1:h}, x_{2:h+1}, ..., x_{n-h+1:n}\}$ to produce a feature map:

$$\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}] \tag{3}$$

with $c \in R^{n-h+1}$. We then apply a max-overtime pooling operation over the feature map and take the maximum value $\hat{c} = max\{\boldsymbol{c}\}$ as the feature. The idea is to capture the most important feature – one with the highest value – for each feature map. This pooling scheme naturally deals with variable sentence lengths.

3.2 Attention-based Context Selection

Our attention model is applied to a rather different kind of scenario, which consist of heterogeneous objects, namely a sentence and two entities. So we seek to give our model the capability to determine which parts of the sentence are most influential with respect to the two entities of interest. For instance,

"That coupled with the $\langle e_1 \rangle$ *death* $\langle /e_1 \rangle$ *and destruction caused by the* $\langle e_2 \rangle$ *storm* $\langle /e_2 \rangle$ *was a very traumatic experience for these residents."*.

Here, the type of relation is "Cause-Effect(e2,e1)".

In this sentence, the non-entity word "caused" is of particular significance in determining the relation "Cause-Effect". Fortunately, we can exploit the fact that there is a salient connection between "caused" and "death". We introduce a word attention mechanism to quantitatively model such contextual relevance of words with respect to the target entities.

In order to calculate the weight of each word in the sentence, we need to feed each word in the sentence and each entity to a multilayer perceptron (MLP). The network structure of the attention weight computation is shown in Figure 3 (a).

Assume that each sentence contains T words. w_{it} with $t \in [1, T]$ represents the words in the ith sentence. e_{ij} with $j \in [1, 2]$ represents the jth entity in the ith sentence. We concatenate the representation of entity e_{ij} and the representation of word w_{it} to get a new representation of word t, i.e., $h_{it}^j = [w_{it}, e_{ij}]$. u_{it}^j quantifies the degree of relevance of the tth word with respect to the jth entity in the ith sentence. This relevance scoring function is computed by the MLP network between the respective embeddings of the word w_{it} and the entity e_{ij}. We named the degree of relevance as the word attention weight, namely, u_{it}^j. The calculation procedure of u_{it}^j is as follows:

$$h_{it}^j = [w_{it}, e_{ij}] \tag{4}$$

$$u_{it}^j = W_a[\tanh(W_{we}h_{it}^j + b_{we})] + b_a \tag{5}$$

The output of the attention MLP network is u_{it}^{j}. Now we can get a normalized importance weight α_{it}^{j} through a softmax function.

$$\alpha_{it}^{j} = \frac{\exp(u_{it}^{j})}{\sum_{t} \exp(u_{it}^{j})} \tag{6}$$

The architecture of our proposed attention layer is shown in Figure 3 (b). After that, we compute the sentence context vector s_{ij} about entity j as a weighted sum of the word in the sentence i based on the weights as follows:

$$s_{ij} = \sum_{t} \alpha_{it}^{j} w_{it} \tag{7}$$

The context vector s_{ij} can be seen as a high level representation of a fixed query "what is the informative word" over the words. The weight of attention MLP network is randomly initialized and jointly learned during the training process.

3.3 MLP Layer

At last, we can obtain the output of three networks, which includes the result of convolution network, and the sentence context vectors of the two entities. We then concatenate all three output vectors into a fixed-length feature vector.

The fixed length feature vector is fed to a multi-layer perceptron (MLP), which is shown in Figure 1. More specifically, first, the vector obtained is fed into a full connection hidden layer to get a more abstractive representation, and then, this abstractive representation is connected to the output layer. For the task of classification, the outputs are the probabilities of different classes, which is computed by a softmax function after the fully-connected layer. We name the entire architecture of our model **Attention-CNN**.

3.4 Model Training

The relation classification model proposed here using attention-based convolutional neural network could be stated as a parameter vector θ. To obtain the conditional probability $p(i|x, \theta)$, we apply a softmax operation over all relation types:

$$p(i|x, \theta) = \frac{e^{o_i}}{\sum_{k=1}^{n} e^{o_k}} \tag{8}$$

Given all the T training examples $(x^{(i)}; y^{(i)})$, we can then write down the log likelihood of the parameters as follows:

$$\mathcal{J}(\theta) = \sum_{i=1}^{T} \log p(y^{i}|x^{i}, \theta) \tag{9}$$

To compute the network parameter of θ, we maximize the log likelihood $\mathcal{J}$ using stochastic gradient descent (SGD). θ are randomly initialized. We implement the back-propagation algorithm and apply the following update rule:

$$\theta \leftarrow \theta + \lambda \frac{\partial \log p(y|x, \theta)}{\partial \theta} \tag{10}$$

Minibatch size	32
Word embedding size	300
Word Position Embedding size	5
Part-of-speech tag Embeddings	10
Word Window size	3
Convolution size	100
Learning rate	0.02

Table 1: Hyperparameters of our model

4 Experiments

4.1 Dataset and Evaluation Metrics

We evaluated our model on the SemEval-2010 Task 8 dataset, which is an established benchmark for relation classification (Hendrickx et al., 2009). The dataset contains 8000 sentences for training, and 2717 for testing. We split 1000 samples out of the training set for validation.

The dataset distinguishes 10 relations, and the former 9 relations are directed, whereas the "Other" class is undirected. In our experiments, We do not distinguish the direction of the relationship. To compare our results with those obtained in previous studies, we adopt the macro-averaged F1-score in our following experiments.

4.2 Parameter Settings

In this section, we experimentally study the effects of different kinds of parameters in our proposed method: Word embedding size, Word Position Embedding size, Word Window size, Convolution size, Learning rate, and Minibatch size. For the initialization of the word embeddings used in our model, we use the publicly available word2vec vectors that were trained on 100 billion words from Google News. Words not present in the set of pre-trained words are initialized randomly. The other parameters are initialized by randomly sampling from the uniform distribution in [-0.1,0.1].

Model	Feature Sets	F_1
SVM	POS, stemming, syntactic pattern, WordNet	78.8
RNN	-	74.8
	+POS, NER, WordNet	77.6
MVRNN	-	82.4
	+POS, NER, WordNet	82.4
CNN	-	78.9
(Zeng et al., 2014)	+WordNet, words around nominals	82.7
FCM	depedency parsing, NER	83.0
CR-CNN	+WordNet, words around nominals	83.7
SDP-LSTM	POS, WordNet, grammar relation	83.7
	-	**84.3**
Attention-CNN	+WordNet, words around nominals	**85.9**

Table 2: Comparison of the proposed method with existing methods in the SemEval-2010 Task 8 dataset.

For other hyperparameters of our proposed model, we take those hyperparameters that achieved the best performance on the development set. The final hyper-parameters are shown in Table 1.

4.3 Results of Comparison Experiments

To evaluate the performance of our automatically learned features, we select six approaches as competitors to be compared with our method.

Table 2 summarizes the performances of our model, SVM (Hendrickx et al., 2009), RNN, MV-RNN (Socher et al., 2012), CNN (Zeng et al., 2014), FCM (Gormley et al., 2015), CR-CNN (Xu et al., 2015b), and SDP-LSTM (Xu et al., 2015c). All of the above models adopt word embedding as representation except SVM. For fair comparison among the different model, we also add two types of lexical features, WordNet hypernyms and words around nominals, as part of the fixed length feature vector to the MLP layer.

We can observe in Table 2 that, Attention-CNN, without extra lexical features such as WordNet and words around nominals, still outperforms previously reported best systems of CR-CNN and SDP-LSTM with F1 of 83.7%, though both of which have taken extra lexical features into account. It shows that our method can learn a robust and effective relation representation. When added with the same lexical features, our Attention-CNN model obtains the result of 85.9%, significantly better than CR-CNN and

SDP-LSTM. In general, richer feature sets lead to better performance. Such neural models as RNN, MVRNN, CR-CNN and SDP-LSTM can automatically learn valuable features, and all of these models heavily depend on the result of the syntactic parsing. However, the error of syntactic parsing will inevitably inhibit the ability of these methods to learn high quality features.

Similarly, Attention-CNN, CNN, and CR-CNN all apply convolution neural network to the extraction of sentence features, but we can see from Table 2 that Attention-CNN yield a better performance of 84.3%, compared with CNN and CR-CNN. One of the reason is that the input of the three models are different. Our model uses word embeddings, position embeddings, part-of-speech embeddings as input. CNN also leverages position embeddings and lexical features. CR-CNN makes use of heterogeneous information along the shortest dependency path between two entities. Our experiments verify that the part-of-speech embeddings used by us contain rich semantic information. On the other hand, our proposed Attention-CNN model can still yield higher F1 without prior NLP knowledge. The reason should be due to that word level attention mechanism is able to better choose which parts of the sentence are more discriminative with respect to the two entities of interest.

Feature Sets	F_1
WF	74.5
+pF	80.7
+POSF	82.6
+WA	84.3
+WA+(Lexical Feature)	85.9

Table 3: Score obtained for various sets of features on the test set. The bottom portion of the table shows the best combination of all the features.

4.4 Effect of Different Feature Component

Our network model primarily contains four sets of features, "Word Embeddings (WF)","Position Embeddings (pF)","Part-of-speech tag Embeddings (POSF)", and "Word Attention (WA)". We performed ablation tests on the four sets of features in Table 3 to determine which type of features contributed the most. From the results we can observe that our learned position embedding features are effective for relation classification. The F1-score is improved remarkably when position embedding features are added. POS tagging embeddings are comparatively more informative, which can boost the F1 by 1.9%. The system achieves approximately 2.3% improvements when adding Word Attention. When all features are combined, we achieve the best result of 85.9%.

4.5 Visualization of Attention

In order to validate whether our model is able to select informative words in a sentence or not, we visualize the word attention layers in Figure 4 for several data from test sets.

Every line in Figure 4 shows a sentence. The size of a word denotes the importance of it. We normalize the word weight to make sure that only important words are emphasized. Given the following sentence as an example,

"The burst has been caused by water hammer pressure."

we can find that the word "caused" was assigned the highest attention score, while words such as "burst" and "pressure" also are important. This makes sense in light of the ground-truth labeling as a "Cause-Effect" relationship. Additionally, we observe that words like "The", "has" and "by" have low attention scores. These are indeed rather irrelevant with respect to the "Component-Whole" relationship.

5 Conclusion

In this paper, we propose an attention-based convolutional neural network architecture for semantic relation extraction. Here, the convolutional neural network architecture is used to extract the features

Relation	Representation of Word Attention Weight
Instrument-Agency	The author of a keygen uses a disassembler to look at the raw assembly code
Message-Topic	The Pulitzer Committee issues an official citation explaining the reasons for the award
Cause-Effect	The burst has been caused by water hammer pressure
Instrument-Agency	Even commercial networks have moved into high-definition broadcast
Component-Whole	The girl showed a photo of apple tree blossom on a fruit tree in the Central Valley
Member-Collection	They tried an assault of their own an hour later, with two columns of sixteen tanks backed by a battalion of Panzer grenadiers

Figure 4: Visualization of Attention.

of the sentence. Our model can make full use of word embedding, part-of-speech tag embedding and position embedding information. Meanwhile, word level attention mechanism is able to better determine which parts of the sentence are most influential with respect to the two entities of interest. Experiments on the SemEval-2010 Task 8 benchmark dataset show that our model achieves better performances than several state-of-the-art systems.

In the future, we will focus on exploring better neural network structure about feature extraction in relation extraction. Meanwhile, because end-to-end relation extraction is also an important problem, we will seek better methods for completing entity and relation extraction jointly.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011 and 61472088), the National High Technology Research and Development Program of China (No. 2015AA015408).

References

Nguyen Bach and Sameer Badaskar. 2007. A review of relation extraction. *Literature review for Language and Statistics II*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Javid Ebrahimi and Dejing Dou. 2015. Chain based rnn for relation classification. In *Proceedings of the Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL: Association for Computational Linguistics*, pages 1244–1249.

Matthew R Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. *arXiv preprint arXiv:1505.02419*.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In *EMNLP*, pages 1372–1376.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 22. Association for Computational Linguistics.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 899–907.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Raymond J Mooney and Razvan C Bunescu. 2005. Subsequence kernels for relation extraction. In *Advances in neural information processing systems*, pages 171–178.

Wenzhe Pei, Tao Ge, and Chang Baobao. 2014. Maxmargin tensor neural network for chinese word segmentation. In *Proceedings of ACL*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *ACL (1)*, pages 1498–1507.

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 697–704. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Fabian M Suchanek, Georgiana Ifrim, and Gerhard Weikum. 2006. Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 712–717. ACM.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 721–729. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015a. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2(3):5.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015b. Semantic relation classification via convolutional neural networks with simple negative sampling. *arXiv preprint arXiv:1506.07650*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015c. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (to appear)*.

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2016. Improved relation classification by deep recurrent neural networks with data augmentation. *arXiv preprint arXiv:1601.03651*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *EMNLP*, pages 647–657.

Table Filling Multi-Task Recurrent Neural Network for Joint Entity and Relation Extraction

Pankaj Gupta
Corporate Technology, Siemens AG
CIS, University of Munich (LMU)
Munich, Germany
gupta.pankaj.ext@siemens.com
pankaj.gupta@campus.lmu.de

Hinrich Schütze
CIS, University of Munich (LMU)
inquiries@cis.lmu.org

Bernt Andrassy
Corporate Technology, Siemens AG
Munich, Germany
bernt.andrassy@siemens.com

Abstract

This paper proposes a novel context-aware joint entity and word-level relation extraction approach through semantic composition of words, introducing a Table Filling Multi-Task Recurrent Neural Network (TF-MTRNN) model that reduces the entity recognition and relation classification tasks to a table-filling problem and models their interdependencies. The proposed neural network architecture is capable of modeling multiple relation instances without knowing the corresponding relation arguments in a sentence. The experimental results show that a simple approach of piggybacking candidate entities to model the label dependencies from relations to entities improves performance.

We present state-of-the-art results with improvements of 2.0% and 2.7% for entity recognition and relation classification, respectively on CoNLL04 dataset.

1 Introduction

Relation classification is defined as the task of predicting the semantic relation between the annotated pairs of nominals (also known as relation arguments). These annotations, for example named entity pairs participating in a relation are often difficult to obtain. Traditional methods are often based on a pipeline of two separate subtasks: Entity Recognition (ER[1]) and Relation Classification (RC), to first detect the named entities and then performing relation classification on the detected entity mentions, therefore ignoring the underlying interdependencies and propagating errors from the entity recognition to relation classification. The two subtasks together are known as End-to-End relation extraction.

Relation classification is treated as a sentence-level multi-class classification problem, which often assume a single relation instance in the sentence. It is often assumed that entity recognition affects the relation classification, but it is not affected by relation classification. Here, we reason with experimental evidences that the latter is not true. For example, in Figure 1, relation *Work_For* exists between *PER* and *ORG* entities, *ORGBased_in* between *ORG* and *LOC*, while *Located_In* between *LOC* and *LOC* entities. Inversely, for a given word with associated relation(s), the candidate entity types can be detected. For example, in Figure 2, for a given relation, say *Located_in*, the candidate entity pair is (*LOC*, *LOC*). Therefore, the two tasks are interdependent and optimising a single network for ER and RC to model the interdependencies in the candidate entity pairs and corresponding relations is achieved via the proposed joint modeling of subtasks and a simple piggybacking approach.

Joint learning approaches (Roth and Yih, 2004; Kate and Mooney, 2010) built joint models upon complex multiple individual models for the subtasks. (Miwa and Sasaki, 2014) proposed a joint entity and relation extraction approach using a history-based structured learning with a table representation; however, they explicitly incorporate entity-relation label interdependencies, use complex features and search heuristics to fill table. In addition, their state-of-the-art method is structured prediction and not based on neural network frameworks. However, *deep learning* methods such as recurrent and convolutional neural networks (Zeng et al., 2014; Zhang and Wang, 2015; Nguyen and Grishman, 2015) treat relation

[1]Entity Recognition (ER) = Entity Extraction (EE); Relation Classification (RC) = Relation Extraction (RE)

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2537–2547, Osaka, Japan, December 11-17 2016.

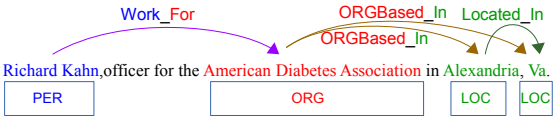


Figure 1: An entity and relation example (CoNLL04 data). *PER*: Person, *ORG*: Organization, *LOC*: Location. Connections are: *PER* and *ORG* by *Work_For*; *ORG* and *LOC* by *OrgBased_In*; *LOC* and *LOC* by *Located_In* relations.

	PER	**LOC**	**ORG**	**Other**
PER	KILL	Live_In	Work_For	⊥
LOC	Live_In	Located _In	ORG Based_In	⊥
ORG	Work_For	ORG Based_In	⊥	⊥
Other	⊥	⊥	⊥	⊥

Figure 2: Entity-Relation dependencies (CoNLL04 dataset).

	Richard	Kahn	,	officer	for	the	American	Diabetes	Association	in	Alexandria	,	Va	.
Richard	*B-PER*, ⊥													
Kahn	⊥	*L-PER*, ⊥												
,	⊥	⊥	*O*, ⊥											
officer	⊥	⊥	⊥	*O*, ⊥										
for	⊥	⊥	⊥	⊥	*O*, ⊥									
the	⊥	⊥	⊥	⊥	⊥	*O*, ⊥								
Ameriacan	⊥	⊥	⊥	⊥	⊥	⊥	*B-ORG*, ⊥							
Diabetes	⊥	⊥	⊥	⊥	⊥	⊥	⊥	*I-ORG*, ⊥						
Association	⊥	*Work_For*	⊥	⊥	⊥	⊥	⊥	⊥	*L-ORG*, ⊥					
in	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	*O*, ⊥				
Alexandria	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	*ORGBased_In*	⊥	*U-LOC*, ⊥			
,	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	*O*, ⊥		
Va	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	*ORGBased_In*	⊥	*Located_In*	⊥	*U-LOC*, ⊥	
.	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	*O*, ⊥

Table 1: Entity-Relation Table for the example in Figure 1. Demonstrates the word-level relation classification via a Table-Filling problem. The symbol (⊥) indicates *no_relation* word pair. Relations are defined on the words, instead of entities. The diagonal entries have the entity types and ⊥ relations to the words itself, while the off-diagonal entries are the relation types.

classification as a sentence-level multi-class classification, and rely on the relation arguments provided in the sentence. Therefore, they are incapable in handling multiple relation instances in a sentence and can not detect corresponding entity mention pairs participating in the relation detected.

We tackle the limitations of joint and deep learning methods to detect entities and relations. The contributions of this paper are as follows:

1. We propose a novel Table Filling Multi-task Recurrent Neural Network to jointly model entity recognition and relation classification tasks via a unified multi-task recurrent neural network. We detect both entity mention pairs and the corresponding relations in a single framework with an entity-relation table representation. It alleviates the need of search heuristics and explicit entity and relation label dependencies in joint entity and relation learning. As far as we know, it is the first attempt to jointly model the interdependencies in entity and relation extraction tasks via multi-task recurrent neural networks.

   We present a word-level instead sentence-level relation learning via word-pair compositions utilising their contexts via Context-aware RNN framework. Our approach has significant advantage over state-of-the-art methods such as CNN and RNN for relation classification, since we do not need the marked nominals and can model multiple relation instances in a sentence.

2. Having named-entity labels is very informative for finding the relation type between them, and vice versa having the relation type between words eases problem of named-entity tagging. Therefore, a simple approach to piggyback candidate named entities for words (derived from the associated relation type(s) for each word) to model label dependencies improves the performance of our system. In addition, the sequential learning approach in the proposed network learns entity and relation label dependencies via sharing model parameters and representations, instead modeling them explicitly.

3. Our approach outperforms the state-of-the-art method by 2.0% and 2.7% for entity recognition and relation classification, respectively on CoNLL04 dataset.

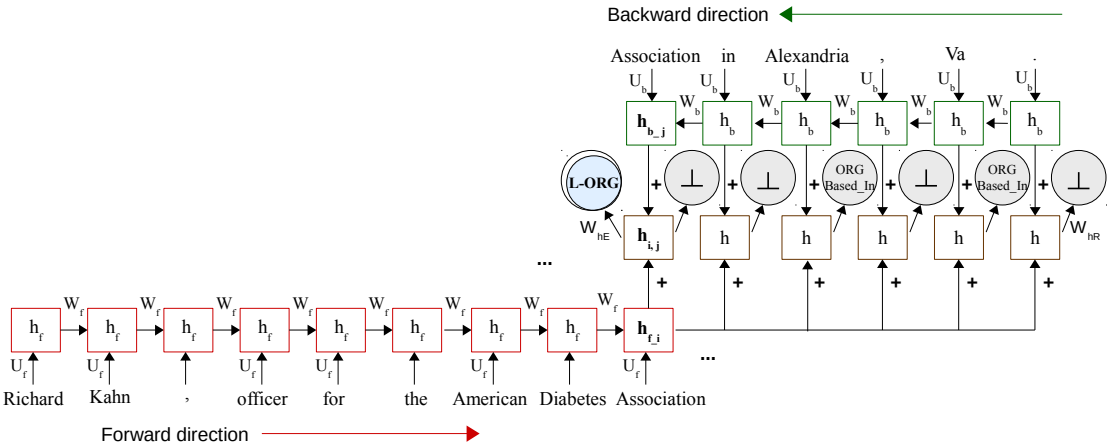


Figure 3: The Table Filling Multi-Task Recurrent Neural Network (TF-MTRNN) for joint entity and word-level relation extraction. Overlapping circle: Entity labels; Single circle: Relation label. In the above illustration, the word *Association* at $t = i$ (where; $t = 0, ..., i, ..., N$) from forward network is combined with each of the remaining words in the sequence (Figure 1), obtained from backward network at each time step, $j = i, ..., N$. Similarly, perform all possible word pair compositions to obtain Table 1. *ORGBased_In* relation in each word-pairs: (*Association*, *Alexandria*) and (*Association*, *Va*).

2 Methodology

2.1 Entity-Relation Table

As the backbone of our model we adopt the table structure proposed by Miwa and Sasaki (2014), shown in Table 1. This structure allows an elegant formalization of joint entity and relation extraction because both entity and relation labels are defined as instances of binary relations between words w_i and w_j in the sentence. An entity label is such a binary relationship for $i = j$, i.e., a cell on the diagonal. A relation label is such a binary relationship for $i \neq j$, i.e., an off-diagonal cell. To eliminate redundancy, we stipulate that the correct label for the pair (w_i, w_j) is relation label r if and only if $i \neq j$, w_i is the last word of a named entity e_i, w_j is the last word of a named entity e_j and $r(e_i, e_j)$ is true.[2] We introduce the special symbol $\perp$ for "no_relation", i.e., no relation holds between two words.

Apart from the fact that it provides a common framework for entity and relation labels, another advantage of the table structure is that modeling multiple relations per sentence comes for free. It simply corresponds to several (more than one) off-diagonal cells being labeled with the corresponding relations.

2.2 The Table Filling Multi-Task RNN Model

Formally, our task for a sentence of length n is to label $n(n+1)/2$ cells. The challenge is that the labeling decisions are highly interdependent. We take a deep learning approach since deep learning models have recently had success in modeling complex dependencies in NLP. More specifically, we apply recurrent neural networks (RNNs) (Elman, 1990; Jordan, 1986; Werbos, 1990) due to their success on complex NLP tasks like machine translation and reasoning.

To apply RNNs, we order the cells of the table into a sequence as indicated in Figure 4 and label – or "fill" – the cells one by one in the order of the sequence. We call this approach *table filling*.

More specifically, we use a bidirectional architecture (Vu et al., 2016b), a forward RNN and a backward RNN, to fill each cell (i, j) as shown in Figure 3. The forward RNN provides a representation of the history $w_1, \ldots, w_i$. The backward network provides a representation of the following context $w_j, \ldots, w_{|s|}$. The figure shows how the named entity tag for "Association" is computed. The forward RNN is shown as the sequence at the bottom. h_{f_i} is the representation of the history and h_{b_j} is the representation of the following context. Both are fed into $h_{i,j}$ which then predicts the label L-ORG. In this case, $i = j$. The prediction of a relation label is similar, except that in that case $i \neq j$.

[2]Relation types (excluding $\perp$) exist only in the word pairs with entity types: (L-*, L-*), (L-*, U-*), (U-*, L-*) or (U-*, U-*), where * indicates any entity type encoded in BILOU (Begin, Inside, Last, Outside, Unit) scheme.

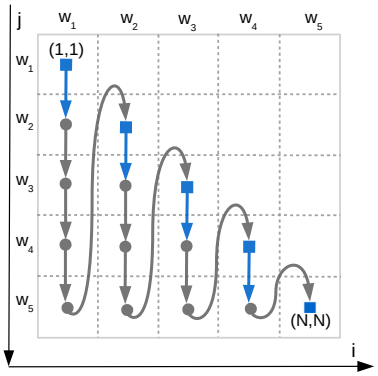


Figure 4: Table Filling/Decoding Order. Filled squares in blue represent both entity and relation label assignments, while filled circles in gray represent only relation label assignments, analogous to entries in Table 1. (i, j) is the cell index in the table, where i and j are the word indices in the sequence.

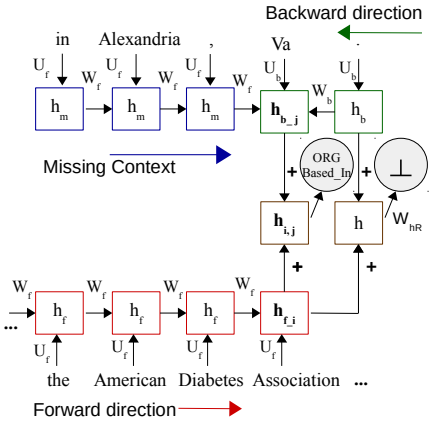


Figure 5: The context-aware TF-MTRNN model. (...) indicates the remaining word pair compositions (Table 1).

Our proposed RNN based framework jointly models the entity and relation extraction tasks to learn the correlations between them, by sharing the model parameters and representations. As illustrated in Figure 3, we use two separate output nodes and weight matrices each for entity and relation classification. An entity label is assigned to a word, while a relation is assigned to a word pair; therefore, EE is performed only when the same words from forward and backward networks are composed.

Dynamics of the proposed TF-MTRNN architecture (Figure 3) are given below:

$$\begin{aligned} s_{R_{i,j\in i:N}} &= g(W_{hR}h_{i,j}); \qquad s_{E_{i,j=i}} = g(W_{hE}h_{i,j}); \qquad h_{i,j} = h_{f_i} + h_{b_j} \\ h_{f_i} &= f(U_f w_i + W_f h_{f_{i-1}}); \qquad h_{b_j} = f(U_b w_j + W_b h_{b_{j+1}}) \end{aligned} \tag{1}$$

where i and j are the time-steps of forward and backward networks, respectively. ith word in the sequence is combined with every jth word, where $j = i, ..., N$ (i.e. combined with itself and the following words in the sequence). N is the total number of words in the sequence. For a given sequence, $s_{R_{i,j}}$ and $s_{E_{i,j}}$ represent the output scores of relation and entity recognition for ith and jth word from forward and backward networks, respectively. Observe that EE is performed on the combined hidden representation $h_{i,j}$, computed from the composition of representations of the same word from forward and backward networks, therefore $i = j$ and resembling the diagonal entries for entities in Table 1. h_{f_i} and h_{b_j} are hidden representations of forward and backward networks, respectively. W_{hR} and W_{hE} are weights between hidden layers ($h_{i,j}$) and the output units of relation and entity, respectively. f and g are activation and loss functions. Applying $argmax$ to $s_{R_{i,j\in i:N}}$ and $s_{E_{i,j=i}}$ gives corresponding table entries for relations and entities, in Table 1 and Figure 4.

2.3 Context-aware TF-MTRNN model

In Figure 3, we observe that when hidden representations for the words *Association* and *Va* are combined, the middle context i.e. all words in the sequence occurring between the word pair in composition are missed. Therefore, we introduce a third direction in the network (Figure 5) with missing context (i.e. *in Alexandria ,*) to accumulate the full context in combined hidden vectors ($h_{i,j}$).

Dynamics of the context-aware TF-MTRNN is similar to Eq. 1, except h_{b_j}, in Figure 5:

$$\begin{aligned} h_{b_j} &= f(U_b w_j + W_b h_{b_{j+1}} + U_f h_{m_{t=T}}) \\ h_{b_{j+1}} &= f(U_b w_{j+1} + W_b h_{b_{j+2}}); \qquad h_{m_t} = f(U_f w_t + W_f h_{m_{t-1}}) \end{aligned} \tag{2}$$

where h_{b_j} is the hidden representation in backward network obtained from the combination of jth word and contexts from backward network and from missing direction, $t = (i + 1, ..., T = j - 1)$, where i and j are the time-steps for forward and backward networks, respectively. $h_{m_{t=i}}$ is initialized with zeros similar to forward and backward networks. There is no missing context when $i = 0$ and $j = 0$ i.e. w_t is NULL and therefore, we introduce an artificial word *PADDING* and use its embedding to initialise w_t.

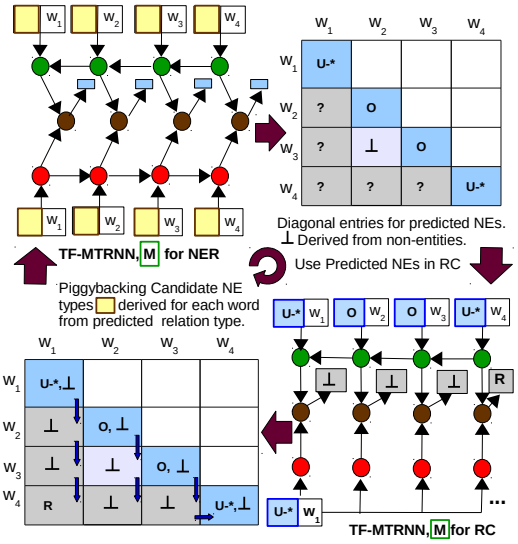


Figure 6: End-to-End Relation Extraction by jointly modeling entity and relation in a unified Multi-task RNN framework, M (TF-MTRNN) and filling an Entity-Relation table. Entity-relation interdependencies modeled by parameter sharing and piggybacking (Section 2.4 and Figure 7). NE: Named Entity; U-* and O: NE in BILOU format; ?:Relation to determine.

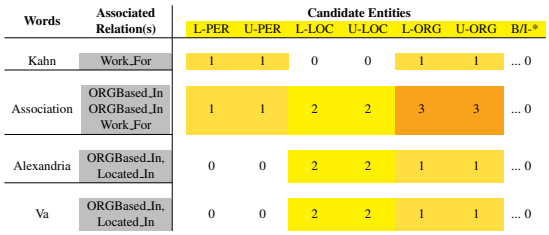


Figure 7: Piggybacking approach to model label dependencies from relations to entities. We do not list all words due to space limitation. * indicates any entity type. Highlight for counts indicate candidate entity importance for corresponding words.

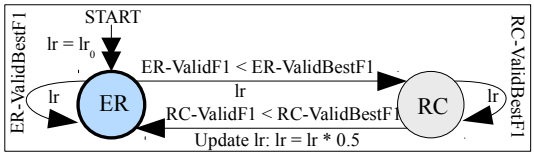


Figure 8: State Machine driven Multi-task Learning. ER:Entity Recognition; RC:Relation Classification; lr:learning rate; $ER - ValidBestF1$:Best entity recognition F_1 score on validation set.

2.4 Piggybacking for Entity-Relation Label Dependencies

Having named-entity labels is very informative for finding the relation type between them, and vice versa having the relation type between words eases problem of named-entity tagging. We model these label interdependencies during the end-to-end relation extraction in Figure 6, where the input vector at time step, t is given by -

$$input_t = \{C_{RE}, E_{ER}, W_{emb}\} \tag{3}$$

where C_{RE} is the count vector to model relation to entity dependencies, E_{ER} is the one-hot vector for predicted entities to model entity to relation dependencies and W_{emb} is the word embedding vector. Therefore, the input vector at each time step, t is the concatenation of these three vectors.

To model *entity to relation* dependency, the TF-MTRNN model, M for NER (Figure 6) first computes entity types, which are represented by diagonal entries of entity-relation table. Each predicted entity type E_{ER} (filled blue-color boxes) is concatenated with its corresponding word embedding vector W_{emb} and then input to the same model, M for relation classification.

To model *relation to entity* dependency, we derive a list of possible candidate entity tags for each word participating in a relation(s), except for $\perp$ relation type. Each word associated with a relation type(s) is determined from relation classification (RC) step (Figure 6). Figure 7 illustrates the entity type count vector for each word of the given sentence (Figure 1). For example, the word *Alexandria* participates in the relation types: *ORGBased_In* and *Located_In*. Possible entity types are {*U-ORG*, *L-ORG*, *U-LOC*, *L-LOC*} for *ORGBased_In*, while {*U-LOC*, *L-LOC*} for *Located_In*. We then compute a count vector C_{RE} from these possible entity types. Therefore, *U-LOC* and *L-LOC* each with occurrence 2, while *U-ORG* and *L-ORG* each with occurrence 1 (Figure 7). The candidate entity types as count vector (filled-yellow color box) for each word is piggybacked to model, M for entity learning by concatenating it with corresponding word embedding vector W_{emb}. This simple approach of piggybacking the count vectors of candidate entities enables learning label dependencies from *relation to entity* in order to improve entity extraction. In addition, multi-tasking by sharing parameters and adapting shared embeddings within a unified network enables learning label interdependencies.

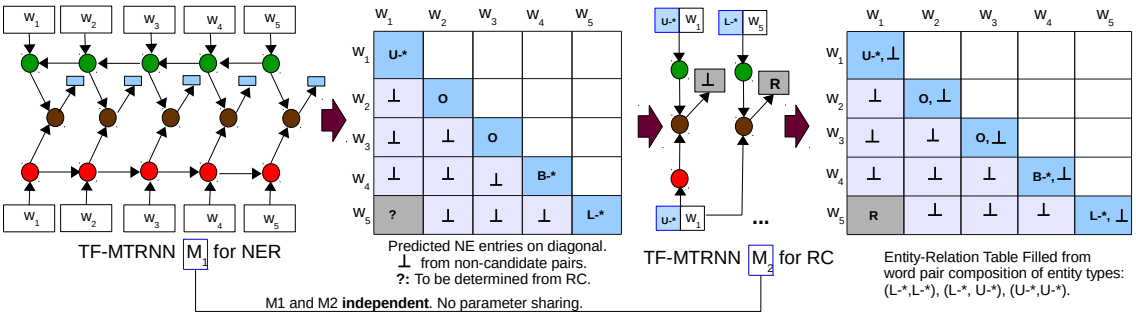


Figure 9: Pipeline Approach in End-to-End Relation Extraction.

2.5 Ranking Bi-directional Recurrent Neural Network (R-biRNN)

Ranking loss has been used in neural architectures (dos Santos et al., 2015) and (Vu et al., 2016b) to handle *artificial* classes. In our experiments, for a given sentence x with class label y^+, the competitive class c^- is chosen the one with the highest score among all competitive classes during SGD step. The basic principle is to learn to maximize the distance between the true label y^+ and the best competitive label c^- for a given data point x. We use the ranking loss to handle the two artificial classes i.e. 'O' and $\perp$ in entity and relation types, respectively. The ranking objective function is defined as-

$$L = \log(1 + \exp(\gamma(m^+ - s_\theta(x)_{y^+}))) + \log(1 + \exp(\gamma(m^- + s_\theta(x)_{c^-})));$$
$$c^- = \arg \max_{c \epsilon C; c \neq y^+} s_\theta(x)_c \tag{4}$$

where $s_\theta(x)_{y^+}$ and $s_\theta(x)_{c^-}$ are the scores for positive y^+ and the most competitive c^- classes. γ controls the penalization of the prediction errors while hyperparameters m^+ and m^- are the margins for the true and competitive classes. We set $\gamma = 2, m^+ = 2.5, m^- = 0.5$, following (Vu et al., 2016b).

The unified architecture (Figure 3) can be viewed as being comprised of two individual models, each for NER and RE (Figure 6). We illustrate that the R-biRNN (Figure 12 in Appendix A) is integrated in TF-MTRNN (Figure 3) and therefore, the unified model leverages R-biRNN (Vu et al., 2016b) effectiveness for entity extraction, where the full context information is availed from the forward and backward network at each input word vector along with the ranking loss at each output node. Figure 12 corresponds to the diagonal entries for named entities in Table 1 and enables entity-entity label dependencies (Miwa and Sasaki, 2014) via sequential learning.

3 Model Training

3.1 End-to-End Relation Extraction

In CoNLL04, more than 99% of the whole word pairs lie in the no_relation class. Therefore, named-entity candidates are required to choose the candidate word pairs in relation learning. In Figure 6 and Figure 9, we demonstrate the joint and pipeline approach for end-to-end relation extraction.

In Figure 6, the candidate relation pairs are chosen by filtering out the non-entities pairs. Therefore, in entity-relation table, we insert 'no_relation' label for the non-entities pairs and RC is not performed. Note that a word pair is chosen for RC in which at least one word is an entity. It allows the model M to correct itself at NER by piggybacking candidate named entities (Figure 7). In addition, it reduces a significant number of non-relation word pairs and does not create a bias towards the no_relation class. However, in Figure 9, the two independent models, M_1 and M_2 are trained for NER and RC, respectively. In pipeline approach, the only relation candidates are word pairs with (U-*, U-*),(L-*, L-*) or (U-*, L-*) entity types. Therefore, only w_1 and w_5 from word sequence are composed in M_2 for RC subtask.

	Features	CoNLL04 Dataset					
		NER			RE		
		P	R	F_1	P	R	F_1
separate	basic	.865	.902	.883	.360	.403	.376
	+POS	.877	.906	.892	.440	.376	.395
	+CF	.906	.914	.910	.454	.390	.410
	+CTX				.499	.434	.453
pipeline	basic				.641	.545	.589
	+POS				.663	.555	.604
	+CF				.661	.585	.621
	+CTX				.736	.616	.671
joint	basic	.885	.889	.888	.646	.531	.583
	+POS	.904	.908	.906	.673	.531	.594
	+CF	.913	.914	.914	.691	.562	.620
	+CTX				.745	.595	.661
	+p'backing	.925	.921	.924	.785	.630	.699
	+ensemble	.936	.935	.936	.832	.635	.721

Figure 10: CoNLL04 dataset: Performance on test set for NER and RE; RE in pipeline always used predicted NEs. POS: part-of-speech; CF: capital features; CTX: context awareness (Figure 5); p'backing: piggybacking predicted and candidate entities in RE and NER, respectively; ensemble: majority vote.

Figure 11: T-SNE view of the semantic entity-relation space for the combined hidden representations of each word pair composition. Relations: (0:*LIVEIN*, 1:*ORGBASEDIN*, 2:*LOCATEDIN*, 3:*WORKFOR*, 4:*KILL*, 5:*NORELATION*). Entity-pair and relation denoted by E1-RELATION-E2 and/or count in [0-5]. 5: misclassified entity-pairs.

3.2 Word Representation and Features

Each word is represented by concatenation of pre-trained 50-dimensional word embeddings[3] (Turian et al., 2010) with N-gram, part-of-speech (POS), capital feature (CF: all-capitalized; initial-capitalized) and piggybacked entity vectors (Section 2.4). The word embeddings are shared across entity and relation extraction tasks and are adapted by updating them during training. We use 7-gram $(w_{t-3}w_{t-2}w_{t-1}w_tw_{t+1}w_{t+2}w_{t+3})$ obtained by concatenating corresponding word embeddings.

3.3 State Machine driven Multi-tasking

Multi-task training is performed via switching across multiple tasks in a block of training steps. However, we perform switches between ER and RC subtasks based on the performance of each task on the common validation set and update learning rate only when task is switched from RC to ER (Figure 8). ER is the task to start for multi-tasking and ER/RC is switched in the following training step, when their $ValidF1$ score is not better than $BestValidF1$ score of previous steps on the validation set.

4 Evaluation and Analysis

4.1 Dataset and Experimental Setup

We use CoNLL04[4] corpus of Roth and Yih (2004). Entity and relation types are shown in Figure 2. There are 1441 sentences with at least one relation. We randomly split these into training (1153 sentences) and test (288 sentences), similar to Miwa and Sasaki (2014). We release this train-test split at `https://github.com/pgcool/TF-MTRNN/tree/master/data/CoNLL04`. We introduce the pseudo-label $\perp$ "no_relation" for word pairs with no relation.

To tune hyperparameters, we split (80-20%) the training set (1153 sentences) into $train$ and validation (dev) sets. All final models are trained on $train$+dev. Our evaluation measure is F_1 on entities and relations. An entity is marked correct if NE boundaries and entity type[5] are correct. A relation for a word pair is marked correct if the NE boundaries and relation type are correct. However, in separate approach, a relation for a word pair is marked correct if the relation type is correct.

[3] with a special token PADDING. Also, used when there is no missing context.

[4] conll04.corp at `cogcomp.cs.illinois.edu/page/resource_view/43`

[5] For multi-word entity mention, an entity is marked correct if atleast one token is tagged correctly.

	Roth&Yih			Kate&Mooney			Miwa&Sasaki			TF-MTRNN		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Person	.891	.895	.890	.921	.942	.932	.931	.948	.939	.932	.988	.959
Location	.897	.887	.891	.908	.942	.924	.922	.939	.930	.974	.956	.965
Organization	.895	.720	.792	.905	.887	.895	.903	.896	.899	.873	.939	.905
(Average)	.894	.834	.858	.911	.924	. 917	.919	.927	.923	.926	.961	.943
Live_In	.591	.490	.530	.664	.601	.629	.819	.532	.644	.727	.640	.681
OrgBased_In	.798	.416	.543	.662	.641	.647	.768	.572	.654	.831	.562	.671
Located_In	.539	.557	.513	.539	.557	.513	.821	.549	.654	.867	.553	.675
Work_For	.720	.423	.531	.720	.423	.531	.886	.642	.743	.945	.671	.785
Kill	.775	.815	.790	.775	.815	.790	.933	.797	.858	.857	.894	.875
(Average)	.685	.540	.581	.672	.607	.622	.845	.618	.710	.825	.664	.737

Table 2: State-of-the-art comparison for EE and RE on CoNLL04 dataset.

4.2 Results

Figure 10 shows results for NER[6] and RE. All models use n-grams for $n = 7$ (Section 3.2). Embedding dimensionality is 50. The notation "+" (e.g., +POS) at the beginning of a line indicates that the model of this line is the same as the model on the previous line except that one more model element (e.g., POS) is added. The separate NER model performs NER only. The separate RE model performs RE only, without access to NER results. The pipeline RE model takes the results of the separate NER model and then performs RE. The joint model is trained jointly on NER and RE. For compactness, we show the results of *two different models* (an NER model and an RE model) in the separate part of the table; in contrast, results for a *single model* – evaluated on both NER and RE – are shown in the joint part.

We make the following observations based on Figure 10. (i) All of our proposed model elements (POS, CF, CTX, piggybacking, ensemble) improve performance, in particular CTX and piggybacking provide large improvements. (ii) Not surprisingly, the pipeline RE model that has access to NER classifications performs better than the separate RE model. (iii) The joint model performs better than separate and pipeline models, demonstrating that joint training and decoding is advantageous for joint NER and RE. (iv) Majority voting[7] (ensemble) results in a particularly large jump in performance and in the overall best performing system; F_1 is .936 for NER and .721 for RE, respectively.

4.3 Comparison with Other Systems

Our end-to-end relation extraction system outperform the state-of-the-art results. We compare the entity and relation extraction performance of our model with other systems (Roth and Yih, 2007; Kate and Mooney, 2010; Miwa and Sasaki, 2014). (Roth and Yih, 2007) performed 5-fold cross validation on the complete corpus (1441 sentences), while (Miwa and Sasaki, 2014) performed 5-cross validation on the data set, obtained after splitting the corpus. We report our results on the test set from random split (80-20%) of the corpus, similar to (Miwa and Sasaki, 2014). Since, the standard splits were not available, we cannot directly compare the results, but our proposed model shows an improvement of 2.0% and 2.7% in F_1 scores for entity and relation extraction tasks, respectively (Table 2).

[6]Our NER model reports 86.80% F1 score, comparable to 86.67% from (Lample et al., 2016) on CoNLL03 shared task using the standard NER evaluation script with strict multi-word entity evaluation, and adapted for BILOU encoding.

[7]Randomly pick one of the most frequent classes, in case of a tie

4.4 Word pair Compositions (T-SNE)

Using t-SNE (der Maaten and Hinton, 2008), we visualize the hidden representations obtained on the composition of hidden vectors of every two words (word pair) in the sentence via TF-MTRNN model. In Figure 11, we show all data points i.e. word pair compositions, leading to natural relations (except $\perp$ denoted by 5). We observe that the entity mention pairs with common relation types form clusters corresponding to each relation in the semantic entity-relation space. We observe that the relation clusters with common entity type lie close to each other, for example, *KILL* has (*PER*, *PER*) entity pairs, which is close to relation cluster *LIVEIN* and *WORKFOR*, in which one of the entities i.e. *PER* is common. While, *KILL* relation cluster is at a distance from *LOCATEDIN* cluster, since they have no common entity.

4.5 Hyperparamter Settings

We use stochastic gradient descent with L2 regularization with a weight of .0001. The initial learning rate for entity and relation extraction is .05 with hidden layer size 200. The learning rate update and task switching is driven by the state machine (Figure 8). Models are trained for 40 iterations performing stochastic gradient descent. We initialize the recurrent weight matrix to be identity and biases to be zero. We use Capped Rectified Linear units (CappedReLu) and ranking loss with default parameters (section 2.5). The entity vectors C_{RE} and E_{ER} are initialized with zero when NER is performed for the first time in entity and relation extraction loop (Figure 6). The models are implemented in Theano (Bergstra et al., 2010; Bastien et al., 2012).

5 Related Work

Recurrent and convolutional neural networks (Zeng et al., 2014; Nguyen and Grishman, 2015; Zhang and Wang, 2015; Vu et al., 2016a) have delivered competitive performance for sentence-level relation classification. Socher et al. (2012) and Zhang and Wang (2015) proposed recurrent/recursive type neural networks to construct sentence representations based on dependency parse trees. However, these sentence-level state-of-the-art methods do not model the interdependencies of entity and relation, do not handle multiple relation instances in a sentence and therefore, can not detect entity mention pairs for the sentence-level relations. Our approach is a joint entity and word-level relation extraction capable to model multiple relation instances, without knowing nominal pairs.

Existing systems (Roth and Yih, 2004; Kate and Mooney, 2010; Miwa and Sasaki, 2014) are complex feature-based models for joint entity and relation extraction. The most related work to our method is (Miwa and Sasaki, 2014); however they employ complex search heuristics (Goldberg and Elhadad, 2010; Stoyanov and Eisner, 2012) to fill the entity-relation table based on structured prediction method. They explicitly model the label dependencies and their joint approach is not based on neural networks. Multi-task learning (Caruana, 1998) via neural networks (Zhang and Yeung, 2012; Seltzer and Droppo, 2013; Dong et al., 2015; Li and J, 2014; Collobert and Weston, 2008) have been used to model relationships among the correlated tasks. Therefore, we present a unified neural network based multi-task framework to model the entity-relation table for end-to-end relation extraction.

6 Conclusion

We proposed TF-MTRNN, a novel architecture that jointly models entity and relation extraction, and showed how an entity-relation table is mapped to a neural network framework that learns label interdependencies. We introduced word-level relation classification through composition of words; this is advantageous in modeling multiple relation instances without knowing the corresponding entity mentions in a sentence. We also introduced context-awareness in RNN network to incorporate missing information, and investigated piggybacking approach to model entity-relation label interdependencies.

Experimental results show that TF-MTRNN outperforms state-of-the-art method for both entity and relation extraction tasks.

Acknowledgements
This research was supported by Siemens AG CT MIC - Machine Intelligence, Munich Germany.

References

Frèdèric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. *Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop*.

James Bergstra, Olivier Breuleux, Frèdèric Bastien, Pascal Lamblin, Guillaume Desjardins Razvan Pascanu, Joseph Turian, David Warde-Farley, , and Yoshua Bengio. 2010. Theano: a cpu and gpu math expression compiler. *In Proceedings of the Python for Scientific Computing Conference (SciPy).*

Rich Caruana. 1998. Multitask learning. *Springer*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing:deep neural networks with multitask learning. *In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland*.

L Van der Maaten and G Hinton. 2008. Visualizing data using t-sne. *Proceedinggs of JMLR*.

Daxiang Dong, Wei He Hua Wu, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1723–1732.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. *In Proceedings of the Association for Computational Linguistics*.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2).

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

M. Jordan. 1986. Serial order: A parallel distributed processing approach. *Published in Tech. Rep. No. 8604. San Diego: University of California, Institute for Cognitive Science*.

Rohit J. Kate and Raymond Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. *In Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 203–212.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition.

Qi Li and Heng J. 2014. Incremental joint extraction of entity mentions and relations. *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 402–412.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. *In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1858–1869.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. *In Proceedings of the NAACL Workshop on Vector Space Modeling for NLP*.

Dan Roth and Wen-Tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. *In Hwee Tou Ng and Ellen Riloff, editors, HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 1–8.

Dan Roth and Wen-Tau Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. *In Hwee Tou Ng and Ellen Riloff, editors, HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*.

Michael L Seltzer and Jasha Droppo. 2013. Multi-task learning in deep neural networks for improved phoneme recognition. *In Proceedings of the IEEE International Confonference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. In proceedings of emnlp/conll. *Association for Computational Linguistics*.

Veselin Stoyanov and Jason Eisner. 2012. Easy-first coreference resolution. *In Proceedings of COLING 2012*, page 25192534.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016a. Combining recurrent and convolution neural networks for relation classification. *In Proceedings of the NAACL.*

Ngoc Thang Vu, Pankaj Gupta, Heike Adel, and Hinrich Schütze. 2016b. Bi-directional recurrent neural network with ranking loss for spoken language understanding. *IEEE/ACM Trans. on Audio, Speech, and Language Processing*.

Paul J Werbos. 1990. Backpropagation through time:what it does and how to do it. *In Proceedings of the IEEE.*

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. *In Proceedings of COLING*.

Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *In ArXiv*.

Y. Zhang and D.-Y. Yeung. 2012. A convex formulation for learning task relationships in multi-task learning. *arXiv preprint arXiv:1203.3536*.

Appendix A. R-biRNN discussed in section 2.5.

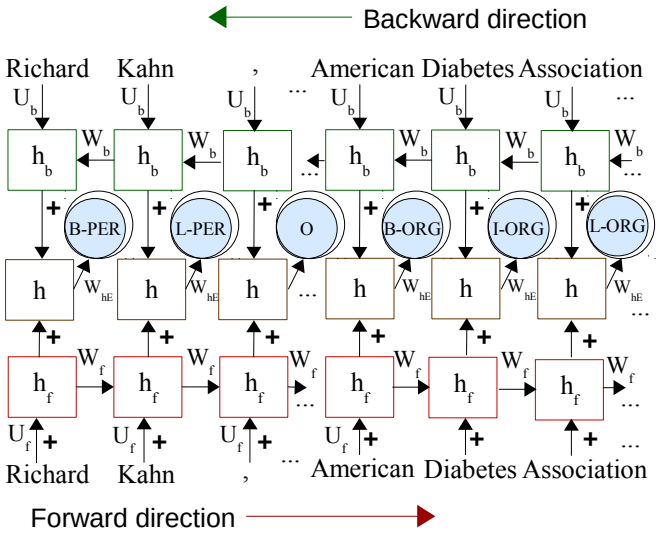


Figure 12: R-biRNN. Disintegrating TF-MTRNN (Figure 3) to illustrate that it is comprised of R-biRNN for entity learning. (...) indicates remaining words in the sentence (Figure 1).

Bilingual Autoencoders with Global Descriptors for Modeling Parallel Sentences

Biao Zhang[1], **Deyi Xiong**[2*], **Jinsong Su**[1], **Hong Duan**[1] **and Min Zhang**[2]
Xiamen University, Xiamen, China 361005[1]
Soochow University, Suzhou, China 215006[2]
zb@stu.xmu.edu.cn, {jssu,hduan}@xmu.edu.cn
{dyxiong, minzhang}@suda.edu.cn

Abstract

Parallel sentence representations are important for bilingual and cross-lingual tasks in natural language processing. In this paper, we explore a bilingual autoencoder approach to model parallel sentences. We extract sentence-level global descriptors (e.g. *min*, *max*) from word embeddings, and construct two monolingual autoencoders over these descriptors on the source and target language. In order to tightly connect the two autoencoders with bilingual correspondences, we force them to share the same decoding parameters and minimize a corpus-level semantic distance between the two languages. Being optimized towards a joint objective function of reconstruction and semantic errors, our bilingual antoencoder is able to learn continuous-valued latent representations for parallel sentences. Experiments on both intrinsic and extrinsic evaluations on statistical machine translation tasks show that our autoencoder achieves substantial improvements over the baselines.

1 Introduction

Neural sentence modeling that learns continuous-valued vector representations for sentences in a low-dimensional latent semantic space, has recently attracted considerable interests in the field of nature language processing (NLP). A variety of models have been proposed (Collobert and Weston, 2008; Socher et al., 2011; Mikolov et al., 2011; Hermann and Blunsom, 2013; Kalchbrenner and Blunsom, 2013; Kalchbrenner et al., 2014; Kim, 2014; Ma et al., 2015; Tai et al., 2015; Zhang et al., 2015a). Most of these models, however, focus on monolingual cases where sentences from a single language are modeled. They do not explore semantic correspondences among parallel sentences. On account of this, monolingual neural sentence models are not naturally fit for bilingual or cross-lingual NLP tasks, such as machine translation, cross-lingual classification and information retrieval.

In order to induce sentence representations in a bilingual rather than monolingual semantic space, researchers have proposed a few approaches, following efforts of bilingual word embeddings (Klementiev et al., 2012; Zou et al., 2013). These studies either explore *recursive/recurrent* neural networks (Zhang et al., 2014; Su et al., 2015) or *bag-of-words* based neural networks (Yih et al., 2011; Chandar et al., 2014; Lauly et al., 2014; Hermann and Blunsom, 2014; Zhou et al., 2015) to learn bilingual sentence representations. The former recursively compose sentences from the bottom up, taking into account bilingual constraints from word alignments. Due to the complexity, they are not easy to be scalable. Additionally, they also suffer from errors and noises of word alignments. In contrast, the latter are relatively simple and scalable. However, they often heavily rely on only one descriptor of the bag-of-words embeddings (e.g. the *avg* representation) and hence are weak in capturing fine-grained complex linguistic phenomena. Therefore we believe that modeling parallel sentences still remains a serious challenge.

Inspired by works on multimodal autoencoders (Ngiam et al., 2011; Wang et al., 2014; Feng et al., 2014; Feng et al., 2015), we explore a novel bilingual autoencoder to model parallel sentences, which is able to incorporate global semantic information into sentence representations. The overall architecture

*Corresponding author.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2548–2558, Osaka, Japan, December 11-17 2016.

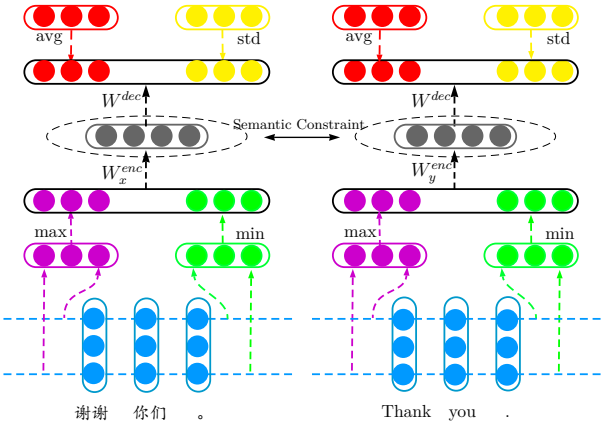


Figure 1: The architecture of our bilingual autoencoder visualized with a parallel sentence pair. Word embeddings are represented in blue color, while the *avg*, *std*, *min* and *max* descriptors are indicated in red, yellow, green and purple colors, respectively. Specifically, we use gray color to denote the hidden layer. The subscripts x and y indicate the source and target language respectively, and the superscripts enc and dec indicate the encoding and decoding process. Each dash line with its corresponding color depicts the information related with that specific descriptor, and the black ellipse around the hidden layer means that the bilingual semantic constraint is built at the corpus rather than sentence level.

is illustrated in Figure 1.[1] It is fast and scalable as we do not use complex recursiveness or recurrence. Comparing with conventional bag-of-words based autoencoders, our model explores several complementary sentence-level statistical descriptors, i.e., *min* (minimum), *max* (maximum), and *avg* (average), *std* (standard deviation) computed over all word vectors, to capture high-level global features. These descriptors alleviate the weakness of conventional bag-of-words representations in feature extraction and their insensitiveness to semantic constraints between sentences. The proposed autoencoder further encodes global descriptors (e.g., *min*, *max*) into hidden representations, and then decodes them into the other descriptors (e.g., *avg*, *std*). In order to capture bilingual correspondences, we force our autoencoder to share the decoding parameters across two languages (i.e., the same W^{dec} in Figure 1), and further optimize the parameters with respect to a corpus-level semantic constraints between the source and target language. This architecture tightly connects two monolingual autoencoders and bridges the gaps between the source and target semantic spaces.

To evaluate the effectiveness of our proposed bilingual autoencoder, we conduct both intrinsic and extrinsic evaluations on statistical machine translation (SMT) tasks. The intrinsic evaluation measures the capability of our model in semantic similarity calculation, while the extrinsic evaluation examines whether our model can be used to improve machine translation quality. Results on the NIST 2006 and 2008 datasets show that our autoencoder can significantly outperform the baseline methods. The main contributions of our work lie in the following three aspects:

- The proposed autoencoder learns bilingual representations for parallel sentences using global descriptors. To the best of our knowledge, this architecture has never been investigated before.
- We incorporate the corpus- rather than sentence-level semantic constraint into sentence embeddings, and share the decoding parameters to further bridge the semantic spaces of two different languages.
- Our model does not rely on word alignments. It is scalable, simple yet effective.

2 Related Work

Previous studies that are related to our work can be roughly divided into three groups: neural sentence modeling, multimodal autoencoders and bilingual autoencoders. We will briefly describe them in this

[1]For illustration, we set the dimensionality of word embedding to 3. Besides, we also omit the dash lines from word embeddings to *avg* and *std* descriptors. Without loss of generality, throughout this paper we assume the input and target to be *min*, *max* and *avg*, *std* respectively.

section and highlight the differences of our work from them.

neural sentence modeling (NSM): NSM is able to provide distributed sentence representations for many NLP tasks. One natural approach to NSM is to produce sentence representations from word embeddings with recursive or recurrent neural networks via composition (Socher et al., 2011; Mikolov et al., 2011; Hermann and Blunsom, 2013). To preserve sequence information over time, Tai et al. (2015) further incorporate LSTMs into tree-structured recursive networks. Unlike these composition-based methods, convolutional neural models utilize convolution layers to explicitly capture task-specific n-grams (Collobert and Weston, 2008; Kalchbrenner et al., 2014; Kim, 2014; Ma et al., 2015; Zhang et al., 2015a). Kalchbrenner and Blunsom (2013) combine convolutional and recurrent neural networks to model discourse representations. Different from these monolingual studies, our model aims at generating bilingual sentence representations. Furthermore we use autoencoders instead of recursive or recurrent neural networks. The descriptors in our model can be considered as a shallow convolution layer over an entire sentence (Zhang et al., 2015a).

multimodal autoencoders: Sentence modeling is performed only on text modality. However, information from different modalities might be complementary to each other. Motivated by this, Ngiam et al. (2011) develop deep multimodal autoencoders based on restricted Boltzmann machines (RBM) to jointly learn features over audio and video modalities. Instead of RBM, Wang et al. (2014) exploit stacked autoencoders for multimodal retrieval, and Feng et al. (2014) propose correspondence autoencoders for cross-modal retrieval. Feng et al. (2015) further investigate the utilization of deep correspondence RBM for cross-modal retrieval.

bilingual autoencoders: As a special case of multimodal autoencoders where each language is viewed as a modality, bilingual autoencoders have drawn attention in recent years. Based on recursive autoencoders, Zhang et al. (2014) incorporate phrase-level bilingual constraints into phrase representation learning. Along this line, Su et al. (2015) further exploit bilingual subphrase correspondences via word alignments for learning bilingual phrase structures and representations. Dissimilarly, Zhou et al. (2015) employ denoising autoencoders to learn bilingual embeddings. Lauly et al. (2014) directly perform encoding and decoding between source and target sentences based on bag-of-words representations. Chandar et al. (2014) exploit joint reconstruction and cross-lingual correlations on bilingual autoencoders. Unlike these models, our model builds autoencoders on global descriptors extracted from word embeddings, and incorporates bilingual semantics via corpus-level bilingual constraints into the hidden layer.

Our work is similar to the work of Zhang et al. (2015b) in that we both aim at modeling parallel sentences under the semantic constraints. The difference lies on the following two aspects: 1) they employ a chunk-based convolutional neural network to model sentences, while we use a much simpler autoencoder network; and 2) they impose the bilingual constraint strictly on the sentence level. Instead, we explore a much more relax constraints on the corpus level. As Chandar et al. (2014)'s bilingual autoencoder is most closely related to ours, we use their autoencoder as our baseline and describe it with more details in the next section.

3 Bilingual Bag-of-Words Autoencoder

The Bilingual Bag-of-Words Autoencoder (BBoWAE) (Chandar et al., 2014) builds two separate feed-forward autoencoders based on bag-of-words representations. The two autoencoders are jointly reconstructed with cross-lingual correlations. Figure 2 shows the overall architecture. Given a parallel sentence pair $(\mathbf{x}, \mathbf{y})$, BBoWAE model first generates corresponding bag-of-words based sentence representations, and then encodes them into the hidden layer:

$$a(\mathbf{x}) = c + W^{\mathcal{X}} v(\mathbf{x}),\ \phi(\mathbf{x}) = h(a(\mathbf{x})); \quad a(\mathbf{y}) = c + W^{\mathcal{Y}} v(\mathbf{y}),\ \phi(\mathbf{y}) = h(a(\mathbf{y})) \tag{1}$$

where $v(\cdot)$ converts sentence into a fixed-size but sparse binary vector, $W^{\mathcal{X}}$ and $W^{\mathcal{Y}}$ implicitly represent language specific word embeddings, c is the bias term shared by both autoencoders and $h(\cdot)$ is an element-wise non-linear function.

Upon these hidden layers, BBowAE model further performs a reconstruction of the original sentence

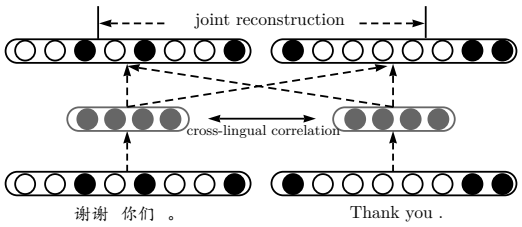


Figure 2: The architecture of BBoWAE model. We use black and white colors to indicate 1 and 0 respectively, which form the bag-of-words representation of a sentence. Notice that the gray nodes are real-valued.

in both languages:

$$\hat{v}(\mathbf{x})_{\mathbf{x}} = h(W^{\mathcal{X}^T}\phi(\mathbf{x}) + b^{\mathcal{X}}), \hat{v}(\mathbf{y})_{\mathbf{x}} = h(W^{\mathcal{Y}^T}\phi(\mathbf{x}) + b^{\mathcal{Y}}) \tag{2}$$

$$\hat{v}(\mathbf{x})_{\mathbf{y}} = h(W^{\mathcal{X}^T}\phi(\mathbf{y}) + b^{\mathcal{X}}), \hat{v}(\mathbf{y})_{\mathbf{y}} = h(W^{\mathcal{Y}^T}\phi(\mathbf{y}) + b^{\mathcal{Y}}) \tag{3}$$

where $b^{\mathcal{X}}$ and $b^{\mathcal{Y}}$ are bias terms, $h(\cdot)$ here is the sigmoid non-linearity and $\hat{v}(\mathbf{x})_{\mathbf{y}}$ denotes the reconstructed representation of $v(\mathbf{x})$ from $v(\mathbf{y})$. The other three reconstructed vectors are similarly defined.

To favour more meaningful bilingual representations, BBoWAE incorporates a cross-lingual correlation error and different reconstruction errors into the objective function:

$$\ell(\mathbf{x},\mathbf{y}) + \ell(\mathbf{y},\mathbf{x}) + \ell(\mathbf{x},\mathbf{x}) + \ell(\mathbf{y},\mathbf{y}) + \beta\ell([\mathbf{x},\mathbf{y}],[\mathbf{x},\mathbf{y}]) - \lambda cor(a(\mathbf{x}), a(\mathbf{y})) \tag{4}$$

where β, λ are hyperparameters, and $[\mathbf{x},\mathbf{y}]$ represents the concatenation of the two sentences. Each loss function $\ell(\cdot)$ is the cross-entropy error between the original bag-of-words representation $v(\cdot)$ and reconstructed representation $\hat{v}(\cdot)$. (e.g. $v(\mathbf{y})$ vs. $\hat{v}(\mathbf{y})_{\mathbf{x}}$).

The term $\ell([\mathbf{x},\mathbf{y}],[\mathbf{x},\mathbf{y}])$ is the joint reconstruction term, where the two sentences are simultaneously presented as input and reconstructed; and the term $cor(a(\mathbf{x}), a(\mathbf{y}))$ is the sum of scalar correlations between $a(\mathbf{x})$ and $a(\mathbf{y})$ across all dimensions. To obtain a stochastic estimate of the correlation, small mini-batches are used during training.

Notice that the sentence representation $a(\mathbf{x})$/$a(\mathbf{y})$ is actually the *sum* of bag-of-words embeddings. Although BBoWAE is effective in learning bilingual word embeddings, it is weak in modeling parallel sentences due to the well known insufficiency of the sum representations. We instead resort to multiple global descriptors over word embeddings and further explore more suitable bilingual constraints.

4 Bilingual Autoencoder with Global Descriptors

We describe our model in two phases: autoencoder with global descriptors and bilingual semantic constraints. The former constitutes the basic structure for modeling monolingual sentences, while the latter explores bilingual constraints to connect the two autoencoders. After the description of the architecture, we present details for parameter inference.

4.1 Autoencoder with Global Descriptors

Our autoencoder is built upon distributed word embeddings, where each word in vocabulary V corresponds to a d-dimensional dense, real-valued vector, and all word vectors are stacked into a word embedding matrix $L \in \mathbb{R}^{d\times|V|}$, where $|V|$ is the vocabulary size.

Given an ordered list of n words in a sentence, we retrieve the i-th word representation from L with its corresponding vocabulary index I_i: $e(x_i) = L_{:,I_i} \in \mathbb{R}^d$. All word vectors in the sentence $\mathbf{x}$ produce the following output matrix: $M = (e(x_1), e(x_2), \ldots, e(x_n)) \in \mathbb{R}^{d\times n}$. From this matrix M, we investigate four sentence-level statistical descriptors: *min*, *max*, *avg* and *std* in each row r as follows:

$$g_r^{min} = \min(M_{r,1:n}), g_r^{max} = \max(M_{r,1:n}), g_r^{avg} = \frac{1}{n}\sum_{i=1}^{n} M_{r,i}, g_r^{std} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(M_{r,i} - g_r^{avg})^2} \tag{5}$$

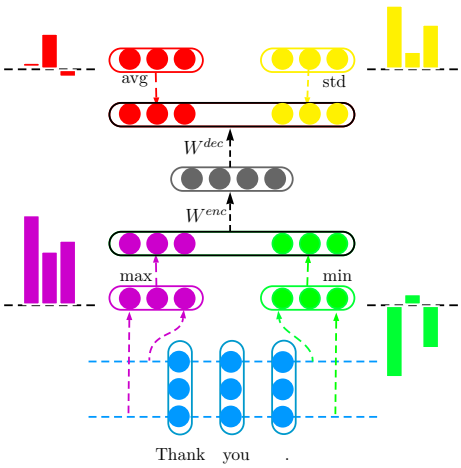


Figure 3: Monolingual autoencoder with global descriptors. The colored histogram on each corner indicates the corresponding descriptor for a toy example. Notice that the black dash line in each histogram represents the value 0.

Figure 3 illustrates the distribution of each descriptor for a toy example[2], from which we can find that these descriptors represent different aspects of the same sentence and are complementary to each other.

After obtaining these global descriptors, we select some of them as our encoding input (e.g. *min*, *max*), and leave the others as our decoding target (e.g. *std*, *avg*). Note that the neural network built on these descriptors is an autoencoder since the input and output layer in the network represent the same sentence.[3] Additionally, this autoencoder can naturally handle variable-length sentences (Notice that $g \in \mathbb{R}^d$). In particular, we concatenate the input descriptors into one vector, and encode it into the hidden layer shown as gray nodes in Figure 3:

$$hid = f(W^{enc}[g^{max}; g^{min}] + b^{enc}) \tag{6}$$

where $W^{enc} \in \mathbb{R}^{m \times 2d}$ and $b^{enc} \in \mathbb{R}^m$ are the encoding parameters. $hid \in \mathbb{R}^m$ is the sentence representation, and $f(\cdot)$ is an element-wise activation function such as $tanh(\cdot)$, which is used for all activation functions in our model. To prevent the hidden layer from being very small, we normalize all output vectors of the hidden layer to have a unit length, $hid = \frac{hid}{\|hid\|}$.

Upon the hidden layer, we stack a decoding layer to *reconstruct* the other descriptors:

$$out = f(W^{dec} hid + b^{dec}) \tag{7}$$

where $W^{dec} \in \mathbb{R}^{m \times 2d}$ and $b^{dec} \in \mathbb{R}^{2d}$ are the decoding parameters.

To integrate the information contained in different descriptors into sentence embedding learning, we train our autoencoder to minimize the following Euclidean distance error with respect to the target descriptors $[g^{avg}; g^{std}]$:

$$E_{ae}(\mathbf{x}) = \frac{1}{2}\|[g^{avg}; g^{std}] - out\|^2 \tag{8}$$

4.2 Bilingual Semantic Constraints

In order to incorporate bilingual correspondences between the source and target sentences into the above-mentioned autoencoder, we employ two kinds of bilingual constraints.

The first bilingual constraint is to share decoding parameters across the two autoencoders, that is, using the same W^{dec} and b^{dec} for them. This ensures that the relations between sentence representations (in the hidden layer) and target descriptors (in the output layer) are consistent across the source and target language. In this way, bilingual information is propagated from one language to the other language

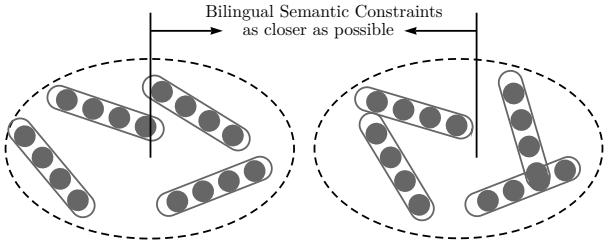


Figure 4: Corpus-level bilingual semantic constraints. The left and right dashed ellipses indicate bilingual semantic spaces.

through these parameters.

The second bilingual constraint is that the center of bilingual spaces should be as close as possible. Therefore we minimize a corpus-level semantic distance defined as follows:

$$E_{sem}(\mathcal{C}) = \sum_{k=1}^{\lceil N/B \rceil} \frac{1}{2} \| \frac{1}{B} \sum_{i=1}^{B} \mathbf{x}_{k,i} - \frac{1}{B} \sum_{i=1}^{B} \mathbf{y}_{k,i} \|^2 \tag{9}$$

where $\mathcal{C}$ represents the entire parallel corpus, which has N parallel sentences with a batch size of B.

Figure 4 shows this constraint, where we try to shorten the distance between the centers of the source and target language spaces. Different from previous works (Zhang et al., 2014; Chandar et al., 2014; Su et al., 2015), we define this distance at the corpus rather than sentence level. The reasons for this are twofold: 1) The potential equivalent translations for a source sentence could be many, and 2) the number of non-equivalent translations for one sentence must be enormous. Therefore, defining the semantic distance at the sentence level would be too strict, and may bring in noises to our model. In contrast, our corpus-level constraint is much more relax and general. Notice that if we set $B = 1$, this constraint goes back to sentence level; If we set $B \to \infty$, it becomes a constraint upon the whole training corpus. Intuitively, B controls the strictness of the semantic constraints.

4.3 Parameter Inference

There are two kinds of errors involved in the overall objective function: the *autoencoder error* in Eq. (8) and the *semantic error* in Eq. (9). Given the training corpus $\mathcal{C}$, the joint training objective is:

$$J(\mathcal{C}) = \frac{\alpha}{N} \sum_{i=1}^{N} (E_{ae}(\mathbf{x}_i) + E_{ae}(\mathbf{y}_i)) + (1-\alpha) E_{sem}(\mathcal{C}) + R(\theta) \tag{10}$$

α is a balance factor, $\theta = \{L_{\mathbf{x}}, L_{\mathbf{y}}, W^{enc}_{\mathbf{x}}, W^{enc}_{\mathbf{y}}, W^{dec}\}$, and $R(\theta)$ is the regularization term: $R(\theta) = \frac{\lambda_L}{2}\|\theta_L\|^2 + \frac{\lambda_W}{2}\|\theta_W\|^2$. For regularization, we divide θ into two sets: θ_L for $(L_{\mathbf{x}}, L_{\mathbf{y}})$ and θ_W for $(W^{enc}_{\mathbf{x}}, W^{enc}_{\mathbf{y}}, W^{dec})$. Accordingly, λ_L and λ_W are the corresponding coefficients.

We employ the toolkit Word2Vec (Mikolov et al., 2013) to pretrain word embeddings for each language on the corresponding part of $\mathcal{C}$, and randomly initialize the other parameters using normal distribution ($\mu = 0, \sigma = 0.01$). To optimize these parameters, we apply the L-BFGS algorithm to the gradient of Eq. (10).

5 Sentence Representation and Semantic Similarity

Given a parallel sentence pair $(\mathbf{x}, \mathbf{y})$, our model will generate sentence representations $(hid_{\mathbf{x}}, hid_{\mathbf{y}})$ with the trained parameters θ^*. Different from Eq. (6) that only uses the input descriptors to calculate the hidden layer, we use both the input and target descriptors to compute the final hidden representation

[2]We sampled the word embeddings uniformly from -1 to 1. Finally, we obtained *Thank*:$(0.95, 0.64, -0.23)$, *you*:$(-0.82, 0.54, 0.72)$ and .:$(-0.12, 0.10, -0.52)$ respectively for illustration.

[3]Therefore, the *autoencoder* in this paper is an extension of the conventional autoencoder.

of $\mathbf{x}$ or $\mathbf{y}$ as these descriptors provide different and complementary views of a sentence. The final hidden representations of the sentence $\mathbf{x}$ and $\mathbf{y}$ are defined as follows:

$$hid_{\mathbf{x}} = f(W_{\mathbf{x}}^{enc,*}[g_{\mathbf{x}}^{min}; g_{\mathbf{x}}^{max}] + W^{dec,*^T}[g_{\mathbf{x}}^{avg}; g_{\mathbf{x}}^{std}] + b^{enc}) \tag{11}$$

$$hid_{\mathbf{y}} = f(W_{\mathbf{y}}^{enc,*}[g_{\mathbf{y}}^{min}; g_{\mathbf{y}}^{max}] + W^{dec,*^T}[g_{\mathbf{y}}^{avg}; g_{\mathbf{y}}^{std}] + b^{enc}) \tag{12}$$

The corresponding semantic similarity is measured by the Euclidean distance between the two hidden representations:

$$Sim(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\|hid_{\mathbf{x}} - hid_{\mathbf{y}}\|^2 \tag{13}$$

The smaller $Sim(\cdot, \cdot)$ is, the more semantically similar the parallel sentence pair is. All the following intrinsic and extrinsic evaluations are based on this similarity measurement.

6 Experiments

In this section, we carried out a series of experiments to validate the effectiveness of our proposed bilingual autoencoders on NIST Chinese-English translation tasks using large-scale bilingual training data. In particular, we investigated 1) whether our model is able to distinguish parallel sentence pairs from nonparallel sentences, and 2) whether our model can improve machine translation quality.

6.1 Setup

Our translation decoder is a state-of-the-art hierarchical phrased-based SMT system (Chiang, 2007). The bilingual training data is the combination of LDC2003E14, LDC2004T07, LDC2005T06, LDC2005T10 and LDC2004T08 (Hong Kong Hansards/Laws/News), which contains 2.9M sentence pairs with 80.9M Chinese words and 86.4M English words. We used a 4-gram language model which was trained on the Xinhua section of the English Gigaword corpus (306M words) using the SRILM[4] toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing.

In addition to the baseline decoder, we also compared our bilingual autoencoder against the abovementioned BBoWAE model (Chandar et al., 2014). To train this model, we used the same bilingual corpus and their open source code[5] with the same hyperparameters as they used. We employed the objective in Eq. (4) except for the correlation term as an entropy-based semantic similarity measure.[6]

Since our model is general in terms of input and target descriptors, we investigated two variants of the proposed bilingual autoencoder: 1) *min* and *max* as the input descriptors, while *avg* and *std* as the target descriptors (MM2AS); and 2) the opposite direction (AS2MM). Throughout all experiments, we set $B = 100$, $\alpha = 0.100$, $\lambda_L = 10^{-6}$, $\lambda_W = 10^{-3}$ and $m = 2d$. With respect to the dimensionality of word embeddings, we tried four different dimensions from 25 to 100 with an increment of 25 each time.

We used the NIST evaluation set of MT05 as our development set, and sets of MT06/MT08 as the test sets. Case-insensitive NIST BLEU (Papineni et al., 2002) was used to measure translation performance. We used minimum error rate training (MERT) (Och, 2003) to optimize the feature weights. In order to alleviate the instability of MERT, we followed Clark et al. (2011) to run MERT three times and report average BLEU scores over the three runs for all our MT experiments.

6.2 Intrinsic Evaluation: Semantic Analysis

The first group of experiments aims at analyzing the ability of our model in distinguishing parallel sequences from nonparallel sequences, at both the phrase and sentence level. For convenience, we used MM2AS model and set $d = 25$ in all the following experiments.

[4]http://www.speech.sri.com/projects/srilm/download.html

[5]http://www.sarathchandar.in/crl.html

[6]We make this choice due to the following two reasons: 1) the correlation term is meaningless for a single sentence pair; 2) the combined reconstruction errors in Eq. (4) provide a balanced and comprehensive similarity measure as they calculate cross entropies of $\mathbf{x}$ to $\mathbf{x}$, $\mathbf{y}$ to $\mathbf{y}$, $\mathbf{x}$ to $\mathbf{y}$, $\mathbf{y}$ to $\mathbf{x}$ and $[\mathbf{x}, \mathbf{y}]$ to $[\mathbf{x}, \mathbf{y}]$.

Source Phrase	Target Candidates
充满 激情	imbued with passions full of passion full of excitement
宾至如归 的 感觉	feeling of being at home feel at home feel welcomed
促进 可 持续 发展	promotion of sustainable development promote sustainable development promote sustained growth

Table 1: Source phrases in our phrase table with their top-3 target translations selected by our model.

System	MT06	MT08	Avg
BBoWAE	77.73	74.57	76.15
Our Model	87.18	85.85	**86.52**

Table 2: Accuracy on recognizing parallel sentence pairs.

6.2.1 Analysis on Phrasal Semantic Similarities

To have a deep look into what our model measures similarities of bilingual phrases, we show some examples from our scored phrase table in Table 1. For each variable-length source phrase, we extract the top-3 target candidates according to their semantic similarity scores calculated in Eq. (13).

Take the source phrase " 宾至如归 的 感觉" as an example, our model succeeds in distinguishing the most semantically equivalent translation "*feeling of being at home*" from the less equivalent translation "*feel welcomed*". Although the candidate "*feel at home*" has almost the same meaning, this candidate is a verb phrase which is not consistent with the noun phrase at the source side from the perspective of syntax. This indicates that the proposed bilingual model is able to capture some semantic and syntactic properties of bilingual sequences.

6.2.2 Evaluation on Sentential Semantic Similarities

To testify the ability of our model at the sentence level, we further collected source sentences and their reference translations from the test sets MT06/MT08 as our parallel sentence pairs. We randomly sampled l target sentences from the target vocabulary based on reference translations and combined sampled target sentences and their corresponding source sentences as nonparallel sentence pairs.[7]

For both our model and BBoWAE model, we test whether the model can assign a higher similarity score to a parallel sentence pair than a sampled nonparallel sentence pair. We employ accuracy as our evaluation metric, and conducted 5 evaluations, for each of which we sampled 3 different target sentences for each source sentence. The final average results are shown in Table 2. We find that our model significantly outperforms BBoWAE model by an absolute improvement of 10%. This demonstrates that our model can learn better semantic representations for parallel sentences than BBoWAE does.

6.3 Extrinsic Evaluation: Machine Translation

The second group of experiments were carried out to study the effectiveness of our model in calculating semantic similarities for SMT task: *decoding with phrasal semantic similarities*. This needs to measure the semantic similarity between a source sequence and its translation candidate according to Eq. (13).

In addition to the conventional four translation probabilities (phrase translation probabilities and lexical weights in both directions), we incorporate the phrasal semantic similarities as an additional feature into our baseline SMT system.

Table 3 summarizes the results, from which we observe that:

1) Our model can significantly improve translation quality on all testsets for any dimensions. Particularly, AS2MM model obtains the best result 28.08 when $d = 100$, which outperforms Baseline and BBoWAE model by up to 1.46 and 1.1 BLEU points respectively.

[7]The length of a sampled sentence is uniformly sampled from 1 to the length of the corresponding reference sentence.

System	d	MT06	MT08	Avg
Baseline	-	30.04	23.19	26.62
BBoWAE	40	30.52	23.43	26.98
MM2AS	25	31.11↑	23.90↑	27.51
	50	31.42⇑	24.19⇑	27.81
	75	31.18↑	23.98↑	27.58
	100	31.11↑	23.91↑	27.51
AS2MM	25	31.29⇑	24.34⇑	27.82
	50	31.10↑	24.06⇑	27.58
	75	31.17↑	23.92↑	27.55
	100	31.64⇑	24.51⇑	28.08

Table 3: Experiment results for different dimensions with phrasal semantic similarities on the test sets. **Avg** = average BLEU scores on the test sets. ⇑/↑:significantly better than BBoWAE ($p < 0.01/0.05$, respectively).

2) The BBoWAE model, designed for learning bilingual word embeddings, is not good at modeling high-level and long sequence representations. The translation result of this model is 26.98, only 0.36 points higher than the Baseline. Additionally, as one major difference between BBoWAE and our model is the corpus-level semantic constraints, this result further demonstrates the superiority of this constraint.

3) It's interesting that the overall result of AS2MM model is slightly better than that of MM2AS model (27.76 vs. 27.60 on average). The reason may be that when back-propagating autoencoder errors onto the hidden and input layer, AS2MM model is easier and more straightforward than MM2AS.

7 Conclusion and Future Work

In this paper, we have presented a simple yet effective and scalable bilingual autoencoder for parallel sentence modeling. We incorporate global descriptors and corpus-level semantic constraints into bilingual sentence representations. Experiment results show that our approach achieves substantial improvements against the baseline models.

For the future work, we would like to explore more variants of our bilingual autoencoder, e.g. taking the *avg*, *max* as inputs and the *std*, *min* as targets. Besides, we will further enhance our autoencoder with semantic and deeper descriptors and verify our model on other bilingual or cross-lingual tasks, such as cross-lingual sentiment classification.

Acknowledgements

The authors were supported by National Natural Science Foundation of China (Grant Nos. 61303082, 61403269, 61432013, 61525205 and 61672440), Natural Science Foundation of Fujian Province (Grant No. 2016J05161) and Natural Science Foundation of Jiangsu Province (Grant No. BK20140355). We also thank the anonymous reviewers for their insightful comments.

References

Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Proc. of NIPS*, pages 1853–1861.

David Chiang. 2007. Hierarchical phrase-based translation. *Comput. Linguist.*, pages 201–228, June.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proc. of HLT*, pages 176–181.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of ICML*, pages 160–167.

Fangxiang Feng, Xiaojie Wang, and Ruifan Li. 2014. Cross-modal retrieval with correspondence autoencoder. In *Proc. of ACMMM*, pages 7–16.

Fangxiang Feng, Ruifan Li, and Xiaojie Wang. 2015. Deep correspondence restricted boltzmann machine for cross-modal retrieval. *Neurocomputing*, pages 50 – 60.

Karl Moritz Hermann and Phil Blunsom. 2013. The Role of Syntax in Vector Space Models of Compositional Semantics. In *Proc. of ACL*, August.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proc. of ACL*, pages 58–68, June.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. *Proc. of CVSC*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *Proc. of ACL*, June.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. of EMNLP*, pages 1746–1751, October.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proc. of COLING*, pages 1459–1474, December.

Stanislas Lauly, Alex Boulanger, and Hugo Larochelle. 2014. Learning multilingual word representations using a bag-of-words autoencoder. *NIPS Workshop*.

Mingbo Ma, Liang Huang, Bowen Zhou, and Bing Xiang. 2015. Dependency-based convolutional neural networks for sentence embedding. In *Proc. of ACL-IJCNLP*, pages 174–179, July.

Tomas Mikolov, Stefan Kombrink, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Proc. of ICASSP*, pages 5528–5531.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*, pages 3111–3119.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. 2011. Multimodal deep learning. In *Proc. of ICML*, pages 689–696.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proc. of ACL*, pages 311–318.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proc. of EMNLP*, pages 151–161.

Andreas Stolcke. 2002. Srilm – an extensible language modeling toolkit. In *Proc. of ICSLP*, volume 2, pages 901–904.

Jinsong Su, Deyi Xiong Xiong, Biao Zhang, Yang Liu, Junfeng Yao, and Min Zhang. 2015. Bilingual correspondence recursive autoencoder for statistical machine translation. In *Proc. of EMNLP*, September.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. of ACL-IJCNLP*, pages 1556–1566, July.

Wei Wang, Beng Chin Ooi, Xiaoyan Yang, Dongxiang Zhang, and Yueting Zhuang. 2014. Effective multi-modal retrieval based on stacked auto-encoders. *Proc. of VLDB Endow.*, pages 649–660, April.

Wen-tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Proc. of CoNLL*, pages 247–256, June.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014. Bilingually-constrained phrase embeddings for machine translation. In *Proc. of ACL*, pages 111–121, June.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015a. Shallow convolutional neural network for implicit discourse relation recognition. In *Proc. of EMNLP*, September.

Jiajun Zhang, Dakun Zhang, and Jie Hao. 2015b. Local translation prediction with global sentence representation. In *Proc. of IJCAI*, IJCAI'15, pages 1398–1404.

HuiWei Zhou, Long Chen, Fulin Shi, and Degen Huang. 2015. Learning bilingual sentiment word embeddings for cross-language sentiment classification. In *Proc. of ACL-IJCNLP*, pages 430–440, July.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Pro. of EMNLP*, pages 1393–1398, October.

Multi-Engine and Multi-Alignment Based Automatic Post-Editing and its Impact on Translation Productivity

Santanu Pal[1], Sudip Kumar Naskar[2], Josef van Genabith[1,3]
[1]Saarland University, Saarbrücken, Germany
[2]Jadavpur University, Kolkata, India
[3]German Research Center for Artificial Intelligence (DFKI), Germany
{santanu.pal, josef.vangenabith}@uni-saarland.de
sudip.naskar@cse.jdvu.ac.in

Abstract

In this paper we combine two strands of machine translation (MT) research: automatic post-editing (APE) and multi-engine (system combination) MT. APE systems learn a target-language-side second stage MT system from the data produced by human corrected output of a first stage MT system, to improve the output of the first stage MT in what is essentially a *sequential* MT system combination architecture. At the same time, there is a rich research literature on *parallel* MT system combination where the same input is fed to multiple engines and the best output is selected or smaller sections of the outputs are combined to obtain improved translation output. In the paper we show that parallel system combination in the APE stage of a sequential MT-APE combination yields substantial translation improvements both measured in terms of automatic evaluation metrics as well as in terms of productivity improvements measured in a post-editing experiment. We also show that system combination on the level of APE alignments yields further improvements. Overall our APE system yields a statistically significant improvement of 5.9% relative BLEU over a strong baseline (English–Italian Google MT) and 21.76% productivity increase in a human post-editing experiment with professional translators.

1 Introduction

The term Post-Editing (PE) is defined as the correction performed by humans over the translation produced by an MT system (Veale and Way, 1997). It is often understood as the process of improving a translation provided by an MT system with the minimum amount of manual effort (TAUS Report, 2010). While MT is often not perfect, post-editing MT can yield productivity gains as post-editing MT output may require less effort compared to translating the same input manually from scratch. MT outputs are often post-edited by professional translators and the use of MT has become an important part of the translation workflow. A number of studies confirm that post-editing MT output can improve translators' performance in terms of productivity and it may positively impact on translation quality and consistency (Guerberof, 2009; Plitt and Masselot, 2010; Zampieri and Vela, 2014). The wide use of MT in modern translation workflows in the localization industry, in turn, has resulted in substantial quantities of PE data which can be used to develop APE systems.

APE (Knight and Chander, 1994) has been proposed as an automatic method for improving raw MT output, before performing actual human post-editing on it. The approach is based on collecting human corrected output of a first stage MT system and using this to train a system to correct errors produced by the MT system, possibly resulting in a productivity increase in the translation process. The advantage of APE relies on its capability to adapt to any black-box MT engine; i.e., upon availability of post-edited data, no incremental training or full re-training of the MT system is required to improve the overall translation quality of the first stage MT system that was involved in the post-edition data collection. APE assumes the availability of source texts (S_{ip}), corresponding MT output (T_{mt}) and the human post-edited (T_{pe}) version of T_{mt}, and APE systems can be modelled as an MT system between $S_{ip}\_T_{mt}$ (i.e., a joint representation of S_{ip} and T_{mt}) and T_{pe}. However, statistical APE (SAPE) systems can also be built

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2559–2570, Osaka, Japan, December 11-17 2016.

without the availability of S_{ip} using only sufficient amounts of parallel "target-side" T_{mt}–T_{pe} text within the statistical MT (SMT) framework.

Usually APE tasks focus on systematic errors made by MT systems - the most frequent ones being incorrect lexical choices, incorrect word ordering, incorrect insertion or deletion of a word. The system presented in this paper explores the use of system combination in APE. System combination in MT has been studied extensively (Matusov et al., 2006; Du et al., 2009; Pal et al., 2014), except in the context of APE. Here we use system combination architectures on three different levels: (i) sequential combination between first-stage system and APE, (ii) parallel combination of alignment systems at the level of the APE and (iii) parallel combination of APE MT systems (including the first stage MT system). More precisely, our approach makes use of a hybrid implementation of multiple alignment combination within phrase-based SAPE (PB-SAPE) and hierarchical PB-SAPE (HPB-SAPE) and a system combination framework (a multi-engine pipeline) – that combines the best translations from the enhanced PB-SAPE, HPB-SAPE and the raw MT output. The model takes T_{mt} as input and provides T_{pe} as output. As we also use the output of the original first stage MT system in some combination experiments, our set-up indirectly also uses S_{ip} information. System combination and hybrid word alignment strategies are commonly used in MT, however to the best of our knowledge the work presented in this paper is the first approach to APE that uses system combination and hybrid word alignment methods within the APE engine. System combination has been found to be a very useful technique in MT where translation hypotheses from multiple MT engines are available. Motivated by the success of system combination in MT, we applied system combination in APE. Similarly, the use of multiple word alignments has been shown to improve MT results (Pal et al., 2013). For our APE, alignments have to be produced on "monolingual" target-side data (T_{mt} and T_{pe}). A particular focus of our paper is to explore the performance of hybrid alignments based on combinations of statistical and edit-distance based aligners in this "monolingual" setting.

The remainder of the paper is organized as follows. Section 2 gives an overview of the related work. Section 3 describes the components of our SAPE system. Section 4 outlines the data and data preprocessing and the experimental setup. Section 5 presents the results of automatic and human evaluation, followed by conclusions and avenues for further research in Section 6.

2 Related Research

APE approaches cover a wide methodological range. Simard et al. (2007a) and Simard et al. (2007b) applied phrase-based SMT (PB-SMT) for post-editing that handles the repetitive nature of errors typically made by rule-based MT (RBMT) systems. The APE system was trained on the output of the rule-based system as the source language and reference human translations as the target language. This APE system was able to correct systematic errors produced by the RBMT system and reduce the post-editing effort. The approach achieved large improvements in performance not only over the baseline rule-based system but also over a similar PB-SMT used in a standalone mode. Denkowski (2015) proposed a method for real time integration of post-edited MT output into the translation model. He extracted a grammar for each input sentence and applied it to the model. Rosa et al. (2012) and Mareček et al. (2011) applied a rule-based approach to APE of English–Czech MT outputs on the morphological level. They used 20 hand-written rules based on the most frequent errors encountered in translation. The method efficiently corrects morpho-syntactic categories of a word such as number, case, gender, person as well as dependency labels. The inclusion of source-language information in APE is also useful to improve the APE performance (Béchara et al., 2011). To overcome data sparsity issues, Chatterjee et al. (2015) proposed a pipeline where the best language model and pruned phrase table are selected through task-specific dense features. Recently, a bidirectional recurrent neural network model of APE using T_{mt}–T_{pe} was proposed by Pal et al. (2016) which consists of an encoder that encodes the MT output into a fixed-length vector from which a decoder provides a post-edited (PE) translation. They reported statistically significant improvement over a strong first stage MT system baseline.

Various automatic or semi-automatic post-processing techniques to implement corrections of repetitive errors have been developed, although often the overall resulting MT output after APE still needs to be

post-edited by humans in order to produce publishable quality translation (Roturier, 2009; TAUS/CNGL Report, 2010). Even though MT and APE output often need human PE, it is often faster and cheaper to post-edit MT and APE output than to perform human translation from scratch.

System combination is a technology where multiple translation outputs from potentially very different MT systems are combined. System combination includes (i) hypothesis selection (Rosti et al., 2007a; Hildebrand and Vogel, 2010), (ii) confusion network based decoding (Matusov et al., 2006; Rosti et al., 2007b) and (iii) model combination (DeNero and Macherey, 2011). The confusion networks are built using backbone selection using either multiple hypotheses as backbones (Leusch and Ney, 2010) or a single backbone (Rosti et al., 2007b; Du et al., 2009) using TER (Snover et al., 2006) or BLEU (Papineni et al., 2002). These alignment metrics select the hypothesis that agrees most with the other hypotheses on average. System combination can improve translation quality significantly which motivated us to apply the strategy for the APE task.

Some of the research mentioned above studied the impacts of various factors and methods in APE on productivity gains. However, those studies were not conducted to observe PE effort in commercial environments. The focus of our study is twofold - to examine how existing word alignment techniques and a system combination framework can be intelligently used to improve monolingual APE, and whether the improvements in APE measured in terms of automatic evaluation metrics translate to measurable productivity gains in human post-editing in commercial translation workflows.

3 System Description

Our APE system consists of four basic components: (i) a target side mono-lingual hybrid word alignment model based on a number of alignment approaches, (ii) PB-SAPE, (iii) HPB-SAPE, and (iv) a system combination module (also including the first stage MT system). The SAPE systems are trained monolingually with Italian T_{mt} generated by Google Translate (GT) and the manually post-edited translations T_{pe}.

3.1 A Hybrid Word Alignment Model for Target Side APE

Previous research in MT demonstrates that a combination of information coming from multiple alignment models can improve translation quality. This can be achieved in different ways, e.g., by combining exactly two bidirectional alignments (Och, 2003; Koehn et al., 2003; DeNero and Macherey, 2011), combining an arbitrary number of alignments (Tu et al., 2012; Pal et al., 2013), by constructing weighted alignment matrices over 1-best alignments from multiple alignments generated by different models (Liu et al., 2009; Tu et al., 2011) etc. Below we apply an alignment combination model to APE.

Our hybrid word alignment method combines word alignments produced by three different statistical word alignment methods: (i) GIZA++ (Och and Ney, 2003) word alignment with grow-diag-final-and (GDFA) heuristic (Koehn, 2010), (ii) Berkeley word alignment (Liang et al., 2006), and (iii) Sym-Giza++ (Junczys-Dowmunt and Szał, 2012) word alignment, as well as two different edit distance based word aligners based on TER (Translation Edit Rate) (Snover et al., 2006) and METEOR (Lavie and Agarwal, 2007). We follow Pal et al. (2013) in combining word alignment tables, however, we additionally use 3-word consistent phrases to generate more alignment links (cf. Section 3.1.3). We integrate the word alignment obtained with this hybrid model into our PB-SAPE (Pal et al., 2015) and HPB-SAPE (Pal, 2015) models.

3.1.1 Statistical Word Alignment

GIZA++ is a statistical word alignment tool which implements IBM models 1–5, an HMM alignment model, as well as the IBM-6 model for covering many to many alignments. The Berkeley word aligner uses an extension of Cross Expectation Maximization and is jointly trained with HMM models. Sym-Giza++ is a modification of GIZA++ . It modifies the counting phase of each model of Giza++ allowing for updating the symmetrisized models between the iterations of the original training algorithm. Sym-Giza++ computes symmetric word alignment models with the capability of taking advantage of multi-processor systems.

3.1.2 Edit Distance-Based Word Alignment

We use two different kinds of edit distance based word aligners where alignments are based on edit distance style MT evaluation metrics – METEOR and TER.

METEOR Alignment: METEOR is an automatic MT evaluation metric which provides an alignment between a translation hypothesis H (i.e., MT output) and a reference translation R (in this case the PE translation). Given a pair of strings such as H and R to be compared the alignment is a mapping between words in H and R, which is built incrementally by the three sequences of word-mapping modules: (i) **Exact:** maps if the words are exactly the same (ii) **Porter stem:** maps if they are the same after stemming (iii) **WN synonymy:** maps if they are synonyms in WordNet. If multiple alignments exist, METEOR selects the alignment for which the word order in the two strings is most similar (i.e. having the fewest number of crossing alignment links). The final alignment is produced as the union of all stage alignments (e.g. Exact, Porter Stem and WN synonymy).

TER Alignment: TER is an edit distance based automatic MT evaluation metric that measures the ratio between the number of edit operations that are required to turn a H into R to the total number of words in R. The allowable edit operations include insertion (I), substitution (S), deletion (D) and phrase shifts (Sh). As a byproduct of finding the minimum edit distance, it produces an alignment between the hypothesis and the reference. In the monolingual SAPE task, we make use of TER alignment as a potential alignment between T_{mt} and T_{pe}. The TER alignment between a T_{pe} and T_{mt} is illustrated in Figure 1. Where, the vertical bar '|' represents a match and *I*, *D* and *S* represent the three post-editing operations – insertion, deletion and substitution, respectively.

TL_{mt}:	w_1	w_2	w_3	ϵ	w_4	w_5	w_6	w_7	w_8	w_9
	|	D	S	I	|	|	S	|	|	|
TL_{pe}:	$\bar{w}_1$	ϵ	$\bar{w}_2$	$\bar{w}_3$	$\bar{w}_8$	$\bar{w}_4$	$\bar{w}_5$	$\bar{w}_6$	$\bar{w}_7$	$\bar{w}_9$

Figure 1: TER based monolingual word alignment

3.1.3 Producing Additional Alignments for Edit-Distance Based Alignment

To generate additional alignment points between parallel sentence pairs, we perform phrase extraction (Koehn et al., 2003)[1] between T_{mt} and T_{pe}. We extract all phrase pairs, T_{mt} phrase (e) and T_{pe} phrase ($\bar{e}$), that are continuous and consistent with the edit distance based monolingual alignments. This phrase extraction process is performed individually for both TER and METEOR based alignments. A phrase pair $(e, \bar{e})$ is consistent with alignment a if Equation 1 is satisfied.

$$(\forall w_i \in e : (w_i, x) \in a \wedge x \in \bar{e}) \wedge \ (\forall \bar{w}_i \in \bar{e} : (y, \bar{w}_i) \in a \wedge y \in e) \tag{1}$$

Unaligned words in a phrase pair are aligned to all the phrase internal words in the other language. Figure 2 depicts the process of generating additional alignments where the solid links represent edit distance based alignments and the dashed links represent the newly established alignments. The newly established alignment points are added to the corresponding (i.e, TER or METEOR) alignment matrix.

3.2 Hybridization

Our method follows the following heuristic. We consider either of the alignments generated by GIZA++ with grow-diag-final-and heuristic (Koehn, 2010) (a_1), Berkeley aligner (a_2), or SymGiza++ (a_3) as the standard alignment since edit distance based TER (a_4) and METEOR (a_5) fail to align many words in the monolingual Italian MT–PE parallel sentences. From the five alignments a_1–a_5, we compute the alignment combination as follows.

[1]For this task, we use 3-words phrases.

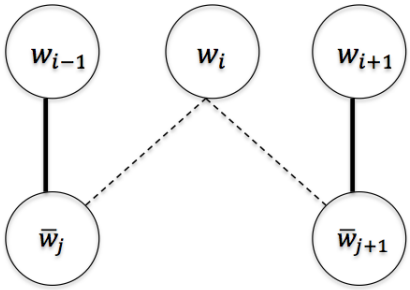


Figure 2: Producing additional alignments (w_i–$\bar{w}_j$, w_i–$\bar{w}_{j+1}$)

- **Step 1:** Choose a standard alignment[2] (S_a) from a_1, a_2 or a_3.
- **Step 2:** Produce a combined alignment $S_c = S_a \cup (a_2 \cap a_3)$, if a_1 is considered as S_a.
- **Step 3:** Delete all the alignment points $a_{ij} \in S_c$ such that $\exists a_{ik} \in a_4 \cup a_5$ where $j \neq k$.
- **Step 4:** Update S_c as $S_c = S_c \cup a_4 \cup a_5$.

3.3 System Combination for APE

Our system combination framework selects the best hypothesis translation from multiple hypotheses produced by different systems. In order to apply the system combination framework on the translations produced by our SAPE systems and the baseline MT system (Google Translate) we implemented the Minimum Bayes Risk (MBR) coupled with the Confusion Network (MBRCN) framework as described in (Du et al., 2009). The MBR decoder (Kumar and Byrne, 2004) selects for each sentence the best system output from the three outputs by minimizing BLEU (Papineni et al., 2002) loss. This output is known as the backbone. A confusion network (Matusov et al., 2006) is built from the backbone while the remaining hypotheses are aligned against the backbone using the edit-distance based alignment methods (cf. Section 3.1.2). The features used to score each arc in the confusion network (CN) are word posterior probability, target language model and length penalties. Minimum Error Rate Training (MERT) (Och, 2003) is applied to tune the CN weights. In our experiments, both APE hypotheses – PB-SAPE and HPB-SAPE, and the baseline Google Translate (GT) output are passed on to the system combination framework which produces the final system output (SC-APE).

4 Experiments

4.1 Data

The post-edition dataset for training the APE systems was developed in the MateCat[3] project. The data consist of 312K parallel sentences of Europarl and client data. The parallel data contains Italian translations (T_{mt}) produced by Google Translate from English as the source language and the corresponding post-edited Italian translations (T_{pe}) produced by professional translators. The parallel data were cleaned and processed by using a preprocessing module (see Section 4.2). After cleaning, we obtained a sentence-aligned MT–PE parallel corpus containing 213,795 sentence pairs. We randomly extracted 1,000 sentences each for the development set and test set from the parallel corpus and treated the remaining data (211,795) as the training set. The language model was built on the Italian Europarl corpus along with the PE side of the training set. The entire monolingual Italian corpus consists of 49,483,285 words.

4.2 Corpus Cleaning and Preprocessing

The MateCat corpus contains some non-Italian as well as non-English words and sentences. Therefore, we applied a language identifier (Shuyo, 2010) on both bilingual English–Italian MT output and MT

[2]Empirically best preforming aligner among the individual aligners (a_1, a_2 or a_3), is considered as S_a.
[3]https://www.matecat.com/

output–PE (Italian) parallel data. We discarded those sentence pairs from the bilingual training data which are considered as belonging to a different language or contain segment(s) in a different language. The same method was also applied to the monolingual Italian data. Next, the parallel corpus was further cleaned using the Gale-Church filtering method described in Tan and Pal (2014). We sorted the entire parallel training corpus based on sentence length and removed duplicates. We applied tokenization and punctuation normalization using the Moses scripts.

4.3 Experimental Settings

In our APE experiments we first integrated the hybrid word alignment model (cf. Section 3.1) into the SAPE engines modelled with PB-SMT (Koehn et al., 2003) and hierarchical PB-SMT (HPB-SMT) (Chiang, 2005). For building our statistical APE system, we used maximum phrase length of 7 and a 5-gram language model trained using KenLM (Heafield, 2011). Model parameters were tuned using MERT (Och, 2003) on the held-out development set.

5 Evaluation

During evaluation we take into consideration the output produced by all the three APE systems: PB-SAPE with hybrid word alignment, HPB-SAPE with hybrid word alignment and the system combination system (SC-APE) which also includes the output from the first stage system Google MT. As a baseline APE system we use a PB-SAPE system with GIZA++ alignment. The evaluation was carried out in two ways: (i) automatic evaluation and (ii) human evaluation of the 1,000 testset sentences automatically post-edited by our SAPE systems. Out of the 1,000 testset sentences, the output of the system combination based final post-editing system (SC-APE) were different from the raw Google Translate translation output for 198 sentences, i.e. only 19.8% of the GT translations are post-edited by the SC-APE system, the remaining sentences are not affected by APE. The entire testset is evaluated with automatic evaluation metrics while only the 198 sentences are subjected to human evaluation.

5.1 Automatic Evaluation

We evaluated the systems using three well known automatic MT evaluation metrics: BLEU, METEOR and TER. We also performed sentence level BLEU evaluation. Table 1 provides a comparison in terms of sentence level BLEU evaluation of the individual APE systems. Based on sentence level BLEU scores, the evaluation results presented in Table 1 show that 159 out of the 198 translations provided by the SC-APE are of better quality than the GT output. However, for the rest (39) of the translations, the GT output is of better quality than the APE output. This may be partly due to the fact that the human post-edited reference translations are biased towards GT output. However, manual analysis revealed that some of these 39 translations are indeed worse than the GT output. Overall, PB-SAPE, HPB-SAPE and SC-SAPE provide gains in terms of translation quality in 0.9%, 3.7% and 12% of the cases, respectively, as measured by S-BLEU. APE quality increases with the integration of the hybrid word alignment (HWA) model (cf. Section 3.2) into the different APE systems (cf. Table 2).

Systems	APE	GT	Tie	% Gain	% Loss
PB-SAPE	65	56	879	6.5%	5.6%
HPB-SAPE	91	54	855	9.1%	5.4%
SC-APE	159	39	802	15.9%	3.9%

Table 1: Automatic evaluation using Sentence-BLEU over 1,000 testset sentences.

Table 2 provides a comparison between the baseline PB-SAPE based on GIZA++ word alignment, PB-SAPE and HPB-SAPE based on hybrid word alignment (HWA), SC-APE and GT. The comparison is carried out in terms of BLEU, METEOR and TER scores. A general trend can be observed across all metrics. Baseline PB-SAPE system fail to improve over GT, while HWA based PB-SAPE, HPB-SAPE and SC-APE improve the translation quality over GT according to all metrics. Among the HWA based three APE systems, SC-APE performs best followed by HPB-SAPE and PB-SAPE in all metrics.

The SC-APE system provides 5.9%, 11% and 2.4% relative improvements over GT in BLEU, TER and METEOR, respectively, and all these improvements are statistically significant ($p < 0.01$). The HPB-SAPE system also provides promising improvements (4.2%, 7.3% and 1.2% in BLEU, TER and METEOR, respectively) over GT while PB-SAPE system yields in modest improvements.

Metric	PB-SAPE (Baseline)	PB-SAPE (HWA)	HPB-SAPE (HWA)	SC-APE	GT (First-Stage MT)
BLEU	59.90	62.70	63.87	**64.90**	61.26
TER	33.52	29.92	28.67	**27.52**	30.94
METEOR	69.54	73.31	73.63	**74.54**	72.73

Table 2: Automatic evaluation of the systems over 1,000 testset sentences.

5.2 Human Evaluation

The human evaluation process was carried out with 4 professional translators by introducing a polling system. The polling system provides every voter with three choices, two of which correspond to two different translation options for every source English segment. Translators act as voters and make a choice between the SC-APE output and the GT first-stage translation, based on whichever translation option looks better to them. Translators were also provided with a third option called *uncertain* (U), applicable whenever they are uncertain about which translation is better, i.e. when they deem both the GT and APE translations to be of equal quality (including equally unusable).

Table 3 shows the results of the polling scheme (human evaluation) of the raw GT output compared to the final automatic post-editing (SC-APE) output. The values in the table represent how many translations were chosen by each translator for individual systems. The polling based evaluation was carried out with 145 (of the 198) sentences. We discarded sentences containing less than six words either in source sentences or in the translations. We conducted the voting process serially to avoid any conflict between the translators. Table 3 shows that translators preferred APE output over the raw MT output. Translators did not have any knowledge about which translations are from which system as the two translation options were presented to them in random order. The winning APE system received on average 49.3% votes compared to 17% votes received by the GT system, while 33.7% votes were neutral as the translators were undecided for those sentences.

The SC-APE system received a total of 280 votes and it received votes from at least one translator for 105 unique segments, while GT received 112 total votes for 61 unique segments and 188 votes were received for 94 unique segments for the *uncertain* category. After detailed analysis we found that all 4 translators agreed on 27 APE translations, 6 GT translation and 9 neutral cases among the 145 sentences.

Translators	APE	GT	U
T1	91	22	32
T2	57	17	71
T3	72	37	36
T4	65	23	58
Average	71.5	24.7	49.2

Table 3: Outcome of polling with four expert translators for 145 sentences.

For the 145 sentences, we measured pairwise inter-annotator agreements between the translators by computing Cohen's κ coefficient (Cohen, 1960). Table 4 shows the inter-annotator agreements. The κ coefficients ranged from 0.141 (between T1 and T2) to 0.54 (between T2 and T4). The overall κ coefficient was 0.330. According to (Landis and Koch, 1977) this correlation coefficient can be interpreted as fair.

Cohen's κ	T1	T2	T3	T4
T1	-	0.141	0.424	0.398
T2	0.141	-	0.232	0.540
T3	0.424	0.232	-	0.248
T4	0.398	0.540	0.248	-

Table 4: Inter-annotator agreement between the translators.

5.3 Time and Productivity Gain Analysis

In order to investigate the performance of the APE system in terms of time and productivity gains, a completely new set of test data was distributed among the four translators. The new test data consists of real-life client segments. SC-APE and GT translations are presented separately to the translators within their daily usage interface (MateCat). Table 5 shows the statistics of how much time on average each individual translator took for the post-editing task. Table 5 also shows the average number of words (per minute, hour) post-edited by each translator. We calculated productivity gain by comparing column 2 (SC-APE) and 3 (GT) in Table 5. Table 5 shows that, SC-APE improves the productivity of the translators in general. Among the 4 translators, SAPE resulted in improved productivity for 3 translators (T1, T2 and T3), while for one translator (T4) it seems to result in productivity loss. If we look at the seconds/word, words/minute, and words/hour measures on the GT data for the 4 translators, it is easily noticeable that T1 is the most efficient post-editor, followed by T2, T4 and T3. However, when the translators work on the SAPE output, T2 is found to be the most productive while T4 is found to be the least productive. The productivity changes vary from 46.6% to -40%, which indicates that the utility of SAPE also varies from person to person. However, even taking into account the decrease in productivity of T4, average productivity increases 12.96% with SAPE. One thing to be noted here is that the productivity loss of T4 perhaps should not be considered for evaluation. We spoke to T4 after the evaluation and found that the translator was not solely concentrating on the post-editing job, switching among different jobs.

	SC-APE			**GT**				
	secs /word	**words /min**	**words /hour**	**secs /word**	**words /min**	**words /hour**	**Gain /hour**	**% Gain**
T1	2.81	21	1260	2.92	20	1200	60	5.0
T2	2.7	22	1320	3.88	15	900	420	46.6
T3	4.82	12	720	6.75	9	540	180	33.3
T4	9.80	6	360	5.84	10	600	-240	-40.0

Table 5: Post editing statistics over GT and SC-APE.

We also conducted a detailed evaluation of the post-editing carried out by the four translators. The results are reported in Table 6. Column 2 (fine grained evaluation score) in Table 6 shows the average of scores assigned to each translator by MateCat based on 5 criteria: tag issues (mismatches, white spaces), translation errors (mistranslation, additions/omissions), terminology and translation consistency, language quality (grammar, punctuation, spelling) and style (readability, consistent style and tone). MateCat also classifies each translator to one of the 4 performance levels[4] – excellent (3), acceptable (2), poor (1) and fail (0), for each of the above mentioned 5 criteria. Column 3 (weight based on quality) shows the sum of the scores indicating performance levels for the 5 criteria. By multiplying the values in column 2 and column 3, we arrive at the final assessment score assigned to each translator.

By weighting the percentage gain (cf. last column in Table 5) with the final assessment scores (cf. last column in Table 6), as in Equation 2, we obtain an average productivity increase of 21.76%. Even

[4] http://www.matecat.com/support/revising-projects/revising-translation-jobs/

	Fine grained Evaluation score (s_f)	Weight based on quality (w_q)	Final Assessment $fa = s_f \times w_q$
T1	4.46	7	31.22
T2	4.44	6	26.64
T3	1.33	2	2.66
T4	2.74	1	2.74

Table 6: Assessment of the post-editors based on their performance and quality.

considering the negative productivity of T4, this overall productivity gain is significant.

$$average\ productivity\ gain = \frac{\sum_{i=1}^{4} gain_i \times f_{ai}}{\sum_{i=1}^{4} f_{ai}} \quad (2)$$

6 Conclusions and Future Work

The use of a single statistical aligner in our PB-SMT based baseline APE fails to improve over raw Google MT output; instead it degrades the performance, as was also reported by (Béchara et al., 2011). This motivated us to use alignment combination models including both statistical and edit-distance based methods in our hybrid word alignment model for APE. By improving word alignment, the APE system automatically acquires better lexical associations and already the "hybrid alignment-based" PB-SAPE system shows improvements over the Google MT baseline. The reason for using a hierarchical phrase extraction model for APE is that it makes the model sensitive to syntactic structures. Moreover, HPB-SAPE captures global reordering by SCFG, helping to correct word order errors to some extent. Integration of our hybrid word alignment into the APE model resulted in both PB-SAPE (S_1) and HPB-SAPE (S_2) producing better translations than GT. System combination based APE (SC-APE) of S_1, S_2 and GT provided further statistically significant improvements over raw MT output. We performed statistical significance testing between GT, S_1, S_2 and SC-APE. S_1 provides statistically significant ($0.01 < p < 0.04$) improvements over GT across all metrics. Similarly S_2 yields statistically significant ($p < 0.01$) improvements over both GT and S_1 in all metrics. Our SC-APE system performs best and results in statistically significant ($p < 0.01$) improvements over all other systems across all metrics. In future, we will try bootstrapping strategies for further tuning the model and add more sophisticated features beyond the lexical level. The future study will also include a comparison of our system performance with Neural APE (Pal et al., 2016). We will also carry out experiments on different datasets including the WMT APE datasets.

Acknowledgments

We would like to thank all the anonymous reviewers for their feedback. We are also thankful to Translated SRL, Rome, Italy. They shared their data for the experiments and enabled the manual evaluation of our system. Santanu Pal is supported by the People Programme (Marie Curie Actions) of the European Union's Framework Programme (FP7/2007-2013) under REA grant agreement no 317471. Sudip Kumar Naskar is supported by Media Lab Asia, DeitY, Government of India, under the Young Faculty Research Fellowship of the Visvesvaraya PhD Scheme for Electronics & IT. Josef van Genabith is supported by funding from the European Unions Horizon 2020 research and innovation programme under grant agreement no 645452 (QT21).

References

Hanna Béchara, Yanjun Ma, and Josef van Genabith. 2011. Statistical Post-Editing for a Statistical MT System. In *Proceedings of MT summit XIII*, pages 308–315.

Rajen Chatterjee, Marion Weller, Matteo Negri, and Marco Turchi. 2015. Exploring the planet of the apes: a comparative study of state-of-the-art methods for mt automatic post-editing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 156–161.

David Chiang. 2005. A Hierarchical Phrase-based Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46.

John DeNero and Klaus Macherey. 2011. Model-based Aligner Combination Using Dual Decomposition. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 420–429.

Michael Denkowski. 2015. *Machine Translation for Human Translators*. Ph.D. thesis, Carnegie Mellon University.

Jinhua Du, Yifan He, Sergio Penkale, and Andy Way. 2009. MATREX: The DCU MT System for WMT 2009. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 95–99.

Ana Guerberof. 2009. Productivity and quality in the post-editing of outputs from translation memories and machine translation. *Localisation Focus*, 7(1):133–140.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197.

Almut Silja Hildebrand and Stephan Vogel. 2010. Cmu system combination via hypothesis selection for wmt'10. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 307–310, Uppsala, Sweden, July. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Arkadiusz Szał. 2012. SyMGiza++: Symmetrized Word Alignment Models for Statistical Machine Translation. In *Proceedings of the 2011 International Conference on Security and Intelligent Information Systems*, pages 379–390.

Kevin Knight and Ishwar Chander. 1994. Automated Postediting of Documents. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 1)*, pages 779–784.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-based Translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA.

Shankar Kumar and William Byrne. 2004. Minimum Bayes Risk Decoding for Statistical Machine Translation. In *Proceedings of the North American Association for Computational Linguistics (NAACL)*, pages 169–176.

J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159–74.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231.

Gregor Leusch and Hermann Ney. 2010. The rwth system combination system for wmt 2010. In *ACL 2010 Joint Fifth Workshop on Statistical Machine Translation and Metrics MATR*, pages 315–320.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by Agreement. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 104–111.

Yang Liu, Tian Xia, Xinyan Xiao, and Qun Liu. 2009. Weighted alignment matrices for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1017–1026, August.

David Mareček, Rudolf Rosa, Petra Galuščáková, and Ondřej Bojar. 2011. Two-step Translation with Grammatical Post-processing. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 426–432.

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing Consensus Translation from Multiple Machine Translation Systems Using Enhanced Hypotheses Alignment. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 33–40.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 160–167.

Santanu Pal, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. 2013. A Hybrid Word Alignment Model for Phrase-Based Statistical Ma-chine Translation. *ACL 2013*, pages 94–101.

Santanu Pal, Ankit Srivastava, Sandipan Dandapat, Josef van Genabith, Qun Liu, and Andy Way. 2014. USAAR-DCU Hybrid Machine Translation System for ICON 2014. In *Proceedings of the 11th International Conference on Natural Language Processing*, Goa, India.

Santanu Pal, Mihaela Vela, Sudip Kumar Naskar, and Josef van Genabith. 2015. USAAR-SAPE: An English–Spanish Statistical Automatic Post-Editing System. In *Proceedings of WMT*, pages 216–221, Lisbon, Portugal.

Santanu Pal, Sudip Kumar Naskar, Mihaela Vela, and Josef van Genabith. 2016. A neural network based approach to automatic post-editing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 281–286, August.

Santanu Pal. 2015. Statistical Automatic Post Editing. In *The Proceedings of the EXPERT Scientific and Technological workshop*, pages 13–22.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318.

Mirko Plitt and François Masselot. 2010. A productivity test of statistical machine translation post-editing in a typical localisation context. *The Prague Bulletin of Mathematical Linguistics*, 93:7–16.

Rudolf Rosa, David Mareček, and Ondřej Dušek. 2012. DEPFIX: A System for Automatic Correction of Czech MT Outputs. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 362–368.

Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie J. Dorr. 2007a. Combining outputs from multiple machine translation systems. In *In Proceedings of the North American Chapter of the Association for Computational Linguistics Human Language Technologies*, pages 228–235.

Antti-Veikko Rosti, Spyros Matsoukas, and Richard Schwartz. 2007b. Improved word-level system combination for machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 312–319.

Johann Roturier. 2009. Deploying Novel MT technology to Raise the Bar for Quality: a Review of Key Advantages and Challenges. In *Proceedings of the twelfth Machine Translation Summit*.

Nakatani Shuyo. 2010. Language Detection Library for Java.

Michel Simard, Cyril Goutte, and Pierre Isabelle. 2007a. Statistical Phrase-based Post-editing. In *In Proceedings of NAACL*, pages 508–515.

Michel Simard, Nicola Ueffing, Pierre Isabelle, and Roland Kuhn. 2007b. Rule-based translation with statistical phrase-based post-editing. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 203–206.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of association for machine translation in the Americas*, pages 223–231.

Liling Tan and Santanu Pal. 2014. Manawi: Using Multi-word Expressions and Named Entities to Improve Machine Translation. In *Proceedings of Ninth Workshop on Statistical Machine Translation*.

TAUS Report. 2010. Post editing in practice. Technical report, TAUS.

TAUS/CNGL Report. 2010. Maschine Translation Post-Editing Guidelines Published. Technical report, TAUS.

Zhaopeng Tu, Yang Liu, Qun Liu, and Shouxun Lin. 2011. Extracting Hierarchical Rules from a Weighted Alignment Matrix. In *In Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1294–1303.

Zhaopeng Tu, Yang Liu, Yifan He, Josef van Genabith, Qun Liu, and Shouxun Lin. 2012. Combining Multiple Alignments to Improve Machine Translation. In *The 24th International conference of computational linguistics (Coling 2012)*, pages 1249–1260.

Tony Veale and Andy Way. 1997. Gaijin: A Bootstrapping, Template-driven Approach to Example-based MT. In *Proceedings of the Recent Advances in Natural Language Processing*.

Marcos Zampieri and Mihaela Vela. 2014. Quantifying the Influence of MT Output in the Translators Performance: A Case Study in Technical Translation. In *Proceedings of the EACL Workshop on Humans and Computer-assisted Translation (HaCat)*, pages 93–98.

Measuring the Effect of Conversational Aspects on Machine Translation Quality

Marlies van der Wees **Arianna Bisazza** **Christof Monz**
Informatics Institute, University of Amsterdam
{m.e.vanderwees,a.bisazza,c.monz}@uva.nl

Abstract

Research in statistical machine translation (SMT) is largely driven by formal translation tasks, while translating informal text is much more challenging. In this paper we focus on SMT for the informal genre of dialogues, which has rarely been addressed to date. Concretely, we investigate the effect of dialogue acts, speakers, gender, and text register on SMT quality when translating fictional dialogues. We first create and release a corpus of multilingual movie dialogues annotated with these four dialogue-specific aspects. When measuring translation performance for each of these variables, we find that BLEU fluctuations between their categories are often significantly larger than randomly expected. Following this finding, we hypothesize and show that SMT of fictional dialogues benefits from adaptation towards dialogue acts and registers. Finally, we find that male speakers are harder to translate and use more vulgar language than female speakers, and that vulgarity is often not preserved during translation.

1 Introduction

Research in statistical machine translation (SMT) has mostly been driven by formal translation tasks. These are, however, not representative for the abundance of informal data emerging on the Internet, for which state-of-the-art SMT systems perform markedly worse (van der Wees et al., 2015a). Recent years have therefore shown an increasing effort in improving SMT for informal text, for example by normalizing noisy text to more formal text (Bertoldi et al., 2010; Banerjee et al., 2012; Ling et al., 2013a), or by enhancing formal training data with user-generated data (Banerjee et al., 2011; Jehl et al., 2012; Ling et al., 2013b).

In this paper we focus on SMT for dialogues, an informal genre that involves, by definition, multiple speakers, and is thus noticeably different from formal text (Fernández, 2014). Formal text is typically written by a single writer with a clear intention (e.g., informing or persuading), and moreover has been editorially controlled according to standards of language use. In dialogues, on the other hand, different *speakers* have different intentions and language use, affected, for example, by their *gender*. Such variations are reflected by *register*, a term referring to socio-situational language variation (Lee, 2001), and *dialogue acts*, functional actions such as questions or answers (Bunt, 1979).

While these and other dialogue-specific aspects have been analyzed in dialogue research (Schlangen, 2005; Fernández, 2014), their impact on SMT has hardly been studied. A likely explanation is the lack of adequate evaluation data, i.e., parallel conversations annotated with dialogue-specific variabels. In this paper, we take a first step towards investigating the effect of dialogue acts, speakers, gender, and register on SMT performance by measuring their respective impact on annotated dialogues from movie subtitles.

Since movie dialogues are fictional, we can only consider them as an approximation of real face-to-face conversation. However, several corpus-based studies have shown that, while movie dialogues differ from natural spoken dialogues in terms of spontaneity—they exhibit fewer incomplete utterances, hesitations, and repetitions—, they do not differ to a great extent in terms of linguistic features and main

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2571–2581, Osaka, Japan, December 11-17 2016.

messages (Forchini, 2009; Forchini, 2012; Dose, 2013). In fact, movie dialogues approximate real face-to-face conversation more than for example SMS and chat, in which media constraints influence the flow of a conversation (Whittaker, 2003; Brennan and Lockridge, 2006). Finally, Danescu-Niculescu-Mizil and Lee (2011) have shown that certain psycholinguistic and gender-specific aspects of language are also observed in fictional dialogues, indicating that conclusions drawn from experiments on fictional dialogues generalize at least partially to real spoken conversations.

The structure and contributions of this paper are as follows: First, in Section 2 we annotate multilingual movie dialogues with four dialogue-specific variables; dialogue acts, speakers, gender, and register level, and we release these annotated corpora. In Section 3 we describe our approach to measure how SMT quality is affected by each of the four dialogue-specific aspects. Next, in Section 4 we use our annotated benchmarks to show that (i) performance fluctuations among the studied dialogue dimensions are larger than randomly expected, (ii) male speakers are harder to translate than female speakers and use more vulgar language, and (iii) vulgarity is often not preserved during translation. Finally, in Section 5 we investigate and confirm the hypothesis that SMT of fictional dialogues benefits from adaptation towards various dialogue acts and registers, indicating that apart from domain adaptation, adaptation to other variables should be considered to improve SMT quality for fictional, and potentially real, dialogues.

2 Corpus construction and annotation

To measure the effect of dialogue acts, speakers, gender, and register on SMT performance we need a multilingual dialogue corpus in which utterances are annotated with each of these dialogue aspects. Unfortunately, existing corpora are limited to the English language (Janin et al., 2003; McCowan et al., 2005; Danescu-Niculescu-Mizil and Lee, 2011; Banchs, 2012; Walker et al., 2012) or contain only some of the required annotations (Wang et al., 2016). We therefore first automatically annotate multilingual movie dialogues with the above dialogue dimensions for five language pairs: Arabic-English, Chinese-English, Dutch-English, German-English, and Spanish-English.

For this annotation process we build on two main resources: (i) the OpenSubtitles corpus (Lison and Tiedemann, 2016), containing non-professionally translated subtitles, collected from `www.opensubtitles.org` and cross-lingually aligned using time information, bilingual lexicons, and cognates (Tiedemann, 2008); and (ii) the Internet Movie Script Database (IMSDb)[1], containing English movie scripts with speakers, utterances, and context (e.g., change of scenes). Using these and a number of additional resources we create our annotated corpora as follows:

1. First we collect **speaker-utterance pairs** from IMSDb scripts, based on their respective indentation sizes. We then use the Champollion sentence aligner (Ma, 2006) monolingually to align English subtitles to the English script, and follow the OpenSubtitles alignment links to align foreign subtitles to the English subtitles-script bitext. Finally, we discard the script text from the resulting 'tritext', yielding multilingual speaker-annotated dialogue corpora. Table 1a shows statistics on the average number of speakers and main characters (i.e., speakers with at least 20 utterances) per movie.

2. We learn each speaker's **gender** based on their name's occurrence in a number of online name databases and a list of gender-revealing tags such as 'aunt', 'boy', or 'grandma'. Annotations are available for ∼58% of the utterances, and the average female-to-male ratio is 1:1.7, see Table 1b.

3. We heuristically detect the **dialogue act** of each source-language utterance, considering *questions*, *exclamations* and *declaratives*. Distributions of these dialogue acts differ between language pairs, see Table 1c, but make up on average 28%, 9%, and 63% of the corpora, respectively.

4. We define a **register** label, based on the fraction of colloquial and vulgar expressions in an utterance, according to meta-information from an online dictionary[2]. We consider three register levels: *vulgar*, *colloquial*, and *neutral*, comprising circa 10%, 68%, and 22% of the corpora, respectively. Distribution statistics per language pair are shown in Table 1d.

[1] `www.imsdb.com`
[2] `www.dict.cc`

	a) Avg. speaker statistics		b) Gender statistics			c) Dialogue act statistics			d) Register statistics		
Lang. pair	#Speakers	#Main chars	%M	%F	%Unk.	%Ques.	%Excl.	%Decl.	%Vulg.	%Coll.	%Neut.
AR → EN	44.6	5.6	36.2	20.4	43.4	29.8	6.0	64.2	10.8	67.1	22.1
DE ↔ EN	47.3	5.9	36.1	19.8	44.1	27.2	12.2	60.6	10.3	67.7	22.0
ES ↔ EN	42.5	6.0	37.2	22.9	39.9	28.4	11.2	60.4	10.6	67.7	21.7
NL ↔ EN	43.8	6.0	36.5	22.3	41.2	29.0	4.2	66.8	10.1	68.3	21.6
ZH → EN	42.3	5.6	36.9	21.1	42.0	28.1	10.6	61.3	10.7	68.9	20.4

Table 1: Annotation distributions of the dialogue benchmarks. a) Main characters are speakers with 20 or more utterances. b) Uncertain gender annotations are labeled 'unknown'. c) Annotated dialogue acts: questions, exclamations, declaratives. d) Register levels: vulgar, colloquial, neutral.

Post-processing and annotation quality. The above described alignment and annotation process is done fully automatically, making it prone to errors. We therefore increase the alignment and annotation quality of our corpus by taking a number of measures: First, *before* running the Champollion aligner, we remove context information such as '[moaning]', '[clapping]' or '[chuckles]', which is most prevalent in, but not limited to, subtitles created for hearing-impaired people. In addition, we remove subtitle-specific tokens indicating continuation of a sentence on the next screen or switches in speaker turns, yielding more fluent and less fragmented utterances.

Next, *after* running the alignment process, we favor high-quality alignments by selecting only movies or movie versions (OpenSubtitles typically contains several alternative versions for a single movie (Tiedemann, 2016)) that meet the following criteria; (i) sentence lengths between both language pairs are sufficiently close, (ii) the number of sentences for which ambiguous speakers have been aligned does not exceed a given threshold, (iii) the letter distribution is sufficiently similar to the average distribution of the language. By enforcing these quality standards, we respectively reduce the number of OpenSubtitles alignment errors, Champollion alignment errors, and OCR errors. Finally, we remove utterances with ambiguous speaker labels as these are caused by erroneous Champollion alignments.

Despite efforts to improve alignment quality, our corpus still contains some incorrect alignments. To quantify these, and to verify the correctness of the automatic annotations, we manually inspect randomly selected fragments across different language pairs and movies. Based on evaluation of a sample of 120 utterances, we estimate a final alignment accuracy of 92.5%. In addition, Table 2 shows confusion matrices for manual versus automatic annotation of gender, dialogue acts, and register for the 120 selected utterances. With the overall annotation agreement per variable ranging from 85% to 97.5%, we find that our automatic annotation strategies are very accurate. Disagreement between manual and automatic annotations occurs mostly for speakers labeled with unknown gender, and between the register levels colloquial and neutral, indicating that these categories can benefit from more advanced annotation methods. For example, to better distinguish colloquial and neutral register levels, one could exploit sentence length or language model perplexity.

Gender Annot.		Automatic M	F	U	Total
Manual	**M**	42	0	8	50
	F	1	27	3	31
	U	2	2	35	39
Total		45	29	46	120

Dial.act Annot.		Automatic Q	E	D	Total
Manual	**Q**	28	0	0	28
	E	1	8	0	9
	D	1	1	81	83
Total		30	9	81	120

Register Annot.		Automatic V	C	N	Total
Manual	**V**	9	0	0	9
	C	0	74	6	80
	N	0	12	19	31
Total		9	86	25	120

Table 2: Confusion matrices for manual and automatic annotation of gender (left; M=male, F=female, U=unknown), dialogue acts (center; Q=questions, E=exclamations, D=declaratives), and register levels (right; V=vulgar, C=colloquial, N=neutral).

a) Original German-English OpenSubtitles alignment

German subtitles	**English subtitles**
Erstklassig!	Classic.
Bilanz der Werbekampagne... minus 347 Pfund.	Profit from major sales push, minus £347.
Soll ich... dir einen Cappuccino holen?	Shall I go and get you a cappuccino? // You know, ease the pain a bit. // Yeah.
Als Seelentröster?	
Ja.	Yeah.
Lieber nur einen halben.	Better make it a half.
Mehr kann ich mir nicht leisten.	All I can afford.
Logisch.	Get your logic.
Demi-Cappu. // Kommt sofort.	Demi-cappu coming right up.

b) Annotated German-English dialogue

German utterance	**English utterance**	**Annotations**
Erstklassig! Bilanz der Werbekampagne minus 347 Pfund.	Classic. Profit from major sales push, minus £347.	William, M, neutral, exclamation
Soll ich dir einen Cappuccino holen?	Shall I go and get you a cappuccino? You know, ease the pain a bit.	Martin, M, coll., question
Ja. Lieber nur einen halben. Mehr kann ich mir nicht leisten.	Yeah. Better make it a half. All I can afford.	William, M, coll., declarative
Logisch. Demi-Cappu. Kommt sofort.	Get your logic. Demi-cappu coming right up.	Martin, M, coll., declarative

Table 3: Example dialogue from Notting Hill; a) original sentences in the OpenSubtitles corpus, where // indicates a sentence boundary in many-to-one or one-to-many alignments, and b) final annotated utterances generated in our annotation pipeline, annotated with speaker (William, Martin), gender (M=male, F=female), register level (neutral, colloquial, vulgar) and dialogue act (declarative, exclamation, question). Note that sentences pairs from the original corpus are often merged in the Champollion alignment process, and that erroneous OpenSubtitles alignments are not corrected.

Table 3 shows an example dialogue with annotations and its original form in the OpenSubtitles corpus. Note that our annotated corpora differ from OpenSubtitles since only actual dialogues are included (i.e., no context), many erroneously aligned sentence pairs have been removed, and utterances are longer and less fragmented. The latter is a result of the Champollion alignment process. Since sentences in the IMSDb scripts are typically longer than those in OpenSubtitles, Champollion regularly enforces one-to-many alignments. Following the OpenSubtitles-internal alignment links then yields a large number of many-to-many alignments in which subtitles get merged into longer utterances.

Finally, table 4a lists the statistics of the benchmarks which we use in this paper and make available for download[3]. While the remainder of this paper uses the annotated fictional dialogues to analyze the impact of dialogue-specific aspects on SMT, we believe that our data set may also help to advance dialogue research—today largely confined to the English language—in a multilingual scenario.

3 Measuring dialogue effects on SMT

In this section we measure the effect of dialogue dimensions on SMT performance of fictional dialogues. To this end, we quantify BLEU (Papineni et al., 2002) fluctuations between differences in dialogue acts, speakers, gender, and register, and we determine whether the observed fluctuations are larger than randomly expected.

[3] http://ilps.science.uva.nl/resources/movie-dialogues

	a) Evaluation data		b) Training data	
Languages	#Movies	#Utterances	#Lines	#EN tokens
AR → EN	187	94K	12.3M	122M
DE ↔ EN	220	123K	9.7M	85M
ES ↔ EN	161	87K	16.3M	156M
NL ↔ EN	238	129K	17.9M	171M
ZH → EN	211	107K	6.3M	59M

Table 4: Specifications of parallel training and evaluation data. Training data consists of OpenSubtitles corpora, evaluation data consists of speaker-annotated dialogues.

3.1 Basic experimental setup

We run our experiments using an in-house phrase-based SMT system similar to Moses (Koehn et al., 2007), with features including lexicalized reordering, linear distortion with limit 5, and lexical weighting. Our systems are trained on 59M–171M tokens (depending on the language pair, see Table 4b) of unannotated OpenSubtitles corpora. We use Kneser-Ney smoothed 5-gram language models (500M–1.7B tokens, depending on the language pair) that linearly interpolate OpenSubtitles with various LDC and WMT corpora using weights optimized on a held-out set of OpenSubtitles data. Systems are tuned using pairwise ranking optimization (PRO) (Hopkins and May, 2011) on a different held-out OpenSubtitles set. The resulting systems are thus at all levels adapted to the movie dialogues translation task rather than the general domain.

3.2 Approximate randomization testing

When translating dialogues, we naturally observe *some* BLEU variations across categories such as different dialogue acts or speakers. An important question is whether the observed differences are to be expected (the null hypothesis), or whether they are indicators that one category is truly harder to translate than another (the alternative hypothesis). We test this hypothesis with an approximate randomization approach (Edgington, 1969; Noreen, 1989).

While approximate randomization (also known as approximate permutation) is often used to compare the mean and variance of *two* groups, it can be adapted to our setting with multiple categories. To this end, we compute BLEU for each of the categories in a dialogue variable (e.g., vulgar, colloquial, and neutral utterances for the dialogue variable of register level). Next, we randomly permute category labels over utterances, following the original distribution of utterances per category, and we recompute BLEU for the randomized labels.

As our test statistic of interest, we define and measure the *mean absolute BLEU difference*, which captures BLEU fluctuations between categories:

$$\text{MBD} = \frac{2}{|S|^2 - |S|} \sum_{i=1}^{|S|} \sum_{j=i+1}^{|S|} \mid \text{BLEU}_i - \text{BLEU}_j \mid \quad (1)$$

Here S is the set of categories for a given dialogue variable (e.g., $S_{\text{register}} = \{\text{vulgar, colloquial, neutral}\}$), and BLEU_i the BLEU score for category i. Each pair of categories (i, j) is compared exactly once in terms of BLEU scores. Note that MBD is a specific instance of *mean absolute difference* (MD) or *Gini mean absolute difference* (GMD), a measure of statistical dispersion which has shown to be superior to other common statistical dispersion measures such as variance, standard deviation and interquartile range (Yitzhaki, 2003).

Next, we compute the p-value by counting how often (in a total of 1,000 permutations) we observe an MBD value that is at least as extreme as the one observed for the real categories. If for a given dialogue variable $p \leq 0.05$ or $p \leq 0.01$, we conclude that this variable has a weakly or strongly significant impact on SMT quality, respectively.

For dialogue acts, gender and register we permute labels over the entire benchmark. For speakers we only permute labels within each movie since inter-movie variations in BLEU are affected by many other factors (e.g., script writers, translators, movie genre) which distract from the impact of speakers. In addition, when computing speaker-specific BLEU, we only include main characters (i.e., speakers with at least 20 utterances) to avoid BLEU's instability on small documents .

	a) BLEU per dialogue act				b) Speaker-ssMBD	c) BLEU per gender			d) BLEU per register			
Languages	Quest.	Excl.	Decl.	MBD		Male	Female	MBD	Vulg.	Coll.	Neut.	MBD
AR → EN	23.1	20.3	19.3	2.5▲	14.9%	20.4	22.0	1.6▲	17.2	21.3	19.7	2.8▲
DE → EN	24.0	20.9	21.4	2.0▲	22.3%	21.4	22.7	1.2▲	17.7	21.9	24.7	4.6▲
ES → EN	28.9	25.7	26.8	2.1▲	18.0%	26.7	28.0	1.3▲	24.4	27.4	28.8	2.9▲
NL → EN	26.7	29.0	23.7	3.5▲	23.9%	23.9	26.8	2.9▲	20.8	24.8	26.7	4.0▲
ZH → EN	15.3	15.2	12.9	1.6▲	16.6%	12.8	14.4	1.6▲	10.9	13.4	13.1	1.7▲
EN → DE	17.7	18.5	16.5	1.3▲	15.9%	16.7	17.6	0.9▲	13.6	16.6	19.9	4.2▲
EN → ES	16.8	16.5	21.5	3.3▲	13.7%	18.9	19.7	0.8▲	17.2	19.2	21.0	2.6▲
EN → NL	25.5	22.7	24.6	1.8▲	20.6%	23.9	26.5	2.6▲	21.4	24.6	26.3	3.3▲

Table 5: BLEU for dialogue acts, speakers, gender, and register, translated using baseline SMT trained and tuned on OpenSubtitles corpora. MBD: mean absolute BLEU difference, see Equation (1), all statistically significant at $p \leq 0.01$ (▲). ssMBD: percentage of movies with statistically significant speaker-MBD at $p \leq 0.05$.

4 Results

In this section we discuss the observed BLEU fluctuations (see Table 5) for our four dialogue variables of interest, guided by Spanish-to-English and English-to-German examples in Table 6, to which we provide pointers (EX#) in the text.

4.1 The effect of dialogue acts on SMT quality

As shown in Table 5a, there are substantial performance fluctuations between dialogue acts for all language pairs. However, there is no consistent pattern between different languages. For instance, we observe punctuation errors (EX1) for ES↔EN, and verb drop (EX2) and wrong word order (EX3) for EN→DE. This makes it particularly interesting to further investigate how dialogue acts can be exploited to improve translation quality of (fictional) dialogues. Improving SMT for the dialogue acts under consideration resembles cross-lingual question answering (Tiedemann, 2009). However, when considering finer dialogue act granularities, it may be profitable to exploit context information, which is not used in our current SMT setup.

4.2 The effect of speakers on SMT quality

In Table 5b we report the percentage of movies per language pair for which the observed MBD is statistically significant at $p \leq 0.05$, which is 18.6% on average. Since there are too many speakers to report individual BLEU scores, we randomly select 100 German-English movies, and compute for each of these ΔMBD as the difference between MBD for real speakers and the average MBD for randomized labels:

$$\Delta\text{MBD} = \text{MBD}_{real} - \overline{\text{MBD}}_{random} \quad (2)$$

Figure 1 shows that inter-speaker BLEU fluctuations among real speakers are often larger than inter-speaker BLEU fluctuations among randomized speaker tags. These findings suggest that, while domain adaptation is an established task in SMT, conversational SMT may benefit—at least for the fraction of movies with statistically significant speaker differences—from a fine-grained adaptation at the speaker

Annotations	ES source	ES→EN SMT output	EN reference
S1, M, neutral, declarative	se acabaron los días de olvidar, han empezado los de recordar.	the days are over, have begun to forget them to remember.	the days of forgetting are over. the days of remembering have begun.
S2, F, colloquial, question	¿sabes qué pareces cuando hablas así?	you know what you look like when you talk like that?	know when you go on what you sound like?
S1, M, vulgar, declarative	un *j***do*(EX4) hombre sensato.	a sensible man.(EX6)	i sound like a sensible *f***ing* man.
S2, F, colloquial, excl.	un pato. ¡cuac, cuac!	a duck. [quack, quack!(EX1)	you sound like a duck. quack, quack

Annotations	EN source	EN→DE SMT output	DE reference
S1, M, neutral, declarative	the days of forgettin' are over. the days of remembering have begun.	die tage von vergisst sind vorbei. die tage von an(EX2) haben begonnen.	ja, aber jetzt kommen die tage des erinnerns.
S2, F, colloquial, question	know when you go on what you sound like?	weiß, wenn du auf(EX2) was du klingst wie?(EX3)	weißt du, wie du klingst?
S1, M, vulgar, declarative	i sound like a sensible *f***ing* man.	ich klinge wie ein vernünftig *verd***ter* mann.	wie ein vernünftiger(EX5) mann.
S2, F, colloquial, declarative	you sound like a duck. quack, quack	du klingst wie eine ente. quak, quak	nein, wie eine ente! quak, quak, quak!

Table 6: Censored ES→EN (top) and EN→DE (bottom) translation examples of an annotated dialogue, originating from Pulp Fiction and involving two speakers: Pumpkin (S1, M=male) and Honeybunny (S2, F=female). Examples of phenomena marked with (EX#) are discussed in Sections 4.1–4.4.

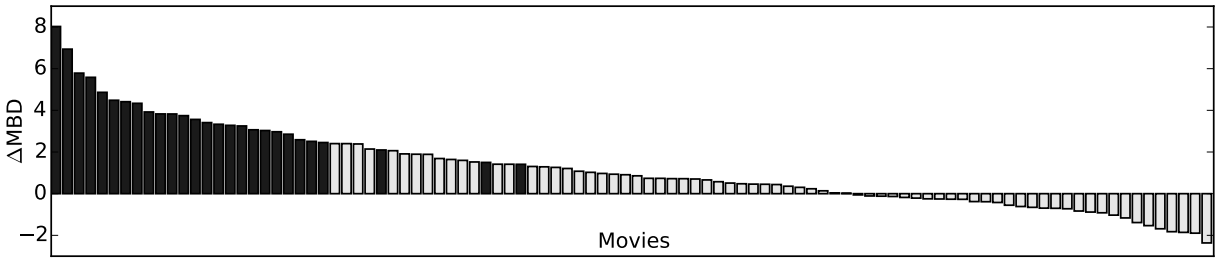


Figure 1: ΔMBD (see Section 4.2) for 100 randomly selected German-English benchmark movies. Black bars indicate movies with statistical significant positive ΔMBD at $p \leq 0.05$.

level, as proven successful in speech recognition research (Shinoda, 2011), and related to recent work on personalizing machine translation (Mirkin et al., 2015; Mirkin and Meunier, 2015).

Finally, since our analysis is carried out on fictional dialogues, it may be worth investigating to what extent BLEU scores fluctuate between actors or script writers rather than only characters, however this requires additional annotation.

4.3 The effect of gender on SMT quality

Table 5c shows that BLEU scores per gender follow a similar pattern in all language pairs. Male speakers are significantly harder to translate than female speakers, despite the fact that male speakers are likely better represented in the parallel OpenSubtitles data, based on the male-to-female ratio in our evaluation sets. However, we find that female utterances are better covered by the language model, with perplexity values on average 8% higher for males than females. This finding is consistent with recent work by Wang et al. (2016), who show that SMT can benefit from gender-adapted language models but do not provide gender-specific BLEU scores. Gender differences in movie dialogues have also been reported by Danescu-Niculescu-Mizil and Lee (2011), who show that characters adapt their language easier to females than to males.

However, we have to be careful drawing conclusions about the impact of gender on real spoken dialogues from our observations on fictional dialogues. Since the vast majority of movie scripts are written by men (Lauzen, 2016), our findings reflect differences between language of male and female characters as perceived by male writers. The observed BLEU differences might therefore be based on stereotypes rather than real gender differences. On the other hand, a tremendously large body of work has studied gender and language or discourse (Tannen, 1994; Wodak, 1997; Holmes and Meyerhoff, 2008, among others), indicating that these concepts are closely intertwined. To the best of our knowledge, our work is the first to study the impact of gender on SMT, albeit for scripted dialogues, and we believe that meta-information about a speaker's gender is also a potential source to customize SMT for real dialogues.

4.4 The effect of register on SMT quality

The results per register (Table 5d) show that SMT quality is worst for vulgar utterances and generally best for neutral sentences. While consistent with previous findings that informal language is hard to translate (van der Wees et al., 2015a), this observation cannot solely be attributed to poor model coverage, since colloquial and vulgar language are well-covered in our OpenSubtitles-trained systems.

When manually inspecting human translations for vulgar expressions, we find that these vary from literal (EX4) to very nuanced (EX5) translations, yielding inconsistent SMT output. We also observe that vulgarity is often not preserved in (both human and machine) translation (EX6): A comparison of vulgarity scores shows that, while vulgarity sometimes increases, the number of vulgar utterances in the SMT output is on average 35% lower than in the reference set.

Finally, since poor SMT quality is observed for both male characters and vulgar language, we hypothesize that the two might co-occur. Indeed, the average vulgarity scores for males are 64% higher than for females, which may in part explain the observed SMT quality between genders.

5 Preliminary adaptation towards dialogue variables

We observed that BLEU scores significantly fluctuate between differences along dialogue dimensions. This finding suggests that SMT for fictional dialogues may benefit from adaptation towards different categories along these dimensions. To verify this hypothesis we run a number of adaptation experiments, in which we adapt our baseline SMT systems towards different dialogue acts and different registers—two dialogue aspects which can be computed straightforwardly for the unannotated training corpora.

We adapt our systems at two levels: First, we create category-specific language models by interpolating our general movie dialogue language model with a language model trained on only the most relevant subset of the bitext's target side. We determine relevant sentences by applying the same annotation guidelines that were used for annotation of the benchmarks (Section 2). Second, we tune our systems on held-out sets selected according to the same criteria, thus comprising category-specific data.

We run adaptation experiments for the language pairs with the largest observed MBD; Dutch-English, English-Spanish, and Arabic-English for dialogue acts, and German-English, English-German, and Dutch-English for register. Note that the aim of our adaptation experiments is to verify whether SMT performance can benefit from a simple adaptation approach at the fine-grained level of different dialogue-specific aspects, rather than presenting a novel SMT adaptation approach.

The results of our adaptation experiments are shown in Table 7. The first observation we can make is that the adapted systems result in substantially lower mean absolute BLEU differences (MBD) for both dialogue dimensions—dialogue act and register level—for all language pairs except Arabic-English. This means that most of the adapted systems generate translations of more uniform quality with a lower degree of fluctuation in BLEU. Further, the BLEU scores for the individual categories of both dialogue dimensions show that the lower MBD scores are due to statistically significant improvements for most of the dialogue acts and registers. The only case where our simple adaptation method causes a statistically significant drop in BLEU is for the translation of questions from Dutch into English. Vulgar and colloquial language profit particularly well from language model adaptation, while results for question and exclamation marks are more variable between language pairs. Finally, we would like to emphasize that we do not claim that the simple adaption method used here constitutes the best adaptation approach

	MBD		Questions			Exclamations			Declaratives		
Language pair	Base.	Adapt.	Base.	Adapt.	Diff.	Base.	Adapt.	Diff.	Base.	Adapt.	Diff.
NL → EN	3.5	3.2	26.7	26.5	–0.2$^{\triangledown}$	29.0	28.9	–0.1	23.7	24.1	+0.4$^{\blacktriangle}$
EN → ES	3.3	2.8	16.8	17.5	+0.7$^{\blacktriangle}$	16.5	17.3	+0.8$^{\blacktriangle}$	21.5	21.5	0.0
AR → EN	2.5	2.6	23.1	24.3	+1.2$^{\blacktriangle}$	20.3	20.4	+0.1	19.3	20.9	+1.6$^{\blacktriangle}$

	MBD		Vulgar			Colloquial			Neutral		
Language pair	Base.	Adapt.	Base.	Adapt.	Diff.	Base.	Adapt.	Diff.	Base.	Adapt.	Diff.
DE → EN	4.6	4.2	17.7	18.4	+0.7$^{\blacktriangle}$	21.9	22.7	+0.8$^{\blacktriangle}$	24.7	24.7	0.0
EN → DE	4.2	3.8	13.6	14.6	+1.0$^{\blacktriangle}$	16.6	17.8	+1.2$^{\blacktriangle}$	19.9	20.3	+0.4$^{\blacktriangle}$
NL → EN	4.0	2.8	20.8	23.1	+2.3$^{\blacktriangle}$	24.8	25.6	+0.8$^{\blacktriangle}$	26.7	27.3	+0.6$^{\blacktriangle}$

Table 7: Results of adaptation experiments. Top: adaptation towards dialogue acts for the 3 language pairs with the largest mean absolute BLEU difference (MBD, see Equation (1)) between dialogue acts. Bottom: adaptation towards registers for the 3 language pairs with the largest MBD between register levels. Statistical significance against the baseline at $p \leq 0.05$ ($\triangle$/$\triangledown$) and $p \leq 0.01$ ($\blacktriangle$/$\blacktriangledown$) is measured using approximate randomization (Riezler and Maxwell, 2005).

for dialogue-specific phenomena, but rather that already a simple adaptation approach can benefit from our dialogue-specific annotations.

6 Conclusions and implications

While SMT research has mostly been driven by formal translation tasks, very little work has been reported on SMT for informal genres such as dialogues, a genre that differs substantially from formal text and thus poses different translation challenges. Following the previous finding that genre and topic affect SMT differently (van der Wees et al., 2015b), we have in this paper analyzed the impact of dialogue-specific aspects in SMT for fictional dialogues. We created and released a movie-dialogue benchmark in which utterances are annotated with dialogue acts, speakers, gender, and register, and we studied the effect of these four variables on SMT performance.

Our analysis shows that BLEU fluctuations for all variables are often significantly larger than randomly expected. When looking at specific dialogue aspects, we found that the register level has a significant impact on translation quality, with translations of vulgar utterances being of substantially lower quality than neutral or even colloquial utterances for all language pairs under consideration. Similarly we found large variations in translation quality between different dialogue acts, although we did not detect a consistent pattern between different languages; e.g., questions can be more difficult to translate than exclamations for one language pair, while the reverse is true for another language pair.

These findings suggest that conversational SMT may benefit from adaptation at fine-grained levels. We tested and confirmed this hypothesis in a series of simple adaptation experiments.

Finally, we found that male speakers are harder to translate and use more vulgar language than female speakers, and that vulgarity is often not preserved during translation. While our analyses are carried out on fictional dialogues, we believe that our findings generalize at least partially to other types of dialogues, and are thus valuable for advancing conversational SMT.

Acknowledgements

This research was funded in part by the Netherlands Organization for Scientific Research (NWO) under project number 639.022.213. We thank Raquel Fernández for sharing valuable insights from dialogue research, Tobias Schnabel for providing feedback on the approximate randomization approach, and the anonymous reviewers for their thoughtful comments.

References

Rafael E. Banchs. 2012. Movie-DiC: a movie dialogue corpus for research and development. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 203–207.

Pratyush Banerjee, Sudip Kumar Naskar, Johann Roturier, Andy Way, and Josef van Genabith. 2011. Domain adaptation in statistical machine translation of user-forum data using component level mixture modelling. In *Proceedings of the XIII Machine Translation Summit*, pages 285–292.

Pratyush Banerjee, Sudip Kumar Naskar, Johann Roturier, Andy Way, and Josef van Genabith. 2012. Domain adaptation in SMT of user-generated forum content guided by OOV word reduction: Normalization and/or supplementary data. In *Proceedings of the 16th Conference of the European Association for Machine Translation*, pages 169–176.

Nicola Bertoldi, Mauro Cettolo, and Marcello Federico. 2010. Statistical machine translation of texts with misspelled words. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 412–419.

S.E. Brennan and Calion B. Lockridge. 2006. Computer-mediated communication: A cognitive science approach. In *Encyclopedia of language and linguistics,*, pages 775–780. Elsevier Ltd., Oxford, UK.

Harry Bunt. 1979. Conversational principles in question-answer dialogues. In *Zur Theorie der Frage*, pages 119–141. Narr Verlag.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 76–87.

Stefanie Dose. 2013. Flipping the script: A corpus of american television series (CATS) for corpus-based language learning and teaching. *Corpus Linguistics and Variation in English: Focus on Non-native English*, 13.

Eugene S. Edgington. 1969. Approximate randomization tests. *The Journal of Psychology*, 72(2):143–149.

Raquel Fernández. 2014. Dialogue. In *The Oxford Handbook of Computational Linguistics (2 ed.)*. Oxford University Press.

Pierfranca Forchini. 2009. Spontaneity reloaded: American face-to-face and movie conversation compared. In *Corpus Linguistics*.

Pierfranca Forchini. 2012. *Movie language revisited. Evidence from multi-dimensional analysis and corpora*. Peter Lang, Internationaler Verlag der Wissenschaften.

Janet Holmes and Miriam Meyerhoff. 2008. *The handbook of language and gender*, volume 25. John Wiley & Sons.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362.

Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, et al. 2003. The ICSI meeting corpus. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP'03)*, pages I–364. IEEE.

Laura Jehl, Felix Hieber, and Stefan Riezler. 2012. Twitter translation using translation-based cross-lingual retrieval. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 410–421.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 177–180.

Martha M. Lauzen. 2016. The celluloid ceiling: Behind-the-scenes employment of women on the top 100, 250, and 500 films of 2015. Technical report, Center for the study of women in television and film.

David Y.W. Lee. 2001. Genres, registers, text types, domains and styles: Clarifying the concepts and navigating a path through the BNC jungle. *Language Learning & Technology*, 5(3):37–72, September.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2013a. Paraphrasing 4 microblog normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 73–84.

Wang Ling, Guang Xiang, Chris Dyer, Alan Black, and Isabel Trancoso. 2013b. Microblogs as parallel corpora. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 176–186.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*.

Xiaoyi Ma. 2006. Champollion: A robust parallel text sentence aligner. In *LREC 2006: Fifth International Conference on Language Resources and Evaluation*, pages 489–492.

Iain McCowan, Jean Carletta, W Kraaij, S Ashby, S Bourban, M Flynn, M Guillemot, T Hain, J Kadlec, V Karaiskos, et al. 2005. The AMI meeting corpus. In *Proceedings of the 5th International Conference on Methods and Techniques in Behavioral Research*, volume 88.

Shachar Mirkin and Jean-Luc Meunier. 2015. Personalized machine translation: Predicting translational preferences. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2019–2025.

Shachar Mirkin, Scott Nowson, Caroline Brun, and Julien Perez. 2015. Motivating personality-aware machine translation. In *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1102–1108.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley-Interscience.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Stefan Riezler and John T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64.

David Schlangen. 2005. Modelling dialogue: Challenges and approaches. *Künstliche Intelligenz*, 3/05:23–28.

Koichi Shinoda. 2011. Speaker adaptation techniques for automatic speech recognition. In *Proceedings of APSIPA ASC*.

Deborah Tannen. 1994. *Gender and discourse*. Oxford University Press.

Jörg Tiedemann. 2008. Synchronizing translated movie subtitles. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*, pages 1902–1906.

Jörg Tiedemann. 2009. Translating questions for cross-lingual QA. In *Proceedings of the 13th Annual Conference of the European Association for Machine Translation*, pages 112–119.

Jörg Tiedemann. 2016. Finding alternative translations in a large corpus of movie subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, pages 3518–3522.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2015a. Five shades of noise: Analyzing machine translation errors in user-generated text. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 28–37.

Marlies van der Wees, Arianna Bisazza, Wouter Weerkamp, and Christof Monz. 2015b. What's in a domain? Analyzing genre and topic differences in statistical machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 560–566.

Marilyn A Walker, Grace I Lin, and Jennifer Sawyer. 2012. An annotated corpus of film dialogue for learning and characterizing character style. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 1373–1378.

Longyue Wang, Xiaojun Zhang, Zhaopeng Tuy, Andy Way, and Qun Liu. 2016. Automatic construction of discourse corpora for dialogue translation. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, pages 2748–2754.

Steve Whittaker. 2003. Theories and methods in mediated communication. In *The Handbook of Discourse Processes*, pages 243–286. Erlbaum.

Ruth Wodak. 1997. *Gender and discourse*. Sage.

Shlomo Yitzhaki. 2003. Gini's mean difference: A superior measure of variability for non-normal distributions. *Metron*, 61(2):285–316.

Enriching Phrase Tables for Statistical Machine Translation Using Mixed Embeddings

Peyman Passban, Qun Liu and Andy Way
ADAPT Centre
School of Computing
Dublin City University, Ireland
firstname.lastname@adaptcentre.ie

Abstract

The phrase table is considered to be the main bilingual resource for the phrase-based statistical machine translation (PBSMT) model. During translation, a source sentence is decomposed into several phrases. The best match of each source phrase is selected among several target-side counterparts within the phrase table, and processed by the decoder to generate a sentence-level translation. The best match is chosen according to several factors, including a set of bilingual features. PBSMT engines by default provide four probability scores in phrase tables which are considered as the main set of bilingual features. Our goal is to enrich that set of features, as a better feature set should yield better translations. We propose new scores generated by a Convolutional Neural Network (CNN) which indicate the semantic relatedness of phrase pairs. We evaluate our model in different experimental settings with different language pairs. We observe significant improvements when the proposed features are incorporated into the PBSMT pipeline.

1 Introduction

PBSMT models sentence-level translation with a phrase-based setting in which sentences are decomposed into different phrases (Koehn et al., 2007; Koehn, 2009). At each step, for a given source phrase the best candidate among the target phrases is selected as its translation. Phrasal translations are combined together to produce the sentence-level translation. This is a high-level view of PBSMT and there are many other processes involved in the main pipeline. Different bilingual and monolingual features are taken into account to make the final translation as adequate and fluent as possible. In this paper we only focus on the phrase-pairing process and try to enrich that part. The standard baseline bilingual features in the PBSMT pipeline by default are: the phrase translation probability $\phi(e|f)$, inverse phrase translation probability $\phi(f|e)$, lexical weighting $lex(e|f)$ and inverse lexical weighting $lex(f|e)$. These scores are computed based on the co-occurrence of phrase pairs in training corpora and do not indicate any other information about phrases, their relation or context. Our goal in this research is to extend this set of features by incorporating semantic information of phrase pairs.

Word embeddings are numerical representations of words which preserve semantic and syntactic information about words themselves and their context (Huang et al., 2012; Luong et al., 2013; Mikolov et al., 2013a; Mikolov et al., 2013b). They also preserve information about word order. These types of contextual, syntactic and semantic information can be quite useful for machine translation (MT), but being by its very nature a bilingual application, it requires bilingual (cross-lingual) information. Accordingly, we need methods to train bilingual embeddings via which we can access syntactic and semantic information about the source side as well as the target.

Several papers have explored the training of bilingual embeddings (Mikolov et al., 2013b; Zou et al., 2013; Cho et al., 2014; Gouws et al., 2015; Passban et al., 2015a; Zhao et al., 2015; Passban et al., 2016) with different architectures for different tasks such as MT and document classification. In our work we also try to follow the same research line. We propose a multi-plane data structure and a CNN to train

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2582–2591, Osaka, Japan, December 11-17 2016.

mixed embeddings. Using the proposed data structure, source and target words are linked together, so we call our embeddings mixed. Our model is a bilingual extension to well-known embedding models (Mikolov et al., 2013a; Pennington et al., 2014) with a quite different architecture which is fine-tuned for MT tasks.

The reminder of the paper is structured as follows. In Section 2 we briefly review some similar models which train bilingual embeddings. Section 3 discusses our neural model and how we incorporate results from our model into the PBSMT pipeline. Section 4 explains the results from several experiments to show the impact of the proposed model. Finally, Section 5 concludes the paper with some avenues for future work.

2 Background

One of the most successful neural models proposed for training word embeddings is *Word2Vec* (Mikolov et al., 2013a). In such models words are encoded into an n-dimensional feature space and represented by numerical vectors called embeddings. Embeddings are able to preserve different types of information about words. *Word2Vec* trains word-level embeddings. Le and Mikolov (2014) proposed a new architecture which is an extension to *Word2Vec* which scales up the model to train document-level (phrase, sentence and any chunk of text) embeddings. Although these models are very useful for natural language processing (NLP) tasks, they only provide monolingual information which is not adequate for cross-lingual NLP, MT, multilingual text classification etc. Therefore, some models have been proposed in this regard.

A sample of training bilingual embeddings was proposed in Mikolov et al. (2013b) and Zhao et al. (2015). They separately project words of source and target languages into embeddings, then try to find a transformation function to map the source embedding space into the target space. The transformation function was approximated using a small set of word pairs. This approach allows the construction of a word-level translation engine with a very large amount of monolingual data and only a small number of bilingual word pairs. The cross-lingual transformation mechanism enables the engine to search for translations for OOV (out-of-vocabulary) words by consulting a monolingual index which contains words that were not observed in the parallel training data. This is a word-level translation/transformation but clearly MT is more than a word-level process. To go beyond and train document-level bilingual embeddings several models have been proposed and applied to MT and document classification tasks (Zou et al., 2013; Cho et al., 2014; Gouws et al., 2015; Passban et al., 2016). These models have different architectures which we try to address in the next sections.

3 Proposed Model

The proposed pipeline can be briefly explained in five steps: a) A PBSMT engine is trained and tuned on a bilingual parallel corpus. b) By use of the same training corpus, mixed embeddings are trained by our CNN. c) The proposed CNN takes a pair of translationally equivalent source and target sentences (s, t) and tries to link related words from both sides. d) Input words are mixed through a multi-plane data structure and a specific convolution function. e) At the end of training, we wish to have embeddings which preserve different types of monolingual and bilingual information. f) Finally, we use mixed embeddings to enrich the bilingual feature set of the phrase table. Sections 3.1 and 3.2 explain the training method and the way we use word embeddings in the PBSMT pipeline, respectively.

3.1 Training Mixed Embeddings

Generally, to train word embeddings a neural network processes an input sequence of words at each pass. One word is randomly selected (w_p) from the input sequence which is considered as the sequence label and expected to be predicted at the output layer. This architecture is known as the CBOW (Continuous-Bag-Of-Words) model (Mikolov et al., 2013a) and all other embedding models can be viewed as variations of this model. The goal of such a process is to use context words (preceding and following words around w_p) to predict w_p. With this technique, word embeddings are informed about contextual information and word order. Moreover, since they are trained in a shared space, they are connected to each other.

In the forward pass, embeddings for context words are combined to make the prediction. Each neural network has a loss function which penalizes wrong predictions. Error values are computed based on the loss function and back-propagated to the network. Network parameters are updated with respect to error values as word embeddings comprise part of those parameters, they are updated at each pass. For more information about embedding learning, see Goldberg and Levy (2014).

In existing embedding models the input data structure is a matrix and the setting is monolingual. Each column in the matrix includes an embedding (which is a vector) for one of context words. In our model we expand the input matrix to a 2-plane tensor (each plane is a matrix) in order to change the monolingual setting into a bilingual version. Training instances in our setting are a pair of translationally equivalent source and target sentences. The first and second planes include embeddings for source and target words, respectively. In embedding models w_p is not included in the input matrix. Similarly we do not have w_p in our input tensor. We randomly select a word either from the source or target side of (s, t) as w_p and remove all information about it and its translation(s)[1] from the input tensor. What remains after removing w_p and its translation(s) are 'context words'. Embeddings for source context words are placed in the first plane by the order of their appearance in s. Then the counterpart/translation of each column in the first plane is retrieved (among target-side embeddings) according to the alignment function, and placed in the same column in the second plane.

The example below clarifies the structure of the input tensor. For *s="I will ask him to come immediately ."* with a Farsi[2] translation *t="mn āz āū xvāhm xvāst ke fūrn byāyd ."*, the word alignment provided by a PBSMT engine is $a(s, t)$ = [0-0, 1-3, 2-4, 3-1, 3-2, 4-7, 5-6, 6-7, 7-6, 8-8] which is illustrated in Figure 1.

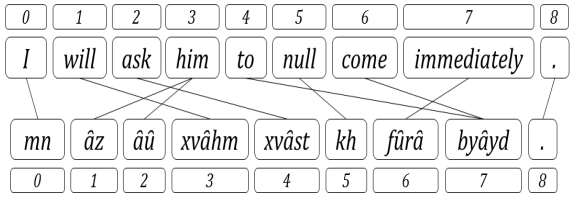


Figure 1: Word alignments provided by a PBSMT engine for a given (s, t) example. *'him'* is selected as w_p so *'him'* and its translations are excluded from the input tensor.

$a(.)$ is an alignment function which generates a list of i-j tuples. i indicates the position of a source word w_i^s within s and j is the position of the translation of w_i^s within t (namely w_j^t). If *'him'* is randomly selected as w_p, *'him'* and its translations (*'āz'* and *'āu'*) are all removed from the input tensor, and embeddings for the rest of the words are loaded into the input tensor according to i-j tuples. Embeddings for source words except *'him'* are sequentially placed in the first plane. For the second plane, each column c includes the embedding for the translation of a source word located in the c-th column of the first plane. If the embedding of each word is referred to by $\mathcal{E}$, the order of source and target embeddings in the first and second planes is as follows:

$$p_1 = [\mathcal{E}(w_0^s), \mathcal{E}(w_1^s), \mathcal{E}(w_2^s), \mathcal{E}(w_4^s), \mathcal{E}(w_5^s), \mathcal{E}(w_6^s), \mathcal{E}(w_7^s), \mathcal{E}(w_8^s)]$$

$$p_2 = [\mathcal{E}(w_0^t), \mathcal{E}(w_3^t), \mathcal{E}(w_4^t), \mathcal{E}(w_7^t), \mathcal{E}(w_5^t), \mathcal{E}(w_7^t), \mathcal{E}(w_6^t), \mathcal{E}(w_8^t)]$$

.

Our CNN takes the 2-plane tensor as its input and combines its planes through a specific convolution function. The new multi-plane convolution function is a simple extension of the standard convolution function which is formulated as in (1). It takes a multi-plane data structure (in our case, 2-plane) and

[1]Sometimes the alignment function assigns more than one target word to a given source word.

[2]We used the DIN transliteration standard to show the Farsi alphabets: `https://en.wikipedia.org/wiki/Persian_alphabet`.

generates another data structure with one or more planes:

$$\mathcal{M}_{p,i,j} = \sum_{l=1}^{|P|} \sum_{w=1}^{w_{\mathcal{F}}} \sum_{h=1}^{h_{\mathcal{F}}} \mathcal{F}_{p,l,w,h} \times \mathcal{I}_{l,(i-1)+w,(j-1)+h} \quad (1)$$

where $\mathcal{I}$, $\mathcal{M}$ and $\mathcal{F}$ are the input, output and filter,[3] respectively. $\mathcal{M}$, is the results of the fusion process (mixing embeddings) should have one to many planes which are referred to by the p subscript. Each plane in $\mathcal{M}$ is a matrix and its values are accessible by the (i, j) coordinates, i.e. $\mathcal{M}_{p,i,j}$ shows the value of the i-th row and the j-th column in the p-th plane. l is the plane index and $|P|$ shows the number of input planes. In our setting both $\mathcal{I}$ and $\mathcal{F}$ are 2-plane tensors so $|P| = 2$. $w_{\mathcal{F}}$ and $h_{\mathcal{F}}$ are the width and height of each plane in the filter. Finally the (w, h) tuple shows the coordinates of each plane in the filter $\mathcal{F}$. The first subscript of filter (p) indicates to which plane in $\mathcal{M}$ the filter belongs.

3.1.1 Network Architecture

The first layer of our architecture is a lookup table which includes word embeddings. The lookup table can be viewed as a matrix of weights whose values are updated during training. For each training sample (s, t), w_p is selected. Embeddings for context words are retrieved from the lookup table and placed within the input tensor, based on the alignment function. Through the multi-plane convolution, planes are convolved together. In our setting, the output of convolution is a matrix ($\mathcal{M}$). Based on Equation (1) for multi-plane convolution, it is possible to map the 2-plane input to a new structure with one to many planes, however we generate a structure with only one plane (a matrix). According to experimental results structures with more than one plane provide slightly better results but considerably delay the training phase. The new generated matrix contains information about source and target words, their order and relation. After multi-plane convolution we apply *max-pooling* to select the strongest features in 2×2 windows. We reshape the matrix to a vector and apply non-linearity by a *Rectifier* function. To prevent over-fitting, we place a *Dropout* layer (Srivastava et al., 2014) with $p = 0.4$ after *Rectifier*. The output of the *Dropout* layer is a vector which is passed to a *Softmax* layer.

Using the *Softmax* layer we try to predict the right class of the input which should be w_p. *Softmax* is a scalar function which maps its input to a value in the range [0,1], which is interpreted as the probability of predicting w_p given the input. In our network, similar to most embedding-training models, the objective is to maximize the log probability of w_p given the context, as in (2):

$$\frac{1}{n} \sum_{j=1}^{n} \log p(w_p | C_{2p}) \quad (2)$$

where C_{2p} is the context information represented by the 2-plane data structure. The network was trained using stochastic gradient descent and back-propagation (Rumelhart et al., 1988). All parameters of the model are randomly initialized over a uniform distribution in the range [-0.1,0.1]. Filter, weights, bias values and embeddings are all network parameters which are tuned during training. Our embedding size is 100 in all experiments. The network architecture is illustrated in Figure 2.

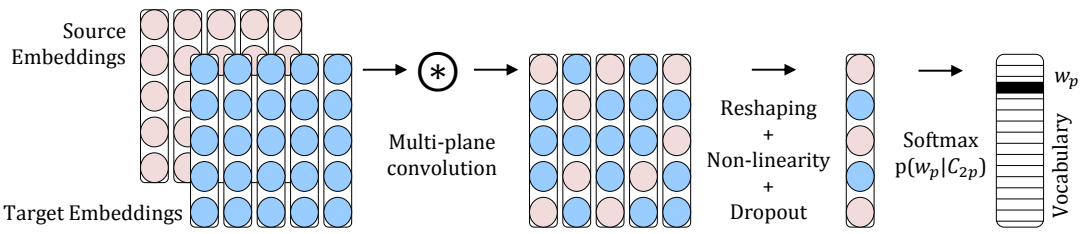


Figure 2: Network Architecture.

[3] *'Filter'* is referred to as *'Kernel'* in the literature.

3.2 Using Mixed Embeddings in the SMT Pipeline

After training we have mixed/bilingual word embeddings. Since we link/align equivalent words, those which are each other's translation are expected to have close embeddings. In the phrase table, parallel phrase pairs are stored, with their relevance being indicated by four default probabilities. We add two new scores to those probabilities.

As the first proposed score, for a given pair of source and target phrases, we retrieve embeddings for all source words and compute the average of those embeddings which gives us $\overline{v}_s$. Embeddings are numerical vectors, so we can easily compute the average of several vectors which produces another vector with the same dimensionality. We do the same for the target phrase, namely we retrieve embeddings for all target words and compute the average which provides $\overline{v}_t$. Using the Cosine similarity, we compute the distance between $\overline{v}_s$ and $\overline{v}_t$ and map the value to the range [0,1]. This new number is the first score we define to enrich the bilingual feature set.

As the second score, for each phrase pair we compute $score_2(s_p, t_p)$ by the computation explained in (3):

$$score_2(s_p, t_p) = \prod_{i=1}^{|s_p|} \frac{1}{|\{\alpha_i\}|} \sum_{i \in \{\alpha_i\}} \mathbf{C}(w_i^{s_p}, \alpha(w_i^{s_p})) \tag{3}$$

where $\alpha(.)$ is a word-level alignment function for the phrase pair (s_p, t_p), $\{\alpha_i\}$ is the set of target positions aligned to $w_i^{s_p}$ and $\mathbf{C}$ is the value of the Cosine distance mapped to the range [0,1]. For each phrase pair in the phrase table, we add these two new scores as the additional bilingual features and tune the PBSMT engine using both previous and new features.

4 Experimental Results

To show the impact of the proposed scores we perform several experiments. In the first experiment we evaluate the model on the English–French (En–Fr) pair. Results are reported in Table 1.

System	**En→Fr**			**Fr→En**		
	200K	500k	1M	200k	500k	1M
Baseline	34.4	35.2	35.7	33.8	34.5	35.4
Extended	35.5	36.0	36.0	35.1	35.2	35.5
Improvement	**+1.1**	**+0.8**	**+0.3**	**+1.3**	**+0.7**	+0.1

Table 1: Experimental results on the En–Fr pair. The numbers indicate the BLEU scores. The bold-faced scores indicate improvements are statistically significant according to paired bootstrap re-sampling with $p = 0.05$.

BLEU (Papineni et al., 2002) is used as the evaluation metric. We trained 3 baseline systems over datasets of 200K, 500K and 1 million (1M) parallel sentences. As the test set we use a corpus of 1.5K parallel sentences and the validation set includes 2K parallel sentences. All sentences are randomly selected from the En–Fr part of the Europarl (Koehn, 2005) collection. In our models we use 5-gram language models trained using the IRSTLM toolkit (Stolcke, 2002) and we tune models via MERT (Och, 2003). The extended systems those which include new scores within their phrase tables. In the extended systems we keep everything unchanged. only adding two new scores to the phrase table and the re-tune the PBSMT engine. The bold-faced scores indicate improvements are statistically significant according to paired bootstrap re-sampling with $p = 0.05$ (Koehn, 2004).

As Table 1 shows, adding new features considerably boosts the PBSMT model, especially for small-size datasets. For example, the BLEU score reported for the En→Fr system trained on the 500K dataset is 35.2, while a better performance (35.5) is achievable on the smaller 200K dataset in the presence of the new features. Usually as the size of the phrase table grows, the impact of such models attenuate, as large(r) phrase tables are rich enough to cover different cases and do not need to be enriched. Accordingly, such models are more suitable for small/medium-size datasets or low-resource languages. Other

similar models (Gao et al., 2013; Zou et al., 2013) were also evaluated on datasets with almost 500K parallel sentences and reported similar improvements.

4.1 Discussion

In this section we try to address different issues about the proposed model to analyse it from different perspectives. First we wish to discuss the advantages of the proposed architecture. We use a convolutional model to mix source and target words. We believe that the convolutional module enables the network to generate high-quality embeddings. To show the impact of this module we train a simple baseline using *Word2Vec*.

For the baseline model, we prepare a mixed training corpus, so that for each training sentence in the corpus, some words (up to 40% of whole words) are randomly selected and substituted with their translations. For example, to train mixed embeddings for their use in the En→Fr PSMT engine, some English words are randomly replaced with their French translations, and then the *Word2Vec* model is used to train mixed embeddings. For the Fr→En model we do the same and manipulate the French training corpus. In this model, instead of combining words via the NN we explicitly mix them in the training corpus so the context window for each w_p includes words from both sides. To train this model we used the *CBOW* setting (Mikolov et al., 2013a) with a context window of 10 words. For the En→Fr and Fr→En directions the new *Word2Vec*-based model performs with BLEU scores of 34.2 and 34.1, respectively. The new model degrades the En→Fr baseline engine (see the 200K-baseline model in Table 1) by -0.2 BLEU points and the improvement provided for the Fr→En engine is only +0.3. In the new model words are explicitly mixed together but the final result is not satisfactory. Therefore, the simple combination of words is not enough to boost the PBSMT engine and words should be processed efficiently.

The *Word2Vec*-based model is a simple baseline to show the impact of the proposed CNN but we wish to compare our model to other state-of-the-art and more powerful models. Unfortunately, datasets and source-code for the models by Gao et al. (2013) and Zou et al. (2013) (which are the most related works to ours) are not available so we cannot directly compare ours to those models. Gao et al. (2013) also uses an in-house translation decoder which makes the comparison even harder. Recently, Passban et al. (2016) proposed a feed-forward architecture to train bilingual phrase embeddings. The idea behind their model is similar to that of Devlin et al. (2014). They performed their evaluation on Farsi (Fa) which is a low-resource and morphologically rich language. Clearly, this model is an appropriate alternative to be compared to our CNN which can (partly) show the advantages/disadvantages of the proposed CNN compared to the feed-forward architecture. In Passban et al. (2016), the network concatenates phrase-level embeddings of source and target phrases to predict w_p, and error values are back-propagated to word and phrase embeddings. After training, the model provides bilingual phrase- and word-level embeddings. They use the similarity between parallel phrases as a new feature function (which is refered to as *sp2tp* in their paper). Using the same dataset and experimental setting, we compare the CNN to that model. The En–Fa model is trained using the TEP++ corpus (Passban et al., 2015b) which is a collection of ∼600K parallel sentences. The test, validation and training sets include 1K, 2K and 500K sentences, respectively. Results for this experiment are illustrated in Table 2. We tried to compare our CNN to a feed-forward architecture on the same task and same dataset. As the table shows, the proposed CNN performs better than a similar feed-forward architecture.

System	**En→Fa**	**Fa→En**
Baseline	21.03	29.21
Passban et al. (2016)	21.46 (+0.43)	29.71 (+0.50)
Our model	21.58 (**+0.55**)	29.93 (**+0.72**)

Table 2: Experimental results on the En–Fa pair.

In addition to quantitative evaluations, we look at the output of our engines to confirm that the proposed

pipeline affects the translation process in a particular way. Some examples are provided in Table 3. Based on our analysis, the new features positively affect word selection. Extended translations include better words than baseline translations. Furthermore, translations provided by the extended models have better grammatical structures. They are also semantically close(r) to the reference translations. The second and fourth examples are a clear indication to these issues. For the fourth example, in spite of a very low BLEU score the translation provided by the extended engine is a perfect translation.

System	Translation	sBLEU
	Example 1 (En)	
Reference	i would like to return to the matter of the embargo to conclude .	100
Baseline	i would like to revisit (to) the issue of the embargo in conclusion .	27.58
Extended	i would like to return to the issue of the embargo to conclude .	78.25
	Example 2 (En)	
Reference	subject to these remarks , we will support the main thrust of the fourçans report . however , we have to criticise the commission ' s economic report for lacking vision .	100
Baseline	it is on these observations that we shall broadly the (main thrust) fourçans report (. however) we must also consider the economic report from the commission (' s) a certain lack(ing) of breath .	7.39
Extended	it is under these comments that we will approve in its broad outlines the fourçans report by UNK however , (we) the commission ' s economic report a certain lack(ing) of breath .	22.37
	Example 3 (Fr)	
Translation	furthermore , the european union will always be open to all europeans who accept its values .	-
Reference	par ailleurs , l ' union européenne sera toujours ouverte à tous les européens qui acceptent ses valeurs .	100
Baseline	en outre , l ' union européenne devra toujours être ouverte à tous ces européens qui acceptent de ses valeurs .	33.46
Extended	en outre , l ' union européenne sera toujours ouverte à tous les européens qui acceptent de ses valeurs .	74.83
	Example 4 (Fa)	
Translation	anyway your collection will have its emerald star back in .	-
Reference	به هرحال آن ستاره سبز به کلکسیون آکواریوم تو برمیگردد .	100
Baseline	به هرحال (آن) مجموعه جا دوستان آنها زمردین میارم .	13.74
Extended	به هرحال آن مجموعه سبز به کلکسیون آکواریوم (تو) میاید .	26.48

Table 3: Translation results from different models. Differences between reference and candidate translations are underlined and missing translations are shown within parentheses. sBLEU indicates the sentence-level BLEU score.

Finally, we asked native French and Farsi speakers to evaluate our results from the perspectives of fluency and adequacy. We prepared a list of 100 sentences, randomly selected from translations of the 200k-baseline and extended models (see Table 1). Evaluators marked each translation's fluency and adequacy with scores in the range of 1 to 5. Fluency and adequacy scales are defined in Table 4a and results obtained from this experiment are reported in Table 4b.

As Table 4 shows, the proposed model positively affects both the fluency and adequacy of translations. To discuss this experiment with more details we report exact numbers for the Farsi evaluation which are shown in Figure 3. Each translation is marked with two scores. Clearly, there are 100 fluency and 100

	Fluency	Adequacy
1	incomprehensible	none
2	disfluent	little meaning
3	non-native	much meaning
4	good	most meaning
5	flawless	all meaning

(a) Fluency and adequacy scales.

	Fluency		Adequacy	
	Base	Ext	Base	Ext
En→Fr	2.96	**3.03**	3.05	**3.22**
En→Fa	2.43	**2.63**	3.10	**3.43**

(b) Average fluency and adequacy scores for 100 translations. Base and Ext show the baseline and extended systems.

Table 4: Human evaluation results on 100 French (Fr) and Farsi (Fa) translations.

adequacy scores for each evaluation set (Table 4b reports the average of these 100 scores). Results can be interpreted from different perspectives, some of which we briefly mention below. For the baseline model, the fluency rate of 38% of translations is 3, but this percentage is raised to 45% in the extended model. 27% of the baseline translations are disfluent but in the extended model this is reduced to 19%. For the adequacy feature the condition is even better. Translations which could not properly convey the meaning are changed to translations which are more acceptable for our evaluators. Figure 3 illustrates these changes and shows how our model improves both fluency and adequacy of translations. The number of bad translations is reduced in the extended model and correspondingly, the number of high-quality translations is increased.

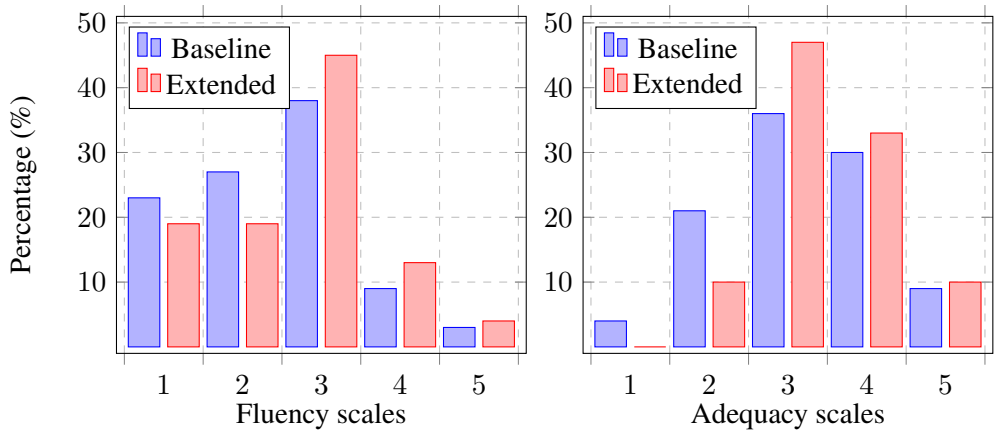


Figure 3: Human evaluation results on En→Fa translation.

5 Conclusion

We proposed a new CNN to train bilingual embeddings and incorporated our embeddings in the PBSMT pipeline. We showed that our embeddings provide useful information. Our model and other similar models are more suitable for medium-size datasets and low-resource languages. We evaluated our model on English, French and Farsi and were able to obtain significant improvements for all of them. Boosting the Farsi engine is a valuable achievement for us, as Farsi is a low-resource and morphologically rich language which makes the translation process hard for any PBSMT engine. We also performed human evaluations, whose results confirmed the impact of our model.

The proposed CNN has a good potential for handling complex structures. For our future work we wish to extend the input tensor with several layers to process richer source-side information. Each word in the proposed architecture could be accompanied with extra information such as morphological and syntactic information.

Acknowledgments

We would like to thank the three anonymous reviewers for their valuable and constructive comments and the Irish Center for High-End Computing (`www.ichec.ie`) for providing computational infrastructures. This research is supported by Science Foundation Ireland at ADAPT: Centre for Digital Content Platform Research (Grant 13/RC/2106).

References

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *The 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1370–1380, Baltimore, USA.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2013. Learning semantic representations for the phrase translation model. *Microsoft Technical Report*.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *Technical Report*.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882, Korea.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180, Prague, Czech Republic.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*, volume 5, pages 79–86, Phuket, Thailand.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *The International Conference on Machine Learning (ICML)*, Beijing, China.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, Sofia, Bulgaria.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 160–167, Sapporo, Japan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, USA.

Peyman Passban, Chris Hokamp, and Qun Liu. 2015a. Bilingual distributed phrase representation for statistical machine translation. In *MT SummitXV*, pages 310–318, Miami, Florida.

Peyman Passban, Andy Way, and Qun Liu. 2015b. Benchmarking smt performance for Farsi using the tep++ corpus. In *The 18th Annual Conference of the European Association for Machine Translation (EAMT)*, pages 82–88, Antalya, Turkey. EAMT.

Peyman Passban, Chris Hokamp, Andy Way, and Qun Liu. 2016. Improving phrase-based SMT using cross-granularity embedding similarity. In *The 19th Annual Conference of the European Association for Machine Translation (EAMT)*, pages 129–140, Riga, Latvia.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Empiricial Methods in Natural Language Processing*, Doha, Qatar.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling*, 5:3.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Andreas Stolcke. 2002. SRILM — An Extensible Language Modeling Toolkit. In *Proceedings of Intl. Conf. Spoken Language Processing*, volume 2, pages 901–904, Denver, Colorado, USA.

Kai Zhao, Hany Hassan, and Michael Auli. 2015. Learning translation models from monolingual continuous representations. In *CThe onference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL HLT 2015)*, Denver, Colorado, USA.

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1393–1398, Seattle, Washington, USA.

Anecdote Recognition and Recommendation

Wei Song[†], **Ruiji Fu**[‡], **Lizhen Liu**[†], **Hanshi Wang**[†], **Ting Liu**[§]
[†]Information Engineering, Capital Normal University, Beijing
[‡]Iflytek Research Beijing, Beijing
[§]Harbin Institute of Technology, Harbin
{wsong, lzliu, hswang}@cnu.edu.cn, rjfu@iflytek.com, tliu@ir.hit.edu.cn

Abstract

We introduce a novel task *Anecdote Recognition and Recommendation*. An anecdote is a story with a point revealing account of an individual person. Recommending proper anecdotes can be used as evidence to support argumentative writing or as a clue for further reading.

We represent an anecdote as a structured tuple — $< person, story, implication >$. Anecdote recognition runs on archived argumentative essays. We extract narratives containing events of a person as the anecdote story. More importantly, we uncover the anecdote implication, which reveals the meaning and topic of an anecdote. Our approach depends on discourse role identification. Discourse roles such as *thesis*, *main ideas* and *support* help us locate stories and their implications in essays. The experiments show that informative and interpretable anecdotes can be recognized. These anecdotes are used for anecdote recommendation. The anecdote recommender can recommend proper anecdotes in response to given topics. The anecdote implications contribute most for bridging user interested topics and relevant anecdotes.

1 Introduction

Building technical tools to assist learning and writing is of great significance and challenging. While a number of tools have been developed for giving feedback on spelling (Bangert-Drowns, 1993), grammar patterns (Yen et al., 2015) and organization (Burstein et al., 2003b), little exists to provide support during planning and composition process. During the process, an automated system, that can effectively collect topic oriented evidence and reading materials, will greatly reduce the cognitive load.

This paper introduces the ***anecdote recognition and recommendation*** task. An anecdote is a story with a point revealing account of an individual person or an incident. We aim to recognize anecdotes from texts and recommend anecdotes according to given topics. The recommended anecdotes can be used as evidence to support argumentative writing.

The argumentative essay is a genre of writing that requires the writer to investigate a topic and establish a position in a concise manner. In addition to create good claims, the quality of argument is greatly affected by the effectiveness of evidence. Finding relevant evidence is not easy, since it heavily depends on long-term accumulation of materials (from reading and observation) and the ability to retrieve and figure out right ones from memory. This process brings in great challenges for both novice and more sophisticated writers. Anecdotal evidence is one of the most commonly used evidence types (Hornikx, 2005). For a given claim, a system that can provide relevant anecdotes would help writers find good evidence and potentially improve the organization and the quality of essays. Recommending anecdotes can be the first step to recommend all types of evidence.

Recognizing anecdotes is related to previous work that extracts story-like elements from text. For example, Chambers and Jurafsky (2008) propose an unsupervised approach to extract narrative event chains from newswire text. A narrative event chain is a set of narrative events that share a common participant. However, extracting stories only is not sufficient. Suppose that we are writing an essay about *the value of life*, how can a system understand which stories are related to this topic? The stories are

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2592–2602, Osaka, Japan, December 11-17 2016.

Field	Content
Person	Steve Jobs
Story	Steve Jobs changed the digital world. He devoted his life to the digital revolution.
Implication	The value of life is to change the world.

Table 1: An extracted anecdote. The story is a narrative consisting of human-centric events and the implication is an automatically extracted text span, from which we can infer the meaning of Jobs's story.

	Topic: The value of life
Rank	Recommended Anecdotes
1	**Person**: Steve Jobs **Story**: Steve Jobs changed the digital world. Steve Jobs devoted his life to the digital revolution. **Implication**: The value of life is to change the world.
3	**Person**: Bruno **Story**: Bruno, who was burned to death, was a martyr of modern science. **Implication**: The value of life depends on its donation rather than its duration.
2	**Person**:Helen Keller **Story**:Helen Keller is known for her efforts in learning and her arduous work for the disabled. **Implication**: The life means fighting against the fate.

Table 2: An example of anecdote recommendation. Anecdotes are ranked according to the given topic.

objective facts, but the writing goals are subjective. The semantic relatedness between them are less direct as discussed in (Rinott et al., 2015). To close this gap, we have to figure out the meanings or intents that these stories want to express.

Considering the above issues, **we define an anecdote as a structured tuple, including *person*, *story* and *implication***. The *story* describes the factual information about the story of specific persons. The *implication* indicates the meanings and significance of the story. We recognize anecdotes from essays and re-use the anecdotes to assist future argumentative writing.

Our approach is based on discourse role identification, which automatically recognizes discourse roles such as the *thesis*, *main ideas*, *support* and *conclusion* in argumentative essays. These discourse roles are closely associated with the anecdote stories and implications.

To recognize anecdote stories, we propose a human-centric approach. The narratives containing events related to a shared person and playing a discourse role as *support* are extracted as an anecdote story. To recognize anecdote implications, we assume that the stance expressed by authors of essays implies the implications of anecdotes. Therefore, we choose the *thesis*, *main ideas* and *conclusion* that the stories support as their implications.

Table 3 presents an example of the extracted anecdotes. Our method can recognize in an essay that one implication of the story that *Steve Jobs changed the digital world* is that *the value of life is to change the world*, because they are identified as the discourse roles *support* and *main idea* respectively and the former supports the latter.

In this way, we recognize anecdotes from archived essays and store them to build an anecdote database. Based on this database, we can recommend anecdotes in response to user queries, which represent their interested topics. Table 2 shows an example of the results of anecdote recommendation. The suggestive anecdotes provide representative persons and brief descriptions of their stories related to the given topic. The recommendations would motivate uses to choose proper ones as evidence.

To summarize, we make the following contributions in this paper:

- We introduce the anecdote recognition and recommendation task. We explicitly define the structure of anecdotes and automatically extract factual anecdote stories and anecdote implications based on discourse role identification. The structured anecdotes are readable, interpretable and searchable. They can be recommended as potential evidence to support argumentative writing.

- Human evaluation demonstrates that accurately extracting anecdotes is indeed feasible. Moreover, the results in anecdote recommendation show that the recommended anecdotes have good relevance and usefulness. Anecdote implications contribute most for bridging anecdotes and user intent.

2 Related Work

2.1 Writing Assistance

There have been many tools providing technical support for assisting and evaluating writing at lexical and discourse levels (Bangert-Drowns, 1993; Burstein et al., 2003a; Yen et al., 2015) or based on collaboration (Noël and Robert, 2004; Nebeling et al., 2016). This paper extends existing work and proposes to recognize and recommend anecdotes for assisting argumentative writing. Our work is related to work on quote recommendation (Tan et al., 2015) and citation recommendation (He et al., 2010). But the focuses and techniques are quite different. To the best of our knowledge, our work is the first to recommend structured factual evidence to support argumentative writing.

2.2 Argumentation Mining

Argumentation mining aims to identify the components and their relationships in argumentation (Stab and Gurevych, 2014; Peldszus and Stede, 2015; Abbas and Sawamura, 2012; Lippi and Torroni, 2015; Feng and Hirst, 2011). Similar work focuses on identifying discourse roles in student essays and scientific abstracts (Burstein et al., 2003b; Guo et al., 2010). Our work focuses on recognizing anecdotes based on discourse role identification in student essays. Moreover, we are also interested in the association between roles, such as factual evidence and the arguments they support.

Our work is close to (Rinott et al., 2015), which aims to recommend evidence according to given claims. The main differences include: (1) In (Rinott et al., 2015), the system runs over dedicated manually labeled data. Instead, our approach automates all aspects of anecdote extraction and recommendation based on information extraction and discourse role identification. (2) Our approach explicitly defines and recognizes the implications of the anecdotal stories. This makes the stories more understandable and closes the semantic relatedness gap between the stories and user interested topics. (3) We focus on a specific application scenario: anecdote recommendation for argumentative writing.

2.3 Narrative Modeling

We extract anecdote stories by extracting human centric events. Story extraction is a kind of narrative modeling. A story is usually viewed as a sequence of events (Chambers and Jurafsky, 2008) based on information extraction (Etzioni et al., 2011). Much work has been done for extracting facts and events from various resources (Banko et al., 2007; Ritter et al., 2012; Bamman and Smith, 2015; Bamman et al., 2014). Similar techniques have been used in educational applications. For example, factual information in argumentative essays has be exploited for essay scoring (Klebanov and Higgins, 2012).

Our work differs from existing work in two folds: (1) We combine event extraction and discourse role identification for extracting anecdote stories. (2) We also uncover the implications of these stories in order to close the gap between objective facts and subjective human intent.

3 Data and Task

3.1 Data

The task we propose is recognizing and recommending anecdotes for assisting argumentative writing. We focus on dealing with argumentative essays, because there exist rich evidence used by the authors to support their claims. We aim to extract anecdotes from archived essays and use them as suggestions to users who are planning to write on similar topics. The users can use the recommended anecdotes as evidence directly, or view them as a clue to find more materials.

In order to recognize anecdotes that are likely to be used for supporting writing, we collect data from an online essay collection, LELE KeTang.[1] This collection contains different types of student essays such as *argumentative essays* and *narrative essays*. The types can be read through the tags of essays. We collected 16618 argumentative essays written by students from senior high school and above in Chinese.

[1] http://www.leleketang.com/zuowen/

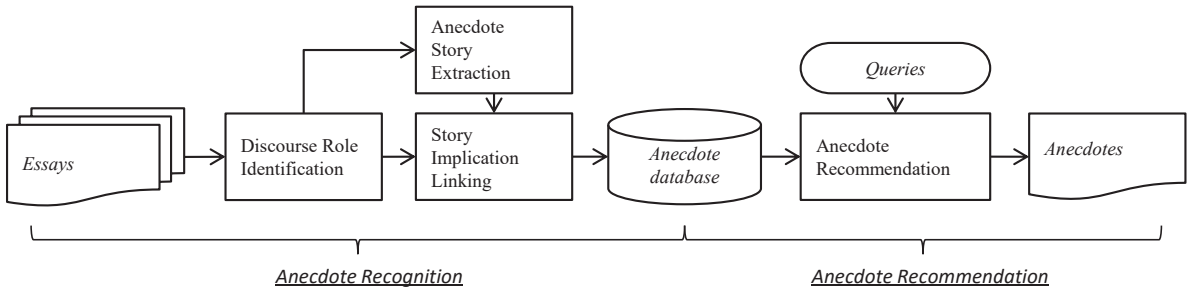


Figure 1: The framework of Anecdote Recognition and its application on Anecdote Recommendation.

3.2 Task Overview

We formally define an anecdote as a structured tuple, $< person, story, implication >$, in order to make it readable, interpretable and searchable.

An anecdote story is a concise summary of the factual events of a certain person. We view anecdote story recognition as a human-centric event extraction task. An anecdote story consists of a set of narratives describing the events of the persons.

The anecdote implication of an anecdote implies the meaning and topic of the anecdote story. We call it *implication* because the meanings or topics of the anecdote are hard to be read directly from the story itself, since the story is usually a factual description. Therefore, we have to infer the implication by means of extra information and strategies. The functions of anecdote implication should include: (1) It interprets the factual information and demonstrates how others take stance on the facts; (2) It closes the gap between objective facts and subjective writing goals to make anecdotes searchable by topics.

The general architecture of our approach is shown in Figure 1. The anecdote recognition module recognizes anecdotes from essays and stores them in a database. Anecdote stories and implications are extracted based on discourse role identification. The anecdote recommendation as an application can recommend anecdotes according to user queries.

Role	**Definition**
Introduction	introduces the background and/or grabs readers' attention
Thesis	states the main claim on the issue for which the author is arguing
Main idea	asserts ideas or aspects that are related to the thesis
Support	provides evidence to explain or support the thesis and the main ideas
Conclusion	concludes the whole essay
Other	doesn't fit into the above elements or makes no meaningful contribution

Table 3: Definitions of discourse roles.

4 Anecdote Recognition

Anecdote recognition is to extract stories and their implications from a given essay. We realize it based on discourse role identification. *Discourse roles* represent the contributions that sentences can make to text organization. Table 3 lists the discourse roles we use, which are inspired by (Burstein et al., 2003b). Our motivation is that the stories and implications relate to different discourse roles in an argumentative essay. Stories are mainly used as evidence to support the *thesis* and *main ideas* proposed by the writer. Therefore, the *thesis* and *main ideas* could be viewed as the implications of the stories in the same essay. We use the *thesis*, the *main idea* and the *conclusion* sentences as implication candidates. The reason we distinguish these roles is because they control different ranges of an essay. The *thesis* and the *conclusion* set up the main tune and conclude the whole essay, while the *main ideas* mainly control the local zones.

According to our motivation, anecdote recognition is divided into three steps: discourse role identification, story extraction and story-implication linking. Before anecdote extraction, an essay text is preprocessed by a pipeline of components including word segmentation, POS tagging, named entity

tagging, dependency parsing and semantic role labeling with HIT-LTP (Che et al., 2010). Also, we use pairs of quotations to locate quotes and view each quote as a whole.

4.1 Discourse Role Identification

We identify discourse roles based on a linear-chain Conditional Random Field (CRF) model (Lafferty et al., 2001) in order to capture the correlations among sequential predictions. The features for each sentence are mainly inspired by the previous work (Burstein et al., 2003b; Stab and Gurevych, 2014):

- **Position features** We use the relative position (beginning, middle, end) of the sentence in its paragraph and its paragraph in the essay, and the number of the sentence in the document as features.
- **Indicator features** We use manually collected cue words/phrases like *in my opinion*, *first of all* and *in conclusion* as indicators. Boolean features are designed for them.
- **Lexical features** We construct boolean features for connectives and modal verbs (such as *should*). We don't use unigrams and bigrams as features because they are sparse on a small dataset.
- **Structural features** We use the number of words, the number of clauses in the sentence and the number of sentences in the same paragraph and the ending punctuation as structural features.
- **Human and quote features** Boolean features are designed respectively to indicate whether the sentence contains human mentions, first person pronouns, third person pronouns and quotes.
- **Thesis word features** Two boolean features are used to indicate whether the sentence contains the words in essay title and the automatic extracted thesis words.

The motivation of thesis word features is that the words that reveal the thesis of an essay would help distinguish *thesis*, *main idea* and *conclusion* sentences from others. We only consider nouns, verbs and adjectives as candidate thesis words. The words in essay titles are used as thesis words. In addition, we attempt to extract thesis words automatically. We observe that the words that distribute globally in an essay tend to indicate the topic of the essay. Therefore, we rank words according to the number of paragraphs a word occurs and view the top ranked words as thesis words.

The model is learned on an annotated dataset that would be introduced in §6.1. The learned model would be used to predict the discourse roles in new essays. These identified discourse roles provide supporting information for anecdote story and implication extraction.

4.2 Story Extraction

We propose a human-centric approach to recognize anecdote stories, since anecdotes usually center around certain persons. Story extraction is conducted in two steps: recognizing human mentions, and extracting narrative events related to human mentions.

Recognizing human mentions A human mention is an observed textual reference to an individual or a group of people. Our approach considers person names, human noun phrases (NPs) and third-person pronouns. The person names are identified according to the POS tagging results provided by HIT-LTP. Human NPs refer to non-specific persons like *soldiers*. We build a dictionary containing human NPs which have a definition as HUMAN and a POS as NOUN in the common-sense knowledge base HowNet (Dong and Dong, 2006). Strings that match entries in this dictionary are marked as human NPs.

Many references to persons are in the form of pronouns. Since we have marked person names and human NPs, the pronouns should choose antecedents from them. We implement a rule-based approach considering the gender and number compatibility, syntactic roles and distance constraints. Unresolved pronouns are discarded.

Narrative event extraction Similar to (Chambers and Jurafsky, 2008), we assume that a story consists of a chain of events involving the same person. For each human mention, we extract all sentences containing it or its references. Then we conduct semantic role labeling on these sentences and extract the <agent, predicate, recipient > tuples as events. If the agent field contains the human mention, the tuple

would be retained. Since the stories have to be shown, we keep the sentences containing the retained tuples as the anecdote story candidates.

Notice that sentences that contain human mentions might be not a part of a story. Consider the following two sentences:

sentence 1 *Steve Jobs devoted his life to the digital world and designed a series of revolutionary products.*

sentence 2 *What we learn from Steve Jobs' story is that we should try our best to pursue our dreams.*

We can see that sentence 2 actually expresses an opinion rather than facts. To resolve this, we only retain the candidates with the discourse role *support* as the anecdote story.

4.3 Story Implication Linking

By now, we have extracted a set of anecdote stories. As mentioned earlier, we consider the sentences with discourse roles *thesis*, *main idea* and *conclusion* sentences as the potential implications of the stories. We have to link the stories and implications together.

We assign *thesis* and *conclusion* sentences to every anecdote story in the same essay as parts of their implications, since they cover the whole essay. The *main ideas* argument from multiple aspects. Therefore, we should link main ideas to nearby stories.

To deal with it, we train a story-idea classifier to determine whether a story and a main idea should be linked. For simplicity, we merge adjacent main idea sentences within the same paragraph as a main idea block. We derive features for every pair of a story and a main idea block. The features include (1) Paragraph distance, which is the difference between their paragraph numbers; (2) The number of shared words; (3) Connectives, since some of which like *therefore* are key indicators of the linking relation; (4) Whether they share human mentions or their references.

We train a logistic regression classifier on manually labeled story-idea pairs. For the story in a positive pair, we choose the nearest but unlinked main idea block (if there is one) to form a negative pair. During prediction, for each extracted story, we apply the classifier to determine whether it should be linked to the closest main idea blocks before and after it within 2 paragraphs. If the prediction is positive, the corresponding main idea blocks are used as part of the anecdote implication.

After story-implication linking, we get a set of <*person, story, implication*> tuples as anecdotes. By accumulating anecdotes from all available essays, we construct an anecdote database.

5 Anecdote Recommendation

Anecdote recommendation is an application of anecdote recognition. Once writers decide the main topic to address, they would attempt to find evidence to support their claims. Anecdote recommendation aims to recommend anecdotes as suggestions according to user interested topics.

The task can be described as follows: with the anecdote database C available, given a query q, the recommender returns a subset of anecdotes E, which are ranked top by a ranking model. The ranking depends on the relatedness measurement between the anecdotes and the queries.

Relatedness measurement We are interested to study which information—the anecdote story or the anecdote implication—contributes more for measuring the true relevance. Therefore, we focus on comparing the following strategies for ranking anecdotes:

- **Query-Story (QS)**. Rank based on the relatedness between the anecdote story and the query.
- **Query-Implication (QI)**. Rank based on the relatedness between the anecdote implication and the query.

To compute the semantic relatedness between the query and a text, we compare the BM25 that is a term-matching based ranking model (Robertson and Zaragoza, 2009) and a word embedding based approach (WE) — We use the average word embedding to represent a text. We are interested to see whether word embedding based approach can alleviate the semantic gap problem.

Learning to Rank Based on the above relatedness measurements, we get four ranking functions: QS-BM25, QI-BM25 and QS-WE, QI-WE. We adopt the learning to rank framework to integrate them. We

use linear ranking functions and transform the ranking problem into a two-class classification problem (Herbrich et al., 1999; Joachims, 2002). For each query, given two comparable instances whose feature vectors and labels are (x_i, y_i) and (x_j, y_j), we transform it into $(x', y') = (x_i - x_j, sign(y_i - y_j))$. After transformation, we use the SVM classifier with linear kernel to learn a ranking function.

The learned weights of some ranking functions might be negative. Since all signals should contribute positively to the final ranking, a constant is added to each weight to make sure all the weights are positive.

Training Data We build the training data based on *pseudo queries*. For an anecdote, its pseudo query is the title of the essay, from which the anecdote is extracted. It is reasonable to assume that essay titles can be used to simulate writers' search intent. We consider binary relevance labels: *relevant* and *irrelevant*. Given an essay title as a pseudo query, the anecdotes that are extracted from essays with the same title are viewed as relevant, while randomly sampled anecdotes are labeled as irrelevant to the pseudo query. In this way, we can build a training data without any manual labor.

6 Evaluation

6.1 Evaluating Discourse Role Identification

Data Two annotators manually labeled 200 student essays with discourse roles at sentence level according to a set of initial guidelines. The percentage agreement between annotators is 0.84. They discussed to reach new standards and then reviewed all annotations together. The distribution of the six discourse roles is unbalanced. The *support* discourse role accounts for 52% of all sentences, while the *introduction, thesis, main idea* and *conclusion* sentences account for 9%, 5%, 18% and 9% respectively.

Evaluation settings We conducted experiments on the corpus using 5-fold cross-validation. The precision (P), recall (R) and F_1 score are reported.

Discourse role	P	R	F_1
Introduction	73.1	74.6	73.8
Thesis	66.7	61.1	63.7
Main idea	69.0	60.9	64.6
Support	83.2	86.4	84.8
Conclusion	81.8	84.5	83.2

Table 4: Performance on identifying five discourse roles.

Results The experimental results on the corpus are shown in Table 4. We can see that our method can recognize *support* sentences well. In contrast, the *thesis* and *main ideas* are identified with moderate performance. By analyzing the errors, we found that many errors are related to the boundaries of *thesis* sentences. Some errors come from distinguishing *introduction* and *thesis*. In some cases, *introduction* sentences also involve thesis related words, although they don't explicitly make a claim, while some essays make claims at the beginning without placing any introduction. Some other errors come from incorrectly distinguishing *thesis* and *main ideas*. When there are multiple *thesis* sentences in an essay, the ones that appear later might be identified as *main idea* incorrectly. In addition, due to the imbalanced data, more sentences tend to be classified into the majority class.

6.2 The Anecdote Database

Our anecdote recognizer ran on the dataset introduced in §3 to construct an anecdote database. The database contains 26060 anecdotes, involving 9762 persons.

Data and evaluation settings We randomly sampled 200 anecdotes and asked two raters, who are students from the department of Literature, to evaluate the quality of the anecdotes. The anecdote story and implication were shown to the raters. The evaluation is based on the criteria below by judging story and implication jointly. Here is a description of the criteria:

Good	The story is understandable and complete. The implication can interpret the story.
Poor Story	The story is hard to be understood. The implication is unable to be judged.
Poor Implication	The story is understandable and complete. The implication can't interpret the story.

	Good (%)	Poor Story(%)	Poor Implication(%)
Rater1	61	22	17
Rater2	62	25	13

Table 5: The manual evaluation results by two raters.

Results Table 9 shows the evaluation results by two raters. We can see that more than 60% of the anecdotes in average are judged as *good*. This means that the anecdote stories and the corresponding implications can be effectively extracted and paired together. About 23% of the extracted anecdotes are judged as *poor story*. Most of these errors are caused by incorrectly recognized user names and the failure of pronoun resolution. 15% of the extracted anecdotes are judged as *poor implication*.

We manually labeled the story-idea pairs in 50 essays. The story-idea classifier can achieve an accuracy of 87% by cross-validation. We observe that the stories and their implications have strong locality correspondence that they tend to be within the same paragraphs.

The results show that for most recognized stories, our method can find proper implication for them. This proves that discourse role identification is a promising way for anecdote implication recognition.

6.3 Evaluating Anecdote Recommendation

6.3.1 Automated Evaluation

Data We conducted experiments on the extracted anecdote database. We stored the title of the essay from which each anecdote is extracted. There are 8141 distinct titles in all. These titles are used as queries to simulate user interested topics. We randomly sampled 1000 queries respectively to construct the training set, development set and the test set. We collected relevant instances for each query as described in §5. Each query has 3.4 relevant anecdotes in average. For each training query, we randomly sampled the same number of irrelevant instances in order to maintain a balanced training data. For each query in development set and test set, we viewed all anecdotes from the anecdotes database as candidates.

Experimental settings The parameters of SVM were tuned on the development dataset. We trained a Word2Vec model using the skip-gram algorithm with hierarchical softmax (Mikolov et al., 2013) on a dataset from Baidu Baike. The vector size is 50. The vocabulary size is 1, 825, 833. We adopt the commonly used mean average precision (MAP) and nDCG (Järvelin and Kekäläinen, 2002) as metrics.

Model	MAP	nDCG@1	nDCG@5
QS-BM25	0.357	0.406	0.387
QI-BM25	0.738	0.624	0.766
QS-WE	0.362	0.386	0.384
QI-WE	0.716	0.59	0.747
LTR	**0.744**	**0.64**	**0.786**

Table 6: The evaluation results of anecdote recommendation on pseudo queries.

Results Table 6 presents the performance of various strategies. We can see that QS-BM25 and QS-WE perform worst among all strategies. This proves that vocabulary gap exists between queries and factual story descriptions of anecdotes. In contrast, great improvements are obtained when measuring relatedness between anecdotes and queries with anecdote implications. Both QI-BM25 and QI-WE achieve much better results compared with QS-BM25 and QS-WE. This indicates that the anecdote implications play important roles in bridging user intent and anecdotes. The word embedding based approaches don't have obvious superior performance compared with conventional BM25 approaches. But by combining all signals together, LTR — that represents our learning to rank model, gains further improvement compared with any single model. This means that learning the model automatically based on pseudo query strategy is feasible.

6.3.2 Human Evaluation

Pseudo queries based evaluation indicates that proposed approach can achieve high precision on top results. To gain deeper understanding of the usefulness and interpretability, we conducted evaluation on

Topic (translation, year)
尝试(Try to do, 1994), 战胜脆弱(Overcome the weakness, 1998), 答案是丰富多彩的(The answer is rich, 2000), 心灵的选择(The Choice of heart, 2002), 转折(The turning point, 2003), 包容(Be tolerant, 2004), 我有一双隐形的翅膀(I have a pair of invisible wings, 2009), 诚信(Be honest, 2011), 深入灵魂的热爱(Deep passion, 2015).

Table 7: Essay topics for user study.

score	Description of the meanings of scores
4	The anecdote is representative and clear. It can be used as evidence directly for the given topic.
3	The anecdote is representative but not complete. It provides clues for further exploring the details.
2	The anecdote is relevant, but not representative.
1	The anecdote is irrelevant to the given topic.

Table 8: Descriptions of the meaning of each score.

manually labeled data.

Topics We chose nine argumentative topics from the recent years' college entrance examinations in Bejing, China. The topics are shown in Table 7. We only chose the topics that the essay titles are fixed so that students needn't derive their own titles according to the prompt.

Annotation For each given topics, we used the QS-BM25, QI-BM25 and LTR systems to retrieve the top 20 anecdotes respectively. We merged their results together (removing the duplicate ones) and asked 5 raters to evaluate all the results respectively using a numerical score from 1 to 4. The descriptions of the meaning of each score is shown in Table 8.

score	1	2	3	4
QS-BM25	54%	16 %	9 %	21%
QI-BM25	28%	20%	11 %	41%
LTR	26%	13%	16%	45%

Table 9: Distributions of raters' scores on manually labeled data.

6.3.3 Results

Table 9 shows the average percentage of anecdotes belonging to each score retrieved by each system. More than 70% of anecdotes extracted by LTR and QI-BM25 are relevant to the given topics. In contrast, QS-BM25 provides many more irrelevant anecdotes. This indicates that anecdote implications are better to represent the topics of the anecdotes.

Based on LTR, about 45% anecdotes are viewed as representative enough and can be used as evidence directly. A large ratio of these anecdotes are about famous people. About 16% anecdotes are considered as representative, but users have to further explore, e.g., by search, to gain more information. Parts of the low quality anecdotes are due to incorrectly recognized person names and the failure of pronoun resolution. In addition, difficult queries, like *I have a pair of invisible wings*, lead to poor performance. Such queries are kind of rhetorical device so that students have to derive the thesis themselves. Since our method is mainly based on keywords, the relevance of retrieved anecdotes is not good except that some students had used similar expressions to express their claims.

6.4 Discussions

The results show that the proposed anecdote recommender can provide relevant and representative anecdotes to user interested topics. Two main factors contribute for this. First, we use argumentative essays as data resource so that the anecdotes extracted actually have been chosen carefully by the essay authors. As a result, most of them are representative. Second, we give structures to anecdotes that usually are not considered as a structure. The anecdote implication contributes great for finding relevant anecdotes. The idea of structured retrieval can be extended to other problems as well.

On the other hand, several limitations exist. First, anecdotes extracted from student essays may be limited in narrow scopes. With the development of argumentation mining, we can extend our work to

other domains. Second, we mainly consider the relevance of anecdotes in evaluation but don't care about whether the anecdotes support or attack the given topic. Third, information beyond topical relevance such as the popularity of an anecdote or a person can be exploited. Such information is not incorporated in the current *pseudo query* based training procedure. Further relevance feedback can be conducted. Finally, organizing and diversifying the recommended anecdotes are also interesting but ignored in this study. We leave these as future work.

7 Conclusion

This paper proposes anecdote recognition and recommendation task to support argumentative writing. We not only extract concise and informative anecdote stories, but also uncover the implications of facts. The enriched structured representation makes the extracted anecdotes interpretable and searchable. Our approach is based on discourse role identification in essays. The experimental results demonstrate the effectiveness of our approach. More than 60% of our extracted anecdotes have both well described stories and interpretable implications. The extracted anecdotes can be applied for anecdote recommendation. With the help of anecdote implications, the recommender is able to suggest relevant anecdotes in response to simulated user intent. This proves that anecdote implications are useful for closing the semantic gap between factual evidence and user interested topics.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (No.61402304,No.61303105), the Beijing Municipal Natural Science Foundation (No.4154065) and the Humanity & Social Science General Project of Ministry of Education (No.14YJAZH046).

References

Safia Abbas and Hajime Sawamura. 2012. Argument mining based on a structured database and its usage in an intelligent tutoring environment. *Knowledge and information systems*, 30(1):213–246.

David Bamman and Noah A Smith. 2015. Open extraction of fine-grained political statements. pages 76–85.

David Bamman, Ted Underwood, and Noah A Smith. 2014. A bayesian mixed effects model of literary character. In *ACL (1)*, pages 370–379.

Robert L Bangert-Drowns. 1993. The word processor as an instructional tool: A meta-analysis of word processing in writing instruction. *Review of Educational research*, 63(1):69–93.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction for the web. In *IJCAI*, volume 7, pages 2670–2676.

Jill Burstein, Martin Chodorow, and Claudia Leacock. 2003a. Criterionsm online essay evaluation: An application for automated evaluation of student essays. In *IAAI*, pages 3–10.

Jill Burstein, Daniel Marcu, and Kevin Knight. 2003b. Finding the write stuff: Automatic identification of discourse structure in student essays. *Intelligent Systems, IEEE*, 18(1):32–39.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL*, volume 94305, pages 789–797. Citeseer.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 13–16. Association for Computational Linguistics.

Zhendong Dong and Qiang Dong. 2006. *HowNet and the Computation of Meaning*. World Scientific.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. 2011. Open information extraction: The second generation. In *IJCAI*, volume 11, pages 3–10.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 987–996. Association for Computational Linguistics.

Yufan Guo, Anna Korhonen, Maria Liakata, Ilona Silins Karolinska, Lin Sun, and Ulla Stenius. 2010. Identifying the information structure of scientific abstracts: an investigation of three different schemes. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, pages 99–107. Association for Computational Linguistics.

Qi He, Jian Pei, Daniel Kifer, Prasenjit Mitra, and Lee Giles. 2010. Context-aware citation recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 421–430. ACM.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 1999. Support vector learning for ordinal regression. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 1, pages 97–102. IET.

Jos Hornikx. 2005. A review of experimental research on the relative persuasiveness of anecdotal, statistical, causal, and expert evidence. *Studies in Communication Sciences*, 5(1):205–216.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM.

Beata Beigman Klebanov and Derrick Higgins. 2012. Measuring the use of factual information in test-taker essays. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 63–72. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289. Morgan Kaufmann.

Marco Lippi and Paolo Torroni. 2015. Argument mining: A machine learning perspective. In *Theory and Applications of Formal Argumentation*, pages 163–176. Springer.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Michael Nebeling, Alexandra To, Anhong Guo, Adrian A de Freitas, Jaime Teevan, Steven P Dow, and Jeffrey P Bigham. 2016. Wearwrite: Crowd-assisted writing from smartwatches. In *Proceedings of CHI*.

Sylvie Noël and Jean-Marc Robert. 2004. Empirical study on collaborative writing: What do co-authors do, use, and like? *Computer Supported Cooperative Work (CSCW)*, 13(1):63–89.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 938–948.

Ruty Rinott, Lena Dankin, Carlos Alzate, Mitesh M Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence–an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in NLP (EMNLP), Lisbon, Portugal*, pages 17–21.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM.

Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *EMNLP 2014*, pages 46–56.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2015. Learning to recommend quotes for writing. In *AAAI*, pages 2453–2459.

Tzu-Hsi Yen, Jian-Cheng Wu, Joanne Boisson, Jim Chang, and Jason Chang. 2015. Writeahead: Mining grammar patterns in corpora for assisted writing. *ACL-IJCNLP 2015*, page 139.

Training Data Enrichment for Infrequent Discourse Relations

Kailang Jiang, Giuseppe Carenini, Raymond T. Ng
Department of Computer Science
University of British Columbia
Vancouver, BC, Canada, V6T 1Z4
{jiangkl, carenini, rng}@cs.ubc.ca

Abstract

Discourse parsing is a popular technique widely used in text understanding, sentiment analysis and other NLP tasks. However, for most discourse parsers, the performance varies significantly across different discourse relations. In this paper, we first validate the underfitting hypothesis, i.e., the less frequent a relation is in the training data, the poorer the performance on that relation. We then explore how to increase the number of positive training instances, without resorting to manually creating additional labeled data. We propose a training data enrichment framework that relies on co-training of two different discourse parsers on unlabeled documents. Importantly, we show that co-training alone is not sufficient. The framework requires a filtering step to ensure that only "good quality" unlabeled documents can be used for enrichment and re-training. We propose and evaluate two ways to perform the filtering. The first is to use an agreement score between the two parsers. The second is to use only the confidence score of the faster parser. Our empirical results show that agreement score can help to boost the performance on infrequent relations, and that the confidence score is a viable approximation of the agreement score for infrequent relations.

1 Introduction

Discourse parsing is widely used in text understanding (Allen et al., 2014), sentiment analysis (Bhatia et al., 2015) and other NLP tasks (Guzmán et al., 2014) (Gerani et al., 2016). A multi-sentential discourse parser takes a document as input, and returns its discourse structure that shows how clauses and sentences are related in the document, via the use of various discourse relations. For instance, the benchmark RST-DT dataset (Carlson et al., 2001) uses 18 discourse relations. Studies in the past decade on discourse parsing, such as (Ji and Eisenstein, 2014), (Feng and Hirst, 2014), and (Joty et al., 2015), greatly improved the performance of discourse parsing in general. However, it has been observed that the performance across the discourse relations varies significantly (Joty et al., 2015), and that poor performance may be linked to underfitting, i.e., a lack of training data (Feng and Hirst, 2014).

In this paper, we investigate the underfitting hypothesis and study how to improve the situation. Different discourse relations are usually unevenly distributed in a dataset, and some of them occur much less frequently than other relations. We call the former the *infrequent* relations. For example, in the very popular corpus — Rhetorical Structure Theory Discourse Treebank (RST-DT) (Carlson et al., 2001) which contains 385 documents, the frequency of "Elaboration" is 31.04%, while the frequency of "Summary" is only 0.88%. In another benchmark corpus the Penn Discourse Treebank (PDTB) 2.0 (Prasad et al., 2008), which contains about 2400 documents with discourse relations labeled for each pair of adjacent sentences, the relation "Conjunction" occurs 8759 times through the entire corpus, while the relations "Exception" and "Pragmatic concession" only appear 17 and 12 times respectively (Hernault et al., 2010). Given that the performance of most discourse parsers depends on the availability of training data, the key question here is whether underfitting affects the infrequent relations more than the frequent

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2603–2614, Osaka, Japan, December 11-17 2016.

ones. In Section 4, we will explicitly show that parsing performance of relations is correlated with the frequency of the relations.

Clearly, every discourse relation, infrequent or not, would benefit from the availability of more high-quality training data. However, creating such high-quality labeled data takes much time and effort to manually annotate documents with their discourse structures and relations. The question here is whether the infrequent relations are worthy of the extra effort required. It turns out that many infrequent relations actually play important roles in various NLP tasks. For example, the "Comparison" relation from RST-DT is known to indicate disagreement in a conversation (Horn, 1989) (Allen et al., 2014). Moreover, the "Instantiation" relation from PDTB is regarded as an important feature for sentence specificity prediction (Li and Nenkova, 2015).

The main objective of this paper is to explore how to mitigate the underfitting problem for infrequent relations — without manually creating labeled data for those relations. In particular, we aim to exploit the availability of a much larger amount of unlabeled data. The first step of our approach is to apply existing discourse parsers to the unlabeled data to generate more instances of infrequent relations, which are then used to re-train the existing parsers. Such co-training approaches have proved to be effective in solving similar problems in natural language processing (Li and Nenkova, 2015) and information retrieval (Blum and Mitchell, 1998).

There is, however, a fatal flaw relying on co-training alone. If existing discourse parsers are poor in determining infrequent relations, the extra (re-)training instances of infrequent relations created from unlabeled data may not be of high quality. Indeed, adding poor quality re-training instances would exacerbate the underfitting problem of infrequent relations. The second step of our approach is to apply a *filtering* step to the instances created from unlabeled data. The intention behind the filtering step is to *enrich* the re-training - that is, to select only the "high quality" instances to be used for re-training.

The workflow of our enrichment approach is shown in Figure 1, when it is applied to two discourse parsers, P1 and P2. P1 and P2 are initially trained on labeled data and then are applied to unlabeled data to generate new high-quality training examples for further re-training.

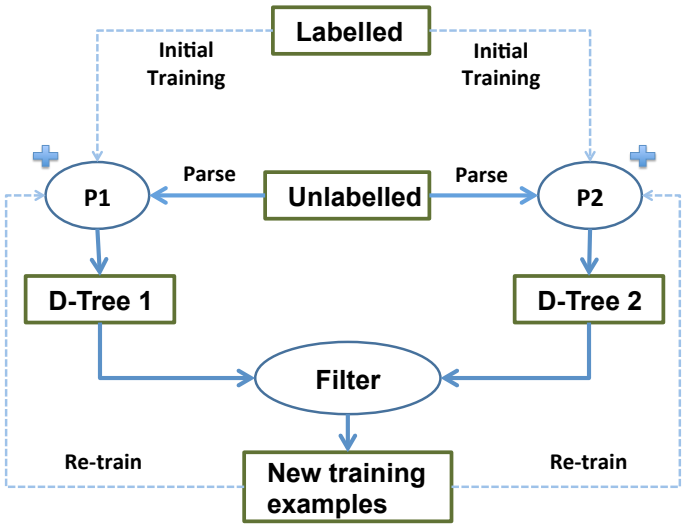


Figure 1: Workflow of our enrichment approach

The specific contributions of our paper are as follow:

- We explore one form of enrichment based on the notion of *agreement score* between two discourse parsers. Inspired by the theory on the success of ensembling for general classification (Dietterich, 2000), we choose two very different discourse parsers, namely the CKY-like CODRA parser by (Joty et al., 2015) and the Shift-Reduce (SR) parser by (Ji and Eisenstein, 2014). While Section 3 will give more details on why these two parsers are chosen, the key is that the parsers are based on

very different algorithms and feature sets for discourse parsing. Our agreement score is based on the F-score measure for comparing discourse trees as proposed in (Marcu, 2000). Only the discourse relation instances in discourse trees with high-enough agreement scores pass through the filter for re-training purposes. Section 4 will show that such enrichment with agreement score improves the performance of infrequent relations.

- We explore another form of enrichment based on just the confidence score of the SR-parser. The rationale is that while the CODRA parser is generally more accurate than the SR-parser, the SR-parser is two orders of magnitude faster. If a high-enough threshold on the confidence score of the SR-parser is used for enrichment, Section 4 will investigate whether the confidence score is a good approximation of the agreement score. If this approach is successful, an even larger number of unlabeled documents can be parsed rapidly to be used for re-training.

2 Related work

Recent discourse parsers have improved the overall performance of discourse parsing in different ways. (Joty et al., 2013) (Joty et al., 2015) proposed a two-stage document-level discourse parser CODRA, which builds a discourse tree by applying an optimal parsing algorithm to probabilities inferred from two Conditional Random Fields: one for intrasentential parsing and the other for multisentential parsing. This approach achieves good performance in discourse relation labeling. Based on their idea, (Feng and Hirst, 2014) developed a similar but much faster model that adopts a greedy bottom-up approach, with two linear-chain CRFs applied in cascade as local classifiers. On the other hand, (Ji and Eisenstein, 2014) proposed a Shift-Reduce (SR) parser that combines the machinery of large-margin transition-based structured prediction with representation learning. This method also reports a good overall performance with linear running time. However, all these state-of-the-art discourse parsers still perform badly on infrequent relations due to insufficient training examples.

The problem of lacking training examples also impacts other aspects of discourse parsing, for example parsing implicit relations. A key distinction in discourse parsing is between explicit and implicit relations. The former are signaled by a cue phrase like "because", while the latter are not and consequentially are more difficult to identify. Several studies have been conducted to tackle the problem of classifying implicit relations which do not have many explicit features and examples. (Zhou et al., 2010) presents a method to predict the missing connective based on a language model trained on an unlabeled corpus. The predicted connective is then used as a feature to classify the implicit relation. (McKeown and Biran, 2013) tackles the feature sparsity problem by aggregating implicit relations into larger groups. And recently (Lan et al., 2013) combines different data through multi-task learning. The method performs implicit and explicit relation classification in the PDTB framework as two tasks and relies on multi-task learning to obtain higher performance.

(Liu et al., 2016) proposes a multi-task neural networks that combines RST-DT, PDTB and unlabeled data together through multi-task learning process, and gets performance improvements on relatively infrequent relations, though they only apply their scheme on the four coarse top-level relations. Their scheme is based on retrieving more training instances from unlabeled data through cue phrases. This approach of using explicit examples to predict implicit examples has been shown to produce mixed results (Sporleder and Lascarides, 2008). Moreover, (Joty et al., 2015) has shown that there are many more features beyond cue phrases that are useful for discourse parsing. (Hernault et al., 2010) proposes a feature vector extension approach to improve classification of infrequent discourse relations. The approach is based on word co-occurrence. Partly because a simple discourse parser was used, their approach is shown to produce only minimal improvements in performance.

Unlike (Liu et al., 2016) and (Hernault et al., 2010), we aim to exploit more advanced parsers with higher performance, and also keep the finer-granularity of the relations, especially focusing on the infrequent relations. We employ the idea of *co-training*, which is first introduced by (Blum and Mitchell, 1998) with its application in helping the search engine better classify "academic course home page". Similar co-training efforts have been found effective in many NLP problems when only a small amount of labeled data is available. For example, (Wan, 2009) proposes a co-training approach for cross-lingual

sentiment classification, while (Li and Nenkova, 2015) applies co-training on predicting sentence specificity.

3 Our Enrichment Approach

The workflow of our enrichment approach is shown in Figure 1. First we use the labeled data to provide initial training of the two parsers. Then each parser is used to produce a discourse tree for each unlabeled document. After that, we apply a filtering step to select those "high quality" discourse trees, which are added to the original labeled data to form the "enriched training data" to re-train the two parsers.

In our approach, the first parser we pick is the CODRA parser (Joty et al., 2015), which applies a CKY parsing algorithm to probabilities inferred from two Conditional Random Fields for both intra-sentential and multi-sentential parsing. We pick the CODRA parser because of its optimal CKY parsing algorithm and its accuracy. The second parser we pick is the SR-parser (Ji and Eisenstein, 2014), which transforms the surface features into a latent space that facilitates RST discourse parsing. The main advantage of the SR-parser is that it can train and parse documents in almost linear time (regarding the document length), while the CODRA parser needs cubic time. Our choice of the two parsers is partly based on the fact that they rely on very different algorithms and feature sets, which is desired by the co-training algorithm. Although another discourse parser (Feng and Hirst, 2014) also delivers state-of-the-art performance, its approach and features are very similar to CODRA's, so we only wanted to select one of them. And due to the fact that Feng's parser is not publicly available and our existing experience on CODRA, we picked CODRA in our approach. Another reason of our choice on the SR-parser is that discourse parsing of documents in general can be slow in both training and parsing. Thus, the SR-parser is attractive in allowing us to explore the tradeoffs between accuracy and efficiency.

Co-training alone, however, is not sufficient. Since both the CODRA parser and the SR-parser performs poorly on infrequent relations, the extra (re-)training instances of infrequent relations created from unlabeled data may not be of high quality. The key idea is to enrich the re-training by selecting only the "high quality" instances. In this paper we investigate two forms of enrichment, based on the agreement score between the two parsers, and the confidence score given by the SR-parser.

To produce the agreement score between the two parsers, we use both parsers to parse every unlabeled document. Then we treat the parse tree produced by the CODRA parser as the ground truth, because in general the CODRA parser is more accurate than the SR-parser. After that, we treat the parse tree produced by the SR-parser as testing, and use the F-score for comparing discourse trees proposed in (Marcu, 2000) as the agreement score. Finally, if the agreement score passes a preset threshold, the unlabeled document is regarded as reliable and the discourse tree is added to enrich re-training.

The second form of enrichment examined in this paper is based on using the confidence score of the SR-parser as an approximation of the agreement score between the two parsers. The SR-parser generates a discourse tree by performing a set of actions. More specifically, each action creates a node in the tree by combining two text spans and by selecting a discourse relation for the pair. Since each action is chosen with a certain confidence score (which technically is the distance between the chosen action and the hyperplane, provided by the underlying Linear SVC algorithm), we use the average confidence of the actions performed to create the tree as the confidence score of the entire tree. If this approach is successful, an even larger number of unlabeled documents can be parsed rapidly for re-training.

Furthermore, we can also control the filtering threshold score at a finer granularity. That is, if the parser allows a user to feed a partial discourse tree or even just some nodes for training, we can set the filtering threshold on each node instead of each document. In this case, if a discourse tree has a high agreement/confidence score, but some nodes on the tree only have a low score, instead of adding the entire discourse tree to the new training set, we can remove those nodes with low score and add only those nodes with high scores to the new training set. This way, we can filter at a finer granularity. Moreover, based on this node-level filtering framework, we can actually set different threshold for different types of nodes, for example, we can set a lower threshold for infrequent relations, and a higher one for frequent relations.

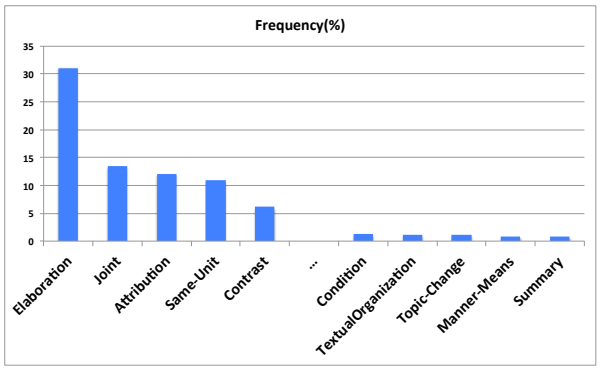


Figure 2: Distribution of the most frequent and the least frequent 5 relations in RST-DT

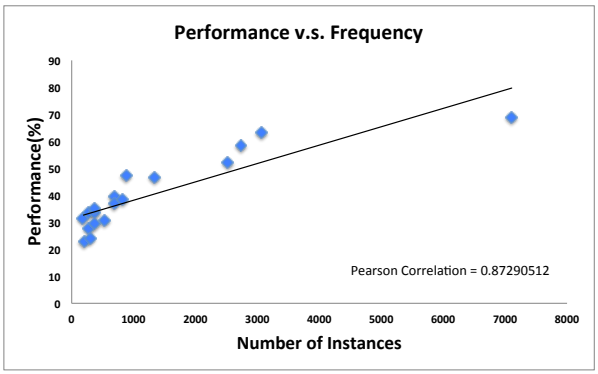


Figure 3: Performance versus Frequency for each relation

4 Empirical Evaluation

4.1 Datasets

In this paper, we use the RST-DT dataset as the gold standard labeled data. It consists of 385 documents selected from Penn Treebank (Marcus et al., 1993), which are all originally articles from the Wall Street Journal. Those 385 documents were divided by the author into two groups: the training set consisting of 347 documents, and the test set 38 documents. For results reported in this paper, we used those 347 documents as the initial training set. The remaining 38 documents made up the test set used to evaluate the performance of the parser re-trained using the enriched dataset.

For the unlabeled documents, we used 2000 Wall Street Journal articles from the Penn Treebank dataset (Marcus et al., 1993). In other words, the gold standard dataset and unlabeled dataset are from the same source; but there is no document belonging to both.

In discourse parsing, there are various performance measurements, such as on the structure (i.e., hierarchical spans) and the labels (i.e., nuclearity and relation classification). The results reported here focuses on relation classification. To evaluate the parsing performance based on the gold standard, we use the standard F-score measure, which is the harmonic mean of precision and recall (Abney et al., 1991). More specifically, we use the F-score measure for comparing discourse trees, as proposed in (Marcu, 2000).

4.2 The Underfitting Hypothesis: Performance vs Frequency

As for the discourse relations, we examine all the 18 coarse-grained relations defined in (Carlson et al., 2001). Figure 2 shows the most frequent and the least frequent five relations in all the 385 documents in the RST-DT dataset. We can see that the most frequent relations can be two order of magnitude higher in frequencies than those of the infrequent ones. For example, the "Elaboration" relation makes up over 31% of all the nodes in the entire dataset, while the "Topic Change" relation accounts for less than 0.5%.

Given the large disparity in relation frequencies, we next examine whether infrequent relations suffer from worse performance than the frequent relations, i.e., the underfitting hypothesis of a lack in training data of the infrequent relations. Here we used the 347 documents to train the SR-parser, and then tested the parser on the 38 documents. Figure 3 shows the performance of each relation (i.e., F-score) versus its frequency. We can see that for each relation, its performance has high correlation with its frequency. Indeed, the Pearson correlation coefficient is 0.87, validating the underfitting hypothesis. This suggests that it would be a reasonable approach to boost the performance of infrequent relations by enriching their training instances.

4.3 Effect of Enrichment on Infrequent Relations

The first form of enrichment examined below is based on the agreement score between the two parsers, as discussed in the previous section. Table 1 shows the improvements on the F-scores from the SR-parser

of the top-8 infrequent relations, based on a threshold of 0.5 in the filtering step. The different columns of the table show an increasing number of unlabeled documents used in enrichment, from 500 documents to 2000 documents. Figure 4 shows the relative F-score improvements across all the 18 relations, ranked from left to right in ascending order of frequency. As a specific example, the F-score of "Topic Change" improves 5.88% with 500 documents, and 13.15% with 2,000 documents.

Relation	500	1000	1500	2000
Summary	2.13	2.80	3.91	5.16
Manner-Means	16.62	21.13	21.61	22.08
Topic-Change	5.88	7.21	12.88	13.15
TextualOrganization	1.42	3.31	7.49	8.14
Condition	3.91	8.69	12.44	18.55
Comparison	3.19	6.06	6.95	10.42
Evaluation	2.83	4.76	8.09	10.98
Topic-Comment	2.69	4.55	6.73	9.48

Table 1: Relative F-scores improvements (%) on the top-8 infrequent relations

As shown in the table and the figure, there is a positive effect on performance by enrichment based on the agreement score. The larger the number of unlabeled documents used, the higher is the gain in performance for the top-8 infrequent relations. The exact magnitude of the gain varies.

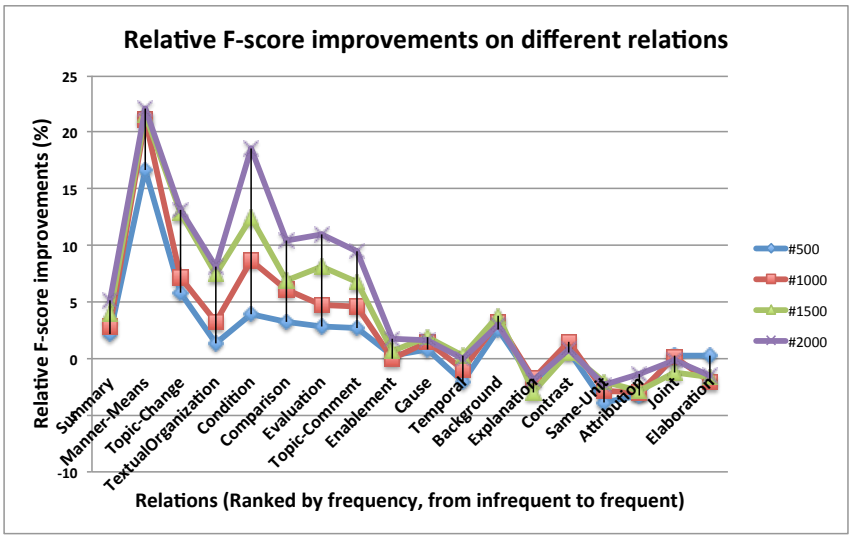


Figure 4: Relative F-score improvements on different relations

So far we have described data enrichment in terms of the number of unlabeled documents. The more detailed analysis is to examine the actual number of training instances created from the unlabeled documents for each relation. Figure 5 shows the actual number of training instances added for each relation, represented as a percentage relative to the frequency of the instances in the original training dataset. For example, for the "Condition" relation, there is a 35% increase in the actual number of instances with 500 documents, and this figure jumps to over 150% with 2,000 documents. With these additional training instances, the gain in F-score for the "Condition" relation is 18.55% from Table 1. For the "Topic Change" relation, it is a pleasant surprise that there is a relative F-score improvement of 13.15% based on about 50% more training instances.

The reader may wonder with 2000 more unlabeled documents, why there is only a modest increase in training instances for some of the infrequent relations. This increase of course depends on the filtering threshold. One temptation based on Table 1 is to lower the threshold to admit more training instances. This leads us to one of the most striking features of Figure 4 on how the relations are separated into

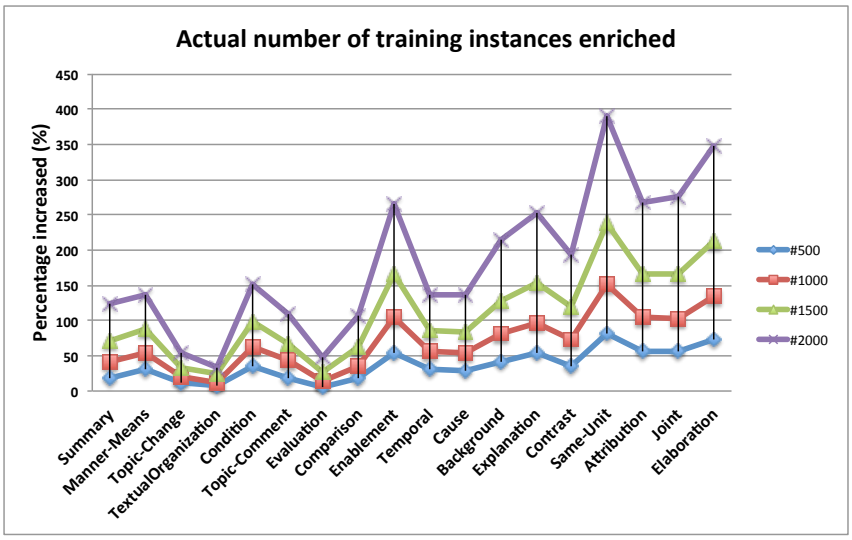
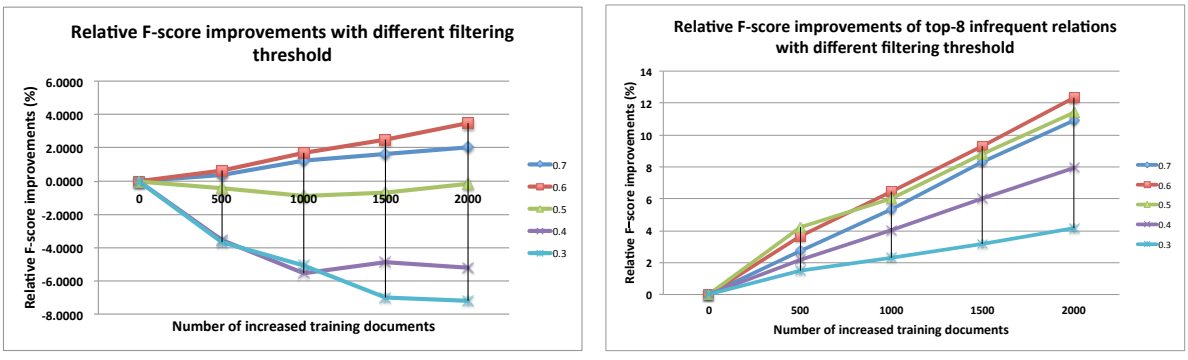


Figure 5: Actual number of training instances enriched (%)

two clusters. While there are improvements for the infrequent relations, there is no gain, or even small negative impact, on the frequent relations. This phenomenon clearly shows that co-training *without filtering* can be harmful to performance. The filtering step is essential to guard against adding "false positive" instances for re-training. If the filtering threshold is set too low, then the frequent relations may suffer. On the other hand, if the filtering threshold is set too high, then only few training instances will be added to benefit the infrequent relations.

4.4 The Impact of the Filtering Threshold

The results presented so far are based on a filtering threshold of 0.5. To examine the impact of the filtering threshold on performance, we vary the threshold. Figure 6(a) shows how the relative F-score improvement changes with a filtering threshold from 0.3 to 0.7 aggregated across all the 18 relations. The results shown in the figure are based on all the instances in the entire dataset. In other words, the performance of the frequent relations, due to their much higher frequencies, completely dominates the performance of the infrequent ones. Thus, Figure 6(b) shows a corresponding graph aggregated across only the top-8 infrequent relations.



(a) Across all the 18 relations

(b) Across the top-8 infrequent relations

Figure 6: Changes in relative F-score with varying filtering agreement score threshold

Compared with the filtering threshold of 0.5 shown previously, there is further improvement when the threshold is raised to 0.6 and 0.7. Particularly from Figure 6(b), there is considerable improvement across the top-8 infrequent relations. Interestingly, the peak performance gain occurs with the threshold of 0.6 — not 0.7. This shows that when the threshold is raised from 0.6 to 0.7, the reduction in the number of

documents passing through the filter hurts the gain in performance.

The reader may wonder whether this kind of performance improvements will continue to grow under the effective threshold with more unlabeled resources added in. To explore the answer to this question, we employ the New York Times text corpus (Sandhaus, 2008) by adding a small subset of its documents to our existing unlabeled documents. Then we conduct the same experiment with the expanded unlabeled resources, and the result in Figure 7 shows that the performance will continue to improve at a lower rate and finally tend to stabilize.

Next let us examine the situation when the filtering threshold is reduced from 0.5 to 0.4 and 0.3. Aggregated across all the 18 relations, Figure 6(a) clearly shows that there is performance loss. Consistent with the performance loss shown in Figure 4 for the frequent relations, this is the situation when the extra training instances passing through the filter introduce too much noise and hurt overall performance. Interestingly, Figure 6(b) shows that there is always a positive performance gain for the top-8 infrequent relations, regardless of whether the filtering threshold is 0.3 or 0.7. This suggests that infrequent relations and frequent relations may need different threshold. We will follow up on this heuristic in Section 4.7.

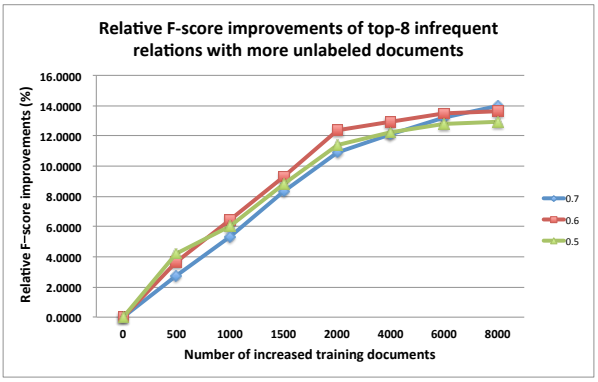


Figure 7: The impact of more unlabeled resources

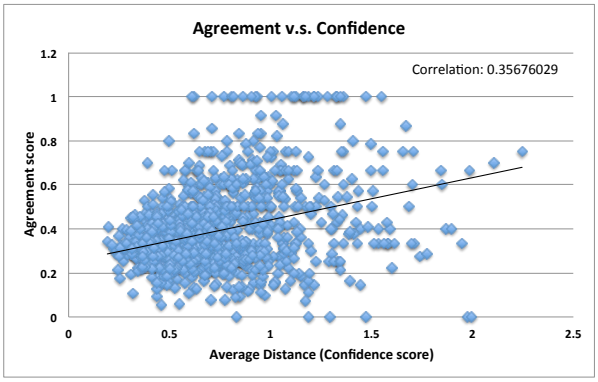


Figure 8: Agreement score v.s. confidence score

4.5 Using the Confidence Score to Approximate the Agreement Score

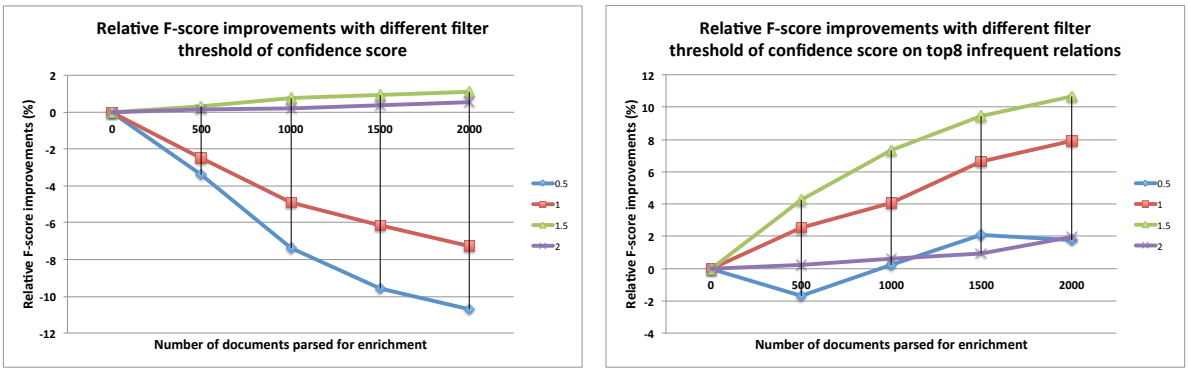
As discussed in Section 3, we explore a second form of enrichment. The agreement score reported so far requires the use of both the CODRA parser and the SR-parser. The former takes cubic time and the latter takes linear time. The idea here is to assess whether the confidence score generated from the faster SR-parser can be used to approximate the agreement score. If this approach is successful, an even larger number of unlabeled documents can be parsed rapidly to be used for re-training.

The first step of the assessment is to calculate the correlation between the agreement score and the confidence score of the SR-parser. As shown in Figure 8, which plots the correlation for all the 2,000 unlabeled documents, there is a weak correlation between the two scores. While the overall correlation is 0.36, it is promising to see that when the confidence score becomes higher (e.g., greater than 1.5), the correlation with the agreement score becomes stronger. It is also important to note that there is a significant drop in the number of documents passing the confidence score threshold of 2.

Corresponding to the two graphs in Figure 6, the two graphs in Figure 9 show the performance change using the confidence score of the SR-parser with varying filtering threshold. Figure 9(a) shows how the relative F-score changes with a filtering threshold from 0.5 to 2 aggregated across all the 18 relations. Like in Figure 6(a) before, the performance of the frequent relations, due to their superior frequencies, completely dominates the performance of the infrequent ones. Thus, Figure 9(b) shows a corresponding graph aggregated across only the top-8 infrequent relations.

In Figure 9(b), the peak performance gain occurs when the confidence score threshold is 1.5. Even when the confidence score is lowered to 1.0, the performance gain is still reasonable with 2,000 documents. But somewhat surprisingly, the performance gain drops significantly when the confidence score threshold is raised to 2. This can be explained by looking more closely back at Figure 8. The confidence

score threshold of 2 is too restrictive and very few unlabeled documents satisfy it; hence, the actual number of additional documents admitted for re-training is significantly reduced.



(a) On all 18 relations

(b) On top-8 infrequent relations

Figure 9: Overall F-score improvements with different enriched data quality via confidence score

A first glance of Figure 9(a) seems to suggest that using the confidence score of the SR-parser is ineffective. The best performance gain across all the 18 relations is barely above 1%, which is smaller than the corresponding gain in Figure 6(a). This ineffectiveness is completely due to the behavior of the frequent relations. However, Figure 9(b) paints a rather different picture. For the top-8 infrequent relations, there is a peak performance gain of about 10% with 2,000 documents. This gain is almost as good as the peak performance gain shown in Figure 6(b) with 2,000 documents. Given that the SR-parser is significantly faster than the CODRA parser, it is promising to use the confidence score of the SR-parser to approximate the agreement score, so that a larger number of unlabeled documents can be used for enrichment.

4.6 Adding enriched training instances in an iterative manner

The results shown so far are based on one round of re-training. As shown in Figure 1, data enrichment can be done iteratively. The table below shows the relative F-score improvement on the top-8 infrequent relations when enrichment is done in increments of 500 documents. Here we process 500 unlabeled documents, re-train the SR-parser with the documents passing through the filter, then process the next batch of 500 documents, and so on.

# of documents	Basic	Iterative (batches of 500 documents)
1000	4.05	4.61
1500	6.65	7.47
2000	7.90	8.95

Table 2: Relative F-scores improvements (%) on the top-8 infrequent relations

The results shown in the table used the confidence score of 1 as the filtering threshold. The first column is precisely the curve in Figure 9(b) for the confidence score of 1. The first row in the table, for example, shows that doing re-training twice (500 documents each time) boosts the performance when compared with re-training done once at the end. Similarly, the other rows show that there is some value in iterative re-training.

4.7 Filtering at a finer granularity

All the filtering experiments above are performed at the document level. That is, we calculate an agreement/confidence score for an entire discourse tree of a document, and if the score passes the threshold, this entire discourse tree along with every node on it will be added to the new training instances, even though some nodes on this tree may have low scores. So in this section, we will explore the idea of filtering at a finer granularity, e.g. at the node level. Due to the different mechanism of the two parsers used in

our framework, we picked the CODRA to conduct this experiment, because it is easier to filter discourse structures at node level and train its new model with partial discourse structures using CODRA. While we could not find a direct way to do it with the SR-parser.

In this experiment, we have performed both doc-level filtering and node-level filtering using the same experiment setting: we use the confidence score of CODRA itself to filter new candidate training examples, and the threshold is set to 0.5 here. The number of unlabeled documents used here is 500. The doc-level filtering works as described above, and for node-level filtering, every node with a confidence score higher than the threshold will be added to the new training set to retrain CODRA, no matter whether the document's discourse tree has a high confidence score that passes the threshold. Results of the two experiments are shown in Table 3. We can see that filtering at node-level has an advantage over filtering at doc-level for most discourse relations. And it is noteworthy that frequent relations are generally unharmed at node-level filtering, unlike at doc-level filtering.

Relation	**Doc-level**	**Node-level**	**Relation**	**Doc-level**	**Node-level**
Summary	4.265	6.811	Cause	1.827	1.965
Manner-Means	8.677	12.581	Temporal	-0.296	-0.246
Topic-Change	1.201	1.801	Background	-0.209	0.105
TextualOrganization	5.669	7.122	Explanation	-0.317	-0.106
Condition	6.656	7.488	Contrast	-0.066	0.131
Comparison	4.527	4.527	Same-Unit	-0.109	0.145
Evaluation	1.696	1.993	Attribution	-0.120	0.052
Topic-Comment	1.360	2.039	Joint	-0.211	-0.015
Enablement	2.999	3.314	Elaboration	-0.058	0.014

Table 3: Relative F-scores improvements (%) at different filtering granularities

Based on the control of filtering at a finer granularity, we can actually do more with the filtering threshold. Since in this case we can compare the score of each node to a threshold to determine whether it should be added to the new training set, we can actually set different thresholds for different types of relations. Though how to set different thresholds for different relations is still to be explored, we have run a small experiment with two different thresholds for infrequent and frequent relations separately and it shows a small increase on the performance. So we believe with more reasonable threshold set for different relations, in the future, greater improvements can be expected from using a varying threshold.

5 Conclusion

As the number of applications of discourse parsing in NLP is constantly growing, any improvement in discourse parsing performance can have considerable impact. In this paper, we first validate the underfitting hypothesis, i.e., the less frequent a relation is in the training data, the poorer the performance on that relation. This is a phenomenon that applies to most discourse parser. One solution is, of course, to create more labeled data, ideally for all the relations. However, given the resources required for manually creating labeled data for discourse parsing, we explore in this paper a training data enrichment framework that relies on co-training of the CODRA parser and the SR-parser on unlabeled documents. We also investigate using both the agreement score and the confidence score of the SR-parser to filter away "low quality" documents, whose presence in the re-training can hurt the performance. Our empirical results show that agreement score filtering can boost the performance of infrequent relations considerably. Our results also show that for infrequent relations, the confidence score of the SR-parser can also be used as a fast approximation of the agreement score.

So far our results show that our data enrichment framework is not effective for frequent relations. In ongoing work, we are studying how to augment our enrichment framework to boost the performance of even the frequent relations, and the varying threshold might be a promising solution. In the future, we plan to apply our framework to enrich training data for discourse structure and nuclearity analysis, and also to apply it to other discourse dataset(s) labeled in different ways (e.g. PDTB).

References

Steven Abney, S Flickenger, Claudia Gdaniec, C Grishman, Philip Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith Klavans, Mark Liberman, et al. 1991. Procedure for quantitatively comparing the syntactic coverage of english grammars. In *Proceedings of the workshop on Speech and Natural Language*, pages 306–311. Association for Computational Linguistics.

Kelsey Allen, Giuseppe Carenini, and Raymond T Ng. 2014. Detecting disagreement in conversations using pseudo-monologic rhetorical structure. In *EMNLP*, pages 1169–1180.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from rst discourse parsing. In *Proceedings of the Empirical Methods in Natural Language Processing, (EMNLP)*.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue - Volume 16*, SIGDIAL '01, pages 1–10, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *ACL (1)*, pages 511–521.

Shima Gerani, Giuseppe Carenini, and Raymond T Ng. 2016. Modeling content and structure for abstractive review summarization. *Computer Speech & Language*.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2014. Using discourse structure improves machine translation evaluation. In *ACL (1)*, pages 687–698.

Hugo Hernault, Danushka Bollegala, and Mitsuru Ishizuka. 2010. A semi-supervised approach to improve classification of infrequent discourse relations using feature vector extension. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 399–409. Association for Computational Linguistics.

Laurence Horn. 1989. *A natural history of negation*. Chicago: University of Chicago Press.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *ACL (1)*, pages 13–24.

Shafiq R Joty, Giuseppe Carenini, Raymond T Ng, and Yashar Mehdad. 2013. Combining intra-and multi-sentential rhetorical parsing for document-level discourse analysis. In *ACL (1)*, pages 486–496.

Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. 2015. Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*.

Man Lan, Yu Xu, Zheng-Yu Niu, et al. 2013. Leveraging synthetic discourse data via multi-task learning for implicit discourse relation recognition. In *ACL (1)*, pages 476–485. Citeseer.

Junyi Jessy Li and Ani Nenkova. 2015. Fast and accurate prediction of sentence specificity. In *AAAI*, pages 2281–2287.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. *arXiv preprint arXiv:1603.02776*.

Daniel Marcu. 2000. *The theory and practice of discourse parsing and summarization*. MIT press.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Kathleen McKeown and Or Biran. 2013. Aggregated word pair features for implicit discourse relation disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 69–73. The Association for Computational Linguistics.

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

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Caroline Sporleder and Alex Lascarides. 2008. Using automatically labelled examples to classify rhetorical relations: An assessment. *Natural Language Engineering*, 14(3):369–416.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 235–243. Association for Computational Linguistics.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1507–1514. Association for Computational Linguistics.

Inferring Discourse Relations from PDTB-style Discourse Labels for Argumentative Revision Classification

Fan Zhang **Diane Litman** **Katherine Forbes Riley**
University of Pittsburgh
Pittsburgh, PA, 15260
{zhangfan,litman}@cs.pitt.edu, katherineforbesriley@gmail.com

Abstract

Penn Discourse Treebank (PDTB)-style annotation focuses on labeling local discourse relations between text spans and typically ignores larger discourse contexts. In this paper we propose two approaches to infer discourse relations in a paragraph-level context from annotated PDTB labels. We investigate the utility of inferring such discourse information using the task of revision classification. Experimental results demonstrate that the inferred information can significantly improve classification performance compared to baselines, not only when PDTB annotation comes from humans but also from automatic parsers.

1 Introduction

Widely used in discourse research, Penn Discourse Treebank (PDTB)-style annotation (Prasad et al., 2008) adopts a lexically grounded approach by anchoring discourse relations according to discourse connectives. In a typical PDTB annotation process, an annotator first locates discourse connectives (explicit or implicit) then annotates text spans as their arguments. While the process of manual PDTB annotation has been demonstrated to yield reliable results (Alsaif and Markert, 2011; Danlos et al., 2012; Zhou and Xue, 2015; Zeyrek et al., 2013), it yields more shallow annotation when compared to another widely-used discourse scheme, namely Rhetorical Structure Theory (RST) (Mann and Thompson, 1988; Carlson et al., 2002). This is because when using RST a text is represented as a hierarchical discourse tree, while when using PDTB the relations exist only locally (typically between sentences or clauses).

The lack of discourse information across larger contexts potentially limits the utility of PDTB-style labels. Feng et al. (2014) found that when applied to the tasks of sentence ordering and essay scoring, an RST-style discourse parser outperformed a PDTB-style parser. Performance on both tasks was also likely impacted by parsing errors. To address both the local nature of PDTB-style annotations as well as the errors introduced by state-of-the-art discourse parsers, we propose to first build paragraph-level discourse structures from annotated PDTB labels, then to infer discourse relations based on these structures. We hypothesize that features extracted from inferred relations will improve performance in downstream applications, compared to features extracted from only original annotations.

To verify our hypotheses, we choose the task of argumentative revision classification (Zhang and Litman, 2015; Zhang and Litman, 2016), which aims to identify the purpose of an author's revisions during argumentative writing. This task first detects the differences between two drafts of a paper at the sentence level, then labels each revision using one of five categories: *Claim/Ideas*, *Warrant/Reasoning/Backing*, *Evidence*, *General Content* and *Surface*. While Zhang and Litman (2016) demonstrated decent classification performance without using discourse structure, an error analysis of their results suggests that discourse relations might improve performance (e.g., their current system has trouble differentiating between changing reasoning (*Warrant/Reasoning/Backing*) versus smoothing transitions (*General Content*)). One of the corpora used in their work has recently been annotated with PDTB-style relations, which allows us to explore the utilization of both manually and automatically-produced discourse relations for revision classification. Revision classification also allows us to utilize not only the discourse structure of each version of a paper, but also the differences in discourse structure across versions.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2615–2624, Osaka, Japan, December 11-17 2016.

Draft2 Essay	(1) The lustful are those who long and crave for one another. (2) The person guilty of lust is put in this layer of hell because of his over indulgence of sexual-pleasure. (3) The man who is stuck in this layer is Hue Heffner. (4) He has devoted his entire life for other people 's lustful pleasure and his own. (5) He has spent millions on working on his mansion which is for the purpose of other lustful desires. (6) People who were stuck in this layer are constantly whipped around and " banging " into one another. (7) What you do in your Earthly presence follows with you into Hell. (8) For him and like many others he is now tortured in a whirlwind of torment with others lustful accommodators with himself.
Annotated PDTB	(1->2, EntRel), (2->3, Expansion), (3->4, Contingency), (4->5, Expansion), (5->6, EntRel), (6->7, Contingency), (7->8, Contingency)

Table 1: A paragraph from an essay about putting contemporaries into different levels of hell (top), and annotated PDTB relations between sentences (bottom). The paragraph can be divided into two segments (Section 3.2). In the first segment (sentences (1) to (3)) the author introduces the person to be put in the lustful layer. In the second segment (sentences (4) to (8)), the author states why this person belongs there and how he will be treated. PDTB relations are processed from PDTB annotations ignoring the discourse connectives, e.g. (1->2, EntRel) represents the discourse information: (Arg1: Sentence1, Arg2: Sentence2, Relation Type: EntRel).

2 Related Work

Previous applications of PDTB-style annotations have typically been based on the extraction of PDTB relation occurrence patterns. Lin et al. (2011) encoded PDTB information into the entity-grid (Barzilay and Lapata, 2008) model for textual coherence evaluation. Mithun and Kosseim (2013) utilized PDTB relations for blog summarization. A limitation of the pattern approach is that since it targets a whole paragraph/essay, it is not straightforward to use for prediction tasks on individual sentences. In our work we propose to infer contextual PDTB discourse relations for each sentence, thus enabling the utilization of less-local PDTB information during single sentence prediction.

The construction of our discourse structure looks similar to the building of an RST tree (Duverle and Prendinger, 2009), and there are also prior efforts in combining the benefits of RST and PDTB (e.g., when building a Chinese discourse corpus (Li et al., 2014)). However, the focus of our work is different. The construction of an RST tree aims at creating a discourse representation of the whole sentence/paragraph/essay. We focus on grouping semantically similar text and assigning higher priority to specific local discourse relations. We expect our structure to be able to select the relations that should be propagated to improve performance in downstream applications.

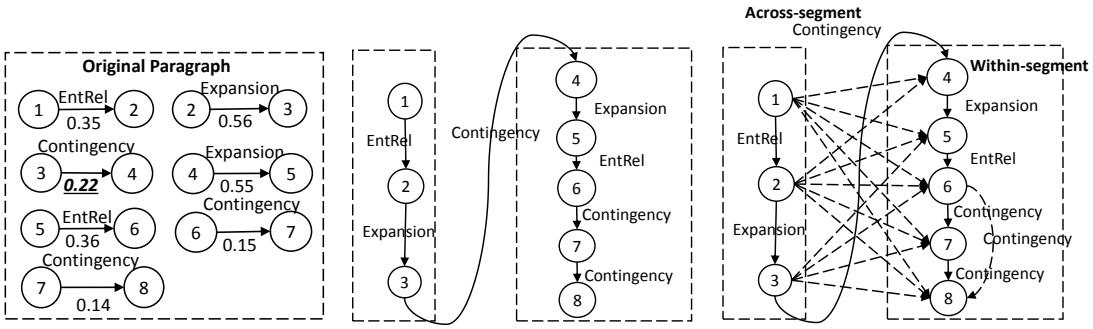
3 Inferring Discourse Information from PDTB-style Labels

3.1 Intuitions for PDTB relation inference

Different from other discourse annotations, the PDTB annotation schema anchors at the labeling of discourse connectives and labels text spans around the connective. The annotator either locates the "Explicit" connectives or manually fills in the "Implicit" connectives between two text spans. The text span where the connective structurally attaches to is called **Arg2**, while the other text span is called **Arg1**. The spans are usually used at the level of sentence/phrase. In (Prasad et al., 2014), five relation types are annotated: *Explicit*, *Implicit*, *AltLex*, *EntRel* and *NoRel*. Within the *Explicit*/*Implicit* relations, the senses of relations are further categorized at multiple levels. In Level-1, the relations are categorized to 4 senses: *Comparison*, *Contingency*, *Expansion* and *Temporal*. In this paper we focus on the type/sense of Level-1 relations only and ignore the discourse connectives[1]. *Arg1*, *Arg2* and the discourse relation type/sense are used as demonstrated in Table 1. For the *Explicit*/*Implicit* relations, we use the sense of the relation directly to represent the relation. Below we explain our intuitions for inferring new discourse relations within the paragraph. Sections 3.2 and 3.3 then detail our corresponding computational approaches.

Intuition 1. Latent discourse relations can be inferred from annotated discourse relations. In this paper we explore two possible cases: **1) Same type transition**: If sentence A has relation type T with sentence

[1]We plan to explore connectives in future work.



(a) Original PDTB annotation (b) After linear segmentation (c) After relation propagation

Figure 1: The construction of PDTBSegment structure of the example in Table 1. As sentence similarity between 3 and 4 is 0.22, smaller than the value 0.56 (before) and 0.55 (after), the paragraph is segmented to segment(1-3) and segment (4-8). Afterwards relations are inferred both within the segment and across the segments. The dashed lines represent the propagated relations.

B and sentence B has the same relation type with sentence C, we can infer that A has relation type T with C. In the example in Table 1, a *Contingency* relation between sentences 6 and 8 will be inferred from the *Contingency* relationships between sentences (6,7) and sentences (7,8). **2) Across segment propagation**: If a paragraph can be segmented to text segments semantically dissimilar to each other (i.e. the two text segments are serving different semantic purposes), the discourse relation of sentences on the boundary of two segments can be propagated to infer weaker relations between all sentences in the segments. In the example in Table 1, due to the discourse relation between sentences 3 and 4 and the segment boundary between them, the segment from 4 to 8 will also be viewed as a reasoning (*Contingency*) of the segment from 1 to 3 (why and how Hue Heffner belongs to the lustful level), and weak relations are inferred between sentences (1,2,3) and (4,5,6,7,8).

Intuition 2. The importance of discourse relations to argumentation varies even if the relation types are the same. The relations connecting the semantically dissimilar segments are likely to be more important than the relations within a segment. In Table 1, the *Contingency* relation between sentences 3 and 4 transits the thesis introduction to the arguments supporting the thesis. The *Contingency* relation between sentences 6 and 7 is just a transition to smooth the description of how Hue Heffner is going to be treated.

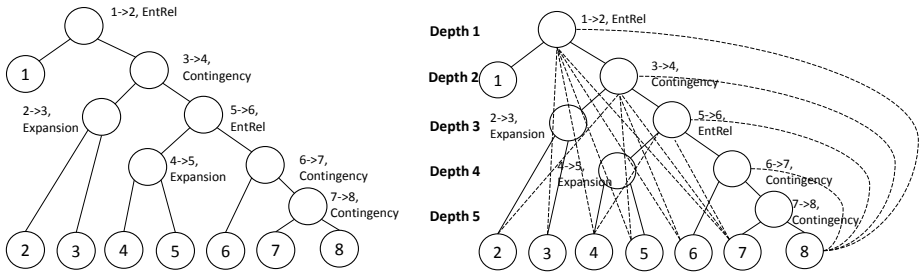
3.2 PDTBSegment

Based on intuition 1, the *PDTBSegment* approach emphasizes the inference of discourse relations.

Step1. Linear segmentation. Inspired by the TextTiling algorithm (Hearst, 1997) for linear segmentation, we utilize the "valley" of semantic similarity scores between sentences as the segmentation boundary. The summed word-embedding vector is calculated for each sentence[2] and cosine value between vectors is used as the similarity score. Similarity scores indicates a possible segmentation boundary. In the example of Figure 1(a), the similarity between (2,3) and the similarity between (4,5) are larger than the similarity between (3,4), in other words, sentence 3 and 4 has a low similarity score preceded by and followed by high similarity scores, thus the paragraph is first segmented into segment (1,2,3) and segment (4,5,6,7,8) as in Figure 1(b).

Step2. Relation inference. **1) Within segment.** We conduct "same type transition" for sentences within the same segment. As in Figure 1(c), there exists relation *Contingency* between 6 and 8 as the same relationship exists between 6, 7 and 7, 8. **2) Across segment.** "Across-segment propagation" is conducted for sentences in different segments. If there exists relation (type T) between two segments Seg1 and Seg2, a relation with type T is inferred for each sentence in Seg1 and each sentence in Seg2. In Figure 1(c), we propagate the *Contingency* relations between sentence (1,2,3) and sentences (4,5,6,7,8).

[2]Pre-trained word2vec vectors from (Mikolov et al., 2013).



(a) After tree structure construction (b) After relation propagation

Figure 2: PDTBTree structure of Table 1 example. The dashed lines represent the propagated relations.

Segment	1(Arg2)	2	3	4	5	6	7	8
1(Arg1)	N/A[a]	Ent	N/A[b]	Cont(2,0)[c]	Cont(2,1)	Cont(2,2)	Cont(2,3)	Cont(2,4)
2	N/A	N/A	Expan	Cont(1,0)	Cont(1,1)	Cont(1,2)	Cont(1,3)	Cont(1,4)
3	N/A	N/A	N/A	Cont	Cont(0,1)	Cont(0,2)	Cont(0,3)	Cont(0,4)
4	N/A	N/A	N/A	N/A	Expan	N/A	N/A	N/A
5	N/A	N/A	N/A	N/A	N/A	EntRel	N/A	N/A
6	N/A	N/A	N/A	N/A	N/A	N/A	Cont	Cont(1)
7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Cont
8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Tree	1(Arg2)	2	3	4	5	6	7	8
1(Arg1)	N/A	Ent-1	Ent-1(0,1)[d]	Ent-1(0,1)	Ent-1(0,2)	Ent-1(0,2)	Ent-1 (0,3)	Ent-1(0,3)
2	N/A	N/A	Expan-3	Cont-2(1,0)	Cont-2(1,1)	Cont-2(1,1)	Cont-2(1,2)	Cont-2(1,3)
3	N/A	N/A	N/A	Cont-2	Cont-2(0,1)	Cont-2(0,1)	Cont-2(0,2)	Cont-2(0,3)
4	N/A	N/A	N/A	N/A	Expan-4	Ent-3(1,0)	Ent-3(1,1)	Ent-3(1,2)
5	N/A	N/A	N/A	N/A	N/A	Ent-3	Ent-3(0,1)	Ent-3(0,2)
6	N/A	N/A	N/A	N/A	N/A	N/A	Cont-4	Cont-4(0,1)
7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Cont-5(0,1)
8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 2: Relation matrix constructed for the PDTBSegment approach (Upper) and the PDTBTree approach (Below). Ent is short for EntRel, Expan short for Expansion and Cont short for Contingency.

[a]Relationship between clauses within the sentence is not used in relation inference.

[b]No relations can be inferred between 1 and 3.

[c]Cont(2,0) means distance to real Arg1 is 2 and distance to real Arg2 is 0. Here the inferred relation is coming from the labeled relation (3->4, Contingency). 3 is the real Arg1 and 4 is the real Arg2. Distance to real Arg1 is 2 as the the distance between 1 and 3 is 2.

[d]Ent-1(0,1) stands for Depth 1 distance, distance=0 for Arg1 and distance = 1 for Arg2.

3.3 PDTBTree

PDTBTree focuses on intuition 2 using sentence aggregation. To better separate important discourse relations, a hierarchical tree structure is constructed for each paragraph and relations then inferred.

Step 1. Tree construction. As in Figure 2(a), the tree is constructed iteratively starting with each sentence constructed as a leaf node. Semantic similarities between adjacent sentences are calculated in the same manner as the *PDTBSegment* approach. In each round, the two most similar nodes are selected and merged into one node and similarities between the merged node and its adjacent nodes are calculated[3]. The selection and merge of nodes repeats until there is only one root node left.

Discourse relations are assigned to the non-leaf nodes after tree construction. For each tree node, sentences in its left and right child are listed as $Nodes_{left}$ and $Nodes_{right}$. Relations with Arg1 in $Nodes_{left}$ and Arg2 in $Nodes_{right}$ are assigned to the merged node. For example, the discourse relation (1->2, EntRel) is assigned to the root node as sentence 1 is in its left child and sentence 2 is in its right child. After this step we bind each non-leaf node with one or several discourse relations.

Step 2. Relation inference. Relations are first assigned different levels of importance as depths. As in Figure 2(b), the assignment starts at the root node and traverses the whole tree until all the non-leaf nodes

[3]The similarity between merged nodes is calculated as the average of the similarity between their child leaf nodes.

are labeled. Depth starts from 1 and smaller number indicates larger importance. As in the example, we notice that the transition from the thesis to its reasoning (3->4) is recognized as a depth-2 relation while the transitions between sentences 6,7,8 are recognized as depth-4 and depth-5 relations.

Afterwards discourse relations are inferred by traversing up from the leaf nodes back to their parent nodes. The parent node is used as the discourse connector and its child leaf nodes are used as Arg1 and Arg2. For example, in Figure 2(b), sentence 3 is the left child of the node (3->4, Contingency) and sentence 5 is the right child. Thus we infer the discourse relation between 3 and 5 as (3->5, Contingency).

3.4 Constructing the relation matrix

For both approaches, relation matrices are constructed to represent the discourse information as in Table 2. Extraction of features using the matrix is described in the next section. Relations already labeled by the annotator/parser are directly recorded in the matrix. Observing that the reliability of an inferred relation decreases as the number of annotated relations connecting the arguments increases, we record not only the relation types but also the **"distance"** information for the inferred relations.

For the **PDTBSegment** approach, distances are recorded separately for within-segment relations and across-segment relations. For within-segment relations, the distance is recorded according to the number of sentences between Arg1 to Arg2. For example, distance for sentence 6 and 8 is recorded as 2 as there is one sentence between the two sentences. For across-segment relations, distances are recorded for both Arg1 and Arg2 according to their distances to the real Arg1/Arg2 of the across-segment relation as (Dist1, Dist2). For example, distance between sentence 1 and 5 is recorded as (2,1) as there is the distance of 2 between sentence 1 and 3 and there is the distance of 1 between sentence 4 and 5.

For the **PDTBTree** approach, we traverse up from Arg1 and Arg2 to their closest common parent node and count the distances for both arguments as (Dist1, Dist2). In Table 2, distance between sentence 2 and 5 is recorded as (1,1) as we back trace both nodes to their parent node (3->4). As sentence 2 is the real text span in relation node (2->3) and sentence 5 is in the node (5->6), we get distance 1 for sentence 2 as the distances between (2->3) and (3->4) in the tree is 1; similarly, we get distance 1 for sentence 5.

4 Task and Data Description

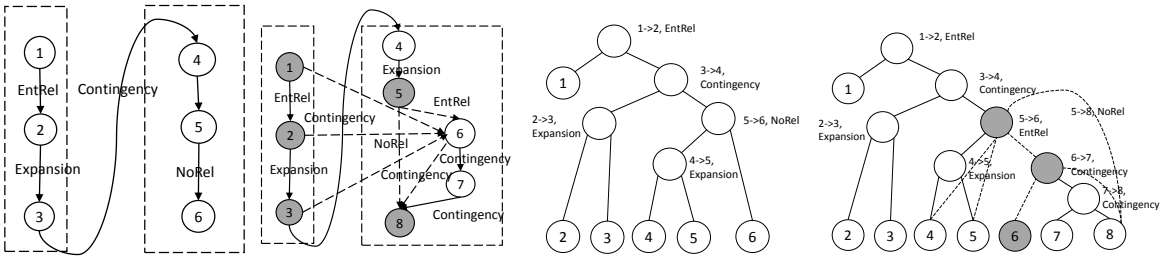
Argumentative revision classification. The task of revision classification aims to detect then categorize an author's changes to their writing. Revision research has been conducted on Wikipedia articles (Bronner and Monz, 2012; Daxenberger and Gurevych, 2013) and argument-oriented study essays (Zhang and Litman, 2015). The example in Table 1 contains a paragraph from a revised student essay. Comparing to its previous draft, the changes are the addition of sentence 6 and sentence 7. The addition of sentence 6 is labeled as (Null->6, "Add", "General Content"). The full classification process involves the alignment of sentences/clauses to locate changes (recognizing the alignment Null->6 and the revision operation "Add") and the classification of change types (identifying the revision type "General Content"). In this work we assume perfect alignment and focus on improving classification performance (the recognition of "General Content") by using inferred PDTB information.

Annotated revision dataset. To evaluate revision classification performance, we use the two corpora used in (Zhang and Litman, 2016)[4]. Each student wrote two essays: *Draft 1* where the students initially write the essay, and *Draft 2* where the students revise Draft 1 after receiving comments from other students. Corpus A contains 47 students (94 essays) and 1267 revised sentence pairs, talking about placing contemporaries into Dante's Inferno. Corpus B contains 63 students (126 essays) and 1044 revised sentence pairs, where students explain the rhetorical strategies used by the speaker/author of a previously read lecture/essay. Distribution of revisions is shown in the first columns of Tables 4 and 5.

PDTB annotation. Recently PDTB discourse information was annotated on corpus A by one of the early developers of the D-LTAG environment (which engendered the PDTB framework)[5] (Forbes et al., 2003; Webber, 2004; Forbes-Riley et al., 2016). Five relation types were annotated: *Explicit*, *Implicit*, *EntRel*, *AltLex* and *NoRel*. Within the *Explicit* and *Implicit* types, four level-1 senses were labeled:

[4]Both corpora are from high school AP English classes.

[5]Thus considered as an expert annotator.



(a) Segment for Draft 1 (b) Segment for Draft 2 (c) Tree for Draft 1 (d) Tree for Draft 2

Figure 3: The change of discourse structure from Draft 1 (D1) to Draft 2 (D2). The gray nodes are the affected nodes and the dashed lines are the affected relations. Sentences are aligned as (1->1), (2->2), (3->3), (4->4), (5->5), (6->8), (Null->6), (Null->7).

Features	Example
Loc	D1-Arg1[a]: N/A, D1-Arg2: N/A, D2-Arg1: Contingency, D2-Arg2: EntRel
Seg	**Individual**: WithinSegment: D1-Arg1: N/A, D1-Arg2: N/A, D2-Arg1: Contingency, D2-Arg2: EntRel, AcrossSegment: D2-Arg2: (Contingency, $\frac{1}{3}$)[b] **Structure**: WithinSegment[c]: (-1, 1, 0, 0, 0.5, 0, 0), AcrossSegment: (0, 0, 0, 0, $\frac{1}{3}$, 0, 0)
Tree	**Individual**: D1-Arg1: N/A, D1-Arg2: N/A, D2-Arg1: Contingency-4, D2-Arg2: (EntRel-1,$\frac{1}{8}$), (Contingency-2, $\frac{1}{4}$), EntRel-3 **Structure**: Depth1Vector: (0,0,0,0,0,0), Depth2Vector:(0,0,0,0,0,0), Depth3Vector: (-1,1,0,0,0,0), Depth4Vector: (0,0, 0,0,1,0,0)

Table 3: Examples of the features extracted for the added sentence 6 in Table 1.

[a]D1-Arg1 means features of sentence acting as Arg1 in the first Draft.
[b](Contingency, $\frac{1}{3}$) represents relation type Contingency with weight $\frac{1}{3}$.
[c]The columns of the change vector are (NoRel, EntRel, AltLex, Comparison, Contingency, Expansion, Temporal).

Comparison, *Contingency*, *Expansion* and *Temporal*. Similarly to (Prasad et al., 2011), the annotator relaxed the structural adjacency constraint, allowing the annotation of relations between non-adjacent sentences. Annotated and inferred PDTB information will be used to construct features for revision classification as described next.

5 Using Inferred PDTB for Classification

In this section we investigate whether using the annotated PDTB information itself can improve the classification performance, and whether such performance can be improved with the inferred PDTB information. For both of our inference approaches, we extract features individually to utilize the inferred relation type and further extract structure change features to utilize the PDTB structures built.

5.1 Features

The *PDTBSegment* and *PDTBTree* structures are constructed for both drafts as in Figure 3. Table 3 shows the PDTB features extracted for the added sentence 6 in Table 1, with features explained below.

Baseline features (Base). Features in (Zhang and Litman, 2015) are used as a baseline. Besides *Unigram* features, *Location* features encode the location of the revised sentence. *Textual* features encode revision operation, sentence length, edit distance between revised sentences and punctuation. *Language* features encode part of speech (POS) unigrams and difference in POS tag counts.

Features using the labeled local PDTB information (Local). Features are extracted as the types of relations a sentence is involved with (i.e. the relation where the sentence acts as Arg1 or Arg2.) Features are extracted for sentences in both drafts. If a sentence is added or deleted, the features for the empty sentence are marked as N/A.

Features using PDTBSegment (Segment).

Individual features Within each draft, the features of sentences are extracted based on the relation ma-

trix. Similar to **Local**, we extract the discourse relation type of each sentence acting as Arg1 and Arg2[6]. Features for across-segment relations are extracted separately since the discourse relations across segments are likely to be more important than relations within segments. Weights are assigned to relations according to their distance information. A within-segment relation with distance (d1) is assigned weight $\frac{1}{d1+1}$; an across-segment relation with distance (d1, d2) is assigned weight $\frac{1}{(d1+1)*(d2+1)}$. If a sentence is involved with multiple same-type relations, the relation with the largest weight is chosen.

Structure change features For these across draft features, the segment structures created for draft 1 and draft2 are compared. Nodes of segment structures are aligned according to the sentence alignment information. After comparison, the aligned nodes that are affected by the revision are selected, where the change of their related relations with the revised sentence are counted. For example in Figure 3(b), sentences 1, 2, 3, 5, 8 are affected by the addition of sentence 6. For sentence 1, 2, 3, sentence 6 brings addition of three across-segment relations. For sentence 5, the original "NoRel" label between sentence 5 and sentence 8 is removed. For sentence 8, relation between sentence 6 and sentence 8 is added. A vector of relation changes is thus created according to the relation matrix.

Features using PDTBTree (Tree).

Individual features Features are collected in a similar manner as the **PDTBSegment** approach. To enlarge the difference of different-depth relations, weight $\frac{1}{2^{d1+d2}}$ is assigned to a relation with distance (d1, d2) .

Structure change features Due to the complexity of the tree structure, only the non-leaf nodes that are directly related to the revised sentence (i.e. the sentence as Arg1 or Arg2 of the relation) are considered in the extraction of structure changes. As in Figure 3(d), the added sentence 6 acts as Arg2 in node (5->6) and Arg1 in node (6->7). Change of relations (4->6), (5->6) are considered as the changed relations of node (5->6). Change of relations (5->8) and (6->8) are the changed relations of node (6->7). Change vectors are calculated in similar manners as the **PDTBSegment** approach at each depth. To avoid data sparsity, the depth number is limited to 4 to reduce the number of features[7].

It is important to notice that as in the standard PDTB annotations, the spans of arguments may cover only a part of a sentence or multiple sentences. If the span covers only a part of the sentence and the two spans of the relation come from two different sentences, we use the relationship as the relation between the two sentences. If the two spans come from one sentence, we consider it as a self-relation. Such information is used in the extraction of local PDTB features but not used in relation inference. If a span of a relation covers multiple sentences, we infer relations as in the PDTBSegment approach (consider the multiple sentences span as a text segment). Also, it is possible to have more than one relation annotation between two sentences. In that case all the relations are kept and used in relation inference.

5.2 Experiments and Results

We repeated the experiment in (Zhang and Litman, 2016) using our new proposed feature group[8]. 5-class revision category classification[9] was conducted with the SVM[10] classifier. Two hypotheses are proposed: 1) For manually labeled PDTB relations, using features extracted from inferred relations has better performance than using baseline features or baseline with only local PDTB features. 2) For automatically labeled PDTB relations, using features extracted from the inferred relations reduces the noise introduced by the PDTB parser and has better performance than using only local PDTB features.

Based on the hypotheses, two experiments were conducted. In both experiments we compared the results using inferred information to the baseline results, and to the results with baseline features plus each individual feature group[11]. In Experiment 1, we experimented with the PDTB features extracted from manual labels on Corpus A (where we have manual annotations). In Experiment 2, we experimented

[6]The row of the sentence in the relation matrix corresponds to Arg1 and the column corresponds to Arg2.

[7]If the depth of tree is larger than 4, the depth of the relation is still considered as 4.

[8]In this paper we focus on the comparison of features and thus do not directly compare our approach with the sequence labeling approach used in their work

[9]*Claim*, *Warrant*, *Evidence*, *General* and *Surface*

[10]SVM model implemented with Weka (Hall et al., 2009).

[11]We also experimented mixing all the features groups together but did not observe significant improvement.

Corpus A	Base	Base+Local	Base+Segment	Base+Tree
Claim(111)	0.540	0.530	0.500	**0.578**‡*
Warrant(390)	0.680	0.693	**0.715**‡*	0.713‡*
Evidence(110)	0.288	0.347*	0.387‡*	**0.415**‡*
General(356)	0.694	0.715	**0.746**‡*	0.724*
Surface(300)	0.868	**0.872**	0.869	0.870
Average(1267)	0.614	0.630	0.642*	**0.658**‡*

Table 4: Experiment 1. With manually labeled PDTB. The average F-measure of 10-fold (student) cross-validation is reported, average represents the unweighted average F1 of all 5 categories. $*$ indicates significantly better than the baseline (paired T-test, $p<0.05$), ‡ indicates significantly better than (Base+local), **bold** indicates best.

Corpus A	Base	B+Local	B+Segment	B+Tree	B-	(B-)+Local	(B-)+Segment	(B-)+Tree
Claim(111)	**0.540**	0.517	0.516	0.518	0.466	0.475⋆	0.501⋆	0.504⋆
Warrant(390)	0.680	0.669	**0.698**‡	0.682	0.658	0.647	0.686⋆	0.671⋆
Evidence(110)	0.288	0.299	0.276	**0.306**	0.274	0.266	0.261	0.282
General(356)	0.694	0.683	**0.702**‡	0.683	0.621	0.643⋆	0.696⋆	0.682⋆
Surface(300)	**0.868**	0.865	**0.868**	0.863	0.841	0.835	0.843	0.844
Average(1267)	0.614	0.605	**0.617**‡	0.609	0.572	0.573	0.597⋆	0.596⋆
Corpus B	Base	B+Local	B+Segment	B+Tree	B-	(B-)+Local	(B-)+Segment	(B-)+Tree
Claim(76)	0.504‡	0.471	0.496‡	**0.512**‡	0.421	0.433	0.443⋆	0.451⋆
Warrant(327)	0.611	0.620	**0.635***	0.609	0.588	0.591	0.610⋆	0.605⋆
Evidence(34)	0.024	0.088	**0.094**	0.044	0.024	0.088	0.088	0.088
General(216)	**0.505**‡	0.459	0.503‡	0.484‡	0.451	0.455	0.477⋆	0.469⋆
Surface(391)	0.867	0.853	**0.872**‡	0.865	0.848	0.851	0.855	0.853
Average(1044)	0.503	0.495	**0.520**‡	0.503	0.466	0.483	0.495⋆	0.493⋆

Table 5: Experiment 2. With automatically labeled PDTB. B short for Base, B- is a weaker baseline using unigram and *Textual* features from the baseline approach. ⋆ indicates significantly better than B-. ‡ indicates significantly better than (B+local), **bold** indicates best.

with the PDTB features extracted from labels generated with Lin's automatic PDTB parser (Lin et al., 2014) on Corpora A[12] and B. All experiments were conducted using 10-fold (student) cross-validation with 300 features selected[13] using learning gain ratio. Tables 4 and 5 demonstrate the results.

Experiment 1 results provide support for our first hypothesis. Comparing to the baseline, Base+Local (using only features from the labeled PDTB relations) yields a significant improvement only when classifying *Evidence* revisions and a non-significant overall average improvement. In contrast, both Base+Segment and Base+Tree (our inference-based approaches) yield several significant improvements over the baseline[14]. Comparing to the baseline, the **PDTBSegment** approach yields significant improvement in the classification of *Warrant*, *Evidence* and *General Content* revisions and the **PDTBTree** approach yields significant improvement in the classification of all revisions except *Surface*. For the minority category *Evidence*, the **PDTBTree** approach improved F1 from 0.288 to 0.415. Comparing to the results using only labeled PDTB, the **PDTBSegment** approach yields significant improvement in the classification of *Warrant*, *Evidence* and *General*, while the **PDTBTree** approach yields significant improvement in the classification of *Claim*,*Warrant* and *Evidence* and a significant overall F1 improvement.

Experiment 2 results support our second hypothesis. On corpus A, we observe a significant performance drop ($p<0.05$) in average F1 score for all the (B+) feature groups comparing to the Experiment 1 results in Table 4, which indicates that the noisy output generated by the parser impacts the performance[15]. While we do not gain significant improvement over the baseline using the inferred relations, we still observe significantly better performance using the **PDTBSegment** approach comparing to using

[12]The same fold is used as Experiment 1.

[13]Features selected only on the training folds each round.

[14]We also tested using just individual features (without the structure change features) and both approaches still significantly outperform the baseline.

[15]We compared the output of the PDTB parser and the manual annotation on Corpus A, indicating that the F-measure for the relation prediction is 0.45.

only the automatically labeled relations. Meanwhile, we observe that the inferred PDTB features can still significantly improve the performance of a weaker baseline with *Unigram* and *Textual* features while the Local features can't, suggesting that the our approach is still effective even with the noisy parser output.

6 Conclusion and Future Works

This paper presented two approaches to construct a paragraph-level discourse structure from local PDTB discourse labels, and infer discourse relations within the structure. We applied our approaches on the task of argumentative revision classification with manually/automatically labeled PDTB information. Results demonstrated that using features extracted from inferred PDTB relations yields better revision classification performance than using features from original PDTB annotations. Results also showed that our method reduced the impact of the automatic PDTB parsing errors.

We believe our work can be expanded from two perspectives. 1) From the PDTB perspective, we can investigate our approach on other traditional PDTB applications such as coherence evaluation to see if we can still observe an improvement on these tasks. 2) From the argumentative revision classification perspective, we plan to explore what aspects of our proposed approach yields most robustness to errors from automatic PDTB parsers. We also want to try other discourse parsers in recent shallow discourse parsing shared tasks (CoNLL, 2016). We also plan to investigate whether the RST-style discourse information can also improve the classification performance and compare the results with our approach. We also plan to investigate whether our approaches can be applied to other type of writings besides argumentative writings.

Acknowledgments

We want to thank the members of the SWoRD and ITSPOKE groups for their helpful feedback and all the anonymous reviewers for their suggestions.

This research is funded by NSF Award #1550635 and the Learning Research and Development Center of the University of Pittsburgh.

References

Amal Alsaif and Katja Markert. 2011. Modelling discourse relations for Arabic. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 736–747. Association for Computational Linguistics.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Amit Bronner and Christof Monz. 2012. User edits classification using document revision histories. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 356–366. Association for Computational Linguistics.

Lynn Carlson, Mary Ellen Okurowski, Daniel Marcu, Linguistic Data Consortium, et al. 2002. *RST discourse treebank*. Linguistic Data Consortium, University of Pennsylvania.

CoNLL. 2016. Conll 2016 shared task. Accessed: 2016-07-15.

Laurence Danlos, Diégo Antolinos-Basso, Chloé Braud, and Charlotte Roze. 2012. Vers le FDTB: French Discourse Tree bank. In *Proceedings of TALN 2012: 19ème conférence sur le Traitement Automatique des Langues Naturelles*, volume 2, pages 471–478. ATALA/AFCP.

Johannes Daxenberger and Iryna Gurevych. 2013. Automatically classifying edit categories in Wikipedia revisions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 578–589, Seattle, Washington, USA, October. Association for Computational Linguistics.

David A Duverle and Helmut Prendinger. 2009. A novel discourse parser based on support vector machine classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 665–673. Association for Computational Linguistics.

Vanessa Wei Feng, Ziheng Lin, Graeme Hirst, and Singapore Press Holdings. 2014. The impact of deep hierarchical discourse structures in the evaluation of text coherence. In *Proceedings of International Conference on Computer Linguistics*, pages 940–949.

Katherine Forbes, Eleni Miltsakaki, Rashmi Prasad, Anoop Sarkar, Aravind Joshi, and Bonnie Webber. 2003. D-LTAG system: Discourse parsing with a lexicalized tree-adjoining grammar. *Journal of Logic, Language and Information*, 12(3):261–279.

Kate Forbes-Riley, Fan Zhang, and Diane Litman. 2016. Extracting PDTB discourse relations from student essays. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 117–127, Los Angeles, September. Association for Computational Linguistics.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Marti A Hearst. 1997. TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64.

Yancui Li, Wenhe Feng, Jing Sun, Fang Kong, and Guodong Zhou. 2014. Building Chinese Discourse Corpus with connective-driven dependency tree structure. In *Proceedings of Empirical Methods of Natural Language Processing*, pages 2105–2114.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 997–1006. Association for Computational Linguistics.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A PDTB-styled end-to-end discourse parser. *Natural Language Engineering*, 20(02):151–184.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in neural information processing systems*, pages 3111–3119.

Shamima Mithun and Leila Kosseim. 2013. Measuring the effect of discourse relations on blog summarization. In *Proceedings of International Joint Conference of Natural Language Processing*, pages 1401–1409.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The Penn Discourse Treebank 2.0. In *Proceedings of Language Resources and Evaluation Conference*.

Rashmi Prasad, Susan McRoy, Nadya Frid, Aravind Joshi, and Hong Yu. 2011. The biomedical discourse relation bank. *BMC bioinformatics*, 12(1):1.

Rashmi Prasad, Bonnie Webber, and Aravind Joshi. 2014. Reflections on the Penn Discourse Treebank, comparable corpora, and complementary annotation. *Computational Linguistics*.

Bonnie Webber. 2004. D-LTAG: extending lexicalized tag to discourse. *Cognitive Science*, 28(5):751–779.

Deniz Zeyrek, Işın Demirşahin, AB Sevdik-Çallı, and Ruket Çakıcı. 2013. Turkish Discourse Bank: Porting a discourse annotation style to a morphologically rich language. *Dialogue and Discourse*, 4(2):174–184.

Fan Zhang and Diane Litman. 2015. Annotation and classification of argumentative writing revisions. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 133–143, Denver, Colorado, June. Association for Computational Linguistics.

Fan Zhang and Diane Litman. 2016. Annotation and classification of argumentative writing revisions. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*, page to appear, San Diego, California, June. Association for Computational Linguistics.

Yuping Zhou and Nianwen Xue. 2015. The Chinese Discourse Treebank: A Chinese corpus annotated with discourse relations. *Language Resources and Evaluation*, 49(2):397–431.

Capturing Pragmatic Knowledge in Article Usage Prediction using LSTMs

Jad Kabbara **Yulan Feng** **Jackie Chi Kit Cheung**
School of Computer Science
McGill University
Montreal, QC, Canada
{jad@cs, yulan.feng@mail, jcheung@cs}.mcgill.ca

Abstract

We examine the potential of recurrent neural networks for handling pragmatic inferences involving complex contextual cues for the task of article usage prediction. We train and compare several variants of Long Short-Term Memory (LSTM) networks with an attention mechanism. Our model outperforms a previous state-of-the-art system, achieving up to 96.63% accuracy on the WSJ/PTB corpus. In addition, we perform a series of analyses to understand the impact of various model choices. We find that the gain in performance can be attributed to the ability of LSTMs to pick up on contextual cues, both local and further away in distance, and that the model is able to solve cases involving reasoning about coreference and synonymy. We also show how the attention mechanism contributes to the interpretability of the model's effectiveness.

1 Introduction

Correctly performing pragmatic reasoning is at the core of many NLP tasks such as information extraction, automatic summarization, and machine translation. We focus in this paper on definiteness prediction, the task of determining whether a noun phrase should be definite or indefinite. In English, one instantiation of this task is to predict whether to use a definite article (*the*), indefinite article (*a(n)*), or no article at all. It has applications in machine translation (Heine, 1998; Netzer and Elhadad, 1998), and in L2 grammatical error detection and correction (Han et al., 2006).

Definiteness prediction is an interesting testbed for pragmatic reasoning, because both contextual and local cues are crucial to determining the acceptability of a particular choice of article. Consider the following example:

(1) *A/#the man entered the room. The/#a man turned on the TV.*

Factors such as discourse context, familiarity, and information status play a role in determining the choice of articles. Here, *man* is introduced into the discourse context by an indefinite article, then subsequently referred to by a definite article. On the other hand, non-context-dependent factors such as local syntactic and semantic restrictions may block the presence of an article. For example, demonstratives (e.g., *this*, *that*), certain quantifiers (e.g., *no*), and mass nouns (e.g., *money*) do not permit articles.

In this work, we investigate Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), a subclass of recurrent neural networks (RNNs) which have been popular recently in a variety of NLP tasks (Sutskever et al., 2014; Mikolov et al., 2010; Dyer et al., 2015). A number of reasons are often cited for the impressive performance gains of RNNs (Goldberg, 2015). First, they are able to take advantage of the patterns inherent in the data to learn features and representations suitable for complex interpretation tasks. Second, they can learn connections between processing units in the same layer, allowing the network to capture relations and patterns over an unbounded number of timesteps. Third, they provide an easy and natural way to inject external semantic knowledge by initializing the parameters of the model in an informed way. For example, the word embeddings can be initialized using pre-trained vectors such as `word2vec` (Mikolov et al., 2013).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2625–2634, Osaka, Japan, December 11-17 2016.

We compare several versions of LSTMs to explore how each of these factors affects article usage prediction. We also explore a version of LSTMs that employs an attention mechanism. The primary motivation for the use of an attention mechanism is to investigate whether such LSTM models focus on certain parts of the sentence when making predictions, and if so, to gain more insight into what parts of the sentence affect the model's prediction.

Our model achieves state-of-the-art performance on definiteness prediction, outperforming a previous, classification-based model by De Felice (2008). Our best model achieves 96.63% accuracy on the WSJ/PTB corpus, representing a relative error reduction of 51% compared to the previous state of the art. Each of the factors we examined (initializing with pre-trained vectors, giving more context, giving POS tags, attention) contributes to the performance of the model, though in different degrees. We perform a number of analyses to understand the behavior of the models, and show in particular how the attention mechanism can be useful for interpreting the model predictions.

The most interesting contribution of this paper is highlighting the suitability of LSTMs for tackling complex cases of article usage where there is no obvious local cue for prediction. We find evidence that LSTMs given an extended context window can resolve cases of article usage that seem to require reasoning about coreferent entities involving synonymy. Our results suggest that recurrent neural network models such as LSTMs are a promising approach to capturing pragmatic knowledge.

2 Related Work

Characterizing definite descriptions has been one of the first problems considered in semantics and pragmatics, and indeed in the philosophy of language. Russell (1905) analyzed definite descriptions as having quantificational force by asserting the existence and uniqueness of the definite NP, whereas Strawson (1950) emphasized their anaphoric nature, and their ability to trigger a presupposition about the existence of the noun phrase in the discourse context.

In terms of corpus-based studies, Poesio and Vieira (1998) analyzed a subset of definite descriptions found in the WSJ, and asked annotators to classify them according to their function. They found that 50% of definite descriptions were classified as discourse-new, 30% as anaphoric, and 18% as associative or bridging. Lee et al. (2009) investigated the role of contextual information in predicting article usage in a user study.

There has been a variety of previous models on article prediction (Knight and Chander, 1994; Minnen et al., 2000; Han et al., 2006; Gamon et al., 2008). De Felice (2008) framed it as a MaxEnt classification task, extracting a number of linguistically motivated features from the context of each head noun. Turner and Charniak (2007) took an approach more similar to our own, viewing article selection as a language modelling problem by training a parser-based language model on the WSJ and North American News Corpus.

More generally, the use of distributed representations in discourse processing is becoming more widespread, with applications in discourse parsing (Kalchbrenner and Blunsom, 2013; Ji and Eisenstein, 2014; Li et al., 2014) and implicit discourse relation detection (Ji and Eisenstein, 2015). There have also been recent efforts to build distributed representations of linguistic units larger than a sentence (Le and Mikolov, 2014; Li et al., 2015). Hill et al. (2016) investigated several variants of LSTMs to predict function and content words, but they did not consider determiners in their study.

3 Model

We introduce several variants of LSTM models for definiteness prediction. LSTMs extend standard RNNs by providing additional memory control that can capture long-term dependencies between data samples that would be difficult to capture with standard RNNs. The memory control allows the model to carefully regulate how much the current input affects the new memory state, how much the previous memory state affects the new memory state and what elements of the memory should play a role in generating the output. We first present the LSTM variants that we experimented with, then describe the input representations which we used for the models.

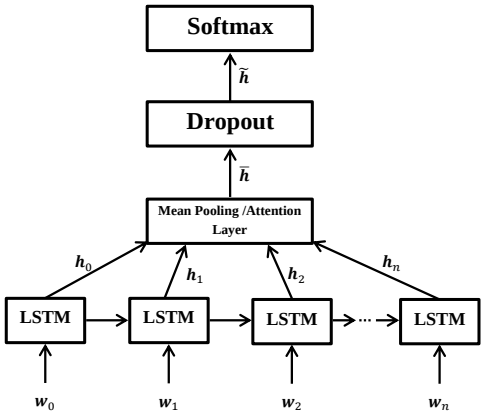


Figure 1: LSTM-based neural network model.

3.1 Vanilla Model

In a standard, "vanilla" LSTM (Figure 1), the model is given an input sequence, $w_0, w_1, \ldots, w_n$. Each w_i is a vector representation of an input word at timestep i, which is fed to an LSTM cell consisting of an input gate, an output gate, a forget gate, a cell state, and a hidden state (see (Graves, 2013) for more details). The outputs of the LSTM cells are fed to a mean pooling layer:

$$\overline{h} = \frac{1}{n} \sum_i h_i \tag{2}$$

Then, the dropout layer randomly suppresses output neurons of $\overline{h}$ during the forward pass of training with a pre-defined probability by setting them to zero. This in effect acts as a regularization mechanism (Srivastava et al., 2014). In the last stage, we perform classification using a three-input softmax unit.

3.2 Attention-based model

Attention mechanisms have recently attracted considerable interest in the deep learning community. Attention mechanisms are loosely inspired by theories of human visual attention in which specific regions of interest have high focus compared to other regions.

Compared to "vanilla" model, the attentive LSTM model replaces the mean pooling layer with a learned attention layer that leads to a weighted average of the timesteps. For each timestep i, the model learns:

$$c_i = \tanh(Mh_i + b) \tag{3}$$

$$a_i = \exp(c_i) / \sum_j \exp(c_j), \tag{4}$$

where M is a learned weight matrix, b is a bias term, c_i reflects the importance of the input at that timestep, and a_i is a normalized version of the importance over all timesteps in the sample. Then, the output is calculated as:

$$\overline{h} = \sum_i a_i h_i. \tag{5}$$

3.3 Input representation

We map each input token w_i to some learned embedding, which is then fed to the input layer of the LSTM. We investigate several different linguistically motivated factors for building this representation

Table 1: Number and distribution of noun phrases in the PTB corpus by the article type.

	none	the	a(n)
Training set	150606	49117	23907
Development set	11987	3898	1867
Test set	14676	4848	2155

beyond the standard "vanilla" LSTM model. First, we experiment with incorporating or not POS tags as a form of syntactic knowledge (+/−**POS**). To incorporate such knowledge, we concatenate the POS to the aforementioned learned embedding before feeding it to the input layer. The other factor that we consider is how to initialize the embedding associated with each word-type. We experiment with initializing it randomly, or with pre-trained vectors, which could inject knowledge derived from a large, external corpus. The intuition behind incorporating pre-trained vectors is that they might help the model recognize bridging references of those involving synonyms (e.g., *a* ***house*** *... the* ***home***). Thus, we compare the following options:

- **Random**: The embedding is initialized randomly.
- **word2vec**: The embedding is initialized by the SkipGram vectors of Mikolov et al. (2013) trained on the Google News corpus (about 100 billion words).
- **GloVe**: The embedding is initialized by the global vectors Pennington et al. (2014) that are trained on the Common Crawl corpus (840 billion tokens).

Both word2vec and GloVe word embeddings consist of 300 dimensions. To test whether compressing those embeddings would lead to a better prediction performance, we investigated the use of PCA to reduce the dimensionality of the word embeddings, but found that this did not influence performance on the development set.

4 Dataset

We use the Penn Treebank (PTB) corpus (Marcus et al., 1993) with the standard section splits for training (01–23), development (00, 24) and testing (22, 23). We extract all of the noun phrases present in the parsed corpus whose head noun's POS tag is one of `NN`, `NNS`, `NNP`, or `NNPS`. Also, we do not lemmatize, and ignore case and punctuation. Finally, we remove any occurrence of the relevant determiners (*the*, *a*, *an*) from all the data sets (training, development, test). The number of samples in the dataset is shown in Table 1.

In our experiments, we adopt one of two different sample configurations: the first focusing on a local context and the second extending the context to encompass one or more sentences. Specifically, for the former, we define a sample to be the set of tokens from the previous head noun of a noun phrase up to and including the head noun of the current noun phrase. For example, take the following passage (head nouns indicated in bold):

(6) For six **years**, T. Marshall Hahn **Jr.** has made corporate **acquisitions** in the George Bush **mode**: kind and gentle.

The following samples –relying on local context– are shown, with their labels: *For six years* – 'none', *T. Marshall Hahn Jr* – 'none', *has made corporate acquisitions* – 'none', *in George Bush mode* – 'the'.

For the sample configuration relying on the extended context, the sample is constructed in the same way (as described above) and, in addition, tokens from the previous sample(s) are added sequentially (in reverse) until a pre-specified total number of tokens per sample is reached. That way, the sample not only reflects local information from the current noun phrase, but also information that is contained in a previous sentence (or more).

Having these two distinct sample configurations allows us to compare, in general, the learning performance of the LSTM network between the two cases and to investigate, in particular, whether those networks are able to resolve cases that exhibit long range dependencies and complex cases that require contextual clues that go beyond the local context of a given article.

5 Experiments

We compare our LSTM models against the following systems. The first is the most frequent baseline (**Baseline**), which labels all noun phrases with the most frequent class of none. The second is a reimplementation of the classification-based method of De Felice (2008) (**LogReg**), which extracts features from a fixed context window for a multinomial logistic regression classifier. Our implementation differs from the one described in that work in two respects. First, we could not gain access to the list of mass and count nouns that she did. We approximated this by crawling the British National Corpus (Burnard, 2007) for instances of nouns that appear after *a/many/few* to identify count nouns, and *much/little/bit of* for mass nouns. During feature extraction on the PTB, nouns that have not been previously encountered are given the label unknown.

5.1 LSTM training

Many variations of the LSTM cell have been proposed in the literature; however, since we implement our model using Lasagne[1], we use the LSTM cell implemented by Lasagne and presented in (Graves, 2013). We use the AdaDelta algorithm (Zeiler, 2012) for learning the optimal network parameters and word embeddings. The model has a number of hyperparameters that were tuned on a development set. We list below the range that we explored (min, max) and the final value that was used [val]:

- Word embedding and hidden-layer dimensionality: (50, 200), [100]
- Dropout probability: (0.2, 0.7), [0.6]
- Minibatch size: (20, 200), [100]

6 Results

We present the results of our experiments in Table 2. Our LSTM models outperform LogReg and the baseline in terms of accuracy despite not having access to an extensive set of hand-crafted linguistic features. Incorporating pre-trained word vectors into the initialization results in a small but consistent improvement in performance, for both the version without and with POS tags. We notice, however, that the particular choice of the embedding, does not seem to make much of a difference, with GloVe very slightly outperforming word2vec. In all cases, using an attention mechanism in the learning model led to an improvement in the accuracy. We also contrast the learning performance of the network when fed samples relying on local context versus samples relying on the extended context. Most interestingly, POS tags do not seem as necessary for high performance in the extended case, as the discrepancy between using or not POS tags is almost negligible. This could be explained by the fact that an extended context allows the network to learn relevant syntactic cues from context which were only available with explicit POS tags in the case of local context. Overall, the best setting is using extended contexts and GloVe embeddings with POS tags and attention in the LSTM model, at 96.63% accuracy.

Our results improve upon different previously reported accuracies on this task. De Felice (2008) reported the best previous result of 92.15% accuracy, though this was on the BNC corpus. Our reimplementation of this work (LogReg) achieved 93.07 % on the WSJ-PTB corpus. Turner and Charniak (2007) reported an accuracy of 86.63% on ten-fold cross-validation over WSJ with additional training on an external corpus. We are able to achieve a higher accuracy on comparable data using a smaller amount of training data, though the models initialized with pre-trained word embeddings could arguably be said to be trained on large amounts of text.

[1] `http://lasagne.readthedocs.org/`

Method			**Accuracy** (%)	
Baseline			67.70	
LogReg			93.07	
	Init.	**POS**	**Local contexts**	**Extended contexts**
LSTM	Random	−POS	83.94 - 83.96	95.82 - 96.08
LSTM	word2vec	−POS	84.91 - 84.93	96.40 - 96.53
LSTM	GloVe	−POS	85.35 - 85.75	96.37 - 96.43
LSTM	Random	+POS	94.11 - 94.12	95.95 - 96.08
LSTM	word2vec	+POS	94.50 - 94.52	96.20 - 96.25
LSTM	GloVe	+POS	94.64 - 94.67	96.38 - **96.63**

Table 2: Accuracy results for article prediction on the WSJ/PTB corpus. The LSTM models are distinguished by their initialization method (**Init.**), whether or not they used POS tags (**POS**), and whether they were give local or extended context. For each result for the LSTM models, we show the result for the model without attention (left) and with attention (right).

Method	**Class**	**P**	**R**	**F1**
Baseline	none	67.70	100.0	80.74
LogReg	a	75.05	70.49	72.70
	none	98.63	98.41	98.52
	the	84.10	86.94	85.50
Local LSTM+a	a	76.30	73.04	74.63
+ GloVe	none	99.55	99.42	99.49
+ POS	the	87.60	89.60	88.59
Extended LSTM+a	a	86.88	91.28	89.02
+ GloVe	none	98.02	96.99	97.50
+ POS	the	95.34	96.25	95.79

Table 3: Precision, Recall, and F1 results by class. We only show the 'none' class results of the baseline, as the baseline is to label everything as 'none'. LSTM+a represents the attention-based model.

These numbers conceal the large differences in performance between the different classes. Table 3 shows the breakdown of the results for each model by class label in terms of precision, recall, and F1. Because of space limitations, we only include results for selected models. Overall, the "none" class is the easiest to predict. This is expected, as various syntactic and semantic cues are highly indicative of the "none" class (e.g., presence of a demonstrative in the context or the head noun being a named entity).

Also, we consider the performance of the models on named entities versus non-named entities, as identified by the POS tag on the head noun. Table 4 shows the accuracy results divided into the two classes. As expected, named entities are easier for the model to predict, due to conventions about article usage related to named entities in English. The LogReg model already achieves high performance on named entities, and it is conceivable that with a larger amount of training data, it can approach the performance of the LSTM models, because it will see more conventions about named entities or classes of named entities. Both LSTM models improve on performance on both classes, and actually obtain a greater absolute improvement on the non-named entities. This could be seen as evidence that they actually are making better predictions for those cases that require long-range dependencies.

Finally, we conducted two-tailed paired sign tests (with a level of significance $\alpha = 0.05$) to examine whether the best performing LSTM model (GloVe + POS with attention and extended context) significantly outperformed the LogReg baseline, as well as other versions of the LSTM model, namely the best LSTM with local context, the best LSTM with random initialization and the best LSTM without

Method	NE	non-NE
Baseline	86.98	61.76
LogReg	97.27	91.77
Local LSTM+a + GloVe + POS	**98.88**	93.44
Extended LSTM+a + GloVe + POS	97.62	**96.48**

Table 4: Test set accuracy results for named entities (N = 5100) and non-named entities (N = 16579).

	Simple Cases		Complex Cases	
	fixed	dup.	syn.	semantics
	86	6	8	100
Total	92		108	

Table 5: Classification of 200 samples incorrectly predicted by the best performing LSTM model on the local context but correctly predicted by the best performing model on the extended context. With categories of simple cases including fixed expressions(*fixed*) and duplication of the head noun(*dup.*), complex cases of synonyms(*syn.*), and cases require semantic understanding(*semantics*)

POS tags. We found that, for the single case of the best LSTM model without POS tags, the difference between that LSTM model and the best performing LSTM model was not statistically significant (p= 0.41). For all the remaining models, we found a highly significant difference ($p < 10^{-6}$) between the best performing LSTM model and each of those models.

7 Analysis

We perform some additional analysis to understand the behavior of the models.

7.1 Local context versus extended context

In order to gain an insight into the possible reasons behind the large performance gains obtained when using the extended context, we compare the best performing LSTM model that uses local context to the best model relying on the extended context and investigate 200 samples out of the 957 samples that were incorrectly predicted by the former (local context) but correctly predicted by the latter (extended context).

We grouped the samples into two main categories: (1) simple cases where the decision can be made based on the noun phrase itself (e.g. fixed expressions such as "the other day", and named entities), or the same head noun was introduced in the earlier discourse context; (2) complex cases where contextual knowledge involving pragmatic reasoning is required (e.g., entity coreference involving synonymy, bridging reference).

Table 5 shows the break-down into those categories. Complex cases accounted for over half of the cases with 108 cases involving synonymy or other complex cases such as bridging references. The model using local context failed to predict those cases correctly, which can be explained by the fact that, with local context only, no obvious cues for predictions were available. This suggests that LSTM networks constitute learning models that can learn to predict complex cases given an appropriate contextual window.

7.2 Qualitative analysis using the attention weights

In order to gain a better understanding of how the model makes its predictions and whether the model is able to resolve complex cases, we consider the attention weights of several samples (i.e., the set of a_i weights). All of the following cases were incorrectly predicted using local context but correctly predicted when the LSTM network was fed samples with extended context. Note although the original sentences (shown below) include the determiners (for the classes 'a', and 'the'), the determiners are removed from

the samples that are fed to the neural network model as mentioned in Section 4. The samples have a fixed length of 50 tokens. To emphasize the article that is the focus of the prediction task in that particular sample, we present such article in bold in each of the examples. Finally, note that since a sample consists of 50 tokens, the average weight for a token is 0.02.

Consider the example:

(7) *... net income for the third quarter of 16.8 million or 41 cents a share reflecting [a] broad-based improvement in the company's core businesses. Retail profit surged but the company it was only a modest contributor to third-quarter results. A year ago, net, at* ***the*** *New York investment banking firm ...*

In this example, in addition to the tokens in the noun phrase "New York investment banking firm" receiving some of the highest weights, both "contributor" and "company" were among the 10 tokens with the highest weight (with all of them receiving a weight of more than 0.04). While the LSTM with local context incorrectly predicted such a sample, high weights on tokens such as "contributor" and "company" suggest that the LSTM had potentially made use of the extended context paying higher "attention" to those relevant tokens while mostly ignoring the contents of the rest of the sentence.

In the following example:

(8) *...companies. In a possible prelude to the resumption of talks between Boeing Co. and striking Machinists union members, a federal mediator said representatives of the two sides will meet with him tomorrow. It could be a long meeting or it could be a short one, said Doug Hammond,* ***the*** *mediator...*

in addition to "mediator" receiving the highest weight (approx. 0.05), both "Doug" and "Hammond" received weights of approx. 0.049 and 0.05. While resolving the case of "the mediator", the network correctly paid attention to the relevant tokens "Doug" and "Hammond".

Consider this example:

(9) *BMA's investment banker Alex Brown & Sons Inc. has been authorized to contact possible buyers for the unit. Laidlaw Transportation Ltd. said it raised its stake in ADT Ltd. of Bermuda to 29.4% from 28%. A spokesman for Laidlaw declined to disclose the price* ***the*** *Toronto transportation...*

In this example, for correctly predicting that "Toronto transportation" should have a label "the", the network focused on the tokens "Toronto" and "transportation" while also attributing high weights to the tokens in "Laidlaw Transportation Ltd." (with all of them receiving a weight of more than 0.04 and figuring in the 10 highest weights).

These samples demonstrate that the attention mechanism can give us useful feedback on why the model is making the predictions that it is, adding interpretability to deep models. Secondly, they provide evidence that RNNs are able to learn complex features in order to place importance to syntactically and semantically relevant cues for this pragmatic reasoning task. By potentially allocating high weights to relevant tokens in an extended context, the LSTM network could also learn to predict complex cases of article usage, explaining its performance gains.

8 Conclusion

We have shown that English article usage, a task requiring reasoning about both local and non-local cues, can be successfully predicted using an LSTM recurrent neural network. Despite using generic features, our model outperforms previous methods for article prediction that rely on limited context. Our model is very successful in predicting the 'none' class and in making predictions for named entities. We perform a series of analyses to investigate whether the performance gains are due to factors that are traditionally cited for RNN models. By examining the attention weights, we find evidence that the models are actually learning complex features such as various syntactic and semantic restrictions, which would have been encoded by manually constructed features in previous models. As for initializing with pre-trained word embeddings, this improves performance consistently across multiple models but only marginally. The

LSTM can also take advantage of long-ranged dependencies in making its prediction, because it can place high attention on any part of the input sequence from an arbitrary distance before the head noun. We also have shown that LSTM networks show strong performance in resolving complex semantic cases when given an extended contextual window spanning two or more sentences.

Our model does not rely on task-specific features, only task-specific training. Thus, the exact same model should be applicable to other tasks involving semantic and pragmatic knowledge. We are currently planning to conduct further experiments on predicting other linguistic constructions involving contextual awareness and presupposition.

References

Lou Burnard. 2007. Reference guide for the british national corpus (XML edition).

Rachele De Felice. 2008. *Automatic error detection in non-native English*. Ph.D. thesis, University of Oxford.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China, July. Association for Computational Linguistics.

Michael Gamon, Jianfeng Gao, Chris Brockett, Alexandre Klementiev, William B. Dolan, Dmitriy Belenko, and Lucy Vanderwende. 2008. Using contextual speller techniques and language modeling for ESL error correction. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I*, pages 449–456.

Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *arXiv preprint arXiv:1510.00726*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting errors in english article usage by non-native speakers. *Natural Language Engineering*, 12(2):115–129.

Julia E. Heine. 1998. Definiteness predictions for japanese noun phrases. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 519–525. Association for Computational Linguistics.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of the 2016 International Conference on Learning Representations*, January.

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

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24, Baltimore, Maryland, June. Association for Computational Linguistics.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-Augmented distributed semantics for discourse relations. *Transactions of the Association for Computational Linguistics*, 3:329–344.

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

Kevin Knight and Ishwar Chander. 1994. Automated postediting of documents. In *AAAI*, volume 94, pages 779–784.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1188–1196.

John Lee, Joel Tetreault, and Martin Chodorow. 2009. Human evaluation of article and noun number usage: Influences of context and construction variability. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 60–63. Association for Computational Linguistics.

Jiwei Li, Rumeng Li, and Eduard H. Hovy. 2014. Recursive deep models for discourse parsing. In *EMNLP*, pages 2061–2069.

Jiwei Li, Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1106–1115, Beijing, China, July. Association for Computational Linguistics.

Mitchell P. Marcus, Mary A. Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH 2010*, volume 2, page 3.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Guido Minnen, Francis Bond, and Ann Copestake. 2000. Memory-based learning for article generation. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 43–48. Association for Computational Linguistics.

Yael D. Netzer and Michael Elhadad. 1998. Generating determiners and quantifiers in hebrew. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 89–96. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Massimo Poesio and Renata Vieira. 1998. A corpus-based investigation of definite description use. *Computational Linguistics*, 24(2):183–216.

Bertrand Russell. 1905. On denoting. *Mind*, pages 479–493.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Peter F. Strawson. 1950. On referring. *Mind*, 59(235):320–344.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Jenine Turner and Eugene Charniak. 2007. Language modeling for determiner selection. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 177–180. Association for Computational Linguistics.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Aspect Based Sentiment Analysis using Sentiment Flow with Local and Non-local Neighbor Information

Shubham Pateria
Samsung R&D Institute
Bangalore, India
s.pateria@samsung.com

Abstract

Aspect-level analysis of sentiments contained in a review text is important to reveal a detailed picture of consumer opinions. While a plethora of methods have been traditionally employed for this task, majority focus has been on analyzing only aspect-centered local information. However, incorporating information from non-local neighbor aspects may capture richer context and enhance sentiment prediction. This may especially be helpful to resolve poor prediction due to ambiguities in review text. The context around an aspect can be incorporated using semantic relations within text and inter-label dependencies in the output. On the output side, this becomes a structured prediction task. However, non-local label correlations are computationally heavy and intractable to infer for structured prediction models like Conditional Random Fields (CRF). Moreover, some prior intuition is required to incorporate non-local context. Thus, inspired by previous research on multi-stage prediction[1], we propose a two-level model for aspect-based analysis. The proposed model uses predicted probability estimates from first level to incorporate neighbor information in the second level. The model is evaluated on data taken from SemEval Workshops and Bing Liu's review collection. It shows comparatively better performance against few existing methods. Overall, we get prediction accuracy in a range of 83-88% and almost 3-4 point increment against baseline (first level only) scores.

1 Introduction

The voice of consumer is growing stronger. With numerous platforms now available for providing reviews, consumers find it easy to share their opinions and sentiments about a product, service or other subjects. Thus, it becomes essential to analyze such reviews in order to identify consumers preferences and grievances. Sentiment analysis for consumer reviews (or general text) is a prominent research area. Such analysis can be done on various levels - global (collection of text), sentence-level (where sentiment is assigned to one full sentence) or aspect-based. In Aspect Based Sentiment Analysis (ABSA), the problem of interest is to estimate sentiment associated with a specific aspect within a review. An example is given below,

Example 1. The **movie** had a $brilliant_+$ story. The **location** was $awesome_+$ and I must highly $praise_+$ the **camera-work**. However, I have to differ regarding the **acting**. It is hard to comprehend XYZ's **style** in such a role. Her on-screen **presence** is not the usual; I have to say, her **act** left me in a very bad_- mood. The rest of the **cast** was $ok_o ... average_o$ at best.

Here, the text in bold marks aspect and italic text marks sentiment-indicators. Also, (+,-,o) underscore notations indicate positive, negative and neutral sentiments, respectively. Usually, there can be one or several aspects within a single sentence. The sentiment associated with any aspect can mostly be inferred by checking the terms associated with it (e.g., *awesome* - **location**). However, this may not be very

[1] (Krishnan and Manning, 2006; Hoefel and Elkan, 2008)

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2635–2646, Osaka, Japan, December 11-17 2016.

beneficial if the statement is ambiguous. For e.g., *'I have to differ regarding the acting'* does not clearly indicate any sentiment on its own. Such ambiguities may be found frequently in reviews. Due to varied styles of different review writers, use of uncommon (even obscure) terms or phrases, terms or phrases conveying conflicting sentiments, or even due to limited data, a prediction system may be expected to mis-classify sentiments. One way to identify such ambiguities is by using prediction confidence scores (discussed in Section 4.2).

While addressing ambiguities, it may be assumed that there generally is some inherent *flow* in the sentiments. A review may have elements of discourse, such that, discourse-markers[2] like 'and', 'also', 'but' etc. can be used to identify sentiment flow or transition. Such terms may or may not be explicitly used, but presence of flow can be assumed. This idea of flow is not new. Recently, *Sentiment Flows* have been studied by Wachsmuth et al. (2015). They incorporated flow information while predicting global sentiments and also identified some frequent types of flows (Wachsmuth et al., 2014). Analysis using sentiment flow requires neighbor information. Here, a neighbor can be local or non-local. In Example 1, initial sentiment flow is positive. Then, the flow is broken by *However* and the sentences that follow are ambiguous. Towards the end, *very bad mood* sets a negative polarity towards **act**. For multi-class classification, it is difficult to predict sentiment associated with **acting** just from information of its local neighbors (**style** and **camera-work**). It is important to incorporate distant aspect **act**'s sentiment to approximately predict the negative shift in mood. This task can become more complex with more involved semantics (such as in reviews by expert critics).

The neighbor-dependencies can be holistically modeled by also considering correlations among polarity labels, thus making ABSA a structured prediction task. Previously, modified version of Condition Random Field (CRF) classifier has been proposed to predict local sentiments (Mao and Lebanon, 2006). However, while CRFs perform well for local dependencies, they may not be very suitable in standard form for ABSA after we consider non-local neighbors as well because inference over long-range would be expensive (Kazama and Torisawa, 2007; Krishnan and Manning, 2006). Moreover, we believe it would also be beneficial to incorporate textual terms surrounding local and non-local neighbors as input features. This would be difficult without some pre-intuition about non-local neighbor sentiments, lest the the input representation itself becomes complex.

To address these issues, we propose a two-level model for ABSA. The proposed model first performs classification using a baseline set of features. Based on this, the probability estimates are obtained which give indication about ambiguities, as well as, preliminary information about neighbor sentiments. Another classifier on top of this uses the local and non-local neighbor information (first-stage probabilities as well as textual terms) for prediction. In this paper, we discuss a preliminary work on this model. The rest of the paper is organized as follows. The structure used for internal representation of reviews is discussed in Section 3. The classification model using SVM classifier in both stages is discussed in Section 4. Further, in order to test a linear-chain CRF at second stage of our model, an independent experiment is performed using available CRF software. The CRF experiment uses different setup from SVM+SVM model and thus it is not meant for comparison with SVMs, but for independent evaluation. This is briefly discussed in sub-section 4.5 . An evaluation of the models is discussed in Section 5.

2 Related Works in ABSA

Three major steps in ABSA are aspect-term extraction, category detection and polarity estimation. There have been significant amount of work in these areas. Major work related to ABSA has appeared in SemEval Workshops[3]. Some notable contributions in these workshops discuss good practical methods for aspect and category extraction (Brun et al., 2016; Khalil et al., 2016; Toh et al., 2016; Saias, 2015). A lot of work has been done on aspect extraction, however, we would like to focus our discussion on sentiment prediction. The basic form of sentiment analysis at sentence-level or aspect-level uses local context of an aspect for input feature representation. Some notable works include that by Nakagawa et al. (2010) who use dependency-tree structures to model local word interactions; Choi and Cardie (2008)

[2]https://en.wikipedia.org/wiki/Discourse_marker

[3]http://alt.qcri.org/semeval2016/task5/, http://alt.qcri.org/semeval2015/task12/

apply inference rules for polarity reversals. Moreover, deep learning methods have also been explored for aspect-level (Wang et al., 2015) and sentence-level (Socher et al., 2013) analysis by exploiting vector representations of aspect-related terms. Discourse information has been very much favored to expand the context around sentiment targets. Discourse-based analysis has been profoundly covered in some previous works (Somasundaran et al., 2009a; Somasundaran et al., 2008; Somasundaran et al., 2009b; Mukherjee et al., 2012). These works cover different types of discourse relations, in detail, for sentiment analysis. Polanyi and Zaenen (2006) discuss valence-shift over sentences due to discourse markers. It is also natural to consider discourse for sentiment flow, as will be discussed later in this paper. Discourse has been used to model neighbor relations as well. Pang and Lee (2004) have explored the consistency of sentiment between neighbors. Also, Zhou et al. (2011) use the sentiment consistency or contrast as constraint on polarity assigned to neighbors. Lazaridou et al. (2013) encoded discourse relations into their supervised classifier's input features. Similar techniques are used in our model, however, for both local and non-local context. Apart from relational structures within text, it is also beneficial to model correlation among polarity labels. An important work in this direction is by Mao and Lebnon (2006) who introduced a modified CRF model to predict ordinal polarity labels. Wachsmuth et al. (2015) follow this work and discuss sentiment flow adaptability across domains. They also discuss ideas to constraint the label sequence lengths. The grouping method discussed later in this paper is inspired by their work. However, these works are still constrained to correlations between adjacent labels only. Some of the notable and relevant work exploring long distance (non-local) information are: work by Somasundaram et al. (2008) on opinion target relations and its application as constraint on predicted sentiments; work by Zhang et al. (2013) on using discourse relations as constraints for Markov Logic Network; and of special interest is the work by Yang and Cardie (2014) who incorporate discourse and coreference constraints into Posterior Regularization (PR) of a CRF. The works discussed above use some form of discourse or coreference relations for feature embedding or inference constraints. However, while we use discourse markers to separate aspect context, we do not restrict neighbor feature embedding based on discourse only. Instead, we propose a two-level model with a base level prediction from which a probability distribution over sentiment labels can be obtained. This serves as an intermediate intuition about sentiments. Using this, at second level, we sample important non-local neighbor units to embed non-local context into input features. Thus, such sampling is not necessarily restricted to presence of coreference or discourse. However, following the work of Yang and Cardie (2014), it would be interesting to explore the use of CRF at second level with PR constraints involving base level probabilities.

3 Review Representation

In this section, we discuss the structure of a review in the form of a graph of aspect-centered nodes.

3.1 Review Structure

A review can be modeled as a non-directed graph of connected aspects and sentiment nodes. The graph is depicted in Figure 1. For simplicity, the sentiment values are not shown as separate nodes but included within the aspect nodes. We define following attributes of the review graph:

- Aspect-Units (or Units): A Unit *U* is a node in review graph, and the basic entity which bundles the parameters associated with an aspect. Formation of units is discussed in Section 3.2
  U : (*aspect-terms*; *related-terms*; *sentiment information*)

- Groups: A group *G* is a cluster of continuous units with similar sentiment. In Example 1, under the assumption of sentiment flow, "The movie had a $brilliant_+$ story", "The location was $awesome_+$" and "I must highly $praise_+$ the camera-work" can be grouped under positive sentiment category. Similar grouping applies to other sentences or phrases as well. Consider the polarity sequence in Example 1, PP**P**-XXX-**NO**O, where P is positive label, N is negative, O is neutral and X ambiguous. After the grouping mentioned above, we consider that labels marked in bold can represent the polarity of a group of same sentiments. Henceforth, these will be referred to as *terminal labels*. A group *G* contains information about the cluster of units as,

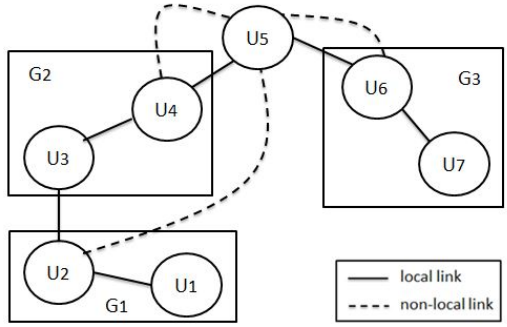


Figure 1: Review Structure around unit U_5.

Type	Markers
Additive	and, or, also, therefore, furthermore, consequently, thus, as a result, hence, subsequently, eventually, in addition, additionally, moreover, as well as
Contrast	though, although, however, but, despite, yet, still, nonetheless, nevertheless, in spite

Figure 2: Common discourse markers with shallow categorization.

G (*polarity*; *count*; *distance*)
Thus, polarity sequence in Example 1 can be reduced to this form as G(P;3;1)-G(X;3;0)-G(N;1;1)-G(O;2;2). Here, distance is taken from ambiguous group.

- Links: A Link *L* is a connection between two units or groups or a unit and a group. A link may or may not contain information about the sentiment flow. Three types of links are used:
  L{+} : Additive links. Terms like 'and', 'Moreover', 'Also' etc. make L{+} links.
  L{c} : Contrast links. Terms like 'but', 'however' etc. make L{c} links.
  L{x} : Undefined links, where a clear connection between two units may not exist. However, sentiment flow (or transition) can still exist. For instance, in Example 1, there no clear link between "**act** left me in a very bad_{-} mood" and "The rest of the **cast** was ok_{o}" This would be considered as L{x}.

3.2 Aspect Unit Formation

In order to form a unit, the terms in a review related to an aspect are extracted using parse relations. We use Stanford Parser (Marneffe et al., 2006) for this purpose. Consider following sentence : *The rice*

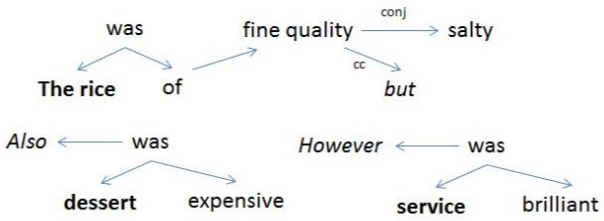


Figure 3: Approx. parsing.

was of fine quality but very salty. Also, dessert was a bit expensive. However, service was brilliant. A selective approximate parsing is shown in Figure 3. Here. *but* is an internal connector. *Also* and *However* are terminal connectors. Using the dependency relations, the units can be formed as follows: U_1 : {The rice was of fine quality but very salty}, U_2 : {dessert was a bit expensive}, U_3 : {service was brilliant}, with connections as: U_1–**(Also)**–U_2–**(However)**–U_3. *Also* and *However* become part of the Links.

4 Classification Model

The overall sentiment prediction process is divided into 2 stages: Base Prediction and Level-2 Prediction. Consider a feature set described for each aspect unit as $\phi(U_i)$ = (ϕ_1, ϕ_2...ϕ_m), where U_i is the i-th aspect unit in a sequence and any ϕ_j is a feature. Thus, input consists of a set ϕ = { $\phi(U_1)$, $\phi(U_2)$,...,$\phi(U_N)$

}. The aim of first stage or Level-1 prediction is to obtain an intermediate set of sentiment labels $\hat{P}$ = { $\hat{P}(U_1)$, $\hat{P}(U_2)$...$\hat{P}(U_N)$ } along with probability estimates for each. These predictions are used to form groups and sample neighbor information to be added to each input vector $\phi(U_i)$ for final prediction. We discuss our model with-respect-to 3 class classification (positive, negative and neutral) below.

4.1 Baseline Features

These are aspect-centered features formed using text surrounding a given aspect term(s).

Sentiment Scores : The scores of sentiment-indicator terms are aggregated into feature sets,
$\phi_{1k}(U_i) : (score_{positive}, score_{negative}, score_{neutral})$

Here, *k* indicates *kth* type of score-set. We use five score-set types (3 from lexicon corpus + 1 keyword-based + 1 neutral terms) as discussed below:

Sentiment Lexicons from external corpus: Bing Liu's lexicons (Bing Liu, 2012), MPQA subjectivity clues (Wiebe et al., 2005) and SentiWordnet (Stefano et al., 2010) lexicon corpus are used to obtain scores. Bing Liu's and MPQA corpus provide binary scores (positive: 1, negative: 0). These are used as binary features. SentiWordnet provides a range of scores for positive and negative categories.

Category Keywords: Apart from lexicons available in the external corpus, there may be terms which convey sentiments relative to categories. For e.g., *the acting was cheap* conveys negative sentiment while *the price was cheap* is positive despite the same term *cheap*. Such keywords are extracted by dividing the review data into category-specific documents and obtaining TF-IDF scores to identify frequent keywords and corresponding sentiment types. Frequency thresholds of min:0.3 & max:0.8 are set, based on best performance in our experiment.

Neutral terms: Several terms which occur in neutral sentences are not scored in external corpus. These are extracted by identifying frequent words or bi-grams in a collection of neutral sentences. The most frequent ones used in this paper are: 'average', 'normal', 'simple', 'okay', 'ok', 'not great', 'nothing great', 'mediocre', 'not good', 'decent', 'as expected', 'reasonable', 'moderate', 'typical', 'alright', 'fair'.

The score assigned to each sentiment-indicator is also subject to negation. In case of negation, binary scores are simply reversed. For SentiWordnet scores, negation is made in proportion to the scores as: pos = pos + $\frac{(neg-pos)}{2}$ and neg = neg + $\frac{(pos-neg)}{2}$. Here, pos and neg are positive and negative scores, respectively. A significant work on negation problem has been done by Zhu et al. (2014).Moreover, a unit may contain multiple sentiment terms. Thus, the scores are aggregated and normalized. However, as discussed in Section 3.2, the terms within an aspect-unit may be connected by discourse markers. The simplest strategy that can be used is to weigh the sentiment-indicators' scores according to their position in a unit and their relation with discourse markers (Mukherjee et al., 2012).

Bi-grams formed using terms in a unit. Bi-grams around negation terms are taken separately.
$\phi_{21}(U_i)$: *(bi-grams around negation)*, $\phi_{22}(U_i)$: *(other bi-grams)*,

A binary feature for **aspect-category** type can be used (if category has been extracted). This feature has minor effect (Table 3).
$\phi_3(U_i) : (category\ type)$

Local Context window: sentiment score features from previous and next aspect units.
$\phi_4(U_i)$: $\{\phi_{11}...\phi_{1n}\}(U_{i-1})$ and $\{\phi_{11}...\phi_{1n}\}(U_{i+1})$ (n=5 in this case)

4.2 Level-1 or Base Prediction Model

Base prediction is performed using feature set $\{\phi_{11}...\phi_{1n}, \phi_{21}, \phi_{22}, \phi_3, \phi_4\}$ (n=5 in this case). The distribution of features is non-linear as well as high-dimensional and Support Vector Machine (SVM) classifier with Radial Basis Function (RBF)[4] kernel is well-suited for this task due to its high-dimensional mapping and good margin. We use classifier from scikit-learn SVM (SVC) package[5]. A set of primary prediction labels $\hat{P}$ = { $\hat{P}(U_1)$, $\hat{P}(U_2)...\hat{P}(U_N)$ } (N = no. of aspects in a single review) is obtained from the base model using confidence scores over the *c* classes (c=3 in this case).

Confidence Scores or Probabilities

The scikit-learn SVM package provides two methods to obtain such scores[6]. First is the *predict_proba* function which provides probability distribution over different classes based on multi-class variant for Platt Scaling (Wu et al., 2004). Second is the *decision_function* which indicates the distance of input points from the hyperplane (or decision boundary). The prediction method (*predict*) of SVM uses *decision_function*. Platt Scaling based estimation may cause disagreement between outcome of *predict* function and the obtained $\arg\max$ (*predict_proba*). However, in the experiments, we found that $\arg\max$ (*predict_proba*) always corresponds to true class label when prediction is strong (high probability assigned to one class). When the classifier fails, the probability values across different classes have small separation (section 4.3). Thus, we stick to Platt Scaling (*predict_proba*) and obtain following:

proba(U_i) : $\{l_1, l_2, ..., l_c\}$, gives probability distribution (summing to 1) over c classes, such that,

$\hat{P}(U_i) = \arg\max$ (*proba*(U_i))

4.3 Ambiguity Criteria

As discussed in Section 4.2, *proba* values are used as confidence scores for base predictions. The ambiguous units are identified using *proba* by detecting low difference between any two probability values (low confidence). Following criteria is used for detection,

$\forall$ (l_q, l_r) $\in$ *proba*(U_i), where q $\neq$ r,
  if $|l_q - l_r| \leq$ *T1* **and** ($l_q >$ *T2* **or** $l_r >$ *T2*) **then**
    U_i is **ambiguous**

In our experiment with 3-class classification, we set *T1* = 0.20 and *T2* = 0.33 based on observations made on available data.[7]

4.4 Level-2 Prediction Model

Having obtained *proba* values, the final requirement is to predict polarity label set P = {$P(U_1)$, $P(U_2)...P(U_N)$}. The process of Level-2 model training and prediction are discussed below.

Training

Firstly, in order to incorporate local and non-local neighbor information, the neighbor units are grouped as described in Figure 4. Assignment of ambiguous unit's sentiment to a group is avoided here. This ensures that neighbor information consists of high confidence values during final prediction stage. After grouping, feature set $\mathbf{F}(U_i)$ is produced for a unit U_i as follows:

f_1 : list of *G*(polarity) values for max. 3 groups before unit, f_2 : list of *G*(polarity) values for max. 3 groups after unit. If unit U_i lies within *kth* group G_k, then the group is temporarily divided into

[4]http://research.cs.tamu.edu/prism/lectures/pr/pr_l19.pdf

[5]http://scikitlearn.org/stable/modules/generated/sklearn.svm.SVC.html

[6]http://scikit-learn.org/stable/modules/svm.html#scores-probabilities

[7]Since, three polarity classes are used, a homogeneous distribution will allot probability close to 0.33 to each. If either l_q or l_r has value greater than 0.33 (*T2*), assuming third value to be 0.33, then a l_q value of 0.53 will make l_r 0.13. In this case, l_q can be chosen as non-ambiguous and the difference between l_q and third value would be 0.20 (0.53-0.33), set as *T1*.

For m aspects in a review (training sample)
$k \leftarrow 1$
for i $\leftarrow$ 1 to *m* do
  if i = 1 **then**
    add U_i to G_k
    G_k{polarity} $\leftarrow$ {$\hat{P}(U_i)$, *proba*(U_i)}; G_k{count} $\leftarrow$ 1
  else if $\hat{P}(U_i) = \hat{P}(U_{i-1})$ **or** $\hat{P}(U_i)$ is *ambiguous* **then**
    add U_i to G_k
    G_k{count} $\leftarrow$ G_k{count} + 1 (0.5 if U_i **is** ambiguous)
  else
    $k \leftarrow k+1$
    add U_i to G_k
    G_k{polarity} $\leftarrow$ {$\hat{P}(U_i)$, *proba*(U_i)}
    G_k{count} $\leftarrow$ 1

Figure 4: Group formation using level-1 predictions.

$\{G_{k1}, U_i, G_{k2}\}$

f_3 : list of *G*(count) values for max. 3 groups before unit, f_4 : list of *G*(count) values for max. 3 groups after unit, f_5 : list of distances of max. 3 groups before unit, f_6 : list of distances of max. 3 groups after unit. The orders for these are maintained as per f_1 and f_2.

f_7 : link (type) between U_i and immediately previous group, f_8 : link (type) between U_i and immediately next group,

f_9 : Local feature set $\{\phi_{11}...\phi_{1n}, \phi_{21}, \phi_{22}, \phi_3, \phi_4\}(U_i)$, with ϕ_4 modified as

$\phi_4 = \{\phi_{11}...\phi_{1n}, \phi_{21}, \phi_{22}, \phi_3\}(U_{Terminal})$,

embedding the features of *terminal units* of max. 3 groups before and max. 3 after.

$f_{10} : \alpha(U_i)$, where

$$\alpha(U_i) = \begin{cases} 0 & , if\ U_i\ is\ ambiguous \\ argmax\ (proba(U_i))\ +\ 1 & , otherwise \end{cases}$$

These features are used to train a SVM classifier.

Prediction

The prediction on new data (or evaluation data) is made in a sequential manner (one-by-one). The $\hat{P}$ and *proba* are already available at this stage.

The final output is the required polarity set *P* = { $P(U_1)$, $P(U_2)$...$P(U_N)$}.

4.5 Experiment with CRF

In this paper, we have focused on method to incorporate non-local context information into input representation of aspect units. However, such method makes i.i.d assumption for output labels. Under sentiment flow property, there must be correlations between polarity labels as well, both adjacent and long-distance. Devising an efficient structured prediction model using long-distance dependencies is beyond scope of current work and is kept for future. Instead, we experiment with simple linear-chain CRF to get a glimpse into its performance on review text. CRFSUITE (Okazaki, 2007) is used to build a CRF classifier in python. This software provides an internal implementation of linear-chain (first-order Markov) CRF (Sutton and McCallum, 2010). This classifier is used at Level-2 of our model instead of SVM. However, SVM is preferred as base classifier (Level-1) due to its maximum-margin advantage (Hoefel and Elkan, 2008). For CRF (Level-2), features f_1 to f_{10} are used. However, prediction is made over full sequence of output labels and features $\mathbf{F}(U_1)$ to $\mathbf{F}(U_N)$ are fed together. Thus, the grouping is

Make Groups as discussed previously. Let U_i:G indicate the group that U_i belongs to. U_i:G_{prev} indicates the group previous to U_i:G and U_i:G_{next} indicates next group
for i ← 1 to N:
    P(U_i) ← final prediction for U_i
    replace polarity information of U_i with P(U_i) and new *proba*
    if P(U_i) = U_i:G{polarity} **then** *pass*
    else
        if U_i **is** first unit in a group **then**
            if P(U_i) = U_i:G_{prev} {polarity} **then** add U_i to previous group
            else make new group for U_i
        else if U_i **is** last unit in a group **then**
            if P(U_i) = U_i:G_{next} {polarity} **then** add U_i to next group
            else make new group for U_i
        else
            make new group for U_i

Figure 5: Final Prediction.

done only once unlike that described in Figure 5. For CRFUITE settings, LBFGS algorithm is used, with 'max_iterations' equal to 1000.

5 Evaluation

5.1 Experiment Setup

The data for experiment is obtained from SemEval Workshop (2016, 2015) data-sets for ABSA (Pontiki et al., 2016). The data is provided for Restaurant domain in English language and contains labels for aspect-terms, category and polarity. Additionally, data is also obtained from Bing Liu's Consumer Review collection (5 + 9 product data)[8]. This data is for Consumer Electronics (CE) domain, in English language, and has ordinal labels (-3, -2, -1, +1, +2, +3). For 3-class classification, (-1, +1) values are mapped to *neutral*, (+2, +3) to *positive* and remaining to negative class. The data divisions are given in Table 1. Bing Liu's data[9] is divided into 70:30 ratio for training and evaluation. Also, data for number of transitions is given in Table 2. While all reviews are used for experiments, reviews with more than 3 transitions are of special interest to study non-local dependencies. The SemEval - 2016 training data is a mix of SemEval - 2015 training & evaluation data. So, 2015 data is used only for comparison. The data-sets are not balanced; for e.g., in 2016 data, the proportions of pos:neg:neutral instances are 1:1/2:1/15, approximately. Thus, before training, the class weights are balanced in the SVM predictor. Aspect-categories are not provided in Bing Liu's data. Thus, the category related baseline features are dropped for this data.

Two types of training and evaluation (or test) setup are used. In *setup1*, only base model is used. The base model is trained on entire training set after 10-fold cross-validation. Then, predictions made on the test set. In *setup2*, 10-fold cross validation is performed on the training set with base model. However, this time the *proba* values for each validation partition are saved. Finally, the *proba* values for all validation partitions of training data are available, so the Level-2 model is trained on the entire training set. The combined model is then used for predictions on test set. Similar process has been used for multi-stage prediction previously (Krishnan and Manning, 2006)

The system is built using scikit-learn and NLTK (Bird et al., 2009) packages in Python 2.7. Parameters of SVM are set using Grid Search. For our experiments, C=100 and gamma in the range of 0.001 to 0.005 work well (gamma = 0.001 is chosen). RBF kernel is used and decision_function type is one-vs-rest. Before feeding into the model, the data is cleared of stopwords (using NLTK stopword list) and special

[8]https://www.cs.uic.edu/ liub/FBS/sentimentanalysis.html
[9]ipod and powershot files excluded due to low context information

Data	#Aspects	pos	neg	neut
SemEval 2016				
Restaurant - training	2500	1650	750	100
Restaurant - evaluation	859	613	206	40
SemEval 2015				
Restaurant - training	1654	1198	404	53
Restaurant - evaluation	845	457	347	45
Bing Liu's data				
All (5 + 9 products)	3933	2130	1036	767

Table 1: Approximate divisions for review data.

#Terminal Labels	#Reviews
SemEval 2016 data	
> 3	53 (training)
	21 (evaluation)
= 3	71 (training)
	19 (evaluation)
< 3	242 (training)
	53 (evaluation)
SemEval 2015 data	
> 3	29 (training)
	25 (evaluation)
= 3	49 (training)
	20 (evaluation)
< 3	183 (training)
	53 (evaluation)
Bing Liu's data	
> 3	148
= 3	120
< 3	321

Table 2: No. of reviews according to terminal labels (i.e. no. of transitions in polarity).

characters. The data is also lemmatized and all terms converted to lowercase. Also, in order to reduce the size of feature set, only 2000 best bi-grams are selected using Chi-square function[10].

5.2 Results and Discussion

The measure used for performance evaluation is the prediction Accuracy[11]. The results of evaluation are provided in Table 3. The results for *setup1* are listed under 'Base Model' and that for *setup2* are under 'Base + Level-2'. For base model, it can be seen that bi-grams and aggregated sentiment scores are the most significant features. It is to be noted that discourse-based aggregation shows good improvement in accuracy. This is expected because discourse can inherently help in incorporating consistency, contrast or negation of sentiment over a sentence, which is difficult to achieve by simple aggregation rules. We believe that with more detailed use of discourse relations, the performance can be further improved. On top of the base model, the combined 'Base + Level-2' model shows higher accuracy scores. Thus, embedding non-local neighbor features does provide richer context information, thereby also resolving sentiments associated with ambiguous sentences or phrases.

The performance of CRF (Level-2) in our experiment is below SVM (Level-2). This may seem counter-intuitive since a CRF should be able to model inter-label dependencies well. However, we wish to emphasize that the experiment with CRF is not aimed at comparison against SVM, but to check the performance of available CRF tool on review data. Firstly, the crfsuite library used for this experiment uses a linear-chain CRF model. If our intuition about non-local dependency holds, then linear-chain CRF should not be sufficient for a performance much superior than SVM. Secondly, the setup of SVM (level-2) is different from the input and output setup for crfsuite. For crfsuite, the full sequence of aspect unit features is fed as input, and full sequence of output labels is predicted at once. On the other hand, in the SVM (level-2) model we form new groups (or modify existing group information) as new labels are predicted. Thus, the context information provided as input changes as prediction proceeds. This is to include as much context information as possible during prediction. The difference in setup does not necessarily mean that CRF should perform below SVM, but it makes a definite comparison unfeasible. Devising a structured prediction method suited for non-local dependencies among polarity labels is kept as a future work. Such work would be more suitable to perform comparison between structured prediction and discrete prediction (i.e., with inter-label independence assumption).

The comparison of our two-stage model against few previous proposals is given in Table 4. Our

[10] http://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection

[11] http://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score

Method	SemEval-16	Bing Liu's
Base Model		
Base (SVM)		
Only n-grams	77.43	78.45
Only sentiment scores - external + neutral (simple aggregation)	79.60	79.20
Only sentiment scores - external + neutral (discourse-based aggregation)	81.75	81.05
Only sentiment scores - all	82.25	81.05*
Sentiment scores + n-grams	83.36	81.48
All features	83.44	81.48*
Base (SVM) + Level-2 (SVM)		
All features	**87.30**	**83.90**
Base (SVM) + Level-2 (CRF)	86.01	81.80

Table 3: Approximate accuracy scores. ($\times$ *category-specific features dropped for Bing Liu's data.)

Method	Accuracy
SemEval 2015 data	
SENTIUE (Saias, 2015)	78.70
ECNU (Zhang and Lan, 2015)	78.11
Base (SVM) + Level-2 (SVM)	**82.80**
SemEval 2016 data	
XRCE (Brun et al., 2016)	**88.12**
IIT-TUDA (Kumar et al., 2016)	86.73
Base (SVM) + Level-2 (SVM)	87.30

Table 4: Comparison against top-2 systems in previous SemEval workshops.

model performs relatively better for SemEval-2015 data. For SemEval-2016 data, the top-scoring system 'XRCE' shows better result. XRCE uses a feedback mechanism which provides information about feature relevance and cross-validation errors to the Feature Design step. We do not explore or replicate XRCE's design in detail. However, we believe that their feedback mechanism leads to more robust features and results in better accuracy.

5.3 Conclusion and Future Work

Concepts like *Sentiment Flow* and *Discourse relations* are important to address semantics of text for sentiment analysis. Such approach basically concerns with: (1) Incorporating local as well as non-local neighbor information as features and (2) structured prediction of a sequence of polarity labels with some constraint on correlation between non-adjacent labels. This problem needs to be approached in holistic manner. However, under non-locality assumption, the existing methods for structured prediction may become too complex. Moreover, relations like discourse and coreference can also be used to embed non-local context as features. However, it is important to extend this concept towards more data-driven approach. Thus, in this paper, we propose a multi-level model where a probability distribution obtained from first level can be used to incorporate non-local neighbor features, in addition to discourse, for further level of prediction. We show that multi-level model with non-local information can achieve some improvement in aspect-based prediction. The model evaluated on different data-sets performs in the 83-88% accuracy range. Nonetheless, it is important to explore more robust methods in theory and practice. The methods proposed by Kazama and Torisawa (2007), and Collins (2002) provide good basis for this. Also, it would be interesting to explore CRF with Posterior Regularization constraints taken from first level predictions, building upon work by Yang and Cardie (2014). Moreover, prediction can be improved by aggregating sentiment at sentence or phrase-level using discourse markers. The technique used in this paper is rudimentary. In Rhetorical Structure Theory, much intricate discourse relations have been proposed and there have been interesting works in this area exploring richer discourse concepts. Thus, in further work, we would expand the discourse relations used in sentiment aggregation as well as for linking aspect-units.

References

Henning Wachsmuth, Johannes Kiesel, and Benno Stein. 2015. Sentiment Flow–A General Model of Web Review Argumentation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 601–611.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, and Gregor Engels. 2014. Modeling Review Argumen-

tation for Robust Sentiment Analysis. *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, 553–564.

Yi Mao and Guy Lebanon. 2006. Isotonic conditional random fields and local sentiment flow. *Advances in neural information processing systems*, 961–968.

Jun'ichi Kazama and Kentaro Torisawa. 2007. A new perceptron algorithm for sequence labeling with non-local features. *Proc. of EMNLP-CoNLL*, Volume 7, 315–324.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. *LREC*.

Swapna Somasundaran, Galileo Namata, Janyce Wiebe, and Lise Getoor. 2009. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, 170–179. Association for Computational Linguistics.

Swapna Somasundaran, Galileo Namata, Lise Getoor, and Janyce Wiebe. 2009. Opinion graphs for polarity and discourse classification. *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing*, 66–74. Association for Computational Linguistics.

Swapna Somasundaran, Josef Ruppenhofer, and Janyce Wiebe. 2008. Discourse level opinion relations: An annotation study. *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, 129–137. Association for Computational Linguistics.

Subhabrata Mukherjee and Pushpak Bhattacharyya. 2012. Sentiment Analysis in Twitter with Lightweight Discourse Analysis. *Proceedings of COLING*, 1847–1864. Association for Computational Linguistics.

Livia Polanyi and Annie Zaenen. 2006. Contextual valence shifters. *Computing attitude and affect in text: Theory and applications*, 1–10.

Bing Liu. 2012. Sentiment Analysis and Opinion Mining. *Synthesis Lectures on Human Language Technologiese.*

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *LREC*, 165—210.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. *Proceedings of the 9th international conference on Language Resources and Evaluation (LREC).*

Xiaodan Zhu, Hongyu Guo, Saif Mohammad, and Svetlana Kiritchenko. 2014. An Empirical Study on the Effect of Negation Words on Sentiment. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 304–313.

Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. 2004. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, Volume 5, 975–1005.

Maria Pontiki, Dimitris Galanis, and others. 2016. SemEval-2016 Task 5: Aspect Based Sentiment Analysis. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 19–30. Association for Computational Linguistics.

Michael Collins. 2002. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 489–496. Association for Computational Linguistics.

Caroline Brun, Julien Perez, and Claude Raux. 2016. XRCE at SemEval-2016 Task 5: Feedbacked Ensemble Modeling on Syntactico-Semantic Knowledge for Aspect Based Sentiment Analysis. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 277–281. Association for Computational Linguistics.

Ayush Kumar, Sarah Kohail, Amit Kumar, Asif Ekbal, and Chris Biemann. 2016. IIT-TUDA at SemEval-2016 Task 5: Beyond Sentiment Lexicon: Combining Domain Dependency and Distributional Semantics Features for Aspect Based Sentiment Analysis. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 1129–1135. Association for Computational Linguistics.

Talaat Khalil and Samhaa R. El-Beltagy. 2016. NileTMRG at SemEval-2016 Task 5: Deep Convolutional Neural Networks for Aspect Category and Sentiment Extraction. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 271–276. Association for Computational Linguistics.

Zhiqiang Toh and Jian Su. 2016. NLANGP at SemEval-2016 Task 5: Improving Aspect Based Sentiment Analysis using Neural Network Features. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 282–288. Association for Computational Linguistics.

Bo Wang and Min Liu. 2015. Deep Learning for Aspect-Based Sentiment Analysis. https://cs224d.stanford.edu/reports/WangBo.pdf.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, Volume 1631.

José Saias. 2015. Sentiue: Target and Aspect based Sentiment Analysis in SemEval-2015 Task 12. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, 767–771. Association for Computational Linguistics.

Zhihua Zhang and Man Lan. 2015. ECNU: Extracting Effective Features from Multiple Sequential Sentences for Target-dependent Sentiment Analysis in Reviews. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, 767–771. Association for Computational Linguistics.

Arjun Mukherjee and Bing Liu. 2012. Aspect extraction through semi-supervised modeling. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, 339–348. Association for Computational Linguistics.

Vijay Krishnan and Christopher D Manning. 2006. An effective two-stage model for exploiting non-local dependencies in named entity recognition. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, 1121–1128. Association for Computational Linguistics.

Guilherme Hoefel and Charles Elkan. 2008. Learning a two-stage SVM/CRF sequence classifier. *Proceedings of the 17th ACM conference on Information and knowledge management*, 271–278. Association for Computing Machinery.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of Conditional Random Fields (CRFs). http://www.chokkan.org/software/crfsuite.

Charles Sutton and Andrew McCallum. 2010. An introduction to conditional random fields. arXiv preprint arXiv:1011.4088. homepages.inf.ed.ac.uk/csutton/publications/crftutv2.pdf.

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

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using CRFs with hidden variables. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 786-794. Association for Computational Linguistics.

Y. Choi and C. Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 793-801. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, 271. Association for Computational Linguistics.

Lanjun Zhou, Binyang Li, Wei Gao, Zhongyu Wei, and Kam-Fai Wong. 2011. Unsupervised discovery of discourse relations for eliminating intra-sentence polarity ambiguities. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 162–171. Association for Computational Linguistics.

Angeliki Lazaridou, Ivan Titov, and Caroline Sporleder. 2013. A bayesian model for joint unsupervised induction of sentiment, aspect and discourse representations. *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*, 1630-1639. Association for Computational Linguistics.

Qi Zhang, Jin Qian, Huan Chen, Jihua Kang, and Xuanjing Huang. 2013. Discourse level explanatory relation extraction from product reviews using first-order logic.

Bishan Yang and Claire Cardie. 2014. Context-aware Learning for Sentence-level Sentiment Analysis with Posterior Regularization. *ACL(1)*, 325-335. Association for Computational Linguistics.

Two-View Label Propagation to Semi-supervised Reader Emotion Classification

Shoushan Li, Jian Xu, Dong Zhang, Guodong Zhou*
Natural Language Processing Lab
School of Computer Science and Technology, Soochow University, China
lishoushan@suda.edu.cn, jxu1017@stu.suda.edu.cn
dzhang@stu.suda.edu.cn, gdzhou@suda.edu.cn

Abstract

In the literature, various supervised learning approaches have been adopted to address the task of reader emotion classification. However, the classification performance greatly suffers when the size of the labeled data is limited. In this paper, we propose a two-view label propagation approach to semi-supervised reader emotion classification by exploiting two views, namely *source* text and *response* text in a label propagation algorithm. Specifically, our approach depends on two word-document bipartite graphs to model the relationship among the samples in the two views respectively. Besides, the two bipartite graphs are integrated by linking each *source* text sample with its corresponding *response* text sample via a length-sensitive transition probability. In this way, our two-view label propagation approach to semi-supervised reader emotion classification largely alleviates the reliance on the strong sufficiency and independence assumptions of the two views, as required in co-training. Empirical evaluation demonstrates the effectiveness of our two-view label propagation approach to semi-supervised reader emotion classification.

1 Introduction

***Source* Text (*News*)**: *An earthquake of 7.0 magnitude struck China in Lushan County of Ya'an, Sichuan Province, causing serious casualties and property losses, and millions of people in distress……* **Writer Emotion:** *Neutral* **Reader Emotion**:  (*Sadness*),  (*Worry*)
***Response* Text (*Comments*)**: (1) *Be sure to cherish the golden rescue time.* (2) *Why there is always an earthquake, so sad, wish all the best.* (3) *I experienced this quake, all too suddenly, and I will never forget.*

Figure 1: An example of a news article, together with its writer and reader emotions

Emotion classification aims to predict the involving emotion towards a piece of text. For a particular text, there always exist two kinds of emotions, namely writer emotion and reader emotion, where the former concerns the emotion produced by the writer who writes the text and the latter concerns the

* Corresponding author

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2647–2655, Osaka, Japan, December 11-17 2016.

emotion produced by the reader who reads the text. For instance, in Figure 1, given the news text about earthquake, the writer emotion is more likely to be *neutral* due to the professionalism of the news reporter, while the reader emotion might be *sadness*, or *worry*. Recent years have seen growing interest in reader emotion classification due to its importance in more and more real-life applications, such as content recommendation and online advertisement.

Conventional approaches to reader emotion classification conceptualize the task as a supervised learning problem and rely on a large-scale human-annotated data for model learning. Although such supervised approaches deliver reasonably good performance, the reliance on labeled data, which is normally difficult and highly expensive to obtain, presents a major obstacle to the widespread application of reader emotion classification.

To alleviate the problem above, Liu et al. (2013) originally propose a semi-supervised learning approach to reader emotion classification to improve the performance by enlarging the labeled data with automatically inferred annotations of unlabeled instances. Their basic idea mainly lies on unique characteristics in reader emotion analysis, different from the case in writer emotion analysis. That is, apart from the *source* text (e.g., *news* text), another type of text, the *response* text (e.g., *comment* text) written by the reader as a response to the *source* text, is available to help determine the reader emotion of the *source* text. For example, in Figure 1, the *comment* "*Why there is always an earthquake, so sad, wish all the best*" explicitly express reader emotion *sadness*. Therefore, the *source* text and the *response* text are casted respectively as two views in a co-training algorithm to perform semi-supervised learning.

However, the success of co-training largely depends on two strong underlying assumptions, i.e., sufficiency and independence, of the two views (Blum and Mitchell, 1998), which are actually violated in reader emotion classification when the *source* text and *response* text are utilized as two views.

On one hand, the *response* text often lacks sufficient information to correctly predict the label of an instance, since the response text tends to be short. For example, in Figure 1, if there is only one existing comment, e.g., (1) "*Be sure to cherish the golden rescue time*", the reader emotion is difficult to predict because no emotion is clearly expressed in this sentence. Even worse, as an extreme example, the *source* text (e.g., some newly posted *news*) sometimes has no *response* at all.

On the other hand, the *response* text normally depends on the *source* text, since both the *response text* and the *source* text talk about the same topics. It is really hard for them to meet the view independence assumption in co-training.

In this paper, we propose a novel semi-supervised learning approach, namely two-view label propagation (LP), to reader emotion classification. As an extension of traditional label propagation with a single view (Zhu and Ghahramani, 2002), our two-view LP approach depends on two graphs, i.e., one depicting the connections among the *source* text samples and the other depicting the connections among the *response* text samples. Besides, the two graphs are integrated by linking each *source* text sample with its corresponding *response* text sample to capture the dependence between the *source* text and the *response* text. Such a two-view LP approach thus avoids the independence assumption, as required in traditional co-training. Finally, we assign a variable weight between each *source* text sample and its *response* text sample to address the information insufficiency in the *response* text. Specifically, we design a length-sensitive linear function to calculate the transition probability between the source and response text samples.

The remainder of this paper is organized as follows. Section 2 overviews related work on emotion classification. Section 3 introduces the baseline approach to semi-supervised reader emotion classification with single-view label propagation. Section 4 presents our two-view label propagation approach to semi-supervised reader emotion classification. Section 5 empirically evaluates our approach. Finally, Section 6 gives the conclusion and future work.

2 Related Work

Among the large number of studies in sentiment analysis over the last decade (Pang et al., 2002; Turney, 2002; Alm et al., 2005; Wilson et al., 2009), only a small portion focus on emotion classification.

Besides those on emotion resource construction, such as emotion lexicon building (Xu et al., 2010; Volkova et al., 2012; Staiano and Guerini, 2014) and sentence-level or document-level corpus construction (Quan and Ren, 2009; Das and Bandyopadhyay, 2009), most of previous studies on emotion classification are devoted to designing novel classification approaches to emotion classification (Alm et al.,

Symbol	Definition
L_s	Labeled *source*-text data
L_r	Labeled *response*-text data
U_s	Unlabeled *source*-text data
U_r	Unlabeled *response*-text data
G_s	Graph of the *source*-text data
G_r	Graph of the *response*-text data
$G_{s,r}$	Joint graph of both the *resource* and *response* text data
M_s	The transition probability matrix among the *source* text data
M_r	The transition probability matrix among the *response* text data
$M_{s,r}$	The transition probability matrix among the *source* and *response*-text data

Table 1: Symbol definition

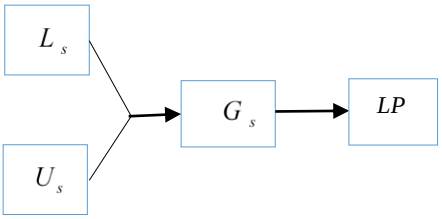


Figure 2: The framework of single-view label propagation approach to semi-supervised learning on reader emotion classification

2005; Chen et al., 2010; Purver and Battersby, 2012; Hasegawa et al., 2013; Qadir and Riloff, 2014), mainly from the supervised learning paradigm.

Compared with above studies on writer emotion classification, studies on reader emotion classification are much limited. Lin et al. (2007) first describe the task of reader emotion classification on news articles with some standard machine learning approaches. Lin et al. (2008) further exploit more features to improve the performance.

More recently, Liu et al. (2013) propose a co-training approach to semi-supervised learning on reader emotion classification by considering the *message* text and the *comment* text as two views. However, their success is much limited due to the required two strong assumptions on co-training, i.e. sufficiency and independence assumptions on the two views in co-training.

3 Single-view LP to Semi-supervised Reader Emotion Classification

In reader emotion classification, each target (e.g., a *news* article) is represented by two kinds of text, namely *source* text and *response* text. Formally, we refer the training data containing the *source* text samples as L_s and the one containing the *response* text samples as L_r. In this study, we only consider two emotion categories, i.e., *positive* and *negative* emotions. The task of semi-supervised learning on reader emotion classification is to leverage the training data L_s and L_r, together with the unlabeled data U_s and U_r, to train a classifier. For clarity, Table 1 illustrates some important symbols.

Figure 2 illustrates the framework of the LP-based semi-supervised approach to when only the view of the *source* text is utilized. Traditional label propagation (LP) is a graph-based semi-supervised learning approach with a single view (Zhu and Ghahramani, 2002). In general, a LP-based approach to semi-supervised learning consists of two main steps: graph construction to represent the relationship among the document samples and label propagation to propagate the labels of the labeled data to the unlabeled

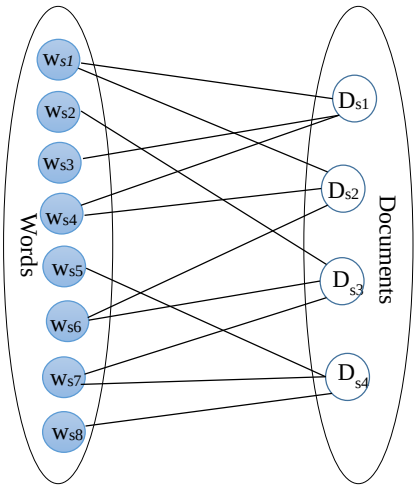


Figure 3: The word-document bipartite graph

Input:

P: The $n \times 2$ matrix, while p_{ir} represents the probability of document D_i (i=1...n) with label r (r=0,1);

M : The $n \times n$ transition probability matrix

Output:

The unlabeled data with prediction labels

Procedure:

1) Initialize P as P_0
   a) Assign each labeled sample with a fixed probability distribution (1, 0) or (0,1) according to its label r;
   b) Assign each unlabeled sample with an initial probability distribution (0.5, 0.5);
2) Loop until P converges;
   a) Propagate the labels of any vertex to nearby vertices by $P_t = M^T \cdot P_{t+1}$;
   b) Clamp the labeled data, that is, replace the probabilities of the labeled samples in P_{t+1} with their initial ones in P_0;
3) Assign each unlabeled instance with a label by computing $\arg\max_r p_{ir}$

Figure 4: The LP algorithm

data in the obtained graph.

In detail, in the first step, we adopt a word-document bipartite graph to model the relationship among the document samples due to its excellent performance in sentiment classification (Sindhwani and Melville, 2008). Figure 3 illustrates the structure of the word-document bipartite graph, in which the nodes consist of two parts: all documents and all words extracted from the documents. An undirected edge (D_i, w_k) exists if and only if document D_i contains word w_k. Let x_{ik} be the frequency of word w_k in document D_i. From the bipartite graph, the probability of walking from document D_i to word w_k can be calculated as $x_{ik} / \sum_k x_{ik}$ and the probability of walking from word w_k to document d_j can be calculated as $x_{jk} / \sum_j x_{jk}$. Thus the probability of walking from document D_i to document D_j though the word w_k can be calculated as $(x_{ik} / \sum_k x_{ik}) \cdot (x_{jk} / \sum_j x_{jk})$. When all words are considered, we get the transition probability from D_i to D_j as:

$$t_{ij} = \sum_k \frac{x_{ik}}{\sum_k x_{ik}} \cdot \frac{x_{jk}}{\sum_j x_{jk}} \quad (1)$$

and the transition probability matrix $M = \{t_{ij}\}$.

In the second step, we adopt the standard LP algorithm to perform semi-supervised learning. In detail, Figure 4 illustrates the LP algorithm (Zhu and Ghahramani, 2002), during which the probabilities of the labeled data are clamped in each loop using their initial ones and act as a force to propagate their labels to the unlabeled data.

4. Two-View LP to Semi-supervised Reader Emotion Classification

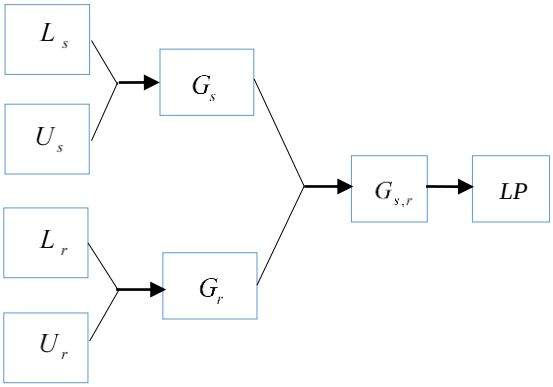


Figure 5: The framework of our two-view label propagation approach to semi-supervised learning on reader emotion classification

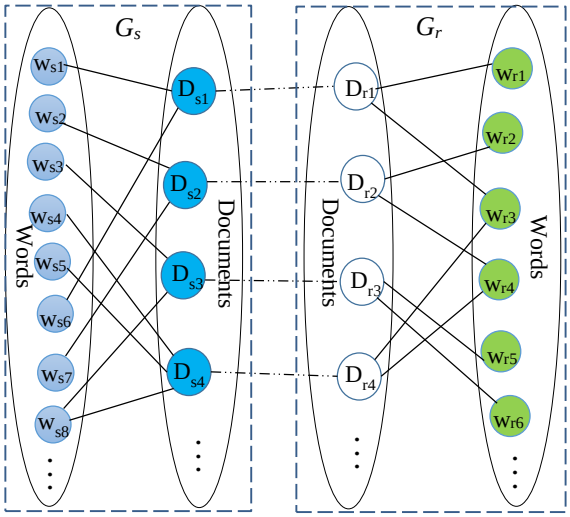


Figure 6: The joint two-view graph that contains both the *source* and *response* text sub-graphs

Figure 5 illustrates the framework of our LP-based semi-supervised approach to reader emotion classification when two views, i.e., the *source* text and the *response* text, are utilized. The graph in our approach consists of two sub-graphs, i.e., G_s and G_r, and each of them is modeled as a word-document bipartite graph. Each pair of the *source* text document and its corresponding *response* text document is connected to join the two sub-graphs together. Figure 6 illustrates the joint two-view graph of both the *source* text and *response* text data.

Basically, the transition probability between the *source* text document and its corresponding *response* text document can be set to 1, assuming that they exhibit the same category. Therefore, the transition probability matrix $M_{s,r}$ of the joint graph $G_{s,r}$ can be represented as follows:

$$M_{s,r} = \begin{bmatrix} M_s & I \\ I & M_r \end{bmatrix} \quad (2)$$

Where I is an identity matrix of dimension n, containing ones along the diagonal and zeros in all other positions; M_s and M_r are the transition probability matrixes, calculated using formula (1) in the sub-graphs, G_s and G_r, respectively.

However, when a *response* text contains too few information or even no words, it becomes a noisy sample. In this scenario, it is not appropriate to propagate the emotion label of its *source*-text sample to this noisy sample. Therefore, for the *source* text sample and its corresponding *response* text sample (e.g., D_{s1} and D_{r1} as shown in Figure 6, we design a function to measure the transition probability by taking the length of the *response* text into account, as follows:

$$t_{s_i,r_i}(z_{r_i}) = \begin{cases} 1 & z_{r_i} \geq l_{max} \\ z_{r_i} / l_{max} & z_{r_i} < l_{max} \end{cases} \quad (3)$$

Where z_{r_i} is the length of the response text document D_{r_i}, defined as the number of the words in the *response* text document; l_{max} denotes the threshold of the document length. If the length is larger than this threshold, it is given a transition probability of 1. If the length is smaller than this threshold, the longer the *response* text sample it is, the higher transition probability it has.

Accordingly, the transition probability matrix $M_{s,r}$ of the joint graph $G_{s,r}$ can be refined as follows:

$$M_{s,r} = \begin{bmatrix} M_s & I' \\ I' & M_r \end{bmatrix} \quad (4)$$

Where I' is a matrix of dimension n, containing the transition probabilities, calculated using formula (3).

5. Experimentation

We systematically evaluate our semi-supervised learning approach to reader emotion classification.

5.1 Experimental Settings

Data collection: The data is collected from Yahoo! Kimo News (http://tw.news.yahoo.com). Each *news* article and its *comments* are considered as a *source*-text sample and a *response*-text sample, respectively. Besides, each *news* article is voted with emotion tags from eight categories: *happy*, *sad*, *angry*, *mean-ingless*, *boring*, *heartwarming*, *worried*, and *useful*. Following Liu et al. (2013), we consider *happy* and *heartwarming* as *positive* category while *sad*, *angry*, *boring*, and *worried* as *negative* category. The emotion label of a news article is automatically derived from the votes, i.e. the *news* article with over 10 votes of *positive* (*negative*) emotions is assigned with a *positive* (or *negative*) label. Unlike Liu et al. (2013), we do not filter those news articles with less than 5 comments.

Data setting: We randomly select 1300 positive and 1300 negative *source* and *response* instances for the empirical study. Among them, 300 positive and 300 negative *source* and *response* instances are used as test data while the remaining 1000 positive and 1000 negative *source* and *response* instances are used as training data. In the training data, we select 0.5%, 1%, and 2% data as initial labeled data and the remaining data as unlabeled data respectively.

Features: Each *source* (or *response*) text is treated as a bag-of-words and transformed into binary vectors encoding the presence or absence of word unigrams.

Classification algorithm: The maximum entropy (ME) classifier implemented with the Mallet Toolkits (http://mallet.cs.umass.edu/).

Evaluation Measurement: The performance is evaluated using the standard accuracy measurement.

Significance test: *T*-test is used to evaluate the significance of the performance difference between two approaches (Yang and Liu, 1999).

5.2 Experimental Results on Single-view LP

In this section, we compare different approaches to semi-supervised learning on reader emotion classification when only one view is utilized. For fair comparison, we implement following approaches:

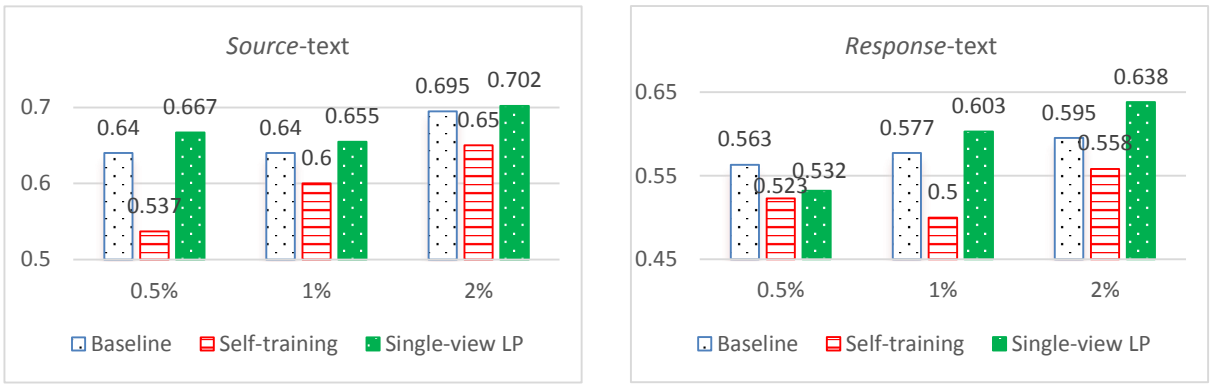


Figure 7: Performance comparison of different semi-supervised learning approaches to reader emotion classification when only one view is utilized

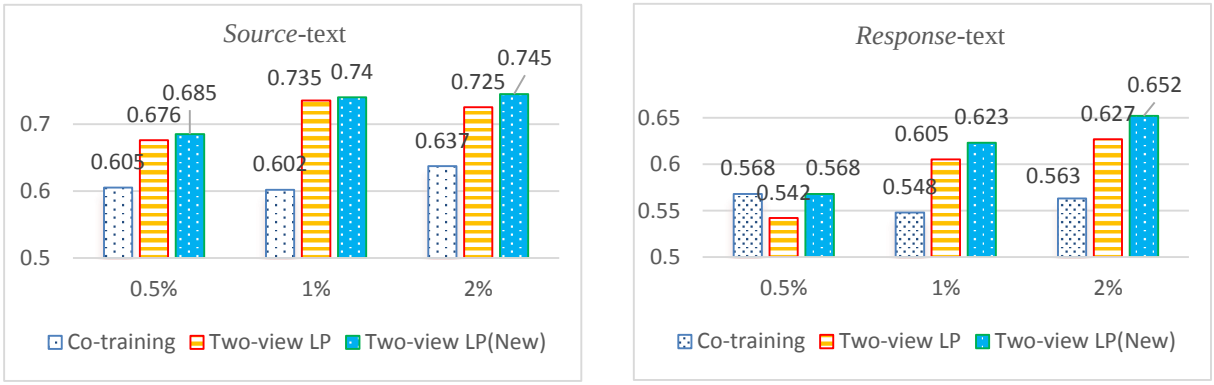


Figure 8: Performance comparison of different semi-supervised learning approaches to reader emotion classification when two views are utilized

- **Baseline**: using only labeled data to train the classifier for predicting the reader emotion of each view (No unlabeled data is used.)
- **Self-training**: using self-training, a simple bootstrapping approach, to iteratively add the high-confident unlabeled samples as automatically labeled samples in each view.
- **Single-view LP**: first using the word-document bipartite graph to model the relationship among the document samples in each view and then applying label propagation to perform semi-supervised learning as introduced in Section 3.1.

Figure 7 shows the performance of different approaches when either the *source*-text or the *response*-text view is utilized. From the figure, we can see that self-training performs dramatically worse than the single-view LP approach, even worse than the baseline approach. In general, the single-view LP approach is effective on using unlabeled data to improve the performance, although the average improvement is limited with around 2%. Besides, the single-view LP approach fails in the case when only 0.5% of the training data are used as initial labeled data. This is possibly because too few initial labeled samples make it extremely difficult to correctly bootstrap enough unlabeled samples.

5.3 Experimental Results on Two-view LP

In this section, we compare different approaches to semi-supervised learning on reader emotion classification when two views are utilized. For comparison, we implement following approaches:

- **Co-training**: using the *source*–text and *response*-text as two views in co-training, as described in Liu et al. (2013).
- **Two-view LP**: our two-view LP approach with the unique transition probability of 1 between the *source*-text sample and its *response*-text sample.
- **Two-view LP (New)**: the two-view LP approach with a variable transition probability between the *source* text sample and its corresponding *response* text sample, as calculated with formula (3). Here,

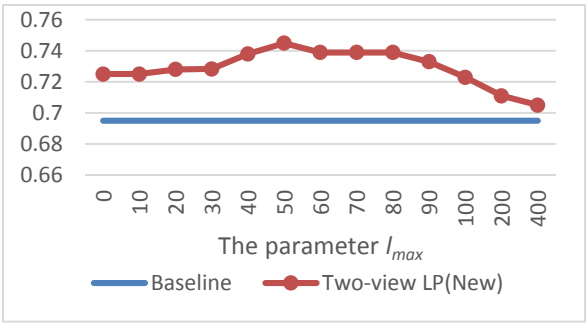


Figure 9: Sensitiveness of the performance on parameter l_{max}

parameter l_{max} is fine-tuned to be 50.

Figure 8 shows the performance of different approaches when both the *source*-text and the *response*-text views are utilized. From this figure, we can see that the two-view LP approach performs much better than co-training. Significance test shows that the improvement of two-view LP (New) over two-view LP is significant (p-value<0.05). This indicates the appropriateness of a variable transition probability between the *source* text sample and its corresponding *response* text sample.

From both Figure 7 and Figure 8, we can see that co-training fails to exploit unlabeled data to improve the performance, quite different from that in Liu et al. (2013). This is mainly due to the fact that the *response* text samples in our data contain much less comments. This makes the sufficiency assumption violated in co-training. Furthermore, we find that two-view LP significantly outperforms single-view LP (p-value<0.05) with a large margin. This verifies the effectiveness of using two views to perform semi-supervised learning on reader emotion classification.

Finally, Figure 9 shows the performance of two-view LP (New) with varying values of parameter l_{max} when testing on the *source*-text samples. Due to the space limitation, we only show the performance when using 2% training data as the initial labeled data. From this figure, we can see that our approach performs consistently well when the parameter is set from 40 to 90, which is a very broad range.

6. Conclusion

In this paper, we propose a novel approach, namely, two-view label propagation, to semi-supervised learning on reader emotion classification. Our approach consists of two main steps: (1) constructing a joint graph containing two word-document bipartite sub-graphs; (2) performing label propagation to incorporate the unlabeled data. Furthermore, we design a length-sensitive function to measure the transition probability from a *source* text sample to its responding *response* text sample. Experimental studies demonstrate that our two-view label propagation approach is capable of employing the two views and unlabeled data to improve the performance.

This work mainly focuses on reader emotion classification with only two categories, i.e., *positive* and *negative* emotions. In the future work, we will explore semi-supervised learning on reader emotion classification when more fine-grained categories, such as *happiness*, *sadness*, and *anger*, are considered. Moreover, given the wide potential of the two-view LP approach, we will explore it in other NLP tasks where two views are involved.

Acknowledgements

This research work has been partially supported by three NSFC grants, No.61273320, No.61375073, No.61331011.

References

Cecilia Ovesdotter Alm, Dan Roth and Richard Sproat. 2005. Emotions from Text: Machine Learning for Text-based Emotion Prediction. In *Proceedings of EMNLP-05*, pages 579-586.

Avrim Blum and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-training. In *Proceedings of COLT-98*.

Ying Chen, Sophia Yat Mei Lee, Shoushan Li and Chu-Ren Huang. 2010. Emotion Cause Detection with Linguistic Constructions. In *Proceeding of COLING-10*, pages 179-187.

Dipankar Das and Sivaji Bandyopadhyay. 2009. Word to Sentence Level Emotion Tagging for Bengali Blogs. In *Proceedings of ACL-09*, pages 149-152.

Takayuki Hasegawa, Nobuhiro Kaji, and Naoki Yoshinaga. 2013. Predicting and Eliciting Addressee's Emotion in Online Dialogue. In *proceedings of ACL-13*, pages 964-972.

Kevin Hsin-Yih Lin, Changhua Yang and Hsin-Hsi Chen. 2007. What Emotions do News Articles Trigger in Their Readers? In *Proceeding of SIGIR-07*, poster, pages 733-734.

Kevin Hsin-Yih Lin, Changhua Yang and Hsin-Hsi Chen. 2008. Emotion Classification of Online News Articles from the Reader's Perspective. In *Proceeding of the International Conference on Web Intelligence and Intelligent Agent Technology*, pages 220-226.

Huanhuan Liu, Shoushan Li, Guodong Zhou, Chu-Ren Huang, and Peifeng Li. 2013. Joint Modeling of News Reader's and Comment Writer's Emotions. In *proceedings of ACL-13*, short paper, pages 511-515.

Bo Pang, Lillian Lee and Shivakunmar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *Proceedings of EMNLP-02*, pp.79-86.

Matthew Purver and Stuart Battersby. 2012. Experimenting with Distant Supervision for Emotion Classification. In *Proceedings of EACL-12*, pages 482-491.

Ashequl Qadir and Ellen Riloff. 2014. Learning Emotion Indicators from Tweets: Hashtags, Hashtag Patterns, and Phrases. In *Proceedings of EMNLP-14*, pages 1203-1209.

Changqin Quan. and Fuji Ren. 2009. Construction of a Blog Emotion Corpus for Chinese Emotional Expression Analysis. In *Proceedings of EMNLP-09*, pages 1446-1454.

Vikas Sindhwani and Prem Melville. 2008. Document-Word Co-Regularization for Semi-supervised Sentiment Analysis. In *Proceedings of ICDM-08*, pages 1025-1030.

Jacopo Staiano. and Marco Guerini. 2014. Depeche Mood: a Lexicon for Emotion Analysis from Crowd-Annotated News. In *Proceedings of ACL-14*, pages 427-433.

Peter D. Turney. 2002. Thumbs up or Thumbs down? Semantic Orientation Applied to Unsupervised Classification of comments. In *Proceedings of ACL-02*, pages 417-424.

Svitlana Volkova, William B. Dolan and Theresa Wilson. 2012. CLex: A Lexicon for Exploring Color, Concept and Emotion Associations in Language. In *Proceedings of EACL-12*, pages 306-314.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2009. Recognizing Contextual Polarity: An Exploration of Features for Phrase-Level Sentiment Analysis. *Computational Linguistics*, vol.35(3), pages 399-433.

Ge Xu, Xinfan Meng and Houfeng Wang. 2010. Build Chinese Emotion Lexicons Using A Graph-based Algorithm and Multiple Resources. In *Proceeding of COLING-10*, pages 1209-1217.

Yiming Yang and Xin Liu. 1999. A Re-Examination of Text Categorization Methods. In *Proceedings of SIGIR-99*, pages 42-49.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from Labeled and Unlabeled Data with Label Propagation. *CMU CALD Technical Report*. CMU-CALD-02-107.

A Joint Sentiment-Target-Stance Model for Stance Classification in Tweets

Javid Ebrahimi, Dejing Dou, Daniel Lowd
Department of Computer and Information Science, University of Oregon
Eugene, Oregon 97403, USA
{javid,dou,lowd}@cs.uoregon.edu

Abstract

Classifying the stance expressed in online microblogging social media is an emerging problem in opinion mining. We propose a probabilistic approach to stance classification in tweets, which models stance, target of stance, and sentiment of tweet, jointly. Instead of simply conjoining the sentiment or target variables as extra variables to the feature space, we use a novel formulation to incorporate three-way interactions among sentiment-stance-input variables and three-way interactions among target-stance-input variables. The proposed specification intuitively aims to discriminate sentiment features from target features for stance classification. In addition, regularizing a single stance classifier, which handles all targets, acts as a soft weight-sharing among them. We demonstrate that discriminative training of this model achieves the state-of-the-art results in supervised stance classification, and its generative training obtains competitive results in the weakly supervised setting.

1 Introduction

Stance Classification (SC) is the task of inferring from text whether the author is in favor of a given target, against it, or has a neutral position toward it. This task, which can be complex even for humans (Walker et al., 2012a), is related to argument mining, subjectivity analysis, and sentiment classification. Generic sentiment classification is formulated as determining whether a piece of text is positive, negative, or neutral. However, in SC, systems must detect favorability toward a given (pre-chosen) target of interest. In this sense, SC is more similar to target-dependent sentiment classification (Jiang et al., 2011), with a major difference that the target of the stance might not be explicitly mentioned in text or might not be the target of the opinion (Mohammad et al., 2016). For example, the tweet below implies a stance against Donald Trump, through expressing support for Hillary Clinton.

> Target: Donald Trump
> My vote is definitely for Hillary. Can't trust #gop candidates.

This is an interesting task to study on social networks because of the abundance of personalized and opinionated language. Given the growing significance of the role social media is playing in our world, studying stance classification can be beneficial among others, in identifying electoral issues and understanding how public stance is shaped (Mohammad et al., 2015).

SemEval 2016 Task 6 organizers (Mohammad et al., 2016) released a joint stance and sentiment annotated dataset. Studying the correlation between sentiment and stance and how the former can help detect the latter is an important research question that we address in this paper. Our approach relies on one observation for stance detection in tweets. Ignoring general words and stopwords, a lot of the time, we can expect a rough dichotomy on the remaining n-grams of the tweets. Concretely, a stance-related n-gram either refers to a topic related to the target or bears a sentiment. In Table 1 *Christian*, *religion*, *Feminism*, and *campaign* are of the first type, while *murder* and *enjoyed* are of the second type. We design the model such that the probability of a stance y given the text x, and its associated target t and

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2656–2665, Osaka, Japan, December 11-17 2016.

(1) **Target**: legalization of abortion, **Stance**:Against, **Sentiment**:Negative Hillary, Here's one Christian whose religious views will never adapt to include abortion. Abortion is murder. (2) **Target**: Hillary Clinton, **Stance**:Favor, **Sentiment**:Positive Enjoyed @jamiaw article on feminism + @hillaryclinton. We are building campaign that engages ppl through an intersectional lens.

Table 1: Two examples from SemEval 2016 Task 6.A on two of the targets specified in the corpus.

sentiment s, is proportional to the product of two components. The first component measures the consistency of x with sentiment s and stance y, while the second component measures the consistency of x with target t and stance y. The model learns how to discriminate among the target-specific features and sentiment-specific features, the latter of which might be generalized across different targets. This is further improved by performing regularization on one single classifier, as opposed to a different classifier for each target, which so far has been the standard way to do stance classification.

Our discriminative model works effectively for supervised stance classification tasks. However, manual annotation requires painstaking work by researchers, which can be even more difficult for tasks such as sentiment annotation (Mohammad, 2016). To this end, we propose a generative model, which works properly for stance prediction especially in weakly supervised settings, in which labeled instances are few and labels might be noisy.

Our contributions are as follows:

1. We address the modeling of interactions among target of stance, stance itself, and sentiment in text, by an undirected graphical model.

2. We use one single classifier for stance classification across multiple targets, as opposed to previous works, which use a separate classifier for each target. We demonstrate how our particular model specification and shared regularization can improve stance classification across multiple targets.

3. We develop both discriminative and generative training algorithms, which achieve the state-of-the-art results on supervised and competitive results for weakly supervised stance classification tasks, respectively.

2 Related Work

Previous work has focused on congressional debates (Thomas et al., 2006; Yessenalina et al., 2010), company-internal discussions (Agrawal et al., 2003), and debates in online forums (Anand et al., 2011; Somasundaran and Wiebe, 2010). There is a growing interest in performing stance classification on other media. For example, Faulkner (2014) detected document-level stance in student essays. Sobhani et al. (2015) extracted arguments used in online news comments to detect stance. The data from the Emergent Project[1] was used to classify the stance of article headlines (Ferreira and Vlachos, 2016). SemEval-2016 Task 6 (Mohammad et al., 2016) involved two stance detection subtasks in tweets in supervised and weakly supervised settings.

Somasundaran and Wiebe (2010) developed a baseline for stance classification using features based on modal verbs and sentiments. Anand et al. (2011) augmented the *n*-gram features with lexicon-based and dependency-based features. FrameNet semantic frames have also been incorporated in (Hasan and Ng, 2013; Hasan and Ng, 2014). SC has newly been posed as collective classification. For example, citation structure (Burfoot et al., 2011) or rebuttal links (Walker et al., 2012b), are used as extra information to model agreements or disagreements in debate posts and to infer their labels. In (Murakami and Raymond, 2010) a *maximum cut* method is used to aggregate stances in multiple posts to infer a user's stance on the target. Sridhar et al. (2015) use Probabilistic Soft Logic (PSL) to collectively classify the stance of users and stance in posts. PSL has also been used to augment a weakly-labeled tweet collection by incorporating Twitter's network-based features (Ebrahimi et al., 2016). Similarly, Rajadesingan et al.

[1] http://towcenter.org/research/lies-damn-lies-and-viral-content/

(2014), use a retweet-based label propagation method, which starts from a set of opinionated users and labeled tweets by the people who are in the retweet network. Since arguments and counter-arguments occur in sequences, Hasan and Ng (2014) were able to pose stance classification in debate forums as a sequence labeling task.

Tweets pose some challenges that preclude the use of standard off-the-shelf NLP feature extractors (Dey and Haque, 2009). Tweets have restricted length, which sometime makes the author use unstructured or incoherent statements. This is aggravated by the highly informal language that is common on Twitter, which includes grammatical errors. Not surprisingly, the results of SemEval 2016 Task 6 (Mohammad et al., 2016) showed the effectiveness of simple word *n*-grams and character *n*-grams, in addition to deep neural network approaches that automatically extract features. We use *n*-gram features in this work. But we will briefly discuss how our (discriminative) formulation can be incorporated into neural nets too.

In the next sections, we present effective baselines for joint modeling of targets, sentiments, and stances by a simple log-linear approach. We develop both generative and discriminative models and perform experiments on SemEval 2016 Tasks 6.A and 6.B (Mohammad et al., 2016).

3 STS: Joint Sentiment-Target-Stance Modeling

3.1 Log-Linear STS Model

In this paper, $\mathbf{x} \in \mathbb{R}^V$ is a vector of input features and $y \in Y$ is a discrete stance label, which we handle by a one-hot vector $\boldsymbol{e}_y \in \mathbb{R}^M$. Similar vectors, $\boldsymbol{e}_s \in \mathbb{R}^P$ and $\boldsymbol{e}_t \in \mathbb{R}^Q$, are defined for sentiment and target variables respectively. The model is defined over tensors $\Lambda^1 \in \mathbb{R}^{P \times V \times M}$ and $\Lambda^2 \in \mathbb{R}^{Q \times V \times M}$. Λ^1 and Λ^2 govern the sentiment-stance-feature and target-stance-feature interactions respectively. We define the following energy function,

$$E(y|\mathbf{x}, s, t; \Lambda^1, \Lambda^2) = -\sum_k \mathbf{x}_k(\lambda^1_{s,k,y} + \lambda^2_{t,k,y})$$

The negative of the energy associated with the stance label y, given the text $\mathbf{x}$, and its associated target t and sentiment s, is equal to the summation of two components. The first one measures the consistency of $\mathbf{x}$ with sentiment s and stance y, while the second one measures the consistency of $\mathbf{x}$ with target t and stance y. This specification is a log-linear model with feature functions $\phi^1(t, k, y) = \mathbb{1}(T = t, \mathbf{x}_k = 1, Y = y)$ and $\phi^2(s, k, y) = \mathbb{1}(S = s, \mathbf{x}_k = 1, Y = y)$, where $\mathbb{1}$ is the indicator function.

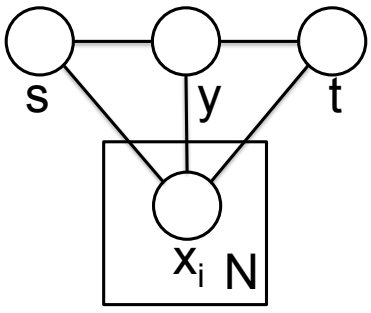


Figure 1: Plate model for STS

We build $p_\theta(y, s, t, \mathbf{x})$, a generatively trained model, and $p_\theta(y|s, t, \mathbf{x})$, a discriminatively trained model. For both models, the inference problem, the probability of output y conditioned on inputs, is given by,

$$p(y|\mathbf{x}, s, t) = \frac{\exp\big(\mathbf{x}^T(\boldsymbol{e}_s^T \Lambda^1 \boldsymbol{e}_y + \boldsymbol{e}_t^T \Lambda^2 \boldsymbol{e}_y)\big)}{\sum_{y'} \exp\big(\mathbf{x}^T(\boldsymbol{e}_s^T \Lambda^1 \boldsymbol{e}_{y'} + \boldsymbol{e}_t^T \Lambda^2 \boldsymbol{e}_{y'})\big)} \quad (1)$$

where $\boldsymbol{a}^T \Lambda \boldsymbol{b}$ is a bilinear tensor product which results in a vector, $h \in \mathbb{R}^V$. We use this tensor-based notation mainly to facilitate the description of the generative training algorithm.

3.2 Generative Training

Consider the training dataset $\mathcal{D}_{train}$, containing instances of the binary feature vector $\mathbf{x}$. To train a generative model, we use minimization of the negative joint log-likelihood.

$$\mathcal{L}(\mathcal{D}_{train}) = -\sum_{i=1}^{|\mathcal{D}_{train}|} \log p(\mathbf{y}_{(i)}, \mathbf{s}_{(i)}, \mathbf{t}_{(i)}, \mathbf{x}_{(i)})$$

In order to minimize the negative log-likelihood, we would compute its gradient with respect to the model parameters. The exact gradient, for any parameter $\theta \in \{\Lambda^1, \Lambda^2\}$, can be written as,

$$\frac{\partial \log p(\mathbf{y}_{(i)}, \mathbf{s}_{(i)}, \mathbf{t}_{(i)}, \mathbf{x}_{(i)})}{\partial \theta} = -\mathbf{E}_{\text{data}}\left[\frac{\partial E(\mathbf{y}_{(i)}, \mathbf{s}_{(i)}, \mathbf{t}_{(i)}, \mathbf{x}_{(i)})}{\partial \theta}\right] + \mathbf{E}_{\text{model}}\left[\frac{\partial E(\mathbf{y}, \mathbf{s}, \mathbf{t}, \mathbf{x})}{\partial \theta}\right] \quad (2)$$

While the first expectation can be computed in closed form, the second expectation is intractable due to the partition function. However, we can approximate it by generating a sample from the underlying distribution estimated by an MCMC algorithm such as Gibbs sampling. But instead of running the Gibbs chain for the whole burn-in period, we can run the chain for only k steps. This approximation method is called k-contrastive divergence (CD) (Hinton, 2002), which can be interpreted as optimizing a difference of Kullback-Leibler divergences. The novelty of this approach is in setting the sampler's initial state for variables at a training sample $(\mathbf{y}_{(i)}, \mathbf{s}_{(i)}, \mathbf{t}_{(i)}, \mathbf{x}_{(i)})$; this way the energy surface is modified only around the data points. We used CD-1 in our work.

Given the conditional independence assertions and the binary features, it is straightforward to show[2],

$$p(\mathbf{x}|y, s, t) = \prod_k p(x_k = 1|y, s, t) = \prod_k \text{sigm}\big(\boldsymbol{e}_{\mathbf{x}_k}^T(\boldsymbol{e}_s^T \Lambda^1 \boldsymbol{e}_y + \boldsymbol{e}_t^T \Lambda^2 \boldsymbol{e}_y)\big)$$

where $\boldsymbol{e}_{\mathbf{x}_k}$ is the one-hot representation of feature $\mathbf{x}_k$. We note that if sentiment and target variables were treated as additional input variables and conditional independence was applied to them as well, this generative specification would become identical to *naive Bayes*, and no approximation would be necessary. It can also be shown that $p(y|\mathbf{x}, s, t)$ and other conditional distributions needed to perform Gibbs sampling, $p(s|\mathbf{x}, y)$ and $p(t|\mathbf{x}, y)$, all follow a softmax distribution (Salakhutdinov and Hinton, 2009). See Equation 1.

The gradient with respect to the l_{th} slice of the tensors Λ^1 and Λ^2 can be approximated by:

$$\frac{\partial E(\theta)}{\partial \Lambda^1_{[l]}} = -\sum_i (\mathbf{y}^l_{(i)} \boldsymbol{e}_{s_{(i)}} \mathbf{x}^T_{(i)} - \hat{\mathbf{y}}^l_{(i)} \hat{\mathbf{s}}_{(\mathbf{i})} \hat{\mathbf{x}}^T_{(i)}) \quad (3)$$

$$\frac{\partial E(\theta)}{\partial \Lambda^2_{[l]}} = -\sum_i (\mathbf{y}^l_{(i)} \boldsymbol{e}_{t_{(i)}} \mathbf{x}^T_{(i)} - \hat{\mathbf{y}}^l_{(i)} \hat{\mathbf{t}}_{(\mathbf{i})} \hat{\mathbf{x}}^T_{(i)}) \quad (4)$$

where $\hat{\mathbf{y}}_i$, $\hat{\mathbf{x}}_i$, $\hat{\mathbf{t}}_i$, and $\hat{\mathbf{s}}_i$ are samples from $p(y|\mathbf{x}, s, t)$, $p(\mathbf{x}|y, s, t)$, $p(t|\mathbf{x}, y)$, and $p(s|\mathbf{x}, y)$ after CD-1, respectively. However, we use a version of CD (Mnih and Hinton, 2007) in which, instead of sampling and obtaining a binary vector, we set $\hat{\mathbf{y}}_i$, $\hat{\mathbf{x}}_i$, $\hat{\mathbf{t}}_i$, and $\hat{\mathbf{s}}_i$ to the vector of probabilities given by the respective probability distributions.

3.3 Discriminative Training

To train a discriminative model, we minimize the *cross-entropy* error,

$$J(\theta) = -\sum_i^{|\mathcal{D}_{train}|} \sum_l \mathbb{1}(\mathbf{y}_{(i)} = l) \log p(l|\mathbf{s}_{(i)}, \mathbf{t}_{(i)}, \mathbf{x}_{(i)})$$

[2]For the sake of simplicity of presentation, we omit the bias units which are needed for generative training.

	Overall				Atheism	Climate	Feminism	Hillary	Abortion
Method	F_{favor}	$F_{against}$	$MicF_{avg}$	$MacF_{avg}$	F_{avg}	F_{avg}	F_{avg}	F_{avg}	F_{avg}
CNN	61.98	72.67	67.33	58.57	63.34	**52.69**	51.33	64.41	61.09
RNN	59.32	76.33	67.82	56.02	61.47	41.63	**62.09**	57.67	57.28
SVM	62.98	74.98	68.98	58.01	65.19	42.35	57.46	58.63	66.42
MaxEnt	60.78	73.41	67.10	56.37	60.82	41.43	55.73	59.87	63.99
NB	58.05	71.11	64.58	55.51	63.69	40.46	49.58	64.77	58.64
Disc-TS	62.96	76.12	69.55	58.85	61.90	41.73	56.76	63.91	**69.94**
Disc-STS	**64.43**	**77.62**	**71.03**	**61.40**	65.52	41.18	57.90	**74.48**	67.94
Gen-STS	61.43	77.02	69.23	60.41	**67.09**	50.04	53.77	71.25	59.92

Table 2: Results for Task A, reporting the official competition metric, overall $MicF_{avg}$, along with F_{avg} for each individual target, and the average of all individual F_{avg}, $MacF_{avg}$.

The gradients with respect to the l_{th} slice of the tensor Λ^1 and Λ^2 can be computed exactly,

$$\frac{\partial J(\theta)}{\partial \Lambda^1_{[l]}} = -\sum_i \boldsymbol{e}_{s_{(i)}} \mathbf{x}^T_{(i)} \big(\mathbb{1}(\mathbf{y}_{(i)} = l) - p(l|\mathbf{s}_{(i)}, \mathbf{t}_{(i)}, \mathbf{x}_{(i)})\big) \tag{5}$$

$$\frac{\partial J(\theta)}{\partial \Lambda^2_{[l]}} = -\sum_i \boldsymbol{e}_{t_{(i)}} \mathbf{x}^T_{(i)} \big(\mathbb{1}(\mathbf{y}_{(i)} = l) - p(l|\mathbf{s}_{(i)}, \mathbf{t}_{(i)}, \mathbf{x}_{(i)})\big) \tag{6}$$

In both discriminative and generative cases, the gradients can be regarded as update rules for the weights of the model, which are untied based on the sentiment (updates on Λ^1) or the target (updates on Λ^2) in the tweet.

4 Experiments

SemEval 2016 Task 6 (Mohammad et al., 2016) defined two stance classification tasks/datasets. The first one (Task 6.A) was a traditional supervised task, while the second one (Task 6.B) was a weakly supervised task wherein no tweet was stance-annotated. For both tasks, we used binary n-gram features: word n-grams (1–3 gram) and character n-grams (2–5 gram). We used $\ell 2$ regularization for our discriminative (Disc-STS) and generative (Gen-STS) models. For Task A, model hyper-parameters were estimated by cross-validation on the training set. For Task B, we used the dataset of the supervised task as the development set. Gen-STS was trained by stochastic gradient descent, and the learning rate was set to 0.0005. Disc-STS was trained in batch mode, and we used L-BFGS for optimization. Task B contains noisy stance labels; because of this, we performed early stopping in training to avoid overfitting to the wrong model. To this end, during parameter tuning on the development set, we used a larger range for the progress threshold in our grid search.

4.1 Supervised Task

SemEval-2016 Task 6.A provided stance-annotated tweets toward five targets: "Atheism", "Climate Change is a Real Concern", "Feminist Movement", "Hillary Clinton", and "Legalization of Abortion". The dataset contained 2,914 and 1,249 tweets for training and testing respectively.

4.1.1 Results

Table 2 shows the results for Task A. CNN (Wei et al., 2016) and RNN (Zarrella and Marsh, 2016) are convolutional neural network and recurrent neural network models that were the second best and the best system in the competition respectively. Both systems use pre-trained word embeddings before training for the task, which improves generalization and allows them to achieve good results on the task. The SVM classifier was the linear-kernel SVM used by task organizers, which was trained on the same features as ours (i.e., word n-grams (1–3 gram) and character n-grams (2–5 gram)). Two other reasonable baselines, which resemble our discriminative and generative models respectively, are *maximum entropy* (MaxEnt), and *naive Bayes* (NB) classifiers.

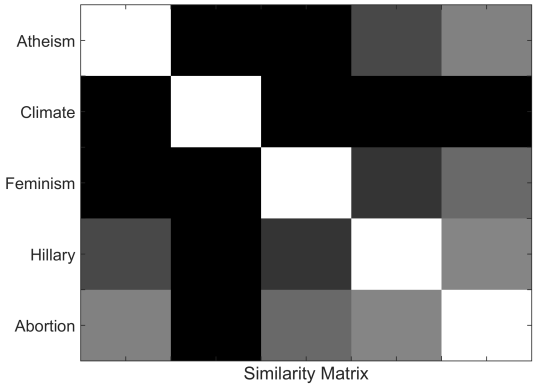


Figure 2: Similarity matrix of the weight vectors for task A targets. Lighter color denotes higher similarity.

The $MicF_{avg}$ metric is the mean of F_{favor} and $F_{against}$, which are the harmonic mean of Recall and Precision for each class. The metric $MicF_{avg}$ can be regarded as a micro-average of F-scores across targets. Alternatively, one could also determine the mean of the F_{avg} scores for each of the targets, the mean of which determines the ($MacF_{avg}$) metric.

Both Disc-STS and Gen-STS gain substantial gains over their natural baselines, MaxEnt and NB. Disc-STS improves the previous state-of-the-art results by 2.05% and 3.4% in Micro F1 and Macro F1 scores respectively. CNN and Gen-STS perform better on the "Climate" target, which is highly imbalanced (i.e., only 3.8% against). Unlike all the other baselines, which trained separate classifiers for each target, our approach can benefit from generalized features across multiple targets. Figure 2 displays the cosine similarity between the weight vectors for each of the targets. The weights used for this measure, were taken from the slice for that target, namely $\Lambda^2_{[target,:,:]}$.

Comparing MaxEnt and Disc-TS, the biggest improvement is found on the "Abortion" target. The performance on the targets, which are more similar to other targets in the corpus, is generally boosted significantly, compared with those that are not. The only difference between the two models is shared regularization across all the targets, which is causing the improvement.

Another way to investigate the inner workings of the model is to check if the model is able to discriminate sentiment features from target features. To do this, we represent the words based on the weights associated with them in the model. We concatenate the word-specific slices in the tensor parameter, namely $\Lambda^1_{[:,word,:]}$ and $\Lambda^2_{[:,word,:]}$, and compute the cosine similarity between pairs of word vectors. Table 3 shows the most similar words to 4 query words: two target-based and two sentiment-bearing words. It can be seen that among the top words similar to the sentiment-bearing words are some other sentiment-bearing words (positive or negative). The words similar to "climate" are clearly related to the target of "climate change". The words similar to "anti-choice" are about the target of "abortion", in addition to another related target, "feminism".

Given the significance of regularization and the dichotomy on the features, group lasso regularization (Yogatama and Smith, 2014), based on sentiment and target groups, can potentially improve our results.

We also report results on two subsets of the test set; (1) a subset where opinion is expressed toward the target; (2) a subset where opinion is expressed toward some other entity. Table 4 shows these results along with the overall $MicF_{avg}$, for the ease of reference.

4.2 Weakly Supervised Task

SemEval-2016 Task 6.B provided around 78,000 tweets associated with "Donald Trump". The tweets were gathered by polling Twitter for hashtags associated with Donald Trump. The protocol of the task only allowed minimal manual labeling, i.e. "tweets or sentences that are manually labeled for stance" were not allowed, but "manually labeling a handful of hashtags" and the use of other resources, e.g. lexicons, sentiment analyzers, etc., was permitted. This test set contained 707 tweets.

anti-choice	climate	excellent	crap
anti-abortion	global	together	asks
#feminism	co2	note	hack
effort	last	despite	hates
mentality	june	interesting	slut
benefits	warming	hate	shatter
unsafe	agriculture	retarded	#vaw
#reprorights	environmental	hatred	misogynistic
cunt	summer	warrior	adultery
types	mines	1st	either
banned	reducing	scum	societal

Table 3: Top similar words to 4 query words.

	Opinion toward		All
Method	Target	Other	
CNN	71.07	**46.66**	67.33
RNN	72.49	44.48	67.82
SVM	74.54	43.20	68.98
Disc-TS	74.60	44.95	69.55
Disc-STS	76.36	46.44	**71.03**
Gen-STS	**76.53**	43.39	69.23

Table 4: Results for Task A (the official competition metric F_{avg}) on different subsets of the test data.

4.2.1 Preprocessing

We only considered the tweets which contain no URL, are not retweets, are not shorter than 40 characters, and have at most three hashtags and three mentions. Following the protocol of the task, we start from labeling some hashtags. Among the most frequent hashtags in the training data, we manually labeled a handful of hashtags that are favorable to Trump, e.g., *#MakeAmericaGreatAgain*, and a handful of hashtags that are against him, e.g., *#TrumpYoureFired*. See Table 5 for a complete list of these hashtags. This weakly supervised approach gives us a dataset with *noisy* labels; for example, the tweet "*his #MakeAmericaGreatAgain #Tag is a bummer.*" is against Trump, incorrectly labeled favorable. Tweets that have at least one positive, or one negative hashtag/regex, and do not have both a positive and a negative hashtag/regex, are considered as our initial *favorable* and *against* instances. The final weakly labeled dataset consisted of a modest number of 1367 instances (544 against and 823 favorable).

We use a sentiment analyzer for tweets, VADER (Hutto and Gilbert, 2014), to classify the sentiments of the tweets. Here, we are dealing with only one target, but we still classify the tweets based on their topics. To do this, we use the standard topic modeling technique, LDA (Blei et al., 2003). This gives us an approximate fine-grained view of the topics of discussion in the data (e.g., immigration, Mexico, Obama, etc.). The number of topics (potential targets) was determined by the Elbow method (Thorndike, 1953), which was found to be 4. Finally, the topic distributions for the tweets were binarized (i.e., one for the dimension with the maximum value and zero for others).

4.2.2 Results

In Table 6 we compare our results with the best system in Task 6.B, which is the same CNN (Wei et al., 2016) system in Task 6.A, and state-of-the-art model, BiCond (Augenstein et al., 2016), which uses a bidirectional conditional LSTM encoding model. To handle the "neither" class we do the following: if the absolute value of the difference between the probability values of the two classes is less than a random number ($\epsilon|\epsilon \in (0, 0.1]$), then we classify it as "neither".

Figure 3 shows the impact of the amount of the training data on the performance of our models. Due to the limited nature of our data collection scheme, which tends to exploit only parts of the space of the

Favor.	#makeamericagreatagain, #illegalimmigration, #boycottmacys, #trumpisright, #trumpsright, #benghazi, #liberal-logic, #illegalimmigrant, #patriot, #standwithtrump, #leftists, #trumpfortriumph, #gotrump, #nobama
Against.	#gopclowncar, #racist, #hateisnotpresidential, #mexicanpride, #narcissist, #trumpsucks, #boycotttrump, #hishair, #proudlatina, #proudmexican, #trumpyourefired, #donaldtrumpsucks, #dumptrump, #partyofhate

Table 5: Stance-indicative hashtags used to collect favorable and opposing tweets.

Method	F_{favor}	$F_{against}$	F_{avg}
CNN	57.39	55.17	56.28
BiCond	**61.38**	54.68	**58.03**
Gen-STS	57.08	**56.38**	56.73
Disc-STS	39.59	55.43	47.51

Table 6: Evaluation on SemEval-2016 Task 6.B.

data, it is reasonable to expect that after a certain amount of data is seen, the performance of the system improves marginally as more training data is added. The discriminative model converges more quickly and performs poorly. Its performance improves marginally after seeing only 10% of the training data (i.e., 137 instances) and deteriorates soon. On the other hand, the generative model converges later with a much better F1 score. We also added 5% misclassification noise to the stance labels in the task A dataset but did not observe a similar pattern; instead, the discriminative classifier performed consistently better than the generative one and was less sensitive to the noise.

What we see in Figure 3 can be ascribed to the fragmentary view of the data created because of the hashtag-based process of bootstrapping a training set. In other words, the small number of tweets, which we harvest, covers only part of the test-data distribution. This is worsened by the lack of neutral tweets in the bootstrapped training set. Previous works have shown that variants of generative models alone, or their combination with discriminative models (Larochelle and Bengio, 2008; Nigam et al., 2000), are useful for classification especially when the amount of training data is limited (NG and Jordan, 2002). A detailed analysis of this phenomenon will be undertaken in the future.

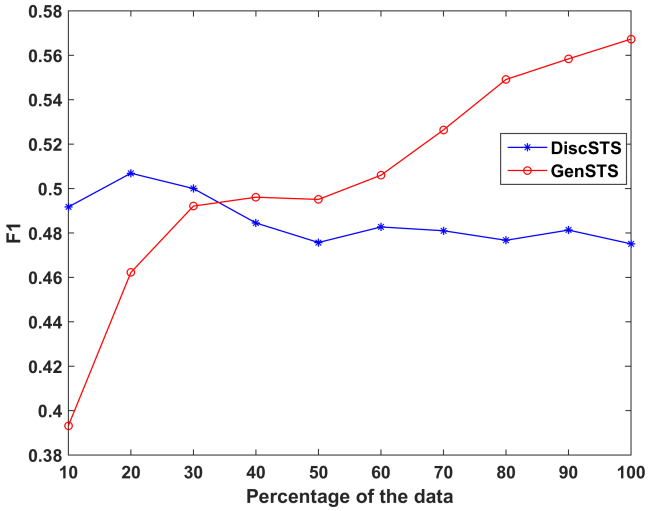


Figure 3: Comparison of GenSTS and DiscSTS on Task b. F1 is plotted against the amount of training data, i.e., percentage of the noisy-labeled data actually used for training. DiscSTS performs better initially and converges more quickly. GenSTS performs significantly better as more data is added.

5 Conclusion and Future Work

In this paper, we presented a log-linear approach for stance classification on tweets. The model employed sentiment and target variables in a novel way, wherein three-way interactions among input-sentiment-

stance variables and three-way interactions among input-target-stance variables were measured. Our findings show that the best way to use sentiments to improve stance classification is through these multi-way interactions. In addition, we demonstrated that by simply sharing regularization parameters among multiple targets, we are able to generalize features across multiple targets. While discriminative models are known to work better in classification tasks, generative models can also be useful when the data sample is small. Our results on a weakly labeled stance dataset proved that our generative model can in fact be much more effective than its discriminative counterpart.

For future work, our model can be easily incorporated in deep discriminative neural nets by replacing the standard softmax layer, effectively creating a multi-dimensional softmax layer. This has applications in tasks, wherein metadata exists; for example, a sentiment classification task for product reviews, in which metadata about the user and the products are also available. Moreover, the generative learning can be improved by replacing contrastive divergence with a more recent sampling method, SampleRank (Rohanimanesh et al., 2011), and using F1 score as the cost function.

6 Acknowledgement

This work was supported by NIH grant R01GM103309 and ARO grant W911NF-15-1-0265.

References

Rakesh Agrawal, Sridhar Rajagopalan, Ramakrishnan Srikant, and Yirong Xu. 2003. Mining newsgroups using networks arising from social behavior. In *Proceedings of WWW*, pages 529–535.

Pranav Anand, Marilyn Walker, Rob Abbott, Jean E Fox Tree, Robeson Bowmani, and Michael Minor. 2011. Cats rule and dogs drool!: Classifying stance in online debate. In *Proceedings of the 2nd workshop on computational approaches to subjectivity and sentiment analysis*, pages 1–9.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of EMNLP*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Clinton Burfoot, Steven Bird, and Timothy Baldwin. 2011. Collective classification of congressional floor-debate transcripts. In *Proceedings of ACL*, pages 1506–1515.

Lipika Dey and SK Mirajul Haque. 2009. Opinion mining from noisy text data. *International Journal on Document Analysis and Recognition*, 12(3):205–226.

Javid Ebrahimi, Dejing Dou, and Daniel Lowd. 2016. Weakly supervised tweet stance classification by relational bootstrapping. In *Proceedings of EMNLP*.

Adam Faulkner. 2014. Automated classification of stance in student essays: An approach using stance target information and the wikipedia link-based measure. In *Proceedings of FLAIRS*, pages 174–179.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of NAACL-HLT*, pages 12–17.

Kazi Saidul Hasan and Vincent Ng. 2013. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of IJCNLP*, pages 1348–1356.

Kazi Saidul Hasan and Vincent Ng. 2014. Why are you taking this stance? Identifying and classifying reasons in ideological debates. In *Proceedings of EMNLP*, pages 751–762.

Geoffrey Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Clayton J Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of ICWSM*, pages 216–225.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of ACL*, pages 151–160.

Hugo Larochelle and Yoshua Bengio. 2008. Classification using discriminative Restricted Boltzmann Machines. In *Proceedings of ICML*, pages 536–543.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of ICML*, pages 641–648.

Saif M Mohammad, Xiaodan Zhu, Svetlana Kiritchenko, and Joel Martin. 2015. Sentiment, emotion, purpose, and style in electoral tweets. *Information Processing & Management*, 51(4):480–499.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of SemEval*, pages 31–41.

Saif M Mohammad. 2016. A practical guide to sentiment annotation: Challenges and solutions. In *Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*.

Akiko Murakami and Rudy Raymond. 2010. Support or oppose?: classifying positions in online debates from reply activities and opinion expressions. In *Proceedings of COLING*, pages 869–875.

Andrew Y NG and Michael I Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Proceedings of NIPS*.

Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2-3):103–134.

Ashwin Rajadesingan and Huan Liu. 2014. Identifying users with opposing opinions in twitter debates. In *Proceedings of SBP*, pages 153–160.

Khashayar Rohanimanesh, Kedar Bellare, Aron Culotta, Andrew McCallum, and Michael L Wick. 2011. Samplerank: Training factor graphs with atomic gradients. In *Proceedings of ICML*, pages 777–784.

Ruslan R Salakhutdinov and Geoffrey E Hinton. 2009. Replicated softmax: an undirected topic model. In *Proceedings of NIPS*, pages 1607–1614.

Parinaz Sobhani, Diana Inkpen, and Stan Matwin. 2015. From argumentation mining to stance classification. In *Proceeding of the 2nd Workshop on Argumentation Mining*, pages 67–77.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124.

Dhanya Sridhar, James Foulds, Bert Huang, Lise Getoor, and Marilyn Walker. 2015. Joint models of disagreement and stance in online debate. In *Proceedings of ACL*, pages 116–125.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of EMNLP*, pages 327–335.

Robert L Thorndike. 1953. Who belongs in the family? *Psychometrika*, 18(4):267–276.

Marilyn A Walker, Pranav Anand, Rob Abbott, Jean E Fox Tree, Craig Martell, and Joseph King. 2012a. That is your evidence?: Classifying stance in online political debate. *Decision Support Systems*, 53(4):719–729.

Marilyn A Walker, Pranav Anand, Robert Abbott, and Ricky Grant. 2012b. Stance classification using dialogic properties of persuasion. In *Proceedings of NAACL-HLT*, pages 592–596.

Wan Wei, Xiao Zhang, Xuqin Liu, Wei Chen, and Tengjiao Wang. 2016. pkudblab at semeval-2016 task 6 : A specific convolutional neural network system for effective stance detection. In *Proceedings of SemEval*, pages 384–388.

Ainur Yessenalina, Yisong Yue, and Claire Cardie. 2010. Multi-level structured models for document-level sentiment classification. In *Proceedings of EMNLP*, pages 1046–1056.

Dani Yogatama and Noah A. Smith. 2014. Linguistic structured sparsity in text categorization. In *Proceedings of ACL*, pages 786–796.

Guido Zarrella and Amy Marsh. 2016. MITRE at semeval-2016 task 6: Transfer learning for stance detection. In *Proceedings of SemEval*, pages 458–463.

SenticNet 4: A Semantic Resource for Sentiment Analysis Based on Conceptual Primitives

Erik Cambria, Soujanya Poria, Rajiv Bajpai
Nanyang Technological University
50 Nanyang Ave, Singapore 639798
{cambria, sporia
rbajpai}@ntu.edu.sg

Björn Schuller
Imperial College London
180 Queens' Gate, Huxley Bldg.,
London SW7 2AZ, UK
bjoern.schuller@imperial.ac.uk

Abstract

An important difference between traditional AI systems and human intelligence is the human ability to harness commonsense knowledge gleaned from a lifetime of learning and experience to make informed decisions. This allows humans to adapt easily to novel situations where AI fails catastrophically due to a lack of situation-specific rules and generalization capabilities. Commonsense knowledge also provides background information that enables humans to successfully operate in social situations where such knowledge is typically assumed. Since commonsense consists of information that humans take for granted, gathering it is an extremely difficult task. Previous versions of SenticNet were focused on collecting this kind of knowledge for sentiment analysis but they were heavily limited by their inability to generalize. SenticNet 4 overcomes such limitations by leveraging on conceptual primitives automatically generated by means of hierarchical clustering and dimensionality reduction.

1 Introduction

The opportunity to capture the opinion of the general public has raised growing interest within both the scientific community as well as the business world, due to the remarkable benefits to be had from marketing and financial prediction, which has led to many exciting open challenges (Pang and Lee, 2008; Liu, 2012). Mining opinions and sentiments from natural language, however, is an extremely difficult task as it requires a deep understanding of most of the explicit and implicit, regular and irregular, syntactic and semantic rules of a language. Existing approaches to sentiment analysis mainly rely on parts of text in which opinions are explicitly expressed such as polarity terms, affect words, and their co-occurrence frequencies. However, opinions and sentiments are often conveyed implicitly through latent semantics, which make purely syntactic approaches ineffective.

SenticNet (Cambria et al., 2014) captures such latent information in terms of *semantics* and *sentics*, i.e., the denotative and connotative information commonly associated with real-world objects, actions, events, and people. SenticNet steps away from blindly using keywords and word co-occurrence counts, and instead relies on the implicit meaning associated with commonsense concepts. Superior to purely syntactic techniques, SenticNet can detect subtly expressed sentiments by enabling the analysis of multiword expressions that do not explicitly convey emotion, but are instead related to concepts that do so. The main limitation of SenticNet is that it is unable to generalize instances of concepts, e.g., `eat_pasta` or `slurp_noodles`: unless there is an exact match, SenticNet 3 raises a not-found error.

In SenticNet 4, however, both verb and noun concepts are linked to primitives so that, for example, concepts such as `eat_pasta` or `slurp_noodles` are generalized as `INGEST_FOOD`. In this way, most concept inflections can be captured by the knowledge base: verb concepts like `eat`, `slurp`, `munch` are all represented by their conceptual primitive `INGEST` while noun concepts like `pasta`, `noodles`, `steak` are replaced with their ontological parent `FOOD`.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2666–2677, Osaka, Japan, December 11-17 2016.

The idea behind this generalization is that there is a finite set of mental primitives for affect-bearing concepts and a finite set of principles of mental combination governing their interaction. Conceptual primitives are automatically discovered in SenticNet through the ensemble application of hierarchical clustering and dimensionality reduction.

The rest of the paper is organized as follows: Section 2 presents related work in the field of sentiment analysis; Section 3 proposes an excursus on conceptual primitives; Sections 4 and 5 describe in detail how noun concepts and verb concepts are generalized, respectively; Section 6 proposes experimental results on two different state-of-the-art datasets; finally, Section 7 provides concluding remarks.

2 Related Work

Sentiment analysis systems can be broadly categorized into knowledge-based or statistics-based systems (Cambria, 2016). While the use of knowledge bases was initially more popular for the identification of sentiment polarity in text, recently sentiment analysis researchers have been increasingly using statistics-based approaches, with a special focus on supervised statistical methods. Pang et al. (Pang et al., 2002) pioneered this trend by comparing the performance of different machine learning algorithms on a movie review dataset and obtained a 82% accuracy for polarity detection. A recent approach by Socher et al. (Socher et al., 2013) obtained a 85% accuracy on the same dataset using a recursive neural tensor network. Yu and Hatzivassiloglou (Yu and Hatzivassiloglou, 2003) used semantic orientation of words to identify polarity at sentence level. Melville et al. (Melville et al., 2009) developed a framework that exploits word-class association information for domain-dependent sentiment analysis.

More recent studies exploit microblogging text or Twitter-specific features such as emoticons, hashtags, URLs, @symbols, capitalizations, and elongations to enhance sentiment analysis of tweets. Tang et al. (Tang et al., 2014a) used a convolutional neural network (CNN) to obtain word embeddings for words frequently used in tweets. Dos Santos et al. (dos Santos and Gatti, 2014) also focused on deep CNN for sentiment detection in short texts. Recent approaches also focus on developing word embeddings based on sentiment corpora (Tang et al., 2014b). Such word vectors include more affective clues than regular word vectors and produce better results for tasks such as emotion recognition (Poria et al., 2016b) and aspect extraction (Poria et al., 2016a).

Statistical methods, however, are generally semantically weak (Cambria and White, 2014). This means that, with the exception of obvious affect keywords, other lexical or co-occurrence elements in a statistical model have little predictive value individually. As a result, statistical text classifiers only work with acceptable accuracy when given a sufficiently large text input. Hence, while these methods may be able to affectively classify user's text on the page or paragraph level, they do not work well on smaller text units such as sentences. Concept-level sentiment analysis, instead, focuses on a semantic analysis of text through the use of web ontologies or semantic networks, which allows for the aggregation of the conceptual and affective information associated with natural language opinions (Cambria and Hussain, 2015; Gezici et al., 2013; Araújo et al., 2014; Bravo-Marquez et al., 2014; Recupero et al., 2014).

By relying on large semantic knowledge bases, such approaches step away from the blind use of keywords and word co-occurrence counts, relying instead on the implicit features associated with natural language concepts. Unlike purely syntactic techniques, concept-based approaches are also able to detect sentiments expressed in a subtle manner; e.g., through the analysis of concepts that do not explicitly convey any emotion, but which are implicitly linked to other concepts that do so. The bag-of-concepts model can represent semantics associated with natural language much better than bag-of-words. In the latter, in fact, concepts like `pretty_ugly` or `sad_smile` would be split into two separate words, disrupting both semantics and sentics of the input sentence.

3 Conceptual Primitives

It is inherent to human nature to try to categorize things, events and people, finding patterns and forms they have in common. One of the most intuitive ways to relate two entities is through their similarity. According to Gestalt theory (Smith, 1988), similarity is one of six principles that guide human perception of the world.

Similarity is a quality that makes one thing or person like another and 'similar' means having characteristics in common. There are many ways in which objects can be perceived as similar, based on things like color, shape, size and texture. If we move away from mere visual stimuli, we can apply the same principles to define similarity between concepts based on shared semantic features. Previous versions of SenticNet exploited this principle to cluster natural language concepts sharing similar affective properties. Finding groups of similar concepts, however, does not ensure full coverage of all possible semantic inflections of multiword expressions.

In this work, we leverage on such similarities to deduce conceptual primitives that can better generalize SenticNet's commonsense knowledge. This generalization is inspired by different theories on conceptual primitives, including Roger Schank's conceptual dependency theory (Schank, 1972), Ray Jackendoff's work on explanatory semantic representation (Jackendoff, 1976), and Anna Wierzbicka's book on primes and universals (Wierzbicka, 1996), but also theoretical studies on knowledge representation (Minsky, 1975; Rumelhart and Ortony, 1977). All such theories claim that a decompositional method is necessary to explore conceptualization. In the same manner a physical scientist understands matter by breaking it down into progressively smaller parts, a scientific study of conceptualization proceeds by decomposing meaning into smaller parts. Clearly, this decomposition cannot go on forever: at some point we must find semantic atoms that cannot be further decomposed. This is the level of conceptual structure; mental representation that encodes basic understanding and commonsense by means of primitive conceptual elements out of which meanings are built.

In SenticNet, this 'decomposition' translates into the generalization of multiword expressions that convey a specific set of emotions and, hence, carry a particular polarity. The motivation behind this process of generalization is that there are countless ways to express the same concept in natural language and having a comprehensive list of all the possible concept inflections is almost impossible. While lexical inflections such as conjugation and declension can be solved with lemmatization, semantic inflections such as the use of synonyms or semantically-related concepts need to be tackled by analogical reasoning.

If multiword expressions like `attain_knowledge` and `acquire_know-how` are encountered in text, SenticNet 3 is unable to process them because there is no entry for such concepts in the knowledge base. SenticNet 3, however, does contain a multiword expression that is highly semantically related to those two concepts, that is, `acquire_knowledge`. By working at the primitive level, SenticNet 4 is able to bridge this semantic gap, as `attain_knowledge`, `acquire_know-how`, and `acquire_knowledge` are all represented by the same conceptual primitive: `GET_INFORMATION`.

By automatically inferring conceptual primitives for SenticNet concepts, we aim to broadly extend the coverage of the commonsense knowledge base and better perform sentiment analysis tasks such as polarity detection and emotion recognition from text. As shown in the next two sections, this is done via generalizing noun concepts by means of hierarchical clustering as well as discovering conceptual primitives for verb concepts by means of dimensionality reduction.

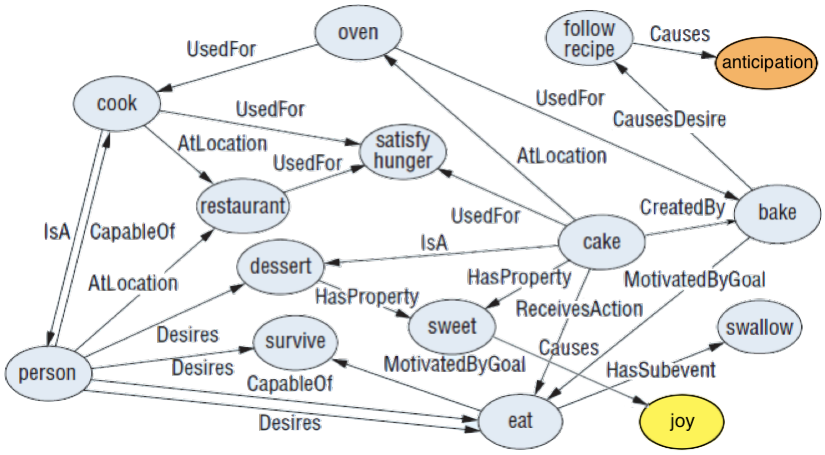


Figure 1: A sketch of the AffectNet graph showing part of the semantic network for the concept `cake`.

4 Noun Concept Generalization

The first step towards generalizing multiword expressions in SenticNet is to build a hierarchical classification of its noun concepts (or object concepts) so that nouns such as `cat`, `dog` or `pet` can be identified as `ANIMAL`. Such classification is implemented by applying hierarchical clustering on a semantic network of commonsense knowledge. It is important to note that each generalization inherits the emotional information and the polarity of its instance concepts. In the case of `cat` and `dog`, for example, the primitive is actually `ANIMAL+` since `cat` and `dog` are associated with positive emotions. Conversely, for animals that are associated with negative emotions such as fear (e.g., `white_shark`) or disgust (e.g., `cockroach`), the corresponding primitive is `ANIMAL−`.

4.1 AffectNet

AffectNet (Cambria and Hussain, 2015) is an affective commonsense knowledge base built upon ConceptNet (Speer and Havasi, 2012), the graph representation of the Open Mind corpus, and WordNet-Affect (Strapparava and Valitutti, 2004), a linguistic resource for the lexical representation of affect (Fig. 1). The resource is represented as a semantic network where nodes are multiword expressions of commonsense knowledge and the links between these are relations that interconnect them. The knowledge encoded by AffectNet is constantly expanding as new versions of ConceptNet are continuously released and new affective commonsense knowledge is crowdsourced through games. AffectNet is first converted into a matrix by dividing each assertion into two parts: a concept and a feature, where a feature is simply the assertion with the first or the second concept left unspecified such as 'a wheel is part of' or 'is a kind of liquid'.

The entries in the resulting matrix are positive or negative numbers, assigned according to the reliability of the assertions, with their magnitude increasing logarithmically with the confidence score. Because the AffectNet graph is made of triples based on the format <concept-relationship-concept>, the entire knowledge repository can be visualized as a large matrix, with every known concept of some statement being a row and every known semantic feature (relationship+concept) being a column. Such a representation has several advantages including the possibility to perform cumulative analogy (Tversky, 1977), executed by first selecting a set of nearest neighbors (in terms of similarity) of the input concept and then by projecting known properties of this set onto unknown properties of the concept (Table 1).

4.2 Group Average Agglomerative Clustering

Direct objects in verb+noun concepts, such as `buy_cake` or `eat_burger`, exhibit semantic coherence in that they tend to generate lexical items and phrases with related semantics. Most words related to the same verb tend to share some semantic characteristics. Our commonsense-based approach is similar to the process undertaken by humans when finding similar items – we look at what the *meanings* of the items have in common. In AffectNet, concepts inter-define one another, with directed edges indicating semantic dependencies between concepts.

Concepts	Semantic Features (*relationship+concept*)							
	..	*Causes* joy	*IsA* event	*UsedFor* housekeeping	*LocatedAt* party_venue	*PartOf* celebration	*MotivatedByGoal* clean_room	..
⋮		⋮	⋮	⋮	⋮	⋮	⋮	
`wedding`	..	0.94	0.86	0	0.79	0.88	0	..
`broom`	..	0	0	0.83	0	0	0.87	..
`buy_cake`	..	**?**	**0.78**	**0**	**0.80**	**0.91**	**0**	..
`birthday`	..	0.97	0.85	0	0.99	0.98	0	..
`sweep_floor`	..	0	0	0.79	0	0	0.91	..
⋮		⋮	⋮	⋮	⋮	⋮	⋮	

Table 1: Cumulative analogy allows for the inference of new pieces of knowledge by comparing similar concepts, e.g., `buy_cake` causes joy because `wedding` and `birthday` (which are similar) do so.

The traditional way to define features for any particular concept c in a semantic network is to consider the set of concepts reachable via outbound edges from c. The proposed algorithm exploits hierarchical clustering to generate from such features conceptual primitives, which represent the core semantics of each concept. Based on experiments with various clustering algorithms, e.g., k-means and expectation-maximization clustering, we determined that group average agglomerative clustering (GAAC) provides the highest accuracy. GAAC partitions data into trees (Berkhin, 2006) containing *child* and *sibling* clusters. It generates dendrograms specifying nested groupings of data at various levels (Jain and Dubes, 1988). During clustering, concepts are represented as vectors of commonsense features extracted from AffectNet. The proximity matrix is constructed with concepts as rows and features as columns. If a feature is an outbound link of a concept, the corresponding entry in the matrix is 1 and it is 0 in other situations. Cosine distance is used as the distance metric. Agglomerative algorithms are bottom-up in nature. GAAC, in particular, consists of the following steps:

1. Compute proximity matrix. Each data item is an initial cluster.
2. From the proximity matrix, form pair of clusters by merging. Update proximity matrix to reflect merges.
3. Repeat until all clusters are merged.

The resulting dendrogram is pruned at a height depending on the number of desired clusters. The *group average* between the clusters is given by the average similarity distance between the groups. Distances between two clusters and similarity measures are given by the equations below:

$$X_{sum} = \sum_{c_m \in \omega_i \cup \omega_i} \sum_{c_n \in \omega_i \cup \omega_j, c_n \neq c_m} \overrightarrow{c_n}.\overrightarrow{c_m} \tag{1}$$

$$sim\,(\omega_i, \omega_j) = \frac{1}{(N_i + N_j)\,(N_i + N_j - 1)} X_{sum} \tag{2}$$

where $\overrightarrow{c}$ is the vector of the concept of length c, vector entries are boolean (1 if the feature is present, 0 otherwise), and N_i, N_j is the number of features in ω_i and ω_j, respectively (which denote clusters). The main drawback of the hierarchical clustering algorithm is its running complexity (Berkhin, 2006), which averages $\theta(N^2 log\ N)$. We chose to utilize average link clustering as our clustering is connectivity-based. The concept proximity matrix consists of features from AffectNet and 'good' connections are deemed to occur when two concepts share multiple features. After clustering, the number of clusters is determined and the dendrogram is pruned accordingly.

Each cluster is later split into a positive and a negative sub-cluster. Cluster instances are assigned to either the positive or the negative sub-cluster depending on their polarity in SenticNet 3. For example, `cobra` and `cat` end up being in the same cluster (`ANIMAL`) after applying GAAC but, since they have opposite polarity in SenticNet 3, they are later assigned to different sub-clusters (`ANIMAL-` and `ANIMAL+`, respectively). Noun concepts for which no specific categorization is discovered are grouped under one of three most general noun primitives, namely: `SOMETHING`, `SOMEONE`, and `SOMEWHERE` (also divided into positive and negative sub-clusters). Table 2 provides an example of the results of polarity-driven feature-based clustering for 24 noun concepts.

SOMETHING- ANIMAL-	SOMETHING+ ANIMAL+	SOMEONE- PROFESSIONAL-	SOMEONE+ PROFESSIONAL+	SOMEWHERE- NATURE-	SOMEWHERE+ NATURE+
cockroach	horse	gravedigger	doctor	dry_steppe	oasis
rat	cat	coroner	scientist	desert	sandy_beach
cobra	puppy	executioner	teacher	wild_forest	natural_park
termite	pet	mortician	sea_captain	polar_desert	seaside

Table 2: Example of feature-based clustering for polarity-driven conceptual primitive inference.

5 Verb Concept Generalization

The second step in generalizing SenticNet concepts is to define conceptual primitives for verb concepts (or action concepts) so that, for example, verbs like `acquire`, `attain` or `collect` can be identified as `GET`. Such classification is implemented by applying dimensionality reduction techniques on the vector space representation of AffectNet. As with noun concepts, verb concepts are also associated with a polarity but, in this case, polarity is more relevant to the opposite meanings (or outcomes) these action concepts represent, as in `INCREASE` versus `DECREASE`. This allows for reasoning about verb+noun combinations to be as per algebraic multiplication, where negative multiplied by positive (or vice versa) results in a negative, e.g., `DECREASE_GAIN` (or `INCREASE_LOSS`), multiplying two positives produces a positive, e.g., `INCREASE_PLEASURE`, and negative multiplied by negative results in a positive, e.g., `DECREASE_PAIN`.

5.1 AffectiveSpace

The human mind constructs intelligible meanings by continuously compressing over vital relations (Fauconnier and Turner, 2003). The compression principles aim to transform diffuse and distended conceptual structures into more focused versions so they can become more congenial for human understanding. In order to emulate such a process, we use simple but powerful meta-algorithms which underlie neuronal learning (Lee et al., 2011). These meta-algorithms should be fast, scalable, effective, with few-to-no specific assumptions and biologically plausible. Optimizing all the $\approx 10^{15}$connections formed through the last few million years of evolution is very unlikely. Objectively speaking, however, nature probably only optimizes the global connectivity (mainly the white matter) but leaves the other details to randomness (Balduzzi, 2013).

In this work, we use random projections (Bingham and Mannila, 2001) on the matrix representation of AffectNet in order to compress the semantic features associated with commonsense concepts and, hence, better perform analogical reasoning on these. Random projections are a data-oblivious method to map an original high-dimensional dataset into a much lower-dimensional subspace by using a Gaussian $\mathrm{N}(0,1)$ matrix, while at the same time, preserving pair-wise distances with high probability. This theoretically-solid and empirically-verified statement follows Johnson and Lindenstrauss's Lemma (Balduzzi, 2013), which states that, with high probability, for all pairs of points $\mathrm{x}, \mathrm{y} \in \mathrm{X}$ simultaneously:

$$\sqrt{\frac{m}{d}} \parallel x - y \parallel_2 (1-\varepsilon) \leq \parallel \Phi x - \Phi y \parallel_2 \leq \sqrt{\frac{m}{d}} \parallel x - y \parallel_2 (1+\varepsilon) \tag{3}$$

where X is a set of vectors in Euclidean space, d is the original dimension of this Euclidean space, m is the dimension of the space we wish to reduce the data points to, ε is a tolerance parameter measuring the maximum allowed distortion extent rate of the metric space, and Φ is a random matrix. Structured random projections for making matrix multiplication much faster was introduced in (Sarlos, 2006). When the number of features is much larger than the number of training samples ($d \gg n$), subsampled randomized Hadamard transform (SRHT) is preferred, as it behaves very much like Gaussian random matrices but accelerates the process from $\mathcal{O}(n\,d)$ to $\mathcal{O}(n\,log\,d)$ time (Lu et al., 2013). Following (Tropp, 2011; Lu et al., 2013), for $d = 2^p$ (where p is any positive integer), a SRHT can be defined as:

$$\Phi = \sqrt{\frac{d}{m}} \mathrm{RHD} \tag{4}$$

where • m is the number we want to subsample from d features randomly;

• R is a random $m \times d$ matrix. The rows of R are m uniform samples (without replacement) from the standard basis of $\mathbb{R}^d$;

• H$\in \mathbb{R}^{d \times d}$ is a normalized Walsh-Hadamard matrix, which is defined recursively:

$H_d = \begin{bmatrix} H_{d/2} & H_{d/2} \\ H_{d/2} & -H_{d/2} \end{bmatrix}$ with $H_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}$;

• D is a $d \times d$ diagonal matrix and the diagonal elements are i.i.d. Rademacher random variables.

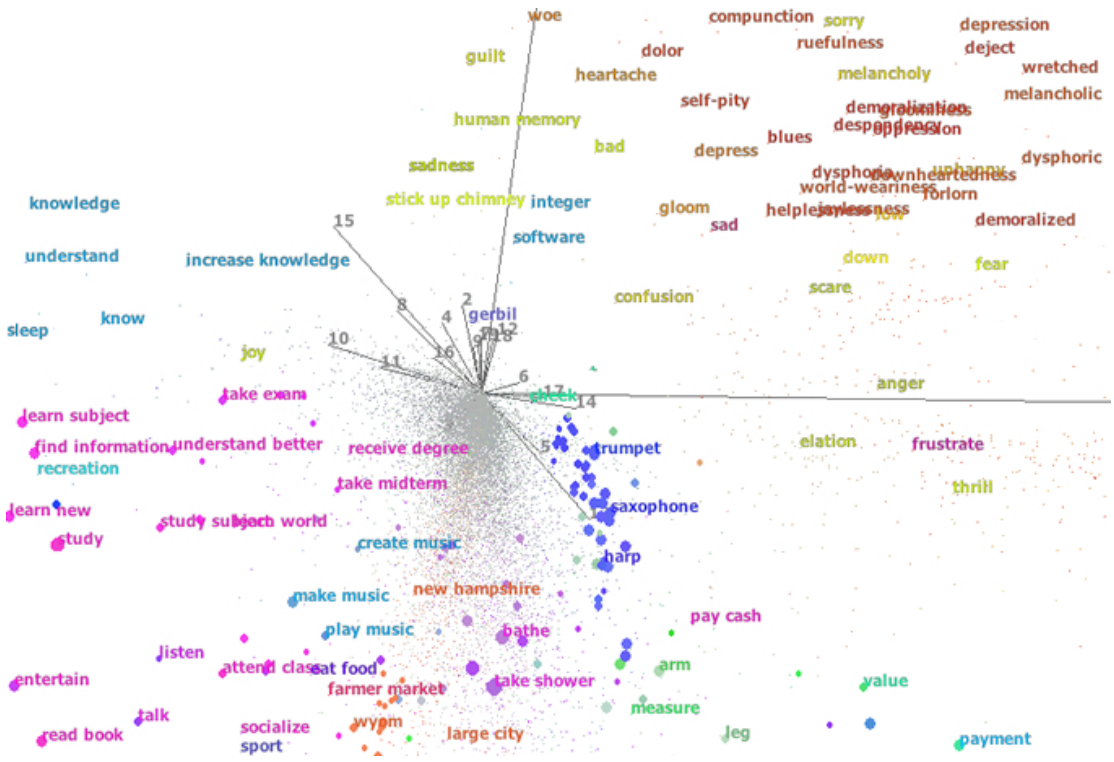


Figure 2: In AffectiveSpace, commonsense concepts gravitate around positive and negative emotions.

Our subsequent analysis only relies on the distances and angles between pairs of vectors (i.e., the Euclidean geometry information) and it is sufficient to set the projected space to be logarithmic in the size of the data (Ailon and Chazelle, 2010) and, hence, apply SRHT. The result is AffectiveSpace (Cambria et al., 2015), a vector space model where commonsense concepts and emotions are represented by vectors of m coordinates (Fig. 2).

By exploiting the information sharing property of random projections, concepts with the same semantic and affective valence are likely to have similar features – that is, concepts conveying the same meaning and emotion tend to fall near each other in AffectiveSpace. Similarity does not depend on concepts' absolute position in the vector space, but rather on the angle these make with the origin. For example, concepts such as `birthday_party`, `celebrate`, and `buy_cake` are found very closely positioned in the vector space, while concepts like `lose_faith`, `depressed`, and `shed_tear` are found in a completely different direction (nearly opposite with respect to the centre of the space).

5.2 Semi-Supervised Verb Propagation

AffectiveSpace allows for analogical reasoning about multiword expressions so that concepts such as `buy_groceries` and `go_shopping` will be detected as being semantically similar. In order to generalize verb concepts, however, we need to discriminate such reasoning according to actions, so that a concept like `buy_groceries` would be associated with concepts related to the verb `buy`, e.g., `buy_milk` or `purchase_vegetable`.

To this end, we leverage on VerbNet (Schuler, 2005), the largest English verb lexicon currently available, and Sentic LDA (Poria et al., 2016c), a classification framework that integrates commonsense in the calculation of word distributions in the linear discriminant analysis (LDA) algorithm. In particular, we use a semi-supervised version of Sentic LDA in order to incorporate both supervised (VerbNet-labeled) and unsupervised information in such a way that a proper semantic space which reflects the desired information (verb concepts) is obtained. Given a set of verbs and a large amount of unlabeled instances in AffectiveSpace, the between-class scatter is to be maximized and the within-class scatter of VerbNet instances is to be minimized, keeping the semantic relatedness of all the other instances simultaneously.

Each instance is denoted as $v_i \in \mathcal{A}^m$, which is the m-dimensional vector after being processed by random projections. For each verb instance, there is a label $y_i \in \{1, \ldots, q\}$, where q is the number of verb classes. Then, the between-class scatter and the within-class scatter matrices are defined as follows:

$$S_w = \sum_{j=1}^{q} \sum_{i=1}^{l_j} (v_i - \mu_j)(v_i - \mu_j)^T \tag{5}$$

$$S_b = \sum_{j=1}^{q} l_j (\mu_j - \mu)(\mu_j - \mu)^T \tag{6}$$

where $\mu_j = \frac{1}{l_j} \sum_{i=1}^{l_j} v_i$ $(j = 1, 2, ..., q)$ is the mean of the samples in class j, l_j is the number of verb instances in class j and $\mu = \frac{1}{l} \sum_{i=1}^{l} v_i$ is the mean of all the labeled samples. A total scatter matrix on all the instances in AffectiveSpace is also defined:

$$S_t = \sum_{i=1}^{k} (v_i - \mu_k)(v_i - \mu_k)^T \tag{7}$$

where k is the total number of instances in AffectiveSpace and μ_k is the mean of all the instances. Our objective is then to find a projection matrix W to project the semantic space to a lower-dimensional space, which is more discriminative towards verb concepts:

$$W^* = \arg \max_{W \in \mathcal{A}^{m \times m'}} \frac{|W^T S_b W|}{|W^T (S_w + \lambda_1 S_t + \lambda_2 I) W|} \tag{8}$$

where I is identity matrix, and λ_1 and λ_2 are control parameters, obtained through a grid search, for balancing the trade-off between verb discriminant and semantic regularizations. The optimal solution is given by:

$$(S_w + \lambda_1 S_t + \lambda_2 I) w_j^* = \eta_j S_b w_j^* \quad j = 1, ..., m' \tag{9}$$

where w_j^* $(j = 1, ..., m')$ are the eigenvectors corresponding to the m' largest eigenvalues of $(S_w + \lambda_1 S_t + \lambda_2 I)^{-1} S_b$. Here, $m' = q - 1$ is selected, where q is the total verb primitive number. After the projection, the new space preserves both semantic relatedness and action concept grouping based on the information coming from AffectNet and VerbNet, respectively.

6 Evaluation

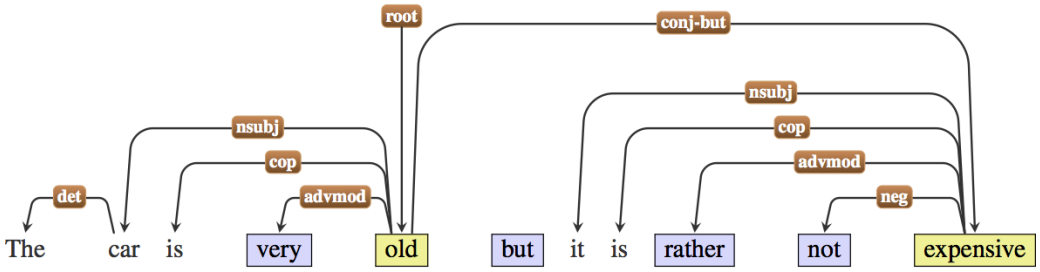
In order to perform a qualitative evaluation of SenticNet 4 (available both as a standalone XML repository[1] and as an API[2]), we asked five annotators to judge the plausibility of inferred conceptual primitives. We obtained an overall accuracy of 91% with Cohen's kappa score of 0.84. As for the quantitative evaluation, we tested SenticNet 4 against two well-known sentiment resources, namely: the Blitzer Dataset (Blitzer et al., 2007) and the Movie Review Dataset (Pang and Lee, 2005).

6.1 Performing Polarity Detection with SenticNet

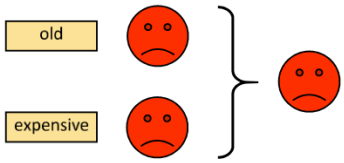
While SenticNet can be used as any other sentiment lexicon, e.g., concept matching or bag-of-concepts model, the right way to use the knowledge base for the task of polarity detection is in conjunction with sentic patterns (Poria et al., 2015). Sentic patterns are sentiment-specific linguistic patterns that infer polarity by allowing affective information to flow from concept to concept based on the dependency relation between clauses. The main idea behind such patterns can be best illustrated by analogy with an electronic circuit, in which few 'elements' are 'sources' of the charge or signal, while many elements operate on the signal by transforming it or combining different signals. This implements a rudimentary type of semantic processing, where the 'meaning' of a sentence is reduced to only one value: its polarity.

[1] `http://sentic.net/senticnet-4.0.zip`

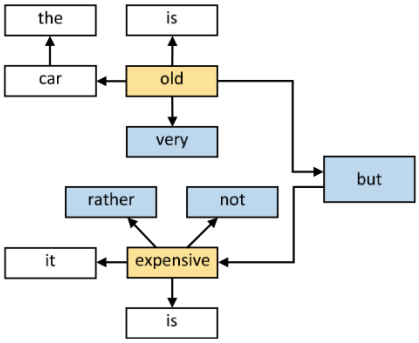
[2] `http://sentic.net/api`



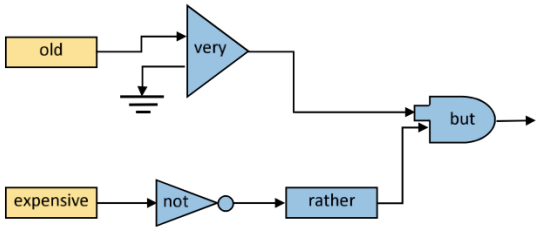
(a) Dependency tree of a sentence.



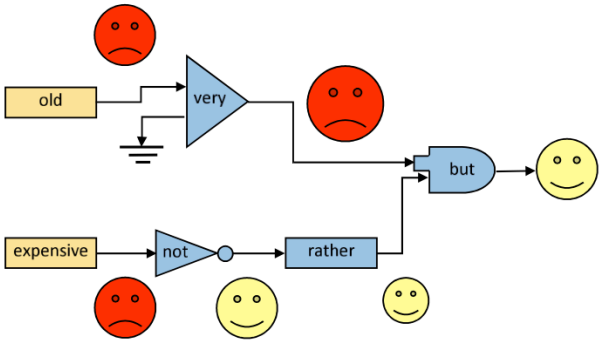
(b) The old way: averaging over a bag of sentiment words. The overall polarity of a sentence is given by the algebraic sum of the polarity values associated with each affect word divided by the total number of words.



(c) The dependency tree of a sentence resembles an electronic circuit: words shown in blue can be thought as a sort of "boolean operations" acting on other words.



(d) The electronic circuit metaphor: sentiment words are "sources" while other words are "elements", e.g., *very* is an amplifier, *not* is a logical complement, *rather* is a resistor, *but* is an OR-like element that gives preference to one of its inputs.



(e) The final sentiment data flow of the "signal" in the "circuit".

Figure 3: In sentic patterns, the structure of a sentence is like an electronic circuit where logical operators channel sentiment data-flows to output an overall polarity.

Sentic patterns are applied to the dependency syntactic tree of a sentence, as shown in Figure 3(a). The only two words that have intrinsic polarity are shown in yellow color; the words that modify the meaning of other words in the manner similar to contextual valence shifters (Polanyi and Zaenen, 2006) are shown in blue. A baseline that completely ignores sentence structure, as well as words that have no intrinsic polarity, is shown in Figure 3(b): the only two words left are negative and, hence, the total polarity is negative. However, the syntactic tree can be re-interpreted in the form of a 'circuit' where the 'signal' flows from one element (or subtree) to another, as shown in Figure 3(c). After removing the words not used for polarity calculation (in white), a circuit with elements resembling electronic amplifiers, logical complements, and resistors is obtained, as shown in Figure 3(d),

Figure 3(e) illustrates the idea at work: the sentiment flows from polarity words through shifters and combining words. The two polarity-bearing words in this example are negative. The negative effect of the word 'old' is amplified by the intensifier 'very'. However, the negative effect of the word 'expensive' is inverted by the negation, and the resulting positive value is decreased by the 'resistor'. Finally, the values of the two phrases are combined by the conjunction 'but', so that the overall polarity has the same sign as that of the second component (positive).

6.2 SenticNet 4 vs. SenticNet 3

The Blitzer Dataset consists of product reviews in seven different domains. For each domain there are 1,000 positive and 1,000 negative reviews. In evaluating SenticNet 4, we only used reviews under the *electronics* category. From these, we randomly extracted 7,210 non-neutral sentences: 3,800 of which were marked as positive and 3,410 as negative. We then compared the performance of SenticNet 4 with its predecessor SenticNet 3 for the task of sentence-level polarity detection, using sentic patterns. The results are shown in Table 3.

Table 3: Comparison on the Blitzer Dataset

Framework	Accuracy
Sentic Patterns and SenticNet 3	87.0%
Sentic Patterns and SenticNet 4	91.3%

6.3 SenticNet 4 vs. Statistical Methods

The Movie Review Dataset includes 1,000 positive and 1,000 negative movie reviews collected from Rotten Tomatoes[3]. Originally, Pang and Lee manually labeled each review as positive or negative. Later, Socher et al. (Socher et al., 2012; Socher et al., 2013) annotated this dataset at sentence level. They extracted 11,855 sentences from the reviews and manually labeled them using a fine-grained inventory of five sentiment labels: *strong positive*, *positive*, *neutral*, *negative*, and *strong negative*. Since in this work we consider only binary classification, we removed neutral sentences from the dataset and merged germane labels. Thus, the final dataset contained 4,800 positive sentences and 4,813 negative ones. The results of the classification with SenticNet 3 and SenticNet 4 are shown in Table 4.

Table 4: Comparison on the Movie Review Dataset

Framework	Accuracy
Socher et al., 2012	80.0%
Socher et al., 2013	85.4%
Sentic Patterns and SenticNet 3	86.2%
Sentic Patterns and SenticNet 4	90.1%

7 Conclusion

The distillation of knowledge from the huge amount of unstructured information on the Web is a key factor for tasks such as social media marketing, brand positioning, and financial prediction. Commonsense reasoning is a good solution for sentiment analysis but the scalability of commonsense knowledge bases is a major factor that jeopardizises the efficiency of concept extraction and polarity detection. A first possible step in solving this problem is to generalize pieces of commonsense knowledge in terms of conceptual primitives that could catch most semantic inflections of natural language concepts.

In this work, we used an ensemble of hierarchical clustering and dimensionality reduction for automatically discovering the primitives for both noun and verb concepts in SenticNet. This generalization process allowed us to largely extend the coverage of the commonsense knowledge base and, hence, to boost the accuracy of SenticNet for sentence-level polarity detection in comparison with both the previous version of the resource and with state-of-the-art statistical sentiment analysis research.

In the future, we plan to discover new conceptual primitives in a more automatic and scalable way by means of dependency-based word embeddings. In particular, we will exploit the internally-learned context embeddings of the skip-gram model in conjunction with the standard target word embeddings, to weigh context compatibility together with word similarity.

[3] `http://rottentomatoes.com`

References

Nir Ailon and Bernard Chazelle. 2010. Faster dimension reduction. *Communications of the ACM*, 53(2):97–104.

Matheus Araújo, Pollyanna Gonçalves, Meeyoung Cha, and Fabrício Benevenuto. 2014. iFeel: A system that compares and combines sentiment analysis methods. In *WWW*, pages 75–78.

David Balduzzi. 2013. Randomized co-training: from cortical neurons to machine learning and back again. *arXiv preprint arXiv:1310.6536*.

Pavel Berkhin. 2006. A survey of clustering data mining techniques. *Grouping multidimensional data*, pages 25–71.

Ella Bingham and Heikki Mannila. 2001. Random projection in dimensionality reduction: applications to image and text data. In *ACM SIGKDD*, pages 245–250.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447.

Felipe Bravo-Marquez, Marcelo Mendoza, and Barbara Poblete. 2014. Meta-level sentiment models for big social data analysis. *Knowledge-Based Systems*, 69:86–99.

Erik Cambria and Amir Hussain. 2015. *Sentic Computing: A Common-Sense-Based Framework for Concept-Level Sentiment Analysis*. Springer, Cham, Switzerland.

Erik Cambria and Bebo White. 2014. Jumping NLP curves: A review of natural language processing research. *IEEE Computational Intelligence Magazine*, 9(2):48–57.

Erik Cambria, Daniel Olsher, and Dheeraj Rajagopal. 2014. SenticNet 3: A common and common-sense knowledge base for cognition-driven sentiment analysis. In *AAAI*, pages 1515–1521, Quebec City.

Erik Cambria, Jie Fu, Federica Bisio, and Soujanya Poria. 2015. AffectiveSpace 2: Enabling affective intuition for concept-level sentiment analysis. In *AAAI*, pages 508–514, Austin.

Erik Cambria. 2016. Affective computing and sentiment analysis. *IEEE Intelligent Systems*, 31(2):102–107.

Cıcero Nogueira dos Santos and Maıra Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78.

Gilles Fauconnier and Mark Turner. 2003. *The Way We Think: Conceptual Blending and the Mind's Hidden Complexities*. Basic Books.

Gizem Gezici, Rahim Dehkharghani, Berrin Yanikoglu, Dilek Tapucu, and Yucel Saygin. 2013. Su-sentilab: A classification system for sentiment analysis in twitter. In *International Workshop on Semantic Evaluation*, pages 471–477.

Ray Jackendoff. 1976. Toward an explanatory semantic representation. *Linguistic Inquiry*, 7(1):89–150.

Anil Jain and Richard Dubes. 1988. *Algorithms for clustering data*. Prentice-Hall, Inc.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and A. Y. Ng. 2011. Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Communications of the ACM*, 54(10):95–103.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Morgan and Claypool.

Yichao Lu, Paramveer Dhillon, Dean P Foster, and Lyle Ungar. 2013. Faster ridge regression via the subsampled randomized hadamard transform. In *Advances in Neural Information Processing Systems*, pages 369–377.

Prem Melville, Wojciech Gryc, and Richard D Lawrence. 2009. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *ACM SIGKDD*, pages 1275–1284.

Marvin Minsky. 1975. A framework for representing knowledge. In Patrick Winston, editor, *The psychology of computer vision*. McGraw-Hill, New York.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124, Ann Arbor.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2:1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *EMNLP*, volume 10, pages 79–86.

Livia Polanyi and Annie Zaenen. 2006. Contextual valence shifters,. In *Computing Attitude and Affect in Text: Theory and Applications*, volume 20 of *The Information Retrieval Series*, pages 1–10. Springer, Berlin, Germany.

Soujanya Poria, Erik Cambria, Alexander Gelbukh, Federica Bisio, and Amir Hussain. 2015. Sentiment data flow analysis by means of dynamic linguistic patterns. *IEEE Computational Intelligence Magazine*, 10(4):26–36.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2016a. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, 108:42–49.

Soujanya Poria, Erik Cambria, Newton Howard, Guang-Bin Huang, and Amir Hussain. 2016b. Fusing audio, visual and textual clues for sentiment analysis from multimodal content. *Neurocomputing*, 174:50–59.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Federica Bisio. 2016c. Sentic LDA: Improving on LDA with semantic similarity for aspect-based sentiment analysis. In *IJCNN*.

Diego Reforgiato Recupero, Valentina Presutti, Sergio Consoli, Aldo Gangemi, and Andrea Nuzzolese. 2014. Sentilo: Frame-based sentiment analysis. *Cognitive Computation*, 7(2):211–225.

David Rumelhart and Andrew Ortony. 1977. The representation of knowledge in memory. In C Anderson, R Spiro, and W Montague, editors, *Schooling and the acquisition of knowledge*. Erlbaum, Hillsdale, NJ.

Tamas Sarlos. 2006. Improved approximation algorithms for large matrices via random projections. In *FOCS*, pages 143–152.

Roger Schank. 1972. Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology*, 3:552–631.

Karin Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, University of Pennsylvania, Computer and Information Science.

Barry Smith, editor. 1988. *Foundations of Gestalt Theory*. Munich and Vienna: Philosophia Verlag.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*, pages 1201–1211.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1642–1654.

Robert Speer and Catherine Havasi. 2012. ConceptNet 5: A large semantic network for relational knowledge. In Eduard Hovy, M Johnson, and G Hirst, editors, *Theory and Applications of Natural Language Processing*, chapter 6. Springer.

Carlo Strapparava and Alessandro Valitutti. 2004. WordNet-Affect: An affective extension of WordNet. In *LREC*, pages 1083–1086, Lisbon.

Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. 2014a. Coooolll: A deep learning system for twitter sentiment classification. In *SemEval*, pages 208–212.

Duyu Tang, Furu Wei, Nan Yang, Ting Liu, and Bing Qin. 2014b. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*, pages 1555–1565.

Joel A Tropp. 2011. Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis*, 3(01n02):115–126.

Amos Tversky. 1977. Features of similarity. *Psychological Review*, 84(4):327–352.

Anna Wierzbicka. 1996. *Semantics: Primes and Universals*. Oxford University Press.

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *EMNLP*, pages 129–136.

Joint Embedding of Hierarchical Categories and Entities for Concept Categorization and Dataless Classification

Yuezhang Li, Ronghuo Zheng, Tian Tian, Zhiting Hu, Rahul Iyer, Katia Sycara
Language Technology Institute, School of Computer Science
Carnegie Mellon University
yuezhanl@andrew.cmu.edu

Abstract

Existing work learning distributed representations of knowledge base entities has largely failed to incorporate rich categorical structure, and is unable to induce category representations. We propose a new framework that embeds entities and categories jointly into a semantic space, by integrating structured knowledge and taxonomy hierarchy from large knowledge bases. Our framework enables to compute meaningful semantic relatedness between entities and categories in a principled way, and can handle both single-word and multiple-word concepts. Our method shows significant improvement on the tasks of concept categorization and dataless hierarchical classification.

1 Introduction

Hierarchies, most commonly represented as tree or directed acyclic graph (DAG) structures, provide a natural way to categorize and locate knowledge in large knowledge bases (KBs) . For example, WordNet (Fellbaum, 1998), Freebase (Bollacker et al., 2008), and Wikipedia [1] use hierarchical taxonomy to organize entities into category hierarchies. These hierarchical categories could benefit applications such as object and concept categorization (Verma et al., 2012; Rothenhäusler and Schütze, 2009), document classification (Gopal and Yang, 2013; Hu et al., 2016), and link prediction on knowledge graphs (Lin et al., 2015; Zhang et al., 2016). In all of these applications, it is essential to have a good representation of categories and entities as well as a good semantic relatedness measure.

In this paper, we propose two models to learn distributed representations of both entities and categories from large-scale knowledge bases (KBs). The **Category Embedding (CE) model** extends the entity embedding method of (Hu et al., 2015) and induces category embeddings in addition to entity vectors. The **Hierarchical Category Embedding (HCE) model** further enhances the CE model by integrating category hierarchical structure. The resulting entity and category vectors effectively capture meaningful semantic relatedness between entities and categories.

We train the category and entity vectors on Wikipedia, and evaluate our methods from two applications: **concept categorization** (Baroni and Lenci, 2010) and **dataless hierarchical classification** (Song and Roth, 2014). Our method achieves state-of-the-art performance on both tasks.

Figure 1 shows an overview of the research elements in this paper. In summary, we make the following contributions:

- We incorporate category information into entity embeddings with the proposed CE model to get entity and category embeddings simultaneously.
- We leverage category hierarchies and develop the HCE model to enhance the embedding quality.
- We propose a new concept categorization method based on nearest neighbor classification. Our method avoids the granularity disparity issue of categories which has plagued traditional clustering-based methods.

This work has been partially funded by ARO award #W911NF1210416
[1] https://www.wikipedia.org

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2678–2688, Osaka, Japan, December 11-17 2016.

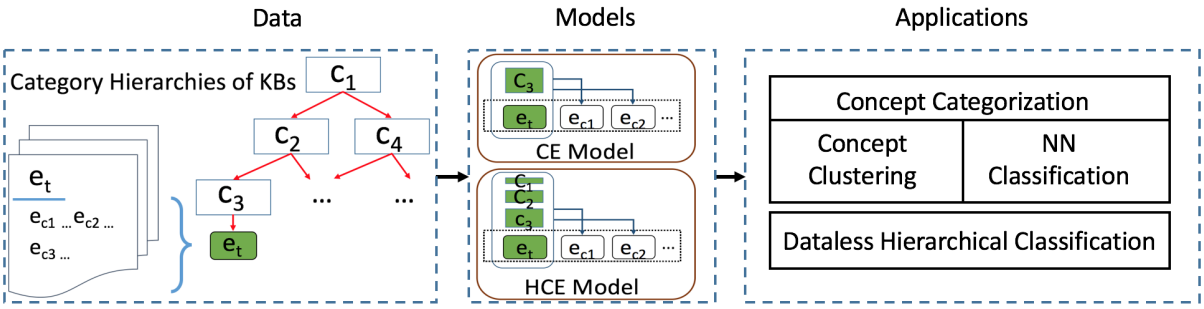


Figure 1: The organization of research elements comprising this paper.

- We construct a new concept categorization dataset from Wikipedia.
- We show strong potential of utilizing entity embeddings on dataless classification.

2 Related Work

Entity embedding method is based on the analysis of distributional semantics, e.g., the recently proposed skip-gram model (Mikolov et al., 2013) that learns to predict context words given target words. This model tries to maximize the average log likelihood of the context words so that the embeddings encode meaningful semantics. For instance, entity hierarchy embedding (Hu et al., 2015) extends it to predict context entities given target entity in KBs. (Yamada et al., 2016) proposed a method to jointly embed words and entities by optimizing word-word, entity-word, and entity-entity predicting models. Our models extend this line of research by incorporating hierarchical category information to embed categories and entities in a joint semantic space.

Relational embedding, also known as knowledge graph embedding, is a family of methods to represent entities as vectors and relations as operations applied to entities such that certain properties are preserved (Paccanaro and Hinton, 2001; Bordes et al., 2011; Nickel et al., 2012; Bordes et al., 2013; Neelakantan and Chang, 2015). For instance, the linear relational embedding (Paccanaro and Hinton, 2001) applies a relation to an entity based on matrix-vector multiplication, while TransE (Bordes et al., 2013) simplifies the operation to vector addition. To derive the embedding representation, they minimize a global loss function considering all (entity, relation, entity) triplets so that the embeddings encode meaningful semantics. Our approach differs from this line as we learn from raw text in addition to knowledge graphs, and methodologically use probabilistic models instead of transition-based models.

Category embedding has been widely explored within different context. (Weinberger and Chapelle, 2009) initially proposed a taxonomy embedding to achieve document categorization and derive the embeddings of taxonomies of topics in form of topic prototypes. Furthermore, (Hwang and Sigal, 2014) proposed a discriminative learning framework that can give category embeddings by approximating each category embeddings as a sum of its direct super-category plus a sparse combination of attributes. These methods have primarily targeted on documents/object classification rather than entity representations.

Several recent methods are proposed to extend word representation to phrases (Yin and Schütze, 2014; Yu and Dredze, 2015; Passos et al., 2014). However, they do not use structured knowledge to derive phrase representations.

3 Joint Embedding of Categories and Entities

In order to find representations for categories and entities that can capture their semantic relatedness, we use existing hierarchical categories and entities labeled with these categories, and explore two methods: 1) **Category Embedding model** (CE Model): it replaces the entities in the context with their directly labeled categories to build categories' context; 2) **Hierarchical Category Embedding** (HCE Model): it further incorporates all ancestor categories of the context entities to utilize the hierarchical information.

3.1 Category Embedding (CE) Model

Our category embedding (CE) model is based on the Skip-gram word embedding model(Mikolov et al., 2013). The skip-gram model aims at generating word representations that are good at predicting *context* words surrounding a *target* word in a sliding window. Previous work (Hu et al., 2015) extends the entity's context to the whole article that describes the entity and acquires a set of entity pairs $\mathbf{D} = \{(e_t, e_c)\}$, where e_t denotes the *target* entity and e_c denotes the *context* entity.

Our CE model extends those approaches by incorporating category information. In KBs such as Wikipedia, category hierarchies are usually given as DAG or tree structures, and entities are categorized into one or more categories as leaves. Thus, in KBs, each entity e_t is labeled with one or more categories $(c_1, c_2, ..., c_k), k \geq 1$ and described by an article containing other *context* entities (see Data in Figure 1).

To learn embeddings of entities and categories simultaneously, we adopt a method that incorporates the labeled categories into the entities when predicting the context entities, similar to TWE-1 model (Liu et al., 2015) which incorporates topic information with words to predict context words. For example, if e_t is the *target* entity in the document, its labeled categories $(c_1, c_2, ..., c_k)$ would be combined with the entity e_t to predict the context entities like e_{c1} and e_{c2} (see CE Model in Figure 1). For each target-context entity pair (e_t, e_c), the probability of e_c being context of e_t is defined as the following softmax:

$$P(e_c|e_t) = \frac{\exp(e_t \cdot e_c)}{\sum_{e \in \mathbf{E}} \exp(e_t \cdot e)}, \tag{1}$$

where $\mathbf{E}$ denotes the set of all entity vectors, and $\exp$ is the exponential function. For convenience, here we abuse the notation of e_t and e_c to denote a target entity vector and a context entity vector respectively.

Similar to TWE-1 model, We learn the target and context vectors by maximizing the average log probability:

$$L = \frac{1}{|\mathbf{D}|} \sum_{(e_c, e_t) \in \mathbf{D}} \Big[\log P(e_c|e_t) + \sum_{c_i \in \mathbf{C}(e_t)} \log P(e_c|c_i) \Big], \tag{2}$$

where $\mathbf{D}$ is the set of all entity pairs and we abuse the notation of c_i to denote a category vector, and $\mathbf{C}(e_t)$ denotes the categories of entity e_t.

3.2 Hierarchical Category Embedding(HCE) Model

From the design above, we can get the embeddings of all categories and entities in KBs without capturing the semantics of hierarchical structure of categories. In a category hierarchy, the categories at lower layers will cover fewer but more specific concepts than categories at upper layers. To capture this feature, we extend the CE model to further incorporate the ancestor categories of the target entity when predicting the context entities (see HCE Model in Figure 1). If a category is near an entity, its ancestor categories would also be close to that entity. On the other hand, an increasing distance of the category from the target entity would decrease the power of that category in predicting context entities. Therefore, given the target entity e_t and the context entity e_c, the objective is to maximize the following weighted average log probability:

$$L = \frac{1}{|\mathbf{D}|} \sum_{(e_c, e_t) \in \mathbf{D}} \Big[\log P(e_c|e_t) + \sum_{c_i \in \mathbf{A}(e_t)} w_i \log P(e_c|c_i) \Big], \tag{3}$$

where $\mathbf{A}(e_t)$ represents the set of ancestor categories of entity e_t, and w_i is the weight of each category in predicting the context entity. To implement the intuition that a category is more relevant to its closer ancestor, for example, "NBC Mystery Movies" is more relevant to "Mystery Movies" than "Entertainment", we set $w_i \propto \frac{1}{l(c_c, c_i)}$ where $l(c_c, c_i)$ denotes the average number of steps going down from category c_i to category c_c, and it is constrained with $\sum_i w_i = 1$.

Figure 2 presents the results of our HCE model for DOTA-all data set (see Section 5.1.1). The visualization shows that our embedding method is able to clearly separate entities into distinct categories.

Figure 2: Category and entity embedding visualization of the DOTA-all data set (see Section 5.1.1). We use t-SNE (Van der Maaten and Hinton, 2008) algorithms to map vectors into a 2-dimensional space. Labels with the same color are entities belonging to the same category. Labels surrounded by a box are categories vectors.

3.3 Learning

Learning CE and HCE models follows the optimization scheme of skip-gram model (Mikolov et al., 2013). We use negative sampling to reformulate the objective function, which is then optimized through stochastic gradient descent (SGD).

Specifically, the likelihood of each context entity of a target entity is defined with the softmax function in Eq. 1, which iterates over all entities. Thus, it is computationally intractable. We apply the standard negative sampling technique to transform the objective function in equation (3) to equation (4) below and then optimize it through SGD:

$$L = \sum_{(e_c,e_t)\in\mathbf{D}} \Big[\log\sigma(e_c\cdot e_t) + \sum_{c_i\in\mathbf{A}(e_t)} w_i \log\sigma(e_c\cdot c_i)\Big] + \sum_{(e'_c,e_t)\in\mathbf{D}'} \Big[\log\sigma(-e'_c\cdot e_t) + \sum_{c_i\in\mathbf{A}(e_t)} w_i \log\sigma(-e'_c\cdot c_i)\Big], \quad (4)$$

where $\mathbf{D}'$ is the set of negative sample pairs and $\sigma(x) = 1/(1+\exp(-x))$ is the sigmoid function.

4 Applications

We apply our category and entity embedding to two applications: concept categorization and dataless hierarchical classification.

4.1 Concept Categorization

Concept[2] categorization, also known as *concept learning* or *noun categorization*, is a process of assigning a concept to one candidate category, given a set of concepts and candidate categories. Traditionally, concept categorization is achieved by **concept clustering** due to the lack of category representations. Since our model can generate representations of categories, we propose a new method of using **nearest neighbor (NN) classification** to directly categorize each concept to a certain category.

The **concept clustering** is defined as: given a set of concepts like *dog, cat, apple* and the corresponding gold standard categorizations like *animal, fruit*, apply a word space model to project all the concepts to

[2]In this paper, concept and entity denote the same thing.

a semantic space and perform clustering. The clustering results can be evaluated by comparing with the gold standard categorizations. Since we have representation of categories, we propose **nearest neighbor (NN) classification** to categorize concepts by directly comparing concept vectors with candidate category vectors. Precisely, given a set of concepts $\mathbf{E}$ and a set of candidate categories $\mathbf{C}$, we convert all concepts to concept vectors and all candidate categories to category vectors. Then we use the equation $c = \text{argmin}_{c_i \in \mathbf{C}} ||c_i - e||$ to assign the concept vector e with category c.

Since purity works as a standard evaluation metric for clustering (Rothenhäusler and Schütze, 2009), to compare our model with the concept clustering, we also use purity to measure our model's performance. Specifically, **purity** is defined as:

$$\text{purity}(\mathbf{\Omega}, \mathbf{G}) = \frac{1}{n} \sum_k \max_j |\omega_k \cap g_j|, \tag{5}$$

where $\mathbf{\Omega}$ denotes a clustering solution, $\mathbf{G}$ is a set of gold standard classes, n is the number of instances, k is the cluster index and j is the class index, ω_k represents the set of labels in a cluster and g_j is the set of labels in a class. A higher purity indicates better model performance.

4.2 Dataless Classification

Dataless classification uses the similarity between documents and labels in an enriched semantic space to determine in which category the given document is. It has been proved that explicit semantic analysis (ESA) (Gabrilovich and Markovitch, 2007) has shown superior performance on dataless classification (Song and Roth, 2014). ESA uses a bag-of-entities retrieved from Wikipedia to represent the text. For example, the document *"Jordan plays basketball"* can be represented as a ESA sparse vector of {*Michael Jordan:48.3, Basketball:29.8, Air Jordan: 28.8, Outline of basketball: 28.5, Chicago Bulls: 23.6*} in an ESA implementation. After converting documents and short label descriptions to ESA vectors, a nearest neighbor classifier is applied to assign labels for each document.

Due to sparsity problem of ESA vectors, sparse vector densification is introduced to augment the similarity calculation between two ESA vectors (Song and Roth, 2015). This is achieved by considering pairwise similarity between entities in ESA vectors. However, they simply use word2vec (Mikolov et al., 2013) to derive entity representation. We extend it by directly applying entity embeddings and show the potential of entity embeddings to improve dataless classification.

We use averaged F_1 scores to measure the performance of all methods (Yang, 1999). Let TP_i, FP_i, FN_i denote the true-positive, false-positive and false-negative values for the ith label in label set T, the micro- and macro-averaged F_1 scores are defined as: $MicroF_1 = 2\overline{P} * \overline{R}/(\overline{P} + \overline{R})$ and $MacroF_1 = \frac{1}{|T|} \sum_i^{|T|} \frac{2P_i * R_i}{P_i + R_i}$, where $P_i = TP_i/(TP_i + FP_i)$ and $R_i = TP_i/(TP_i + FN_i)$ are precision and recall for ith label, $\overline{P} = \sum_i^{|T|} TP_i / \sum_i^{|T|} (TP_i + FP_i)$ and $\overline{R} = \sum_i^{|T|} TP_i / \sum_i^{|T|} (TP_i + FN_i)$ are averaged precision and recall for all labels.

5 Experiments

In the experiments, we use the dataset collected from Wikipedia on Dec. 1, 2015[3] as the training data. We preprocess the category hierarchy by pruning administrative categories and deleting bottom-up edges to construct a DAG. The final version of data contains 5,373,165 entities and 793,856 categories organized as a DAG with a maximum depth of 18. The root category is "main topic classifications". We train category and entity vectors in dimensions of 50, 100, 200, 250, 300, 400, 500, with batch size $B = 500$ and negative sample size $k = 10$.

With the training dataset defined above, we conduct experiments on two applications: concept categorization and dataless hierarchical classification.

[3] https://dumps.wikimedia.org/wikidatawiki/20151201/

5.1 Concept Categorization

5.1.1 Datasets

There are two datasets used in this experiment. The first one is the **Battig** test set introduced by (Baroni and Lenci, 2010), which includes 83 concepts from 10 categories. The **Battig** test set only contains single-word concepts without any multiple-word concepts (e.g., "table tennis"). Hence, using this dataset restricts the power of concept categorization to single-word level. We use this dataset because it has been used as a benchmark for most previous approaches for concept categorization.

Due to the limitations of the **Battig** test set, we construct a new entity categorization dataset **DOTA** (Dataset Of enTity cAtegorization) with 450 entities categorized into 15 categories (refer to Appendix A). All the categories and entities are extracted from Wikipedia, so the resulting dataset does not necessarily contains only single-word entities. Thus, the dataset can be split into two parts, **DOTA-single** that contains 300 single-word entities categorized into 15 categories and **DOTA-mult** that contains 150 multiple-word entities categorized into the same 15 categories. We design the **DOTA** dataset based on the following principles:

Coverage vs Granularity: Firstly, the dataset should cover at least one category of Wikipedia's main topics including "Culture", "Geography", "Health", "Mathematics", "Nature", "People", "Philosophy", "Religion", "Society" and "Technology". Secondly, categories should be in different granularity, from large categories (e.g.,"philosophy") to small categories (e.g., "dogs"). Large categories are ones that are located within 5 layers away from the root, medium categories are 6-10 layers away from the root, while small categories have distance of 11-18 to the root. Our dataset consists of 1/3 large categories, 1/3 medium categories, and 1/3 small categories.

Single-Words vs Multiple-Words: Previous concept categorization datasets only contain single-words. However, some concepts are multiple-words and cannot be simply represented by single-words. For example, the concept "hot dog" is very different from the concept "dog". Therefore, we make each category of the dataset contain 10 multiple-word entities and 20 single-word entities.

5.1.2 Baselines

We compare our entity and category embeddings with the following state of the art word and entity embeddings.

WE$_{Mikolov}$(Mikolov et al., 2013): We trained word embeddings with Mikolov's word2vec toolkit[4] on the same Wikipedia corpus as ours (1.7 billion tokens) and then applied the Skip-gram model with negative sample size of 10 and window size of 5 in dimensionality of 50, 100, 200, 250, 300, 400, 500.

WE$_{Senna}$ (Collobert et al., 2011): We downloaded this 50-dimension word embedding[5] trained on Wikipedia over 2 months. We use this embedding as a baseline because it is also trained on Wikipedia.

Given the above semantic representation of words, we derive each entity/category embedding by averaging all word vectors among each entity/category label. If the label is a phrase that has been embedded (e.g., Mikolov's embedding contains some common phrases), we direct use the phrase embedding.

HEE (Hu et al., 2015): This Hierarchical Entity Embedding method uses the whole Wikipedia hierarchy to train entity embeddings and distance metrics. We used the tools provided by the authors to train entity embeddings on the same Wikipedia corpus as ours. We set the batch size B = 500, the initial learning rate η = 0.1 and decrease it by a factor of 5 whenever the objective value does not increase, and the negative sample size k = 5 as suggested by the authors. This method doesn't provide a way of learning category embeddings. Therefore, it cannot be used in the **nearest neighbor classification** method.

TransE (Bordes et al., 2013): It is a state of the art relational embedding method introduced in Section. 2, which embeds entities and relations at the same time. It can be extended to derive category embeddings, since categories can be seen as a special entity type. To make comparisons fair enough, we adopt three different versions of TransE to derive entity and category embedding to compare with our methods. **TransE**$_1$: Use entities and categories as two entity types and the direct-labeled-category relation. So the triplets we have is in the form of (entity, direct-labeled-category, category). **TransE**$_2$: Add

[4]https://code.google.com/archive/p/word2vec/

[5]http://ronan.collobert.com/senna/

to TransE$_1$ with the hierarchy structure of wikipedia categories. Namely, add the super-category relation between categories. Therefore we have new triplets in the form of (category, super-category, category).
TransE$_3$: To make it use full information as ours, we extend TransE$_2$ to utilize context entity relationship described in Section. 3.1. Therefore we add new triplets in the form of (entity, context-entity, entity).

HC: To further evaluate the advantage of utilizing category hierarchy in training entity and category embedding, we also compare our Hierarchical Category Embedding (HCE) model with our Category Embedding (CE) model that has no hierarchical information.

5.1.3 Results

In the experiments, we used scikit-learn (Pedregosa et al., 2011) to perform clustering. We tested k-means and hierarchical clustering with different distance metrics (euclidean, cosine) and linkage criterion (ward, complete, average). All these choices along with the vector dimensionality are treated as our models' hyper-parameters. For selecting hyper-parameters, we randomly split the Battig and Dota datasets to 50% of validation data and 50% of test data, evenly across all categories. We trained all the embeddings (except SENNA) on the same Wikipedia dump and tuned hyper-parameters on the validation set. For experiments on Dota dataset, since the ground truth is contained in our Wikipedia corpus, we deleted all category-entity links contained in Dota dataset from our category hierarchy to train HEE, TransE and HCE embeddings to make comprison fair enough.

Table. 1a shows the experimental results of the **concept clustering** method. It is clear that **hierarchical category embedding** (HCE) model outperforms other methods in all datasets. Our model achieves a purity of 89% on Battig and 89% on DOTA-all.

	Battig	DOTA-single	DOTA-mult	DOTA-all
WE$_{Senna}$ (50,50)	0.74	0.61	0.43	0.45
WE$_{Mikolov}$ (400,400)	0.86	0.83	0.73	0.78
HEE (400,400)	0.82	0.83	0.80	0.81
TransE$_1$ (300,250)	0.67	0.71	0.68	0.69
TransE$_2$ (400,300)	0.73	0.78	0.75	0.76
TransE$_3$ (200,400)	0.43	0.53	0.50	0.51
CE (300,200)	0.84	0.86	0.83	0.85
HCE (400,400)	**0.89**	**0.92**	**0.88**	**0.89**

(a) Purity of **concept clustering** method

	Battig	DOTA-single	DOTA-mult	DOTA-all
WE$_{Senna}$ (50,50)	0.44	0.52	0.32	0.45
WE$_{Mikolov}$ (400,400)	0.74	0.74	0.67	0.72
HEE	-	-	-	-
TransE$_1$ (300,250)	0.66	0.72	0.69	0.71
TransE$_2$ (400,300)	0.75	0.80	0.77	0.79
TransE$_3$ (300,400)	0.46	0.55	0.52	0.54
CE (200,200)	0.79	0.89	0.85	0.88
HCE (400,400)	**0.87**	**0.93**	**0.91**	**0.92**

(b) Purity of **NN classification** method

Table 1: Purity of **nearest neighbor (NN) classification** and **concept clustering** methods with different embeddings on Battig and DOTA datasets. The two numbers given in each parentheses are the vector dimensionality among {50,100,200,250,300,400,500} that produce the best results on Battig and Dota validation set respectively. All the purity scores are calculated under such choices of hyper-parameters on the test set.

For word embeddings like Mikolov and Senna, the performance drops a lot from single-word entity categorization to multiple-word entity categorization, because these embeddings mainly contain single words. To get the embeddings of multiple-word, we use the mean word vectors to denote multiple-word embeddings. However, the meaning of a multiple-word is not simply the aggregation of the meaning of the words it contains. The good performance of HEE shows the high quality of its entity embeddings. By incorporating category hierarchy information, TransE$_2$ gets an advantage over TransE$_1$. This advantage can also be observed when we incorporate category hierarchy information into CE. However, further incorporating of context entities deteriorates the performance of TransE. This may due to the nature of transition-based models that assume relationships as transitions among entities. They get much noise when introduced with many context entity triplets along with only one entity-to-context-entity relation type.

Table.1b shows the experimental results of the **nearest neighbor (NN) classification** method. The results indicate the feasibility of using category vectors to directly predict the concept categories without clustering entities. Our model achieves a purity of 87% on Battig and 92% on DOTA-all.

5.2 Dataless Classification

5.2.1 Datasets

20Newsgroups Data(20NG): The 20 newsgroups dataset (Lang, 1995) contains about 20,000 newsgroups documents evenly categorized to 20 newsgroups, and further categorized to six super-classes. We use the same label description provided by (Song and Roth, 2014).

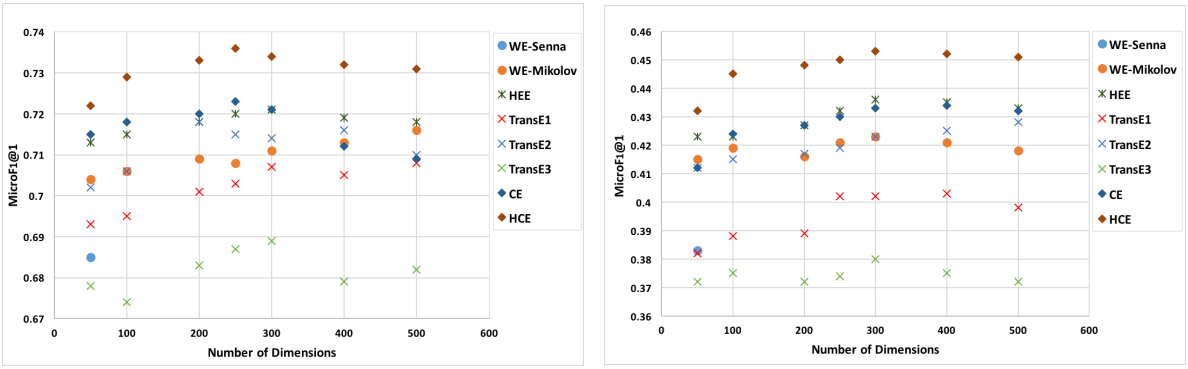
RCV1 Dataset: The RCV1 dataset (Lewis et al., 2004) contains 804,414 manually labeled newswire documents, and categorized with respect to three controlled vocabularies: industries, topics and regions. We use topics as our hierarchical classification problem. There are 103 categories including all nodes except for the root in the hierarchy, and the maximum depth is 4. To ease the computational cost of comparison, we follow the chronological split proposed in (Lewis et al., 2004) to use the first 23,149 documents marked as training samples in the dataset. The dataset also provides the name and the description of each category label.

5.2.2 Implementation Details

The baseline of similarity measure between each label and each document is cosine similarity between corresponding ESA vectors. We use ESA with 500 entities, and augment similarity measure by plugging in different embeddings introduced in Section. 5.1.2 with Hungarian sparse vector densification method described in (Song and Roth, 2015). The Hungarian method is a combinatorial optimization algorithm aiming to find an optimal assignment matching the two sides of a bipartite graph on a one-to-one basis. We average over all cosine similarities of entity pairs produced by the Hungarian method to produce similarity between ESA vectors. In addition, we cut off all entity similarities below 0.85 and map them to zero empirically.

For classification, We use the bottom-up pure dataless hierarchical classification algorithm which proved to be superior in (Song and Roth, 2014) and set the threshold δ to be 0.95 empirically.

5.2.3 Results



(a) 20newsgroups: bottom-up (ESA:0.682)

(b) RCV1: bottom-up (ESA:0.371)

Figure 3: MicroF$_1$@1 results of ESA augmented with different embeddings for dataless hierarchical classification. It is clear the ESA densified with HCE performs best on dataless hierarchical classification.

We perform dataless hierarchical classification using the same ESA settings, densified with different embedding methods for similarity calculation. Figure. 3 presents the classification performance with different embedding methods in dimensionality of {50,100,200,250,300,400,500}. It is clear that ESA densified with HCE stably outperform other competitive methods on these two datasets, which indicates the high quality of entity embeddings derived by HCE.

Besides, we observe that entity embedding methods such as TransE$_2$, HEE and HCE perform better than word embedding methods. This is because ESA represents text as bag-of-entities, and entity embeddings successfully encode entity similarities. Compared with bag-of-words baselines in (Song and Roth, 2014), it is clear that bag-of-entities models can better capture semantic relatedness between documents and short category descriptions.

6 Conclusion

In this paper, we proposed a framework to learn entity and category embeddings to capture semantic relatedness between entities and categories. This framework can incorporate taxonomy hierarchy from large scale knowledge bases. Experiments on both concept categorization and dataless hierarchical classification indicate the potential usage of category and entity embeddings on more other NLP applications.

References

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

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, number EPFL-CONF-192344.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611.

Siddharth Gopal and Yiming Yang. 2013. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 257–265. ACM.

Zhiting Hu, Poyao Huang, Yuntian Deng, Yingkai Gao, and Eric P Xing. 2015. Entity hierarchy embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, volume 1, pages 1292–1300.

Zhiting Hu, Gang Luo, Mrinmaya Sachan, Eric Xing, and Zaiqing Nie. 2016. Grounding topic models with knowledge bases. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*.

Sung Ju Hwang and Leonid Sigal. 2014. A unified semantic embedding: Relating taxonomies and attributes. In *Advances in Neural Information Processing Systems*, pages 271–279.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th international conference on machine learning*, pages 331–339.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. *arXiv preprint arXiv:1504.06658*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, pages 271–280. ACM.

Alberto Paccanaro and Geoffrey E Hinton. 2001. Learning distributed representations of concepts using linear relational embedding. *Knowledge and Data Engineering, IEEE Transactions on*, 13(2):232–244.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830.

Klaus Rothenhäusler and Hinrich Schütze. 2009. Unsupervised classification with dependency based word spaces. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 17–24. Association for Computational Linguistics.

Yangqiu Song and Dan Roth. 2014. On dataless hierarchical text classification. In *AAAI*, pages 1579–1585.

Yangqiu Song and Dan Roth. 2015. Unsupervised sparse vector densification for short text similarity. *Proc. North Am. Chapter Assoc. Computat. Linguistics*, pages 1275–1280.

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

Nakul Verma, Dhruv Mahajan, Sundararajan Sellamanickam, and Vinod Nair. 2012. Learning hierarchical similarity metrics. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2280–2287. IEEE.

Kilian Q Weinberger and Olivier Chapelle. 2009. Large margin taxonomy embedding for document categorization. In *Advances in Neural Information Processing Systems*, pages 1737–1744.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. *arXiv preprint arXiv:1601.01343*.

Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. *Information retrieval*, 1(1-2):69–90.

Wenpeng Yin and Hinrich Schütze. 2014. An exploration of embeddings for generalized phrases. In *ACL (Student Research Workshop)*, pages 41–47.

Mo Yu and Mark Dredze. 2015. Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242.

Hao Zhang, Zhiting Hu, Yuntian Deng, Mrinmaya Sachan, Zhicheng Yan, and Eric P Xing. 2016. Learning concept taxonomies from multi-modal data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

A The DOTA dataset: 300 single-word entities and 150 multi-word entities from 15 Wikipedia Categories

Category	Entities
beverages	juice, beer, milk, coffee, tea, cocktail, wine, liqueur, sake, vodka, mead, sherry, brandy, gin, rum, latte, whisky, cider, gose, rompope, orange juice, masala chai, green tea, black tea, herbal tea, coconut milk, corn syrup, soy milk, rose water, hyeonmi cha
sports	bowling, football, aerobics, hockey, karate, korfball, handball, floorball, skiing, cycling, racing, softball, shooting, netball, snooker, powerlifting, jumping, wallball, volleyball, snowboarding, table tennis, floor hockey, olympic sports, wheelchair basketball, crab soccer, indoor soccer, table football, roller skating, vert skating, penny football
emotions	love, anxiety, empathy, fear, envy, loneliness, shame, anger, annoyance, happiness, jealousy, apathy, resentment, frustration, belongingness, sympathy, pain, worry, hostility, sadness, broken heart, panic disorder, sexual desire, falling in love, emotional conflict, learned helplessness, chronic stress, anxiety sensitivity, mental breakdown, bike rage
weather	cloud, wind, thunderstorm, fog, snow, wave, blizzard, sunlight, tide, virga, lightning, cyclone, whirlwind, sunset, dust, frost, flood, thunder, supercooling, fahrenheit, acid rain, rain and snow mixed, cumulus cloud, winter storm, blowing snow, geomagnetic storm, blood rain, fire whirl, pulse storm, dirty thunderstorm
landforms	lake, waterfall, stream, river, wetland, marsh, valley, pond, sandstone, mountain, cave, swamp, ridge, plateau, cliff, grassland, glacier, hill, bay, island, glacial lake, drainage basin, river delta, stream bed, vernal pool, salt marsh, proglacial lake, mud volcano, pit crater, lava lake
trees	wood, oak, pine, evergreen, willow, vine, shrub, birch, beech, maple, pear, fir, pinales, lauraceae, sorbus, buxus, acacia, rhamnaceae, fagales, sycamore, alhambra creek, alstonia boonei, atlantic hazelwood, bee tree, blood banana, datun sahib, druid oak, new year tree, heart pine, fan palm
algebra	addition, multiplication, exponentiation, tetration, polynomial, calculus, permutation, subgroup, integer, monomial, bijection, homomorphism, determinant, sequence, permanent, homotopy, subset, factorization, associativity, commutativity, real number, abstract algebra, convex set, prime number, complex analysis, natural number, complex number, lie algebra, identity matrix, set theory
geometry	trigonometry, circle, square, polyhedron, surface, sphere, cube, icosahedron, hemipolyhedron, digon, midpoint, centroid, octadecagon, curvature, curve, zonohedron, cevian, orthant, cuboctahedron, midsphere, regular polygon, uniform star polyhedron, isogonal figure, icosahedral symmetry, hexagonal bipyramid, snub polyhedron, homothetic center, geometric shape, bragg plane, affine plane
fish	goldfish, gourami, koi, cobitidae, tetra, goby, danio, wrasse, acanthuridae, anchovy, carp, catfish, cod, eel, flatfish, perch, pollock, salmon, triggerfish, herring, cave catfish, coachwhip ray, dwarf cichlid, moray eel, coastal fish, scissortail rasbora, flagtail pipefish, armoured catfish, hawaiian flagtail, pelagic fish
dogs	spaniel, foxhound, bloodhound, beagle, pekingese, weimaraner, collie, terrier, poodle, puppy, otterhound, labradoodle, puggle, eurasier, drever, brindle, schnoodle, bandog, leonberger, cockapoo, golden retriever, tibetan terrier, bull terrier, welsh springer spaniel, hunting dog, bearded collie, picardy spaniel, afghan hound, brittany dog, redbone coonhound
music	jazz, blues, song, choir, opera, rhythm, lyrics, melody, harmony, concert, comedy, violin, drum, piano, drama, cello, composer, musician, drummer, pianist, hip hop, classical music, electronic music, folk music, dance music, musical instrument, disc jockey, popular music, sheet music, vocal music
politics	democracy, law, government, liberalism, justice, policy, rights, utilitarianism, election, capitalism, ideology, egalitarianism, debate, regime, globalism, authoritarianism, monarchism, anarchism, communism, individualism, freedom of speech, political science, public policy, civil society, international law, social contract, election law, social justice, global justice, group conflict
philosophy	ethics, logic, ontology, aristotle, plato, rationalism, platonism, relativism, existence, truth, positivism, metalogic, subjectivism, idealism, materialism, aesthetics, probabilism, monism, truth, existence, western philosophy, contemporary philosophy, cognitive science, logical truth, ancient philosophy, universal mind, visual space, impossible world, theoretical philosophy, internal measurement
linguistics	syntax, grammar, semantics, lexicon, speech, phonetics, vocabulary, phoneme, lexicography, language, pragmatics, orthography, terminology, pronoun, noun, verb, pronunciation, lexicology, metalinguistics, paleolinguistics, language death, historical linguistics, dependency grammar, noun phrase, comparative linguistics, word formation, cognitive semantics, syntactic structures, auxiliary verb, computational semantics
vehicles	truck, car, aircraft, minibus, motorcycle, microvan, bicycle, tractor, microcar, van, ship, helicopter, airplane, towing, velomobile, rocket, train, bus, gyrocar, cruiser, container ship, school bus, road train, tow truck, audi a6, garbage truck, hydrogen tank, light truck, compressed air car, police car

Latent Topic Embedding

Di Jiang
Baidu, Inc., China
jiangdi@baidu.com

Lei Shi
Baidu, Inc., China
shilei06@baidu.com

Rongzhong Lian
Baidu, Inc., China
lianrongzhong@baidu.com

Hua Wu
Baidu, Inc., China
wu_hua@baidu.com

Abstract

Topic modeling and word embedding are two important techniques for deriving latent semantics from data. General-purpose topic models typically work in coarse granularity by capturing word co-occurrence at the document/sentence level. In contrast, word embedding models usually work in fine granularity by modeling word co-occurrence within small sliding windows. With the aim of deriving latent semantics by capturing word co-occurrence information at different levels of granularity, we propose a novel model named *Latent Topic Embedding* (LTE), which seamlessly integrates topic generation and embedding learning in one unified framework. We further propose an efficient Monte Carlo EM algorithm to estimate the parameters of interest. By retaining the individual advantages of topic modeling and word embedding, LTE results in better latent topics and word embedding. Experimental results verify the superiority of LTE over the state-of-the-arts in real-life applications.

1 Introduction

Topic modeling and word embedding are gaining significant momentum in the field of text mining. General-purpose topic models such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and Sentence LDA (Jo and Oh, 2011) usually utilize word co-occurrences at the document/sentence level to compose the "topics", which capture the latent semantics between words. These models are plagued by the simplistic bag-of-words assumption, which ignores the valuable sub-sequence information between words. Some recent endeavors introduced n-gram information into topic models (Wallach, 2006), however, the size of vocabulary is significantly enlarged and these techniques are hardly feasible for real-life applications. Therefore, the technique of topic modeling needs a remedy for solving the word sequence problem with fairly low cost. Word embedding models such as Word2Vec (Mikolov et al., 2013a) map words into distributed representations. Word embedding models primarily focus on the word co-occurrences within small sliding windows, which enable word embedding to capture (at least partially) the information of word sequences. One key problem of the existing word embedding models is that they are typically short-sighted and are not aware of the themes of the document.

With their differences, the core of topic modeling and word embedding is based upon the assumption that the words co-occurring frequently should have semantic commonality. In light of their individual advantages and drawbacks, we see that the two techniques are essentially complimentary and can be integrated to enhance each other. In this paper, we propose the *Latent Topic Embedding* (LTE) to seamlessly integrate topic modeling and word embedding in one framework. In the generative process of LTE, we assume that the observed words in document can be generated through two channels: one is through the Multinomial distribution and the other is based upon topic embeddings as well as word embeddings. In this way, the embedding information influences the result of topic modeling while the topic information affects the training of word embeddings in return. LTE enables topic modeling to utilize word sequence information and it equips word embeddings with the document-level vision. We propose a Monte Carlo

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2689–2698, Osaka, Japan, December 11-17 2016.

EM algorithm to efficiently infer the parameters of interest in LTE. Extensive experiments on real-life applications verify the superiority of LTE over several strong baselines.

The rest of this paper is organized as follows. In Section 2, we review the related work. In Section 3, we discuss the technical details of LTE. In Section 4, we illustrate how to conduct parameter inference for LTE. In Section 5, we present the experimental results. Finally, we conclude this paper in Section 6.

2 Related Work

The present work is related to previous research on topic modeling and word embedding. Latent Dirichlet allocation (LDA) is a generative probabilistic model for collections of discrete data such as text corpora (Blei et al., 2003)(Griffiths and Steyvers, 2004). LDA and its variants has been widely employed in many texting mining scenarios (Wang and McCallum, 2006)(Krestel et al., 2009)(Xu et al., 2009)(Jiang et al., 2013) and demonstrated promising performance. It is worth mentioning that some work such as the bigram topic model (Wallach, 2006) aims to alleviate the negative effect of bag-of-words assumption in LDA. However, considerable computational cost is involved since the bigram model creates a multinomial distribution for each pair of the topics and the words, the amount of which is usually voluminous. While topic modeling received intensive research in the field of Bayesian network research, word embedding received much attention in the field of neural network. Word embedding (Bengio et al., 2003) is proposed to fight the curse of dimensionality by learning a distributed representation for words which allows each training sentence to inform the model about an exponential number of semantically neighboring sentences. Mikolov presented several extensions of Skip-gram that improve both the quality of the vectors and the training speed (Mikolov et al., 2013b) (Mikolov et al., 2013a). Paragraph vector that is an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts was proposed in (Le and Mikolov, 2014). Word embedding is adapted for incorporating contextual information in learning vector-space representations of situated language (Bamman et al., 2014). A more relevant work is (Liu et al., 2015), which inputs the result of topic modeling into word embedding models to learn the topical word embedding. The major difference between this work and ours is that they did not aim to integrate topic modeling and word embedding and yet only utilizes the result of topic modeling as the input of word embedding models. Recently, (Nguyen et al., 2015) extended two Dirichlet multinomial topic models by incorporating word embeddings to improve the word-topic mapping. (Li et al., 2016) proposed a generative model that replaces the Multinomial word generation assumption of LDA with embedding based assumption.

Although topic modeling and word embedding receive intensive attention in recent years, to the best of our knowledge, there is no previous endeavor on integrating them together as a joint learning task to enhance each other. LTE paves the way for collectively modeling of word co-occurrence information at different granularity levels while retaining the topic modeling result as well as the word embedding result.

3 Generative Process of Latent Topic Embedding

Latent Topic Embedding (LTE) views each document as a bag of sentences and each sentence is composed of words. The generative process of LTE is formally depicted in Algorithm 1. For each topic k, the corresponding multinomial topic-word distribution ϕ_k is drawn from $Dirichlet(\beta)$. When generating a document, a multinomial document-topic distribution θ_d is drawn from $Dirichlet(\alpha)$. For each sentence s in the document, we draw a latent topic z_{ds} based on the document-topic distribution. For each token in the document, we drawn an indicator i from Bernoulli(τ). If i is 0, the word w is generated according to topic-word distribution $\phi_{z_{ds}}$. If i is 1, the word w is generated according to topic-word distribution $P(w|z_{ds}, C_w, M)$, which is defined as follows:

$$P(w|z_{ds}, C_w, M) = P(v_w|\mathbf{x}_w) = \frac{e^{\mathbf{x}_w \cdot v_w}}{\sum_{w'} e^{\mathbf{x}_w \cdot v_{w'}}}. \quad (1)$$

In Eq. (1), C_w stands for the sliding window for w. Specifically, C_w contains several words that precedes w. where $M = \{\mathbf{v_w}, \mathbf{v_z}\}$ stands for the word embedding and the topic embedding, $\mathbf{x}_w$ is the result

Algorithm 1: Generative Process

for *each topic* $k \in (1, 2, ..., K)$ **do**
  draw a word distribution $\phi_k \sim$ Dirichlet (β);
end
for *each document* d **do**
  draw a topic distribution $\theta_d \sim$ Dirichlet (α);
  for *each sentence* s *in* d **do**
    draw a topic $z_{ds} \sim$ Multinomial (θ_d)
    for *each token in* s **do**
      draw an indicator $i \sim$ Bernoulli(τ)
      if $i = 0$ **then**
        generate word $w \sim$ Multinomial $(\phi_{z_{ds}})$
      end
      else
        generate word $w \sim P(w|z_{ds}, C_w, M)$
      end
    end
  end
end

of element-wise addition of the word embeddings of C_w and the topic embedding indexed by z_{ds} (i.e., $\mathbf{x}_w = \mathbf{v}_{c_w} \oplus v_{z_{ds}}$) and v_w is the embedding of w. The parameters of interest are ϕ, θ and M.

4 Training LTE

In Section 4.1, we describe how to sample the latent topics for sentences. In Section 4.2, we discuss how to optimize the vectors via stochastic gradient descent. The parameter inference algorithm is formally presented in Section 4.3.

4.1 Sampling Latent Variables

By translating the generative process of LTE into joint distribution, we aim to maximize the likelihood of the observed words $\mathbf{w}$: $P(\mathbf{w}|\alpha, \beta, \tau, M)$. Ideally, we would compute optimal M by maximizing $P(\mathbf{w}|\alpha, \beta, \tau, M)$ directly. However, evaluating this likelihood is intractable and what can be computed is the complete likelihood $p(\mathbf{w}, \mathbf{i}, \mathbf{z}|\alpha, \beta, \tau, M)$:

$$P(\mathbf{w}, \mathbf{i}, \mathbf{z}|\alpha, \beta, \tau, M) = P(\mathbf{z}|\alpha)P(\mathbf{i}|\tau)P(\mathbf{w}|\mathbf{z}, \mathbf{i}, \beta, M)$$
$$= \Big(\frac{\Gamma(\sum_{z=1}^{T}\alpha_z)}{\prod_{z=1}^{T}\Gamma(\alpha_z)}\Big)^D \prod_{d=1}^{D} \frac{\prod_{z=1}^{T}\Gamma(m_{dz}+\alpha_z)}{\Gamma(\sum_{z=1}^{T}(m_{dz}+\alpha_z))} \Big(\frac{\Gamma(\sum_{v=1}^{V}\beta_v)}{\prod_{v=1}^{V}\Gamma(\beta_v)}\Big)^T \prod_{z=1}^{T} \frac{\prod_{v=1}^{V}\Gamma(n_{zv}+\beta_v)}{\Gamma(\sum_{v=1}^{V}(n_{zv}+\beta_v))} \quad (2)$$
$$\prod_{d=1}^{D}\prod_{s\in d}\prod_{w\in s} P(v_w|\mathbf{x}_w)^{\mathcal{I}(i_w=1)} \times (1-\tau)^A \tau^B,$$

where m_{dz} is the number of sentences that are assigned to topic z in document d. n_{zv} is the number of times that v is assigned to topic z through Multinominal distribution and $\Gamma(\cdot)$ indicates Gamma function, A is the number of 0 that are generated by the Bernoulli distribution and B is the number of 1 that are generated by the Bernoulli distribution. By applying Bayes rule, the full conditional of assigning topic k to z_{ds} is obtained as follows:

$$P(z_{ds} = k, \mathbf{i}_{ds}|\mathbf{w}, \mathbf{z}_{-ds}, \mathbf{i}_{-ds}, \alpha, \beta, \tau, M) = (1-\tau)^{A_s}\tau^{B_s}$$
$$\frac{m_{dk}+\alpha_k}{\sum_{k'=1}^{K}(m_{dk'}+\alpha_{k'})} \frac{\Gamma(\sum_{w=1}^{W}(n_{kw}+\beta_w))}{\Gamma(\sum_{w=1}^{W}(n_{kw}+\beta_w+N_{iw}))} \prod_{w\in\mathbf{W}\&i_w=0} \frac{\Gamma(n_{kw}+\beta_w+N_{iw})}{\Gamma(n_{kw}+\beta_w)} \prod_{w\in s\&i_w=1} P(v_w|\mathbf{x}_w) \quad (3)$$

The possible combinations of $\mathbf{i}_{ds}$ is exponential in the length of the sentence s. Similar to (Nguyen et al., 2015), we conduct approximation of the above equation and integrate out $\mathbf{i}_{ds}$,

$$P(z_{ds}=k|\mathbf{w},\mathbf{z}_{-ds},\mathbf{i}_{-ds},\alpha,\beta,\tau,M)\approx\frac{m_{dk}+\alpha_k}{\sum_{k'=1}^{K}(m_{dk'}+\alpha_{k'})}\prod_{w\in s}\big((1-\tau)\frac{n_{kw}+\beta_w}{\sum_{w=1}^{W}(n_{kw}+\beta_w)}+\tau P(v_w|\mathbf{x}_w)\big) \tag{4}$$

Exactly calculating $P(v_w|\mathbf{x}_w)$ is computational infeasible, since the normalization term involves all the words in the vocabulary. Thus, we utilize noise contrastive estimation (NCE) to approximate it. The advantage of NCE is that it allows us to fit models that are not explicitly normalized making the training time effectively independent of the vocabulary size. Thus, we will be able to drop the normalization factor from the above equation, and simply use $e^{\mathbf{x}_w\cdot v_w}$ in place of $P(\mathbf{x}_w|v_w)$. Similar to the method described in (Mnih and Teh, 2012)(Dyer, 2014), we fixing the normalized constants in $P(\mathbf{x}_w|v_w)$ to 1, then we obtain the following approximation:

$$P(z_{ds}=k|\mathbf{w},\mathbf{z}_{-ds},\mathbf{i}_{-ds},\alpha,\beta,\tau,M)\propto(m_{dk}+\alpha_k)\prod_{w\in s}\big((1-\tau)\frac{n_{kw}+\beta_w}{\sum_{w=1}^{W}(n_{kw}+\beta_w)}+\tau e^{\mathbf{x}_w\cdot v_w}\big) \tag{5}$$

For each word w in sentence s, its latent indicator i_w is sampled as follows:

$$P(i_w=0|z_{ds}=k)\propto(1-\tau)\frac{n_{kw}+\beta_w}{\sum_{w=1}^{W}(n_{kw}+\beta_w)} \tag{6}$$

$$P(i_w=1|z_{ds}=k)\propto\tau e^{\mathbf{x}_w\cdot v_w} \tag{7}$$

The above sampling process repeats for a predefined number of iterations. It is worth mentioning that there are works about scaling up Gibbs sampling or make it more efficient. Since the topic of designing better Gibbs sampling algorithms is beyond the scope of this paper, interested readers may refer to (Newman et al., 2009) and (Wang et al., 2009) for more detailed information.

4.2 Embedding Optimization

Now we convert the joint likelihood in Eq. (2) to its logarithm form, which is defined as follows:

$$\begin{aligned}\mathcal{L}(\mathbf{w},\mathbf{i},\mathbf{z};\alpha,\beta,\tau,M)=D\log\Big(\frac{\Gamma(\sum_{z=1}^{T}\alpha_z)}{\prod_{z=1}^{T}\Gamma(\alpha_z)}\Big)+\sum_d\sum_z\log(\Gamma(m_{dz}+\alpha_z))-\\ \sum_d\log(\Gamma(\sum_z(m_{dz}+\alpha_z)))T\log\Big(\frac{\Gamma(\sum_{v=1}^{V}\beta_v)}{\prod_{v=1}^{V}\Gamma(\beta_v)}\Big)+\sum_z\sum_v\log(\Gamma(n_{zv}+\beta_v))-\\ \sum_z\log(\Gamma(\sum_v(n_{zv}+\beta_v)))+\sum_d\sum_{s\in d}\sum_{w\in s\&i_w=1}\log P(v_w|\mathbf{x}_w)+A\log(1-\tau)+B\log\tau.\end{aligned} \tag{8}$$

Eq. (8) is a separable function. Each hyperparameter can be independently maximized. The hyperparameters α, β and τ can be straightforwardly optimized by Newton-Raphson algorithm like (Blei et al., 2003). As we usually utilize fixed α and β, the focus now is to illustrate how to optimize the vectors in M through maximizing $\sum_d\sum_{s\in d}\sum_{w\in s\&i_w=1}\log P(v_w|\mathbf{x}_w)$whose corresponding NCE log-likelihood is as follows:

$$\begin{aligned}\sum_d\sum_{s\in d}\sum_{u\in w\cup NEG(w)}\Big\{&l_u^{c_w}\cdot\log[\sigma(\mathbf{x}_w\cdot v_u-\log(\frac{|NEG|}{|V|}))]+\\ &[1-l_u^{c_w}]\cdot\log[1-\sigma(\mathbf{x}_w\cdot v_u-\log(\frac{|NEG|}{|V|}))]\Big\},\end{aligned} \tag{9}$$

where $\sigma(\cdot)$ to denote the sigmoid function, |NEG| is the number of negative samples for each word and $|V|$ is the size of vocabulary. We use stochastic gradient descent to optimize the embedding, the update formula for v_u in C_w is as follows:

$$v_u:=v_u+\eta\sum_{u'\in w\cup NEG(w)}\Big[l_{u'}^{c_w}-\sigma(\mathbf{x}_w\cdot v_{u'}-\log(\frac{|NEG|}{|V|}))\Big]\cdot v_{u'}, \tag{10}$$

where $NEG(w)$ stands for the negative samples of w. The update formula for the topic embedding v_z is as follows:

$$v_z := v_z + \eta \sum_{u' \in w \cup NEG(w)} \left[l_{u'}^{c_w} - \sigma(\mathbf{x}_w \cdot v_{u'} - \log(\frac{|NEG|}{|V|})) \right] \cdot v_{u'}. \tag{11}$$

4.3 Monte Carlo EM

Based on the above discussion, we now formally present the parameter inference of LTE in Algorithm 2. After applying this algorithm, we obtain the quantities of interest such as Θ, Φ the topic embeddings and the word embeddings. Note that LTE covers both the outputs of topic model and the output of word embedding. Theoretically, it can be applied in any scenario where topic modeling or word embedding is previously utilized. In the experiments, we will show that retaining both the outputs of topic model and word embedding is critical for comprehensively capturing different kinds of latent semantics in text.

Algorithm 2: Monte Carlo EM

repeat
  run Gibbs sampling according to Eq. (5)(6)(7) ;
  optimize the corresponding parameters according to Eq. (10) and (11);
until *a predefined number of iterations*;

5 Experiments

In this section, we evaluate the performance of LTE. Unless otherwise stated, the experimental results are obtained when the size of the embedding is set to 20 and the size of the sliding window is 5. Similar insights are obtained when varying the two parameters and we skip them due to space limitation. In Section 5.1, we present some topic examples. In Section 5.2, we show the result of perplexity evaluation. In Section 5.3, we evaluate the the performance LTE through a task of topical word extraction.

Table 1: LTE Topic Examples (The number in brackets is the frequency of the word)

	Multinomial Perspective	Embedding Perspective
Topic1	Taiwan(2332), China(30904), issue(19080), unity(2165), relationship(6052), principle(2256), Taiwan independence(20), people(4172), mainland(1125), peace(699)	party(2), legislator(21), two states theory(1), tamper(42), attentively(1), Frank Hsieh(2), beautify(141), Taiwan independence(20), Tsai Ing-wen(12)
Topic2	space(4507), satellite(244), technology(9673), system(7348), country(10619), international(5937), research(6571), data(3845), utilize(4035), earth(1170)	battery(484), spacecraft(10), sun(1686), antenna(156), circuit(259), airship(121), optics(97), transducer(221), physics(120), satellite(244)
Topic3	children(2950), woman(1053), violence(490), committee(936), behavior(4218), family(3918), society(10239), government(6141), measure (1734), right(1565)	drugster(12), antenatal(35), cancer(676), diarrhea(310), teenager(617), girl(2160), sexual abuse(4), patriarch(2431), nonage(63), Zhu Lin(5)
Topic4	central government(1838), conference(2004), work(18347) people(4172), the Communist Party of China(436), National People's Congress(408), member(3986), today(8322), State Department(970), committee member(493)	Hebei province(993), vice-governor(40), Shenyang city(657), Public security bureau(384), deputy mayor(118), deputy secretary(106), deputy director general(191), Hupei(585), accept bribes(125)

5.1 LTE Topics

An informal but important measure of the success of the proposed model is the plausibility of the discovered search topics (Doyle and Elkan, 2009). Hence, we can qualitatively evaluate LTE through viewing its latent topics. An import feature of LTE is that it discovers latent topics from two perspectives. The first perspective is based on the Θ parameter, which corresponds to the Multinomial distributions over the vocabulary. The second perspective is based on the topic embedding and word embedding, i.e., the

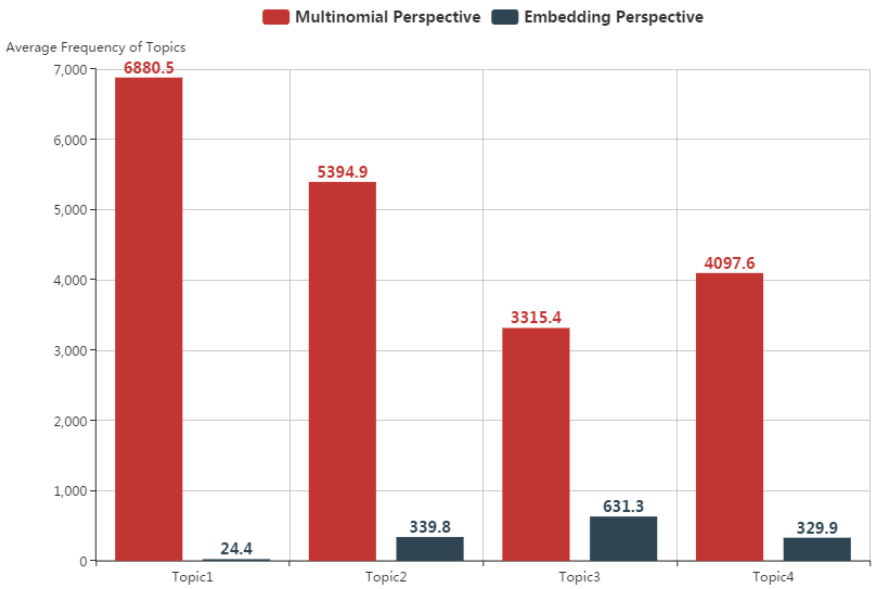


Figure 1: Average Word Frequency of the Two Perspectives

words whose embeddings have the highest cosine similarity with the topic embedding can be considered as the content of this topic. We utilize Web page dataset for the experiment. Some topic examples are presented in Table 1[1]. We observe that the words are semantically coherent in both of the two perspectives. For example, Topic 1 is about political issues between mainland China and Taiwan, Topic 2 is related to space technology, Topic 3 discusses the well-being of women and children and Topic 4 contains words about the political system of China. For each topic, the words from the two perspectives are semantically relevant and complimentary to each other.

An important insight is obtained from analyzing the frequencies of words in topics. The average word frequencies of the two perspectives are presented in Figure 1. We can see that word frequency of the second perspective is significantly smaller than that of the first perspective. For example, in Topic 1, the average word frequency of the first perspective is 6880.5 while that of the second perspective is only 24.4. This phenomenon shed light on an big advantage of LTE in text mining: bridging the semantic relevance between words with different frequencies. LTE overcomes the inherent problem of topic models that the topics are usually dominated by words of high frequency. By using the topic and word embeddings, we can effectively discover the semantics of words of relatively low frequency.

5.2 Perplexity Evaluation

We proceed to quantitatively compare LTE with LDA and the state-of-the-arts (i.e., Topical Word Embedding (TWE-1) (Liu et al., 2015) and Latent Feature-Dirichlet Multinomial Mixture (LFDMM) (Nguyen et al., 2015)) in terms of perplexity, which is a standard measure of evaluating the generalization performance of a probabilistic model (Rosen-Zvi et al., 2004). A lower perplexity indicates better generalization performance. A holdout dataset containing about ten thousand Web pages are utilized for perplexity evaluation. The result of perplexity comparison is presented in Figure 2. Since TWE-1 reuses the result of LDA, they have exactly the same performance in terms of perplexity. When varying the number of topics from 10 to 100, LTE always achieves the lowest perplexity, showing that generative process of LTE is a reasonable assumption for the data. An important observation is that LTE significantly outperforms LFDMM, showing that adding the sentence assumption and jointly utilizing word embedding and topic embedding to generate words result in better fit for the latent data structure of natural language documents. Perplexity is an indicator of the quality of the Multinomial topics. We observe that jointly

[1]The original Chinese words are translated into English to enhance readability.

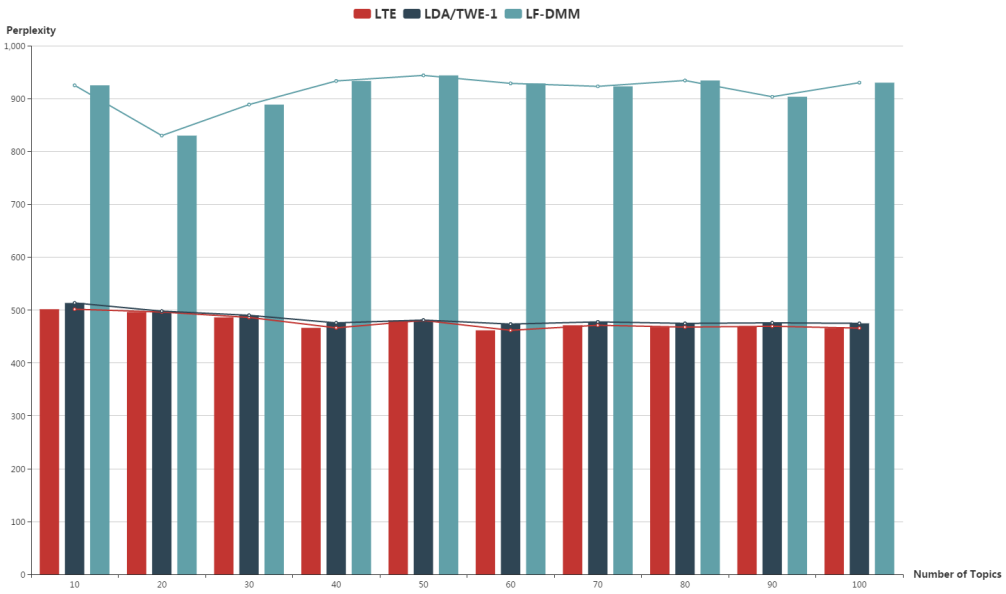


Figure 2: Perplexity on Holdout Data

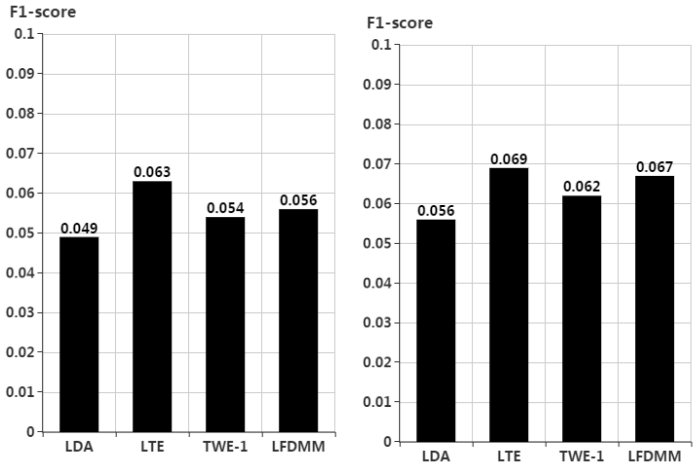
training of multinomial topics and embeddings does not harm the quality of the Multinomial topics. Rather, the joint training paradigm of LTE slightly improves the quality of the Multinomial topics. This observation verifies our assumption that the collectively utilizing co-occurrence information of different granularity has the potential of improving the performance of topic models.

5.3 Topical Word Extraction

We now evaluate the performance of LTE in the scenario of topical word extraction, which is critical for natural language understanding in modern search engines. Given a document, the goal of topical word extraction is to find some words that are highly relevant to the document theme. Conventionally, LDA plays an important role in topical word extraction (Zhao et al., 2011)(Pasquier, 2010). The existing methods based LDA are usually plagued by the weakness of capturing the semantics of words with low frequency. In this section, we study whether the embeddings generated by LTE are able to alleviate this problem. Ten thousands Web pages are utilized for this evaluation and the ground truth (i.e., the words that are highly relevant to the document theme) is manually prepared by human experts.

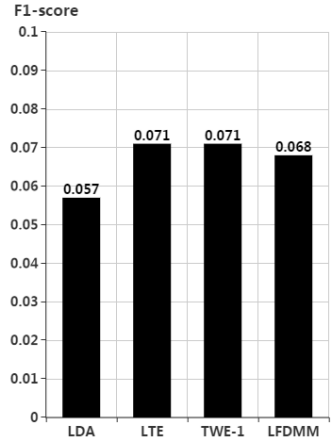
To derive the topical words for a document d, we first calculate the score of each word w in d and the score reflect the relevance between w and the themes of d. Then we sort all the words according to their scores and select the top-k words as the topical words of d. For TWE-1, LFDMM and LTE, the score of a word w is calculated based on embeddings by $score(w) = \sum_z P(z|d) \cos(v_w, v_z)$, where cos is the cosine similarity between two embeddings. As for LDA, we rely on the multinomial topics and calculate the score by $score(w) = \sum_z P(z|d)P(w|z)$. We compare the performance of these models in terms of $F1$ score, which is the harmonic mean of precision and recall.

The experimental result is shown in Figure 3. The models under-study tend to have higher $F1$ scores when the number of topical words increases. We observe that LDA always demonstrates the worst performance. The reason is that LDA is prone to select the frequent words and risks missing some words highly relevant to the document theme. In contrast, embedding information is less sensitive to the effect of word frequency. Therefore, TWE-1, LFDMM and LTE demonstrate better performance than LDA when the number of topical words varies from 3 to 10. LTE always demonstrates the highest $F1$ score. Comparing to TWE-1 and LFDMM which either reuse the output of LDA or Word2Vec, LTE jointly trains the Multinomial parameters and the embeddings, which are complimentary to each other and is effective to result in better topic modeling results and embeddings.

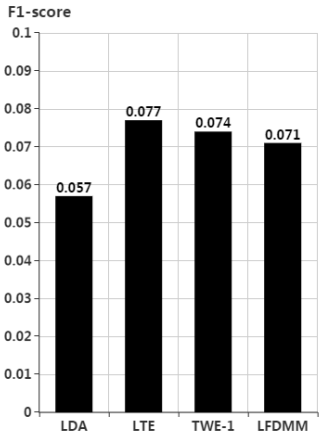


(a) Topical Word Amount = 3

(b) Topical Word Amount = 5



(c) Topical Word Amount = 7



(d) Topical Word Amount = 10

Figure 3: Topical Word Extraction

6 Conclusion

In this paper, we propose LTE to seamlessly integrate topic model and word embedding into one joint learning framework. We discuss a Monte Carlo EM algorithm for learning the parameter of LTE. LTE does not only output topic-related distributions but also generates distributed representation for words and latent topics. By applying LTE, we obtain coherent latent topics and the embedding generated by LTE are effective for identifying topical words of documents. Extensive experiments verify our assumption that topic modeling and word embedding are potentially complimentary for each other. While LTE is a specific model for off-the-shelf usage, the technique discussed in this paper can be easily transfer to many other scenarios where integrating other topic modeling and word embedding techniques are needed.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments and suggestions. This work is supported by National Basic Research Program of China (973 program No. 2014CB340505) .

References

David Bamman, Chris Dyer, and A. Noah Smith. 2014. Distributed representations of geographically situated language. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 828–834. Association for Computational Linguistics.

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

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Gabriel Doyle and Charles Elkan. 2009. Accounting for burstiness in topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 281–288. ACM.

Chris Dyer. 2014. Notes on noise contrastive estimation and negative sampling. *arXiv preprint arXiv:1410.8251*.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235.

Di Jiang, Kenneth Wai-Ting Leung, Wilfred Ng, and Hao Li. 2013. Beyond click graph: Topic modeling for search engine query log analysis. In *International Conference on Database Systems for Advanced Applications*, pages 209–223. Springer.

Yohan Jo and Alice H Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824. ACM.

Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. 2009. Latent dirichlet allocation for tag recommendation. In *Proceedings of the third ACM conference on Recommender systems*, pages 61–68. ACM.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *ICML*.

Shaohua Li, Tat-Seng Chua, Jun Zhu, and Chunyan Miao. 2016. Generative topic embedding: a continuous representation of documents. In *Proceedings of The 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2009. Distributed algorithms for topic models. *The Journal of Machine Learning Research*, 10:1801–1828.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics*, 3:299–313.

Claude Pasquier. 2010. Task 5: Single document keyphrase extraction using sentence clustering and latent dirichlet allocation. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 154–157. Association for Computational Linguistics.

Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 487–494. AUAI Press.

Hanna M Wallach. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984. ACM.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433. ACM.

Yi Wang, Hongjie Bai, Matt Stanton, Wen-Yen Chen, and Edward Y Chang. 2009. Plda: Parallel latent dirichlet allocation for large-scale applications. In *Algorithmic Aspects in Information and Management*, pages 301–314. Springer.

Gu Xu, Shuang-Hong Yang, and Hang Li. 2009. Named entity mining from click-through data using weakly supervised latent dirichlet allocation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1365–1374. ACM.

Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim, and Xiaoming Li. 2011. Topical keyphrase extraction from twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 379–388. Association for Computational Linguistics.

Neural-based Noise Filtering from Word Embeddings

Kim Anh Nguyen and **Sabine Schulte im Walde** and **Ngoc Thang Vu**
Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
Pfaffenwaldring 5B, 70569 Stuttgart, Germany
{nguyenkh,schulte,thangvu}@ims.uni-stuttgart.de

Abstract

Word embeddings have been demonstrated to benefit NLP tasks impressively. Yet, there is room for improvement in the vector representations, because current word embeddings typically contain unnecessary information, i.e., *noise*. We propose two novel models to improve word embeddings by unsupervised learning, in order to yield word denoising embeddings. The word denoising embeddings are obtained by strengthening salient information and weakening noise in the original word embeddings, based on a deep feed-forward neural network filter. Results from benchmark tasks show that the filtered word denoising embeddings outperform the original word embeddings.

1 Introduction

Word embeddings aim to represent words as low-dimensional dense vectors. In comparison to distributional count vectors, word embeddings address the problematic sparsity of word vectors and achieved impressive results in many NLP tasks such as sentiment analysis (e.g., Kim (2014)), word similarity (e.g., Pennington et al. (2014)), and parsing (e.g., Lazaridou et al. (2013)). Moreover, word embeddings are attractive because they can be learned in an unsupervised fashion from unlabeled raw corpora. There are two main approaches to create word embeddings. The first approach makes use of neural-based techniques to learn word embeddings, such as the Skip-gram model (Mikolov et al., 2013). The second approach is based on matrix factorization (Pennington et al., 2014), building word embeddings by factorizing word-context co-occurrence matrices.

In recent years, a number of approaches have focused on improving word embeddings, often by integrating lexical resources. For example, Adel and Schütze (2014) applied coreference chains to Skip-gram models in order to create word embeddings for antonym identification. Pham et al. (2015) proposed an extension of a Skip-gram model by integrating synonyms and antonyms from WordNet. Their extended Skip-gram model outperformed a standard Skip-gram model on both general semantic tasks and distinguishing antonyms from synonyms. In a similar spirit, Nguyen et al. (2016) integrated distributional lexical contrast into every single context of a target word in a Skip-gram model for training word embeddings. The resulting word embeddings were used in similarity tasks, and to distinguish between antonyms and synonyms. Faruqui et al. (2015) improved word embeddings without relying on lexical resources, by applying ideas from sparse coding to transform dense word embeddings into sparse word embeddings. The dense vectors in their models can be transformed into sparse overcomplete vectors or sparse binary overcomplete vectors. They showed that the resulting vector representations were more similar to interpretable features in NLP and outperformed the original vector representations on several benchmark tasks.

In this paper, we aim to improve word embeddings by reducing their noise. The hypothesis behind our approaches is that word embeddings contain unnecessary information, i.e. *noise*. We start out with the idea of learning word embeddings as suggested by Mikolov et al. (2013), relying on the distributional hypothesis (Harris, 1954) that words with similar distributions have related meanings. We address those

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2699–2707, Osaka, Japan, December 11-17 2016.

distributions in embedded vectors of words that decrease the value of such vector representations. For instance, consider the sentence *the quick brown fox gazing at the cloud jumped over the lazy dog*. The context *jumped* can be used to predict the words *fox*, *cloud* and *dog* in a window size of 5 words; however, a *cloud* cannot *jump*. The context `jumped` is therefore considered as noise in the embedded vector of `cloud`. We propose two novel models to smooth word embeddings by filtering noise: We strengthen salient contexts and weaken unnecessary contexts.

The first proposed model is referred to as *complete word denoising embeddings model (CompEmb)*. Given a set of original word embeddings, we use a filter to learn a denoising matrix, and then project the set of original word embeddings into this denoising matrix to produce a set of complete word denoising embeddings. The second proposed model is referred to as *overcomplete word denoising embeddings model (OverCompEmb)*. We make use of a sparse coding method to transform an input set of original word embeddings into a set of overcomplete word embeddings, which is considered as the "overcomplete process". We then apply a filter to train a denoising matrix, and thereafter project the set of original word embeddings into the denoising matrix to generate a set of overcomplete word denoising embeddings. The key idea in our models is to use a filter for learning the denoising matrix. The architecture of the filter is a feed-forward, non-linear and parameterized neural network with a fixed depth that can be used to learn the denoising matrices and reduce noise in word embeddings. Using state-of-the-art word embeddings as input vectors, we show that the resulting word denoising embeddings outperform the original word embeddings on several benchmark tasks such as word similarity and word relatedness tasks, synonymy detection and noun phrase classification. Furthermore, the implementation of our models is made publicly available[1].

The remainder of this paper is organized as follows: Section 2 presents the two proposed models, the loss function, and the sparse coding technique for overcomplete vectors. In Section 3, we demonstrate the experiments on evaluating the effects of our word denoising embeddings, tuning hyperparameters, and we analyze the effects of filter depth. Finally, Section 4 concludes the paper.

2 Learning Word Denoising Embeddings

In this section, we present the two contributions of this paper. Figure 1 illustrates our two models to learn denoising for word embeddings. The first model on the top, the complete word denoising embeddings model "CompEmb" (Section 2.1), filters noise from word embeddings $\mathbf{X}$ to produce complete word denoising embeddings $\mathbf{X}^*$, in which the vector length of $\mathbf{X}^*$ in comparison to $\mathbf{X}$ is unchanged after denoising (called *complete*). The second model at the bottom of the figure, the overcomplete word denoising embeddings model "OverCompEmb" (Section 2.2), filters noise from word embeddings $\mathbf{X}$ to yield overcomplete word denoising embeddings $\mathbf{Z}^*$, in which the vector length of $\mathbf{Z}^*$ tends to be greater than the vector length of $\mathbf{X}$ (called *overcomplete*).

For the notations, let $\mathbf{X} \in \mathbb{R}^{V \times L}$ is an input set of word embeddings in which V is the vocabulary size, and L is the vector length of $\mathbf{X}$. Furthermore, $\mathbf{Z} \in \mathbb{R}^{V \times K}$ is the overcomplete word embeddings in which K is the vector length of $\mathbf{Z}$ ($K > L$); finally, $\mathbf{D} \in \mathbb{R}^{L \times L}$ is the pre-trained dictionary (Section 2.4).

2.1 Complete Word Denoising Embeddings

In this subsection, we aim to reduce noise in the given input word embeddings $\mathbf{X}$ by learning a denoising matrix $\mathbf{Q_c}$. The complete word denoising embeddings $\mathbf{X}^*$ are then generated by projecting $\mathbf{X}$ into $\mathbf{Q_c}$. More specifically, given an input $\mathbf{X} \in \mathbb{R}^{V \times L}$, we seek to optimize the following objective function:

$$\underset{\mathbf{X},\mathbf{Q_c},\mathbf{S}}{\operatorname{argmin}} \sum_{i=1}^{V} \|\mathbf{x_i} - f(\mathbf{x_i}, \mathbf{Q_c}, \mathbf{S})\| + \alpha \|\mathbf{S}\|_1 \tag{1}$$

where f is a filter; $\mathbf{S}$ is a lateral inhibition matrix; and α is a regularization hyperparameter. Inspired by studies on sparse modeling, the matrix $\mathbf{S}$ is chosen to be symmetric and has zero on the diagonal.

[1] `https://github.com/nguyenkh/NeuralDenoising`

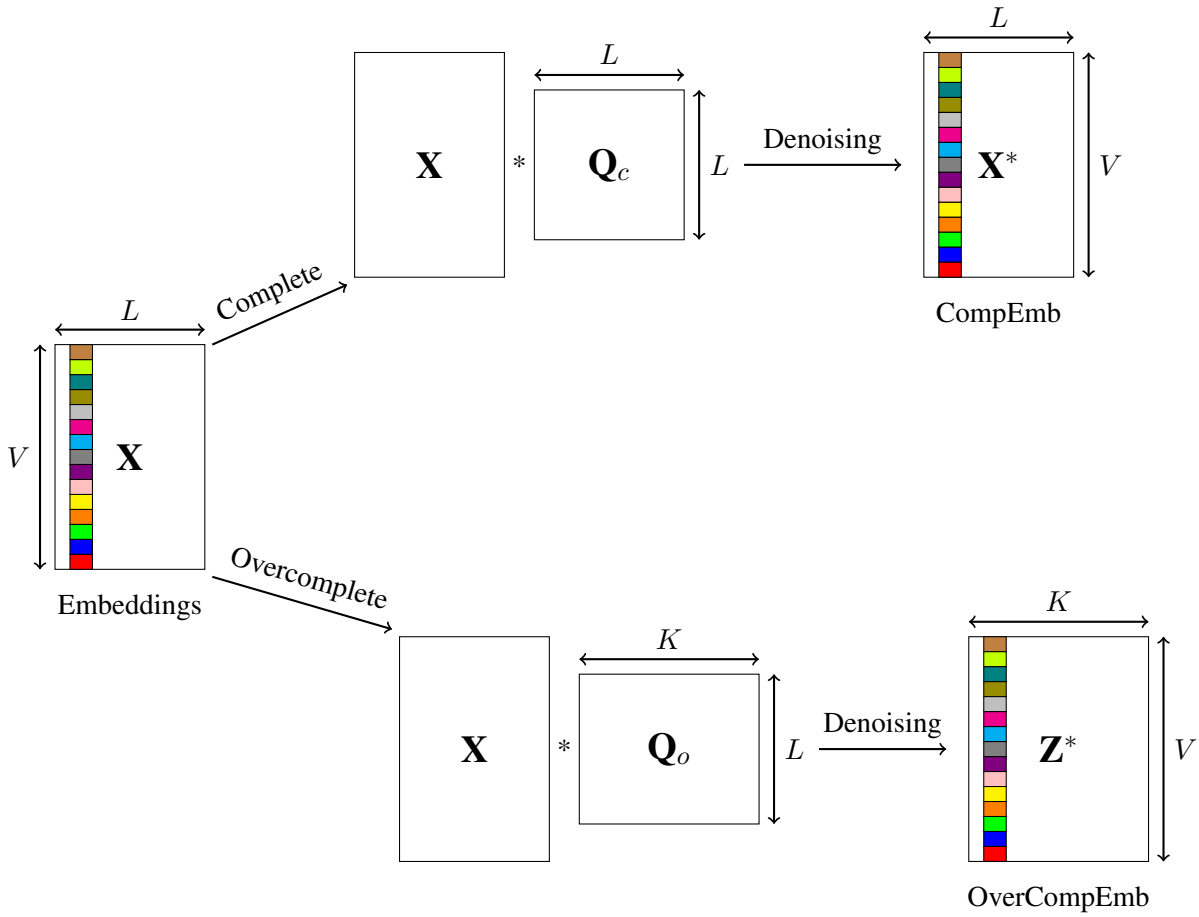


Figure 1: Illustration of word denoising embeddings methods, with complete word denoising embeddings at the top, and overcomplete word denoising embeddings at the bottom.

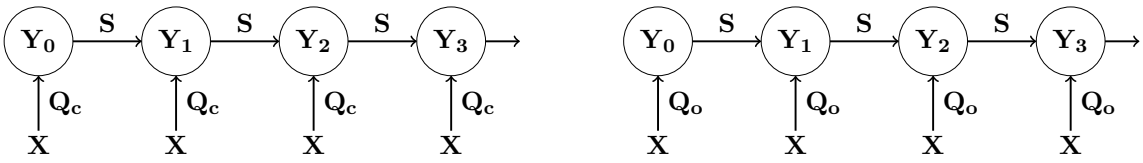
The goal of this matrix is to implement excitatory interaction between neurons, and to increase the convergence speed of the neural network (Szlam et al., 2011). More concretely, the matrices $\mathbf{Q_c}$ and $\mathbf{S}$ are initialized with $\mathbf{I}$ and E, which are identity matrices, and the Lipschitz constant:

$$\begin{aligned}&\mathbf{Q_c} = \tfrac{1}{E}\mathbf{D}; \mathbf{S} = \mathbf{I} - \tfrac{1}{E}\mathbf{D^T D}\\ &E > \text{ the largest eigenvalue of } \mathbf{D^T D}\\ &\mathbf{D} \in \mathbb{R}^{L\times L} \text{ be pre-trained dictionary}\end{aligned}$$

The underlying idea for reducing noise is to make use of a filter f to learn a denoising matrix $\mathbf{Q_c}$; hence, we design the filter f as a non-linear, parameterized, feed-forward architecture with a fixed depth that can be trained to approximate $f(\mathbf{X}, \mathbf{Q_c}, \mathbf{S})$ to $\mathbf{X}$ as in Figure 2a. As a result, noise from word embeddings will be filtered by layers of the filter f. The filter f is encoded as a recursive function by iterating over the number of fixed depth T, as the following recursive Equation 2 shows:

$$\begin{aligned}&\mathbf{Y} = f(\mathbf{X}, \mathbf{Q_c}, \mathbf{S})\\ &\mathbf{Y}(0) = \mathcal{G}(\mathbf{X}\mathbf{Q_c})\\ &\mathbf{Y}(k+1) = \mathcal{G}(\mathbf{X}\mathbf{Q_c} + \mathbf{Y}(k)\mathbf{S})\\ &0 \le k < T\end{aligned} \tag{2}$$

$\mathcal{G}$ is a non-linear activation function. The matrices $\mathbf{Q_c}$ and $\mathbf{S}$ are learned to produce the lowest possible error in a given number of iterations. Matrix $\mathbf{S}$, in the architecture of filter f, acts as a controllable matrix to filter unnecessary information on embedded vectors, and to impose restrictions on further reducing the



(a) CompEmb (b) OverCompEmb

Figure 2: Architecture of the filters with the fixed depth $T = 3$.

computational burden (e.g., solving low-rank approximation problem or keeping the number of terms at zero (Gregor and LeCun, 2010)). Moreover, the initialization of the matrices $\mathbf{Q_c}$, $\mathbf{S}$ and E enhances a highly efficient minimization of the objective function in Equation 1, due to the pre-trained dictionary $\mathbf{D}$ that carries the information of reconstructing $\mathbf{X}$.

The architecture of the filter f is a recursive feed-forward neural network with the fixed depth T, so the number of T plays a significant role in controlling the approximation of $\mathbf{X}^*$. The effects of T will be discussed later in Section 3.4. When $\mathbf{Q_c}$ is trained, the complete word denoising embeddings $\mathbf{X}^*$ are yielded by projecting $\mathbf{X}$ into $\mathbf{Q_c}$, as shown by the following Equation 3:

$$\mathbf{X}^* = \mathcal{G}(\mathbf{X}\mathbf{Q_c}) \tag{3}$$

2.2 Overcomplete Word Denoising Embeddings

Now we introduce our method to reduce noise and overcomplete vectors in the given input word embeddings. To obtain overcomplete word embeddings, we first use a sparse coding method to transform the given input word embeddings $\mathbf{X}$ into overcomplete word embeddings $\mathbf{Z}$. Secondly, we use overcomplete word embeddings $\mathbf{Z}$ as the intermediate word embeddings to optimize the objective function: A set of input word embeddings $\mathbf{X} \in \mathbb{R}^{V \times L}$ is transformed to overcomplete word embeddings $\mathbf{Z} \in \mathbb{R}^{V \times K}$ by applying sparse coding method in Section 2.4. We then make use of the pre-trained dictionary $\mathbf{D} \in \mathbb{R}^{L \times K}$ and $\mathbf{Z} \in \mathbb{R}^{V \times K}$ to learn the denoising matrix $\mathbf{Q_o}$ by minimizing the following Equation 4:

$$\underset{\mathbf{X},\mathbf{Q_o},\mathbf{S}}{\operatorname{argmin}} \sum_{i=1}^{V} \|\mathbf{z_i} - f(\mathbf{x_i}, \mathbf{Q_o}, \mathbf{S})\| + \alpha\|\mathbf{S}\|_1 \tag{4}$$

The initialization of the parameters $\mathbf{Q_o}$, $\mathbf{S}$, E and α follows the same procedure as described in Section 2.1, and with the same interpretation of the filter architecture in Figure 2b. The overcomplete word denoising embeddings $\mathbf{Z}^*$ are then generated by projecting $\mathbf{X}$ into the denoising matrix $\mathbf{Q_o}$ and using the non-linear activation function $\mathcal{G}$ in the following Equation 5:

$$\mathbf{Z}^* = \mathcal{G}(\mathbf{X}\mathbf{Q_o}) \tag{5}$$

2.3 Loss Function

For each pair of term vectors $\mathbf{x_i} \in \mathbf{X}$ and $\mathbf{y_i} \in \mathbf{Y} = f(\mathbf{X}, \mathbf{Q_c}, \mathbf{S})$, we make use of the cosine similarity to measure the similarity between $\mathbf{x_i}$ and $\mathbf{y_i}$ as follows:

$$sim(\mathbf{x_i}, \mathbf{y_i}) = \frac{\mathbf{x_i} \cdot \mathbf{y_i}}{\|\mathbf{x_i}\|\|\mathbf{y_i}\|} \tag{6}$$

Let Δ be the difference between $sim(\mathbf{x_i}, \mathbf{x_i})$ and $sim(\mathbf{x_i}, \mathbf{y_i})$, equivalently $\Delta = 1 - sim(\mathbf{x_i}, \mathbf{y_i})$. We then optimize the objective function in Equation 1 by minimizing Δ; and the same loss function is also applied to optimize the objective function in Equation 4. Training is done through Stochastic Gradient Descent with the Adadelta update rule (Zeiler, 2012).

2.4 Sparse Coding

Sparse coding is a method to represent vector representations as a sparse linear combination of elementary atoms of a given dictionary. The underlying assumption of sparse coding is that the input vectors can be reconstructed accurately as a linear combination of some basis vectors and a few number of non-zero coefficients (Olshausen and Field, 1996).

The goal is to approximate a dense vector in $\mathbb{R}^L$ by a sparse linear combination of a few columns of a matrix $\mathbf{D} \in \mathbb{R}^{L \times K}$ in which K is a new vector length and the matrix $\mathbf{D}$ be called a *dictionary*. Concretely, given V input vectors of L dimensions $\mathbf{X} = [x_1, x_2, ..., x_V]$, the dictionary and sparse vectors can be formulated as the following minimization problem:

$$\min_{\mathbf{D} \in \mathcal{C}, \mathbf{Z} \in \mathbb{R}^{K \times V}} \sum_{i=1}^{V} \|\mathbf{x_i} - \mathbf{D}\mathbf{z}_i\|_2^2 + \lambda \|\mathbf{z}_i\|_1 \tag{7}$$

$\mathbf{Z} = [z_1, ..., z_V]$ carries the decomposition coefficients of $\mathbf{X} = [x_1, x_2, ..., x_V]$; and λ represents a scalar to control the sparsity level of $\mathbf{Z}$. The dictionary $\mathbf{D}$ is typically learned by minimizing Equation 7 over input vectors $\mathbf{X}$. In the case of overcomplete representations $\mathbf{Z}$, the vector length K is typically implied as $K = \gamma L$ ($\gamma > 0$).

In the method of overcomplete word denoising embeddings (Section 2.2), our approach makes use of overcomplete word embeddings $\mathbf{Z}$ as the intermediate word embeddings reconstructed by applying a sparse coding method to word embeddings $\mathbf{X}$. The overcomplete word embeddings $\mathbf{Z}$ are then utilized to optimize Equation 4. To obtain overcomplete word embeddings $\mathbf{Z}$ and dictionaries, we use the `SPAMS` package[2] to implement sparse coding for word embeddings $\mathbf{X}$ and to train the dictionaries $\mathbf{D}$.

3 Experiments

3.1 Experimental Settings

As input word embeddings, we rely on two state-of-the-art word embeddings methods: word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). We use the `word2vec` tool[3] and the web corpus *ENCOW14A* (Schäfer and Bildhauer, 2012; Schäfer, 2015) which contains approximately 14.5 billion tokens, in order to train Skip-gram models with 100 and 300 dimensions. For the GloVe method, we use pre-trained vectors of 100 and 300 dimensions[4] that were trained on 6 billion words from Wikipedia and English Gigaword. The $\tanh$ function is used as the non-linear activation function in both approaches. The fixed depth of filter T is set to 3; further hyperparameters are chosen as discussed in Section 3.2. To train the networks, we use the `Theano` framework (Theano Development Team, 2016) to implement our models with a mini-batch size of 100. Regularization is applied by dropouts of 0.5 and 0.2 for input and output layers (without tuning), respectively.

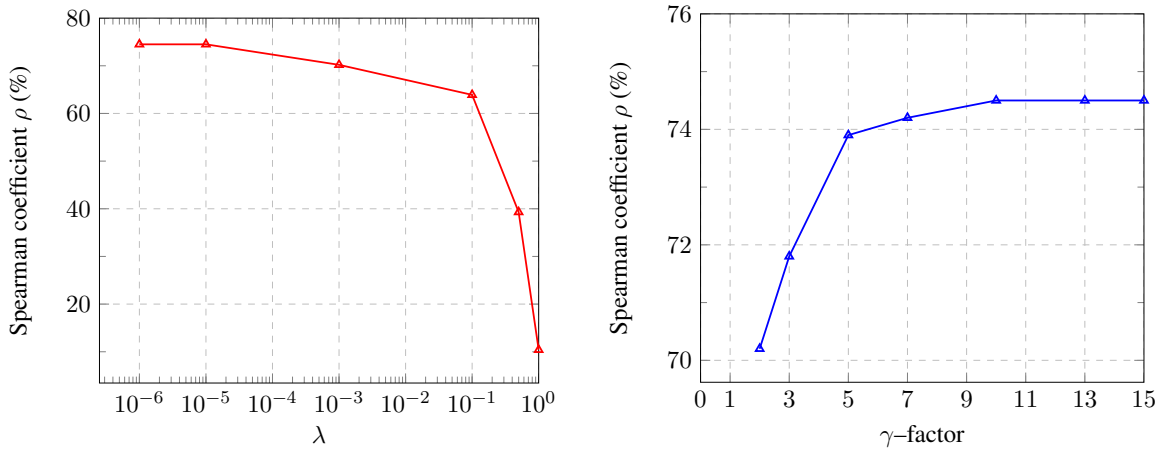
3.2 Hyperparameter Tuning

In both methods of denoising word embeddings, the ℓ_1 regularization penalty α is set to 0.5 without tuning in Equation 1 and 4. The method of learning overcomplete word denoising embeddings relies on the mediate word embeddings $\mathbf{Z}$ to minimize the objective function in Equation 4. The sparsity of $\mathbf{Z}$ depends on the ℓ_1 regularization λ in Equation 7; and the length vector K of $\mathbf{Z}$ is implied as $K = \gamma L$. Therefore, we aim to tune λ and γ such that $\mathbf{Z}$ represents the nearest approximation of the original vector representation $\mathbf{X}$. We perform a grid search on $\lambda \in \{1.0, 0.5, 0.1, 10^{-3}, 10^{-6}\}$ and $\gamma \in \{2, 3, 5, 7, 10, 13, 15\}$, developing on the word similarity task WordSim353 (to be discussed on Section 3.3). The hyperparameter tunings are illustrated in Figures 3a and 3b for sparsity and overcomplete vector length tuning, respectively. In both approaches, we set λ to 10^{-6} and γ to 10 for the sparsity and length of overcomplete word embeddings.

[2] http://spams-devel.gforge.inria.fr

[3] https://code.google.com/p/word2vec/

[4] http://www-nlp.stanford.edu/projects/glove/



(a) Sparsity of sparse coding.

(b) Length of overcomplete vectors.

Figure 3: Illustration of hyperparameter tuning.

3.3 Effects of Word Denoising Embeddings

In this section, we quantify the effects of word denoising embeddings on three kinds of tasks: similarity and relatedness tasks, detecting synonymy, and bracketed noun phrase classification task. In comparison to the performance of word denoising embeddings, we take into account state-of-the-art word embeddings (Skip-gram and GloVe word embeddings) as baselines. Besides, we also use the public source code[5] to re-implement the two methods suggested by Faruqui et al. (2015) which are vectors $\mathbf{A}$ (sparse overcomplete vectors) and $\mathbf{B}$ (sparse binary overcomplete vectors).

The effects of the word denoising embeddings on the tasks are shown in Table 1. The results show that the vectors $\mathbf{X}^*$ and $\mathbf{Z}^*$ outperform the original vectors $\mathbf{X}$, $\mathbf{A}$ and $\mathbf{B}$, except for the NP task, in which the vectors $\mathbf{B}$ based on the 300-dimensional GloVe vectors are best. The effect of the vectors $\mathbf{Z}^*$ is slightly less impressive, when compared to the overcomplete vectors $\mathbf{X}^*$. The overcomplete word embeddings $\mathbf{Z}$ strongly differ from the word embeddings $\mathbf{X}$; hence, the denoising is affected. However, the performance of the vectors $\mathbf{Z}^*$ still outperforms the original vectors $\mathbf{X}$, $\mathbf{A}$ and $\mathbf{B}$ after the denoising process.

3.3.1 Relatedness and Similarity Tasks

For the relatedness task, we use two kinds of datasets: MEN (Bruni et al., 2014) consists of 3000 word pairs comprising 656 nouns, 57 adjectives and 38 verbs. The WordSim-353 relatedness dataset (Finkelstein et al., 2001) contains 252 word pairs. Concerning the similarity tasks, we evaluate the denoising vectors again on two kinds of datasets: *SimLex-999* (Hill et al., 2015) contains 999 word pairs including 666 noun, 222 verb and 111 adjective pairs. The WordSim-353 similarity dataset consists of 203 word pairs. In addition, we evaluate our denoising vectors on the WordSim-353 dataset which contains 353 pairs for both similarity and relatedness relations. We calculate cosine similarity between the vectors of two words forming a test pair, and report the Spearman rank-order correlation coefficient ρ (Siegel and Castellan, 1988) against the respective gold standards of human ratings.

3.3.2 Synonymy

We evaluate on 80 TOEFL (Test of English as a Foreign Language) synonym questions (Landauer and Dumais, 1997) and 50 ESL (English as a Second Language) questions (Turney, 2001). The first dataset represents a subset of 80 multiple-choice synonym questions from the TOEFL test: a word is paired with four options, one of which is a valid synonym. The second dataset contains 50 multiple-choice synonym questions, and the goal is to choose a valid synonym from four options. For each question, we compute the cosine similarity between the target word and the four candidates. The suggested answer is the candidate with the highest cosine score. We use accuracy to evaluate the performance.

[5] https://github.com/mfaruqui/sparse-coding

Vectors		Simlex-999 Corr.	MEN Corr.	WS353 Corr.	WS353-SIM Corr.	WS353-REL Corr.	ESL Acc.	TOEFL Acc.	NP Acc.
SG-100	**X**	33.7	72.9	69.7	74.5	65.5	48.9	62.0	72.8
	X*	33.2	72.8	70.6	74.8	66.0	53.0	**64.5**	78.5
	Z*	**35.9**	**74.4**	**71.2**	**75.2**	**68.1**	53.0	62.0	**79.1**
	A	32.5	69.8	65.5	69.5	60.2	**55.1**	51.8	78.8
	B	31.9	70.4	65.8	72.6	62.2	53.0	58.2	74.1
SG-300	**X**	36.1	74.7	71.0	75.9	66.1	**59.1**	72.1	77.9
	X*	**37.1**	**75.8**	**71.8**	**76.4**	**66.9**	**59.1**	74.6	**79.3**
	Z*	36.5	75.0	70.6	**76.4**	64.4	57.1	**77.2**	78.6
	A	32.9	72.4	67.5	71.9	63.4	53.0	65.8	78.3
	B	32.7	71.2	63.3	68.7	56.2	51.0	70.8	78.6
GloVe-100	**X**	29.7	69.3	52.9	60.3	49.5	46.9	82.2	76.4
	X*	**31.7**	**70.9**	**58.0**	**63.8**	**57.3**	53.0	**88.6**	**77.4**
	Z*	30.0	**70.9**	56.0	62.8	53.8	**57.0**	81.0	77.3
	A	30.7	70.7	54.9	62.2	51.2	55.1	78.4	77.1
	B	31.0	69.2	57.3	62.3	53.7	46.9	73.4	76.4
GloVe-300	**X**	37.0	74.8	60.5	66.3	57.2	**61.2**	89.8	74.3
	X*	**40.2**	**76.8**	**64.9**	**69.8**	**62.0**	**61.2**	**92.4**	76.3
	Z*	39.0	75.2	63.0	67.9	59.7	57.1	86.0	75.7
	A	36.7	74.1	61.5	67.7	57.8	55.1	87.3	79.9
	B	33.1	70.2	57.0	62.2	53.0	51.0	91.4	**80.0**

Table 1: Effects of word denoising embeddings. Vectors **X** represent the baselines; vectors **A** and **B** were suggested by Faruqui et al. (2015); the vector length **Z*** is equal to 10 times of vector length **X**.

3.3.3 Phrase parsing as Classification

Lazaridou et al. (2013) introduced a dataset of noun phrases (NP) in which each NP consists of three elements: the first element is either an adjective or a noun, and the other elements are all nouns. For a given NP (such as *blood pressure medicine*), the task is to predict whether it is a left-bracketed NP, e.g., *(blood pressure) medicine*, or a right-bracketed NP, e.g., *blood (pressure medicine)*.

The dataset contains 2227 noun phrases split into 10 folds. For each NP, we use the average of word vectors as features to feed into the classifier by tuning the hyperparameters (w_1, w_2 and w_3) for each element (e_1, e_2 and e_3) within the NP: $\vec{e}_{NP} = \frac{1}{3}(w_1\vec{e}_1 + w_2\vec{e}_2 + w_3\vec{e}_3)$. We then employ the classification of the NPs by using a Support Vector Machine (SVM) with Radial Basis Function kernel. The classifier is tuned on the first fold, and cross-validation accuracy is reported on the nine remaining folds.

3.4 Effects of Filter Depth

As mentioned above, the architecture of the filter f is a feed-forward network with a fixed depth T. For each stage T, the filter f attempts to reduce the noise within input vectors by approximating these vectors based on vectors of a previous stage $T-1$. In order to investigate the effects of each stage T, we use pre-trained GloVe vectors with 100 dimensions to evaluate the denoising performance of the vectors on detecting synonymy in the TOEFL dataset across several stages of T.

The results are presented in Figure 4. The accuracy of synonymy detection increases sharply from 63.2% to 88.6% according to the number of stages T from 0 to 3. However, the denoising performance of vectors falls with the number of stages $T > 3$. This evaluation shows that the filter f with a consistently fixed depth T can be trained to efficiently filter noise for word embeddings. In other words, the number of stages T exceeds a consistent number T (with $T > 3$ in our case), leading to the loss of salient information in the vectors.

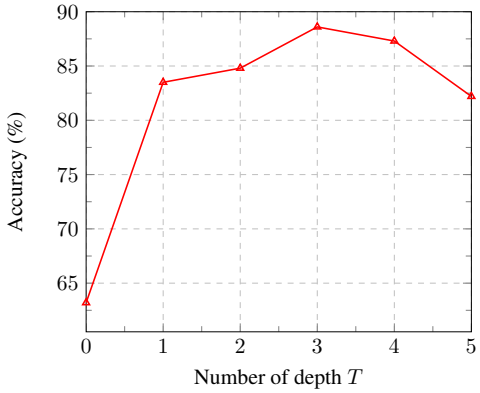


Figure 4: Effects of the filter with depth T on filtering noise.

4 Conclusion

To the best of our knowledge, we are the first to work on filtering noise in word embeddings. In this paper, we have presented two novel models to improve word embeddings by reducing noise in state-of-the-art word embeddings models. The underlying idea in our models was to make use of a deep feed-forward neural network filter to reduce noise. The first model generated complete word denoising embeddings; the second model yielded overcomplete word denoising embeddings. We demonstrated that the word denoising embeddings outperform the originally state-of-the-art word embeddings on several benchmark tasks.

Acknowledgements

The research was supported by the Ministry of Education and Training of the Socialist Republic of Vietnam (Scholarship 977/QD-BGDDT; Kim-Anh Nguyen), the DFG Collaborative Research Centre SFB 732 (Kim-Anh Nguyen, Ngoc Thang Vu), and the DFG Heisenberg Fellowship SCHU-2580/1 (Sabine Schulte im Walde).

References

Heike Adel and Hinrich Schütze. 2014. Using mined coreference chains as a resource for a semantic task. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1447–1452, Doha, Qatar.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artifical Intelligence Research (JAIR)*, 49:1–47.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1491–1500, Beijing, China.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on the World Wide Web*, pages 406–414.

Karol Gregor and Yann LeCun. 2010. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML), Haifa, Israel*, pages 399–406.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to Platos problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

Angeliki Lazaridou, Eva Maria Vecchi, and Marco Baroni. 2013. Fish transporters and miracle homes: How compositional distributional semantics can help NP parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1908–1913, Doha, Qatar.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 746–751, Atlanta, Georgia.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 454–459, Berlin, Germany.

Bruno A. Olshausen and David J. Field. 1996. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Nghia The Pham, Angeliki Lazaridou, and Marco Baroni. 2015. A multitask objective to inject lexical contrast into distributional semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 21–26, Beijing, China.

Roland Schäfer and Felix Bildhauer. 2012. Building large corpora from the web using a new efficient tool chain. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 486–493, Istanbul, Turkey.

Roland Schäfer. 2015. Processing and querying large web corpora with the COW14 architecture. In *Proceedings of the 3rd Workshop on Challenges in the Management of Large Corpora*, pages 28–34, Mannheim, Germany.

Sidney Siegel and N. John Castellan. 1988. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, Boston, MA.

Arthur D. Szlam, Karol Gregor, and Yann L. Cun. 2011. Structured sparse coding via lateral inhibition. *Advances in Neural Information Processing Systems (NIPS)*, 24:1116–1124.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Peter D. Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning (ECML)*, pages 491–502.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Integrating Distributional and Lexical Information for Semantic Classification of Words using MRMF

Rosa Tsegaye Aga
Hochschule Hannover
Hanover, Germany
rosa-tsegaye.aga@hs-hannover.de

Lucas Drumond
Universität Hildesheim
Hildesheim, Germany
ldrumond@gmail.com

Christian Wartena
Hochschule Hannover
Hanover, Germany
christian.wartena@hs-hannover.de

Lars Schmidt-Thieme
Universität Hildesheim
Hildesheim, Germany
schmidt-thieme@ismll.de

Abstract

Semantic classification of words using distributional features is usually based on the semantic similarity of words. We show on two different datasets that a trained classifier using the distributional features directly gives better results. We use Support Vector Machines (SVM) and Multi-relational Matrix Factorization (MRMF) to train classifiers. Both give similar results. However, MRMF, that was not used for semantic classification with distributional features before, can easily be extended with more matrices containing more information from different sources on the same problem. We demonstrate the effectiveness of the novel approach by including information from WordNet. Thus we show, that MRMF provides an interesting approach for building semantic classifiers that (1) gives better results than unsupervised approaches based on vector similarity, (2) gives similar results as other supervised methods and (3) can naturally be extended with other sources of information in order to improve the results.

1 Introduction

In this paper we consider the task of classifying words into a large number of semantic categories. For this, we use two different data sets: 1. A dataset which is used in literature (Bullinaria and Levy, 2007) - to enable compare our results with results reported in the literature, 2. A larger dataset that is derived from a large thesaurus. The second dataset comes close to practical applications for semantic word classification. Organizations maintaining thesauri usually try to keep their thesaurus up to date and frequently add new terminology to the thesaurus. For each new term, they have to decide at what point it has to be inserted. Automatic semantic classification supports exactly this task. However, the classifier should be able to choose from hundreds or even thousands of semantic classes, not just from a dozen.

For semantic word classification, it is a common approach to represent words by context features. Usually, co-occurrence statistics are used as context features. According to the distributional hypothesis, words with similar context features have a similar meaning. Thus, we can use any distance measure between the feature vectors as a measure of semantic similarity. These distances are now commonly used in a nearest neighbor or a nearest centroid (or nearest prototype) classifier. Recently, distributional features have also been used directly to train classifiers that classify pairs of words as being synonymous or not (Hagiwara, 2008; Weeds et al., 2014; Aga et al., 2016). In these approaches, first a vector representation for a pair is build, that is used by a machine learning algorithm. In the following we will use the distributional features directly to categorize the words into a large number of categories. We will also see that the supervised methods outperform the unsupervised ones.

For the given task, we obtain similar results with SVM and MRMF. However, MRMF enables easy integration of different sources which improves the results furthermore. The MRMF does not just aggregate results from different sources, but is also able to model the interaction between the different types of information. As a second source of information we use hypernym information from WordNet. Since, we

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2708–2717, Osaka, Japan, December 11-17 2016.

do not have a mapping from wordnet hypernym classes to our target classes, we have a second learning task. In the first place we show that classification using WordNet is possible, but gives worse results than classification based on distributional features. In the second place, MRMF using both distributional features and WordNet Hypernyms outperforms all other methods. For the SC53 data set that introduced by Bullinaria and Levy (2007) using the Montague and Battig (Battig and Montague, 1969) semantic classes, we get an accuracy of $0,93$. The best result found in literature for the complete data set is 0,86 (Bullinaria and Levy, 2012).

The rest of the paper is organized as follows. Section 2 briefly reviews the related work. Section 3 explains the methodology of the work in detail. Section 4 explains the multi-relational matrix factorization method in detail. Section 5 and Section 6 explain briefly the evaluation of the models and explain the result, respectively. Finally, Section 7 concludes the paper.

2 Related Work

In distributional semantics, words are represented by context features, usually co-occurrence numbers between words or the pointwise mutual information between each word and each context word. It turns out that words with a similar meaning have similar vectors of context features; In other words, semantically similar words occur in similar contexts (Rubenstein and Goodenough, 1965; Saif and Hirst, 2012; Bullinaria and Levy, 2007; Turney and Pantel, 2010; Bullinaria and Levy, 2012; Kiela and Clark, 2014).

Classification of words into different semantic categories was studied by Pekar et al. (2004), who use a k-Nearest Neighbor classifier and investigate different feature weighting schemes and distance measures; Fan and Friedman (2007) study the classification of medical terms using a nearest centroid classifier; Both Bullinaria and Levy (2012) and Keith et al. (2015) use a nearest centroid classifier for the same data set that is also included in our study. However, Keith et al. report only the result for one arbitrary split into test and training set. Thus, their results cannot be compared directly to our and also Bullinaria and Levy results.

Matrix factorization has been used in distributional semantics, e.g. by Giesbrecht (2010) and Van de Cruys et al. (2013) in order to reduce the size of the feature space, but not directly for predicting missing values or for classification. We are not aware of any work using matrix factorization for classification of words into semantic categories.

The integration of distributed and lexical information is an obvious way to go and was also used in a number of studies. Usually a (weighted) average of similarities based on different types of information is used. E.g. Finkelstein et al. (2001) used distributional features (occurrence frequencies of words in various domains) and the cosine of these feature vectors as a distributional similarity measure. This measure is combined linearly with a WordNet based similarity measure. Yih and Qazvinian (2012) use different similarity methods, like corpus based and web based distributional similarity for binary classification tasks (synonymous or not-synonymous). They also used WordNet similarity. For this, they represent a word as a vector in a Synset-space. The vector, thus indicates, to which synsets a word belongs. They finally aggregated the various similarities by taking the average cosine similarity. Camacho-Collados et al. (2015) combined distributional similarity of words based on their occurrence in Wikipedia with a WordNet based similarity measure. They also combined the similarities from both sources by computing the average. Pennacchiotti et al. (2008) also investigate the contribution of distributional models and their combination with Wordnet. They use the a simple back-off model to combine distributional similarity and Wordnet based similarity.

3 Methodology

The task that we considered is to classify words into their semantic category. In this section, we will describe the datasets, the feature construction for the representation of the words and the classification methods that we have used.

3.1 Data Description

Our first dataset is the same with the one used by Bullinaria and Levy (2007). This data set uses 53 of the 56 basic semantic categories introduced by Battig and Montague (1969). In total, the dataset contains 530 words which have been taken from 53 semantic categories. For each category there are 10 typical words. We will refer to this dataset as SC53.

We have compiled a second, similar but much larger dataset from the Eurovoc Thesaurus (Office for Official Publications of the European Communities, 1995). Eurovoc is a multilingual thesaurus developed by the European Commissions Publications Office as a controlled vocabulary for the manual indexation of documents. The Eurovoc thesaurus is divided into 127 micro-thesauri. From each of these micro-thesauri we took the top-level concepts, 528 in total, as semantic categories. For each category we collected all narrower concepts and considered their preferred and alternative labels as terms for that category. We then removed all terms that belong to more than one category or that consist of more than two words. Finally, we removed all categories for which less than 10 terms were found. Now 190 categories with a total of 2386 terms are left. After further cleaning the dataset by removing the words that have a very high or low frequencies in UkWaK, which is a corpus that has been used to construct word representation vectors, 1447 words with 95 semantic categories are left, each containing 10 to 44 terms. We call this dataset $Eurovoc$. Table 1 shows some examples from both datasets.

Dataset	Category	Words
SC53	Fruits	Orange, Strawberry, Banana
	Furniture	Chair, Table, Bed
Eurovoc	ACP countries	Bahamas, Barbados,Cameroon
	Health policy	Dispensary, Hospitalization

Table 1: Some example classifications from the used datasets

3.2 Feature Construction

We use two different representations for each word. The first one is a distributional representation based on word co-occurrences.The second one uses WordNet hypernyms. The two types of representation will be explained in the following subsections.

3.2.1 Distributional Representation

We construct vectors of co-occurring words to represent each word and use them as an input for all our experiments. For building the context vectors, we used UkWaC English corpus.

There are a number of choices that have to be made when building the context vectors for each word. In the following we will use the choices that turned out to yield the best results in a number of different tasks in recent studies by Bullinaria and Levy (2007; 2012) and Kiela and Clark (2014).

After some preliminary experiments we found that including all words in the frequency range from $4 \cdot 10^3$ to $1 \cdot 10^6$ in the UkWaC Corpus as context feature is a good compromise between optimal results and acceptable storage and computing efforts. Each word is now represented by a vector of 17 400 features. All experiments have been done using these distributional features.

Next we have to determine the size of the window for co-occurrence. If the training corpus is large enough all studies show that smaller windows yield better results. We first remove all stop words and then use a window size of two words on the stopped text while respecting sentence boundaries. Syntactic relations are not used to determine the context of a word.

We use positive pointwise mutual information (PPMI) as a degree of co-occurrence, since it was shown to give better results than raw co-occurrence probabilities in a number of different studies (see e.g. (Bullinaria and Levy, 2007; Bullinaria and Levy, 2012)). For a context word c and a (target) word t the PPMI is defined as

$$ppmi(c,t) = \max\left(\log\frac{p(c|t)}{p(c)}, 0\right). \quad (1)$$

3.2.2 WordNet Categories Representation

In order to classify words into semantic categories, we could directly use the semantic categories of the words from WordNet. However, we do not know the relation between the WordNet categories and the target categories. Moreover, our data set contains lots of terms that are not found in WordNet. Thus, we represent each word by the set of all its WordNet hypernyms, i.e. the transitive closure of the hypernym relation applied to each possible meaning of the word. E.g. the word *Mansion* is represented by the set {artifact.n.01, building.n.01, dwelling.n.01, entity.n.01, house.n.01, housing.n.01, location.n.01, mansion.n.02, object.n.01, physical_entity.n.01, region.n.01, sign_of_the_zodiac.n.01, structure.n.01, whole.n.02b }. Finally, each word is represented as a boolean vector in the space of all possible hypernyms.

For the SC53 dataset we could construct WordNet vectors for 520 out of 530 words; in the Eurovoc dataset, 1198 out of 1447 terms were found in WordNet. The average number of hypernyms for each term found in WordNet was 66. The total number of distinct hypernyms for all words is 2896 for the SC53 and 4938 for the Eurovoc data.

3.3 Classification Methods

Bullinaria and Levy (2012) use a nearest centroid classifier for classifying words based on distributional features. In this approach, for every semantic category a feature vector is created by averaging the feature values of all words in the training set belonging to that category. Now the cosine between the feature vector of the word and each centroid vector is computed and the word is assigned to the class with the closest center.

The second classification method is a support vector machine (SVM). We used linear SVM from the liblinear package (Fan et al., 2008) to learn a model and classify words, that words represented by feature vector, to their category. Liblinear is efficient for training, large-scale problems (Fan et al., 2008). The hyper-parameters of the models have been tuned using a grid search from LIBSVM. To find the best C parameter value, we tested the numbers in between 0 and 20 in step 0.05.

The third and main classification method that is used in this paper is multi-relational matrix factorization. We will explain MRMF in detail in the following section.

If we want to use both lexical and distributional information, we can use MRMF as we will show in the following section. An obvious alternative is an ensemble classifier, that uses the results of the classifiers using only one type of information. Thus we also trained an SVM on the results of the SVMs using WordNet and distributional features. Since we have only boolean results from the SVM (a word is assigned to a category or not) we use also a logistic regression classifier. Logistic regression gives probabilities for each class and selects the class with the highest probability. The ensemble classifier now can use the probabilities for each class. Though we expect logistic regression to be inferior to SVM, it might have an advantage to use its class probabilities in an ensemble classifier.

4 Multi-Relational Matrix Factorization

MRMF was introduced by Lippert et al. (2008) for relation prediction in multi-relational domains using matrix factorization. Weighted MRMF as we have used here, was defined by Drumond et al. (2014) to model the different degrees of influence various relations involved in a domain might have.

For MRMF we have three matrices that can be used: the matrix of words and semantic classes, the matrix of words and context features and the matrix of words and WordNet features. The task for MRMF now is to predict values for new words in the first matrix using one or both of the other matrices.

4.1 MRMF on two Matrices

For the problem to classify words that are represented by vectors of WordNet categories and context words, we have followed the same procedure.

Let's assume the following problem: We have

- m words;

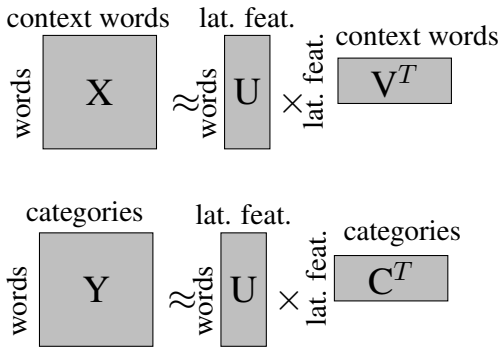


Figure 1: Visual overview of the matrix decomposition used for semantic categorization on two matrices

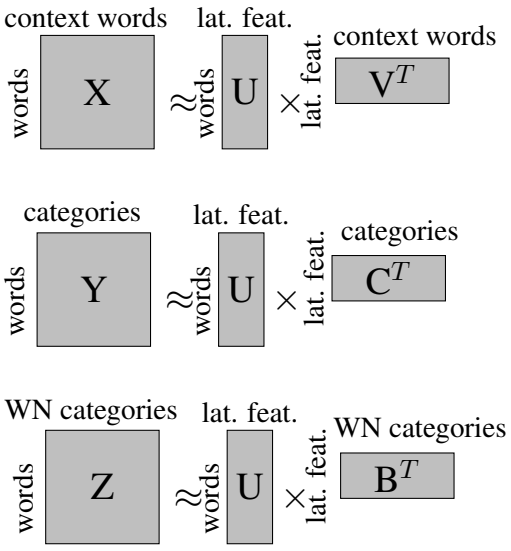


Figure 2: Visual overview of the matrix decomposition used for semantic categorization on three matrices.

Algorithm 1 Block coordinate descent optimization algorithm for L2-MRMF

procedure MRMF-COORDINATE DESCENT
input: X, Y, k, *weight constants* α_X, α_Y, *regularization constants* λ_U, λ_V, λ_C
  $U \sim \mathcal{N}(0, \sigma\mathbf{I})$
  $V \sim \mathcal{N}(0, \sigma\mathbf{I})$
  $C \sim \mathcal{N}(0, \sigma\mathbf{I})$
  repeat
    $U \leftarrow (\alpha_X XV + \alpha_Y YC)\left(\alpha_X V^T V + \alpha_Y C^T C - \lambda_U \mathbf{I}\right)^{-1}$
    $V \leftarrow \left(\left(\alpha_X U^T U - \lambda_V \mathbf{I}\right)^{-1} \alpha_X U^T X\right)^T$
    $C \leftarrow \left(\left(\alpha_Y U^T U - \lambda_C \mathbf{I}\right)^{-1} \alpha_Y U^T Y\right)^T$
  until convergence
  return U, V, C
end procedure

- n features for each word (e.g. positive point wise mutual information (PPMI) values based on the co-occurrence data);
- c semantic categories;

The features are represented by a matrix $X \in \mathbb{R}^{m \times n}$ where each row of X represents the feature vector of a word. We use a second matrix, $Y \in \{0, 1\}^{m \times c}$ with the relation between words and categories. $Y_{i,j}$ has value 1 if the word i belongs to the category c and 0 otherwise.

The idea of matrix factorization is that X can be approximated by the product of two smaller matrices U and V, where U is a matrix of words and latent features and V is a matrix of context features and the same latent features. The number k of latent features can be chosen freely with $k << n$. The second matrix, Y, can be decomposed in the same way. The idea of MRMF is that both decompositions use the same factor matrix U of words and latent features. Thus the latent features now form the link between the context features and the categories. The situation is visualized in Figure 1. The matrices U, V and C are constructed using the training data. If we have a new word w in the test data, we can add it to X and compute it's latent features using V, and thus extend U. From the extended matrix U and C we get the new row for w in Y, that gives us the classification for w.

More formally, X and Y can be factorized as:

$$X \approx UV^T \tag{2}$$

$$Y \approx UC^T \tag{3}$$

for some $U \in \mathbb{R}^{m \times k}$, $V \in \mathbb{R}^{n \times k}$ and $C \in \mathbb{R}^{c \times k}$. The problem is now to minimize the following objective with respect to L2 loss function

$$\begin{aligned} \underset{U,V,B,C}{\arg\min} \quad & \alpha_X \frac{1}{2}||X - UV^T||_F^2 + \alpha_Y \frac{1}{2}||Y - UC^T||_F^2 \\ & + \frac{\lambda_U}{2}||U||_F^2 + \frac{\lambda_V}{2}||V||_F^2 + \frac{\lambda_C}{2}||C||_F^2 \end{aligned} \tag{4}$$

L2 loss function is basically minimizing the sum of the square of the differences between the target value and the estimated values.

4.1.1 Learning Algorithm and Predictions

One of the most often used optimization algorithms is block coordinate descent. Coordinate descent optimizes the objective function through a sequence of one-dimensional optimizations. Coordinate descent is based on the idea that the minimization of a multi-variable function

First U, V and C are initialized with random values. Then the minimization problem is solved for each one of the matrices individually. This is repeated until convergence. The coordinates descent algorithm for the objective with respect to L2 loss function in Equation 4 is given in Algorithm 1

Now, for a set of new words X^{test}, Equation 5 can predict their semantic categories.

$$Y^{\text{test}} \approx U^{\text{test}} C^T \tag{5}$$

However, U^{test} is unknown. The standard way to estimate U^{test} is through a fold-in:

$$U^{\text{test}} = \underset{\hat{U}}{\arg\min} ||X^{\text{test}} - \hat{U}V^T||_F^2 \tag{6}$$

$$U^{\text{test}} = X^{\text{test}} V (V^T V)^{-1} \tag{7}$$

4.1.2 MRMF on three Matrices

The MRMF method allows to integrate elegantly many different sources of information. In our experiment we have integrated the lexical and distributional information by extending the MRMF method described in section 4.1.

Matrices X and Y are the same matrices that we have seen in section 4.1. The newly added matrix Z has the lexical information which has the hypernym information from WordNet. X, Y and Z can be factorized as follows:

$$X \approx UV^T \tag{8}$$

$$Y \approx UC^T \tag{9}$$

$$Z \approx UB^T \tag{10}$$

The overall decomposition of the three matrices for MRMF method is visualized in Figure 2. Besides adding the Z matrix information in the objective function and in the coordinate decent algorithm, we have modified Equation 7 for U^{test} and add the third matrix information as follows:

$$U^{\text{test}} = X^{\text{test}} V (V^T V)^{-1} + Z^{\text{test}} B (B^T B)^{-1} \tag{11}$$

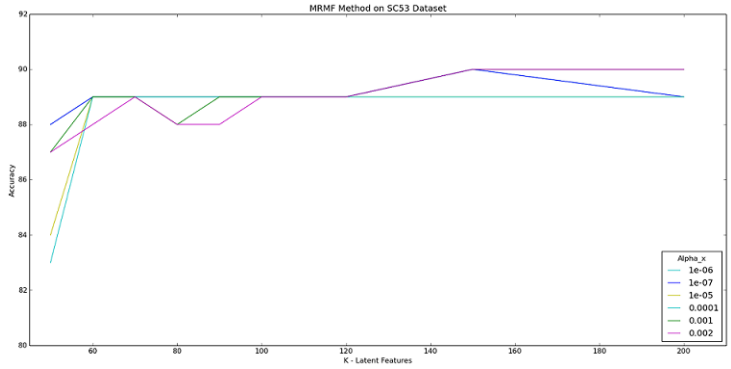


Figure 3: Accuracy of MRMF with different k and α_x parameter values on the SC53 dataset

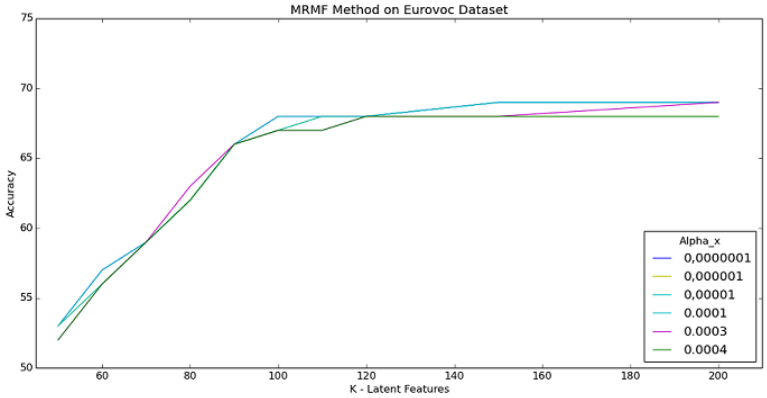


Figure 4: Accuracy of MRMF with different k and α_x parameter values on the Eurovoc dataset

4.2 Parameter Selection

For MRMF, a combination of weight constant, latent features and regularization parameters with a wide range of values was tested to find the best parameter setting. For SC53 dataset, the weight constant α_x and α_z range is in between $\frac{1}{\#\_SC53\_instances}$ and $1{\cdot}10^{-7}$, and in between $\frac{1}{\#\_Eurovoc\_instances}$ and $1{\cdot}10^{-7}$ for the Eurovoc dataset. The weight constant α_y is set to 1; Because Y is the matrix that we are building the model for, and Y^{test} is the matrix we want to predict. The regularization constants λ_u, λ_v and λ_b have used the same range of value which is in between $1 \cdot 10^{-17}$ and $1 \cdot 10^{-22}$. For the latent features k, we considered a range between 50 and 200.

Figure 3 and Figure 4 show the parameters α_x and k performance for the SC53 and Eurovoc datasets, respectively. As both figures show, the k parameter gives high accuracy on value 200 and goes flat after that for both datasets on MRMF_2 method, and also on method MRMF_3. In method MRMF_2, the k parameter has performed better with the α_x parameter value around 0.002 for SC53 and between 0.0003 and 0.0001 for the Eurovoc dataset. MRMF_3 gives optimal results when α_x is 0.001 and α_z is 0.0004 for SC53 dataset and when α_x is $1 \cdot 10^{-6}$ and α_z is $1 \cdot 10^{-7}$ for the Eurovoc derived data.

5 Evaluation

For evaluation we used 10 fold cross validation. For all experiments we used the same stratified split. This is basically the same as the leave-one out setup used by Bullinaria and Levy (2012). However, for equal size classes, in a leave–one–out experiment an intelligent classifier eventually might learn that the element to be classified always belongs to the smallest class. By using stratified croiss-validation we avoid this problem.

	Methods	Eurovoc	Stand. Err.	SC53	Stand. Err.
	Nearest Centroid (reported)	-	-	0,86	-
	Nearest Centroid (reproduced)	0,58	0,14	0,86	0,04
WN	LR	0,45	0,17	0,74	0,11
	SVM	0,50	0,16	0,73	0,10
	MRMF_2M	0,48	0,17	0,79	0,07
DF	LR	0,56	0,16	0,90	0,04
	SVM	0,69	0,10	0,90	0,03
	MRMF_2M	0,69	0,10	0,90	0,03
DF + WN	MRMF_3M	**0,71**	0,10	**0,93**	0,02
	2xLR + SVM	-	-	0,89	-
	2xSVM+ SVM	-	-	0,92	-

Table 2: Accuracy of classification on Eurovoc and SC53 datasets. Results are averages from 10-fold cross validation.

6 Result

Table 2, the result table, summarizes the performance of the methods on each dataset with their standard error (Stand. Err.). Bullinaria and Levy (2007; Bullinaria and Levy (2012) study different design and parameter choices for distributional similarity. The best accuracy, that they reached for the SC53 dataset (using a nearest centroid classifier), was $0,86$. We could reproduce this result using roughly the same choices and parameter settings that were given by Bullinaria and Levy. Applying the same method to the Eurovoc dataset gives an accuracy of 0,58. We used the hypernym features (WN) only in the supervised and hybrid settings.

For both datasets, we see that both SVM and MRMF are superior to the nearest centroid classifier. We see no big differences between the SVM and MRMF. As expected the results from logistic regression (LR) stay a bit behind those results.

Finally, we see that the integration of lexical and distributional information using MRMF clearly improves the result for both data sets. The ensemble methods can also improve the results, but stay behind the result of MRMF_3M. Since the logistic regression results for the Eurovoc data stay much behind the SVM and MRMF results, we did not test the ensemble based on those classifiers.

For the SC53 dataset, both the supervised classifiers using only distributional features and the classifier using a combination of distributional (DF) and lexical (WN) features outperform the best result reported up to now. Keith et al. (2015) report an accuracy of $0,96$ when reproducing the experiment of Bullinaria and Levy, but, as mentioned before, this result is not comparable to ours, since they used only a part of the data for evaluation.

If we look at the word classes predicted by the MRMF for the SC53 data, using both sources of information, we still have a small number of real errors. E.g. the word *mixer* is classified as a non-alcoholic beverage and *nun* as a relative. Most errors, however, are not real errors, like the word *foot* that is classified as a body part by MRMF and is a unit of distance in the dataset. A *knife* is classified as a weapon instead of a kitchen utensil; *shoes* as a type of footwear instead of clothing; and a *bass* as a musical instrument instead of a fish.

Given the type of errors that is made, we can conclude that to the SC53 data set we are close to the highest possible accuracy that can be reached. The Eurovoc dataset clearly is much harder and has still room for improvement.

7 Conclusion

We have studied semantic classification of words using distributional features directly in a strongly supervised learning setting. We have shown on two different data sets, that both SVM and MRMF outperform a distance based classifier, that is commonly used for this task. On a dataset which was used before for the same task, we thus could obtain results that are beyond state of the art.

In order to make a classification task that is closer to real applications, we compiled a new data set with more semantic categories. This data set is clearly much harder, but experiments on this dataset confirm all conclusions from the experiment on the smaller dataset.

In order to improve the results we finally investigated the possibility to include information from WordNet. While an ensemble classifier was not very successful in combining the two sources of information, MRMF was able to integrate the two types of information and improve the results substantially.

Since we are close to the optimal result for the SC53 dataset, we will concentrate on future work on datasets with a larger number of classes. In addition, we will try to find more sources of information that successfully can be integrated in order to improve the accuracy and to explore the possibilities of MRMF.

References

Rosa Tsegaye Aga, Christian Wartena, Lucas Drumond, and Lars Schmidt-Thieme. 2016. Learning thesaurus relations from distributional features. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

W.F. Battig and W.E. Montague. 1969. *Category norms for verbal items in 56 categories: a replication and extension of the Connecticut category norms*. Journal of experimental psychology monograph. American Psychological Association.

John A. Bullinaria and Joseph P. Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behaviour Research Methods*, 39(3):510–526.

John A. Bullinaria and Joseph P. Levy. 2012. Extracting Semantic Representations from Word Co-occurrence Statistics: Stop-lists, Stemming and SVD. *Behaviour Research Methods*, 44(3):890–907.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Nasari: a novel approach to a semantically-aware representation of items. In *NAACL*.

Lucas Rego Drumond, Ernesto Diaz-Aviles, Lars Schmidt-Thieme, and Wolfgang Nejdl. 2014. Optimizing multi-relational factorization models for multiple target relations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 191–200, New York, NY, USA. ACM.

J.-W. Fan and C. Friedman. 2007. Semantic Classification of Biomedical Concepts Using Distributional Similarity. *Journal of the American Medical Informatics Association*, 14(4):467–477.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: the concept revisited. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 406–414.

Eugenie Giesbrecht. 2010. Towards a Matrix-based Distributional Model of meaning. In *Proceedings of the NAACL HLT 2010 Student Research Workshop*, pages 23–28, Los Angeles, California. ACL.

Masato Hagiwara. 2008. A Supervised Learning Approach to Automatic Synonym Identification Based on Distributional Features for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA, Student Research Workshop. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 1–6.

Jeff Keith, Chris Westbury, and James Goldman. 2015. Performance impact of stop lists and morphological decomposition on word–word corpus-based semantic space models. *Behavior Research Methods*, 47(3):666–684.

Douwe Kiela and Stephen Clark. 2014. A Systematic Study of Semantic Vector Space Model Parameters. In *2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 21–30, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christoph Lippert, Stefan Hagen Weber, Yi Huang, Volker Tresp, Matthias Schubert, and Hans-Peter Kriegel. 2008. Relation-Prediction in Multi-Relational Domains using Matrix-Factorization. In *NIPS 2008 Workshop: Structured Input - Structured Output*.

Office for Official Publications of the European Communities. 1995. Thesaurus eurovoc - volume 2: Subject-oriented version.

Viktor Pekar, Michael Krkoska, and Steffen Staab. 2004. Feature weighting for co-occurrence-based classification of words. In *Proceedings of the 20th international conference on Computational Linguistics*, pages 799 – 806. Association for Computational Linguistics.

Marco Pennacchiotti, Diego De Cao, Roberto Basili, Danilo Croce, and Michael Roth. 2008. Automatic induction of framenet lexical units. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 457–465, Stroudsburg, PA, USA. Association for Computational Linguistics.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual Correlates of Synonymy. *Commun. ACM*, 8(10):627–633.

Mohammad Saif and Graeme Hirst. 2012. Distributional Measures of Semantic Distance: A Survey.

Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. 2013. A tensor-based factorization model of semantic compositionality. In *Proceedings of the Conference of the North American Chapter of the Association of Computational Linguistics (HTL-NAACL)*, pages 1142–1151.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2249–2259, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Wen-tau Yih and Vahed Qazvinian. 2012. Measuring word relatedness using heterogeneous vector space models. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 616–620, Stroudsburg, PA, USA. Association for Computational Linguistics.

Semi Supervised Preposition-Sense Disambiguation using Multilingual Data

Hila Gonen
Department of Computer Science
Bar-Ilan University
hilagnn@gmail.com

Yoav Goldberg
Department of Computer Science
Bar-Ilan University
yoav.goldberg@gmail.com

Abstract

Prepositions are very common and very ambiguous, and understanding their sense is critical for understanding the meaning of the sentence. Supervised corpora for the preposition-sense disambiguation task are small, suggesting a semi-supervised approach to the task. We show that signals from unannotated multilingual data can be used to improve supervised preposition-sense disambiguation. Our approach pre-trains an LSTM encoder for predicting the translation of a preposition, and then incorporates the pre-trained encoder as a component in a supervised classification system, and fine-tunes it for the task. The multilingual signals consistently improve results on two preposition-sense datasets.

1 Introduction

Preposition-sense disambiguation (Litkowski and Hargraves, 2005; Litkowski and Hargraves, 2007; Schneider et al., 2015; Schneider et al., 2016), is the task of assigning a category to a preposition in context (see Section 2.1). Choosing the correct sense of a preposition is crucial for understanding the meaning of the text. This important semantic task is especially challenging from a learning perspective as only little amounts of annotated training data are available for it. Indeed, previous systems (see Sections 2.1.1 and 5.4) make extensive use of the vast and human-curated WordNet lexicon (Miller, 1995) in order to compensate for the small size of the annotated data and obtain good accuracies.

Instead, we propose to deal with the scarcity of annotated data by taking a semi-supervised approach. We rely on the intuition that word ambiguity tends to differ between languages (Dagan et al., 1991), and show that multilingual corpora can provide a good signal for the preposition sense disambiguation task. Multilingual corpora are vast and relatively easy to obtain (Resnik and Smith, 2003; Koehn, 2005; Steinberger et al., 2006), making them appealing candidates for use in a semi-supervised setting.

Our approach (Section 4) is based on representation learning (Bengio et al., 2013), and can also be seen as an instance of multi-task (Caruana, 1997), or transfer learning (Pan and Yang, 2010). First, we train an LSTM-based neural network (Hochreiter and Schmidhuber, 1997) to predict a foreign (say, French) preposition given the context of an English preposition. This trains the network to map contexts of English prepositions to representations that are predictive of corresponding foreign prepositions, which are in turn correlated with preposition senses. The learned mapper, which takes into account large amounts of parallel text, is then incorporated into a monolingual preposition-sense disambiguation system (Section 3) and is fine-tuned based on the small amounts of available supervised data. We show that the multilingual signal is effective for the preposition-sense disambiguation task on two different datasets (Section 5).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,
pages 2718–2729, Osaka, Japan, December 11-17 2016.

2 Background

2.1 Preposition Sense Disambiguation

Prepositions are very common, very ambiguous and tend to carry different meanings in different contexts. Consider the following 3 sentences: "You should book a room *for* 2 nights", "*For* some reason, he is not here yet" and "I went there to get a present *for* my mother". The preposition "for" has 3 different readings in these sentences: in the first sentence it indicates DURATION, in the second it indicates an EXPLANATION, and in the third a BENEFICIARY. The preposition-sense disambiguation task is defined as follows: given a preposition within a sentential context, decide which category it belongs to, or what its role in the sentence is. Choosing the right sense of a preposition is central to understanding the meaning of an utterance (Baldwin et al., 2009).

2.1.1 Previous Work and Available Corpora

The preposition-sense disambiguation task was the focus of the SemEval 2007 shared task (Litkowski and Hargraves, 2007), based on the set of senses defined in The Preposition Project (TPP) (Litkowski and Hargraves, 2005), with three participating systems (Ye and Baldwin, 2007; Yuret, 2007; Popescu et al., 2007). Since then, it was tackled in several additional works (Dahlmeier et al., 2009; Tratz and Hovy, 2009; Hovy et al., 2010; Tratz, 2011; Srikumar and Roth, 2013b), some of which used different preposition sense inventories and corpora, based on subsets of the TPP dictionary. Srikumar and Roth (2013b) modeled semantic relations expressed by prepositions. For this task, they presented a variation of the TPP inventory, by collapsing related preposition senses, so that all senses are shared between all prepositions (Srikumar and Roth, 2013a). Schneider et al (2015) further improve this inventory and define a new annotation scheme.

There are two main datasets for this task: the corpus of the SemEval 2007 shared task (Litkowski and Hargraves, 2007), and the Web-reviews corpus (Schneider et al., 2016):

SemEval 2007 Corpus This corpus covers 34 prepositions with 16,557 training and 8096 test sentences, each containing a single preposition example. The sentences were extracted from the FrameNet database,[1] based mostly on the British National Corpus (with 75%/25% of informative-writings/literary). Each preposition has a different set of possible senses, with a range of 2 to 25 possible senses for a given preposition. We use the original split to train and test sets.

Web-reviews Corpus Schneider et al (2015) introduce a new, unified and improved sense inventory and corpus (Schneider et al., 2016) in which all prepositions share the same set of senses (senses from a unified inventory are often referred to as supersenses). This corpus contains text in the online reviews genre. It is much smaller than the SemEval corpus, with 4,250 preposition mentions covering 114 different prepositions which are annotated into 63 fine-grained senses. The senses are grouped in a hierarchy, from which we chose a coarse-grained subset of 12 senses for this work: AFFECTOR, ATTRIBUTE, CIRCUMSTANCE, CO-PARTICIPANT, CONFIGURATION, EXPERIENCER, EXPLANATION, MANNER, PLACE, STIMULUS, TEMPORAL, UNDERGOER. We find the Web-reviews corpus more appealing than the SemEval one: the unified sense inventory makes the sense-predictions more suitable for use in downstream applications. While our focus in this work is the Web-reviews corpus, we are the first to report results on this dataset. For the sake of comparison to previous work, we also evaluate our models on the SemEval corpus.

2.2 Neural Networks and Notation

We use $w_{1:n}$ to indicate a list of vectors, and $w_{n:1}$ to indicate the reversed list. We use $\circ$ for vector concatenation, and $x[j]$ for selecting the j^{th} element in a vector x.

A multi-layer perceptron (MLP) is a non linear classifier. In this work, we focus on MLPs with a single hidden layer and a softmax output transformation, and define the function $MLP(x)$ as:

$$MLP(x) = \text{softmax}(U(\text{g}(Wx + b1)) + b2)$$

[1] http://framenet.icsi.berkeley.edu/

where g is a non-linear activation function such as $ReLU$ or $tanh$, W and U are input-to-hidden and hidden-to-output transformation matrices, and $b1$ and $b2$ are optional bias terms. We use subscripts (MLP_{f1}, MLP_{f2}) to denote MLPs with different parameters.

Recurrent Neural Networks (RNNs) (Elman, 1990) allow the representation of arbitrary sized sequences, without limiting the length of the history. RNN models have been proven to effectively model sequence-related phenomena such as line lengths, brackets and quotes (Karpathy et al., 2015).

In our implementation we use the long short-term memory network (LSTM), a subtype of the RNN (Hochreiter and Schmidhuber, 1997). $LSTM(w_{1:i})$ is the output vector resulting from inputing the items $w_1, ..., w_i$ into the LSTM in order.

3 Monolingual Preposition Sense Classification

We start by describing an MLP-based model for classifying prepositions to their senses. For an English sentence $s = w_1, ..., w_n$ and a preposition position i,[2] we classify to the sense y as:

$$y = \underset{j}{\text{argmax}}\, MLP_{sense}(\phi(s, i))[j]$$

where $\phi(s, i)$ is a feature vector composed of 19 features. The features are based on the features of Tratz and Hovy (2009), and are similar in spirit to those used in previous attempts at preposition sense disambiguation. We deliberately do not include WordNet based features, as we want to focus on features that do not require extensive human-curated resources. This makes our model applicable for use in other languages with minimal change. We use the following features: (1) The embedding of the preposition. (2) The embeddings of the lemmas of the two words before and after the preposition, of the head of the preposition in the dependency tree, and of the first modifier of the preposition. (3) The embeddings of the POS tags of these words, of the preposition, and of the head's head. (4) The embeddings of the labels of the edges to the head of the preposition, to the head's head and to the first modifier of the preposition. (5) A boolean that indicates whether one of the two words that follow the preposition is capitalized. The English sentences were parsed using the spaCy parser.[3]

The network (including the embedding vectors) is trained using cross entropy loss. This model performs relatively well, achieving an accuracy of 73.34 on the Web-reviews corpus, way above the most-frequent-sense baseline of 62.37. On the SemEval corpus, it achieves an accuracy of 74.8, outperforming all participants in the original shared task (Section 5). However, these results are limited by the small size of both training sets. In what follows, we will improve the model using unannotated data.

4 Semi-Supervised Learning Using Multilingual Data

Our goal is to derive a representation from unannotated data that is predictive of preposition-senses. We suggest using multilingual data, following the intuition that preposition ambiguity usually differs between languages (Dagan et al., 1991). For example, consider the following two sentences, taken from the Europarl parallel corpus (Koehn, 2005): "What action will it take to defuse the crisis and tension *in* the region?", and "These are only available *in* English, which is totally unacceptable". In the first sentence, the preposition "in" is translated into the French preposition "dans", whereas in the second one, it is translated into the French preposition "en". Thus, a representation that is predictive of the preposition's translation is likely to be predictive also of its sense.

Learning a representation from a multilingual corpus We train a neural network model to encode the context of an English preposition as a vector, and predict the foreign preposition based on the context vector. The resulting context encodings will then be predictive of the foreign prepositions, and hopefully also of the preposition senses.

We derive a training set of roughly 7.4M instances from the Europarl corpus (Koehn, 2005). Europoarl contains sentence-aligned data in 21 languages. We started by using several ones, and ended up with a

[2] We also support multi-word prepositions in this work. The extension is trivial.

[3] `https://spacy.io/`

subset of 12 languages[4] that together constitute a good representation of the different language families available in the corpus. Though adding the other languages is possible, we did not experiment with them. To extract the training set, we first word-align[5] the sentence-aligned data, and then create a dataset of English sentences where each preposition is matched to its translation in a foreign language. Since the alignment of prepositions is noisier than that of content words, we use a heuristic to improve precision: given a candidate foreign-preposition, we verify that the two words surrounding it are aligned to the two words surrounding the English preposition. Additionally, we filter out, for each English preposition, all foreign prepositions that were aligned to it in less than 5% of the cases.

We then train the context representations according to the following model. For an English sentence $s = w_1, ..., w_n$, a preposition position i and a target preposition p in language L, we encode the context as a concatenation of two LSTMs, one reading the sentence from the beginning up to but not including the preposition, and the other in reverse:

$$ctx(s, i) = LSTM_f(w_{1:i-1}) \circ LSTM_b(w_{n:i+1})$$

This is similar to a BiLSTM encoder, with the difference that the encoding does not include the preposition w_i but only its context. By ignoring the preposition, we force the model to focus on the context, and help it share information between different prepositions. Indeed, including the preposition in the encoder resulted in better performance in foreign preposition classification, but the resulting representation was not as effective when used for the sense disambiguation task.

The context vector is then fed into a language specific MLP for predicting the target preposition:

$$\hat{p} = \underset{j}{\text{argmax}}\, MLP_L(ctx(s, i))[j]$$

The context-encoder and the word embeddings are shared across languages, but the MLP classifiers that follow are language specific. By using multiple languages, we learn more robust representations.

The English word embeddings can be initialized randomly, or using pre-trained embedding vectors, as we explore in Section 5.1. The network is trained using cross entropy loss, and the error is back-propagated through the context-encoder and the word embeddings.

Using the representation for sense classification Once the encoder is trained over the multilingual data, we incorporate it in the supervised sense-disambiguation model by concatenating the representation obtained from the context encoder to the feature vector. Concretely, the supervised model now becomes:

$$y = \underset{j}{\text{argmax}}\, MLP_{sense}(ctx(s, i) \circ \phi(s, i))[j]$$

where $ctx(s, i)$ is the output vector of the context-encoder and $\phi(s, i)$ is the feature vector as before.

The network is trained using cross entropy loss, and the error back-propagates also to the context-encoder and to the word embeddings to maximize the model's ability to adapt to the preposition-sense disambiguation task. The complete model is depicted in Figure 1.

5 Empirical results

Implementation details The models were implemented using PyCNN.[6] All models were trained using SGD, shuffling the examples before each of the 5 epochs. When training a sense prediction model, we use early stopping and choose the best performing model on the development set. The sense-prediction MLP uses $ReLU$ activation, and foreign preposition MLPs use $tanh$, with no bias terms. Unless noted otherwise, we use randomly initialized embedding vectors. For each experiment, we chose the parameters that maximized the accuracy on the dev set.[7] The accuracies we report are the average accuracies over 5 different seeds.

[4] Bulgarian, Czech, Danish, German, Greek, Spanish, French, Hungarian, Italian, Polish, Romanian and Swedish.

[5] Word-alignment is done using the `cdec` aligner (Dyer et al., 2010).

[6] `https://github.com/clab/cnn`

[7] In most of the experiments, the best results are achieved when the hidden-layer of the sense-prediction MLPs is of the size 500, and the preposition embedding is of size 200. In some cases, the best results are achieved with different dimensions.

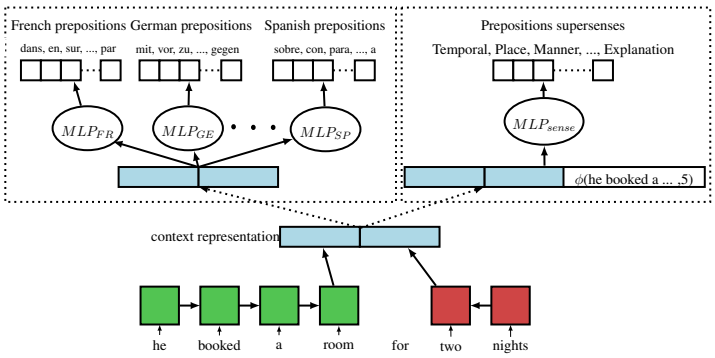


Figure 1: The suggested model for incorporating multilingual data in classifying prepositions to senses. First, a context-encoder (at the bottom, the green and red squares are LSTM cells) is trained on the Europarl corpus, with a different MLP for each language (left dashed frame). Then, the representation obtained from the context-encoder is added to the feature vector when classifying a preposition to senses (right dashed frame).

5.1 Evaluation on the Web-reviews corpus

Using multilingual data Our main motivation in this work was to train a representation which is useful for the preposition-sense disambiguation task. Thus, we compare the performance of our model using the representation obtained from the context-encoder (multilingual model) with the model that does not use this representation (base model). We use the train/test split provided with the corpus. We further split the train set into train and dev sets, by assigning every fourth example of each sense to the dev set, yielding 2552/845/853 instances of train/dev/test.

The results are presented in Table 1. We see an improvement of 2.86 points when using the pre-trained context representations, improving the average result from 73.34 to 76.20.

To verify that the improvement stems from pre-training the context-encoder on multilingual data and not from adding the context-encoder as is, we also evaluated the performance of a model identical to the multilingual model, but with no pre-training on the multilingual data (context model, middle row of Table 1). The context model achieved a very similar result to that of the base model – 73.76, indicating that adding the context-encoder to the base model is not the source of the improvement.

Model	Accuracy
base	73.34 (71.63-73.97)
+context	73.76 (71.86-75.38)
+context(multilingual)	**76.20** (74.91-77.26)

Table 1: The average accuracies on the test set of the Web-reviews corpus on 5 different seeds. Numbers in brackets indicate the min and max accuracy across seeds.

Using monolingual or bilingual data only In order to verify the contribution of incorporating information from 12 languages, we also experiment with monolingual and bilingual models. For the monolingual model we train a model similar to our multilingual one, but when trying to predict the English preposition itself, rather than the foreign one, ignoring the multilingual signal altogether. For the bilingual models we train 12 separate models similar to our multilingual model, where each one is trained only on the training examples of a single language.

As shown in Table 2, both the monolingual and the bilingual models improve over the base model (with the exception of Czech), but no improvement is as significant as that of the multilingual model. In addition, we see that the strength of the model does not depend solely on the number of training examples.

Adding external word embeddings Another way of incorporating semi-supervised data into a model is using pre-trained word embeddings. We evaluate our model when using external word embeddings

These two parameters were tuned on the dev set. The embeddings of the features are of dimension 4, with the exception of the lemmas, which are of dimension 50. The dimension of the input to the LSTMs (word embeddings) is 128. Both LSTMs have a single layer with 100 nodes, thus, the representation of the context obtained from the context-encoder is of dimension 200. The hidden-layer of the foreign-preposition MLP is of size 32.

Language	Accuracy	Improvement	Num. of training examples
None (base model)	73.34 (71.63-73.97)	–	–
Czech	73.06 (72.57-73.86)	-0.28	190,850
Polish	73.93 (73.15-74.79)	+0.59	166,101
Italian	73.97 (72.22-75.26)	+0.63	810,589
Romanian	74.09 (73.15-74.56)	+0.75	205,520
Hungarian	74.42 (73.27-75.15)	+1.08	40,302
Bulgarian	74.44 (73.27-74.91)	+1.10	292,908
Spanish	74.65 (73.51-75.73)	+1.31	1,267,400
German	74.73 (73.74-75.62)	+1.39	603,861
Danish	75.08 (74.21-77.49)	+1.74	1,131,915
Greek	75.12 (74.09-76.20)	+1.78	586,494
French	75.43 (74.21-77.02)	+2.09	1,033,267
English (monolingual)	75.68 (74.79-76.55)	+2.34	7,483,206
Swedish	75.87 (74.68-77.49)	+2.53	1,153,999
All 12 languages	**76.20** (74.91-77.26)	**+2.86**	7,483,206

Table 2: The average accuracies on the test set of the Web-reviews corpus on 5 different seeds, using monolingual and bilingual models, along with the improvement over the base model and the number of training examples in each language. Numbers in brackets indicate the min and max accuracy across seeds.

instead of randomly initialized word embeddings. We perform three experiments: 1. using external word embeddings only for the words that are fed into the context-encoder. 2. using external word embeddings only for the lemmas of the features. 3. using external word embeddings for both.

We use two sets of word embeddings: 5-window-bag-of-words-based and dependency-based, both trained by Levy and Goldberg (2014) on English Wikipedia.[8] As shown in Table 3, both pre-trained embeddings improve the performance of all models in most cases. In all cases, the multilingual model outperforms the base model and the context model, both achieving similar results. Using external word embeddings for both the features and the context-encoder helps the most. The best result of 78.55 is achieved by the multilingual model, improving the result of the base model under the same conditions by 1.71 points.

Model	Context-encoder embeddings only		Feature embeddings only		Embeddings for both	
	Bow	Deps	Bow	Deps	Bow	Deps
base	73.34 (71.63-73.97)	73.34 (71.63-73.97)	76.95 (75.85-77.96)	76.84 (76.32-77.26)	76.95 (75.85-77.96)	76.84 (76.32-77.26)
+context	74.07 (72.10-75.15)	74.42 (73.62-75.03)	76.72 (75.85-77.96)	77.47 (75.85-78.55)	77.14 (76.79-78.08)	77.73 (77.14-78.43)
+context(multilingual)	75.57 (73.51-77.84)	75.90 (75.03-76.55)	77.58 (77.02-78.08)	77.58 (77.14-78.66)	78.45 (77.49-79.48)	**78.55** (77.37-79.37)

Table 3: The average accuracies on the test set of the Web-reviews corpus with different pre-trained embeddings on 5 different seeds. Numbers in brackets indicate the min and max accuracy across seeds. **Bow**: 5-words window; **Deps**: dependency-based.

5.2 Evaluation on the SemEval corpus

Adaptations to the SemEval corpus In the SemEval corpus each preposition has a different set of senses, and the natural approach is to learn a different model for each one. We call this the *disjoint* approach. However, we found this approach a bit wasteful in terms of exploiting the annotated data, and we propose a model that uses the information from all prepositions simultaneously (*unified*). In the unified approach, we create an MLP classifier for each preposition, but all of them share a single input-to-hidden transformation matrix and a single bias term. Formally, for a preposition p, we define its MLP as follows:

$$MLP_p(x) = \text{softmax}(U_p(\text{g}(Wx + b1)) + b2_p)$$

where W is the shared input-to-hidden transformation matrix, $b1$ is the shared bias term, and U_p and $b2_p$ are preposition-specific hidden-to-output transformation matrix and bias term, respectively. This unified model is trained over the training examples of all prepositions together.

The SemEval corpus sometimes provides multiple senses for a given preposition instance. In both the disjoint and the unified approaches we treat these cases by generalizing the cross entropy loss for multiple correct classes. In the common case, where each training example has a single correct class, the

[8] https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/

cross entropy loss is defined as $-\log p_i$, where p_i is the probability that the model assigns to the correct class. Here, instead of using $-\log p_i$, we use $-\log(\sum_{i \in C} p_i)$, where C is the set of correct classes.

Results The model performs well also on the SemEval corpus, achieving an accuracy of 76.9. Note that we use the exact same parameters that were tuned on the dev set of the Web-reviews corpus, with no additional tuning on this corpus.

As shown in Table 4, the unified model, which trains on all prepositions simultaneously, performs better than a separate model for each preposition (disjoint model), and achieves an improvement of 1.3 points when using the multilingual model. In addition, in both cases we get a significant improvement over the base model when using the pre-trained context-representation. In the unified model, adding the pre-trained context-representation improves the result by 2.1 points. As in the case of the Web-reviews corpus, we can see that this improvement does not stem from adding the context representation as is. Pre-training the representation is essential for achieving these improved results.

Model	Disjoint	Unified
base	73.7 (73.3-74.1)	74.8 (74.4-75.4)
+context	73.8 (73.6-74.0)	75.4 (74.8-75.8)
+context(multilingual)	75.6 (75.4-75.8)	**76.9** (76.4-77.7)

Table 4: The average accuracies on the test set of the SemEval corpus on 5 different seeds, with both the disjoint and the unified models. Numbers in brackets indicate the min and max accuracy across seeds.

Similar to the results on the Web-reviews Corpus, when using external word embeddings both for the words that are fed into the context-encoder and for the features, we get an improvement in all models, with an average improvement of 3 points when using the 5-words-window based embeddings. The best result amongst the three models is of 79.6 and is achieved by the multilingual model, improving over the base model by 2.5 points. The results are shown in Table 5.

Note that unlike previous experiments, adding external word embeddings improves the context model over the base model significantly, approaching the results of the multilingual model. For this reason, we also evaluated a model in which we concatenate both contexts: that of the context model (no pre-training), and that of the multilingual model (pre-trained on the multilingual data). In the case where both models achieve similar results, combining both contexts further improves the result, which indicates that they are complementary. The best result of 80.0 is achieved when using both contexts with the 5-window-bag-of-words-based embeddings. We also evaluated this combined model on the Web-reviews corpus, but got no improvement in most cases. This was predictable since in all experiments on that corpus we had a large difference between the results of the context model and of the multilingual model. The only case where we saw an improvement with both contexts was when using dependency-based embeddings for both the features and the context-encoder. The difference between the two datasets can be explained by the much larger size of the SemEval dataset, which allows the context encoder to learn from more data, even without pre-training on multilingual data.

Model	Bow	Deps	None
base	77.1 (76.9-77.2)	76.6 (76.3-76.9)	74.8 (74.4-75.4)
+context	79.5 (78.8-79.9)	78.5 (78.0-78.8)	75.4 (74.8-75.8)
+context(multilingual)	79.6 (79.3-79.9)	79.3 (78.8-79.6)	76.9 (76.4-77.7)
+both contexts	**80.0** (79.8-80.2)	79.2 (78.6-79.5)	77.3 (77.2-77.5)

Table 5: The average accuracies on the test set of the SemEval corpus on 5 different seeds, with the unified model, when using external word embeddings for both the context-encoder and the features. Numbers in brackets indicate the min and max accuracy across seeds. **Bow**: 5-words window; **Deps**: dependency-based; **None**: no external word embeddings.

5.3 Using Ensembles

We create an ensemble by training 5 different models (each with a different random seed), and predict test instances using a majority vote over the models. The results are presented in Table 6. As expected, results in all models further improve when using the ensemble. Using the multilingual context helps also when using the ensemble. We see an improvements of 1.99 points on the web-reviews corpus, improving the

result to 80.54. The performance on the SemEval corpus improves by 1.7 points, and reaches an accuracy of 81.7. These results are higher than those of the base model by 2.93 and 2.2 points, respectively.

Model	Web-reviews Corpus		SemEval Corpus	
	Average	Ensemble	Average	Ensemble
base	76.84 (76.32-77.26)	77.61	77.1 (76.9-77.2)	79.5
+context	77.73 (77.14-78.43)	78.90	79.5 (78.8-79.9)	81.1
+context(multilingual)	78.55 (77.37-79.37)	**80.54**	79.6 (79.3-79.9)	81.2
+both contexts	79.34 (78.43-80.19)	79.84	80.0 (79.8-80.2)	**81.7**

Table 6: The results on both datasets on 5 different seeds as reported in Tables 3 and 5 in comparison to the results using the ensemble. Numbers in brackets indicate the min and max accuracy across seeds.

5.4 Comparison to previous systems

Table 7 compares our SemEval results with those of previous systems. The system of Ye and Baldwin (2007) got the highest result out of the three participating systems in the SemEval 2007 shared task. They extracted features such as POS tags and WordNet-based features, and also high level features (e.g semantic role tags), using a word window of up to seven words, in a Maximum Entropy classifier. Tratz and Hovy (2009) got a higher result with similar features by using a set of positions that are syntactically related to the preposition instead of a fixed window size. The best performing systems are of Hovy et al (2010) and of Srikumar and Roth (2013b). Both systems rely on vast and thoroughly-engineered feature sets, including many WordNet based features. Hovy et al (2010) explored different word choices (i.e, a fixed window vs. syntactically related words) and different methods of extracting them, while Srikumar and Roth (2013b) improved performance by jointly predicting preposition senses and relations.

In contrast, our models do not include any WordNet based features, making them applicable also for languages lacking such resources. Our models achieve competitive results, outperforming most previous systems, despite using relatively few features and performing hyper-parameter tuning only on the different domain Web-reviews corpus.

Model	Accuracy
base	74.8
+context	75.4
+context(multilingual)	76.9
+context(multilingual) + embeddings	79.6
+both contexts + embeddings	80.0
+both contexts + embeddings + ensemble	81.7
Hovy et al (2010) – using WordNet features	84.8
Srikumar and Roth (2013b) – using WordNet features	84.78
Tratz and Hovy (2009) – using WordNet features	76.4
MELB-YB (Ye and Baldwin, 2007) – using WordNet features	69.3
KU (Yuret, 2007)	54.7
IRST-BP (Popescu et al., 2007)	49.6
Most Frequent Sense	39.6

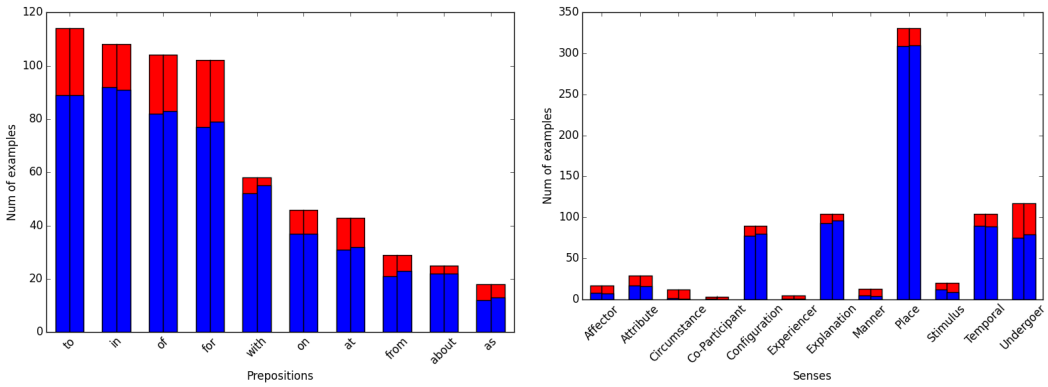
Table 7: The accuracies on the test set of the SemEval corpus, in comparison to previous systems.

5.5 Error Analysis

Figure 2 depicts the percentage of correct assignments of the base model, in comparison to the multilingual model, per sense and per preposition (only the 10 most common prepositions are shown). Both models use pre-trained word embeddings and ensembles. Clearly, there is a systematic improvement across most prepositions and senses.

6 Related work

Transfer learning and representation learning Transfer learning is a methodology that aims to reduce annotation efforts by first learning a model on a different domain or a closely related task, and then transfer the gained knowledge to the main task (Pan and Yang, 2010). Multi-task learning (MTL) is an approach of transfer learning in which several tasks are trained in parallel while using a shared representation. The different tasks can benefit from each other through this representation (Caruana, 1997). In



(a) prepositions (b) senses

Figure 2: Assignments on the dev set of the Web-reviews corpus per preposition (a) and per sense (b). Left bars stand for the base model, right bars stand for the multilingual model. In blue are correct assignments, and in red incorrect ones.

this work we use MTL to improve preposition-sense disambiguation, by using an auxiliary multilingual task – predicting translations of prepositions.

A simple method for sharing information in transfer learning as well as in MTL, is using representations that are shared between related tasks. Representation learning (Bengio et al., 2013) is a closely related field that aims to establish techniques for learning robust and expressive data representations. A well-known effort in this field is that of learning word embeddings for use in a wide range of NLP tasks (Mikolov et al., 2013; Al-Rfou et al., 2013; Levy and Goldberg, 2014; Pennington et al., 2014). While those representations are highly effective in many cases, other scenarios require representations of a full sentence, or of a context around a target word, rather than representations of single words. Contexts are often represented by some manipulation over the embeddings of their words. Such representations have been successfully used for tasks such as context-sensitive similarity (Huang et al., 2012), word sense disambiguation (Chen et al., 2014) and lexical substitution (Melamud et al., 2015). An alternative approach for context representation is encoding a context of arbitrary length into a single vector using LSTMs. This approach has been proven to outperform the previous attempts in a variety of tasks such as Semantic Role Labeling (Zhou and Xu, 2015), Natural Language Inference (Bowman et al., 2015) and Sentence Completion (Melamud et al., 2016). We follow the LSTM-based approach for context representation.

Learning from multilingual data The use of multilingual data for improving monolingual tasks has a long tradition in NLP, and has been used for target word selection (Dagan et al., 1991); word sense disambiguation (Diab and Resnik, 2002); and syntactic parsing and named entity recognition (Burkett et al., 2010), to name a few examples. A dominant approach for exploiting multilingual data is that of cross-lingual projection. This approach assumes a good model exists in one language, and uses annotations in that language in order to constrain possible annotations in another. Projections were successfully used for dependency grammar induction (Ganchev et al., 2009), and for transferring tools such as morphological analyzers and part-of-speech taggers from English to languages with fewer resources (Yarowsky et al., 2001; Yarowsky and Ngai, 2001). A different approach is applying multilingual constraints on existing monolingual models, as done for parsing (Smith and Smith, 2004; Burkett and Klein, 2008) and for morphological segmentation (Snyder and Barzilay, 2008).

Of much relevance to this work are also previous attempts to improve monolingual representations using bilingual data (Faruqui and Dyer, 2014). Previous works focus on creating sense-specific word embeddings instead of the common word-form specific embeddings (Ettinger et al., 2016; Šuster et al., 2016), and also on representing words using their context (Kawakami and Dyer, 2015; Hermann and Blunsom, 2013). While we rely on the assumption most of these works have in common, according to which translations may serve as a strong signal for different senses of words, the novelty of our work is in focusing on prepositions rather than content words, and in jointly representing a context for both a multilingual and a monolingual tasks, which results in an improvement of the monolingual model.

7 Conclusions and Future Work

We show that multilingual data can be used to improve the accuracy of preposition-sense disambiguation. The key idea is to train a context-encoder on vast amounts of parallel data, and by that, to obtain a context representation that is predictive of the sense. We show an improvement of the accuracy in all experiments upon using this representation. Our model achieves an accuracy of 80.54 on the Web-reviews corpus, and an accuracy of 81.7 on the SemEval corpus, with significant improvements over models that do not use the multilingual signals. Our result on the SemEval corpus outperforms most previous works, without using any manually curated lexicons.

Acknowledgements

The work is supported by The Israeli Science Foundation (grant number 1555/15).

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of CoNLL 2013*.

Timothy Baldwin, Valia Kordoni, and Aline Villavicencio. 2009. Prepositions in applications: A survey and introduction to the special issue. *Computational Linguistics*, 35(2):119–149.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proceedings of EMNLP*, pages 877–886.

David Burkett, Slav Petrov, John Blitzer, and Dan Klein. 2010. Learning better monolingual models with unannotated bilingual text. In *Proceedings of CoNLL*, pages 46–54.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28:41–75.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of EMNLP*, pages 1025–1035.

Ido Dagan, Alon Itai, and Ulrike Schwall. 1991. Two languages are more informative than one. In *Proceedings of ACL*, pages 130–137.

Daniel Dahlmeier, Hwee Tou Ng, and Tanja Schultz. 2009. Joint learning of preposition senses and semantic roles of prepositional phrases. In *Proceedings of EMNLP*, pages 450–458.

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of ACL*, pages 255–262.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of ACL*.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Allyson Ettinger, Philip Resnik, and Marine Carpuat. 2016. Retrofitting sense-specific word vectors using parallel text. In *Proceedings of NAACL-HLT*, pages 1378–1383.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*.

Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of ACL-IJCNLP*, pages 369–377.

Karl Moritz Hermann and Phil Blunsom. 2013. Multilingual distributed representations without word alignment. In *Proceedings of ICLR*.

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

Dirk Hovy, Stephen Tratz, and Eduard Hovy. 2010. What's in a preposition? dimensions of sense disambiguation for an interesting word class. In *Proceedings of COLING*.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*, pages 873–882.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *arXiv:1506.02078*.

Kazuya Kawakami and Chris Dyer. 2015. Learning to represent words in context with multilingual supervision. *arXiv:1511.04623*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT summit*, volume 5, pages 79–86.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL*.

Ken Litkowski and Orin Hargraves. 2005. The preposition project. In *Proceedings of the Second ACL-SIGSEM Workshop on the Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*, pages 171–179.

Ken Litkowski and Orin Hargraves. 2007. Semeval-2007 task 06: Word-sense disambiguation of prepositions. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 24–29.

Oren Melamud, Omer Levy, and Ido Dagan. 2015. A simple word embedding model for lexical substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 1–7.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of CoNLL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*.

George A Miller. 1995. Wordnet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP 2014*.

Octavian Popescu, Sara Tonelli, and Emanuele Pianta. 2007. IRST-BP: Preposition disambiguation based on chain clarifying relationships contexts. In *Proceedings of the 4th International Workshop on Semantic Evaluations*.

Philip Resnik and Noah A Smith. 2003. The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380.

Nathan Schneider, Vivek Srikumar, Jena D. Hwang, and Martha Palmer. 2015. A hierarchy with, of, and for preposition supersenses. In *Proceedings of the 9th Linguistic Annotation Workshop*, pages 112–123.

Nathan Schneider, Jena D. Hwang, Vivek Srikumar, Meredith Green, Abhijit Suresh, Kathryn Conger, Tim O'Gorman, and Martha Palmer. 2016. A corpus of preposition supersenses. In *Proceedings of the 10th Linguistic Annotation Workshop*.

David A Smith and Noah A Smith. 2004. Bilingual parsing with factored estimation: Using english to parse korean. In *Proceedings of EMNLP*.

Benjamin Snyder and Regina Barzilay. 2008. Cross-lingual propagation for morphological analysis. In *Proceedings of AAAI*.

Vivek Srikumar and Dan Roth. 2013a. An inventory of preposition relations. *arXiv:1305.5785*.

Vivek Srikumar and Dan Roth. 2013b. Modeling semantic relations expressed by prepositions. *Transactions of the Association for Computational Linguistics*, 1:231–242.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Dániel Varga. 2006. The jrc-acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proceedings of LREC*.

Simon Šuster, Ivan Titov, and Gertjan van Noord. 2016. Bilingual learning of multi-sense embeddings with discrete autoencoders. *arXiv:1603.09128*.

Stephen Tratz and Dirk Hovy. 2009. Disambiguation of preposition sense using linguistically motivated features. In *Proceedings of NAACL-HLT Student Research Workshop and Doctoral Consortium*, pages 96–100.

Stephen Tratz. 2011. *Semantically-enriched parsing for natural language understanding*. Ph.D. thesis, University of Southern California.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proceedings of NAACL*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT*.

Patrick Ye and Timothy Baldwin. 2007. MELB-YB: Preposition sense disambiguation using rich semantic features. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 241–244.

Deniz Yuret. 2007. KU: Word sense disambiguation by substitution. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 207–213.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of ACL*.

Monday mornings are my fave :) #not
Exploring the Automatic Recognition of Irony in English tweets

Cynthia Van Hee, Els Lefever and Véronique Hoste
LT^3, Language and Translation Technology Team
Department of Translation, Interpreting and Communication – Ghent University
Groot-Brittanniëlaan 45, 9000 Ghent, Belgium
Firstname.Lastname@UGent.be

Abstract

Recognising and understanding irony is crucial for the improvement natural language processing tasks including sentiment analysis. In this study, we describe the construction of an English Twitter corpus and its annotation for irony based on a newly developed fine-grained annotation scheme. We also explore the feasibility of automatic irony recognition by exploiting a varied set of features including lexical, syntactic, sentiment and semantic (Word2Vec) information. Experiments on a held-out test set show that our irony classifier benefits from this combined information, yielding an F_1-score of 67.66%. When explicit hashtag information like *#irony* is included in the data, the system even obtains an F_1-score of 92.77%. A qualitative analysis of the output reveals that recognising irony that results from a polarity clash appears to be (much) more feasible than recognising other forms of ironic utterances (e.g., descriptions of situational irony).

1 Introduction

With the emergence of the social web, a large part of our daily communication has moved online. As a result, the past decade has seen an increased research interest in text mining on social media data. The frequent use of irony in this genre (Ghosh and Veale, 2016; Maynard and Greenwood, 2014) has important implications for tasks such as sentiment analysis and opinion mining (Liu, 2015), which aim to extract positive and negative opinions automatically from online text. Irony detection is therefore crucial if we want to push the state of the art in sentiment analysis or, more broadly, any task involving text interpretation (e.g., cyberbullying detection).

Most computational approaches to date model irony by relying solely on categorical labels like irony hashtags (e.g., *#irony, #sarcasm*) assigned by the author of the text. To our knowledge, no guidelines presently exist for the more fine-grained annotation of irony in social media content without exploiting this hashtag information. In order to understand how irony is linguistically realised and how it can be recognised in text, we developed a set of annotation guidelines for identifying specific aspects and forms of irony that are susceptible to computational analysis. We collected a Twitter corpus containing 3,000 English tweets with an irony hashtag and, based on these guidelines, manually annotated them for the presence of irony. We explored the feasibility of automatic irony detection by relying on a varied set of features, including lexical, shallow syntactic, sentiment and lexical semantic information.

The remainder of this paper is structured as follows. Section 2 includes an overview of existing work on defining and modelling irony. Section 3 zooms in on the corpus construction and annotation. Section 4 outlines the experiments, and Section 5 presents the results. Section 6 concludes the paper with some prospects for future research.

2 Related Research

Since many years, irony has been a frequent topic of discussion in linguistics and philosophy. More recently, researchers in the field of natural language processing (NLP) have shown an increasing interest in the subject, trying to formalise the concept of irony, and detect it automatically. Different types of irony can be distinguished. Kreuz and Roberts (1993) define four types of irony: (i) Socratic irony and (ii) dramatic irony, both explained as a tension between what the hearer knows and what the speaker pretends

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2730–2739, Osaka, Japan, December 11-17 2016.

to know (with the latter entailing a performance aspect), (iii) Irony of fate, which involves an incongruence between two situations, similarly to what is commonly understood as situational irony (Lucariello, 1994), and (iv) verbal irony, which implies a speaker who intentionally says the opposite of what they believe.

In this research, we focus on verbal irony and (written forms of) situational irony. A popular definition of verbal irony is saying the opposite of what is meant (Grice, 1975). Though often criticised (e.g., Giora (1995); Sperber and Wilson (1981)), this definition is commonly used in contemporary research on the automatic modelling of irony (Kunneman et al., 2015). When describing how irony works, many studies struggle to distinguish between verbal irony and sarcasm. Some consistently use one of the two terms (e.g., Grice (1975); Sperber and Wilson (1981)), or consider both as essentially the same phenomenon (e.g., Attardo (2000); Reyes et al. (2013)). Other studies claim that sarcasm and verbal irony do differ in some respects, stating that ridicule (Lee and Katz, 1998), hostility and denigration (Clift, 1999), and the presence of a victim (Kreuz and Glucksberg, 1989) play a more important role in sarcasm than in irony. To date, however, experts do not formally agree on the distinction between irony and sarcasm. For the present research, we elaborated a working definition (Section 3.1) that does not distinguish between the two either, and refer to this linguistic form as (verbal) irony.

Automatic irony detection has received increased research interest in the past few years, with one of the main motivations being the improvement of sentiment analysis systems. In effect, as irony implicitly alters the polarity of an utterance, its recognition is important for the development and refinement of sentiment classification systems. Computational approaches to irony detection involve supervised or semi-supervised learning. More recently, researchers have been investigating deep learning by defining neural networks for irony detection. Twitter is a popular data genre when training supervised models for irony detection, one of the main advantages being that it minimises (or even discards) the annotation effort as it contains self-describing hashtags like *#sarcasm* and *#irony*. Davidov et al. (2010) experimented with 6 million tweets and 66,000 Amazon product reviews. They built an algorithm (*SASI*) based on semi-supervised learning and exploited syntactic patterns and punctuation as features, yielding F-scores of 0.79 (Amazon) and 0.83 (Twitter). Reyes et al. (2013) built a corpus of 40,000 tweets and divided it into four topics based on hashtag information (*irony, education, humour* and *politics*). They made use of Decision trees and Naïve Bayes exploiting a rich set of features capturing *style* (e.g., n-grams), *emotional scenarios* (e.g., imagery), *signatures* (e.g., pointedness), and *unexpectedness* (e.g., temporal imbalance). Their approach yields F= 0.72 on recognising the irony topic. Barbieri and Saggion (2014) experimented with the same corpus as Reyes et al. (2013) and used Random Forest and Decision Tree as classifiers. They made use of a varied set of features (word frequency, writing style, punctuation, ambiguity, etc.), and performed binary classification experiments to distinguish irony from each one of the three other topics: education (F= 0.73), humour (F= 0.75), and politics (F= 0.75). Riloff et al. (2013) presented a bootstrapping algorithm to automatically learn sequences of positive sentiment and negative situation phrases from ironic tweets. When adding n-grams and sentiment lexicon features, their SVM classifier achieved an F-score of 0.51. Kunneman et al. (2015) collected a dataset of 800,000 tweets. They made use of Balanced Winnow, exploiting word uni-, bi- and trigrams as features, and obtained an accuracy of 87% on the ironic tweets. In line with Wallace's (2015) claim that text-based features are too shallow and that context and semantics are required for reliable irony detection, a recent study from Ghosh and Veale (2016) describes neural-network-based semantic modelling for irony detection. The researchers compared the performance of an SVM model exploiting shallow features to that of neural networks capturing semantic information and demonstrated that the latter outperformed the SVM model (F= 0.92 vs. 0.73).

It is important to note that, in the above-described papers, training data is often obtained by collecting tweets with hashtags like *#sarcasm* and *#irony* and labelling them accordingly (i.e., tweets containing such hashtags are labeled as ironic, whereas tweets devoid of such hashtags are considered non-ironic). An important contribution of this paper is that, after collecting data based on irony hashtags, all tweets were manually labeled for irony based on a fine-grained annotation scheme (Van Hee et al., 2016b). Furthermore, to estimate the impact of irony hashtags on a detection system, we evaluate the performance

of our classifier before and after removing them from the corpus.

3 Corpus Description

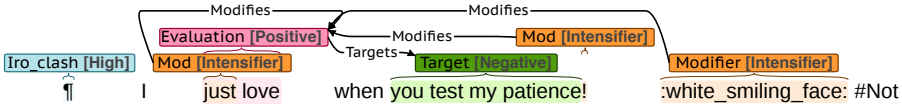
We collected a dataset of 3,000 English tweets by searching for the hashtags *#irony*, *#sarcasm* and *#not*. Unlike many other approaches, all tweets were manually annotated for irony presence based on a newly developed annotation scheme (Van Hee et al., 2016b), which resulted in 2,396 ironic and 604 non-ironic tweets. Some example annotations are presented in Section 3.1.

3.1 Corpus Annotation

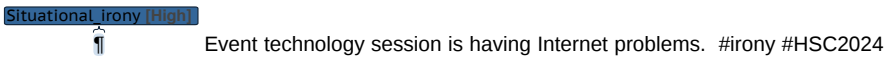
The main goal of our annotation guidelines was to develop a set of reproducible coding principles to mark irony in (social media) text. As we ultimately want to be able to model irony in text, without relying on additional (i.e., hashtag) information provided by the writer of the message, our main focus was on disentangling expressions of verbal irony. In accordance with the classic account of irony, i.e., 'saying the opposite of what is meant' (Grice, 1975), we define verbal irony as an *evaluative expression whose polarity (i.e., positive, negative) is inverted between the literal and the intended evaluation, resulting in an incongruence between the literal evaluation and its context*. While a detailed overview of the annotation procedure is provided in the guidelines (Van Hee et al., 2016b), we briefly discuss the main principles below. Brat was used as annotation environment (Stenetorp et al., 2012).

At the **tweet level**, annotators indicated whether the tweet was (i) ironic by means of a clash (example 1), (ii) contained another type of irony (e.g., situational irony, example 2), or was (iii) not ironic (example 3).

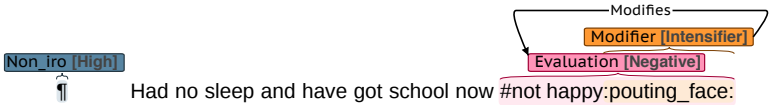
(1)



(2)



(3)



In order to better understand how this irony is realised, the tweets were also annotated **below tweet level**. In case of irony expressed by means of a clash, it was also indicated whether an irony-related hashtag (e.g., *#sarcasm*, *#irony*, *#not*) was required for recognising the irony. Furthermore, annotators were asked to signal variants of verbal irony that are particularly harsh (i.e., carrying a mocking or ridiculing tone with the intention to hurt someone), since it has been shown that harshness may be a useful feature to distinguish between irony and sarcasm (Barbieri and Saggion, 2014). Sentence 4 shows an example of such a harsh tweet.

(4) Shout outs to the guy who took a shower in 1 Million before heading out. The WHOLE BUS thanks you #Sarcasm

The annotators also marked all evaluations contained by the tweet and indicated text spans that contrast with the polarity expressed by that evaluation (i.e., *targets*) (See sentence 5 for an example). For each evaluation:

- the polarity was indicated;
- modifiers were annotated (if present);
- (in the case of ironic tweets) the target of the evaluation was indicated. Also, the implicit polarity of the target was defined based on context, world knowledge or common sense.

(5) The most hideous spider, that makes me feel sooo much better. #not
→ Evaluations: *most hideous* (neg. polarity), *makes me feel sooo much better* (pos. polarity).
→ Modifiers: *most*, and *sooo much*.
→ Targets: *the most hideous spider* (as target of *makes me feel sooo much better*), which holds a negative implicit polarity.

3.2 Annotation Statistics

To assess the validity of the annotations, inter-annotator agreement was measured between three independent annotators. Kappa scores for (i) the annotation of irony (ironic vs. not ironic) and (ii) the decision whether an irony hashtag was required to recognise the irony are 0.72 and 0.67, respectively.

Table 1 presents the distribution of the corpus as divided by the different annotation labels[1]. As can be inferred from the table, most instances that were labeled as ironic belong to the category *ironic by means of a clash*. When we zoom in on the category *other type of irony*, we can distinguish two subcategories: *situational irony* and *other verbal irony*. Whereas the former encompasses (written instances of) ironic situations and comprises the majority of this annotation class, the latter contains instances of irony that describe neither situational irony, nor a clash between two polarities (viz. the literal and the intended one). Nevertheless, they are still considered to be ironic. We refer to Van Hee et al. (2016a) for more details on the corpus statistics.

Ironic by means of a clash	**Other type of irony**		**Not ironic**	**Total**
	Situational irony	*Other verbal irony*		
1,728	401	267	604	3,000

Table 1: Statistics of the annotated corpus: number of instances per annotation category.

For our binary classification experiments, we merged the categories *ironic by means of a clash* and *other type of irony*. As we wanted a balanced dataset (ironic vs. not ironic), we added 1,792 non-ironic tweets to the current corpus. These tweets are from the same Twitter users from which we collected the initial 3,000 tweets (Table 1) and contain no irony-related hashtags. As such, our experimental corpus consists of 4,792 tweets, of which 2,396 are ironic and another 2,396 are not ironic.

4 Experiments

In our classification experiments, we evaluated the viability of automatically recognising verbal irony in tweets. To this end, we constructed a pipeline and ran a series of experiments while exploiting different feature groups. These include standard text classification features (i.e., bags-of-words, lexical and syntactic features), and features based on existing sentiment lexicons. Furthermore, we added semantic features based on Word2Vec clusters, which is, to our knowledge, novel in SVM-based approaches to irony detection.

For the classification experiments, we split the randomised corpus (4,792 instances) into an 80% training and 20% test set for evaluation, resulting in a training set of 3,834 instances and a held-out test set of 958 instances. Both sets show a balanced irony distribution (50% ironic vs. 50% not ironic). Furthermore, all annotation categories (Table 1), as well as the extra non-ironic instances that were added (Section 3.2) are equally distributed among the train and test sets.

4.1 Preprocessing

After constructing the corpus, all emoji were replaced by their name or a description using the Python Emoji module[2] to facilitate annotation and processing of the data. Furthermore, we normalised hyperlinks and @-replies or mentions to *http://someurl* and *@someuser*, respectively.

Other preprocessing steps involve tokenisation and PoS-tagging (Gimpel et al., 2011), lemmatisation (Van de Kauter et al., 2013) and named entity recognition (Ritter et al., 2011).

[1] Due to a refinement of the annotations, the corpus statistics are slightly different from a first version of the annotated corpus described in Van Hee et al. (2016a).

[2] https://github.com/carpedm20/emoji/.

4.2 Information Sources

For the automatic irony detection system, we implemented a variety of features that represent every instance within a (sparse) feature vector.

- As **lexical** features, we included bags-of-words (BoW) features that represent a tweet as a 'bag' of its words or characters. We incorporated token unigrams and bigrams and character trigrams and fourgrams. Furthermore, a set of numeric and binary features were included containing information about (i) character and (ii) punctuation flooding, (iii) punctuation and (iv) capitalisation, (v) hashtag frequency and (vi) the hashtag-to-word ratio, (vii) emoticon frequency, and (viii) tweet length. Where relevant, numeric features were normalised by dividing them by the tweet length in tokens.

- As **syntactic** features, we integrated four Part-of-Speech features for each of the 25 tags in the tagset. These indicate for each PoS-tag (i) whether it occurs in the tweet or not, (ii) whether the tag occurs 0, 1, or > 2 times, (iii) the frequency of the tag in absolute numbers and (iv) as a percentage. Also the number of interjections was added as a feature. Furthermore, we included a binary feature indicating a 'clash' between verb tenses in the tweet (see Reyes et al. (2013)). Finally, we integrated four features indicating the presence of named entities in a tweet: one binary feature and three numeric features, indicating (i) the number of named entities in the text, (ii) the number and (iii) frequency of tokens that are part of a named entity.

- Six **sentiment lexicon** features were implemented based on existing sentiment lexicons: AFINN (Nielsen, 2011), General Inquirer (GI) (Stone et al., 1966), MPQA (Wilson et al., 2005), the NRC Emotion Lexicon (Mohammad and Turney, 2013), Liu's opinion lexicon (Hu and Liu, 2004), and Bounce (Kökciyan et al., 2013). For each lexicon, five numeric and one binary feature were derived:

  - the number of positive, negative and neutral lexicon words averaged over text length;
  - the overall tweet polarity (i.e., the sum of the values of the identified sentiment words);
  - the difference between the highest positive and lowest negative sentiment values;
  - a binary feature indicating whether there is a polarity contrast (i.e., at least one positive and one negative sentiment word from the lexicon are present in the tweet).

  The sentiment lexicon features were extracted in two ways: (i) by considering all tokens in the instance and (ii) by considering only hashtag tokens (e.g., *lovely* from *#lovely*). We took negation cues into account by flipping the polarity of a sentiment word when it occurred in a negation relation.

- As **semantic** information, we used word embedding cluster features generated with Word2Vec (Mikolov et al., 2013). The word embeddings were generated from a separate background corpus of 45,251 English tweets, collected with the hashtags *#sarcasm*, *#irony* and *#not*. More precisely, we ran Word2Vec on this corpus, applying the CBoW model, a context size of 8, a word vector dimensionality of 200 features, and a cluster size of $k = 2{,}000$[3]. The following are two example clusters: [#chistecorto #dailysarcasm #fun #sarcastically #sarcastichumor] and [#exams #nosleep #10am editing essay grading psychology stress revision]. The clusters were implemented as binary features, indicating for each cluster whether a word contained by that cluster occurs in the tweet.

In total, we have four feature groups. Based on each of them, we trained a binary classifier which was then tested on the held-out set. After evaluating the performance of each feature group individually, another experiment was run with the combined feature groups (comprising 100,278 individual features).

[3]To define k, we performed 5-fold cross validation experiments on the training data, exploiting features based on different cluster sizes (100; 200; 500; 1,000 and 2,000).

4.3 Experimental Setup

As mentioned earlier, we conducted binary classification experiments for detecting ironic tweets by exploiting lexical, syntactic, sentiment lexicon and semantic features. One of the contributions of this paper is to measure the impact of irony-related hashtags (e.g., *#irony*) in the dataset. To this end, we ran two sets of experiments based on each feature group: one before and another after removing such hashtags from the corpus.

We used LIBSVM (Chang and Lin, 2011) with the standard RBF kernel as the classification algorithm. As shown by Keerthi and Lin (2003), a nonlinear kernel like RBF (or *Gaussian*) is at least as good as a linear one if it is properly tuned. The LIBSVM parameters C and γ were therefore optimised for each experiment exploiting a different feature group, and by means of a cross-validated grid search on the complete training data. During the parameterisation, γ is varied between 2^{-15} and 2^{3} (stepping by factor 4), while the cost parameter C is varied between 2^{-5} and 2^{15} (stepping by factor 4). In both setups, the optimised values for C and γ were 2^{3} and 2^{-11}, respectively. These optimal parameter settings were then used to build a model for each feature group using all the training data, which was evaluated on the held-out test set (Section 5).

5 Results

This section presents the experimental results. As mentioned earlier, we tested the validity of four different feature groups for automatic irony detection, comprising lexical, syntactic, sentiment and semantic features. Finally, all feature groups were combined to see whether they provide complementary information to the classifier. Each feature group was evaluated in two experimental setups: one where irony hashtags like *#irony*, *#sarcasm* and *#not* were removed from the corpus, and another where this hashtag information was included.

5.1 Experimental Results on the Held-out Test Set

Table 2 presents the results obtained with each feature group separately, and with the combined set. Besides accuracy, we report F_1-score, precision and recall (on the positive class) as the evaluation metrics.

For the baseline system, we included only bag-of-words features. No parameter tuning was done. As the system consistently predicts the positive class (i.e., *ironic*), its recall is 100% while its accuracy is equal to the positive class distribution.

	Setup 1: without hashtag information				*Setup 2: with hashtag information*			
Features	**Accuracy**	**F_1-score**	**Precision**	**Recall**	**Accuracy**	**F_1-score**	**Precision**	**Recall**
BoW (baseline)	48.96	65.73	48.96	100	48.96	65.73	48.96	100
Lexical	66.91	66.38	66.03	66.74	90.92	91.41	85.11	98.72
Syntactic	63.57	64.57	61.63	67.80	80.48	81.68	75.54	88.91
Sentiment	59.29	55.78	59.56	52.45	74.95	71.22	81.37	63.33
Semantic	64.41	63.53	63.73	63.33	88.73	89.45	82.52	97.65
Combined	68.16	**67.66**	67.30	68.02	92.48	**92.77**	87.67	98.51

Table 2: Experimental results on the held-out test set in both setups.

Evidently, the experimental setup with hashtags included (setup 2) performs best. To understand the performance of this system better, we compare its accuracy (92.48%) to a baseline that simply classifies all tweets containing an irony-related hashtag as ironic and all other tweets as not ironic, yielding an accuracy of 87%. In both setups, the best performance is achieved by the combined feature set (F= 67.66 and 92.77). This partly supports the findings of Wallace (2015) that verbal irony cannot be recognised through lexical clues alone. Nevertheless, lexical information does seem to be of key importance for this task, as the corresponding system obtained the second best score (F= 66.38 and 91.41). An explanation could be the nature of Twitter data: due to its limited length, a tweet is prone to be misunderstood, which may encourage people to use explicit lexical clues when speaking ironically. In both setups, sentiment features perform least well for this task (F= 55.78 and 71.22), which demonstrates that (explicit) polarity

information is not sufficient for decent irony recognition. Nevertheless, the annotations revealed that many ironic tweets showed a polarity contrast (mostly between evaluations and *targets*). In future work, we will therefore explore methods to model this clash between explicit and implicit (i.e., inferred from world knowledge) sentiment expressions.

5.2 Analysis of the Results

Judging from the raw performance results, irony detection in Twitter data benefits from a combined set of information sources (i.e., lexical, syntactic, sentiment and semantic). In a next step, we investigate whether the combined system is significantly better than the baseline and the systems we built based on each feature group. To this end, we applied 10 paired samples t-test (α= 0.05) after bootstap resampling (Noreen, 1989): one for each system (including the baseline) that we compare against the combined system, in the two experimental setups. Concretely, we drew samples (n= 10,000) with replacement from the output of each system and of equal size of the output (n= 958, the number of test instances). Each sample was then evaluated using macro-averaged F_1-score (on the positive class), after which we applied a paired samples t-test to compare the mean scores for both systems. We found a significant difference ($p<$ 0.05) for all system pairs.

For the experiments in this paper, we concentrated on two classification labels being ironic and not ironic, thus casting the problem into a detection (or binary classification) task. To this end, the annotation labels *ironic by clash*, *other type of irony* and *situational irony* were combined into the positive class in both our training and test sets. In the following paragraph, we zoom in on these 'subclasses' of irony and evaluate the performance of our system on each one of them in the test set. The subclasses are represented by 346 (*irony by clash*), 62 (*other irony*) and 61 (*situational irony*) instances in the test set. We will also take a closer look at the two types of non-ironic tweets (i.e., with and without an irony-related hashtag) in the dataset.

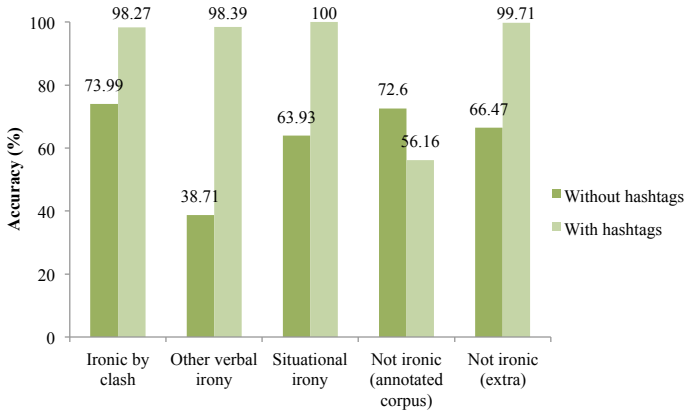


Figure 1: System accuracy on different types of ironic and non-ironic tweets.

As shown in Figure 1, the performance of the system increases significantly when hashtag information is included, reaching an accuracy of up to 100% for the recognition of tweets that describe situational irony. Without this hashtag information, the best performance is achieved on *ironic by clash* (acc.= 73.99%), followed by *situational irony* (acc.= 63.93%). The score on *other verbal irony* is low, however (acc.= 38.71%). This would suggest that detecting verbal irony is (much) more feasible when the irony results from contrasting evaluations, as opposed to other types of verbal irony.

We also had a closer look at the category 'not ironic'. Important to recapitulate is that 25% of these tweets contain an irony hashtag (they were part of the originally collected 3,000 tweets (cf. Table1)). When looking at the classification performance on these tweets, we observe that, when irony hashtags are included in the data, the accuracy obtained is 56%. This demonstrates that the system does not simply rely on hashtag information (since this would result in an accuracy of 0% on this category). Another category that is subject to a qualitative analysis, are the tweets for which annotators had indicated that an

explicit hashtag (e.g., *#irony*) was required to recognise the irony. Intuitively, the system would perform better on the instances where no such hashtag is needed, especially when these hashtags are removed from the data. Effectively, this hypothesis was confirmed by our experiments (setup 1), revealing a higher accuracy on ironic tweets where no hashtag was required to recognise the irony (83.43%) than on those where it was (63.64%). The accuracy on the latter still being relatively high, we can conclude that the irony in our dataset is moderately to strongly lexicalised. This conclusion is in line with the performance of the system exploiting lexical features (cf. Table 2).

We see that our combined system (F_1= 67.66) compares favourably to that of Riloff et al. (2013) (F_1= 0.51), who describe a similar experimental setup to the one presented here. However, comparison with state of the art is not trivial, given the size of our dataset and the different definitions of irony that are used among researchers. In effect, most studies make use of large corpora that are labeled based on hashtag information (e.g., Kunneman et al. (2015), Reyes et al. (2013)). Furthermore, some approaches (e.g., Riloff et al. (2013)) focus on one particular type of irony, whereas the present research takes different types into account.

6 Conclusion and Future Work

In this paper we developed and tested a system for irony detection in English Twitter data. We collected 3,000 tweets with irony hashtags (i.e., *#irony*, *#sarcasm*, and *#not*) and manually annotated them according to a newly developed annotation scheme for verbal irony. To balance the ironic vs. not ironic classes in the experiments, another 1,792 non-ironic tweets from a random Twitter sample were added, which resulted in an experimental corpus of 4,792 tweets. We explored the viability of automatic irony detection using different feature groups (lexical, syntactic, sentiment, semantic and combined). Additionally, we compared the system's performance on our dataset with and without removing the irony-related hashtags. The results on a held-out test set revealed that irony detection benefits from a combined feature set: our binary classifier yields an F_1-score of 67.66% on the dataset devoid of irony hashtags, while obtaining F_1= 92.77% with irony hashtags included in the dataset. Although lexical features are assumed insufficient for decent irony recognition (Wallace, 2015), we experimentally show that they do provide relevant information, as the corresponding system scored second best, after the combined system.

A qualitative analysis of the different types of ironic tweets revealed that our classifier performed best on tweets where the irony results from a polarity contrast (i.e., the polarity of the expressed sentiment is opposite to what is meant). Given that ironic tweets are prone to implicit sentiment, future research will focus on recognising and understanding such implicit evaluations by making use of world knowledge or common sense. This would allow to identify a polarity contrast in typically ironic utterances where a part of the evaluation is implicit, such as *monday mornings* in the sentence *Monday mornings are my fave! :)*.

Acknowledgements

The work presented in this paper was carried out in the framework of the AMiCA (IWT SBO-project 120007) project, funded by the government agency for Innovation by Science and Technology (IWT).

References

Salvatore Attardo. 2000. Irony as relevant inappropriateness. *Journal of Pragmatics*, 32(6):793–826.

Francesco Barbieri and Horacio Saggion. 2014. Modelling Irony in Twitter: Feature Analysis and Evaluation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, LREC'14, Reykjavik, Iceland. European Language Resources Association (ELRA).

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27.

Rebecca Clift. 1999. Irony in conversation. *Language in Society*, 28(4):523–553.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised Recognition of Sarcastic Sentences in Twitter and Amazon. In *Proceedings of the 14th Conference on Computational Natural Language Learning*, CoNLL'10, pages 107–116, Uppsala, Sweden. Association for Computational Linguistics.

Aniruddha Ghosh and Tony Veale. 2016. Fracking Sarcasm using Neural Network. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 161–169, San Diego, California. Association for Computational Linguistics.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech Tagging for Twitter: Annotation, Features, and Experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, HLT'11, pages 42–47, Portland, Oregon. Association for Computational Linguistics.

Rachel Giora. 1995. On irony and negation. *Discourse Processes*, 19(2):239–264.

H. P. Grice. 1975. Logic and Conversation. In Peter Cole and Jerry L. Morgan, editors, *Syntax and Semantics: Vol. 3*, pages 41–58. Academic Press, New York.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD'04, pages 168–177, New York, NY. Association for Computing Machinery.

Sathiya S. Keerthi and Chih-Jen Lin. 2003. Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel. *Neural Computation*, 15(7):1667–1689, July.

Nadin Kökciyan, Arda Çelebi, Arzucan Özgür, and Suzan Üsküdarl. 2013. Bounce: Sentiment classification in Twitter using rich feature sets. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the 7th International Workshop on Semantic Evaluation*, SemEval 2013, pages 554–561, Atlanta, Georgia, USA. Association for Computational Linguistics.

Roger J. Kreuz and Sam Glucksberg. 1989. How to be sarcastic: The echoic reminder theory of verbal irony. *Journal of Experimental Psychology: General*, 118(4):374.

Roger J. Kreuz and Richard M. Roberts. 1993. On Satire and Parody: The Importance of Being Ironic. *Metaphor and Symbolic Activity*, 8(2):97–109.

Florian Kunneman, Christine Liebrecht, Margot van Mulken, and Antal van den Bosch. 2015. Signaling sarcasm: From hyperbole to hashtag. *Information Processing & Management*, 51(4):500–509.

Christopher J. Lee and Albert N. Katz. 1998. The Differential Role of Ridicule in Sarcasm and Irony. *Metaphor and Symbol*, 13(1):1–15.

Bing Liu. 2015. *Sentiment Analysis - Mining Opinions, Sentiments, and Emotions*. Cambridge University Press.

Joan Lucariello. 1994. Situational Irony: A Concept of Events Gone Awry. *Journal of Experimental Psychology: General*, 123(2):129–145.

Diana Maynard and Mark Greenwood. 2014. Who cares about Sarcastic Tweets? Investigating the Impact of Sarcasm on Sentiment Analysis. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, LREC'14, Reykjavik, Iceland.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Finn Årup Nielsen. 2011. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. In *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages*.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses : An Introduction*. Wiley-Interscience.

Antonio Reyes, Paolo Rosso, and Tony Veale. 2013. A multidimensional approach for detecting irony in Twitter. *Language Resources and Evaluation*, pages 238–269.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as Contrast between a Positive Sentiment and Negative Situation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP'13, pages 704–714. Association for Computational Linguistics.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named Entity Recognition in Tweets: An Experimental Study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP'11, pages 1524–1534, Edinburgh, United Kingdom. Association for Computational Linguistics.

Dan Sperber and Deirdre Wilson. 1981. Irony and the Use - Mention Distinction. In Peter Cole, editor, *Radical Pragmatics*, pages 295–318. Academic Press, New York.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: A Web-based Tool for NLP-assisted Text Annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL'12, pages 102–107, Avignon, France. Association for Computational Linguistics.

Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, and Daniel M. Ogilvie. 1966. The General Inquirer: A Computer Approach to Content Analysis. *MIT Press*.

Marjan Van de Kauter, Geert Coorman, Els Lefever, Bart Desmet, Lieve Macken, and Véronique Hoste. 2013. LeTs Preprocess: The multilingual LT3 linguistic preprocessing toolkit. *Computational Linguistics in the Netherlands Journal*, 3:103–120.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2016a. Exploring the Realization of Irony in Twitter Data. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, LREC'16, pages 1794–1799, Portorož (Slovenia). European Language Resources Association (ELRA).

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2016b. Guidelines for Annotating Irony in Social Media Text, version 2.0. Technical Report 16-01, LT3, Language and Translation Technology Team–Ghent University.

Byron C. Wallace. 2015. Computational irony: A survey and new perspectives. *Artificial Intelligence Review*, 43(4):467–483.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT'05, pages 347–354, Vancouver, Canada. Association for Computational Linguistics.

CNN- and LSTM-based Claim Classification in Online User Comments

Chinnappa Guggilla and **Tristan Miller** and **Iryna Gurevych**
Research Training Group AIPHES
Ubiquitous Knowledge Processing (UKP) Lab
Technische Universität Darmstadt
https://www.aiphes.tu-darmstadt.de/
https://www.ukp.tu-darmstadt.de/

Abstract

When processing arguments in online user interactive discourse, it is often necessary to determine their bases of support. In this paper, we describe a supervised approach, based on deep neural networks, for classifying the claims made in online arguments. We conduct experiments using convolutional neural networks (CNNs) and long short-term memory networks (LSTMs) on two claim data sets compiled from online user comments. Using different types of distributional word embeddings, but without incorporating any rich, expensive set of features, we achieve a significant improvement over the state of the art for one data set (which categorizes arguments as factual vs. emotional), and performance comparable to the state of the art on the other data set (which categorizes propositions according to their verifiability). Our approach has the advantages of using a generalized, simple, and effective methodology that works for claim categorization on different data sets and tasks.

1 Introduction

Argumentation mining is a relatively new subfield in natural language processing that aims to automatically identify and extract arguments, and their underlying structures, from textual documents (Moens et al., 2007; Palau and Moens, 2009; Wyner et al., 2010; Feng and Hirst, 2011; Ashley and Walker, 2013; Stab and Gurevych, 2014). Some such documents are written by professionals and contain well-formed, explicit arguments—i.e., propositions supported by evidence and connected through reasoning. However, informal arguments in online argumentative discourses can exhibit different styles. Recent work has begun to model different aspects of these naturally occurring lay arguments, with tasks including stance classification (Somasundaran and Wiebe, 2009; Walker et al., 2012), argument summarization (Misra et al., 2015), sarcasm detection (Justo et al., 2014) and classification of propositions and arguments (Park and Cardie, 2014; Park et al., 2015; Oraby et al., 2015). Of particular interest is the fact that arguments in online user comments, unlike those written by professionals, often have inappropriate or missing justifications. Recognizing such propositions and determining the appropriate types of support can be useful for assessing the strength of the supporting information and, in turn, the strength of the whole argument.

To this end, two previous studies have produced data sets and methods for classifying propositions in online argumentative discourse. The first of these studies (Park and Cardie, 2014) compiled online user comments from a discussion website and developed a framework for automatically classifying each proposition as either "unverifiable", "verifiable non-experiential", or "verifiable experiential", where the appropriate types of support are reason, evidence, and optional evidence, respectively. The second study, Oraby et al. (2015), uses a different online corpus (Walker et al., 2012) of short argumentative responses to quotes, and classifies each response as either "factual" or "feeling" according to whether the support invoked appeals to facts or to emotions. In this paper, we use the term "claim" loosely to refer to an individual proposition (a sentence or independent clause) in an argument, or to a short argumentative text containing one or more propositions.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2740–2751, Osaka, Japan, December 11-17 2016.

In classifying propositions, Park and Cardie (2014) followed previous work such as Reed et al. (2008) and Palau and Moens (2009), employing supervised learning methods. Despite using a rich set of linguistic features, these approaches suffer from low accuracy. Moreover, generating these features can be a tedious and complex process. In this paper, we show that state-of-the-art performance in claim classification for online user comments can be achieved without the need for expensive features. Our work, which employs CNN- and LSTM-based deep neural networks, is inspired by advances in sentence classification (Kim, 2014) and sequence classification (Hochreiter and Schmidhuber, 1997) using distributional word representations and deep learning. In particular, our approach leverages word2vec[1] distributional embeddings, dependency context–based embeddings (Levy and Goldberg, 2014), and factuality/certainty-indicating embeddings for improving claim classification. (We refer to these embeddings as *linguistic embeddings*, as these are compiled from linguistic annotations such as dependency relations, verb modalities, and actuality information.) In this paper, we separately evaluate the usefulness of word and linguistic embeddings in the claim classification task on both the aforementioned data sets. We also concatenate (stack) these embeddings and show how these stacked embeddings, as well as tuning of the hyper-parameters, further improves claim classification performance.

2 Background

In this section, we introduce two main approaches for claim classification: feature-rich supervised learning and distributional word embeddings. We then discuss the recent use of convolutional neural networks and long short-term memory networks in the related task of sentence classification.

2.1 Methods Based on a Rich Set of Features

Oraby et al. (2015) classify arguments as emotional or factual using a set of linguistic patterns extracted from unlabelled arguments, provided the argument matches at least three patterns in the category. Although this approach has good precision, its recall is significantly lower than that of a supervised unigram baseline using Bayesian classifier.

Park and Cardie (2014) classify propositions as verifiable non-experiential, verifiable experiential, or unverifiable using a support vector machine (SVM). The classifier employs a rich set of features including *n*-grams, part of speech tags, imperative expressions, speech events, emotions, sentiment, person, and tense. Though this approach classifies unverifiable statements reasonably well, its performance on the two classes of verifiable propositions is low (44–70% F_1). In light of the observation that certain types of propositions tend to occur together, Park et al. (2015) propose an intuitive extension to this approach, framing the proposition classification task as a sequence-labelling problem. This extended approach employs conditional random fields (CRF) using dictionary-based features along with all the features from the original technique. However, it resulted in lower accuracy than the SVMs.

Ferreira and Vlachos (2016) addressed the task of determining the stance of news article headlines with respect to claims from a data set of rumours. The authors used a logistic regression classifier using various features, such as bag of words, paraphrase entailment alignment scores, and word2vec embedding features, that examine the headline and its agreement with the claim. The work in this paper is focused on stance classification but the claims in the data set are related to the data sets used in our work.

2.2 Distributional Word Embeddings

Traditional supervised learning approaches to NLP tasks depend heavily on manual annotation, and often suffer from data sparseness. Distributional representations of words, also known as word embeddings, can be learned from large, unlabelled corpora using neural networks, and encode both syntactic and semantic properties of words. Studies have found the learned word vectors to capture linguistic regularities and to collapse similar words into groups (Mikolov et al., 2013b). Their utility in tasks such as sentiment classification (Kim, 2014) is well attested.

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

Dependency-based Embeddings. Claims containing multiple clauses or propositions might be better distinguished with the help of dependency embeddings inferred from the respective proposition contexts. Consider the following claim from one of our data sets: "The Governor said that he enjoyed it." In this claim, the main clause, "The Governor said", is the core proposition, which excludes consideration of the remainder. The reason is that "said" is a reporting predicate, so it is unnecessary to verify whether or not the governor really has enjoyed the object mentioned in the subordinate clause. In some other claims, it is the subordinate rather than the main clause predicate that decides the claim type. Park and Cardie (2014) extracted clause-specific features using the Stanford syntactic parser and the Penn Treebank. (Merely using clause tags without capturing dependencies for important clauses may not help much in distinguishing objective verifiable claims from unverifiable subjective ones.) Park and Cardie (2014) also used tense and person counts for distinguishing verifiable claims from unverifiable claims. We hypothesize that word2vec and dependency context–based embeddings can inherently capture these linguistic characteristics and can replace these features. Dependency context based embeddings capture functional similarities across the words using different contexts (Levy and Goldberg, 2014). Komninos and Manandhar (2016) have shown that dependency-based models produce word embeddings that better capture functional properties of words for question type classification and relation detection.

Task-specific Embeddings. Compiling embeddings for the specific vocabulary present in the task data can also be helpful in a classification task. Tang et al. (2014) use enriched task-specific word embeddings and show improvement in a Twitter sentiment classification task. Park and Cardie (2014) compiled a speech-event lexicon containing the most frequent speech anchors (predicates such as "said" and "wrote") from MPQA 2.0, a corpus manually annotated for opinions and other private states. These anchors can help in correctly distinguishing verifiable claims from unverifiable ones when the propositions contain both objective and subjective expressions. In our work, we use factual embeddings learned from the labelled FactBank corpus (Saurí and Pustejovsky, 2009) containing various speech event predicates (see §3.3). Such factual embeddings could help in resolving various predicate ambiguities present in the argumentative propositions.

2.3 Deep Neural Networks for Text Classification

Deep neural networks, with or without word embeddings, have recently shown significant improvements over traditional machine learning–based approaches when applied to various sentence- and document-level classification tasks.

Kim (2014) have shown that CNNs outperform traditional machine learning–based approaches on several tasks, such as sentiment classification, question type classification, and subjectivity classification, using simple static word embeddings and tuning of hyper-parameters. Zhang et al. (2015) proposed character-level CNNs for text classification. Lai et al. (2015) and Visin et al. (2015) proposed recurrent CNNs, while Johnson and Zhang (2015) proposed semi-supervised CNNs for solving a text classification task. Tang et al. (2015) used a document classification approach based on recurrent neural networks (RNNs) and showed an improvement on a sentiment classification task. Palangi et al. (2016) proposed sentence embedding using an LSTM network for an information retrieval task. Zhou et al. (2016) proposed attention-based, bidirectional LSTM networks for a relation classification task. Augenstein et al. (2016) employed a weakly supervised conditional LSTM encoding approach to stance detection for unseen targets on Twitter stance detection data, and presented improved results. RNNs model text sequences effectively by capturing long-range dependencies among the words. LSTM-based approaches based on RNNs effectively capture the sequences in the sentences when compared to the CNN and SVM-based approaches.

3 Claim Classification

Here we present two deep learning–based methods for claim classification, the first of which uses CNNs and the second of which uses LSTMs. In §3.3, we also show how different pre-trained distributional linguistic embeddings are incorporated into CNNs and LSTMs to improve the classification results.

3.1 CNN-based Claim Classification

Collobert et al. (2011) adapted the original CNN proposed by LeCun and Bengio (1995) for modelling natural language sentences. Following Kim (2014), we present a variant of the CNN architecture with four layer types: an input layer, a convolution layer, a max pooling layer, and a fully connected softmax layer. Each claim in the input layer is represented as a sentence comprised of distributional word embeddings. Let $\vec{v}_i \in \mathbb{R}^k$ be the k-dimensional word vector corresponding to the ith word in the sentence. Then a sentence S of length ℓ is represented as the concatenation of its word vectors:

$$S = \vec{v}_1 \oplus \vec{v}_2 \oplus \cdots \oplus \vec{v}_\ell. \tag{1}$$

Word2vec embeddings which are learned using the bag-of-words representation of the contexts yield broad topical similarities, while using dependency-based contexts yields more functional similarities (Levy et al., 2015). In addition, with word2vec (E) embeddings, we use linguistically motivated pre-trained dependency embeddings (D) and task-specific factual embeddings (F) for capturing syntactic and functional regularities encoded in the propositions, in order to better distinguish different types of claims.

To incorporate these linguistic embeddings at word level into the learning process, we extend the network as illustrated in Figure 1a. Inspired by Baroni et al. (2012)'s supervised distributional concatenation method and a linguistically informed CNN (Ebert et al., 2015), we concatenate word2vec (E), dependency (D), and factual (F) word embeddings corresponding to the ith input word into a merged vector $\vec{c}_i \in \mathbb{R}^{k+m+n}$:

$$\vec{c}_i = [\vec{e}_i, \vec{d}_i, \vec{f}_i], \tag{2}$$

where $\vec{e}_i$, $\vec{d}_i$, and $\vec{f}_i$ represent, respectively, the concatenated word2vec, dependency, and factual embeddings corresponding to i_{th} word in the sentence. In the final representation, every input claim from the data set is represented using combined word2vec and linguistic embeddings in the network as in Equation 1, where each $\vec{v}_i = \vec{c}_i$.

In the convolution layer, for a given word sequence within a claim, a convolutional word filter P is defined. Then, the filter P is applied to each word in the sentence to produce a new set of features. We use a non-linear activation function such as rectified linear unit (ReLU) for the convolution process and max-over-time pooling (Collobert et al., 2011; Kim, 2014) at pooling layer to deal with the variable claim size. After a series of convolutions with different filters with different heights, the most important features are generated. Then, this feature representation, Z, is passed to a fully connected penultimate layer and outputs a distribution over different labels:

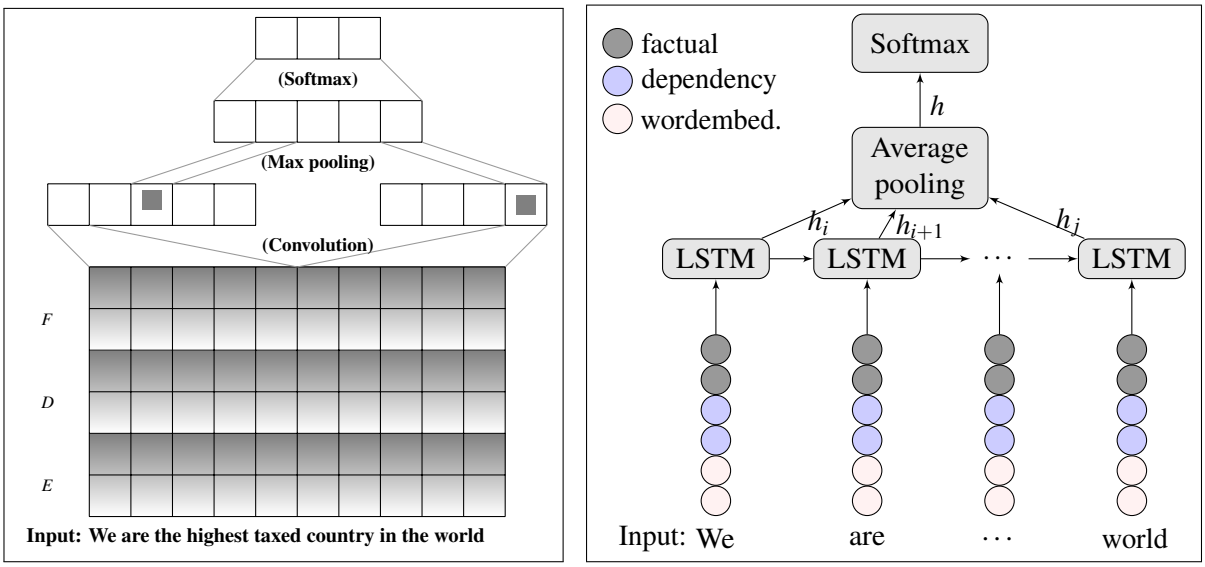
$$y = \text{softmax}(W \cdot Z + b), \tag{3}$$

where y denotes a distribution over different claims labels, W is the weight vector learned from the stacked representation of all embeddings from the training corpus, and b is the bias term.

3.2 LSTM-based Claim Classification

In case of CNN, concatenating words with various window sizes, works as n-gram models but do not capture long-distance word dependencies with shorter window sizes. A larger window size can be used, but this may lead to data sparsity problem. In order to encode long-distance word dependencies, we use long short-term memory networks, which are a special kind of RNN capable of learning long-distance dependencies. LSTMs were introduced by Hochreiter and Schmidhuber (1997) in order to mitigate the vanishing gradient problem (Gers et al., 2000; Gers, 2001; Graves, 2013; Pascanu et al., 2013).

The model illustrated in Figure 1b is composed of a single LSTM layer followed by an average pooling and a softmax regression layer. Each claim is represented as a sentence (S) in the input layer. Thus, from an input sequence, $S_{i,j}$, the memory cells in the LSTM layer produce a representation sequence $h_i, h_{i+1}, \ldots, h_j$. This representation sequence is then averaged over all time steps, resulting in a final feature representation h. Finally, this representation is fed to a logistic regression layer to predict the claim labels for unseen input claims.



(a) CNN with multiple word embeddings

(b) LSTM with multiple word embeddings

Figure 1: Illustration of two methods for claim classification

[There *always* **seems** to be some other amount] [I *later* **must** pay]
predicate (possibility) *modal (obligation)*

Figure 2: Example verifiable non-experiential claim with signals indicating factuality and certainity

As with the CNN architecture shown in the previous section, for each claim, we encode word2vec, dependency, and factual embeddings in the input layer into a variation of the standard LSTM network. As our results demonstrate, the LSTM encoder can effectively capture informative features from the concatenated embedding representation and classify different types of argumentative claims.

3.3 Word Embeddings

In order to better capture the syntactic contexts of words and the factuality indicators of propositions, we employ two linguistically motivated word embeddings in addition to the usual word2vec ones: dependency-based embeddings, and factuality- and certainty-signalling emeddings.

Word2vec Embeddings. We use word embeddings from word2vec which are learned using the skip-gram model of Mikolov et. al (2013a,b) by predicting linear context words surrounding the target words. These word vectors are trained on about 100 billion words from a Google News corpus. As word embeddings alone have shown good performance in various classification tasks, we also use them in isolation, with varying dimensions, in our CNN and LSTM experiments. In the case of CNN, a word embedding size of 300, together with other network parameters, resulted in high accuracy on the claim verifiability data set. In the case of LSTM, word embeddings of size 300 also produced good accuracy on the claim verifiability data set.

Dependency-based Word Embeddings. We use Levy and Goldberg's (2014) dependency-based word embeddings in our claim classification task. These embeddings are learned using dependency-based contexts from an English Wikipedia corpus containing about 175 000 words and over 900 000 distinct syntactic contexts. Dependency-based embeddings are encoded in the input layers of both our CNN and LSTM, as shown in Figure 1. Dependency embeddings of size 100 are concatenated with equally sized word2vec and factual embeddings, resulting in a 300-dimension concatenated embedding vector.

Factuality- and Certainty-signalling Embeddings. We investigate the use of certainty- and factuality-related distributed signals for distinguishing claims. In online argumentative discourse, claims often

	Factual	Feeling	Total
Train	2426	1667	4093
Test	347	239	586
Total	2773	1906	4679

Table 1: Data splits (factual/feeling data set)

	Ver. exp.	Ver. non-exp.	Unver.	Total
Train	900	987	4459	6346
Test	367	370	1687	2424
Total	1267	1357	6146	8770

Table 2: Data splits (verifiability data set)

serve as implicit arguments with inappropriate or missing justification (Park and Cardie, 2014). The certainty and factuality signals present in such claims may be appropriate for determining its factuality or verifiability. As the claims in our data set are objective, subjective and factual types, predicates, adverbs and other modals (related to certainty and factuality) present in FactBank 1.0 may help in better distinguishing various types of claims.

As an example, consider the sentence in Figure 2, a complex claim of type "verifiable non-experiential". The predicate "seems" and the modal verb "must" can be viewed as certainty and factuality information related to the speaker's commitment to their utterance. Factual embeddings of these co-occurrence indicators can help in better identifying the type of the claim. We compile these extra linguistic factual and certainty signals from FactBank (Saurí and Pustejovsky, 2009), a corpus annotated with factuality and certainty indicators very much similar to the word2vec embeddings. These annotations are basically related to certainty, possibility, and probability, with positive and negative polarities. We used the gensim (Řehůřek and Sojka, 2010) word2vec program to compile embeddings from FactBank. We compiled 300-dimensional factual embedding vectors for the words that appear at least five times in FactBank, and for rest of the vocabulary, embedding vectors are assigned uniform distribution in the range of $[-0.25, 0.25]$. In our CNN and LSTM experiments, we integrate factual embeddings (denoted by F above). We also concatenate factual embeddings with other dependency and word embeddings, as shown in Figure 1.

4 Data Sets and Experimental Setup

4.1 Data Sets

Our experiments use the two claim data sets introduced in §1, further details of which are given below.

Factual and Feeling Debate Forum Posts (Walker et al., 2012). This corpus is compiled from the Internet Argument Corpus. It consists of quote–response pairs that are manually annotated according to whether the response is primarily a "factual"- or "feeling"-based argument. In our experiments, we use the training and test splits from Oraby et al. (2015); these consist of claims that can span multiple sentences. The annotation distribution for these splits is shown in Table 1. We also use a development set to tune the hyper-parameters of the model.

Verifiable and Unverifiable User Comments (Park and Cardie, 2014). This corpus consists of 9476 manually annotated sentences and independent clauses from 1047 user comments extracted from the Regulation Room website.[2] Park and Cardie (2014) and Park et al. (2015) used this corpus for examining each proposition with respect to its verifiability to determine the desirable types of support for the analysis of arguments. The propositions are manually annotated with three classes—"verifiable experiential", "verifiable non-experiential", and "unverifiable"—where the support types are evidence, optional evidence, and reason, respectively. The annotation distribution and our train/test splits are shown in Table 2.

4.2 Experimental Setup

We model claim classification as a sentence classification task. We perform binary classification on the factual/feeling data set, and multi-class classification on the verifiability data set. We used Kim's (2014) Theano implementation of CNN for training the CNN model and a variant of the standard Theano implementation[3] for training the LSTM network. We initialized the word2vec, dependency, and factual

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

[3] `http://deeplearning.net/tutorial/lstm.html`

embeddings in both the CNN and LSTM models. Unknown words from the pre-compiled embeddings were initialized randomly in the range $[-0.25, 0.25]$. We updated all three embedding vectors during the training. We also produced a stacked embedding where all three types of embeddings, with dimensionality 100, were concatenated. In the CNN approach, we used a stochastic gradient descent–based optimization method for minimizing the cross entropy loss during the training with the Rectified Linear Unit (ReLU) non-linear activation function. Window filter sizes were set at $[3, 4, 5]$. In the case of LSTM, model was trained using an adaptive learning rate optimizer, ADADELTA (Zeiler, 2012), over shuffled mini-batches with the sigmoid activation function at input, output and forget gates, and the tanh non-linear activation function at cell state.

Tuning Hyper-parameters. We manually explored hyper-parameters such as drop-out (for avoiding over-fitting), and batch size and learning rates (for improving performance) on development sets of both data sets. We performed tuning on the verifiability development data set obtained by splitting the corpus into an 85% training set and a 15% development set. We tuned the hyper-parameters on a 20% development set obtained from Oraby et al. (2015) on the factual vs. feeling data set. We varied batch sizes (12–64), drop-out (0.1–0.6), embedding sizes (50–300), and learning rate (0.0001–0.001) on both data sets and across all embeddings. We obtained the best CNN performance with learning rate decay 0.95, batch size 50, drop-out 0.5, and embedding size 300. For LSTM, we got the best results with learning rate 0.001, drop-out 0.5, and embedding size 300 for both data sets; the optimal batch size was 24 for the verifiability data set but 32 for the factual vs. feeling data set.

SVM Classification on the Factual vs. Feeling Data Set. SVM classifiers find the hyperplane that best discriminates between positive and negative instances (Cristianini and Shawe-Taylor, 2000). We used the SVM classifier SMO (Hall et al., 2009) from the DKPro TC framework (Daxenberger et al., 2014) for factual vs. feeling claims classification. Surface-level top k n-grams are used as features for building the model. We used uni-, bi-, and trigrams, and varied k from 500 to 5000. We obtained the best results with the top 500 n-gram features.

5 Results and Analysis

We compare our methods with several state-of-the-art methods for claim classification, as described in §2. In these tables, the highest accuracy values for precision, recall and F_1 measure are specified in bold font.

Verifiability Data Set. Park and Cardie (2014) and Park et al. (2015) performed claim classification on this data set using SVM and CRF classifiers. The former classifier was found to yield better results. Both approaches employed various lexical and shallow semantic features. The authors also report baseline results using simple unigram features. We considered the SVM-based results[4] a baseline for comparison with ours. The results of our own experiments on the same data set, using CNN and LSTM methods together with the various embeddings mentioned in §3.3, are shown in Table 3. We macro-averaged F_1 across all the classes. Using word embeddings alone in the CNN method, our results (70.47%) were comparable to those of the SVM (68.99%) and exceeded those of the CRF method (63.63%). In a concatenated embeddings setting, CNN achieves 70.34% F_1. The LSTM performance is low when compared to the CNN approach, but comparable to the SVM-based approach. LSTM also performed better than the sequential CRF baseline.

We computed train, validation, and test error rates with respect to the number of epochs during training for the CNN and LSTM approaches. In the case of LSTM, the best classification accuracy is obtained between 5 and 12 epochs, and in case of CNN, at between 5 and 20 epochs. Confusion matrices showing the assignments of our best-performing LSTM and CNN classifiers are shown in Tables 4 and 5, respectively. Both classifiers show a similar pattern of errors. Verifiable experiential and non-experiential claims were not confused as much with each other as they were with unverifiable claims; this may be an artifact of the latter being the majority class. When unverifiable claims were misclassified, they were more

[4]Results are evaluated in a one-vs.-all binary classification setting.

System	Features	Unverifiable			Verifiable non-exp.			Verifiable exp.			Macro
		P	R	F_1	P	R	F_1	P	R	F_1	avg. F_1
Rand.		71.28	69.59	70.42	15.13	15.13	15.13	15.26	15.26	15.26	33.65
SVM	feat.-rich	82.14	89.69	85.75	51.67	37.57	43.51	73.48	62.67	67.65	65.63
SVM	unigram	**86.86**	83.05	84.91	49.88	**55.14**	52.37	66.67	**73.02**	69.70	68.99
CRF		80.35	**93.30**	84.91	**60.34**	28.38	38.60	**74.57**	59.13	65.96	63.63
CNN	word2vec	85.74	88.74	**87.21**	57.19	49.46	53.04	72.07	70.30	71.17	**70.47**
	dep. embed.	86.46	85.95	86.21	55.86	54.05	**54.94**	67.09	71.12	69.05	70.06
	fact. embed.	83.65	87.01	85.30	55.10	43.78	48.79	64.00	65.39	64.69	66.26
	all embed.	85.75	88.14	86.93	54.87	50.27	52.47	67.14	77.38	**71.90**	70.34
LSTM	word2vec	84.86	81.09	82.93	42.66	51.08	46.49	67.21	67.58	67.39	65.60
	dep. embed.	83.31	85.83	84.55	46.09	46.21	46.15	72.38	62.12	66.86	65.85
	fact. embed.	84.50	85.65	85.07	51.12	42.97	46.70	64.02	70.30	67.01	66.26
	all embed.	84.63	82.27	83.44	42.91	54.86	48.16	72.67	61.58	66.67	66.09

Table 3: Classifier performance on the verifiability data set. The SVM and CRF classifiers are those from Park and Cardie (2014); "Rand." is the random baseline.

		Predicted		
		Ver. exp.	Ver. non-exp.	Unver.
Actual	**Ver. exp.**	258	25	84
	Ver. non-exp.	30	159	181
	Unver.	115	127	1445

Table 4: Confusion matrix for LSTM with factual embeddings (verifiability data set)

		Predicted		
		Ver. exp.	Ver. non-exp.	Unver.
Actual	**Ver. exp.**	258	19	90
	Ver. non-exp.	28	183	159
	Unver.	72	118	1497

Table 5: Confusion matrix for CNN with word2vec (verifiability data set)

likely to be labelled as verifiable non-experiential, suggesting that the vocabulary employed in the two classes of claims is similar.

Factual vs. Feeling Claims Data Set. In this data set, claims can span more than one sentence, but we treat these as single sentences for the purposes of our experiments. Oraby et al. (2015) performed unsupervised claim classification on this data set using bootstrapped patterns from both unlabelled and labelled data and report accuracy (F_1) of 41.41%. They also report an F_1 of 64.98% for a naïve Bayes supervised classifier using simple unigram and binary features. The focus of their experiment was to discover more factual- and feeling-related patterns from the unlabelled corpus using a small amount of labelled data. In our experiments, both the CNN (79.56% F_1) and the LSTM-based (75.10% F_1) methods using distributional embeddings show significant improvements over the naïve Bayes and SVM-based approaches as shown in Table 6. CNN achieved good accuracy in all embeddings setting. Sequential LSTM's performance is not better than the CNN approach, but LSTM together with word2vec and factual embeddings performed better on this data set.

Confusion matrices for our best LSTM and CNN classifiers are shown in Tables 7 and 8, respectively. We manually examined those factual claims misclassified as feeling and found that they contained a relatively high proportion of personal pronouns, wh-questions, and negations. While these vocabulary terms are typically associated with feeling claims, they are missing from the factuality embeddings learned from FactBank. By contrast, when feeling claims were misclassified as factual, we found that they tend to contain several distinct propositions or clauses, only one of which was emotional in nature. Properly handling these type of claims would require modelling them with intrapropositional relations.

6 Conclusion and Future Work

In this paper, we presented LSTM- and CNN-based deep neural network methods leverging word2vec and linguistic embeddings, and applied these to argumentative claim classification on two data sets.

On the data set of verifiable and unverifiable claims, our CNN approach using word2vec and concatenated embeddings has shown results comparable to those of a state-of-the-art, feature-rich, SVM-based

System	Features	Factual			Feeling			Macro avg. F_1
		P	R	F_1	P	R	F_1	
Random baseline		59.08	59.08	59.08	40.59	40.59	40.59	49.83
Oraby et al. (2015)	patterns	79.9	40.1	53.4	63.0	19.2	29.4	41.4
Naïve Bayes	unigrams, binary	73.0	67.0	69.8	57.0	65.0	60.7	65.0
SVM	unigrams	76.14	74.86	75.47	64.31	65.81	65.01	70.24
CNN	word2vec	**82.58**	**84.72**	**83.64**	**76.96**	**74.06**	**75.48**	**79.56**
	dep. embed.	78.49	77.81	78.14	68.18	69.04	68.61	73.38
	fact. embed.	76.24	74.93	75.58	64.49	66.12	65.29	70.43
	all embed.	81.98	81.27	81.62	73.14	**74.06**	73.60	77.61
LSTM	word2vec	80.60	77.81	79.18	69.32	72.80	71.02	75.10
	dep. embed.	78.70	76.66	77.66	67.34	69.87	68.58	73.12
	fact. embed.	78.77	81.27	80.00	71.49	68.20	69.81	74.90
	all embed.	77.09	82.42	79.66	71.63	64.43	67.84	73.75

Table 6: Classifier performance on the factual vs. feeling data set.

		Predicted	
		factual	**feeling**
Actual	**factual**	270	77
	feeling	65	174

Table 7: Confusion matrix for LSTM with word2vec (factual vs. feeling data set)

		Predicted	
		factual	**feeling**
Actual	**factual**	294	53
	feeling	62	177

Table 8: Confusion matrix for CNN with word2vec (factual vs. feeling data set)

method. When using an LSTM-based method, the accuracy was somewhat lower, but still better than a CRF. In this case, however, the concatenated embeddings were not any better than the individual ones. On the factual vs. feeling data set, our CNN-based method using word2vec and linguistic embeddings showed good improvements (over 14 percentage points in F_1) over the state-of-the-art Bayes classifier and a 9-point improvement over the SVM baseline, while the LSTM-based method using word2vec and factual embeddings yielded a 10-point improvement over the Bayes classifier and a 5-point improvement over SVM. The LSTM-based method using word2vec and factual embeddings performed better than using other embeddings. We also observed that the performance of sequential LSTM is lower than the CNN but better than the SVM baseline and the sequential CRF method described in prior work.

Our methods are simpler than those described in prior work, and we have demonstrated that they generalize well across claim data sets. Our framework can also be easily adapted to other stacked embeddings to perform various sentence- and document-level classification tasks. In future work, we plan to investigate usage of richer linguistic embeddings, such as factual and word sense embeddings compiled from a larger corpus. We may also consider incorporating inter-proposition predicate relations.

Acknowledgments

This work was funded through the research training group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES, GRK 1994/1) and through the German Research Foundation (DFG).

References

Kevin D. Ashley and Vern R. Walker. 2013. From information retrieval (IR) to argument retrieval (AR) for legal cases: Report on a baseline study. In Kevin D. Ashley, editor, *Legal Knowledge and Information Systems*, volume 259 of *Frontiers in Artificial Intelligence and Applications*, pages 29–38. IOS Press.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. *CoRR*, abs/1606.05464.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Nello Cristianini and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY.

Johannes Daxenberger, Oliver Ferschke, Iryna Gurevych, and Torsten Zesch. 2014. DKPro TC: a Java-based framework for supervised learning experiments on textual data. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 61–66, June.

Sebastian Ebert, Ngoc Thang Vu, and Hinrich Schütze. 2015. A linguistically informed convolutional neural network. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 109–114.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 987–996.

William Ferreira and Andreas Vlachos. 2016. Emergent: A novel data-set for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1163–1168, June.

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

Felix Gers. 2001. *Long Short-term Memory in Recurrent Neural Networks*. Ph.D. thesis, Universität Hannover.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

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

Rie Johnson and Tong Zhang. 2015. Semi-supervised convolutional neural networks for text categorization via region embedding. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 919–927.

Raquel Justo, Thomas Corcoran, Stephanie M. Lukin, Marilyn Walker, and M. Inés Torres. 2014. Extracting relevant knowledge for the detection of sarcasm and nastiness in the social web. *Knowledge-Based Systems*, 69:124–133.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Alexandros Komninos and Suresh Manandhar. 2016. Dependency based embeddings for sentence classification tasks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1490–1500.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2267–2273.

Yann LeCun and Yoshua Bengio. 1995. Convolutional networks for images, speech, and time series. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 255–258. MIT Press, Cambridge, MA.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 302–308.

Omer Levy, Steffen Remus, Chris Biemann, Ido Dagan, and Israel Ramat-Gan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Amita Misra, Pranav Anand, Jean E. Fox Tree, and Marilyn Walker. 2015. Using summarization to discover argument facets in dialog. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 430–440.

Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau, and Chris Reed. 2007. Automatic detection of arguments in legal texts. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law*, pages 225–230.

Shereen Oraby, Lena Reed, Ryan Compton, Ellen Riloff, Marilyn Walker, and Steve Whittaker. 2015. And that's a fact: Distinguishing factual and emotional argumentation in online dialogue. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 116–126.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(4):694–707.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: The detection, classification and structure of arguments in text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 98–107.

Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Proceedings of the 1st Workshop on Argumentation Mining*, pages 29–38.

Joonsuk Park, Arzoo Katiyar, and Bishan Yang. 2015. Conditional random fields for identifying appropriate types of support for propositions in online user comments. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 39–44.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 3, pages 1310–1318.

Chris Reed, Raquel Mochales Palau, Glenn Rowe, and Marie-Francine Moens. 2008. Language resources for studying argument. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, pages 91–100.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*.

Roser Saurí and James Pustejovsky. 2009. FactBank: A corpus annotated with event factuality. *Language Resources and Evaluation*, 43(3):227–268.

Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation for Natural Language Processing*, volume 1, pages 226–234.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 46–56.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for Twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1555–1565.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

Francesco Visin, Kyle Kastner, Kyunghyun Cho, Matteo Matteucci, Aaron C. Courville, and Yoshua Bengio. 2015. ReNet: A recurrent neural network based alternative to convolutional networks. *CoRR*, abs/1505.00393.

Marilyn A. Walker, Jean E. Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012. A corpus for research on deliberation and debate. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 812–817.

Adam Wyner, Raquel Mochales-Palau, Marie-Francine Moens, and David Milward. 2010. Approaches to text mining arguments from legal cases. In Enrico Francesconi, Simonetta Montemagni, Wim Peters, and Daniela Tiscornia, editors, *Semantic Processing of Legal Texts*, volume 6036 of *Lecture Notes in Artificial Intelligence*, pages 60–79. Springer.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. *CoRR*, abs/1212.5701.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 207–212.

Experiments in Idiom Recognition

Jing Peng and **Anna Feldman**
Department of Computer Science
Department of Linguistics
Montclair State University
Montclair, New Jersey, USA 07043

Abstract

Some expressions can be ambiguous between idiomatic and literal interpretations depending on the context they occur in, e.g., *sales **hit the roof*** vs. ***hit the roof** of the car*. We present a novel method of classifying whether a given instance is literal or idiomatic, focusing on verb-noun constructions. We report state-of-the-art results on this task using an approach based on the hypothesis that the distributions of the contexts of the idiomatic phrases will be different from the contexts of the literal usages. We measure contexts by using projections of the words into vector space. For comparison, we implement Fazly et al. (2009)'s, Sporleder and Li (2009)'s, and Li and Sporleder (2010b)'s methods and apply them to our data. We provide experimental results validating the proposed techniques.

1 Introduction

Researchers have been investigating idioms and their properties for many years. According to traditional approaches, an idiom is — in its simplest form— a string of two or more words for which meaning is not derived from the meanings of the individual words comprising that string (Swinney and Cutler, 1979). As such, the meaning of *kick the bucket* ('die') cannot be obtained by breaking down the idiom and analyzing the meanings of its constituent parts, to kick and the bucket. In addition to being influenced by the principle of compositionality, the traditional approaches are also influenced by theories of generative grammar (Flores, 1993; Langlotz, 2006) The properties that traditional approaches attribute to idiomatic expressions are also the properties that make them difficult for generative grammars to describe. For instance, idioms can be syntactically ill-formed (e.g., *by and large*), resistant to grammatical transformations (e.g., *the bucket was kicked by him* $\neq$ 'die'), impervious to lexical substitutions (e.g., *kick the pail* $\neq$ 'die'), and semantically ambiguous without context. This last property of the idioms is what we address in our work. The examples below illustrate the ambiguity[1].

(A1) After the last page was sent to the printer, an editor would **ring a bell**, walk toward the door, and holler " Good night! " (Literal)

(A2) His name never fails to **ring a bell** among local voters. Nearly 40 years ago, Carthan was elected mayor of Tchula... (Idiomatic)

(B1) ...that caused the reactor to literally **blow its top**. About 50 tons of nuclear fuel evaporated in the explosion... (Literal)

(B2) ...He didn't pound the table, he didn't **blow his top**. He always kept his composure. (Idiomatic)

(C1) ...coming out of the fourth turn, slid down the track, **hit** the inside **wall** and then hit the attenuator at the start of pit road. (Literal)

(C2) ...job training, research and more have **hit** a Republican **wall**. (Idiomatic)

Fazly et al. (2009)'s analysis of 60 idioms from the British National Corpus (BNC) has shown that close to half of these also have a clear literal meaning; and of those with a literal meaning, on average

[1]These examples are extracted from the Corpus of Contemporary American English (COCA) (`http://corpus.byu.edu/coca/`)

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2752–2761, Osaka, Japan, December 11-17 2016.

around 40% of their usages are literal.

Just to motivate our work, idioms present great challenges for many Natural Language Processing (NLP) applications. Current machine translation systems (MT), unfortunately, more frequently than not, are not able to translate idiomatic expressions correctly.

Here's an example how the English utterance *He didn't pound the table, he didn't **blow his top**. He always kept his composure.* is translated by Bing and Google Translate from English into Russian and Chinese.

(1) a. English original: He didn't pound the table, he didn't **blow his top**. He always kept his composure.
b. Bing: Chinese: 他没拍几下桌子，他并没有**打击他的上方**。他总是保持镇静。
c. Google: Chinese: 他没有拍桌子，他没有**吹他的上面**。他始终保持着镇定。
d. Bing: Russian: Он не фунт за столом, он не взорвать его сверху. Он всегда держал его спокойствие.
e. Google:Russian: Он не фунт стол, он не взрывал его вершину. Он всегда держал его хладнокровие.

In all the examples above, *blow his top* is translated as 'destruction/explosion of his top/summit', which is clearly not the intended meaning.

In this paper we describe an algorithm for automatic classification of idiomatic and literal expressions. Similar to Peng et al. (2014), we treat idioms as semantic outliers. Our assumption is that the context word distribution for a literal expression will be different from the distribution for an idiomatic one. We capture the distribution in terms of covariance matrix in vector space.

2 Proposed Techniques

We build our work on the following hypotheses:

1. Words representing local topics are likely to associate strongly with a literal expression appearing in that text segment;

2. The context word distribution for a literal expression in word vector space is different from the distribution of an idiomatic one. (This hypothesis is central to the distributional approach to meaning (Firth, 1957; Katz and Giesbrecht, 2006).)

2.1 Projection Based On Local Context Representation

The local context of a literal target verb-noun construction (VNC) must be different from that of an idiomatic one. We propose to exploit recent advances in vector space representation to capture the difference between local contexts (Mikolov et al., 2013a; Mikolov et al., 2013b).

A word can be represented by a vector of fixed dimensionality q that best predicts its surrounding words in a sentence or a document (Mikolov et al., 2013a; Mikolov et al., 2013b). Given such a vector representation, our first proposal is the following. Let v and n be the vectors corresponding to the verb and noun in a target verb-noun construction, as in *blow whistle*, where $v \in \Re^q$ represents *blow* and $n \in \Re^q$ represents *whistle*. Let $\sigma_{vn} = v + n \in \Re^q$. Thus, σ_{vn} is the word vector that represents the composition of verb v and noun n, and in our example, the composition of *blow* and *whistle*. As indicated in Mikolov et al. (2013b), word vectors obtained from deep learning neural net models exhibit linguistic regularities, such as additive compositionality. Therefore, σ_{vn} is justified to predict surrounding words of the composition of, say, *blow* and *whistle* in the literal usage of *blow whistle*. Our hypothesis is that on average, inner product $\sigma_{blowwhistle} \cdot v$, where vs are context words in a literal usage, should be greater than $\sigma_{blowwhistle} \cdot v$, where vs are context words in an idiomatic usage.

Suppose that we have the following sentences: "are you going to *blow the whistle* on the whole lot I mean the university people as well?" and "I *blew the whistle* to start the timed run, and the students ran as hard as they could". *blow the whistle* in the first sentence is idiomatic, while it is literal in the second one. Let v_{blow} be the word vector representing *blow*, and $v_{whistle}$ be the vector representing *whistle*. Thus, in

our notation, we have that $\sigma_{blowwhistle} = v_{blow} + v_{whistle}$. It follows that $p = \sigma_{blowwhistle} \cdot v$ represents the inner product of $\sigma_{blowwhistle}$ and a context word v. The following table shows the inner products of $\sigma_{blowwhistle}$ and context words v in the two sentences, after removing functional words. From the above

Table 1: Inner products with $\sigma_{blowwhistle}$

are	you	going	whole	lot	mean	university	people	well
-0.13	0.28	0.20	-0.03	0.04	0.00	-0.17	-0.14	-0.05
start	timed	run	students	ran	hard	they	could	
0.15	0.12	0.19	-0.22	0.14	0.15	-0.02	0.04	

table, $\sigma_{BlowWhistle}$ has a larger inner product value (0.069) with context words in the literal usage than with context words in the idiomatic usage (0.000), on average.

For a given vocabulary of m words, represented by matrix $V = [v_1, v_2, \cdots, v_m] \in \Re^{q \times m}$, we calculate the projection of each word v_i in the vocabulary onto σ_{vn}

$$P = V^t \sigma_{vn} \tag{1}$$

where $P \in \Re^m$, and t represents transpose. Here we assume that σ_{vn} is normalized to have unit length. Thus, $P_i = v_i^t \sigma_{vn}$ indicates how strongly word vector v_i is associated with σ_{vn}. This projection, or inner product, forms the basis for our proposed technique.

Let $D = \{d_1, d_2, \cdots, d_l\}$ be a set of l text segments (local contexts), each containing a target VNC (i.e., σ_{vn}). Instead of generating a term by document matrix, where each term is $tf \cdot idf$ (product of term frequency and inverse document frequency), we compute a term by document matrix $M_D \in \Re^{m \times l}$, where each term in the matrix is

$$p \cdot idf, \tag{2}$$

the product of the projection of a word onto a target VNC and inverse document frequency. That is, the term frequency (tf) of a word is replaced by the projection (inner product) of the word onto σ_{vn} (1). Note that if segment d_j does not contain word v_i, $M_D(i, j) = 0$, which is similar to $tf \cdot idf$ estimation. The motivation is that topical words are more likely to be well predicted by a literal VNC than by an idiomatic one. The assumption is that a word vector is learned in such a way that it best predicts its surrounding words in a sentence or a document (Mikolov et al., 2013a; Mikolov et al., 2013b). As a result, the words associated with a literal target will have larger projection onto a target σ_{vn}. On the other hand, the projections of words associated with an idiomatic target VNC onto σ_{vn} should have a smaller value. We also propose a variant of $p \cdot idf$ representation. In this representation, each term is a product of p and typical $tf \cdot idf$. That is,

$$p \cdot tf \cdot idf. \tag{3}$$

2.2 Local Context Distributions

Our second hypothesis states that words in a local context of a literal expression will have a different distribution from those in the context of an idiomatic one. We propose to capture local context distributions in terms of scatter matrices in a space spanned by word vectors (Mikolov et al., 2013a; Mikolov et al., 2013b).

Let $d = (w_1, w_2 \cdots, w_k) \in \Re^{q \times k}$ be a segment (document) of k words, where $w_i \in \Re^q$ are represented by a vectors (Mikolov et al., 2013a; Mikolov et al., 2013b). Assuming w_is have been centered without loss of generality, we compute the scatter matrix

$$\Sigma = d^t d, \tag{4}$$

where Σ represents the local context distribution for a given target VNC.

Given two distributions represented by two scatter matrices Σ_1 and Σ_2, a number of measures can be used to compute the distance between Σ_1 and Σ_2, such as Chernoff and Bhattacharyya distances

(Fukunaga, 1990). Both measures require the knowledge of matrix determinant. In our case, this can be problematic, because Σ (4) is most likely to be singular, which would result in a determinant to be zero.

We propose to measure the difference between Σ_1 and Σ_2 using matrix norms. We have experimented with the Frobenius norm and the spectral norm. The Frobenius norm evaluates the difference between Σ_1 and Σ_2 when they act on a standard basis. The spectral norm, on the other hand, evaluates the difference when they act on the direction of maximal variance over the whole space.

3 Experiments

3.1 Methods

We have carried out an empirical study evaluating the performance of the proposed techniques. For comparison, the following methods are evaluated.

1. $tf \cdot idf$: compute term by document matrix from training data with $tf \cdot idf$ weighting.
2. $p \cdot idf$: compute term by document matrix from training data with proposed $p \cdot idf$ weighting (2).
3. $p \cdot tf \cdot idf$: compute term by document matrix from training data with proposed $p \cdot tf \cdot idf$ weighting (3).
4. $CoVAR_{Fro}$: proposed technique (4) described in Section 2.2, the distance between two matrices is computed using Frobenius norm.
5. $CoVAR_{Sp}$: proposed technique similar to $CoVAR_{Fro}$. However, the distance between two matrices is determined using the spectral norm.
6. *Context+ (CTX+)*: supervised version of the CONTEXT technique described in Fazly et al. (2009) (see below).
7. *TextSim*: supervised classification using the Dice coefficient (see below).
8. *GMM*: Gaussian Mixture Model as described in Li and Sporleder (2010b) (see below).

For methods from **1** to **3**, we compute a latent space from a term by document matrix obtained from the training data that captures 80% variance. To classify a test example, we compute cosine similarity between the test example and the training data in the latent space to make a decision.

For methods **4** and **5**, we compute literal and idiomatic scatter matrices from training data (4). For a test example, we compute a scatter matrix according to (4), and calculate the distance between the test scatter matrix and training scatter matrices using the Frobenius norm for method **4**, and the spectral norm for method **5**.

Method **6** corresponds to a supervised version of CONTEXT described in (Fazly et al., 2009). CONTEXT is unsupervised because it does not rely on manually annotated training data, rather it uses knowledge about automatically acquired canonical forms (C-forms). C-forms are fixed forms corresponding to the syntactic patterns in which the idiom normally occurs. Thus, the gold-standard is "noisy" in CONTEXT. Here we provide manually annotated training data. That is, the gold-standard is "clean." Therefore, CONTEXT+ is a supervised version of CONTEXT. We implemented this approach from scratch since we had no access to the code and the tools used in the original article and applied this method to our dataset and the performance results are reported in Table 3.

Method **7** corresponds to a supervised classifier described in Sporleder and Li (2009). In the experiments described in Sporleder and Li (2009), this method achieved the best performance. We used the Dice coeffiient as implemented in Ted Pedersen's Text::Similarity module (`http://www.d.umn.edu/œtpederse/text-similarity.html` to determine the word overlap of a test instance with the literal and non-literal instances in the training set (for the same expression) and then assign the label of the closest set. A similar approach has been described in Katz and Giesbrecht (2006).

Method **8** is based on Li and Sporleder (2010b). Li and Sporleder (2010b) assume that literal and nonliteral data are generated by two different Gaussians. The detection of idiomatic tokens is done by

comparing which Gaussian has the higher probability of generating a specific instance. While the original Li and Sporleder (2010b)'s work uses Normalized Google Distance to model semantic relatedness in computing features (Cilibrasi and Vitányi, 2007; Cilibrasi and Vitányi, 2009), we use inner product between word vectors as described in section 3.3. The main reason is that Google's custom search engine API is no longer free. The detection task is done by a Bayes decision rule, which chooses the category by maximizing the probability of fitting the data into different Gaussian components: $c(x) = \arg\max_{i \in \{l,n\}} \{w_i \times N(x|\mu_i, \Sigma_i)\}$, where c is the category of the Gaussian, μ_i is the mean, Σ_i is the covariance matrix, and w_i is the mixture weight.

Table 2: Datasets: Is = idioms; Ls = literals

Expression	Train	Test
BlowWhistle	20 Is, 20 Ls	7 Is, 31 Ls
LoseHead	15 Is, 15 Ls	6 Is, 4 Ls
MakeScene	15 Is, 15 Ls	15 Is, 5 Ls
TakeHeart	15 Is, 15 Ls	46 Is, 5 Ls
BlowTop	20 Is, 20 Ls	8 Is, 13 Ls
BlowTrumpet	50 Is, 50 Ls	61 Is, 186 Ls
GiveSack	20 Is, 20 Ls	26 Is, 36 Ls
HaveWord	30 Is, 30 Ls	37 Is, 40 Ls
HitRoof	50 Is, 50 Ls	42 is, 68 Ls
HitWall	90 Is, 90 Ls	87 is, 154 Ls
HoldFire	20 Is, 20 Ls	98 Is, 6 Ls
HoldHorse	80 Is, 80 Ls	162 Is, 79 Ls

3.2 Data Preprocessing

We use BNC (Burnard, 2000) and a list of verb-noun constructions (VNCs) extracted from BNC by Fazly et al. (2009) and Cook et al. (2008) and labeled as L (Literal), I (Idioms), or Q (Unknown). The list contains only those VNCs whose frequency was greater than 20 and that occurred at least in one of two idiom dictionaries (Cowie et al., 1983; Seaton and Macaulay, 2002). The dataset consists of 2,984 VNC tokens. For our experiments we only use VNCs that are annotated as I or L. We only experimented with idioms that can have both literal and idiomatic interpretations. We should mention that our approach can be applied to any syntactic construction. We decided to use VNCs only because this dataset was available and for fair comparison – most work on idiom recognition relies on this dataset.

We use the original SGML annotation to extract paragraphs from BNC. Each document contains three paragraphs: a paragraph with a target VNC, the preceding paragraph and following one. Our data is summarized in Table 2.

Since BNC did not contain enough examples, we extracted additional ones from COCA, COHA and GloWbE (http://corpus.byu.edu/). Two human annotators labeled this new dataset for idioms and literals. The inter-annotator agreement was relatively low (Cohen's kappa = .58); therefore, we merged the results keeping only those entries on which the two annotators agreed.

3.3 Word Vectors

For our experiments reported here, we obtained word vectors using the word2vec tool (Mikolov et al., 2013a; Mikolov et al., 2013b) and the text8 corpus. The text8 corpus has more than 17 million words, which can be obtained from `mattmahoney.net/dc/text8.zip`. The resulting vocabulary has 71,290 words, each of which is represented by a $q = 200$ dimension vector. Thus, this 200 dimensional vector space provides a basis for our experiments.

3.4 Datasets

Table 2 describes the datasets we used to evaluate the performance of the proposed technique. All these verb-noun constructions are ambiguous between literal and idiomatic interpretations. The examples below (from the corpora we used) show how these expressions can be used *literally*.

Table 3: Average accuracy of competing methods on 12 datasets

Method	**BlowWhistle**			**LoseHead**			**MakeScene**			**TakeHeart**		
	Pre	Rec	Acc	Pre	Rec	Acc	Pre	Rec	Acc	Pre	Rec	Acc
$tf \cdot idf$	0.23	0.75	0.42	0.27	0.21	0.49	0.41	0.13	0.33	0.65	0.02	0.11
$p \cdot idf$	0.29	0.82	0.60	0.49	0.27	0.48	0.82	0.48	0.53	0.90	0.43	0.44
$p \cdot tf \cdot idf$	0.23	0.99	0.37	0.31	0.30	0.49	0.40	0.11	0.33	0.78	0.11	0.18
$CoVAR_{Fro}$	**0.65**	**0.71**	**0.87**	0.60	0.78	0.58	**0.84**	**0.83**	**0.75**	0.95	0.61	0.62
$CoVAR_{sp}$	0.44	0.77	0.77	**0.62**	**0.81**	**0.61**	0.80	0.82	0.72	0.94	0.55	0.56
$CTX+$	0.17	0.56	0.40	0.55	0.52	0.46	0.78	0.37	0.45	**0.92**	**0.66**	**0.64**
$TextSim$	0.20	0.71	0.41	0.62	0.62	0.55	0.73	0.37	0.43	0.91	0.54	0.54
GMM	0.18	0.55	0.46	0.46	0.48	0.50	0.67	0.54	0.52	0.79	0.36	0.39
	BlowTop			**BlowTrumpet**			**GiveSack**			**HaveWord**		
	Pre	Rec	Acc	Pre	Rec	Acc	Pre	Rec	Acc	Pre	Rec	Acc
$tf \cdot idf$	0.55	0.93	0.65	0.26	0.85	0.36	0.61	0.63	0.55	0.52	0.33	0.52
$p \cdot idf$	0.59	0.58	0.68	0.44	0.85	0.69	0.55	0.47	0.62	0.52	0.53	0.54
$p \cdot tf \cdot idf$	0.54	0.53	0.65	0.33	0.93	0.51	0.54	0.64	0.55	0.53	0.53	0.53
$CoVAR_{Fro}$	**0.81**	**0.87**	**0.86**	0.45	0.94	0.70	0.63	0.88	0.72	0.58	0.49	0.58
$CoVAR_{sp}$	0.71	0.79	0.79	0.39	0.89	0.62	0.66	0.75	0.73	0.56	0.53	0.58
$CTX+$	0.66	0.70	0.75	**0.59**	**0.81**	**0.81**	0.67	0.83	0.76	0.53	0.85	0.57
$TextSim$	0.70	0.69	0.77	0.56	0.83	0.80	**0.68**	**0.83**	**0.77**	**0.54**	**0.85**	**0.58**
GMM	0.41	0.49	0.49	0.25	0.68	0.43	0.45	0.47	0.53	0.42	0.41	0.49
	HitRoof			**HitWall**			**HoldFire**			**HoldHorse**		
	Pre	Rec	Acc	Pre	Rec	Acc	Pre	Rec	Acc	Pre	Rec	Acc
$tf \cdot idf$	0.42	0.70	0.52	0.37	0.99	0.39	0.91	0.57	0.57	0.79	0.98	0.80
$p \cdot idf$	0.54	0.84	0.66	0.55	0.92	0.70	0.97	0.83	0.81	0.86	0.81	0.78
$p \cdot tf \cdot idf$	0.41	0.98	0.45	0.39	0.97	0.43	**0.95**	**0.89**	**0.85**	0.84	0.97	0.86
$CoVAR_{Fro}$	**0.61**	**0.88**	**0.74**	**0.59**	**0.94**	**0.74**	0.97	0.86	0.84	**0.86**	**0.97**	**0.87**
$CoVAR_{sp}$	0.54	0.85	0.66	0.50	0.95	0.64	0.96	0.87	0.84	0.77	0.85	0.73
$CTX+$	0.55	0.82	0.67	0.92	0.57	0.71	0.97	0.64	0.64	**0.93**	**0.89**	**0.88**
$TextSim$	0.56	0.83	0.69	0.92	0.56	0.70	0.97	0.66	0.66	0.93	0.88	0.88
GMM	0.40	0.55	0.51	0.41	0.73	0.53	0.94	0.72	0.70	0.73	0.57	0.57

BlowWhistle: *we can immediately turn towards a high-pitched sound such as whistle being blown. The ability to accurately locate a noise* $\cdots$ **LoseHead**: *This looks as eye-like to the predator as the real eye and gives the prey a fifty-fifty chance of losing its head. That was a very nice bull I shot, but I lost his head.* **MakeScene**: $\cdots$ *in which the many episodes of life were originally isolated and there was no relationship between the parts, but at last we must make a unified scene of our whole life.* **TakeHeart**: $\cdots$ *cutting off one of the forelegs at the shoulder so the heart can be taken out still pumping and offered to the god on a plate.* **BlowTop**: *Yellowstone has no large sources of water to create the amount of steam to blow its top as in previous eruptions.*

4 Results

Table 3 shows the average precision, recall and accuracy of the competing methods on 12 datasets over 20 runs. Table 4 shows the performance of the models by class. The best performance is in bold face. The best model is identified by considering precision, recall, and accuracy together for each model. We calculate accuracy by summing true positives and true negatives and normalizing the sum by the number of examples. Figure 1 shows the aggregated performance in terms of precision, recall and accuracy by the eight competing methods on the 12 data sets. The results show that the $CoVAR$ model outperforms the rest of the models overall and on individual classes.

Table 4: Performance results by class: I denotes the idiom class and L denotes the literal class.

Method	$tf \cdot idf$		$p \cdot idf$		$p \cdot tf \cdot idf$		$CoVar_{Fro}$		$CoVar_{SP}$		$CTX+$		$TextSim$		GMM	
	Is	Ls	Is	Ls	Is	Ls	Is	Ls	Is	Ls	Is	Ls	Is	Ls	Is	Ls
BlowWhistle	0.75	0.35	0.82	0.55	0.99	0.23	0.71	0.90	0.77	0.76	0.56	0.37	0.71	0.34	0.55	0.44
LoseHead	0.21	0.92	0.27	0.80	0.30	0.79	0.78	0.27	0.81	0.30	0.52	0.36	0.62	0.43	0.48	0.53
MakeScene	0.13	0.92	0.48	0.70	0.11	0.97	0.83	0.51	0.82	0.40	0.37	0.68	0.37	0.59	0.54	0.46
TakeHeart	0.02	0.93	0.43	0.56	0.11	0.80	0.61	0.69	0.55	0.62	0.66	0.42	0.54	0.50	0.36	0.67
BlowTop	0.93	0.48	0.58	0.74	0.53	0.72	0.87	0.86	0.79	0.79	0.70	0.77	0.69	0.82	0.49	0.49
BlowTrumpet	0.85	0.20	0.85	0.64	0.93	0.38	0.94	0.62	0.89	0.54	0.81	0.81	0.83	0.79	0.68	0.35
GiveSack	0.63	0.49	0.47	0.72	0.64	0.49	0.88	0.61	0.75	0.71	0.83	0.71	0.83	0.72	0.47	0.57
HaveWord	0.33	0.70	0.53	0.56	0.53	0.54	0.49	0.66	0.53	0.62	0.85	0.31	0.85	0.32	0.41	0.56
HitRoof	0.70	0.40	0.84	0.56	0.98	0.12	0.88	0.65	0.85	0.53	0.82	0.58	0.83	0.60	0.55	0.49
HitWall	0.99	0.05	0.92	0.57	0.97	0.12	0.94	0.63	0.95	0.47	0.57	0.78	0.56	0.92	0.73	0.42
HoldFire	0.57	0.46	0.83	0.50	0.89	0.26	0.86	0.54	0.87	0.48	0.64	0.66	0.66	0.72	0.72	0.37
HoldHorse	0.98	0.45	0.81	0.72	0.97	0.63	0.97	0.67	0.85	0.49	0.89	0.86	0.88	0.87	0.57	0.57
Average	0.59	0.53	0.65	0.64	0.66	0.50	**0.81**	**0.64**	0.79	0.56	0.68	0.61	0.70	0.64	0.55	0.49

Interestingly, the Frobenius norm outperforms the spectral norm. One possible explanation is that the spectral norm evaluates the difference when two matrices act on the maximal variance direction, while the Frobenius norm evaluates on a standard basis. That is, Frobenius measures the difference along all basis vectors. On the other hand, the spectral norm evaluates changes in a particular direction. When the difference is a result of all basis directions, the Frobenius norm potentially provides a better measurement. The projection methods ($p \cdot idf$ and $p \cdot tf \cdot idf$) outperform $tf \cdot idf$ overall but not as pronounced as $CoVAR$.

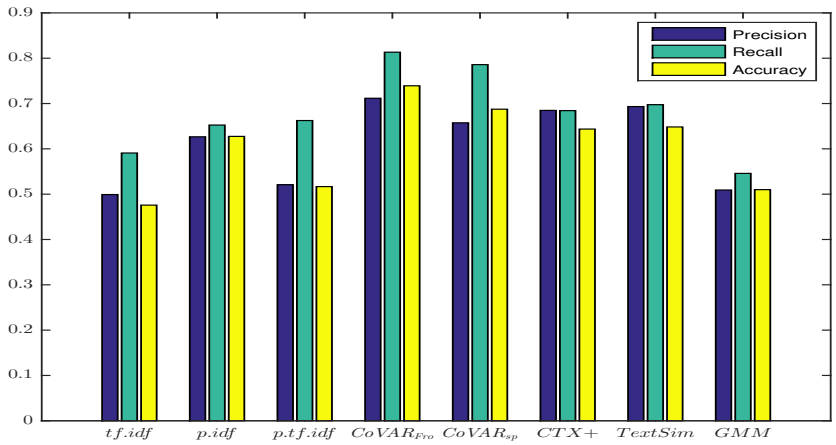


Figure 1: Aggregated performance by the eight competing methods on the 12 data sets.

Finally, we have noticed that even the best model ($CoVAR_{Fro}$) does not perform as well on certain idiomatic expressions. We hypothesize that the model works the best on highly idiomatic expressions. to be more easily interpretable than others.

We decided to conduct a small experiment, in which we asked two human annotators to rank VNCs in our datasets, i.e., rank each VNC token as "highly idiomatic" to "easily interpretable/compositional" on a scale of 5 to 1 (5: highly idiomatic; 1: low idiomaticity) given the context. We averaged the results in Table 5. This task is highly subjective and having two annotators is merely enough to make strong claims. The agreement was very low (30%), because the annotators often disagreed on idiomaticity

scores, such as 2 vs. 3. The annotators tried to avoid ranking the expressions as 100% idiomatic or 100% literal. Measuring the agreement using ranges is reasonable. Thus, if both annotators marked an idiom as 1 or 2, we considered them to be in agreement. The ranges were 1-2, 2-3, 3-4 and 4-5. Applying this method, the annotator agreement increased significantly – 80% (Cohen's Kappa 0.68).

The table shows that the low ranking scores often correspond to the low performance scores of our best model: the model did not perform well on *HaveWord* and the idiomaticity score produced by the human annotators is relatively low (=2). Low idiomaticity suggests indeterminate contexts, which affects the performance of our context-based models. There is a positive correlation between the degree of idiomaticity and the accuracy of the best model ($r = .47$, $p =< .001$).

Table 5: Idiomaticity Rank: 1=low; 5 = high

VNC	HitWall	GiveSack	HaveWord	LoseHead	MakeScene	BlowTop	BlowWhistle	HoldFire	HoldHorse	HitRoof	TakeHeart
Rank	1.5	2	2	2	2.5	3	3	3.5	3.5	4	4

5 Related Work

Previous approaches to idiom detection can be classified into two groups: 1) type-based extraction, i.e., detecting idioms at the type level; 2) token-based detection, i.e., detecting idioms in context. Type-based extraction is based on the idea that idiomatic expressions exhibit certain linguistic properties such as non-compositionality that can distinguish them from literal expressions (Sag et al., 2002; Fazly et al., 2009). While many idioms do have these properties, all idioms fall on the continuum from being compositional to being partly unanalyzable to completely non-compositional (Cook et al., 2007). Katz and Giesbrecht (2006), Birke and Sarkar (2006), Fazly et al. (2009), Li and Sporleder (2009), Li and Sporleder (2010a), Sporleder and Li (2009), and Li and Sporleder (2010b), among others, notice that type-based approaches do not work on expressions that can be interpreted idiomatically or literally depending on the context and thus, an approach that considers tokens in context is more appropriate for idiom recognition. To address these problems, Peng et al. (2014) investigate the bag of words *topic* representation and incorporate an additional hypothesis–contexts in which idioms occur are more affective. Still, they treat idioms as semantic outliers.

6 Conclusions

In this paper we described an original algorithm for automatic classification of idiomatic and literal expressions. We also compared our algorithms against several competing idiom detection algorithms in the literature. The performance results show that our algorithm generally outperforms Fazly et al. (2009)'s, Sporleder and Li (2009), and Li and Sporleder (2010b)'s models (see Table 4). In particular, our method is especially effective when idioms are highly idiomatic. A research direction is to incorprate affect into our model. Idioms are typically used to imply a certain evaluation or affective stance toward the things they denote (Nunberg et al., 1994; Sag et al., 2002). We usually do not use idioms to describe neutral situations, such as buying tickets or reading a book. Even though our method was tested on verb-noun constructions, it is independent of syntactic structure and can be applied to any idiom type. Unlike Fazly et al. (2009)'s approach, for example, our algorithm is language-independent and does not rely on POS taggers and syntactic parsers, which are often unavailable for resource-poor languages. Our next step is to expand this method and use it for idiom detection rather than for idiom classification.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1319846.

References

Julia Birke and Anoop Sarkar. 2006. A clustering approach to the nearly unsupervised recognition of nonliteral language. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'06)*, pages 329–226, Trento, Italy.

Lou Burnard, 2000. *The British National Corpus Users Reference Guide*. Oxford University Computing Services.

Rudi Cilibrasi and Paul M. B. Vitányi. 2007. The google similarity distance. *IEEE Trans. Knowl. Data Eng.*, 19(3):370–383.

Rudi Cilibrasi and Paul M. B. Vitányi. 2009. Normalized web distance and word similarity. *CoRR*, abs/0905.4039.

Paul Cook, Afsaneh Fazly, and Suzanne Stevenson. 2007. Pulling their weight: Exploiting syntactic forms for the automatic identification of idiomatic expressions in context. In *Proceedings of the ACL 07 Workshop on A Broader Perspective on Multiword Expressions*, pages 41–48.

Paul Cook, Afsaneh Fazly, and Suzanne Stevenson. 2008. The VNC-Tokens Dataset. In *Proceedings of the LREC Workshop: Towards a Shared Task for Multiword Expressions (MWE 2008)*, Marrakech, Morocco, June.

Anthony P. Cowie, Ronald Mackin, and Isabel R. McCaig. 1983. *Oxford Dictionary of Current Idiomatic English*, volume 2. Oxford University Press.

Afsaneh Fazly, Paul Cook, and Suzanne Stevenson. 2009. Unsupervised Type and Token Identification of Idiomatic Expressions. *Computational Linguistics*, 35(1):61–103.

John R. Firth. 1957. A synopsis of linguistic theory, 1930–1955. In *Studies in Linguistic Analysis*, Special volume of the Philological Society, pages 1–32. Blackwell, Oxford.

d'Arcais Flores. 1993. The comprehension and semantic interpretation of idioms. *Idioms: Processing, structure, and interpretation*, pages 79–98.

K. Fukunaga. 1990. *Introduction to statistical pattern recognition*. Academic Press.

Graham Katz and Eugenie Giesbrecht. 2006. Automatic Identification of Non-compositional Multiword Expressions using Latent Semantic Analysis. In *Proceedings of the ACL/COLING-06 Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 12–19.

Andreas Langlotz. 2006. *Idiomatic creativity: a cognitive-linguistic model of idiom-representation and idiom-variation in English*, volume 17. John Benjamins Publishing.

Linlin Li and Caroline Sporleder. 2009. A cohesion graph based approach for unsupervised recognition of literal and non-literal use of multiword expresssions. In *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing (ACL-IJCNLP)*, pages 75–83, Singapore.

Linlin Li and Caroline Sporleder. 2010a. Linguistic cues for distinguishing literal and non-literal usages. In *COLING (Posters)*, pages 683–691.

Linlin Li and Caroline Sporleder. 2010b. Using gaussian mixture models to detect figurative language in context. In *Proceedings of NAACL/HLT 2010*.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*.

Geoffrey Nunberg, Ivan A. Sag, and Thomas Wasow. 1994. Idioms. *Language*, 70(3):491–538.

Jing Peng, Anna Feldman, and Ekaterina Vylomova. 2014. Classifying idiomatic and literal expressions using topic models and intensity of emotions. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2019–2027, Doha, Qatar, October. Association for Computational Linguistics.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A Pain in the Neck for NLP. In *Proceedings of the 3rd International Conference on Intelligence Text Processing and Computational Linguistics (CICLing 2002)*, pages 1–15, Mexico City, Mexico.

Maggie Seaton and Alison Macaulay, editors. 2002. *Collins COBUILD Idioms Dictionary*. HarperCollins Publishers, second edition.

Caroline Sporleder and Linlin Li. 2009. Unsupervised Recognition of Literal and Non-literal Use of Idiomatic Expressions. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 754–762, Morristown, NJ, USA. Association for Computational Linguistics.

David A Swinney and Anne Cutler. 1979. The access and processing of idiomatic expressions. *Journal of verbal learning and verbal behavior*, 18(5):523–534.

An Empirical Evaluation of various Deep Learning Architectures for Bi-Sequence Classification Tasks

Anirban Laha
IBM Research India
anirlaha@in.ibm.com

Vikas Raykar
IBM Research India
viraykar@in.ibm.com

Abstract

Several tasks in argumentation mining and debating, question-answering, and natural language inference involve classifying a sequence in the context of another sequence (referred as bi-sequence classification). For several single sequence classification tasks, the current state-of-the-art approaches are based on recurrent and convolutional neural networks. On the other hand, for bi-sequence classification problems, there is not much understanding as to the best deep learning architecture. In this paper, we attempt to get an understanding of this category of problems by extensive empirical evaluation of 19 different deep learning architectures (specifically on different ways of handling context) for various problems originating in natural language processing like debating, textual entailment and question-answering. Following the empirical evaluation, we offer our insights and conclusions regarding the architectures we have considered. We also establish the first deep learning baselines for three argumentation mining tasks.

1 Introduction

Argumentation mining is a relatively new challenge in corpus-based discourse analysis that involves automatically identifying argumentative structures within a corpus. Many tasks in argumentation mining (Lippi and Torroni, 2015a) and debating technologies (Slonim et al., 2014) involve categorizing a sequence in the context of another sequence. For example, in *context dependent claim detection* (Levy et al., 2014), given a sentence, one task is to identify whether the sentence contains a claim relevant to a particular debatable topic (generally given as a context sentence). Similarly in *context dependent evidence detection* (Rinott et al., 2015), given a sequence (possibly multiple sentences), one task is to detect if the sequence contains an evidence relevant to a particular topic. We refer to such class of problems in computational argumentation as *bi-sequence classification* problems—given two sequences s and c we want to predict the label for the target sequence s in the context of another sequence c[1]. Apart from the debating tasks, several other natural language inference tasks fall under the same paradigm of having a pair of sequences. For example, *recognizing textual entailment* (Bowman et al., 2015), where the task is to predict if the meaning of a sentence can be inferred from the meaning of another sentence. Another class of problems originated from question-answering systems also known as *answer selection*, where given a question, a candidate answer needs to be classified as an answer to the question at hand or not.

Recently, deep learning approaches have obtained very high performance across many different natural language processing tasks. These models can often be trained in an end-to-end fashion and do not require traditional, task-specific feature engineering. For many single sequence classification tasks, the state-of-the-art approaches are based on recurrent neural networks (RNN variants like Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014)) and convolution neural network based models (CNN) (Kim, 2014). Whereas for bi-sequence classification, the context sentence c has to be explicitly taken into account when performing the classification for the

[1] In this paper, we shall ignore the subtle distinction between sentence and sequence and both will mean just a text segment composed of words.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2762–2773, Osaka, Japan, December 11-17 2016.

target sentence s. The context can be incorporated into the RNN and CNN based models in various ways. However there is not much understanding in current literature as to the best way to handle context in these deep learning based models. In this paper, we empirically evaluate(see Section 4) the performance of five different ways of handling context in conjunction with target sentence(see Section 3) for multiple bi-sequence classification tasks(see Section 2) using architectures composed of RNNs and(/or) CNNs.

In a nutshell, this paper makes the two major novel contributions:

1. We establish the first deep learning based baselines for three bi-sequence classification tasks relevant to argumentation mining with zero feature engineering.

2. We empirically compare the performance of several ways handling context for bi-sequence classification problems in RNN and CNN based models. While some of these variants are used in various other tasks, there has been no formal comparison of different variants and this is the first attempt to actually list all the variants and compare them on several publicly available benchmark datasets.

2 Bi-Sequence classification tasks

In this section, we will briefly mention the various bi-sequence tasks of interest in the literature of argumentation mining and in the broader natural language inference domain.

2.1 Argumentation Mining

We mainly consider two prominent tasks in argumentation mining, namely, detecting the claims (Levy et al., 2014) and evidences (Rinott et al., 2015), within a given corpus, which are related to a prespecified topic. These two tasks together helps to automatically construct persuasive arguments out of a given corpora. We will define the following four concepts:
Motion - The topic under debate, typically a short phrase that frames the discussion.
Claim - A general, typically concise statement that directly supports or contests the motion.
Motion text - A document/article/discourse that contain claims with high probability.
Evidence - A set of statements that directly supports the claim for a given motion.

2.1.1 Context Dependent Claim Detection (CDCD)

Given a sentence in a motion text the task is to identify whether the sentence contains a claim relevant to the motion or not. This is the claim sentence task introduced by Levy et al. (2014). For example, each of the following sentences includes a claim, marked in italic, for the motion topic in brackets.

1. (**the sale of violent video games to minors**) Recent research has suggested that some *violent video games may actually have a pro-social effect in some contexts, for example, team play*.

2. (**the right to bear arms**) Some gun control organizations say that *increased gun ownership leads to higher levels of crime, suicide and other negative outcomes*.

2.1.2 Context Dependent Evidence Detection (CDED)

Given a segment in a motion text the task is to identify whether the segment contains an evidence relevant to the motion or not (Rinott et al., 2015). We consider evidences of two types in this paper, *Study* and *Expert*. Evidences of type study are generally results of a quantitative analysis of data given as numbers, or as conclusions. The following are two examples for study evidence relevant to the motion topic in brackets.

- (**the sale of violent video games to minors**) A 2001 study found that exposure to violent video games causes at least a temporary increase in aggression and that this exposure correlates with aggression in the real world.

- (**the right to bear arms**) In the South region where there is the highest number of legal guns per citizen only 59% of all murders were caused by firearms in contrast to 70% in the Northeast where there is the lowest number of legal firearms per citizen.

Evidence of type expert is a testimony by a person/group/commitee/organization with some known expertise/authority on the topic. The following are two examples for expert evidence relevant to the motion topic in brackets.

1. (**the sale of violent video games to minors**) This was also the conclusion of a meta-analysis by psychologist Jonathan Freedman, who reviewed over 200 published studies and found that the majority did not find a causal link.

2. (**the right to bear arms**) University of Chicago economist Steven Levitt argues that available data indicate that neither stricter gun control laws nor more liberal concealed carry laws have had any significant effect on the decline in crime in the 1990s.

2.2 Textual Entailment (TE)

This task (Bowman et al., 2015) corresponds to a multiclass setting, where given a pair of sentences (*premise* and *hypothesis*), the task is to identify whether one of them (*premise*) entails, contradicts or is neutral with respect to the other sentence (*hypothesis*). Unlike the other debating tasks seen previously, we cannot call these pair of sentences as context and target as these are more symmetric in nature. Typical examples[2] are the following (premise followed by hypothesis):

- Entailment: A soccer game with multiple males playing - Some men are playing a sport.
- Contradiction: A black race car starts up in front of a crowd of people - A man is driving down a lonely road.
- Neutral: A smiling costumed woman is holding an umbrella - A happy woman in a fairy costume holds an umbrella.

2.3 Answer Selection for Questions

Question Answering (QA) System is a natural extension to the traditional commercial search engines as it is concerned with fetching answers to natural language queries and returning the information accurately in natural human language. A QA system can be either closed-domain or open-domain, the former being restricted to a particular domain while the latter is not. *Answer sentence selection is a crucial subtask of the open-domain question answering problem, with the goal of extracting answers from a set of pre-selected sentences* (Yang et al., 2015). This is again bi-sequence classification task where the pair of sequences being a question and a candidate answer to be selected.

3 Deep Learning models for sequence pairs

All the tasks described in the previous section can be formulated as bi-sequence classification problems where we have to predict the label for the given pair of sequences. For simpler single sequence text classification tasks, RNN or CNN based architectures have become standard baselines. In this section, we will briefly introduce RNN and CNN and then subsequently describe RNN and CNN based architectures for bi-sequence classification tasks. Specifically, we talk about five different ways of handling context along with the target sentence.

3.1 Continuous Bag of Words (CBOW)

One of the simplest forms of sequence representation is the CBOW model, where every word in the sequence produces some word embedding (say, based on word2vec (Mikolov et al., 2013)) and the average of the word embedding vectors over the words produces the representation of the sequence. As is evident, this form of representation totally disregards the word order of the sequence.

3.2 Recurrent neural networks (RNNs)

The RNN model provides a framework for conditioning on the entire history of the sequence without resorting to the Markov assumption traditionally used for modelling sequences. Unlike CBOW, RNNs encode arbitrary length sequences as fixed size vectors without disregarding the word order.

Given an ordered list of n input vectors $\mathbf{x}_1, ..., \mathbf{x}_n$ and an initial state vector $\mathbf{s}_0$, a RNN generates an ordered list of n state vectors $\mathbf{s}_0, ..., \mathbf{s}_n$ and an ordered list of n output vectors $\mathbf{y}_1, ..., \mathbf{y}_n$, that is, $RNN(\mathbf{s}_0, \mathbf{x}_1, ..., \mathbf{x}_n) = \mathbf{s}_1, ..., \mathbf{s}_n, \mathbf{y}_1, ..., \mathbf{y}_n$. The input vectors $\mathbf{x}_i$ (which corresponds to a fixed dimensional representation for each word in the sequence) are presented to the RNN in a sequential fashion and $\mathbf{s}_i$ represents the state of the RNN after observing the inputs $\mathbf{x}_1, ..., \mathbf{x}_i$. The output vector $\mathbf{y}_i$ is a

[2] http://nlp.stanford.edu/projects/snli/

function of the corresponding state vector $\mathbf{s}_i$ and is then used for further prediction. An RNN is given by the following update equations:

$$\mathbf{s}_i = R(\mathbf{x}_i, \mathbf{s}_{i-1}) \quad (1)$$

$$\mathbf{y}_i = O(\mathbf{s}_i) \quad (2)$$

The recursively defined function R takes as input the previous state vector $\mathbf{s}_{i-1}$ and the current input vector $\mathbf{x}_i \in \mathbb{R}^{d_x}$ and results in an updated state vector $\mathbf{s}_i \in \mathbb{R}^{d_s}$. An additional function O maps the state vector $\mathbf{s}_i$ to an output vector $\mathbf{y}_i \in \mathbb{R}^{d_y}$. Different instantiations of R and O will result in the different network structures (Simple RNN, LSTM (Hochreiter and Schmidhuber, 1997), GRU (Cho et al., 2014), etc.). The final state vector $\mathbf{s}_n$ can be thought of as encoding the entire input sequence into a fixed size vector, which can be passed to a softmax layer to produce class probabilities.

3.3 Convolutional Neural Networks (CNNs)

CNNs are built on the premise of locality and parameter sharing which has proven to produce very effective feature representation for images. Following the groundbreaking work by Kim (2014), there has been a lot of interest shown by the text community towards applying CNNs for modelling text representation.

As in case of RNNs defined above, an n-word sentence consists of embedding vectors $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^{d_x}$, one for each word in the sentence. Let $\mathbf{x}_{i:i+j}$ denote the concatenation of words $\mathbf{x}_i, \mathbf{x}_{i+1}, ..., \mathbf{x}_{i+j}$. A convolution operation defined by a non-linear function f applies a filter $\mathbf{w} \in \mathbb{R}^{hd_x}$ to a window of h words to produce a single feature value as given below:

$$c_i = f(\mathbf{w}.\mathbf{x}_{i:i+h-1} + b) \quad (3)$$

$$\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}] \quad (4)$$

In the next step, max-pooling is applied which essentially produces a single feature value $\hat{c} = max\{\mathbf{c}\}$, corresponding to one filter that has been used. The model can have multiple feature values, one for each applied filter, thus producing a feature representation for the input sentence, which can again be passed to a softmax layer to produce class probabilities.

3.4 Bi-Sequence RNN models

For bi-sequence classification tasks we use two RNNs, one RNN to encode the context sentence (*context RNN*) and another separate RNN encode the target sentence (*target RNN*). We define the following five different variants of combining these two RNNs for bi-sequence classification tasks (see Figure 1 for illustration of these variants).

1. **conditional-state**: The final state of the context RNN is fed as the initial state of the target RNN. This way of handling context for RNNs has been previously used in conversational systems (Vinyals and Le, 2015), image description (Vinyals et al., 2015) and image question answering (Ren et al., 2015) systems.

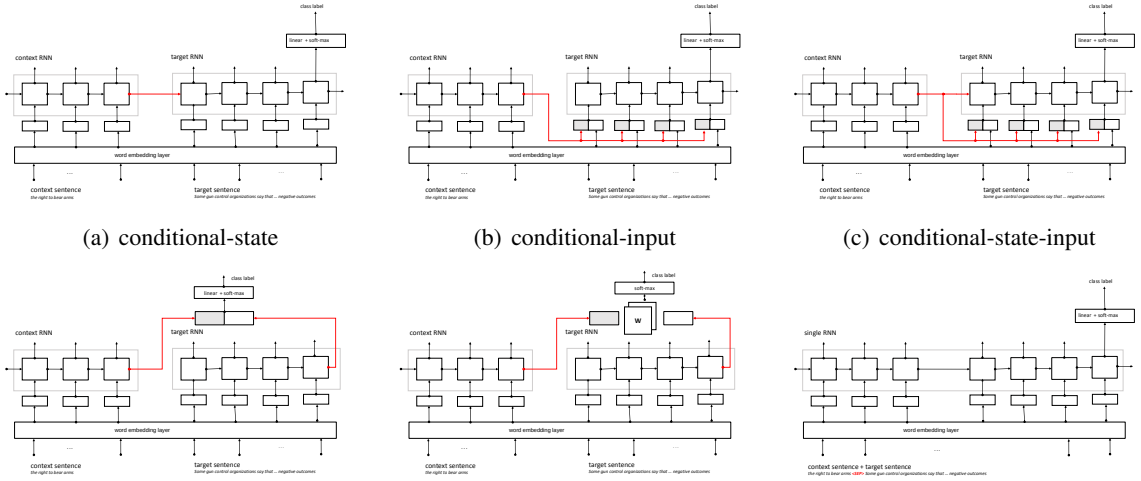
2. **conditional-input**: The final state of the context RNN is fed as auxiliary input (concatenated with every input) for the target RNN. This way of handling context has been previously used in machine translation tasks (Sutskever et al., 2014).

3. **conditional-state-input**: The final state of the context RNN is fed as the initial state of the target RNN and also fed as input for target RNN concatenated with every input.

4. **concat**: The final states of both the context and the target RNN are concatenated and then fed to a softmax layer for the label prediction.

5. **bi-linear**: The final states of both the context and the target RNN are combined using a bi-linear form ($\mathbf{x}^\top \mathbf{W}\mathbf{y}$) with a softmax function for the final prediction. There are different $\mathbf{W}$ for different classes under consideration.

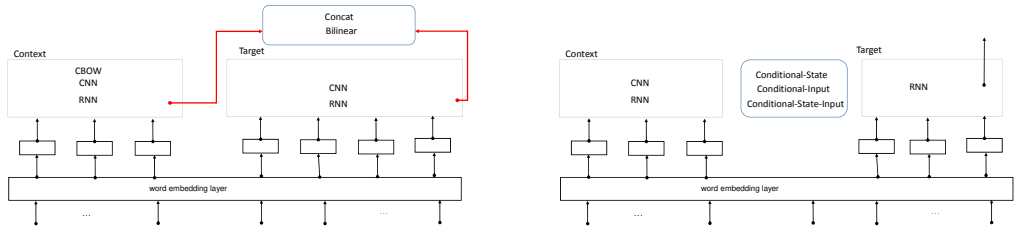
From here on, we would refer to architecture types 1, 2 and 3 as *conditional* variants while the others will be addressed as is. In addition, we consider another baseline variant **concat-sentence**, in which we concatenate the context and the sentence with a special separator token and feed the entire concatenated sequence to a single RNN. For all these variants we use a common embedding layer. Also note that the conditional variants require a common RNN size for both the context and the target RNNs. Even though that restriction is not there for other variants, we choose the same RNN size anyways for convenience.



(a) conditional-state (b) conditional-input (c) conditional-state-input

(d) concat (e) bilinear (f) concat-sentence

Figure 1: RNN based Architectures for bi-sequence classification.



(a) concat-bilinear-all-combinations (b) conditional-all-combinations

Figure 2: Multiple model variants for bi-sequence classification.

3.5 Bi-Sequence model variants

In this paper, we consider multiple ways of extending the bi-sequence architectures mentioned in section 3.4, by replacing RNN with CBOW or CNN either for context or target or both. For the variations *concat* and *bi-linear* (see Fig. 2(a)), we have considered CBOW, RNN and CNN for context representation whereas we have RNN and CNN for target representation. In tasks where context has very few words (say debating tasks or question-answering), a simple representation like CBOW may work for context. However, we haven't considered modelling target using CBOW as targets are usually of larger length. For the *conditional* variants (see Fig. 2(b)), we haven't considered CBOW due to their limited modelling capacity and there is no softmax layer directly on top of context representation to compensate for it (even though softmax is only on top of target RNN). Moreover, target can only be RNN as there is no concept of hidden state for CNNs. Hence, we use CNN and RNN for context whereas RNN for target. In addition, we consider the concat-sentence (mentioned in section 3.4) as a baseline. This leads to 19 architectures for empirical comparison (12 from Fig. 2(a) and 6 from Fig. 2(b) and the baseline).

4 Experiments

We have carried out extensive evaluation of the above architecture variants over a wide range of datasets related to argumentation mining as well as datasets appealing to the larger natural language community

like textual entailment and question answering data. We consider data with class imbalance problem as well as balanced and we do not restrict ourselves to binary classification by working with multiclass dataset as well. As can be found in the Table 1, we consider the following tasks related to the domain of argumentation mining(Aharoni et al., 2014), which is available here [3] :

- Claim Sentence : This is the dataset for the CDCD task defined in section 2.1.1. This is the current benchmark dataset for the Claim Detection task. There are a total 47183 canditate claims distributed among 33 motions.
- EXPERT Evidence : This is corresponding to the CDED task defined in section 2.1.2 for evidence type EXPERT. There are 56985 labelled candidates for 57 different motion topics.
- STUDY Evidence : For evidence type STUDY, the dataset consists of 33534 labelled candidates for 49 motion topics.

Table 1 summarizes all of the datasets above. Interesting point to be noted here is that all the datasets above have very low number of positives and the architectures we are evaluating need to be resilient to the class imbalance problem for these datasets. Other than the debating datasets listed above, we also consider two datasets related to more popular problems in the natural language processing community:

- Textual Entailment (TE) [4] (Bowman et al., 2015) :This dataset consist of around 500K instances evenly distributed across all three classes. So, here we have a multiclass problem in a balanced setting.
- WikiQA [5] (Yang et al., 2015) : There are around 29K labelled question/answer pairs at our disposal.

Task	Motions	Data Size	Positives
Claim Sentence	33	47183	2.77%
EXPERT Evidence	57	56985	4.56%
STUDY Evidence	49	33534	3.74%

Table 1: Argument Mining Datasets.

Task	Train	Dev	Test	Problem	Class
TE	549367	9842	9824	Multiclass	Balance
WikiQA	20360	2733	6165	Binary	Imbalance

Table 2: More Datasets.

4.1 Experimental Setup

For each of the architectures mentioned in section 3.5, we choose the best configuration of hyperparameters based on the validation portion of the particular dataset (For Claim and Evidence datasets, we consider a train:valid:test split of 60:10:30 while for the TE and WikiQA datasets we consider their corresponding given split).

Performance of the best performing configuration for every architecture is reported on the test data using the appropriate metric. As the claim and expert and study Evidences had similar data characteristics (in terms of data size and context and target lengths), we did extensive hyperparam tuning only on the claim dataset and applied the best configurations without further tuning to the expert and study datasets. Exhaustive hyperparam tuning was done on the TE dataset as well because its data characteristics are very different from other datasets.

In addition to reporting the test metrics for argumentation datasets, we carried out Leave-One-Out(leaving one motion out for testing) Mode training and evaluation which is more appropriate for this problem setting as it is crucial that we generalize well to totally unseen motion topics. In this case, we report the macro-average metrics over all motions.

4.2 Hyperparameter Tuning

Considering the number of variations of combinations of architectures we have considered, we have a huge hyperparameter space to deal with. Hence, we decided to fix insignificant hyperparameters and focus only on the relevant ones. We have decided to use word2vec (Mikolov et al., 2013) pretrained models for initializing the word embeddings across CBOW, CNN and RNNs and made them trainable specific to task at hand. In addition we have found through minimal tuning that the Adam (Kingma and Ba, 2015) optimizer seems to work best. We have also found that a learning rate of 0.001 works best

[3] https://www.research.ibm.com/haifa/dept/vst/mlta_data.shtml
[4] http://nlp.stanford.edu/projects/snli/
[5] http://aka.ms/WikiQA

in most scenarios except when the parameter space in some architectures (for ex, *bi-linear*) is large, in which case lower learning rates of 0.0001 or 0.00001 worked well. Rather than tuning the maximum sequence length for context and target sentence, we tried to fix it by getting reasonable values by plotting histogram of sequence lengths and had a cut-off at around 98-99 percentile. The max lengths turned out to be 14 for context and 60 for target for Claim, Evidence and WikiQA datasets while for textual entailment, they turned out to be 30 in both due to the symmetrical nature between premise and hypothesis. We tuned the following hyperparameters:

RNN Model : GRU or LSTM.

RNN Size : 50,100,200,300,400,500,1000.

CNN Filter Sizes : 3,3+4,3+4+5,2+3+4+5.

CNN Number of Filters : 10,20,40,64,128.

L2 Reg coeff for CNN : 0, 0.01, 0.001, 0.0001.

For every architecture type, we carried out the optimization of the relevant hyperparams from the above list over the whole grid. One point to note is that for certain architectures like conditional-state, as output of context is fed in to the hidden state of target RNN, there are some restrictions in the allowable context RNN/CNN hyperparam configurations based on the target RNN settings as the output dimension of the context RNN/CNN needs to match the hidden state dimension of the target RNN.

4.3 Evaluation Metrics

For the datasets Claim, Expert, Study Evidence and WikiQA datasets, we have used standard evaluation measures like Average Precision (Area under Precision Recall Curve) and AUC to choose best hyperparam configurations based on validation data as well as report test metrics. For argument mining specific tasks, we have reported other additional metrics like P@200, R@200, F1@200, P@50, R@50 and F1@50 (Levy et al., 2014) in addition to reporting AUC and Average Precision. Please note for leave-one-out mode, the reported metrics are macro-average over all motion topics. For Textual entailment, since it is a more balanced dataset, reporting valid and test accuracies are standard in the literature and we have done the same.

Task	Context	Target	Architecture	Test AVGP
Claim Sentence	RNN	CNN	**Concat**	**0.307**
	CNN	CNN	Concat	0.304
	Concat-Sentence baseline			0.17
EXPERT Evidence	RNN	RNN	**Conditional-State-Input**	**0.257**
	CNN	CNN	Concat	0.254
	Concat-Sentence baseline			0.225
STUDY Evidence	CNN	CNN	**Concat**	**0.297**
	RNN	CNN	Concat	0.29
	Concat-Sentence baseline			0.236
WikiQA	CBOW	RNN	**Concat**	**0.187**
	CNN	RNN	Conditional-State-Input	0.186

Table 3: Empirical evaluation based on Average Precision on assymetric datasets.

Task	Context	Target	Architecture	Test AUC
Claim Sentence	CNN	CNN	**Concat**	**0.873**
	CNN	RNN	Conditional-State	0.873
	Concat-Sentence baseline			0.831
EXPERT Evidence	RNN	RNN	**Conditional-State**	**0.832**
	RNN	RNN	Conditional-State-Input	0.823
	Concat-Sentence baseline			0.805
STUDY Evidence	CNN	CNN	**Concat**	**0.87**
	CBOW	CNN	Concat	0.864
	Concat-Sentence baseline			0.844
WikiQA	CNN	CNN	**Concat**	**0.74**
	CBOW	RNN	Concat	0.74

Table 4: Empirical evaluation based on AUC on assymetric datasets.

Method	Model	TrainAcc(%)	TestAcc(%)
(Bowman et al., 2015)	Use of features incl unigrams and bigrams	99.7	78.2
(Vendrov et al., 2016)	1024D GRU encoders w/ unsupervised 'skip-thoughts' pre-training	98.8	81.4
(Mou et al., 2016)	300D Tree-based CNN encoders	83.3	82.1
(Cheng et al., 2016)	450D LSTMN with deep attention fusion	88.5	86.3
(Parikh et al., 2016)	200D decomposable attention model with intra-sentence attention	90.5	**86.8**
Conditional-State-RNN-RNN	Simple architecture with RNNs without attention	89.97	82.36

Table 5: Comparison with the state-of-the-art in Textual Entailment dataset.

Method	P@200	R@200	F1@200	P@50	R@50	F1@50	AVGP	AUC
CDCD (Levy et al., 2014)**	9.0	**73.0**	-	18.0	40.0	-	-	-
BoW (Lippi and Torroni, 2015b)	8.2	51.7	14.2	-	-	-	0.117	0.771
TK (Lippi and Torroni, 2015b)	9.8	58.7	16.8	-	-	-	0.161	0.808
TK+Topic (Lippi and Torroni, 2015b)	**10.5**	62.9	**18.0**	-	-	-	0.178	0.823
Concat-CNN-CNN	9.64	61.5	15.8	17.1	27.7	19.2	0.173	0.812
Conditional-State-Input-RNN-RNN	9.56	60.0	15.6	16.6	26.9	18.5	0.162	0.801

Table 6: Results in Leave-One-Motion-Out mode for Claim Sentence Task. **Levy et al. (2014) used a smaller version of the dataset consisting of only 32 motions and also less number of claims. For fair comparison, we also use the same version of dataset as in CDCD and report the results in Appendix A.

Task	P@200	R@200	F1@200	P@50	R@50	F1@50	P@20	P@10	P@5	AVGP	AUC
Claim Sentence Task	9.64	61.5	15.8	17.1	27.7	19.2	22.4	27.9	28.5	0.173	0.812
EXPERT Evidence Task	9.53	64.0	14.5	14.5	35.0	15.7	18.6	21.1	22.5	0.160	0.750
STUDY Evidence Task	8.33	79.5	13.5	15.5	53.9	18.9	20.8	25.3	31.8	0.298	0.836

Table 7: Numbers in Leave-One-Motion-Out mode for all three debating tasks using our approach.

4.4 Results and Discussion

Tables 3 and 4 report the two top ranking architectures for four datasets based on Test AUC and Test Avg Precision. We find that **Concat** is the winning architecture variant across majority of the datasets considered. Moreover, the runner-up architecture type **Conditional-State-Input** is also similar to 'Concat' in the sense that concatenation of context representation is done at the input of the sentence RNN. Now the four datasets we considered are asymmetric in nature as there are significantly fewer contexts (motions or questions) than the targets. Hence the context model does not see enough data for learning and hence, if the learnt context model is fed directly to the hidden state of the target RNN, the improperly learnt context model can play a big role. In contrast if a Concat kind of architecture is used, the linear plus softmax layer can decide on how much importance to give to the context model. Hence, Concat is doing better in this case.

From Textual entailment dataset(which is symmetric in nature), we found that conditional type of architectures are doing better at the Test accuracies. In fact, the winning architecture was **Conditional-State** with RNN-RNN combo, which did better in terms of test accuracy than the feature based models (Bowman et al., 2015) and one tree-based model (Mou et al., 2016). However, it came close to the state-of-the-art attention based model (Parikh et al., 2016). In our work we are empirically evaluating simple architectures for bisequence classification without using more sophisticated tree-based or attention-based models. It is possible that adding attention on top of this will improve the results further.

The *bi-linear* model, is supposed to capture the interaction between the context and target reps via a quadratic form (section 3.4). For the asymmetric datasets, this is not doing well again due to insufficient data for context. Whereas, it does well for the TE data. However, due to the huge parameter space for bi-linear, training times are considerably higher and requires lower learning rate than other architecture types. The runtimes are comparable for the other architecture variants.

From Table 6, the main takeaway is that we are the only deep learning based method with zero feature engineering and we have come very close to the state-of-the-art systems (Levy et al., 2014) and (Lippi and Torroni, 2015b), which are heavily feature-engineered. Here again the winner is a 'Concat' based combination of architecture. Moreover, Tables 6 and 7 are the first deep learning zero feature engineered baselines for all argument mining datasets. Appendix A contains the details of the exhaustive

experiments on all architectures on the different datasets in terms of the best hyperparameters used.

5 Conclusion

In this work, we have considered taking up multiple architectures for bisequence classification tasks, for which not much understanding is there in the current literature. In addition to suggesting winning architecture recipes for different kinds of datasets, we have established deep learning based baselines for argument mining tasks with zero feature engineering. As future work, it remains to be seen how adding attention on top of winning simple architectures fare in terms of benchmark performance.

6 Acknowledgements

We would like to thank Mitesh M. Khapra for the multiple discussions that we had with him leading to this paper. In addition, we are also grateful to our colleagues at IBM Debating Technologies for the numerous suggestions and feedback at different points of time.

References

Ehud Aharoni, Anatoly Polnarov, Tamar Lavee, Daniel Hershcovich, Ran Levy, Ruty Rinott, Dan Gutfreund, and Noam Slonim. 2014. A benchmark dataset for automatic detection of claims and evidence. In *in the Context of Controversial Topics", in Proceedings of the First Workshop on Argumentation and Computation, ACL 2014*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference for Learning Representations, San Diego, 2015*.

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 1489–1500.

Marco Lippi and Paolo Torroni. 2015a. Argument mining: A machine learning perspective. In *Theory and Applications of Formal Argumentation - Third International Workshop, TAFA 2015, Buenos Aires, Argentina, July 25-26, 2015, Revised Selected Papers*, pages 163–176.

Marco Lippi and Paolo Torroni. 2015b. Context-independent claim detection for argument mining. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 185–191.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 130–136, Berlin, Germany, August. Association for Computational Linguistics.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, US., November. Association for Computational Linguistics.

Mengye Ren, Ryan Kiros, and Richard S. Zemel. 2015. Exploring models and data for image question answering. In *In Advances in Neural Information Processing Systems*, pages 2935–2943.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence - an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 440–450.

Noam Slonim, Ehud Aharoni, Carlos Alzate Perez, Roy Bar-Haim, Yonatan Bilu, Lena Dankin, Iris Eiron, Daniel Hershcovich, Shay Hummel, Mitesh M. Khapra, Tamar Lavee, Ran Levy, Paul Matchen, Anatoly Polnarov, Vikas C. Raykar, Ruty Rinott, Amrita Saha, Naama Zwerdling, David Konopnicki, and Dan Gutfreund. 2014. Claims on demand - an initial demonstration of a system for automatic detection and polarity identification of context dependent claims in massive corpora. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference System Demonstrations, August 23-29, 2014, Dublin, Ireland*, pages 6–9.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *4th International Conference for Learning Representations, Puerto Rico, 2016*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal, September. Association for Computational Linguistics.

A Appendix

Method	**P@200**	**R@200**	**F1@200**	**P@50**	**R@50**	**F1@50**	**AVGP**	**AUC**
CDCD (Levy et al., 2014)	**9.0**	**73.0**	-	**18.0**	**40.0**	-	-	-
Concat-CNN-CNN	7.29	60.5	12.4	14.7	31.5	18.3	0.166	0.810
Conditional-State-Input-RNN-RNN	6.87	58.1	11.7	13.5	29.5	17.0	0.163	0.789

Table 8: Results in Leave-One-Motion-Out mode for Claim Sentence Task according to the dataset used by Levy et al. (2014).

Architecture	**Context**	**Target**	**Setting**	**Test AVGP**
Concat	CBOW	CNN	FilterSize:3,Filters:40	0.3
Concat	CBOW	RNN	Cell:GRU,Size:300	0.276
Concat	RNN	CNN	Cell:LSTM,Size:200,FilterSize:3+4+5,Filters:20	**0.307**
Concat	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:64	0.28
Concat	RNN	RNN	Cell:GRU,Size:300	0.27
Concat	CNN	CNN	FilterSize:3+4,Filters:64,L2:0.01	0.304
Bilinear	CBOW	CNN	FilterSize:3+4+5,Filters:40	0.237
Bilinear	CBOW	RNN	Cell:GRU,Size:300	0.263
Bilinear	RNN	CNN	Cell:GRU,Size:200,FilterSize:3+4+5,Filters:64	0.254
Bilinear	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:10	0.268
Bilinear	RNN	RNN	Cell:GRU,Size:200	0.263
Bilinear	CNN	CNN	FilterSize:3+4,Filters:20	0.237
Conditional-State	CNN	RNN	Cell:GRU,Size:200,FilterSize:3+4,Filters:100	0.248
Conditional-State	RNN	RNN	Cell:GRU,Size:200	0.266
Conditional-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3+4+5,Filters:100	0.254
Conditional-Input	RNN	RNN	Cell:GRU,Size:100	0.246
Conditional-State-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3,Filters:300	0.264
Conditional-State-Input	RNN	RNN	Cell:GRU,Size:100	0.247
Concat-Sentence baseline			Cell:GRU,Size:200	0.17

Table 9: Best configurations of all architectures on Claim Sentence Dataset tuning based on AVGP

Architecture	Context	Target	Setting	Test AUC
Concat	CBOW	CNN	FilterSize:3+4+5,Filters:64,L2:0.01	0.863
Concat	CBOW	RNN	Cell:GRU,Size:200	0.868
Concat	RNN	CNN	Cell:GRU,Size:200,FilterSize:3,Filters:40	0.867
Concat	CNN	RNN	Cell:LSTM,Size:200,FilterSize:3+4+5,Filters:20	0.855
Concat	RNN	RNN	Cell:GRU,Size:100	0.864
Concat	CNN	CNN	FilterSize:3,Filters:128,L2:0.01	**0.873**
Bilinear	CBOW	CNN	FilterSize:3,Filters:128,L2:0.01,LR:0.0001	0.831
Bilinear	CBOW	RNN	Cell:GRU,Size:500	0.832
Bilinear	RNN	CNN	Cell:LSTM,Size:100,FilterSize:3+4,Filters:128,LR:0.00001	0.828
Bilinear	CNN	RNN	Cell:LSTM,Size:200,FilterSize:3+4+5,Filters:10,LR:0.0001	0.857
Bilinear	RNN	RNN	Cell:LSTM,Size:300,LR:0.0001	0.855
Bilinear	CNN	CNN	FilterSize:3+4,Filters:64,L2:0.001,LR:0.0001	0.82
Conditional-State	CNN	RNN	Cell:GRU,Size:48,FilterSize:3+4+5,Filters:16,LR:0.0001	0.873
Conditional-State	RNN	RNN	Cell:GRU,Size:100	0.86
Conditional-Input	CNN	RNN	Cell:GRU,Size:50,FilterSize:3,Filters:50,LR:0.0001	0.873
Conditional-Input	RNN	RNN	Cell:GRU,Size:200	0.86
Conditional-State-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3+4,Filters:150,LR:0.00001	0.856
Conditional-State-Input	RNN	RNN	Cell:GRU,Size:200	0.862
Concat-Sentence baseline			Cell:GRU,Size:200	0.83

Table 10: Best configurations of all architectures on Claim Sentence Dataset tuning based on AUC

Architecture	Context	Target	Setting	Test AVGP	Test AUC
Concat	CBOW	CNN	FilterSize:3,Filters:40	0.239	0.81
Concat	CBOW	RNN	Cell:GRU,Size:300	0.251	0.819
Concat	RNN	CNN	Cell:LSTM,Size:200,FilterSize:3+4+5,Filters:20	0.242	0.812
Concat	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:64	0.231	0.794
Concat	RNN	RNN	Cell:GRU,Size:300	0.241	0.811
Concat	CNN	CNN	FilterSize:3+4,Filters:64,L2:0.01	0.254	0.819
Bilinear	CBOW	CNN	FilterSize:3+4+5,Filters:40	0.218	0.79
Bilinear	CBOW	RNN	Cell:GRU,Size:300	0.202	0.788
Bilinear	RNN	CNN	Cell:GRU,Size:200,FilterSize:3+4+5,Filters:64	0.219	0.789
Bilinear	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:10	0.229	0.791
Bilinear	RNN	RNN	Cell:GRU,Size:200	0.214	0.788
Bilinear	CNN	CNN	FilterSize:3+4,Filters:20	0.233	0.792
Conditional-State	CNN	RNN	Cell:GRU,Size:200,FilterSize:3+4,Filters:100	0.226	0.797
Conditional-State	RNN	RNN	Cell:GRU,Size:200	0.254	**0.832**
Conditional-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3+4+5,Filters:100	0.229	0.797
Conditional-Input	RNN	RNN	Cell:GRU,Size:100	0.231	0.817
Conditional-State-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3,Filters:300	0.211	0.796
Conditional-State-Input	RNN	RNN	Cell:GRU,Size:100	**0.257**	0.823
Concat-Sentence baseline			Cell:GRU,Size:200	0.225	0.805

Table 11: Performance of all architectures on EXPERT Evidence Dataset

Architecture	Context	Target	Setting	Test AVGP	Test AUC
Concat	CBOW	CNN	FilterSize:3,Filters:40	0.281	0.864
Concat	CBOW	RNN	Cell:GRU,Size:300	0.279	0.851
Concat	RNN	CNN	Cell:LSTM,Size:200,FilterSize:3+4+5,Filters:20	0.29	0.863
Concat	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:64	0.262	0.829
Concat	RNN	RNN	Cell:GRU,Size:300	0.28	0.842
Concat	CNN	CNN	FilterSize:3+4,Filters:64,L2:0.01	**0.297**	**0.869**
Bilinear	CBOW	CNN	FilterSize:3+4+5,Filters:40	0.271	0.831
Bilinear	CBOW	RNN	Cell:GRU,Size:300	0.202	0.788
Bilinear	RNN	CNN	Cell:GRU,Size:200,FilterSize:3+4+5,Filters:64	0.271	0.833
Bilinear	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:10	0.254	0.839
Bilinear	RNN	RNN	Cell:GRU,Size:200	0.257	0.84
Bilinear	CNN	CNN	FilterSize:3+4,Filters:20	0.275	0.835
Conditional-State	CNN	RNN	Cell:GRU,Size:200,FilterSize:3+4,Filters:100	0.254	0.835
Conditional-State	RNN	RNN	Cell:GRU,Size:200	0.267	0.861
Conditional-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3+4+5,Filters:100	0.245	0.838
Conditional-Input	RNN	RNN	Cell:GRU,Size:100	0.28	0.854
Conditional-State-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3,Filters:300	0.257	0.839
Conditional-State-Input	RNN	RNN	Cell:GRU,Size:100	0.25	0.849
Concat-Sentence baseline			Cell:GRU,Size:200	0.236	0.844

Table 12: Performance of all architectures on STUDY Evidence Dataset

Architecture	Context	Target	Setting	Test AVGP	Test AUC
Concat	CBOW	CNN	FilterSize:3,Filters:40	0.162	0.735
Concat	CBOW	RNN	Cell:GRU,Size:300	**0.187**	0.74
Concat	RNN	CNN	Cell:LSTM,Size:200,FilterSize:3+4+5,Filters:20	0.15	0.727
Concat	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:64	0.119	0.66
Concat	RNN	RNN	Cell:GRU,Size:300	0.171	0.705
Concat	CNN	CNN	FilterSize:3+4,Filters:64,L2:0.01	0.179	**0.74**
Bilinear	CBOW	CNN	FilterSize:3+4+5,Filters:40	0.129	0.672
Bilinear	CBOW	RNN	Cell:GRU,Size:300	0.119	0.656
Bilinear	RNN	CNN	Cell:GRU,Size:200,FilterSize:3+4+5,Filters:64	0.122	0.676
Bilinear	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:10	0.131	0.681
Bilinear	RNN	RNN	Cell:GRU,Size:200	0.149	0.688
Bilinear	CNN	CNN	FilterSize:3+4,Filters:20	0.129	0.712
Conditional-State	CNN	RNN	Cell:GRU,Size:200,FilterSize:3+4,Filters:100	0.122	0.681
Conditional-State	RNN	RNN	Cell:GRU,Size:200	0.171	0.739
Conditional-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3+4+5,Filters:100	0.141	0.713
Conditional-Input	RNN	RNN	Cell:GRU,Size:100	0.184	0.729
Conditional-State-Input	CNN	RNN	Cell:GRU,Size:300,FilterSize:3,Filters:300	0.186	0.726
Conditional-State-Input	RNN	RNN	Cell:GRU,Size:100	0.169	0.714

Table 13: Performance of all architectures on WikiQA Dataset

Architecture	Context	Target	Setting	TrainAcc(%)	ValidAcc(%)	TestAcc(%)
Concat	CBOW	CNN	FilterSize:3+4+5,Filters:128	74.33	69.43	68.44
Concat	CBOW	RNN	Cell:LSTM,Size:400	72.75	69.4	69.02
Concat	RNN	CNN	Cell:GRU,Size:200,FilterSize:3+4+5,Filters:20	74.01	69.34	68.96
Concat	CNN	RNN	Cell:GRU,Size:200,FilterSize:3,Filters:20	72.93	69.99	69.69
Concat	RNN	RNN	Cell:LSTM,Size:200	72.74	69.96	69.46
Concat	CNN	CNN	FilterSize:3+4+5,Filters:64	74.55	69.49	68.97
Bilinear	CBOW	CNN	FilterSize:3+4,Filters:128	83.86	77.1	77.07
Bilinear	CBOW	RNN	Cell:GRU,Size:300	84.78	79.07	78.19
Bilinear	RNN	CNN	Cell:GRU,Size:500,FilterSize:2+3+4+5,Filters:200	84.42	77.68	77.18
Bilinear	CNN	RNN	Cell:GRU,Size:500,FilterSize:2+3+4+5,Filters:200	83.71	78.72	78.6
Bilinear	RNN	RNN	Cell:GRU,Size:1000,LR:0.0001	84.91	81.1	80.3
Bilinear	CNN	CNN	FilterSize:3+4,Filters:128,LR:0.0001	84.51	76.58	76.81
Conditional-State	CNN	RNN	Cell:GRU,Size:500,FilterSize:3,Filters:500	87.77	80.87	80.81
Conditional-State	RNN	RNN	Cell:GRU,Size:500	**89.97**	**82.38**	**82.36**
Conditional-Input	CNN	RNN	Cell:GRU,Size:500,FilterSize:3+4,Filters:250	87.36	80.81	81.1
Conditional-Input	RNN	RNN	Cell:GRU,Size:500	89.02	81.45	80.92
Conditional-State-Input	CNN	RNN	Cell:GRU,Size:500,FilterSize:3,Filters:500	85.78	80.05	79.61
Conditional-State-Input	RNN	RNN	Cell:GRU,Size:500	89.03	81.93	81.38

Table 14: Performance of all architectures on Textual Entailment Dataset

Learning Succinct Models: Pipelined Compression with L1-Regularization, Hashing, Elias–Fano Indices, and Quantization

Hajime Senuma[†‡] and Akiko Aizawa[‡†]

[†]University of Tokyo, Tokyo, Japan
[‡]National Institute of Informatics, Tokyo, Japan
{senuma,aizawa}@nii.ac.jp

Abstract

The recent proliferation of smart devices necessitates methods to learn small-sized models. This paper demonstrates that if there are m features in total but only $n = o(\sqrt{m})$ features are required to distinguish examples, with $\Omega(\log m)$ training examples and reasonable settings, it is possible to obtain a good model in a *succinct* representation using $n \log_2 \frac{m}{n} + o(m)$ bits, by using a pipeline of existing compression methods: L1-regularized logistic regression, feature hashing, Elias–Fano indices, and randomized quantization. An experiment shows that a noun phrase chunking task for which an existing library requires 27 megabytes can be compressed to less than 13 kilobytes without notable loss of accuracy.

1 Introduction

The age of smart devices has arrived. Cisco reports that, as of 2015, smartphones and tablets account for 15% of IP traffic (against 53% for PCs), and further predicts that, by 2020, this share will have grown to 43%, surpassing the 29% of PCs (Cisco Systems, 2016). This trend increases the need for intelligent processing for these platforms. Hence, the study of statistical methods for natural language processing (NLP) systems on mobile devices has received considerable attention in recent years (Ganchev and Dredze, 2008; Hagiwara and Sekine, 2014; Chen et al., 2015).

Among the various issues in this setting, storage costs pose a particular challenge, because the size of the resulting models often grow quickly. Even a simple noun phrase (NP) chunker can easily take dozens of megabytes if naïvely implemented. Suppose we have m features. A direct implementation then consumes $Z(\mathcal{A}) + O(m\zeta)$, where $Z(\mathcal{A})$ represents the size of an *alphabet* $\mathcal{A}$, or a *feature dictionary* that maps a feature string to an index into its parameter vector, and $O(m\zeta)$ represents the space complexity of a dense real vector as the parameter where using ζ bits to achieve a certain amount of float precision ($64m$ if using double-precision floats). These large-sized models are not only inefficient in terms of network bandwidth, but also significantly increase energy consumption (Han et al., 2016). Therefore, it is vital to devise a method that achieves as small a model as possible.

One of the basic properties of information theory says that, to represent a set of n non-negative integers less than m (or, equivalently, a bit vector of size m containing n 1s), we need at least $B_{m,n} = \lceil \log_2 \binom{m}{n} \rceil \approx n \log_2 \frac{m}{n}$ bits. Recent studies in theoretical computer science show that it is possible to compress a set of non-negative integers while keeping some primitive operations under *succinct* representations, that is, data structures using only $B_{m,n} + o(m)$ bits.

As an analogy to succinct data structures, one may ask a question: if there are m features, but only n of these are useful for distinguishing data, how many bits are required to obtain a good classifier? This paper shows that under several reasonable assumptions such as $n = o(\sqrt{m})$, with $\Omega(\log m)$ examples, it is possible to obtain a model that performs similarly to the original classifier, by using only $B_{m,n} + o(m)$ bits.

To evaluate our method, we implemented conditional random fields (CRFs) by using our pipelined architecture and conducted an experiment on an NP chunking model. The results show that 27 megabytes can be reduced to about 13 kilobytes.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2774–2784, Osaka, Japan, December 11-17 2016.

In summary, our work is significant in that

1. we present a pipelined method to acquire a *succinct* model that also has almost the same predictive performance as the model of Ng (2004),
2. although the above method is achieved under some conditions and penalties, the conditions are not ungrounded for NLP tasks and the the penalties are neglible for large m,
3. we also introduce a bitwise trick to perform fast unbiased feature hashing, a part of the pipeline,
4. and our experimentations on sequential labeling tasks achieve smaller models than existing libraries by a factor of one thousand, demonstrating the high practical value of our approach.

This paper is organized as follows. In Section 2, we mention related work. In Section 3, we explain the notation used in the paper. In Section 4, we introduce and analyze our methods. In Section 5, we evaluate our approach by an experiment.

2 Related Work

Ganchev and Dredze (2008) were one of the earliest groups to recognize the importance of acquiring small models for mobile platforms. They also showed that naïve hashing tricks for features (now commonly known as *feature hashing*) can greatly improve the memory efficiency without much loss of accuracy. Shi et al. (2009) showed that feature hashing could be applied to graph models. Weinberger et al. (2009) proposed an unbiased version of feature hashing. Bohnet (2010) applied the method to dependency parsing, and found that it improves not only memory efficiency, but also speed performance. Recent studies (Chen et al., 2015) suggest that hashing tricks also work well in deep neural networks too.

Feature selection through the L1 regularization is also known to be effective, since, roughly speaking, the number of features in the final model results in being logarithmic to the number of total features (Ng, 2004). Online optimization (vis-á-vis batch optimization) with the L1-regularization term gained attention from around 2010 (Duchi and Singer, 2009; Tsuruoka et al., 2009), and with the AdaGrad (Duchi et al., 2011) method becoming particularly popular, because of its theoretical and practical performance.

Compression by quantization is closely related to machine learning. For eaxmple, the Lloyd quantization (Lloyd, 1982) is now used as a popular clustering method known as the K-means clustering. Golovin et al. (2013) showed that simple (simplistic, in fact) quantization techniques actually exhibit good performance in common settings used in NLP; although their methods themselves were basic, they analyzed theoretical effects in detail, and these serve as the key ingredients for our error analysis in Section 4.

The compression of indices is a classical topic in information retrieval (IR). Vigna (2013) introduced the Elias–Fano structure (Fano, 1971; Elias, 1974) into IR, although this was already a popular theme in the succinct data structures community (Grossi and Vitter, 2005; Okanohara and Sadakane, 2007; Golynski et al., 2014). Although other types of succinct structures are common in NLP as well (Watanabe et al., 2009; Sorensen and Allauzen, 2011; Shareghi et al., 2015), the Elias–Fano structure is in that, if regarded as a binary vector, the order of compression ratio depends on the number of 1s (rather than the total length of a vector) and achieves better compression ratio than other succinct models for binary vectors if the ratio of 1s is very low (empirically, below 10%); to put it more concretely, the structure is very attractive when unnecessary indices are culled by some techniques such as L1-regularization. Because of its simplicity and compression ratio, the Elias–Fano structure even gained a certain degree of popularity in industry; for example, as of 2013, it is used as one of the backbones of Facebook's social graph engine (Curtiss et al., 2013). Several variants have been proposed, including sdarry (Okanohara and Sadakane, 2007) and the partitioned Elias–Fano indices (Ottaviano and Venturini, 2014).

Driven by these successes, the study of pipelining compression techniques has flourished in recent years. The web-based morphological parser developed by Hagiwara and Sekine (2014) is similar to our work in its aim and basic scheme, but some parts were not implemented and they gave no theoretical explanation of why pipelined compression would work well. For large-scale neural networks, by using

Han et al. (2015)'s three-step pruning process, Han et al. (2016) gained great empirical success using deep compression, or pipelined compression for a deep neural network with pruning, trained quantization, and Huffman coding.

3 Notation

The base of logarithm is e if not stated explicitly. If the base is x ($x \neq e$), we write $\log_x$. $\mathbb{E}$ denotes an expectation, $\mathbb{R}$ denotes a set of real values, and $\mathbb{N}$ denotes a set of non-negative integers. $\delta_{i,j}$ represents the Kronecker's delta, that is, $\delta_{i,j} = 1$ if $i = j$, and 0 otherwise. $B_{m,n}$ represents the information-theoretic lower bound of encoding a bit vector of size m with n 1s, that is, $n \left\lceil \log_2 \binom{m}{n} \right\rceil$.

4 Learning Succinct Models

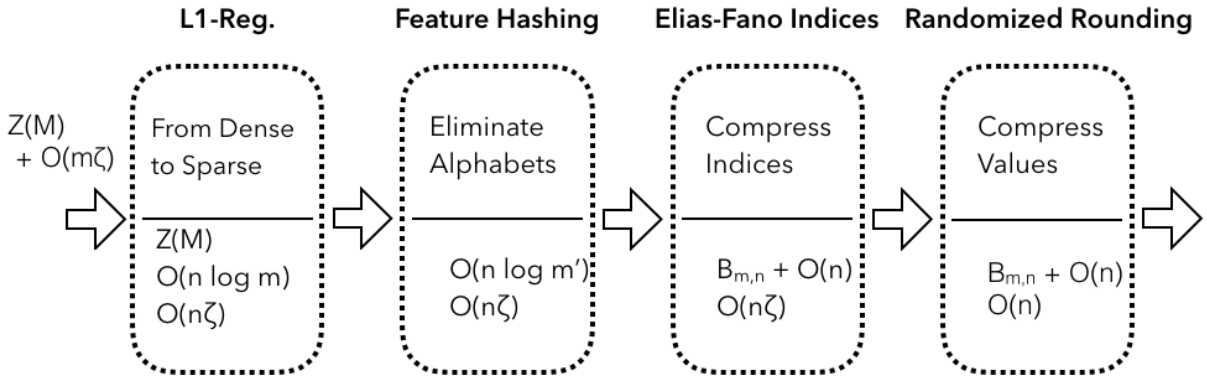


Figure 1: Overview of pipelined compression.

In this section, we present a supervised learning method to obtain a small-sized model for classification tasks. For simplicity, we assume binary classification for each label $y \in \{0, 1\}$ by logistic regression. Suppose there is a set of m features and a dataset $\mathcal{T} = \left\{(\mathbf{x}^{(0)}, y^{(0)}), (\mathbf{x}^{(1)}, y^{(1)}, ..., (\mathbf{x}^{(T-1)}, y^{(T-1)})\right\}$ of T training examples drawn i.i.d. from some distribution $\mathcal{D}$.

Consider the following optimization problem with the L1-regularization term $R(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_1$

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad & \sum_{i=0}^{T-1} \log p(y_i \mid \mathbf{x}_i \,;\, \boldsymbol{\theta}) \\ \text{subject to} \quad & R(\boldsymbol{\theta}) \leq B. \end{aligned} \tag{1}$$

Our goal is to obtain a good parameter $\boldsymbol{\theta}$ that minimizes the expected logistic log loss, that is,

$$\varepsilon(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[-\log p(y \mid \mathbf{x} \,;\, \boldsymbol{\theta})] \tag{2}$$

with as small a model as possible. We also define the empirical loss

$$\hat{\varepsilon}(\boldsymbol{\theta}) = \hat{\varepsilon}_{\mathcal{T}}(\boldsymbol{\theta}) = \frac{1}{T} \sum_{i=0}^{T-1} -\log p(y_i \mid \mathbf{x}_i \,;\, \boldsymbol{\theta}). \tag{3}$$

The m-dimensional vector $\boldsymbol{\theta}$ is often implemented as a float array of size m, consuming $O(m\zeta)$ bits. Concretely, it occupies $64m$ bits if we use double-precision ($\zeta = 64$) and $32m$ bits if single-precision ($\zeta = 32$). As m can be millions, billions, or more in NLP tasks, this size complexity is often non-negligible for mobile and embedded devices (e.g., 762.9 megabytes for 100,000,000 features when using double-precision).

In addition to the parameter vector, we need to store an *alphabet* or a *feature dictionary* $\mathcal{A}\colon \mathbb{N} \to \{0, ..., m-1\}$ such that $\mathcal{A}(\mathfrak{f}_i) = i$ for $\mathfrak{f}_0, ..., \mathfrak{f}_{m-1} \in \mathbb{N}$. For example, if the 12-th element in a feature set

represents "the current token is `appropriate`, the previous token is `I`, and the current label is `VERB`", one possible formulation is $\mathfrak{f}_{12}$ = `utf8("bigram[-1:0]=I:appropriate&label=VERB")`, where `utf8` denotes a non-negative integer given by the UTF-8 bit representation of a string. A weight value for this feature can then be accessed as $\boldsymbol{\theta}_{\mathcal{A}(\mathfrak{f}_{12})}$. There are two problems with the use of alphabets. First, they consume too much storage space (Ganchev and Dredze, 2008). Second, if implemented with a hash table (as is often the case), they are so slow that they become the bottleneck of the entire learning system (Bohnet, 2010). In the following, $Z(\mathcal{A})$ denotes the size of an alphabet $\mathcal{A}$.

4.1 Pipelined method

Let us now introduce a pipelined method to reduce both $O(m\zeta)$ and $Z(\mathcal{A})$.

Definition 4.1. Given inputs $\mathcal{T}$,

1. Train the model by using L1-regularization according to the method of Ng (2004):
   (a) Divide the data $\mathcal{T}$ into a training set $\mathcal{T}_1$ of the size $(1-\gamma)T$ and a development set $\mathcal{T}_2$ of the size γT, for some $\gamma \in \mathbb{R}$.
   (b) For $B = 0, 1, 2, 4, ..., C$, solve the optimization under each of these hyperparameters with $\mathcal{T}_1$, giving $\boldsymbol{\theta}_0, ..., \boldsymbol{\theta}_C$.
   (c) Choose $\boldsymbol{\theta} = \arg\min_{\{i \in 0,1,2,...,C\}} \hat{\varepsilon_{\mathcal{T}_2}}(\boldsymbol{\theta}_i)$, or the best model for the development data set.

   As we performed the L1-regularization, it is safe to assume that the non-zero components are quite small. $Z(\mathcal{A}) + O(n \log m) + O(n\zeta)$; $Z(\mathcal{A})$ for the alphabet, $\lceil \log m \rceil$ bits for each of the n indices, and $zeta$) bits for each of the n values.

2. Eliminate the alphabet and reduce the dimensionality of vectors by using the *unbiased feature hashing* (Weinberger et al., 2009). Given two hash functions $h\colon \mathbb{N} \to \{0, ..., m'-1\}$ and $\xi\colon \mathbb{N} \to \{-1, 1\}$, we define the (unbiased) *hashed feature map* ϕ such that $\phi^{(h,\xi)(\mathbf{x})} = \sum_{j\colon h(j)=i} \xi(i)\mathbf{x}_i$. We apply this ϕ to the parameter vector and all future input vectors. As this map acts as a proxy for the alphabet, we no longer need to store the alphabet. As a side-effect, if we choose $m' < m$, the dimensionality is reduced from m to m'. Note that this is said to be unbiased because given $\langle \mathbf{x}, \mathbf{x}' \rangle_\phi := \left\langle \phi^{(h,\xi)}(\mathbf{x}), \phi^{(h,\xi)}(\mathbf{x}') \right\rangle$, $\mathbb{E}\langle \mathbf{x}, \mathbf{x}' \rangle_\phi = \langle \mathbf{x}, \mathbf{x}' \rangle$. In other words, on average, prediction using a hashed map space is the same as the original version.

   At this point, the size has been reduced to $O(n \log m') + O(n\zeta)$; $\lceil \log m' \rceil$ for each of the n indices and ζ) bits for each of the n values.

3. Compress the set of n indices by using the Elias–Fano scheme (Fano, 1971; Elias, 1974). In brief, n integers are compressed to $n\lceil \log_2 m/n \rceil + f(n)$, where $f(n) = O(n)$ and $f(n) \leq 2Sn$ for some speed-memory trade-off hyperparameter $1 \leq S < 2$ (usually less than 1.01). We omit the details here, but interested readers may consult Vigna (2013)'s good introduction.

   At this point, the size has been reduced to $n\lceil \log_2 m'/n \rceil + O(n) + O(n\zeta)$.

4. Compress the set of values by using the *unbiased randomized rounding* (Golovin et al., 2013). Let μ and ν be some non-negative integers, then it encodes each value as a fixed-point number with Q$\mu.\nu$ encoding, that is, as μ integer bits and ν fractional bits (e.g., if $\mu = \nu = 2$, we can represent a maximum of $(11.11)_2 = 3.75$). Including a sign bit, $\mu + \nu + 1$ for each value. μ and ν are chosen so that they satisfy the conditions of the error anaysis in the following subsection.

   At this point, the size has been reduced to $n\lceil \log_2 m'/n \rceil + O(n)$.

Overall, because $n\lceil \log_2 m/n \rceil = B_{m,n} + O(n)$ (Golynski et al., 2014), choosing $m' < m$ and assuming $n = o(m)$ achieves a model with $B_{m,n} + o(m)$. In the next subsection, however, we assume tighter conditions such as $n = o(\sqrt{m})$ to achieve good error bounds.

4.2 Error analysis

Each compression technique (except Elias–Fano) adds several penalties to the predictive performance. Therefore, intuitively, pipelined compression should lower the performance significantly. Contrary to such intuition, however, the increase in the error is negligible under certain circumstances.

Theorem 4.2. *Suppose that there exist m features in total, but only n features are important, so that the optimal parameter vector $\theta^* \in \mathbb{R}^m$ has only n non-zero components with indices $0 \leq i_0, i_1, ..., i_{n-1} < m$. Further, assume that each non-zero component is bounded by some constant no less than 1, that is, $0 < |\theta_{i_j}| \leq K$ for $j = 0, ..., n-1$ with $K \geq 1$.*

Let us be given a training data set $\mathcal{T} = \left\{(\mathbf{x}^{(0)}, y^{(0)}), (\mathbf{x}^{(1)}, y^{(1)}, ..., (\mathbf{x}^{(T-1)}, y^{(T-1)})\right\}$ of T training examples drawn i.i.d. from some distribution $\mathcal{D}$. Assume that the number of non-zero components of each input is always bounded by some non-negative constant much smaller than m, and that n is also much smaller than m as follows:

$$||\mathbf{x}||_0 \leq k \text{ for any } (\mathbf{x}, y) \sim \mathcal{D}, k \geq 0, k = o(\sqrt{m}), n = o(\sqrt{m}), \text{ and } K = o(m). \tag{4}$$

Then, with probability at least $(1-\delta)(1+f_1(m))$ where $f_1(m) = o(1)$, it is possible to obtain a model with size (in bits) at most

$$B_{m,n} + o(m) \tag{5}$$

with errors to the expected logistic loss

$$\varepsilon(\hat{\theta}) \leq \epsilon_{mul} \left(\varepsilon(\theta^*) + \epsilon_{add}\right) \tag{6}$$

where $\epsilon_{mul} = 1 + f_2(m)$ with f_2 such that $f_2(m) = o(1)$, by performing $C \geq nk$ iterations to find a good hyperparameter and by using the T examples such that

$$T = \Omega((\log m) \cdot \textit{poly}(n, K, \log(1/\delta), 1/\epsilon_{add}, C)). \tag{7}$$

Prior to its proof, let us explain what the above theorem says. If we omit the conditions of Equation (4), we have the same setting used in Ng (2004, Theorem 3.1). Thus, if we perform L1-regularized learning as defined in 1 of Definition 4.1, Ng's theorem implies that Equations (6) and (7) hold, satisfying $f_1(m) = 0$ and $f_2(m) = 0$.

In other words, the above theorem implies that if we add the conditions defined in Equation (4)) to Ng's theorem, it is possible to obtain a succinct representation (Equation (5)) whose predictive performance is nearly as good as that of Ng's theorem, albeit with a probabilistic penalty $f_1(m)$ and an error penalty $f_2(m)$. However, the penalties are negligible because $f_1(m) = o(1)$ and $f_2(m) = o(1)$.

Note that for NLP tasks, some of the conditions can be justified to a certain extent by the power law. Let us follow the discussion presented by Duchi et al. (2011, Section 1.3). In NLP tasks, the appearance of features often follows the power law. Thus, we can assume that the i-th most frequent feature appears with probability $p_i = \min(1, ci^{-\alpha})$ for some $a \in (1, +\infty)$ and a positive constant c. Given $\mathbf{x}$, $\mathbb{E}||\mathbf{x}||_0 \leq c\sum_{i=1}^{m} i^{-\alpha/2} = O(\log d)$ for $\alpha \geq 2$ and $||\mathbf{x}||_0 \leq O(m^{1-\alpha/2})$ for $\alpha \in (1, 2)$. Thus $\mathbb{E}||\mathbf{x}||_0 = o(\sqrt{m})$ for any $\mathbf{x}$ and $\alpha \in (1, +\infty)$. This assumption and derivation by Duchi et al. imply that the condition $k = o(\sqrt{m})$ is not ungrounded. The same discussion applies to $n = o(\sqrt{m})$ too. The condition $K = o(m)$ is less obvious than the other two, but we often assume a good parameter does not have any excessively large values.

We now prove the above theorem.

Proof. If we omit the conditions of Equation (4)), then the method in 4.1.1 can be used to show that Equations (6) and (7) hold, satisfying $\epsilon_{mul} = 1$ (Ng, 2004, Theorem 3.1).

Next, we apply the unbiased feature hashing defined in 4.1.2. As the expected dot-product between vectors in a reduced space is the same as the original, this does not change the expected error (Weinberger et al., 2009). There is, however, one problem: hash collisions. When a collision occurs for an input $\mathbf{x}$, for the vector converted from $\mathbf{x}$, the absolute value for some index may exceed 1. However, the error

analysis of randomized rounding requires the absolute value of any index to be bounded by 1. Hence, applying feature hashing before randomized rounding can break the subsequent analysis. Thankfully, if we use $h : \mathbb{N} \to 0, ..., m'$ such that $||\mathbf{x}||_0 = o(m')$ for any $\mathbf{x} \in \mathcal{D}$, with *high probability* (that is, $1 - o(1)$), collisions never occur. This is a well-known fact resulting from Sterling's approximation (in the following , f, g, h, i are $o(1)$ and irrelevant to the rest of this proof).

$$
\begin{aligned}
\binom{n}{k} = \frac{n!}{k!(n-k)!} &= \frac{(1+f(n))\sqrt{2\pi n}e^{-n}n^n}{k!(1+g(n-k))\sqrt{2\pi(n-k)}e^{-(n-k)}(n-k)^{n-k}} \\
&= \frac{(1+f(n))}{(1+g(n-k))} \cdot \frac{1}{k!} \cdot \left(\frac{n}{n-k}\right)^{1/2} e^{-k} \left(\frac{n}{n-k}\right)^{-k} \left(\frac{n}{n-k}\right)^{n} n^k \\
&= \frac{(1+f(n))}{(1+g(n-k))} \cdot \frac{n^k}{k!} \cdot \left(1-\frac{k}{n}\right)^{k-1/2} e^{-k} \left(1-\frac{k}{n}\right)^{-n} \\
&= \frac{(1+f(n))}{(1+g(n-k))} \cdot \frac{n^k}{k!} \cdot (1+o(1))e^{-k} \frac{1}{\left(1-\frac{k}{n}\right)^n} \\
&= \frac{(1+f(n))(1+h(k))}{(1+g(n-k))(1+i(n))} \cdot \frac{n^k}{k!} \cdot e^{-k} \cdot \frac{1}{e^{-k}} = \frac{(1+f(n))(1+h(k))}{(1+g(n-k))(1+i(n))} \cdot \frac{n^k}{k!}
\end{aligned}
$$

Using this equation, it is possible to estimate the hash collision ratio. If we assume Equation (4), $m' = c\sqrt{m}$ for some positive c satisfies the condition we mentioned above.

We then apply the Elias–Fano structure defined in Definition 4.1.3. Because this is a lossless compression method for indices, the error analysis remains the same. The remaining problem is how to store the values in $o(m)$ bits.

Finally, we apply the unbiased randomized rounding as in Definition 4.1.4. Golovin et al. (2013, Theorem 4.3) implies that if the value of each non-zero component of θ^* is bounded by some K and the number of non-zero components of any input is bounded by some k such that $K \geq 1$ and $k \geq 0$, then it is possible to derive a new parameter vector $\hat{\theta}$, with each value using $(1 + \mu + \nu)$ bits, such that $\varepsilon(\hat{\theta}) \leq \epsilon_{mul} \cdot \varepsilon(\theta^*)$, where

$$
0 \leq \epsilon_{mul} \leq 2\chi\sqrt{2\pi k} \exp\left(\frac{\chi^2 n}{2}\right), \tag{8}
$$

$\mu = \lceil \log_2 K \rceil (= o(\log m)$ if $K = o(m))$, and $\nu = -\log_2 \chi$ (so $\chi = \frac{1}{2^\nu}$). If we use $\nu = \lceil cm^{1/2} \rceil (= \Theta(\sqrt{m}))$ for some positive constant c, assuming the conditions (4) hold, $\epsilon_{mul} = f_2(m)$ with f_2 such that $f_2(m) = o(1)$. Here $f_2(m) = o(1)$ holds because $\lim_{m\to+\infty} \chi\sqrt{k} = \lim_{m\to+\infty} \sqrt{k \cdot \chi^2} = \lim_{m\to+\infty} \sqrt{o(\sqrt{m})/O(2^{m^{1/2}})} = 0$ and the same limit also applies to the $\chi^2 n$ term.

Thus, to store the values, we need $n(1 + \mu + \nu) = o(\sqrt{m}) \cdot (1 + o(\log m) + \Theta(\sqrt{m})) = o(m)$ bits in total. This completes the proof. □

4.3 Practical settings

In practical situations, several violations of the method can make it more efficient.

For example, as presented by Weinberger et al. (2009), it is possible to reduce both storage and RAM usage during training by applying feature hashing prior to training with L1-regularization. In this case, it is not clear whether the above theorem holds, as we solve the optimization problem in the reduced-dimension space rather than the original one. Nevertheless, the predictive performance of resulting models will not change significantly. Weinberger et al. (2009, Corollary 5) pointed out that the number of instances enters logarithmically into the analysis of the maximal canonical distortion. Additionally, Ng (2004) showed that the number of features m affects the number of required instances only logarithmically. Combining these results implies that, if we carefully choose informative instances for training, the distortion in the reduced space is only affected by $\log \log m$, which should be negligible. Thus, we expect the theorem to virtually hold even in this setting.

Another issue involves solving Equation (1). Several concrete optimization algorithms such as AdaGrad (Duchi et al., 2011) solve the dual problem of Equation (1), that is, $\arg\max_{\boldsymbol{\theta}} \sum_{i=0}^{T-1} \log p(y_i|\mathbf{x}_i;\boldsymbol{\theta}) - \lambda R(\boldsymbol{\theta})$ where $\lambda \propto \frac{1}{B}$ for $B > 0$. This is said to be dual because, for any $\boldsymbol{\theta}$ obtained by Equation (1) with some hyperparameter B, there is exactly one hyperparameter λ that gives the same $\boldsymbol{\theta}$. Unfortunately, it is often impossible to get the exact relationship between the hyperparameters B and λ. However, because $\lambda \propto \frac{1}{B}$, a good value may be found by trying $\lambda = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, ...$

4.4 Bitwise feature hashing

On first examination, unbiased feature hashing (Weinberger et al., 2009) may appear to be twice as costly as the naïve version (that is, always $\xi(i) = 1$ in Definition 4.1.2), as it requires two hash functions to be computed: h and ξ. Actually, by leveraging some bitwise operations, we need only one hash computation and just another three CPU cycles per feature to perform the unbiased version. Let us consider the following algorithm to convert k items of an original index–value pair to the same number of items of pairs under feature hashing. In the algorithm, `>>` denotes an arithmetic right shift (also called a signed right shift).

Data: A 2-tuple of indices and values $(I; V)$ such that $I \in \mathbb{N}^k$ and $V \in \mathbb{R}^k$ for some k.
Dimensionality m' satisfying $m' = 2^a$ for some $a \in \mathbb{N}$.
Result: A 2-tuple of indices and values $(I^*; V^*)$ such that $I^* \in \mathbb{N}^k$ and $V^* \in \mathbb{R}^k$ for some k and $0 \leq i' < m' < 2^{32}$ for any $i \in I^*$.

$MASK \leftarrow m' - 1$;
for *int* $i = 0$*;* $i < k$*;* $i \leftarrow i + 1$ **do**
  $t \leftarrow h(I_i)$;
  sign $\leftarrow (t \ \texttt{>>}\ 31) \mid 1$;
  $I_i^* \leftarrow t \ \texttt{\&}\ MASK$;
  $V_i^* \leftarrow sign \cdot V_i$;
end
return $(I^*; V^*)$;

Algorithm 1: Bitwise unbiased feature hashing

The improvements over the original version are:

1. the original version requires two hash functions, whereas our version requires only one,
2. the resulting index–value pairs are not merged—because the dot-product between two vectors is distributive ($a \cdot (b + c) = a \cdot b + a \cdot c$), there is no need to merge indices—so there is no need to use slow data structures such as hash maps for merging, and
3. compared with Bohnet (2010)'s approach, which uses a `%` (modulo) operator to gain the desired dimension, our approach uses `&` (bitwise mask), so it runs faster in common architectures. The downside is that the resulting dimension will be limited to a power of two.

5 Experimental Results

To evaluate our approach, we implemented a linear-chained conditional random field (CRF) (Lafferty et al., 2001; Sha and Pereira, 2003) and tested it on an NP chunking task. For the optimization problem, we used the AdaGrad, an online optimization algorithm, with the L1-regularization term by diagonal primal-dual subgradient update (Duchi et al., 2011).

5.1 Implementation

We implemented our CRF library with ECMAScript 6, but its core component for optimization computation was handwritten by asm.js[1], a subset of ECMAScript 6 (that is, so-called JavaScript 6). The language

[1] ".js" is not a file extension, but a part of the name of the language.

was designed mainly by Mozilla as a performance improvement technique for web platforms. Although it is conformant to ECMAScript 6, the language is actually lower-level than C, and Mozilla claims it is only at most twice as slow as native code. We used the language because, by using a JavaScript-compatible language, it is possible to embed the resulting classifier in web pages for use as a fundamental library in web systems, not only on desktop machines but also on mobile platforms.

5.2 NP chunking

μ \ ν	3	2	1
3	0.9720	0.9703	0.9621
2	0.9673	0.9681	0.9610
1	0.9470	0.9460	0.9392

Table 1: Macro-averaged F1 values under various μ and ν values for $Q\mu.\nu$ encoding.

To test our CRF implementation, following Sha and Pereira (2003), we performed an NP chunking task using the CoNLL-2000 text chunking task data (Tjong Kim Sang and Buchholz, 2000). In the shared task data, the labels B-NP and I-NP were retained, but all other labels were converted to O. Therefore, this is basically a sequential labeling problem with three labels. The features are the same as those used by Sha and Pereira (2003), except that they reformulated two consecutive labels as a new one, whereas we used the original labels. The total number of features is 1,015,621.

The training dataset consists of 8,936 instances, and the test dataset contains 2,012. Of the training data set, we used 7,148 instances (4/5) for training and the remaining 1,788 instances as development data.

For feature hashing, we used 2^{20} as the resulting dimension. There are three hyperparameters for AdaGrad: δ, η, and λ. For the first two, we simply used $\delta = \eta = 1.0$. To find a good value for λ, or the coefficient of the regularization term, we tried $\{1.0, 0.5, 0.25, 0.125, ..., \}$, and found that $\lambda = 0.5^{14} \sim 0.00006104$ gave the best results, with a macro-averaged F1 score of 0.9722. The number of active features (L0-norm) was 8,824. We also tested various μ and ν for the unbiased randomized rounding, and found that $\mu = \nu = 3$ gave a macro-averaged F1 score of 0.9720, which is slightly less than the original. In this case, the number of active features further decreased to 6,785. Overall, the size of our model was compressed to 12,745 bytes.

We compared our implementation with CRFSuite 1.2, a popular linear-chained CRF library written in C++ (Okazaki, 2007). The library has a convenient functionality that allows strings to be used as feature IDs. The downside is that the resulting model tends to be large, because it requires an alphabet to be stored. Using this library, the obtained model had a size of 27 megabytes, and achieved a macro-averaged F1 score was 0.973009.

In summary, although our method is inferior to the existing implementation by 0.001 in terms of F1 score, the size efficiency is more than 2,000 times better.

5.3 Performance of bitwise feature hashing

We also tested the performance of the bitwise feature hashing described in Section 4.4. For comparison, we prepared three functions as follows.

1. *Unbiased* feature hashing. The algorithm is similar to the one given by Bohnet (2010) but we employed two hash functions to implement the unbiased version and added values when collisions occurred, whereas Bohnet's method just overwrites the previous value. We used JavaScript's built-in object type as a dictionary (or hash map).

2. *Non-merge* feature hashing. This is almost identical to the standard version, but instead of using a dictionary (or hash map) structure, it emits the resulting index–value pairs into array structures

without merging the indices. The code was implemented without asm.js tuning, as this does not leverage bitwise operations.

3. *Bitwise* feature hashing with asm.js tuning (Section 4.4).

The environment used in this experiment was OS X 10.11.6 with 2.2 GHz Intel Core i7. The code was micro-benchmarked by Benchmark.js 2.1.0, developed by Mathias Bynens [2], under the following web browsers: Apple Safari 10.0 (with the JavaScript engine JavaScriptCore, also known as Nitro), Google Chrome 53.0.2785.143 (V8), and Mozilla Firefox 49.0.1 (SpiderMonkey). The experimental results are shown below.

	Standard	Non-merge	Bitwise
Apple Safari 10	1,108	3,426	**6,670**
Google Chrome 53	506	2,524	**4,710**
Mozilla Firefox 49	239	737	**4,446**

Table 2: Performance of each feature hashing implementation. Higher is better. Unit: 1k ops/sec.

The results in this table show that the bitwise version is more than five times faster than the standard one in all environments.

As we see, Safari was faster than the other browsers, possibly because the code was run under Apple's OS. Firefox was slightly slower than Chrome in general settings, but with asm.js enabled it achieved almost the same performance.

In conclusion, the bitwise feature hashing can offer powerful performance improvement when the task involves a number of features and processing those features is the bottleneck of the learning system.

6 Conclusions

This paper showed that pipelining compression methods for machine learning can achieve a succinct model, both theoretically and practically. Although each technique is classical and well-studied, our result is significant in that these techniques actually complement each other, leading to drastic compression without harming the predictive power. We also examined bitwise feature hashing, a subtle but powerful modification for improving processing performance. In future work, we will try to incorporate other compression techniques to build various pipeline configurations and compare the performance of each configuration.

Acknowledgements

This work was supported by CREST, Japan Science and Technology Agency (JST). We are grateful to three anonymous reviewers for their helpful comments.

[2] `https://github.com/bestiejs/benchmark.js`

References

Bernd Bohnet. 2010. Very High Accuracy and Fast Dependency Parsing is not a Contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING '10)*, pages 89–97.

Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. 2015. Compressing Neural Networks with the Hashing Trick. *Proceedings of the 32nd International Conference on Machine Learning*, pages 2285–2294, apr.

Cisco Systems. 2016. The Zettabyte Era: Trends and Analysis. Technical report.

Michael Curtiss, Iain Becker, Tudor Bosman, Sergey Doroshenko, Lucian Grijincu, Tom Jackson, Sandhya Kunnatur, Soren Lassen, Philip Pronin, Sriram Sankar, Guanghao Shen, Gintaras Woss, Chao Yang, and Ning Zhang. 2013. Unicorn: A System for Searching the Social Graph. *Proceedings of the VLDB Endowment*, 6(11):1150–1161, aug.

John Duchi and Yoram Singer. 2009. Efficient Online and Batch Learning using Forward Backward Splitting. *Journal of Machine Learning Research*, 10:2873–2898.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Peter Elias. 1974. Efficient Storage and Retrieval by Content and Address of Static Files. *Journal of the ACM*, 21(2):246–260.

Robert M. Fano. 1971. On the Number of Bits Required to Implement An Associative Memory. Technical report, Memorandum 61, Computer Structures Group, MIT, Cambridge, MA.

Kuzman Ganchev and Mark Dredze. 2008. Small Statistical Models by Random Feature Mixing. In *Proceedings of the ACL08 HLT Workshop on Mobile Language Processing*, pages 19–20.

Daniel Golovin, D. Sculley, H. Brendan McMahan, and Michael Young. 2013. Large-Scale Learning with Less RAM via Randomization. *Proceedings of the 30th International Conference on Machine Learning*, 28:325–333, mar.

Alexander Golynski, Alessio Orlandi, Rajeev Raman, and S. Srinivasa Rao. 2014. Optimal Indexes for Sparse Bit Vectors. *Algorithmica*, 69(4):906–924, aug.

Roberto Grossi and Jeffrey Scott Vitter. 2005. Compressed Suffix Arrays and Suffix Trees with Applications to Text Indexing and String Matching. *SIAM Journal on Computing*, 35(2):378–407, jan.

Masato Hagiwara and Satoshi Sekine. 2014. Lightweight Client-Side Chinese/Japanese Morphological Analyzer Based on Online Learning. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*, pages 39–43.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning bothWeights and Connections for Efficient Neural Networks. In *Advances in Neural Information Processing Systems 28*, pages 1135–1143.

Song Han, Huizi Mao, and William J. Dally. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *International Conference on Learning Representations*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289.

Stuart P. Lloyd. 1982. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137.

Andrew Y. Ng. 2004. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of the 21 st International Conference on Machine Learning*.

Daisuke Okanohara and Kunihiko Sadakane. 2007. Practical Entropy-Compressed Rank / Select Dictionary. In *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 60–70.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of conditional random fields (CRFs).

Giuseppe Ottaviano and Rossano Venturini. 2014. Partitioned Elias-Fano Indexes. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 273–282.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, number June, pages 134–141, Morristown, NJ, USA. Association for Computational Linguistics.

Ehsan Shareghi, Matthias Petri, Gholamreza Haffari, and Trevor Cohn. 2015. Compact, Efficient and Unlimited Capacity: Language Modeling with Compressed Suffix Trees. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2409–2418.

Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, and S.V.N. Vishwanathan. 2009. Hash Kernels for Structured Data. *Journal of Machine Learning Research*, 10:2615–2637.

Jeffrey Sorensen and Cyril Allauzen. 2011. Unary Data Structures for Language Models. In *INTERSPEECH 2011*, pages 1425–1428.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic Gradient Descent Training for L1-regularized Log-linear Models with Cumulative Penalty. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 477–485.

Sebastiano Vigna. 2013. Quasi-Succinct Indices. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining - WSDM '13*, pages 83–92, New York, New York, USA. ACM Press.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2009. A Succinct N-gram Language Model. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 341–344.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature Hashing for Large Scale Multitask Learning. In *Proceedings of the 26th International Conference on Machine Learning*.

Bad Company—
Neighborhoods in Neural Embedding Spaces Considered Harmful

Johannes Hellrich
Graduate School "The Romantic Model. Variation—Scope—Relevance"
Friedrich Schiller University Jena
Jena, Germany
johannes.hellrich@uni-jena.de

Udo Hahn
Jena University Language & Information Engineering (JULIE) Lab
Friedrich Schiller University Jena
Jena, Germany
http://www.julielab.de

Abstract

We assess the reliability and accuracy of (neural) word embeddings for both modern and historical English and German. Our research provides deeper insights into the empirically justified choice of optimal training methods and parameters. The overall low reliability we observe, nevertheless, casts doubt on the suitability of word neighborhoods in embedding spaces as a basis for qualitative conclusions on synchronic and diachronic lexico-semantic matters, an issue currently high up in the agenda of Digital Humanities.

1 Introduction

Distributional methods applied to large-sized, often temporally stratified corpora have markedly enhanced the methodological repertoire of both synchronic and diachronic computational linguistics and are getting more and more popular in the Digital Humanities (see Section 2.2). However, using such quantitative data as a basis for qualitative, empirically-grounded theories requires that measurements should not only be accurate, but also reliable. Only under such a guarantee, quantitative data can be assembled from different experiments as a foundation for trustful theories.

Measuring word similarity by word neighborhoods in embedding space can be used to detect diachronic shifts or domain specific usage, by training word embeddings on suited corpora and comparing these representations. Additionally, lexical items near in the embedding space to the lexical item under scrutiny can be considered as approximating its meaning at a given point in time or in a specific domain. These two lines of research converge in prior work to show, e.g., the increasing association of the lexical item *'gay'* with the meaning dimension of homosexuality (Kim et al., 2014; Kulkarni et al., 2015). Neural word embeddings (Mikolov et al., 2013) are probably the most influential among all embedding types (see Section 2.1). Yet, we gathered evidence that the inherent randomness involved in their generation affects the reliability of word neighborhood judgments and demonstrate how this hampers qualitative conclusions based on such models.

Our investigation was performed on both historical (for the time span of 1900 to 1904) and contemporary texts (for the time span of 2005 to 2009) in two languages, English and German. It is thus a continuation of prior work, in which we investigated historical English texts only (Hellrich and Hahn, 2016a), and also influenced by the design decisions of Kim et al. (2014) and Kulkarni et al. (2015) which were the first to use word embeddings in diachronic studies. Our results cast doubt on the reproducibility of such experiments where neighborhoods between words in embedding space are taken as a computationally valid indicator for properly capturing lexical meaning (and, consequently, meaning shifts).

2 Related Work

2.1 Word Embeddings

Word embeddings, i.e., low (several hundred) dimensional vector word representations encoding both semantic and syntactic information, are currently one of the most influential methods in computational

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2785–2796, Osaka, Japan, December 11-17 2016.

linguistics. The word2vec family of algorithms, developed from heavily trimmed artificial neural networks, is a widely used and robust way to generate such embeddings (Mikolov et al., 2013; Levy et al., 2015). Its skip-gram variant predicts plausible contexts for a given word, whereas the alternative continuous bag-of-words variant tries to predict words from contexts; we focus on the former as it is generally reported to be superior (see e.g., Levy et al. (2015)). There are two strategies for managing the huge number of potential contexts a word can appear in. Skip-gram hierarchical softmax (SGHS) uses a binary tree to more efficiently represent the vocabulary, whereas skip-gram negative sampling (SGNS) updates only a limited number of word vectors during each training step. SGNS is preferred in general, yet SGHS showed slight benefits in some reliability scenarios in our prior investigations (Hellrich and Hahn, 2016a).

There are two sources of randomness involved in the training of neural word embeddings: First, the random initialization of all word vectors before any examples are processed. Second, the order in which these examples are processed. Both can be replaced by deterministic alternatives,[1] yet this would simply replace a random distortion with a fixed one, thus providing faux reliability only useful for testing purposes. A range of other word embedding algorithms was inspired by word2vec, either trying to avoid the opaqueness stemming from its neural network heritage (GloVe; still using random initialization, see Pennington et al. (2014)) or adding capabilities, like using syntactic information during training (Levy and Goldberg, 2014) or modeling multiple word senses (Bartunov et al., 2016; Panchenko, 2016). Levy et al. (2015) created SVD_{PPMI}, a variant of the classical pointwise mutual information co-occurrence metric (see e.g., Manning and Schütze (1999, pp.178–183)), by transferring pre-processing steps and hyper-parameters uncovered by the development of these algorithms, and reported similar or slightly better performance than SGNS on evaluation tasks. It is conceptually not affected by reliability problems, as there is no random initialization or relevant processing order.

Word embeddings capture both syntactic and semantic information (and arguably also social biases, see Bolukbasi et al. (2016)) in vector form and can thus be evaluated by their ability to calculate the similarity of two words and perform analogy-based reasoning; there exist several other evaluation methods and more test sets than discussed here, see e.g., Baroni et al. (2014). Mikolov et al. (2013) provide an analogy test set for measuring performance as the percentage of correctly calculated analogies for test cases such as the frequently cited *'king'–'queen'* example (see Section 3). Word similarity is evaluated by calculating Spearman's rank coefficient between embedding-derived predictions and a gold standard of human word similarity judgments. Finkelstein et al. (2002) developed a widely used test set with 353 English word pairs,[2] a similar resource for German with 350 word pairs was provided by Zesch and Gurevych (2006).[3] Recent work cautions that performance on such tasks is not always predictive for performance in down-stream applications (Batchkarov et al., 2016).

2.2 Diachronic Application

Word embeddings can be used rather directly for tracking semantic changes, namely by measuring the similarity of word representations generated for one word at different points in time—words which underwent semantic shifts will be dissimilar with themselves. These models must either be trained in a continuous manner where the model for each time span is initialized with its predecessor (Kim et al., 2014; Hellrich and Hahn, 2016b), or a mapping between models for different points in time must be calculated (Kulkarni et al., 2015; Hamilton et al., 2016). The first approach cannot be performed in parallel and is thus rather time-consuming, if texts are not subsampled. We nevertheless discourage using samples instead of full corpora, as we observed extremely low reliability values between different samples (Hellrich and Hahn, 2016a). Word embeddings can also be used in diachronic studies without any kind of mapping to track clusters of similar words over time and, thus, model the evolution of topics (Kenter et al., 2015) or compare neighborhoods in embedding space for preselected words (Jo, 2016). Besides temporal variations, word embeddings can also used to analyze geographic ones, e.g., the distinction between US American and British English variants (Kulkarni et al., 2016). Most of these studies were

[1] In fact, in some implementations, yet not in ours, vectors are initialized via a deterministic process.

[2] www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

[3] www.ukp.tu-darmstadt.de/data/semantic-relatedness/german-relatedness-datasets/

performed with algorithms from the `word2vec` family, respectively `GloVe` in Jo (2016), and are thus likely to be affected by the same systematic reliability problems on which we focus here. Only Hamilton et al. (2016) used SVD_{PPMI} in some of their very recent experiments and showed it to be adequate for exploring historical semantics.

The Google Books Ngram corpus (GBN; Michel et al. (2011), Lin et al. (2012)) is used in most of the studies we already mentioned, including our current study and its predecessor (Hellrich and Hahn, 2016a). It contains about 6% of all books published between 1500 and 2009 in the form of n-grams (up to pentagrams), together with their frequency for each year. This corpus has often been criticized for its opaque sampling strategy, as its constituent books remain unknown and can be shown to form an unbalanced collection (Pechenick et al., 2015). GBN is multilingual, with its English part being subdivided into regional segments (British, US) and topic categories (general language and fiction texts). Diachronic research focuses on the English Fiction part, with the exception of some work relating to German data (Hellrich and Hahn, 2016b).

3 Evaluation Methods

Reliability, in this study, is judged by training three identically parametrized models for each experiment and by comparing the n next neighbors (by cosine distance) for each word modeled by the experiments with a variant of the Jaccard coefficient (Manning and Schütze, 1999, p.299). The 3-dimensional array $W_{i,j,k}$ contains words ordered by closeness (i) for a word in question (j) according to an experiment (k). The reliability r for a specific value of n ($r@n$) is defined as the magnitude of the intersection of similar words produced by all three experiments with a rank of n or lower, averaged over all t words modeled by these experiments and normalized by n, which is the maximally achievable score for this value of n:

$$r@n := \frac{1}{t * n} \sum_{j=1}^{t} || \bigcap_{k=1}^{3} \{W_{1 \leq i \leq n,j,k}\} || \tag{1}$$

Accuracy, in this study, is measured considering two different approaches—analogy and similarity. The *analogy* approach uses the English test set developed by Mikolov et al. (2013) by calculating the percentage of correct analogies made by a `word2vec` model. It contains groups of four words connected via the analogy relation '::' and the similarity relation '$\sim$', as exemplified by the expression *'king'* $\sim$ *'queen'* :: *'man'* $\sim$ *'woman'*. The *similarity* approach covers both English and German by calculating Spearman's rank correlation coefficient between the similarity judgments made by a `word2vec` model for a word pair (e.g., *'bread'* and *'butter'*) and the human judgment thereof (Finkelstein et al., 2002; Zesch and Gurevych, 2006). Pairs containing words not modeled for the time span in question, such as the at that time non-existent *'FBI'* in the early 20th century, are simply ignored. All three test sets are based on contemporary language and current world knowledge and might thus not fully match the requirements for historical texts, yet are also used for these due to the lack of a suitable alternative. Accuracy values were calculated independently for each of the three identically parametrized models and subsequently averaged, but resulting deviations were negligible.

4 Experimental Set-up

4.1 Corpus

Our experiments[4] were performed on the German part and the English Fiction part of the GBN; the latter is known to be less unbalanced than the general English part (Pechenick et al., 2015). Both corpus splits differ in size and contain mainly contemporary texts (from the past fifty years), as is evident from Figure 1; note the logarithmic axis and the negative impact of both World Wars on book production. Following Kulkarni et al. (2015), we trained our models on all 5-grams occurring during five consecutive years for the two time spans,[5] 1900–1904 and 2005–2009; the number of 5-grams[6] for each time span

[4]Code used in experiments available from `https://github.com/hellrich/coling2016`

[5]This is due to computational demands, e.g., using 8 parallel processes on a server with Intel Xeon E5649@2.53Ghz processors five days were necessary to complete each of ten training epochs for SGNS with 2005–2009 English Fiction data.

[6]Note that we treat 5-grams with k occurrences during the same time span as k different 5-grams.

is listed in Table 1. The two languages share a similar number of 5-grams for 1900–1904, yet not for 2005–2009. 5-grams from both corpus parts were lower cased for training. The German part was not only taken as is, but also orthographically normalized using the CAB service (Jurish, 2013).[7] We incorporated this step because major changes in German orthography occurred during the 20th century, an issue that could hamper diachronic comparisons, e.g., archaic *'Gemüth'* (in English: "mind, emotional disposition") became modern *'Gemüt'*. Table 1 shows the resulting reduction in the number of types, bringing the morphologically richer German to levels below English (yet this reduction is in line with the respective corpus sizes).

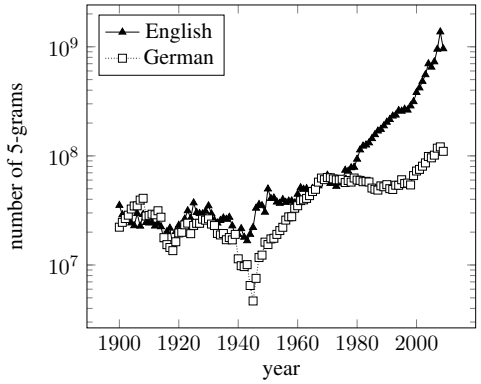


Figure 1: Number of 5-grams per year (on the logarithmic y-axis) in the English Fiction part and the German part of the GOOGLE BOOKS NGRAM corpus.

Language	Time Span	5-grams	Types
English	1900–1904	143M	80k
English	2005–2009	4,658M	216k
German	1900–1904	135M	111k
Normalized German			72k
German	2005–2009	546M	243k
Normalized German			179k

Table 1: Number of 5-grams and lemma types contained in the English Fiction part and the German part of the GOOGLE BOOKS NGRAM corpus for the two time spans used in our experiments.

4.2 Training

We used the PYTHON-based GENSIM[8] implementation of `word2vec` to independently train word embeddings for each time span with 200 dimensions, a context window of 4 (limited by the 5-gram size), a minimum frequency of 10, and 10^{-5} as the threshold for downsampling frequent words. We processed the full subcorpora for each time span, due to the extremely low reliability values between samples we observed in previous investigations (Hellrich and Hahn, 2016a). We tested both SGNS with 5 noise words and SGHS training strategies and trained for 10 iterations, saving the resulting embeddings after each epoch. During each epoch the learning rate was decreased from 0.025 to 0.0001. The averaged cosine values between word embeddings before and after an epoch are used as a convergence measure c (Kim et al., 2014; Kulkarni et al., 2015). It is defined for a vocabulary with n words and a matrix W containing word embedding vectors (normalized to length 1) for words i from training epochs e and e-1:

$$c := \frac{1}{n} \sum_{i=1}^{n} W_{i,e} \cdot W_{i,e-1} \tag{2}$$

We also define Δc, the change of c during subsequent epochs e-1, as another convergence criterion:

$$\Delta c := c_e - c_{e-1} \tag{3}$$

5 Results

Table 2 shows the performance of the systems trained according to the settings described in Section 4.2, as measured by similarity accuracy and top-1 reliability (see below for other cut-offs). We make the following observations:

[7] `www.deutschestextarchiv.de/demo/cab/`
[8] `www.radimrehurek.com/gensim/`

1. Both accuracy and reliability are higher for SGNS than for SGHS for all tested combinations of languages and time spans, if 10 training epochs are used.
2. If only one training epoch is used—as in many other experimental set-ups reported in the literature—there is only little difference in accuracy between SGNS and SGHS, but SGHS is clearly better in terms of reliability.
3. Accuracy is higher for 2005–2009 than for the 1900–1904 interval, with the exception of non-normalized German (which can most likely be explained by the temporal currency of the test sets).
4. Normalization of German data slightly decreases reliability, yet increases accuracy.

Training Scenario			Top-1 Reliability			Similarity Accuracy		
Language	Time Span	Embeddings	1 Epoch	5 Epochs	10 Epochs	1 Epoch	5 Epochs	10 Epochs
English Fiction	1900–1904	SGNS	0.11	0.33	0.40	0.45	0.51	0.51
		SGHS	0.23	0.33	0.33	0.46	0.45	0.45
	2005–2009	SGNS	0.36	0.54	0.57	0.58	0.58	0.57
		SGHS	0.36	0.39	0.38	0.55	0.52	0.52
German	1900–1904	SGNS	0.20	0.47	0.54	0.45	0.56	0.56
		SGHS	0.34	0.43	0.42	0.48	0.49	0.47
	2005–2009	SGNS	0.31	0.50	0.53	0.51	0.54	0.54
		SGHS	0.34	0.38	0.36	0.49	0.48	0.47
Normalized German	1900–1904	SGNS	0.19	0.45	0.52	0.47	0.55	0.57
		SGHS	0.32	0.42	0.42	0.47	0.48	0.48
	2005–2009	SGNS	0.30	0.48	0.52	0.54	0.59	0.60
		SGHS	0.33	0.37	0.36	0.51	0.52	0.52

Table 2: Accuracy and reliability among top-1 words for threefold repetition of different training scenarios after completing 1, 5 and 10 training epochs, respectively.

We also measured analogy accuracy for the English Fiction data sets, and observed no negative effect of multiple training epochs, yet a more pronounced gap between training methods, e.g., 36% of all analogies were correct for SGNS and only 27% for SGHS after one epoch on 1900–1904 data.

In the following, we further explore system performance as influenced, e.g., by word frequency, word ambiguity and the number of training epochs. For German, we focus on the normalized version due to the overall similar performance and suitability for further applications.

Influence of Neighborhood Size. Reliability at different top-n cut-offs is very similar for all languages and time spans under scrutiny, confirming previous observations in Hellrich and Hahn (2016a) and strengthening the suggestion to use only top-1 reliability for evaluation. Figure 2 illustrates this phenomenon with an SGNS trained on 1900–1904 English Fiction data. We assume this to be connected with the general decrease in `word2vec` embedding utility for high values of n already observed by Schnabel et al. (2015).

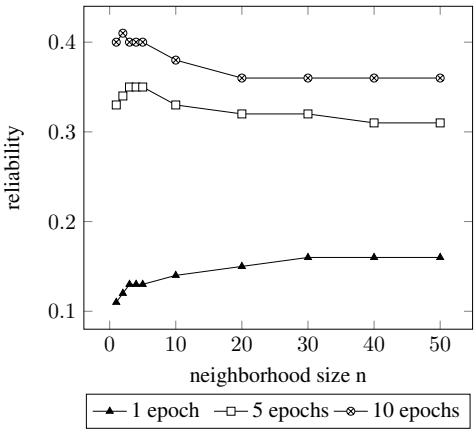


Figure 2: Effect of neighborhood size parameter n in reliability calculation for SGNS embeddings trained on 1900–1904 English Fiction data.

Influence of Word Frequency. Figures 3 and 4 depict the influence of word frequency (as percentile ranks) for English, as well as orthographically normalized German. Negative sampling is overall more reliable, especially for words with low or medium frequency. Word frequency has a less pronounced effect on reliability for German and negative sampling is again preferable, especially for low or medium frequency words. The 21 English

words reported to have undergone traceable semantic changes in prior work[9] are all frequent with percentiles between 89 and 99—for such high-frequency words hierarchical softmax performs similarly or even slightly better. The relatively low reliability for medium-frequency English words, as compared to German ones, could be caused by a peculiar pattern of word co-occurrences, illustrated in Figures 5 and 6 for 1900–1904 English Fiction, respectively normalized German. Medium-frequency English words have fewer co-occurrences with low-frequency words than German ones, which might result in a lack of specific contexts for these words during training and thus hamper embedding quality.

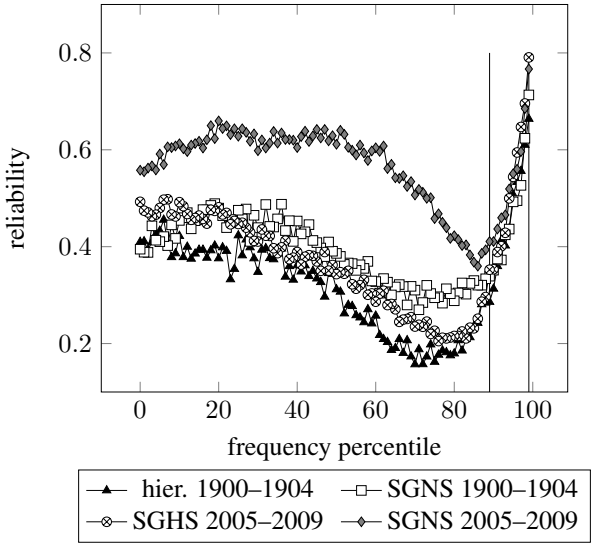


Figure 3: Influence of frequency percentile on reliability for models trained for 10 epochs on English Fiction data from 1900–1904 and 2005–2009. Words reported to have changed their semantics during the 20th century fall into the frequency range marked by the vertical lines.

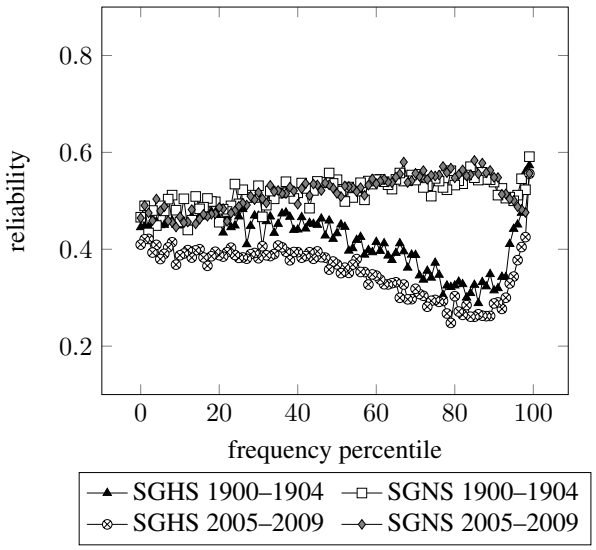


Figure 4: Influence of frequency percentile on reliability for models trained for 10 epochs on orthographically normalized German data from 1900–1904 and 2005–2009.

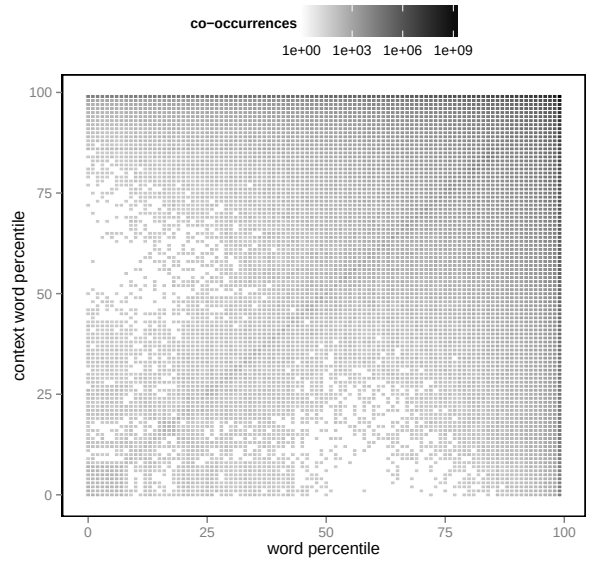


Figure 5: Number of co-occurrences (indicated by shade; only values above mode) between words and context words per frequency percentile for English Fiction 1900–1904 data.

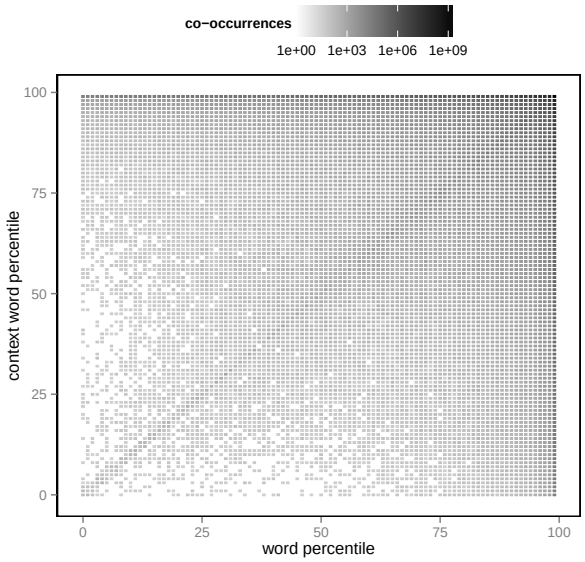


Figure 6: Number of co-occurrences (indicated by shade; only values above mode) between words and context words per frequency percentile for normalized German 1900–1904 data.

[9]Kulkarni et al. (2015) compiled the following list based on prior work (Wijaya and Yeniterzi, 2011; Gulordava and Baroni, 2011; Jatowt and Duh, 2014; Kim et al., 2014): *card*, *sleep*, *parent*, *address*, *gay*, *mouse*, *king*, *checked*, *check*, *actually*, *supposed*, *guess*, *cell*, *headed*, *ass*, *mail*, *toilet*, *cock*, *bloody*, *nice* and *guy*.

Influence of Word Ambiguity. Entries in lexical databases, such as WORDNET[10] (Fellbaum, 1998) and its German counterpart GERMANET[11] (Lemnitzer and Kunze, 2002), can be employed to approximate the effect of word ambiguity on reliability. The number of synsets a word belongs to (i.e., the number of its senses) seems to be positively correlated with top-1 reliability for English, as shown in Figure 7, whereas orthographically normalized German is less affected by ambiguity as Figure 8 reveals. This counter-intuitive effect for English seems to be caused by the low ambiguity of infrequent words—results become more uniform, if analysis is limited to high frequency words (e.g., 90th frequency percentile or higher).

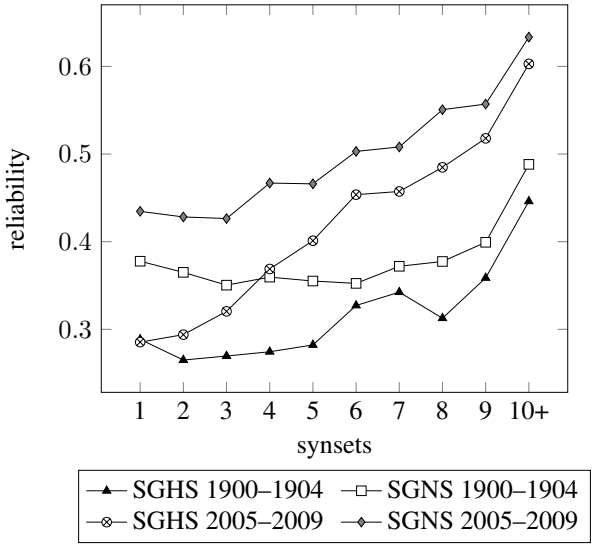


Figure 7: Influence of ambiguity (measured by the number of WORDNET synsets) on top-1 reliability for models trained for 10 epochs on English Fiction data from 1900–1904 and 2005–2009.

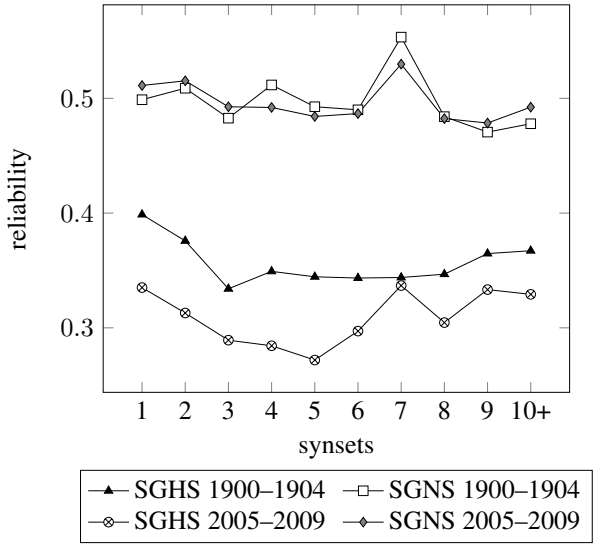


Figure 8: Influence of ambiguity (measured by the number of GERMANET synsets) on top-1 reliability for models trained for 10 epochs on orthographically normalized German data from 1900–1904 and 2005–2009.

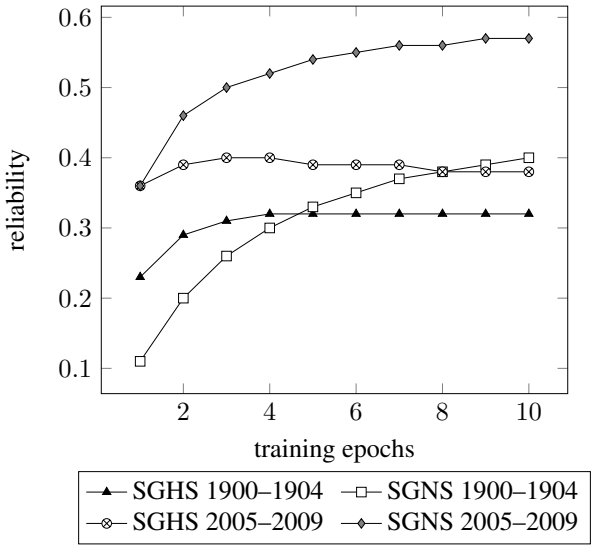


Figure 9: Top-1 reliability as influenced by the number of training epochs, for English Fiction data relative to the 1900–1904 and 2005–2009 time slices.

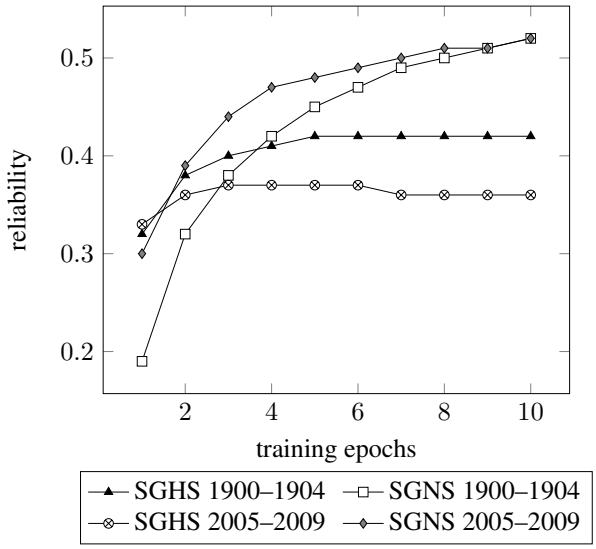


Figure 10: Top-1 reliability as influenced by the number of training epochs, for orthographically normalized German data relative to the 1900–1904 and 2005–2009 time slices.

Influence of the Number of Training Epochs. Model reliability and accuracy depend on the number of training epochs, as shown in Figures 9 and 10 for English and normalized German, respectively. For

[10]We used WORDNET 3.0 and the API provided by the Natural Language Toolkit (NLTK): `www.nltk.org`

[11]We used GERMANET 11.0 and the PYGERMANET API: `https://pypi.python.org/pypi/pygermanet`

both languages and time spans negative sampling outperforms hierarchical softmax, if training lasts for a sufficient number of epochs. The number of necessary epochs for negative sampling to become superior seems to be linked to both language and corpus size, as it is lower for 2005–2009 than for 1900–1904 data. While reliability continues to increase for each subsequent epoch under negative sampling, there are clear diminishing returns and even regression under hierarchical softmax.

To test for potential overfitting effects, we analyzed similarity accuracy as influenced by the number of training epochs (some values are already given in Table 2). Figures 11 and 12 show the results for English and orthographically normalized German, respectively. Note that accuracy is assessed on a test set for modern-day language, and can thus not be considered a fully valid yardstick. Accuracy behaves similar to reliability, as under the negative sampling condition it clearly profits from multiple training epochs. This effect is more pronounced for smaller corpora; the biggest corpus (i.e., English Fiction 2005–2009) shows a slight regression in accuracy after more than 5 training epochs.

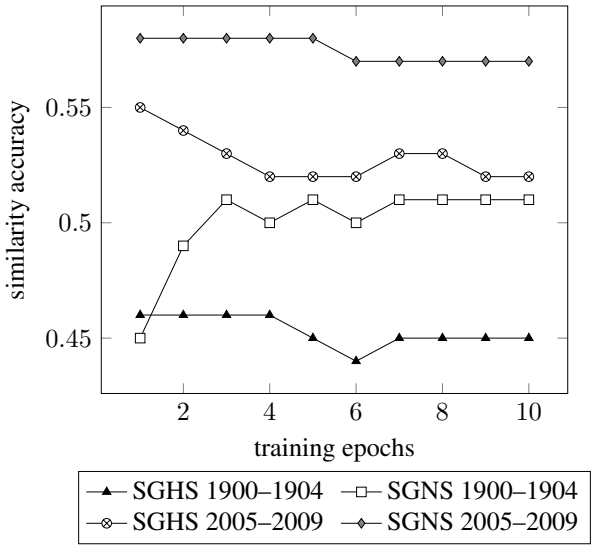


Figure 11: Similarity accuracy as influenced by the number of training epochs for English Fiction data data relative to the 1900–1904 and 2005–2009 time slices. Error bars are not displayed on purpose due to constant values for each training method.

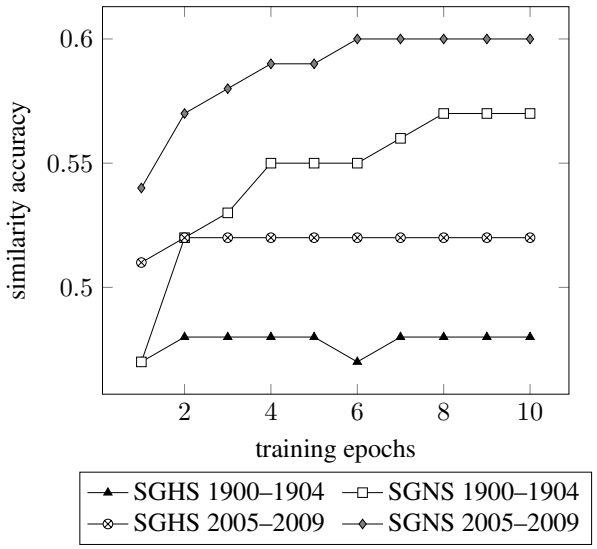


Figure 12: Similarity accuracy as influenced by the number of training epochs for orthographically normalized German data relative to the 1900–1904 and 2005–2009 time slices. Error bars are not displayed on purpose due to constant values for each training method.

Conclusions. Both reliability and accuracy point towards negative sampling with 4 to 6 training epochs (6 being better for smaller and 4 being better for larger corpora) as the optimal training regime for all tested combinations of languages and time spans (implicitly, this is also a test on largely varying corpus sizes, see Table 1). Such a training scheme yields models with high reliability without losses in accuracy (that would indicate overfitting). Figure 13 shows Δc, i.e., the difference of the convergence measure c (Equations (2) and (3) averaged over all three models) between subsequent epochs, for both German and English data from the intervals 1900–1904 and 2005–2009. Few changes occur after 4–6 epochs, which could be alternatively expressed as a Δc of about 0.003. The convergence criterion proposed by Kulkarni et al. (2015), i.e., $c = 0.9999$, was never reached (this observation might be explained by Kulkarni et al.'s decision not to reset the learning rate for each training epoch, as was done by us and Kim et al. (2014)).

SVD_{PPMI}, which are conceptually not bothered by the reliability problems we discussed here, were not a good fit for the hyperparameters we adopted from Kulkarni et al. (2015). Hamilton et al. (2016) reports similarity accuracy superior to SGNS, whereas for our set-up results in pretests were about 10 percent points worse than skip-gram embeddings, e.g., only 0.35 for 1900–1904 English Fiction.

Finally, to want to illustrate how this reliability problem affects qualitative conclusions. In Table 3 we provide some examples in which three negative sampling models for 1900–1904 English Fiction did not agree on the closest neighbor for words in question (mostly drawn from the list in Footnote 9). The most

inconsistent word neighborhoods are provided for *'romantic'* which is connected to *'lazzaroni'*,[12] *'fanciful'* and *'melancholies'*. This holds despite the high frequency (94th percentile) and moderate ambiguity (5 synsets) of the target item *'romantic'*.

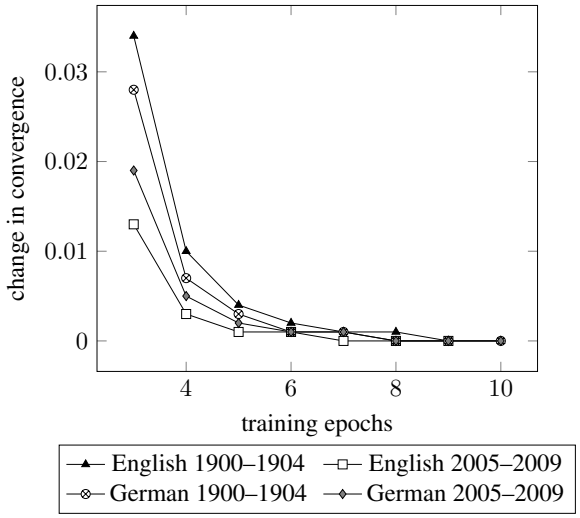


Figure 13: Change of averaged convergence measurement c between each epoch and its predecessor for models of orthographically normalized German and English Fiction trained with negative sampling on the 1900–1904 and 2005–2009 time slices. Values for epochs 1 and 2 would be one magnitude higher and are thus not displayed.

Word	Disputed Closest Neighbor
romantic	lazzaroni, fanciful, melancholies
parent	child, child, mother
mouse	mice, rat, cat
checked	checking, check, checking
check	cheque, checked, cheque
guess	reckon, reckon, suppose
headed	headedness, haired, haired
ass	atheist, fool, fool
toilet	ironing, dressing, dressing
cock	cocks, arty, hen
bloody	mistyken, mistyken, wreaks
nice	stunner, fine, fine

Table 3: A sample list of target lexical items for which three identically parametrized systems (trained with negative sampling on 1900–1904 English Fiction data) disagreed on the closest neighbor. Examples are mostly drawn from the list of the twenty-one aforementioned words (see Footnote 9) that were claimed to have undergone changes during the 20th century.

6 Discussion

Our investigation into the accuracy and reliability of skip-gram word embeddings shows even the most reliable systems too often provide inconsistent word neighborhoods. This carries unwarranted potential for erroneous conclusions on a word's semantic evolution as was shown, e.g., for the lexical item *'romantic'* and English Fiction texts from the 1900–1904 time slice. We are thus skeptical about using word neighborhoods in skip-gram embedding space to adequately capture natural languages' lexical semantics (for English and German, at least). While we found some mitigation strategies, i.e., training for multiple epochs or using our convergence criterion of $\Delta c \lesssim 0.003$, we assume SVD_{PPMI} to be conceptually superior. Future work might try to provide general guidelines for proper hyperparameter selection for SVD_{PPMI}, especially regarding complete temporal slices of the GBN (Hamilton et al. (2016) used samples). Alternatively, training several identically parametrized SGNS/SGHS models and combining them into an ensemble might constitute an easy way to reduce the reliability problems we described, yet at the price of exorbitant computational costs.

Acknowledgments

This research was conducted within the Graduate School *"The Romantic Model. Variation – Scope – Relevance"* (`http://www.modellromantik.uni-jena.de/?lang=en`) supported by grant GRK 2041/1 from the *Deutsche Forschungsgemeinschaft (DFG)*.

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. *Don't count, predict!* A systematic comparison of context-counting vs. context-predicting semantic vectors. In Daniel Marcu, Kristina Toutanova, and Hua Wu, editors, *ACL 2014 — Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. Baltimore, Maryland, USA, June 22-27, 2014*, volume 1: Long Papers, pages 238–247, Stroudsburg/PA. Association for Computational Linguistics (ACL).

[12] A historical group of lower-class persons from Naples ("lazzarone, n", 2016).

Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry P. Vetrov. 2016. Breaking sticks and ambiguities with adaptive skip-gram. In Arthur Gretton and Christian C. Robert, editors, *AISTATS 2016 — Proceedings of the 19th International Conference on Artificial Intelligence and Statistics. Cadiz, Spain, May 7-11, 2016*, number 51 in JMLR Workshop and Conference Proceedings, pages 130–138.

Miroslav Batchkarov, Thomas Kober, Jeremy Reffin, Julie Weeds, and David J. Weir. 2016. A critique of word similarity as a method for evaluating distributional semantic models. In Omer Levy, Felix Hill, Anna Korhonen, Kyunghyun Cho, Roi Reichart, Yoav Goldberg, and Antoine Bordes, editors, *Proceedings of the 1st Workshop on Evaluating Vector Space Representations for NLP @ ACL 2016. Berlin, Germany, August 12, 2016*, pages 7–12, Stroudsburg/PA. Association for Computational Linguistics (ACL).

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Quantifying and reducing stereotypes in word embeddings. In James Faghmous, Rayid Ghani, Matt Ghee, Gideon S. Mann, Aleksandra Mojsilovic, and Kush R. Varshney, editors, *Proceedings of the Workshop on #Data4Good: Machine Learning in Social Good Applications @ ICML 2016. New York City, NY, USA, June 24, 2016*, pages 41–45.

Christiane Fellbaum, editor. 1998. WORDNET*: An Electronic Lexical Database*. MIT Press, Cambridge/MA; London/England.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131, January.

Kristina Gulordava and Marco Baroni. 2011. A distributional similarity approach to the detection of semantic change in the GOOGLE BOOKS NGRAM corpus. In Sebastian Padó and Yves Peirsman, editors, *GEMS 2011 — Proceedings of the Workshop on GEometrical Models of Natural Language Semantics @ EMNLP 2011. Edinburgh, UK, July 31, 2011*, pages 67–71, Stroudsburg/PA. Association for Computational Linguistics (ACL).

William L. Hamilton, Jure Leskovec, and Daniel Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In Antal van den Bosch, Katrin Erk, and Noah A. Smith, editors, *ACL 2016 — Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. Berlin, Germany, August 7-12, 2016*, volume 1: Long Papers, pages 1489–1501, Stroudsburg/PA. Association for Computational Linguistics (ACL).

Johannes Hellrich and Udo Hahn. 2016a. An assessment of experimental protocols for tracing changes in word semantics relative to accuracy and reliability. In Beatrice Alex and Nils Reiter, editors, *LaTeCH 2016 — Proceedings of the 10th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities @ ACL 2016, Berlin, Germany, August 11, 2016*, pages 111–117, Stroudsburg/PA. Association for Computational Linguistics (ACL).

Johannes Hellrich and Udo Hahn. 2016b. Measuring the dynamics of lexico-semantic change since the German Romantic period. In *Digital Humanities 2016 — Conference Abstracts of the 2016 Conference of the Alliance of Digital Humanities Organizations (ADHO). 'Digital Identities: The Past and the Future'. Kraków, Poland, 11-16 July 2016*, pages 545–547.

Adam Jatowt and Kevin Duh. 2014. A framework for analyzing semantic change of words across time. In *JCDL '14 — Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries. London, U.K., September 8-12, 2014*, pages 229–238, Piscataway/NJ. IEEE Computer Society Press.

Eun Seo Jo. 2016. Diplomatic history by data. Understanding Cold War foreign policy ideology using networks and NLP. In Maciej Eder and Jan Rybicki, editors, *Digital Humanities 2016 — Conference Abstracts of the 2016 Conference of the Alliance of Digital Humanities Organizations (ADHO). 'Digital Identities: The Past and the Future'. Kraków, Poland, 11-16 July 2016*, pages 582–585.

Bryan Jurish. 2013. Canonicalizing the Deutsches Textarchiv. In Ingelore Hafemann, editor, *Proceedings of Perspektiven einer corpusbasierten historischen Linguistik und Philologie. Internationale Tagung des Akademienvorhabens "Altägyptisches Wörterbuch" an der Berlin-Brandenburgischen Akademie der Wissenschaften. Berlin, Germany, December 12-13, 2011*, number 4 in Thesaurus Linguae Aegyptiae, pages 235–244. Berlin-Brandenburgische Akademie der Wissenschaften.

Tom Kenter, Melvin Wevers, Pim Huijnen, and Maarten de Rijke. 2015. Ad hoc monitoring of vocabulary shifts over time. In James Bailey and Alistair Moffat, editors, *CIKM '15 — Proceedings of the 24th ACM International Conference on Information and Knowledge Management. Melbourne, Australia, October 19-23, 2015*, pages 1191–1200, New York/NY, USA. Association for Computing Machinery (ACM).

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. In Cristian Danescu-Niculescu-Mizil, Jacob Eisenstein, Kathleen R. McKeown, and Noah A. Smith, editors, *Proceedings of the Workshop on Language Technologies and Computational Social Science @ ACL 2014. Baltimore, Maryland, USA, June 26, 2014*, pages 61–65, Stroudsburg/PA. Association for Computational Linguistics (ACL).

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi, editors, *WWW '15 — Proceedings of the 24th International Conference on World Wide Web. Florence, Italy, May 18-22, 2015*, volume Technical Papers, pages 625–635, New York/NY. Association for Computing Machinery (ACM).

Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. 2016. Freshman or fresher? Quantifying the geographic variation of language in online social media. In Krishna P. Gummadi, Markus Strohmaier, Eric Gilbert, Michael Macy, and Claudia Wagner, editors, *ICWSM-16 — Proceedings of the 10th International AAAI Conference on Web and Social Media. Cologne, Germany, May 17-20, 2016*, pages 615–618, Palo Alto/CA. Association for the Advancement of Artificial Intelligence (AAAI), AAAI Press.

"lazzarone, n". 2016. In *OED Online*. Oxford University Press. `http://www.oed.com/view/Entry/106565` (accessed June 16, 2016).

Lothar Lemnitzer and Claudia Kunze. 2002. GERMANET: Representation, visualization, application. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Angel Martin Municio, Daniel Tapias, and Antonio Zampolli, editors, *LREC 2002 — Proceedings of the 3rd International Conference on Language Resources and Evaluation. Las Palmas, Canary Islands, Spain, 27 May - June 2, 2002*, pages 1485–1491, Paris. European Language Resources Association (ELRA).

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In Daniel Marcu, Kristina Toutanova, and Hua Wu, editors, *ACL 2014 — Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. Baltimore, Maryland, USA, June 22-27, 2014*, volume 2: Short Papers, pages 302–308, Stroudsburg/PA. Association for Computational Linguistics (ACL).

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Yuri Lin, Jean-Baptiste Michel, Erez Lieberman Aiden, Jon Orwant, William Brockman, and Slav Petrov. 2012. Syntactic annotations for the Google Books Ngram Corpus. In Min Zhang, editor, *ACL 2012 — Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics. Jeju Island, Korea, July 10, 2012*, volume System Demonstrations, pages 169–174, Stroudsburg/PA. Association for Computational Linguistics (ACL).

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge/MA.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182, January.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR 2013 — Workshop Proceedings of the International Conference on Learning Representations. Scottsdale, Arizona, USA, May 2-4, 2013*.

Alexander Panchenko. 2016. Best of both worlds: Making word sense embeddings interpretable. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan E. J. M. Odijk, and Stelios Piperidis, editors, *LREC 2016 — Proceedings of the 10th International Conference on Language Resources and Evaluation. Portorož, Slovenia, 23-28 May 2016*, pages 2649–2655, Paris. European Language Resources Association (ELRA-ELDA).

Eitan Adam Pechenick, Christopher M. Danforth, and Peter Sheridan Dodds. 2015. Characterizing the GOOGLE BOOKS corpus: Strong limits to inferences of socio-cultural and linguistic evolution. *PLoS One*, 10(10):e0137041, October.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GLOVE: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *EMNLP 2014 — Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. Doha, Qatar, October 25-29, 2014*, pages 1532–1543, Stroudsburg/PA. Association for Computational Linguistics (ACL).

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In Lluís Márquez, Chris Callison-Burch, and Jian Su, editors, *EMNLP 2015 — Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal, 17-21 September 2015*, pages 298–307, Red Hook/N.Y. Association for Computational Linguistics (ACL).

Derry Tanti Wijaya and Reyyan Yeniterzi. 2011. Understanding semantic change of words over centuries. In Sergej Sizov, Stefan Siersdorfer, Philipp Sorg, and Thomas Gottron, editors, *DETECT '11 — Proceedings of the 2011 International Workshop on DETecting and Exploiting Cultural diversiTy on the Social Web @ CIKM 2011. Glasgow, U.K., October 24, 2011*, pages 35–40, New York/N.Y. Association for Computing Machinery (ACM).

Torsten Zesch and Iryna Gurevych. 2006. Automatically creating datasets for measures of semantic relatedness. In John Nerbonne and Erhard W. Hinrichs, editors, *Proceedings of the Workshop on Linguistic Distances @ COLING-ACL 2006. Sydney, Australia, 23 July 2006*, pages 16–24, Stroudsburg/PA. Association for Computational Linguistics (ACL).

Implementing a Reverse Dictionary, based on word definitions, using a Node-Graph Architecture

Sushrut Thorat
Center for Mind/Brain Sciences
University of Trento
Rovereto, TN 38068, Italy
sushrut.thorat94@gmail.com

Varad Choudhari
Department of Computer Science
Rajarambapu Institute of Technology
Islampur, MH 415414, India
varad.choudhari@gmail.com

Abstract

In this paper, we outline an approach to build graph-based reverse dictionaries using word definitions. A reverse dictionary takes a phrase as an input and outputs a list of words semantically similar to that phrase. It is a solution to the Tip-of-the-Tongue problem. We use a distance-based similarity measure, computed on a graph, to assess the similarity between a word and the input phrase. We compare the performance of our approach with the Onelook Reverse Dictionary and a distributional semantics method based on *word2vec*, and show that our approach is much better than the distributional semantics method, and as good as Onelook, on a 3k lexicon. This simple approach sets a new performance baseline for reverse dictionaries.[1]

1 Introduction

A *forward dictionary* (FD) maps words to their definitions. A *reverse dictionary* (RD) (Sierra, 2000), also known as an *inverse dictionary*, or *search-by-concept dictionary* (Calvo et al., 2016), maps phrases to single words that approximate the meaning of those phrases. In the Oxford Learner's Dictionary[2], one definition of 'brother' is 'a boy or man who has the same mother and father as another person'. A reverse dictionary will map not only this phrase to 'brother', but also phrases such as 'son of my parents'. A reverse dictionary is primarily a solution to the *Tip of the Tongue* problem (Schwartz and Metcalfe, 2011) which regularly plagues people when they want to articulate their thoughts. It can also be used in the treatment of *word selection anomic aphasia* (Rohrer et al., 2008), a neurological disorder in which patients can identify objects and understand semantic properties, but cannot name the object or produce one word to describe the concept.

Popular languages let us create a multitude of phrases from a finite number of words. A static database of all possible phrases is unbound, if not infinite (Ziff, 1974). We need to dynamically compute the output word(s) from the input phrase. To map a phrase to a word, we have to compute the meanings of the phrase and the word (Fromkin et al., 2011). The *principle of compositionality* states that the meaning of an expression is composed of the meaning of its parts and the way they are combined structurally. The most basic parts, words, can be defined in terms of word definitions, references to objects, or lexical relations and hierarchies. Computing the meaning of a phrase requires constructing the constituent tree and recognising the relationship between the constituents, which is a complex, open problem.

Compositional Distributional Semantic Models have been used towards computing the meaning of a phrase, with partial success (Baroni, 2013; Erk, 2012). Recurrent neural networks show promise in learning continuous phrase representations. They are used towards syntactic parsing beyond discrete categories such as NP and VP, in an attempt to capture phrasal semantics (Socher et al., 2010). A recent work has used neural language embedding models (RNNs with LSTMs) to understand phrases by embedding dictionaries (Hill et al., 2015). But it doesn't perform exceptionally better than the existing reverse dictionaries (OneLook, etc.)

[1] The test data and a demo code can be found at: https://github.com/novelmartis/RD16demo

[2] www.oxfordlearnersdictionaries.com; Accessed: February, 2016

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2797–2806, Osaka, Japan, December 11-17 2016.

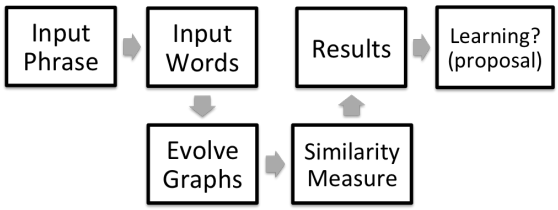


Figure 1

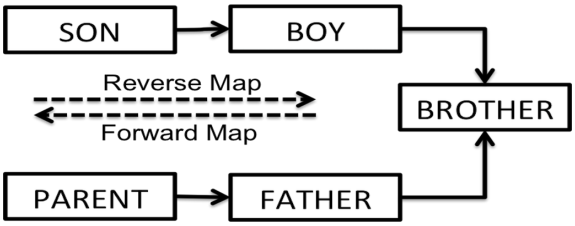


Figure 2

Figure 1: Operation of the Reverse Dictionary. The graphs' connectivities are based on the *reverse map*, a concept we will introduce shortly.
Figure 2: Each solid arrow indicates a *in the definition of* relation. This is the reverse map leading the phrase 'Son of my parents' to the word 'brother', after extraction of the input words. Note that this is one of the many sets of connections to all words on the graph from that phrase.

If we are to ignore the ordering of words in a phrase, the performance of such a system would not be maximal. But we could then work just with well-studied lexical relations. Research into building reverse dictionaries has mainly focused on lexical relations than the structural or contextual combination of words. The attempts in defining a *similarity measure* between words have been summarised in (Mihalcea et al., 2006). An attempt towards situational understanding and contextual selection of words can be seen in (Granger, 1982). The creation of WordNet (Miller, 1995) boosted the use of lexical relations and hierarchies, as in (Dutoit and Nugues, 2002; El-Kahlout and Oflazer, 2004; Shaw et al., 2013; Méndez et al., 2013; Calvo et al., 2016). Most of these approaches extract *input words* from the input phrase and expand their search through lexical relations and hierarchies, towards a similarity measure between the phrase and the words. (Zock and Schwab, 2008) take inspiration from human word synthesis and implement a user-guided search to the desired word. All the mentioned approaches have achieved partial success, but the problem stays unsolved.

We explore the possibility of using word definitions towards establishing semantic similarity between words and phrases. Definitions are dense sources of semantic information about the words (which makes it difficult to extract information from them without using exact syntactic structures such as constituent trees), and we employ them exclusively in our approach. We assume that the significance of the meaning of a word to a definition is proportional to its frequency throughout the definitions in the FD. We extract the meaning from the *content words* (Fromkin et al., 2011) contained in the phrase. We split the input phrase into these component *input words*, implement a graph-search through related words (relation through definition), and use a *distance-based* similarity measure to compute words which are representative of the meaning of the input phrase. A graph encodes the word relations in its connectivity matrix, on which the similarity measures are computed. We now detail our approach.

2 System Description

The block diagram of the operation of the RD is depicted in Fig. 1. We now discuss the concept of the reverse map, central to the structure of our graph, and the process of obtaining the connectivity matrix underlying our graph.

2.1 The Reverse Map

In a *forward map*, words branch out to the words that are contained in their definitions. In a *reverse map*, words branch out to the words whose definitions they are contained in. An example of a reverse map[3] is shown in Fig. 2.

If the input phrase is a definition, a search depth of one (branching out from the words of the input phrase to the definitions they are contained in) of the reverse map will lead to the required word. A search depth beyond one provides us with semantic information about the words whose definitions encompass

[3]Based on the definitions from the Oxford Learner's Dictionary.

or relate to the concepts that encompass or relate to the input words, and so on. Increasing search depth obscures the relationship between words, which is the basis for the definition of the similarity measure we employ. A reverse map suggests semantic convergence in a shallow search, although the convergence might occur on multiple words, which is acceptable as they might be semantically-similar words. Intuitively, a forward search seems to 'fragment' the meaning of the input word, and is expected to perform worse than the reverse search in defining word relationships in our approach.

2.2 Connectivity Matrix of the Reverse Map

The steps in the construction of the connectivity matrix, based on the reverse map, are as follows. Our inputs are a forward dictionary, a list of functional words, and a lemmatizer. We process the forward dictionary to retain content words in their base form. We then construct the forward-linked list, transform it into a back-linked list, and then construct the back-linked connectivity matrix. We can also construct the forward-linked connectivity matrix in a similar way.

2.2.1 Processing the Forward Dictionary

A forward dictionary (FD) is a two-dimensional list. The rows in the first column contain the words, and the rows in the second column contain the corresponding definitions. We reduce all words in column one to their lemmas, their base forms[4]. We then delete all the *functional words*[5] (Fromkin et al., 2011), and the corresponding definitions in column two. For our purposes, we pool all the definitions of a particular word into a single cell, parse them through the lemmatizers and delete all the functional words within them. We term the resulting list the *forward-linked list*. We now generate the *back-linked matrix*.

2.2.2 The Back-Linked Matrix

We number the words in column one of the forward-linked list in the alphabetical order (word-id). We substitute all the words in column two by their word-ids. We then create a list which points a word (written in column one) to the words whose definitions it lies in (written in column two). We call this list the back-linked list.

We then generate the *back-linked matrix* (BLM) which represents the connections (weights) between the nodes (words, in this case). If j is a word-id in the second column of the back-linked list, and i is the word-id in the corresponding row, but in column one, then the element (j, i) in the BLM is turned to 1. After iterating through all the elements in the list, we set the diagonals of the BLM to zero (no self-connections).

We will see in section 3 that many words in a dictionary do not appear in any definition, and so cannot contact all words in the wordlist through the reverse map. But we would like to obtain the similarity measure between any two words in the wordlist. As a simple measure in ensuring complete connectivity, we build a *mixed back-linked matrix* (mBLM) which has forward-linked connections for words that do not have sufficient back-linked connections (they cannot connect to all the words in the lexicon, through the reverse map). We will assess in section 4 the change in performance by the inclusion of the said forward links.

2.3 The Node-Graph Architecture

Each word is represented by a node in the graph. The connections between the nodes in our graph are given by the BLM. We create a graph for each *input word* (we obtain these from the input phrase by parsing it through the same operations as the definitions), and activate the input word node in its corresponding graph (the explanation is provided in section 5). We then simultaneously evolve the graphs one hop at a time. We are, in effect, expanding the tree of words to be able to effectively implement the similarity measure. If we implement n hops, we term it a *n-layered search* (n is also termed as the 'depth' of search).

[4]Using the *pattern lemmatizer* (Smedt and Daelemans, 2012) and *wordnet morphy* (Bird et al., 2009).

[5]The functional words were obtained from Higgins, 2014: http://myweb.tiscali.co.uk/wordscape/museum/funcword.html

2.4 The Similarity Measure

We use a distance-based measure of similarity.

We define the distance $d_{Y,X}$ from a word X to another word Y as the depth of search required to evolve a state with only $S_X = 1$, to the first state with $S_Y = 1$. Note that $d_{Y,X} \neq d_{X,Y}$.

We calculate the frequencies of appearances, $\{\nu_Z\}$ throughout definitions, for all words $\{Z\}$ in the wordlist.

We define the similarity measure $E_{W,P}$ of a word W to an input phrase P containing the input words $\{P_i\}$ as:

$$\mathrm{E}_{W,P} = \frac{\sum_i \left(\nu_{P_i} \times d_{W,P_i}\right)^{-1}}{\sum_i \nu_{P_i}^{-1}}$$

We weighted the inverse of the distances between the words with the inverse of the frequencies of the input words. So, the similarity measure includes a measure of 'semantic importance' of each input word in the input phrase. We calculate the similarity measure of each word to the input phrase, and output the words in the decreasing order of similarity. As every word is connected to every other word in the reverse map given apt search depth, the similarity measure becomes important in finding relevant output. Our similarity measure states, the smaller the distances from the input words, the more similar is the word to the input phrase. Minimal distances ensure that the semantic similarity remains meaningful.

2.5 System Summary

The user inputs a phrase. Input (content) words are extracted from the phrase. Graphs are generated for each input word, and in each graph, the state of the node corresponding to the input word is turned to 1. The graphs are evolved to the *maximum non-redundant* search depth (see section 3). The similarity measure, to the input phrase, is computed for every word in the lexicon, and the words are ranked according to their similarity measures, leading to the output.

3 Graph exploration

We construct BLMs and mBLMs based on the processed[6] Oxford 3000 wordlist[7], and a BLM for the entire WordNet (Miller, 1995) lexicon (WL). We use the Oxford Learner's dictionary (OLD), Merriam-Webster dictionary[8] (MW), and WordNet (WN) as forward dictionaries for the Oxford 3000 wordlist, and WordNet for the WordNet lexicon (WL). We also build a BLM and a mBLM by pooling definitions (Fusion BLM) from the three forward dictionaries, for the 3k wordlist, to check the effect of using multiple dictionaries on performance.

Before we move on to analyse the performances, let's look deeper into the connectivity matrices we generated. All the BLMs and mBLMs are sparse[9]. We use the *compressed sparse row* format from SciPy (Jones et al., 2001) to store and process our matrices.

In the 3k wordlist case, the number of connections in the Fusion BLM is greater than the BLMs built with individual FDs. In Fig. 3(a), we see that there are 190 words which cannot excite the entire graph through the Fusion BLM. So, we build a mBLM, as proposed in section 2.2.2, and ensure complete connectivity of the graph, as seen in Fig. 3(b). As all words can connect to all other words at the most in 9 steps, a search depth greater than 9 would be redundant when we use the Fusion BLM. The *maximum non-redundant* search depths for the individual BLMs are as follows: 11 (OLD), 9 (WN), and 11 (MW).

The maximum required search depth for the WordNet lexicon BLM is 19. 53, 711 words out of 82, 603 do not map to any word in the reverse map. Those words are infrequent in the language and are not used to define other words. Fig. 4(b) depicts the distribution of the number of back-linked connections from the words in the reverse map for the 80k WL BLM ($\mu = 7.81$, $\sigma = 62.86$, max $= 6163$), as compared to the distribution for the 3k WN BLM ($\mu = 18.10$, $\sigma = 36.14$, max $= 615$) in Fig. 4(a). The huge number

[6]Words which appeared in Oxford Learner's dictionary definitions, but were not part of the wordlist, were added to the wordlist for consistency. The modified wordlist contains 3107 words, and is referred to as 3k, in this paper.

[7]http://www.oxfordlearnersdictionaries.com/about/oxford3000

[8]Accessed: February, 2016

[9]Sparsity (proportion of 0's in the matrices): 0.99 (3k Fusion BLM), and 0.99 (WordNet lexicon BLM)

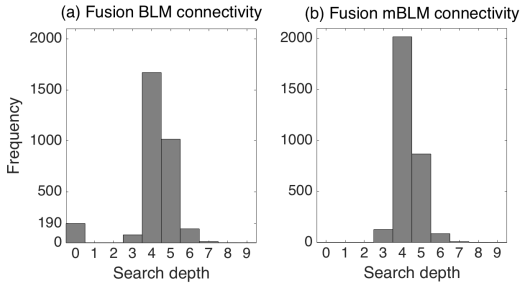


Figure 3

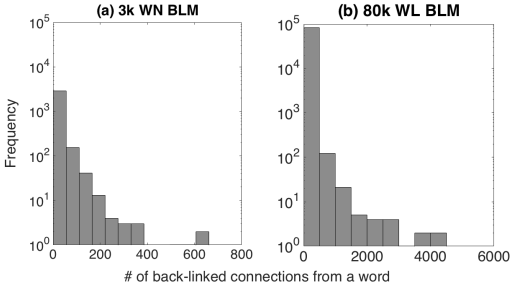


Figure 4

Figure 3: Distribution of the minimum search depth required by a word to excite the entire graph. If a word is not able to do that, a value of zero is assigned to its minimum search depth.
Figure 4: Distribution of the number of back-linked connections, for each word, in the reverse map.

of backward-linked connections for some words in 80k WL BLM would confound the accuracy of the similarity measures, and a drop in performance is expected.

4 Performance Analysis

The only available online reverse dictionary is the Onelook Reverse Dictionary (Beeferman, 2003), with which we will compare our algorithm's performance. Onelook is a commercial application, and its architecture is proprietary. We know that it indexes 1061 dictionaries and resources such as Wikipedia and WordNet. The lexicon of Onelook is much bigger than 3k. In the performance comparison, we state the performance with (termed as 'corr') and without adapting the outputs to the 3k lexicon.

We also compare our approach with a distributional semantics method, based on *word2vec* which represents words as vectors in a linear structure that allows analogical reasoning. In that vector space, the vector 'king + woman - man' will have a high similarity with the vector for 'queen' (Mikolov et al., 2013a; Mikolov et al., 2013b). We average the vector representations of input words, and search the word vectors most similar to the resulting vector (cosine similarity). This is an established method of building phrase representations from word representations (Mitchell and Lapata, 2010). The performance of such an approach[10] is shown in Table. 1 (as W2V).

4.1 Performance Test

The reverse dictionary outputs multiple candidate words. We introduced users to the concept of the reverse dictionary and asked them to generate phrases they would use to get to a given word, if they would have forgotten the word but retained the concept. 25 such users generated 179 phrases, a sample of which is presented in Figure. 5. The performance is gauged by the ranks of the words in the outputs of their user-generated phrases[11].

We also test all the approaches on one-line definitions for the 179 words, obtained from the Macmillian Dictionary[12].

4.2 Performance results

Example runs of the RD, using the 3k Fusion mBLM, are presented in Fig. 6. The distributions of ranks, for the various BLMs/mBLMs (whichever has greater % of ranks under 100 for each case), *word2vec*, and Onelook, are stated in Table. 1. Onelook did not provide any outputs for 18 phrases out of the 179 user-generated phrases, and 72 out of the 179 definitions from the Macmillan dictionary. Instead of

[10]Based on the implementation of $word2vec$ by Daniel Rodriguez at https://github.com/danielfrg/word2vec, trained on a corpus with 15.8 million words, and a vocabulary of 98k.

[11]An input phrase can have multiple semantically similar words. Analysing the semantic quality of each output would be the ideal test. This could be done using a function of the sum of the ranks of each output weighted with their distances (in a high-dimensional semantic space such as $word2vec$) from the target word. However, previous approaches have used just the rank of the target word (which is nevertheless a good indicator of performance), and here we do the same.

[12]Accessed: May, 2016

Words	Phrases
Variation	A change or changes between two or more things
Attractive	Something that is catchy
Plus	The operation used to increase
Church	Place to meet god
Possession	Taking full control over a thing

Figure 5

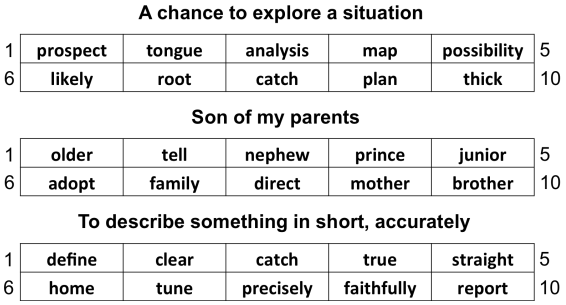


Figure 6

Figure 5: Sample user-generated phrases, used for testing the performance of the RD.
Figure 6: The first 10 outputs obtained using the 3k Fusion mBLM (n=9), for three input phrases.

Test Type →	Macmillian Word Definitions (179)			User Concept Descriptions (179)		
Evaluation → Models ↓	**Accuracy** @1/10/100	Rank Median	Rank σ	**Accuracy** @1/10/100	Rank Median	Rank σ
Onelook	.19/.41/.65	5	24	.04/.21/.40	10	26
Onelook, *corr**	.20/.46/.68	3	20	.07/**.26**/.52	13	30
W2V	.01/.06/.20	23	30	.01/.05/.18	34	28
W2V, *corr**	.02/.11/.29	21	26	.01/.08/.26	21	29
Chance, 3k	$10^{-4}/10^{-3}$/.03	50	29	$10^{-4}/10^{-3}$/.03	50	29
Fusion, FLM	.02/.10/.21	12	28	.01/.07/.22	16	21
Fusion, mBLM	.25/**.55**/**.84**	4	22	**.10**/.23/**.53**	14	26
OLD, mBLM	**.26**/.52/.78	4	23	.04/.17/.43	14	25
WN, BLM	.08/.27/.54	11	26	.06/.18/.41	14	26
MW, mBLM	.17/.39/.63	5	20	.05/.20/.43	15	25
WL, 80k	.03/.15/.36	18	26	.05/.11/.24	14	25
WL, *corr**	.07/.26/.52	10	25	.07/.18/.35	10	23

Table 1: Performance of the various models. Accuracy @n is the fraction of the phrases with the rank of the target word less than or equal to n, in their outputs. σ is the standard deviation. Only the phrases with target words having ranks less than 100 were considered in calculating the median and variance. The 3k cases (OLD, WN, MW, Fusion) were evaluated at a search depth of 11, and the 80k case (WL) at a search depth of 19. **corr* indicates the cases where the outputs were truncated to fit in the 3k lexicon, for fair comparison. (Note: Accuracy - higher is better; Rank median - lower is better; Rank variance - lower is better.)

considering these as failures, we factor out these phrases while evaluating Onelook. The performance of all approaches is significantly better than chance, as seen through the comparison of performance with 'Chance' which represents the expected values of performance for random rank assignments to the target words[13] (considering the 3k lexicon). The cases of interest are highlighted in the table.

All the 3k cases using a BLM/mBLM have a higher performance than the 3k Fusion forward-linked matrix (FLM). Fusion of the individual 3k BLMs yields better performance. The 3k Fusion BLM performs at least as well as Onelook. The use of mBLMs is fruitful as they increase the performance in some cases. The performance does not change much across search depth as seen in Table. 2, suggesting that our approach works well even at a shallow search. Deeper search is required only when a phrase is semantically vague or non-specific, and markedly different from dictionary definitions. Both our ap-

[13]The expected value of the accuracy @k, over random rank assignments, is given by: $\sum_{n=0}^{P_r} \frac{n}{P_r} \cdot \frac{^{P_r}C_n k^n (N-k)^{P_r-n}}{N^{P_r}} = \frac{k}{N}$, where P_r is the number of test phrases, and N is the size of the lexicon.

Accuracy ↓	$n = 1$	$n = 2$	$n = 3$	$n = 10$
@1	.08	.07	**.10**	**.10**
@10	**.25**	.22	.23	.23
@100	**.55**	.52	.53	.53

Table 2: Performance across search depth (n) for the Fusion mBLM case. The output becomes stable beyond a search depth of 3. The search depth at which the output becomes stable varies with the BLMs.

proach and Onelook outperform the W2V approach. We conclude that our approach works well with a 3k wordlist. Although the ranks' median and variance are indicative of the performance (hit rate, and robustness), they are marred by the accuracies, so we do not use them in our inferences.

However, the performance drops significantly when the entire processed WordNet lexicon (WL, 80k) is the FD. The words that lie in the definitions of other words are a small subset of the WL wordlist. As seen in Fig. 4, there are 163 words in the WL wordlist which map to more than 500 words in the reverse map. Therefore, the distances of multiple words to the input words are similar, obscuring the semantic content of the similarity measure. This is a potential limitation of our approach, for which there is no trivial fix.

We also assessed the performance of the Fusion mBLM on the 200 test phrases used in (Hill et al., 2015). The size of their lexicon is 66k. We cannot upscale the outputs of our 3k cases to 66k, so a direct fair comparison with their results is not possible. However, we can downscale the outputs of Onelook (on the 200 phrases) to 3k and compare with it, thus providing an indirect comparison with the approach used by Hill et al. The @1/10/100 accuracies of the Fusion mBLM are .16/.39/.62. But 33 target words do not lie in the 3k lexicon. The accuracies excluding the corresponding phrases are **.19**/**.46**/**.74**. The @1/10/100 accuracies of the Onelook (scaled to 3k) are .08/.21/.30. But 101 phrases do not return any outputs. The accuracies excluding those phrases are **.16**/**.42**/**.61**. The accuracies of Onelook and the RNN approaches in Hill et al. are equivalent. We thus conclude that the performance of our approach is at least as good as the RNN approaches, on a 3k lexicon.

5 Recommendations

The graph structure opens up a semantic dimension by letting us mutate the level of significance a word has in a definition, through the connectivity matrix. We can introduce this information in the similarity measure by scaling the weights of the connections between the words with distances equal to *one*. The definitions provided in the dictionary cannot populate the new dimension. One could consider the use of semantic rules, or lexical relations, or user feedback. Such a learning algorithm could use further exploration.

There are multiple points in our approach which could use either improvement or emphasis. We use multiple graphs for calculating the similarity measure. This is done because we do not want the distance of a word from an input word to be a function of all the input words. Using *Spiking Neural Networks* (Ghosh-Dastidar and Adeli, 2009), we could implement the similarity measure using a single graph by frequency tagging the distances from each input word, although it isn't clear how much advantage it would confer in terms of performance.

A matrix of pair-wise distances between all words could be used to evaluate the similarity measures, instead of evolving a graph. Such a matrix won't be sparse, and in the case of a 80k lexicon would be 50 gigabytes in size (compared to 10 megabytes in CSR sparse format for the BLM), making it impractical to deploy the algorithm on mobile devices. Execution time and memory requirement are not a problem for our approach. Our approach is an easy and computationally cheap method of implementing semantic search with a graph, which performs at least as well as the Onelook reverse dictionary.

The performance drops significantly when the WordNet 80k lexicon is used (the mBLM doesn't help). Use of multiple forward dictionaries might boost the performance, as in the case of Fusion mBLM, but as mentioned in section 4.2, the branching factor of the graph is too high, obscuring the similarity measure. Although this might make the approach impractical, it does serve as a new baseline. A simple approach

like ours can rival the performance of sophisticated algorithms used by Onelook and (Hill et al., 2015), suggesting that the information being retrieved by those algorithms is somewhat basic. This calls for methods which could significantly outperform a simple approach like ours, towards encoding deeper phrasal semantics.

Dealing with multi-word expressions isn't straightforward in our approach. We separate all words in the input phrase towards implementing our similarity measure. Detecting multi-word expressions would require recursive parsing of the phrase, something which is more suited to recurrent neural network-based approaches (Hill et al., 2015). This isn't a major concern for our task though, as the input phrase is supposed to be a simple description of the concept in mind, in which case the user is more likely to input 'to die' than 'kick the bucket'. Multi-word expressions are also rarely used to define words or other multi-word expressions. So, they could be treated as one node with no back-linked connections but with multiple forward-linked connections (the definition of that expression), and thus be encompassed in our approach as outputs, but not as inputs (which we do in the 80k WordNet case).

6 Concluding Remarks

We reported the construction of a reverse dictionary based on a node-graph architecture, which derives its semantic information exclusively from dictionary definitions. The approach works at least as well as the Onelook reverse dictionary and a RNN-based approach described in (Hill et al., 2015), on a lexical size of 3k words, but the performance deteriorates, to below Onelook's, when scaled to a lexicon with 80k words. The performance still stays significant (as compared to the 'Chance'), and greater than a forward map approach. Furthermore, this approach can be generalised to any language given an appropriate forward dictionary, lemmatizer, and a list of functional words.

Recent distributional approaches use vector representations for word meaning, derived through similarities gauged by the occurrences and co-occurrences of words in a large corpus (Erk, 2012). The performance of one of these approaches, known as *word2vec* (Mikolov et al., 2013a; Mikolov et al., 2013b), on our test is poor, as seen in Table. 1 (under 'W2V'). The performance suggests that phrasal semantics doesn't necessarily follow a linear additive structure. Indeed, researchers have been trying to find other mathematical structures and approaches which would be suitable for phrasal semantics (Baroni and Zamparelli, 2010; Socher et al., 2011), but with partial success and on specific types of phrases.

A class of Artificial Neural Networks (ANNs), called Recurrent Neural Networks (RNNs) are being used for tasks such as machine translation (Cho et al., 2014) and generating natural image captions (Karpathy and Fei-Fei, 2015), among others (Zhang and Zong, 2015). These 'deep' networks are not trained on, or to obtain, discrete syntactic categories such as NP and VP. Instead they are provided with the inputs and expected outputs (task-dependent) while training. The learning paradigm generates features (often incomprehensible in terms of classical linguistics) to effectively implement the given task[14], which seems to be better than using predetermined features. (Hill et al., 2015) use such a network to implement a reverse dictionary, and it performs at least as well as Onelook. The performance is noteworthy, but the fact that a simple approach like ours can rival it suggests that the RNN-based approaches require further research before doing for reverse dictionaries (and phrasal semantics, in general) what Convolutional Neural Networks (CNNs) did for visual object classification (Chatfield et al., 2014).

It seems that the focus on constituent trees and the structural combination of words cannot be compromised upon. RNNs might be the way forward, in this regard, as they could develop properties encompassing and surpassing those classical linguistic features.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

[14]"The Unreasonable Effectiveness of Recurrent Neural Networks" by Andrej Karpathy - http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Marco Baroni. 2013. Composition in distributional semantics. *Language and Linguistics Compass*, 7:511–522.

Doug Beeferman. 2003. Onelook reverse dictionary. http://onelook.com/reverse-dictionary.shtml.

Slaven Bilac, Wataru Watanabe, Taiichi Hashimoto, Takenobu Tokunaga, and Hozumi Tanaka. 2004. Dictionary search based on the target word description. In *Proc. of the Tenth Annual Meeting of The Association for Natural Language Processing (NLP2004)*, pages 556–559.

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

Hiram Calvo, Oscar Mndez, and Marco A. Moreno-Armendriz. 2016. Integrated concept blending with vector space models. *Computer Speech & Language*.

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Dominique Dutoit and Pierre Nugues. 2002. A lexical database and an algorithm to find words from definitions. In *ECAI*, pages 450–454.

Ilknur Durgar El-Kahlout and Kemal Oflazer. 2004. Use of wordnet for retrieving words from their meanings. In *Proceedings of the global Wordnet conference, GWC2004*, pages 118–123.

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

Victoria Fromkin, Robert Rodman, and Nina Hyams. 2011. *An Introduction to Language*. Wadsworth, Cengage Learning, 9 edition.

Samawoy Ghosh-Dastidar and Hojjat Adeli. 2009. Spiking neural networks. *International Journal of Neural Systems*, 19(04):295–308.

Richard H. Granger. 1982. Scruffy text understanding: design and implementation of 'tolerant' understanders. In *Proceedings of the 20th annual meeting on Association for Computational Linguistics, ACL'82*, pages 157–160.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2015. Learning to understand phrases by embedding the dictionary. *arXiv preprint arXiv:1504.00548*.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001. SciPy: Open source scientific tools for Python. [Online; accessed 2016-03-16].

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185.

Oscar Méndez, Hiram Calvo, and Marco A. Moreno-Armendáriz. 2013. A reverse dictionary based on semantic analysis using wordnet. In *Proceedings of the Mexican International Conference on Artificial Intelligence, MICAI 2013*, pages 275–285.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st national conference on Artificial intelligence, AAAI'06*, volume 1, pages 775–780.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A. Miller. 1995. Wordnet: A lexical database for english. In *Communications of the ACM*, volume 38, pages 39–41.

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

Jonathan D. Rohrer, William D. Knight, Jane E. Warren, Nick C. Fox, Martin N. Rossor, and Jason D. Warren. 2008. Word-finding difficulty: a clinical analysis of the progressive aphasias. *Brain*, 131:8–38.

Bennett L. Schwartz and Janet Metcalfe. 2011. Tip-of-the-tongue (tot) states: retrieval, behavior, and experience. *Mem Cognit.*, 39(5):737–749.

Bennett L. Schwartz. 1999. Sparkling at the end of the tongue: The etiology of tip-of-the-tongue phenomenology. *Psychonomic Bulletin and Review*, 6(3):379–393.

Ryan Shaw, Anindya Datta, Debra VanderMeer, and Kaushik Dutta. 2013. Building a scalable database-driven reverse dictionary. *IEEE Transactions on Knowledge and Data Engineering*, 25(3):528–540.

Gerardo Sierra. 2000. The onomasiological dictionary: a gap in lexicography. In *Proceedings of the Ninth EURALEX International Congress, EURALEX 2000*, pages 223–235.

Tom De Smedt and Walter Daelemans. 2012. Pattern for python. *Journal of Machine Learning Research*, 13:2063–2067.

Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Jiajun Zhang and Chengqing Zong. 2015. Deep neural networks in machine translation: An overview. *IEEE Intelligent Systems*, 15.

Paul Ziff. 1974. The number of english sentences. In *Foundations of Language*, volume 11, pages 519–532. Springer.

Michael Zock and Didier Schwab. 2008. Lexical access based on underspecified input. In *Proceedings of the Workshop on Cognitive Aspects of the Lexicon*, COGALEX '08, pages 9–17.

Is an Image Worth More than a Thousand Words? On the Fine-Grain Semantic Differences between Visual and Linguistic Representations

Guillem Collell
Computer Science Department
KU Leuven
3001 Heverlee, Belgium
gcollell@kuleuven.be

Marie-Francine Moens
Computer Science Department
KU Leuven
3001 Heverlee, Belgium
sien.moens@cs.kuleuven.be

Abstract

Human concept representations are often grounded with visual information, yet some aspects of meaning cannot be visually represented or are better described with language. Thus, vision and language provide complementary information that, properly combined, can potentially yield more complete concept representations. Recently, state-of-the-art distributional semantic models and convolutional neural networks have achieved great success in representing linguistic and visual knowledge respectively. In this paper, we compare both, visual and linguistic representations in their ability to capture different types of fine-grain semantic knowledge—or attributes—of concepts. Humans often describe objects using attributes, that is, properties such as shape, color or functionality, which often transcend the linguistic and visual modalities. In our setting, we evaluate how well attributes can be predicted by using the unimodal representations as inputs. We are interested in first, finding out whether attributes are generally better captured by either the vision or by the language modality; and second, if none of them is clearly superior (as we hypothesize), what type of attributes or semantic knowledge are better encoded from each modality. Ultimately, our study sheds light on the potential of combining visual and textual representations.

1 Introduction

Vision and language capture complementary information that humans automatically integrate in order to build mental representations of concepts. Certain concepts or properties of objects cannot be explicitly visually represented while, at the same time, not all the properties are easily expressible with language. For example, there are clearly visual differences between cats and dogs although these are not easy to describe with language. Recent advances in deep learning had led to breakthroughs in learning unimodal representations (a.k.a. embeddings) in both, computer vision (CV) and natural language processing (NLP) (LeCun et al., 2015). However, the automatic integration of visual and linguistic modalities is still a challenging—and usually task-dependent—problem that has gained increasing popularity within the NLP and CV communities. Lately, several studies have achieved reasonable success in integrating visual and linguistic representations, showing improvement over the unimodal baselines in simple linguistic tasks such as concept similarity (Lazaridou et al., 2015; Kiela and Bottou, 2014; Silberer and Lapata, 2014)—which is only possible if vision and language encode complementary knowledge.

In this paper we do not tackle the problem of *how* to integrate both modalities, but instead, we systematically study *what* type of fine-grain semantic knowledge is encoded in each modality, shedding light on the potential benefit of combining vision and language. By fine-grain semantics we refer to the recognition of different types of *attributes* or properties (e.g., shape, function, sound, etc.) that concrete nouns might exhibit. A recent study by Rubinstein et al. (2015) evidenced that state-of-the-art linguistic-only representations do not succeed at capturing all types of attributes equally well. Here, we extend their work into the multimodal domain by comparing the performance between visual and linguistic representations at encoding different types of attributes. In contrast with Rubinstein et al. (2015)'s unimodal research, here we aim to answer two different research questions. First, whether either vision or

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2807–2817, Osaka, Japan, December 11-17 2016.

language provide a superior ground for capturing fine-grain attributes; and second, if none of them is clearly superior—as we hypothesize—what type of attributes are better captured by each modality. To the best of our knowledge, no studies have systematically compared vision and language in these terms. Ultimately, this work provides insight on building better representations of concepts, which in turn is essential towards improving automatic language understanding.

The rest of the paper is organized as follows. In the next section we review and discuss related work. In Section 3 we describe our design for evaluating the success of visual and text embeddings in encoding attributes. Next, we present and discuss our experimental results. Finally, in conclusions and future work, we summarize our findings and suggest future lines of research.

2 Related Work

2.1 Unimodal Representations

Convolutional neural networks (CNN) have rapidly become the state-of-the-art approach in computer vision (Krizhevsky et al., 2012). To some extent, CNN algorithms emulate human visual perception, in which the learning occurs at different levels of abstraction—or layers of a network. On the language side, distributional models (DMs) have been employed for learning semantic representations a long time ago. These are based on the distributional hypothesis: *Words which are similar in meaning occur in similar contexts* (Rubenstein and Goodenough, 1965). Recently, neural-based distributional models or word embeddings (Mikolov et al., 2013; Pennington et al., 2014) have achieved great success, rapidly replacing the old DMs (Turney et al., 2010) based on word co-occurrence counts (Baroni et al., 2014). Instead of counting words, neural-based DMs capture words co-occurrences by trying to predict the context given a word (skip-gram) or by trying to predict a word given its context (CBOW). Alternative approaches such as generative probabilistic models that learn the probability distribution of a vocabulary word in a context window as a latent variable have also been proposed (Deschacht et al., 2012; Deschacht and Moens, 2009).

2.2 Multimodal Representations

There exist certain properties of perceptible objects that are poorly captured by language. For example, everyone can easily tell from an image whether a face is attractive, yet if one has to describe with language what properties make a face attractive will certainly struggle. Recently, CNN-based computer vision models have achieved reasonable success in the task of recognizing attractiveness (Rothe et al., 2015). Furthermore, psychological research evidences that human concept formation is strongly grounded in visual perception (Barsalou, 2008). All this suggests that linguistic representations can benefit from visual grounding. In this direction, recent studies have shown that multimodal embeddings are often able to outperform text-only embeddings in simple semantic tasks such as concept similarity (Lazaridou et al., 2015; Silberer and Lapata, 2014; Kiela and Bottou, 2014) or categorization (i.e., grouping objects into categories such as "fruit", "furniture", etc.) (Silberer and Lapata, 2014). Several ways of combining representations from both modalities have been devised so far—yet an exhaustive review of them would deviate from the target of this work. To name a few, Kiela and Bottou (2014) proposed the simple concatenation of visual and text representations, although more sophisticated methods such as the extension of skip-gram models into the multimodal domain (Lazaridou et al., 2015) or to apply stacked autoencoders to the unimodal representations (Silberer and Lapata, 2014) have also been considered.

2.3 Attribute Representations

Previous multimodal research often evaluates representations in word similarity tasks (Lazaridou et al., 2015; Silberer and Lapata, 2014; Kiela and Bottou, 2014) which offer rather a coarse-grain indicative of the quality of the embeddings and are not very informative about fine-grain aspects of meaning—or attributes. This gap is thus an important motivation for the present study. Attributes are often used by humans to describe objects (McRae et al., 2005), providing a powerful way to represent knowledge in terms of shape, color, taxonomic information, etc. Several studies have leveraged attributes to build representations that can be used in linguistic and visual tasks. For example, Silberer and Lapata (2014)

used an attribute-based hidden representation from stacked autoencoders as multimodal embeddings. The attributes were learned from both, visual and textual input. In contrast with Silberer and Lapata (2014), here we aim at spotting differences on the fine-grain semantic knowledge encoded by vision and language instead of building multimodal representations and to use them in a task. Furthermore, attribute representations exhibit the advantage that they can be learned transcending the task or the modality at hand, e.g., from either linguistic or visual input. Leveraging this transcendence, Lampert et al. (2009) showed that it is possible to classify objects from unseen classes (i.e., zero-shot learning) by training with attributes specified by humans—such as shape or color—instead of using images. Afterwards, new classes can be identified provided that one has their list of attributes at hand. Further, attributes transcend class boundaries. For instance, the attribute "stripped" can be learned from zebras, bees or tigers (Lampert et al., 2009). In this direction, Farhadi et al. (2009) proposed that the goal of image recognition should be describing rather than naming, that is, for instance, labeling "spotty dog" instead of just "dog" or replacing "unknown class" by "hairy and four-legged."

2.4 Attribute Prediction

The categorization of attributes proposed by McRae et al. (2005) has been widely used in NLP and CV studies (Baroni and Lenci, 2008; Silberer and Lapata, 2014; Rubinstein et al., 2015). Their attribute taxonomy includes individual attributes that belong to more general attribute types (e.g., *tactile* or *taxonomic*). For example, *has_legs* is a *form_and_surface* attribute, while *is_a_bird* or *is_a_fruit* are instances of *taxonomic* attributes.

Rubinstein et al. (2015) distinguished between two types of attributes: *taxonomic* and *attributive* properties. An *attributive* property is any of the remaining attribute types from McRae et al. (2005) categorization that are not taxonomic (Tab. 1). These include attribute types such as *tactile* (e.g., *is soft*), *form_and_surface* (e.g., *is made of metal*) or *encyclopedic* (e.g., *is_dangerous*). By trying to predict attributes using word embeddings as input, Rubinstein et al. (2015) concluded that DMs are significantly better at capturing *taxonomic* attributes rather than *attributive* properties—showing thus a limitation of the distributional hypothesis for certain attributes. Their findings align with previous research that showed that *taxonomic* attributes are generally more abundant in text compared to *attributive* properties (Baroni and Lenci, 2008). While Rubinstein et al. (2015) investigated whether DMs are equally good at capturing each type of attribute, our research questions are different. First, we want to answer whether there are differences between textual and visual representations in the type of attributes that they encode; and second, where these differences are. In other words, we present an inter-modality analysis while Rubinstein et al. (2015) performed only intra-modality comparisons.

In addition to the survey of Rubinstein et al. (2015), the closest work to ours is a study by Bruni et al. (2012) who showed that a very particular type of attribute, namely *color*, is better captured by visual representations than by DMs. Here, we go one step further and compare performance between visual and text embeddings for a larger number of visual attributes, as well as for other non-visual attributes such as *taxonomic*, *functional* or *encyclopedic*.

3 Approach and Experimental Settings

In this section we describe the procedure that we follow in order answer our research questions. An explanatory diagram is shown in Fig. 1.

3.1 Visual Representations

We use ImageNet (Russakovsky et al., 2015) as our source of visual information. ImageNet is currently the largest labeled image bank, with a coverage of 21,841 WordNet synsets (or meanings) (Fellbaum, 1998) and 14,197,122 images. Our choice of ImageNet is motivated by: (i) large word coverage; (ii) images are generally clean and with the relevant object at the foreground; and (iii) replicability of our experiments. Here, we only keep synsets with more than 50 images, and we set an upper bound of 500 images per synset for computational reasons. After this selection, 11,928 synsets are kept.

We extract a 4096-dimensional vector of features for each image using the output of the last layer

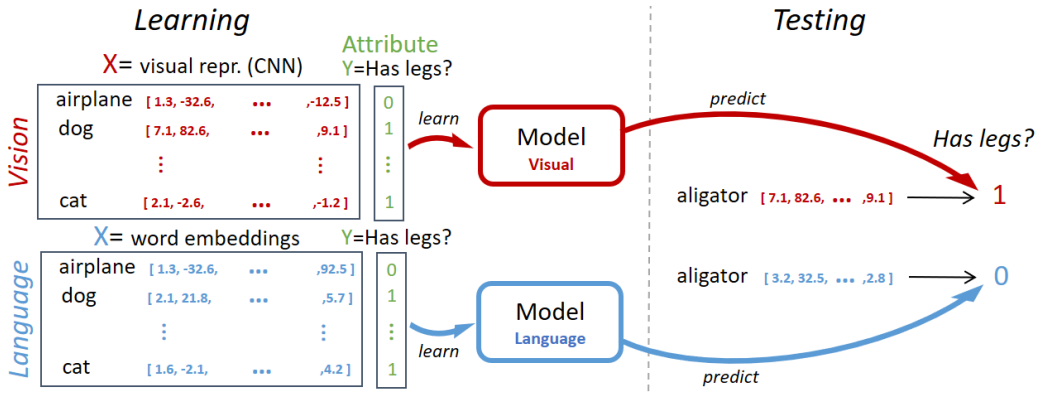


Figure 1: Overview of our experimental setting. Attributes are learned from the embeddings of each modality (left side), and afterwards new concepts are classified on whether the attribute is present or not (classification) or to which degree the attribute is present (regression). For clarity, we omitted the regression problem since its setting is identical to classification except for a continuous output $\mathcal{Y}$ instead of 0/1.

(before the softmax layer) of a pre-trained AlexNet CNN model implemented with Caffe toolkit (Jia et al., 2014). Other than CNN, there exist a variety of methods for obtaining visual features such as SIFT (Lowe, 1999), HOG (Dalal and Triggs, 2005) or SURF (Bay et al., 2006); to name a few. An exhaustive comparison will deviate from the goal of this paper, which is to show that at least some visual embeddings are able to better represent certain attributes than state-of-the-art DMs. Thus, we employ an off-the-shelf CNN model, as CNNs generally outperform the old approaches such as SIFT, HOG or SURF (LeCun et al., 2015). Additionally, we have repeated our experiments with ResNet (He et al., 2015), a more recent CNN model known to outperform AlexNet in image classification. Similar results are obtained with both models, suggesting thus that our vision-language comparisons are relatively independent of the CNN choice.

For each concept, several ways of integrating the representations from its individual images into a single vector could be devised. Here, we apply the following two common approaches (Kiela and Bottou, 2014):

(i) Averaging: Computes the component-wise average of the CNN feature vectors of individual images. This is equivalent to the cluster center of the individual representations.

(ii) Maxpool: Computes the component-wise maximum of the CNN feature vectors of individual images. This approach makes sense intuitively because CNN vector components can be interpreted as "visual properties."

For simplicity of notation we henceforth refer to the averaged and maxpooled visual representations as VIS_{avg} and VIS_{max} respectively.

3.2 Word Embeddings

We employ 300-dimensional GloVe vectors (Pennington et al., 2014) pre-trained in the largest available corpus (840B tokens and a 2.2M words vocabulary from Common Crawl corpus) from the author's website[1]. For completeness, we have repeated our experiments with word2vec embeddings (Mikolov et al., 2013) and we have found GloVe to perform slightly better. Thus, we report results with GloVe as it provides a stronger baseline to compare visual representations with.

3.3 McRae et al. Dataset

The data set collected by McRae et al. (2005) consists of data gathered from human participants that were asked to list properties—attributes—of concrete nouns. For each noun, 30 participants listed its attributes. For example, for "airplane", the attribute *has_wings* (i.e., a *form_and_surface* attribute) was

[1] http://nlp.stanford.edu/projects/glove

listed by 20 subjects, while the attribute *used_for_travel* (i.e., a *function* attribute) was listed by 7. The McRae et al. (2005) data contains 541 concepts, 2,526 different attributes, and 10 attribute types.

3.4 Binary Classification Setup

To evaluate fine-grain semantic understanding of the different embeddings, we evaluate how well the attributes from McRae et al. (2005) can be predicted by using the embeddings as input (Fig. 1). We use both a classification and a regression setting.

For each attribute a, we build a data set with the concepts to which this attribute applies as the positive class instances and the rest of concepts form the negative class. For example, a "beetle" is a negative instance and "airplane" a positive instance for the attribute a = *is_large*. We consider that an attribute applies to a noun if a minimum of 5 people have listed it[2]. Table 1 shows a summary of the number of positive class concepts per attribute type. For each attribute a we learn a predictor:

$$f_a : \mathcal{X} \rightarrow \mathcal{Y}$$

where $\mathcal{X} \subset \mathbb{R}^d$ is the input space of (d-dimensional) concept representations and $\mathcal{Y} = \{0, 1\}$ the binary output space.

To guarantee that the number of positive instances is enough to actually learn the attributes, only attributes with at least 25 positive instances are kept. This leads to a total of 43 attributes (Fig. 3), which can be seen as a total of 43 data sets. The concept selection in ImageNet described in Sect. 3.1 results in a visual coverage of 400 concepts (out of 541 from McRae et al. (2005) data), and, for a fair vision-language comparison, only the word embeddings (from GloVe) of these nouns are employed. Hence, our training data $\{(\overrightarrow{x_i}, y)\}_{i=1}^{400}$ consists of 400 instances. Since we have three types of representations (GloVe, VIS_{avg} and VIS_{max}), three different models f_a^{GloVe}, f_a^{avg} and f_a^{max} are learned from the three different input data $X_{GloVe} \in \mathbb{R}^{400\times300}$, $X_{avg} \in \mathbb{R}^{400\times4096}$ and $X_{max} \in \mathbb{R}^{400\times4096}$ respectively, where $X = \{\overrightarrow{x_i}\}_{i=1}^{400}$ and $\overrightarrow{x_i} \in \mathcal{X}$, $y \in \mathcal{Y}$.

Classification performance is evaluated with the F1 measure of the positive class—that is, the harmonic mean of precision and recall—since F1 is insensitive to class imbalance.

Attribute type	# Attr.	Avg. # concepts	SD
encyclopedic	4	32.7	1.5
function	3	46	27.9
sound	1	34	-
tactile	1	26	-
taste	1	33	-
taxonomic	7	42	24.8
color	7	42.4	12.0
form_and_surface	14	63.7	29.9
motion	4	37.5	5.7

Table 1: Attribute types, number of attributes per attribute type (# Attr.), and average number of concepts (i.e., positive instances) per attribute type (Avg. # concepts) with their respective standard deviations (SD).

3.5 Regression Setup

Let us consider the same scenario as in the classification one above, yet with a continuous output space $\mathcal{Y} = [0, 1]$ instead of a binary. The proportion of participants (out of 30) who have listed the attribute a for a given concept is taken as ground truth $y \in \mathcal{Y}$. This can be interpreted as the saliency of attribute a for this concept. Regression performance is evaluated with the Spearman ρ correlation coefficient between the predicted outputs and the ground truth.

[2] This threshold was set by McRae et al. (2005)

3.6 Experimental Setup

In both classification and regression we perform 2 runs of 5-fold stratified cross validation. That is, we create 5 (stratified) disjoint splits, repeating it with two different seeds. The use of 5 folds is convenient since the data set is small and contains just a few instances of the minority class—which ranges from 6.25% to 30.5% of the data. Thus, the use of $4/5$-th of the data for learning (and $1/5$-th for testing) is more likely to yield well-learned attributes than smaller training proportions. To handle class imbalance in classification we set the training class weights inversely proportional to the class priors.

This work relies on the basic assumption that, if a given attribute can be predicted using some embeddings as input (e.g., by means of a classifier or a regressor), then these embeddings contain encoded information about this attribute. The inverse is not necessarily true, that is, if an attribute cannot be predicted from a given embedding, this does not necessarily mean that the information is not present. In this case, the bottleneck might be any sort of technical issue such as our classifier choice, regularizer choice, data scaling, etc. Thus, in order to validate our conclusions, we repeated the same experiments with different classifier choices (SVM, logistic regression, bagging ensemble of decision trees and AdaBoost); different regressors (SVM, neural networks, ensemble of regression trees and gradient boosting); and data scalings (max/min scaling and component-wise centering plus normalizing by the standard deviation). We found results to be notably stable across classifier and regressor choices. We report results with a linear SVM regressor and a linear SVM classifier, both implemented with the scikit machine learning toolkit (Pedregosa et al., 2011). Data scaling affected only the performance of VIS_{max}, conceivably because their values are extreme by definition.

4 Results and Discussion

4.1 Intra-Modality Performance per Attribute Type

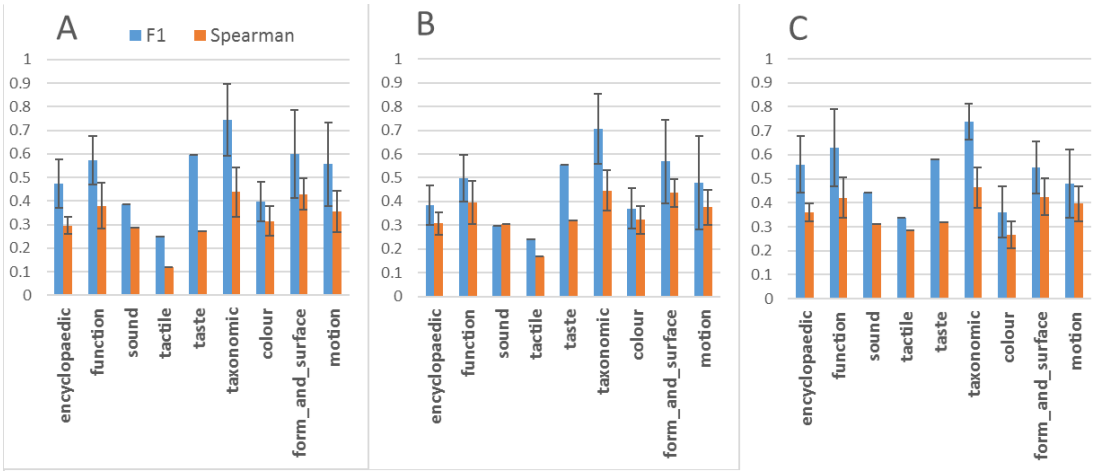


Figure 2: Averages of F1 (classification) and Spearman (regression) measures per attribute type (i.e., averaging individual attributes) for VIS_{avg} (A), VIS_{max} (B) and GloVe (C). Error bars show standard error.

As a first noteworthy finding, one may observe from Fig. 2 the—perhaps unexpected—resemblance that visual and textual representations present at predicting different types of attributes, which suggests interesting commonalities between visual and textual representations. Further, this seems to indicate that some attribute types are genuinely more difficult to predict than others (e.g., *tactile*), conceivably because the nouns to which these attributes apply tend to have little in common in terms of both, visual resemblance and word co-occurrences in similar contexts. This can be further appreciated in Fig. 5 (bottom row) from the scattered pattern of the attribute *is_soft*, which proves to be a difficult target for both, vision and language. In turn, this suggests that neither vision nor language might be sufficiently informative about certain attribute types (e.g., *tactile* or *sound*).

From Fig. 2 (C) it can also be observed that our results align with Rubinstein et al. (2015)'s findings with DMs. That is, from text-only embeddings, *taxonomic* attributes can be generally more accurately predicted than most of the *attributive* properties (i.e., all attribute types from Tab. 1 except *taxonomic*). We additionally find a similar behavior for visual embeddings (Fig. 2 A and B).

4.2 Visual Vs. Text Performance

Fig. 3 provides an answer to our first research question, showing that, clearly, neither vision nor language absolutely dominates the other in grasping fine-grain semantic knowledge but they rather show preference for different attributes. In general, visual embeddings (especially VIS_{avg}) perform better than GloVe in three main attribute types: *motion*, *form_and_surface* and *color* (Fig. 3 and 4). On the other hand, GloVe clearly outperforms vision in *encyclopedic* and *function* attribute types (Fig. 3 and 4), which are seldom visual. For the *taxonomic* type, vision or language clearly dominate in different individual attributes (Fig. 3). The visual performance gains with respect to GloVe (in e.g., *is_a_bird*) are particularly interesting since previous research evidenced that *taxonomic* is the attribute type where text-only DMs are the strongest (Baroni and Lenci, 2008; Rubinstein et al., 2015). Hence, these results suggest that the representation of taxonomic knowledge can further benefit from visual grounding.

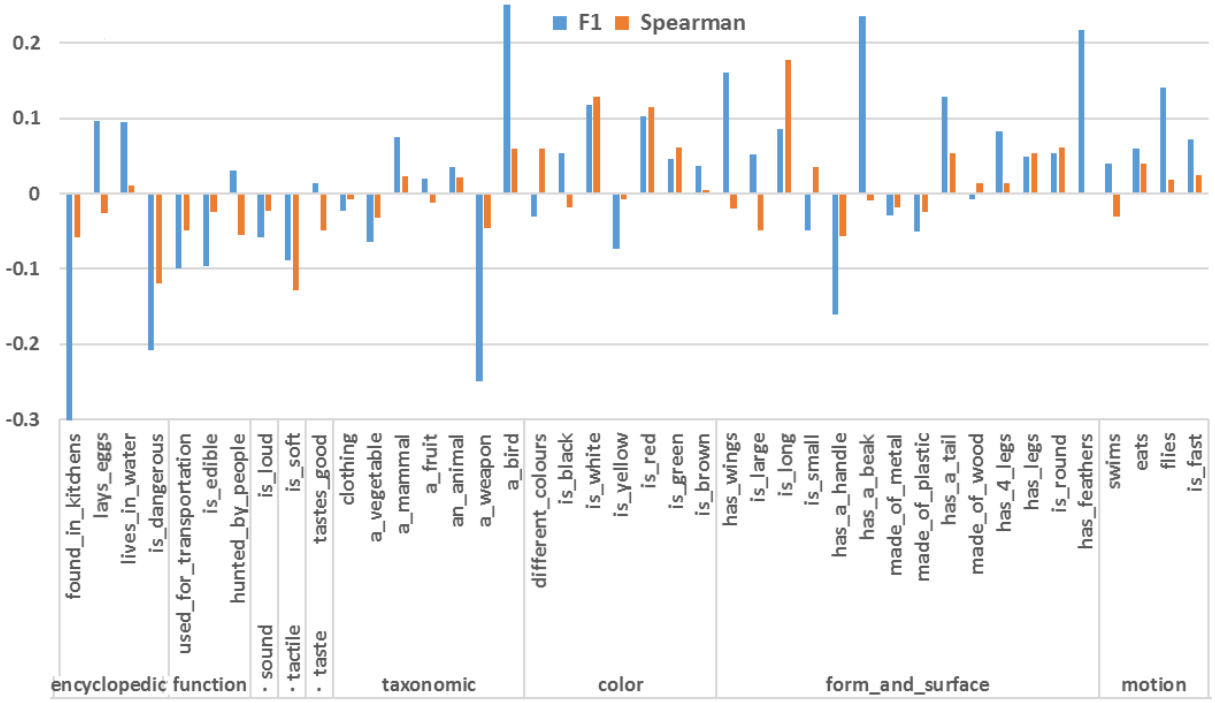


Figure 3: Difference of performance between VIS_{avg} minus GloVe. Attributes are shown on the horizontal axis and grouped by their type. Positive bars indicate better performance of visual embeddings and negative bars otherwise. Results with VIS_{max} are omitted as they exhibit almost identical patterns as VIS_{avg}, yet slightly worse.

Interestingly, even in the attribute types where either vision or language generally dominate, there are exceptions. For example, VIS_{avg} seems to outperform GloVe in classifying *lays_eggs* (i.e., an *encyclopedic* attribute), while GloVe seems to capture better *has_a_handle* (a *form_and_surface* attribute) which is predominantly visual. It is important to notice that visual attributes do not equal "less abstract" knowledge. For example, the visual attribute *has_a_handle* clearly requires more abstract semantic understanding than purely sensory visual attributes such as *is_green* since the definition of "handle" is clearly functionally-motivated rather than visual. For instance, the ball-shaped handle of a door has virtually no visual resemblance with the handle of a bag, yet they both have the same function. All this suggests that not only the attribute type is important but there are other factors to be taken into account. More concretely, the visual resemblance among objects to which the same attribute applies

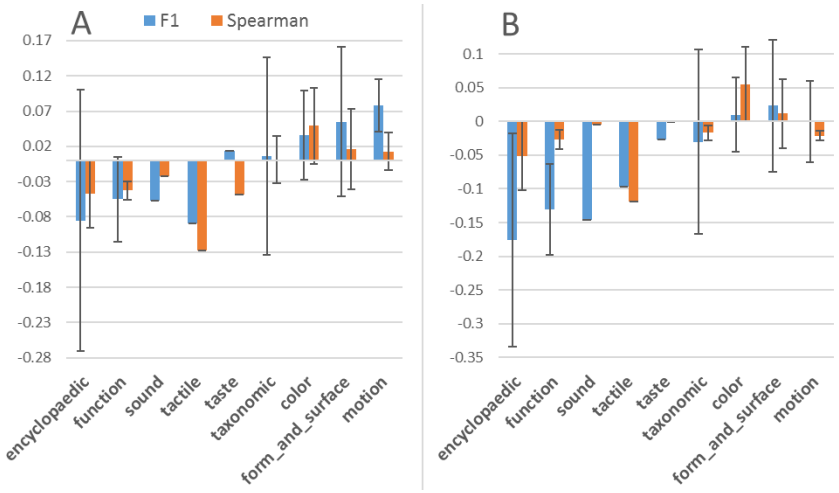


Figure 4: Averages of performance difference per attribute type. For each attribute type (e.g., taxonomic, taste, etc.), the bar indicates the average performance difference of its set of attributes. Plot A shows performance difference between VIS_{avg} and GloVe and B between VIS_{max} and GloVe. As in Fig. 3, positive bars indicate better performance of visual embeddings and negative bars otherwise. Error bars show standard error.

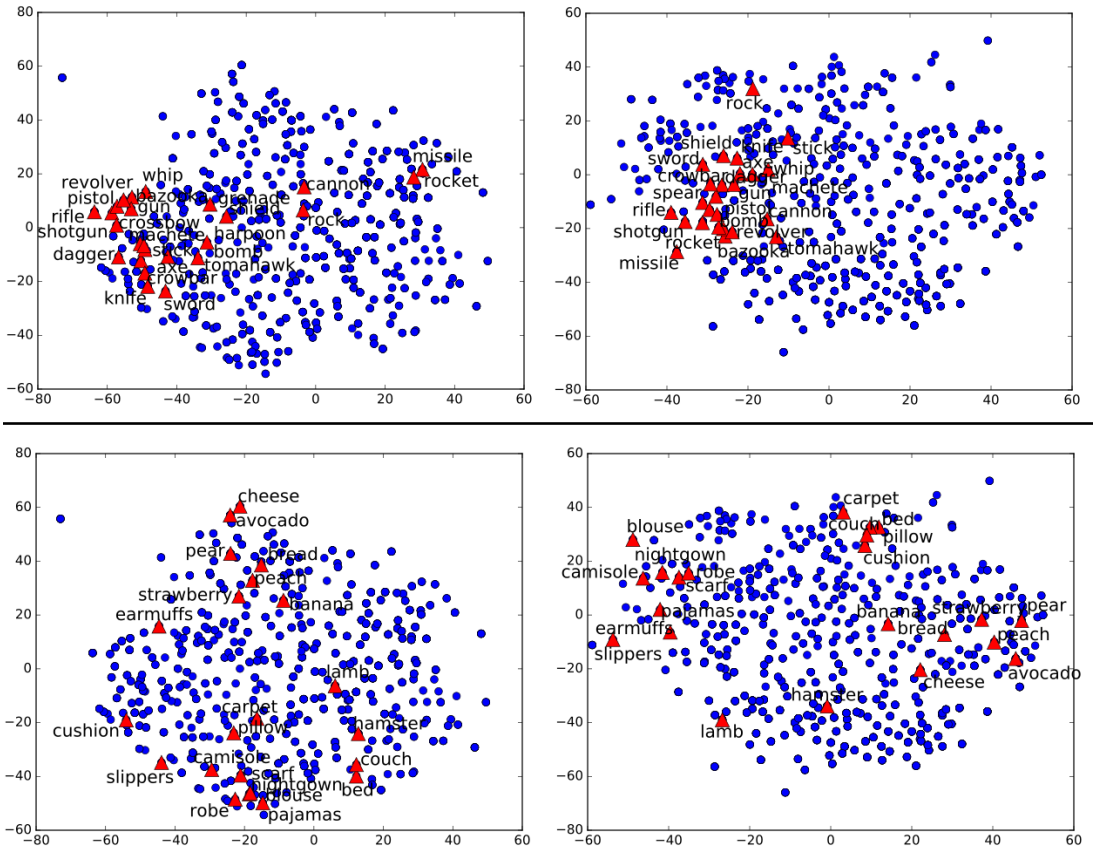


Figure 5: T-SNE visualization (Maaten and Hinton, 2008) of the VIS_{avg} embeddings (left column) and of the GloVe embeddings (right column). Red triangles show the positive class for the attribute *a_weapon* (top row) and *is_soft* (bottom row), while blue circles correspond to the negative class words.

seems to play a role. For example, *a_weapon* and *is_a_bird* are both *taxonomic* attributes although GloVe clearly dominates in the first and vision in the second one (Fig. 3). A closer inspection reveals that

the concepts from *is_a_bird* (e.g., "canary", "chicken", "eagle", "penguin") exhibit an important visual resemblance, whilst those from *a_weapon* (e.g., "rock", "bow", "rifle", "axe") do not. From Fig. 5 (top row) one can observe that the instances of *a_weapon* present a more scattered pattern in the visual embeddings (left) than in the linguistic ones (right). The same applies to *has_a_handle*, which one would perhaps—wrongly—expect that it might be better captured by vision. Contrarily, visual resemblance generally yields vectors that are closer in the visual space, making class boundaries easier to learn.

Even though the performances of classification and regression models (F1 and Spearman, respectively) are markedly correlated (Fig. 2, 3 and 4), there are a few exceptions. Small "contradictions" are plausible considering that classification and regression are two different problems—detection and estimation respectively—in which the learner is exposed to different (output) data. However, just a few bars show an opposite sign, none of them showing extreme opposite values.

5 Conclusions and Future Work

Overall, the present study adds evidence to the fact that visual and textual representations encode different semantic aspects of concepts. Crucially, we find that neither vision nor language are superior to the other in grasping every aspect of meaning, but they rather dominate in different attribute types. More concretely, vision proves generally better at capturing *form_and_surface*, *color* and *motion* attributes while language proves better at *encyclopedic* and *function* attributes. However, even within these general trends, we find that there are important exceptions in which visual resemblance among the concepts to which an attribute applies seems to play an important role. As an additional finding, we find that vision and language present a surprisingly similar pattern of predictive capacity for the different attributes types (Fig. 2), and that neither vision nor language succeed at capturing certain attribute types (e.g., *tactile* or *sound*). This suggests that other perceptual modalities can further improve the representations.

Taken together, we conclude that fine-grain attribute understanding can benefit from the integration of visual and textual representations. Thus, our results align with previous multimodal research that evaluates vision and language in coarse-grain tasks such as concept similarity and conclude that multimodal representations outperform the unimodal baselines (Lazaridou et al., 2015; Silberer and Lapata, 2014; Kiela and Bottou, 2014). Ultimately, our findings provide insights that can help building better multimodal representations by taking into account the types of semantic knowledge that vision and language selectively capture. In turn, better representations can improve automatic language understanding by providing a more human-like and perceptually grounded processing. Furthermore, we believe that the taxonomic knowledge encoded in visual representations can be further exploited towards building methods that automatically identify taxonomic relationships between concrete (perceptible) nouns, offering potential alternatives to WordNet (Fellbaum, 1998). Finally, recent work in computer vision showed that better fine-grain semantic understanding of objects can be achieved if images are segmented into the objects' individual parts (Vedaldi et al., 2014). We believe that this is an interesting direction to extend our work.

Acknowledgements

This work has been supported by the CHIST-ERA EU project MUSTER[3].

References

Marco Baroni and Alessandro Lenci. 2008. Concepts and properties in word spaces. *Italian Journal of Linguistics*, 20(1):55–88.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*, pages 238–247.

Lawrence W Barsalou. 2008. Grounded cognition. *Annu. Rev. Psychol.*, 59:617–645.

[3] http://www.chistera.eu/projects/muster

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. Surf: Speeded up robust features. In *ECCV*, pages 404–417. Springer.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *ACL*, pages 136–145. ACL.

Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, pages 886–893. IEEE.

Koen Deschacht and Marie-Francine Moens. 2009. Using the latent words language model for semi-supervised semantic role labeling. In *EMNLP*.

Koen Deschacht, Jan De Belder, and Marie-Francine Moens. 2012. The latent words language model. *Computer Speech & Language*, 26(5):384–409.

Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. 2009. Describing objects by their attributes. In *CVPR*, pages 1778–1785. IEEE.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *EMNLP*, pages 36–45.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105.

Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. 2009. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, pages 951–958. IEEE.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skip-gram model. *arXiv preprint arXiv:1501.02598*.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

David G Lowe. 1999. Object recognition from local scale-invariant features. In *ICCV*, volume 2, pages 1150–1157. IEEE.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *JMLR*, 9(Nov):2579–2605.

Ken McRae, George S Cree, Mark S Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior research methods*, 37(4):547–559.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Rasmus Rothe, Radu Timofte, and Luc Van Gool. 2015. Some like it hot-visual guidance for preference prediction. *arXiv preprint arXiv:1510.07867*.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Dana Rubinstein, Effi Levi, Roy Schwartz, and Ari Rappoport. 2015. How well do distributional models capture different types of semantic knowledge? In *ACL*, volume 2, pages 726–730.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *ACL*, pages 721–732.

Peter D Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *JAIR*, 37(1):141–188.

Andrea Vedaldi, Siddharth Mahendran, Stavros Tsogkas, Subhransu Maji, Ross Girshick, Juho Kannala, Esa Rahtu, Iasonas Kokkinos, Matthew B Blaschko, David Weiss, et al. 2014. Understanding objects in detail with fine-grained attributes. In *CVPR*, pages 3622–3629.

On the contribution of word embeddings to temporal relation classification

Paramita Mirza
Max Planck Institute for Informatics
Saarland Informatics Campus, Germany
paramita@mpi-inf.mpg.de

Sara Tonelli
Fondazione Bruno Kessler
Trento, Italy
satonelli@fbk.eu

Abstract

Temporal relation classification is a challenging task, especially when there are no explicit markers to characterise the relation between temporal entities. This occurs frequently in inter-sentential relations, whose entities are not connected via direct syntactic relations making classification even more difficult. In these cases, resorting to features that focus on the semantic content of the event words may be very beneficial for inferring implicit relations. Specifically, while morpho-syntactic and context features are considered sufficient for classifying event-timex pairs, we believe that exploiting distributional semantic information about event words can benefit supervised classification of other types of pairs. In this work, we assess the impact of using word embeddings as features for event words in classifying temporal relations of event-event pairs and event-DCT (document creation time) pairs.

1 Introduction

The classification of temporal relations between events in text has been long studied and attacked from different perspectives in the NLP community. However, existing approaches heavily rely on information overtly expressed in text, such as explicit temporal markers (e.g. *before*, *during*), the tense, aspect and modality of event words, as well as specific syntactic constructions. In case overt indicators are missing, the task becomes significantly more challenging, as is often the case when two events take place in different sentences, or in anchoring an event to the document creation time (DCT). See for example the sentences in (i), where the label for the event pair (e_1, e_2) is BEFORE, and the sentence in (ii), where e INCLUDES the DCT.

(i) *When Wong Kwan* ***spent*** $_{e_1}$ *seventy million dollars for this house, he thought it was a great deal. He* ***sold*** $_{e_2}$ *the property to five buyers and said he'd double his money.*

(ii) *The U.N. Security Council on Aug. 6 ordered a global* ***embargo*** $_{e}$ *on trade with Iraq as punishment for seizing Kuwait.*

Inter-sentential event relations are quite frequent, covering for example 32.76% of the event pairs in the TempEval-3 evaluation corpus (UzZaman et al., 2013). Around 42.37% of pairs of an event and a time expression in the same corpus are actually pairs of an event and the DCT. Moreover, the TimeBank corpus contains 718 temporal relations which are co-ordinated by temporal signals, i.e., only 11.2% of all temporal links (Derczynski and Gaizauskas, 2013). These make research on implicit temporal ordering very relevant.

One common approach, first proposed in Marcu and Echihabi (2002), incorporates word-based information in the form of word pair feature vectors. Conventionally, a word is converted into a symbolic ID, which is then transformed into a feature vector using a *one-hot* representation: the feature vector has the same length as the size of the vocabulary, and only one dimension is on. From a machine learning point of view, this type of sparse representation makes parameter estimation extremely difficult and prone to

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2818–2828, Osaka, Japan, December 11-17 2016.

over-fitting. It is also very challenging to achieve any interesting semantic generalization with this representation. Consider for instance, (*attack*, *injured*) that would be at equal distance from a synonymic pair (*raid*, *hurt*) and an antonymic pair (*died*, *shooting*).

Other approaches make use of semantic features extracted from external knowledge bases such as WordNet synsets (Fellbaum, 1998) and VerbOcean semantic relations between verbs (Chklovski and Pantel, 2004), capturing for instance that *marriage* happens-before *divorce*. Mirza and Tonelli (2014) exploit the list of event duration distribution from Gusev et al. (2011) for temporal relation classification, showing that it gives no benefit to classifier performance. The problem with such knowledge bases is that they have limited coverage, while approaches based on distributional semantics require no supervision and have a much better coverage.

The main goal of this work is to assess the contribution of dense vector representations of words and word pairs to temporal relation type classification, as detailed in Section 3. Specifically, we want to establish (i) which vector combination schemes are more suitable for classifying pairs of events, (ii) how well word embeddings can be used for this particular task compared to traditional features (Section 4.2), and finally, (iii) whether the combination of traditional features and word embeddings yields a better performance than using the two components in isolation. To the latter purpose, we compare vector concatenation and stacked learning (Section 4.3).

Experiments and evaluations are performed on the TimeBank-Dense corpus (Chambers et al., 2014), which was designed to address the sparsity issue in existing corpora with temporal annotation. We also compare our approach with CAEVO, a CAscading EVent Ordering system evaluated on the same corpus (Section 5).

2 Related Work

Many natural language processing applications such as information extraction (IE), question answering (QA), topic detection and tracking require understanding about temporally located events, i.e., to anchor events in time and order them. This temporal information is often modelled as a graph, with *times* and *events/states* as the nodes and *temporal relations* holding between them as the arcs. The details of how these three primitives are expressed in English, as well as their conceptual background (Allen, 1984; Moens and Steedman, 1987) have been discussed in Setzer (2001), and formalized in the TimeML annotation standard (Pustejovsky et al., 2003). In this work we focus on the task of ordering temporal entities, i.e., the classification of temporal relation types.

Current state-of the-art systems for temporal ordering resort to data-driven approaches (Bethard, 2013; Laokulrat et al., 2013; Mirza and Tonelli, 2014) or hybrid approaches combining rules and supervised classifiers (D'Souza and Ng, 2013; Chambers et al., 2014; Mirza and Tonelli, 2016). In building the classification models, most approaches rely primarily on morpho-syntactic features as well as lexical semantic information derived from WordNet synsets (Chambers et al., 2007; Laokulrat et al., 2013; Chambers et al., 2014) and VerbOcean semantic relations between verbs (Mani et al., 2006; D'Souza and Ng, 2013).

Other approaches exploit sentence-level semantic information, i.e. predicate-argument structure, as features for the classifiers (Llorens et al., 2010; Laokulrat et al., 2013; D'Souza and Ng, 2013). However, the evaluation results of TempEval-3 (UzZaman et al., 2013) show that a system with basic morpho-syntactic and lexical semantic features, such as ClearTK (Bethard, 2013), is hard to beat even if using more sophisticated semantic features. Indeed, ClearTK indirectly uses distributed lexical semantic features in the form of context (tokens appearing) between events.

As far as we know, there is no work on the task of ordering/anchoring temporal entities which specifically addresses the issue of implicit relations often recurring when two events are in different sentences, or when an event is related to the DCT. Such implicit relations are probably covered by hand-crafted rules or features based on the tense, aspect and modality of event words (Chambers et al., 2014), but sometimes such an overt indicator is lacking, as exemplified in previous examples (Section 1).

Most works on implicit discourse relations focused on the Penn Discourse Treebank (PDTB) (Prasad et al., 2008), in which relations are annotated at the discourse level and organized into a three-level hier-

archy. The top level relations, for example, include *Temporal*, *Contingency*, *Comparison* and *Expansion*. Braud and Denis (2015) presented a detailed comparative studies for assessing the benefit of unsupervised word representations, i.e. one-hot word pair representations against low-dimensional ones based on Brown cluster (Brown et al., 1992) and word embeddings, for identifying implicit discourse relations in PDTB. However, only the top level relations are considered, for instance, whether there exists a *Temporal* relation without investigating further into the more fine-grained temporal ordering.

3 Classifying Temporal Relations

Temporal relations, or temporal links, are annotations that connect markables bearing temporal information in a text, and express their temporal order. TimeML (Pustejovsky et al., 2003) is the widely known annotation framework for creating such representation that was used in the TempEval series, i.e., evaluation exercises focusing on temporal information processing. One of the main tasks in TempEval is temporal relation (TLINK) classification: given a pair of temporal entities (te_1 , te_2), namely events and time expressions (timex), determine their ordering relations (e.g. BEFORE, AFTER, INCLUDES, etc.).

Our goal is to compare classification performance on temporal relations using traditional features and word embeddings, which have recently shown to achieve good generalisation capabilities in several NLP tasks. Specifically, we want to evaluate whether, and in which configurations, they can contribute to advance state-of-the-art performance on temporal relation classification. To this purpose, we build and evaluate two different classifiers.

3.1 Temporal Relation Classification with Traditional Features

The first system is inspired by state-of-the-art approaches presented at TempEval-3 (UzZaman et al., 2013). Following UTTime (Laokulrat et al., 2013), we build three LIBLINEAR (Fan et al., 2008) classifiers (L2-regularized logistic regression): one for event-document creation time (E-D), one for event-timex (E-T) and one for event-event (E-E) edges. For timex-timex (T-T) relations, we implement a simple set of rules based on the values of time expressions, which proved to be effective for most T-T edges.

E-D, E-T and E-E Classifiers A set of features, listed in Table 1, is used for each type of edge, largely inspired by the best performing systems in TempEval-2 (Verhagen et al., 2010) and TempEval-3 (UzZaman et al., 2013) campaigns. We assume that pairs of temporal entities are given, and we rely on EVENT and TIMEX3 *attributes* in annotated TimeML documents, *morphosyntactic information* generated by MorphoPro (Pianta et al., 2008) and *dependency information* from the Mate tools (Bjorkelund et al., 2010).

Derczynski and Gaizauskas (2013) show the importance of *temporal signals* in temporal relation labelling, hence, we include also a similar set of features. However, we take the list of temporal signals from the TimeBank corpus, further expand it using the Paraphrase Database (Ganitkevitch et al., 2013), and manually cluster synonymous signals together, e.g. {*before*, *prior to*, *in advance of*}. The cluster ID is then included in the feature set instead of the signal text.

Note that the only *lexical semantic information* we include in the feature set is the Wordnet semantic similarity/relatedness (Lin, 1998) between event words. In order to have a feature vector of reasonable size, we simplify the possible values of some features during the one-hot encoding:

- *dependencyPath*. We only consider the existence of a dependency path between an E-E pair when it describes coordination, subordination, subject or object relation. For E-T pairs, we only consider the dependency path expressing temporal modification.
- *tempSignalCluster*. Given a temporal signal, we include the *clusterID* of the cluster containing synonymous signals, e.g. {*before*, *prior to*, *in advance of*} instead of the signal text.
- *wnSim*. The value of WordNet similarity measure is discretized as follows: $sim \leq 0.0$, $0.0 < sim \leq 0.5$, $0.5 < sim \leq 1.0$ and $sim > 1.0$.

T-T Rules Only temporal expressions of types DATE and TIME are considered in the hand-crafted set of rules, based on their *normalized values*. For example, *7 PM tonight* with `2015-12-12T19:00` as value IS_INCLUDED in *today* with `2015-12-12` as value.

Feature	TLINK E-D	TLINK E-T	TLINK E-E	Rep.	Description
EVENT attributes					
class	x	x	x	one-hot	EVENT attributes as specified in TimeML.
tense	x	x	x	one-hot	
aspect	x	x	x	one-hot	
polarity	x	x	x	one-hot	
sameClass			x	binary	Whether e_1 and e_2 have the same EVENT attributes.
sameTenseAspect			x	binary	
samePolarity			x	binary	
TIMEX3 attributes					
type	x	x		one-hot	TIMEX3 attributes as specified in TimeML.
Morphosyntactic information					
PoS	x	x	x	one-hot	Part-of-speech tags of e_1 and e_2.
phraseChunk	x	x	x	one-hot	Shallow phrase chunk of e_1 and e_2.
samePoS		x	x	binary	Whether e_1 and e_2 have the same PoS.
Textual context					
entityOrder		x		binary	Appearance order of e_1 and e_2 in the text.[1]
sentenceDistance		x	x	binary	0 if e_1 and e_2 are in the same sentence, 1 otherwise.
entityDistance		x	x	binary	0 if e_1 and e_2 are adjacent, 1 otherwise.
Dependency information					
dependencyPath			x	one-hot	Dependency path between e_1 and e_2.
isMainVerb	x	x	x	binary	Whether e_1/e_2 is the main verb of the sentence.
hasModalVerb	x	x	x	binary	Whether e_1/e_2 is governed by a modal verb.
Temporal signals					
tempSignalCluster		x	x	one-hot	Cluster ID of temporal signal existing around e_1 and e_2.
tempSignalPosition		x	x	one-hot	Temporal signal position w.r.t e_1/e_2 (BETWEEN, BEFORE, BEGIN, etc.)
tempSignalDependency		x	x	one-hot	Temporal signal dependency path between signal tokens and e_1/e_2.
lexical semantic information					
wnSim			x	one-hot	WordNet similarity computed between the lemmas of e_1 and e_2.

Table 1: Feature set for TLINK classification model for event-document creation time (E-D), event-timex (E-T) and event-event (E-E) pairs, along with representation type (Rep.) and brief description.

3.2 Temporal Relation Classification with Word Embeddings

Recently there has been an increasing interest in using word embeddings as an alternative source of information to traditional hand-crafted features. Word embeddings represent (embed) the semantics of a word in a continuous vector space, where semantically similar words are mapped to nearby points. The underlying principle is the *Distributional Hypothesis* (Harris, 1954), which states that words which are similar in meaning occur in similar contexts.

Baroni et al. (2014) divide approaches based on this principle into two categories: (i) *count-based* models and (ii) *predictive* models. They also provide a systematic comparison of word vectors from the two models, on a wide range of lexical semantic tasks, including semantic relatedness, synonym detection, concept categorization, selectional preferences and analogy. The main takeaway is that predictive models, such as Word2Vec (Mikolov et al., 2013), are shown to perform better than count-based ones.

Levy et al. (2015) reveal that much of the performance gains of word embeddings are due to hyperparameter optimizations rather than the embedding algorithms themselves, thus refuting the claim that prediction-based methods are superior to count-based approaches. However, they also state that the Skip-Gram model with Negative Sampling (SGNS), which is used to build Word2Vec pre-trained word vectors, can be a robust baseline since it does not significantly underperform in any scenario.

In this work, we explore how well word embeddings can be used—as lexical semantic features—to capture the temporal order of events (e.g. *attack* often happens BEFORE *injured*) and the temporal anchoring of an event to the document creation time (e.g. *embargo* usually spans longer than a day, hence, INCLUDES the DCT). Again, we build LIBLINEAR (Fan et al., 2008) classifiers (L2-regularized logistic regression), one for E-D and another for E-E pairs. Instead of the traditional feature sets explained in Section 3.1, word embeddings are used as feature vectors.

[1]The order of e_1 and e_2 in E-E pairs is always according to the appearance order in the text, while in E-T pairs, e_2 is always a timex regardless of the appearance order.

Pre-trained word vectors We take pre-trained word vectors from Word2Vec[2], which are 300-dimensional vectors for 3 million words and phrases trained on part of Google News dataset (about 100 billion words). Given an E-E pair (e_1, e_2), we retrieve the pair of word vectors $(\vec{w}_1, \vec{w}_2)$ based on vector look-up for the head words of e_1 and e_2 in the pre-trained word vectors. Meanwhile, for an E-D pair (e, t) we retrieve word vectors $\vec{w}$ according to the head word of e.

Vector combinations For E-E pairs, we test three different strategies in combining the word vectors of a pair of events: we consider (i) *concatenation* $(\vec{w}_1 \oplus \vec{w}_2)$, (ii) *addition* $(\vec{w}_1 + \vec{w}_2)$ and (iii) *subtraction* $(\vec{w}_2 - \vec{w}_1)$, as vector combination schemes. Note that in (i) the word ordering information is retained, which is not the case in (ii) and (iii).

We only consider word embeddings for events, specifically their head words, because the embeddings for all events annotated in the dataset, which are mostly verbs and nouns, are readily obtainable from the pre-trained word vectors. Meanwhile, representing time expressions with single word vectors is non-trivial, since most of them express dates (e.g. *Friday the 13th*) and times (e.g. *half past ten*), which are usually multi-word expressions.

4 Experimental Setup and Evaluation

We present two sets of experiments: first, we investigate how well word embeddings can be used for temporal relation classification compared with traditional features, and then we analyse whether the combination of these two types of features is beneficial.

4.1 Dataset: TimeBank-Dense

We evaluate the temporal classifiers using the TimeBank-Dense corpus (Chambers et al., 2014), which was created to address the sparsity issue in existing TimeML corpora. Using a specialized annotation tool, annotators were prompted to label all pairs of events and time expressions in the same sentence, all pairs of events and time expressions in two adjacent sentences, and all pairs of events and document creation time. This solution was introduced to solve the problem of sparse annotation of temporal relations in the TempEval-3 evaluation corpus, which made it difficult to evaluate and compare different systems.

The VAGUE relation introduced at the first TempEval task (Verhagen et al., 2007) was also adopted in TimeBank-Dense to cope with ambiguous temporal relations, or to indicate pairs for which no clear temporal relation exists. The resulting corpus contains 12,715 temporal relations under 6 labels, i.e. BEFORE, AFTER, INCLUDES, IS_INCLUDED, SIMULTANEOUS and VAGUE, over 36 documents taken from TimeBank. Annotation here is much denser than in the TimeBank corpus, which contains 6,418 temporal relations under 14 labels over 183 documents.

We follow the experimental setup in Chambers et al. (2014), in which the TimeBank-Dense corpus is split into a 22 document training set, a 5 document development set and a 9 document test set.[3] All the classification models are trained using the training set, as well as the rule set development for T-T edges. We evaluate our classification performances on (i) stratified 10-fold cross validation over the training set and (ii) on the test set.

4.2 Experiment 1: Comparing Traditional Features vs. Word Embeddings

In Table 2 we report the performances (micro-averaged F1-scores) of each classifier using word vectors $\vec{w}$ as features, compared with the classifier performance using traditional features $\vec{f}$, evaluated on stratified 10-fold cross-validation. For E-E pairs, we also report the F1-scores for each vector combination scheme. Since we classify all possible event pairs in the dataset, precision and recall are the same.

From the different vector combinations, concatenation $(\vec{w}_1 \oplus \vec{w}_2)$ is shown to be the best combination. Using the concatenated Word2Vec embeddings $(\vec{w}_1 \oplus \vec{w}_2)$ as features results in .605 F1-score, significantly better than using only traditional features (.529 F1-score). The fact that this representation retain the word order information may be the reason why it beats the other vector combinations.

[2] http://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/

[3] Available at http://www.usna.edu/Users/cs/nchamber/caevo/.

TLINK type	E-D		E-E			
	$\vec{f}$	$\vec{w}$	$\vec{f}$	$(\vec{w}_1 \oplus \vec{w}_2)$	$(\vec{w}_1 + \vec{w}_2)$	$(\vec{w}_2 - \vec{w}_1)$
BEFORE	**.551**	.547	.338	**.521**	.281	.508
AFTER	.171	**.326**	.330	**.544**	.325	.504
SIMULTANEOUS	-	-	**.094**	-	.032	.080
INCLUDES	.245	**.428**	.140	**.363**	.065	.326
IS_INCLUDED	.478	**.503**	.144	**.385**	.145	.364
VAGUE	.445	**.463**	.664	**.693**	.676	.426
Overall	.449	**.476**	.529	**.605****	.516	.443

Table 2: Micro-averaged F1-scores per TLINK type with different feature vectors, evaluated on stratified 10-fold cross-validation over the training set. ** denotes $p < .01$.

TLINK type	E-D		E-E			
	$\vec{f}$	$\vec{w}$	$\vec{f}$	$(\vec{w}_1 \oplus \vec{w}_2)$	$(\vec{w}_1 + \vec{w}_2)$	$(\vec{w}_2 - \vec{w}_1)$
BEFORE	.587	**.627**	.388	**.443**	.264	.408
AFTER	.265	**.444**	.267	.412	.246	**.467**
SIMULTANEOUS	-	-	-	-	-	-
INCLUDES	.067	**.217**	.066	.068	-	**.156**
IS_INCLUDED	**.559**	.524	.111	**.435**	.154	.155
VAGUE	**.474**	.424	**.612**	.592	.611	.313
Overall	.476	**.479**	.493	**.496**	.444	.338

Table 3: F1-scores per TLINK type with different feature vectors, evaluated on the test set.

With the exception of SIMULTANEOUS and VAGUE, all of the other TLINK types are asymmetric, e.g. BEFORE/AFTER, INCLUDES/IS_INCLUDED.

Note that the classifier with subtracted word vectors $(\vec{w}_2 - \vec{w}_1)$ as features is able to capture event pairs labelled as SIMULTANEOUS, which are failed to be detected by the classifier with concatenated word embeddings. In some cases, the SIMULTANEOUS event pairs are co-referring events that have similar meanings, e.g. (*attack*, *strike*). By subtracting the word vectors of such event headwords, we could get a feature vector close to the origin $(0, 0, ..., 0)$, which can be used by the classifier to capture the SIMULTANEOUS relation. This kind of information is not available if we use concatenated word vectors as features. However, the traditional feature set containing Wordnet semantic similarity/relatedness information still yields a better performance than subtracted word embeddings.

For E-D pairs, using word vectors $\vec{w}$ as features (.476 F1-score) is better than using traditional features $\vec{f}$ (.449 F1-score), in particular for INCLUDES and AFTER labels. Given that in a news text, the document creation time is usually of TIME or DATE types, this means that word embeddings are able to predict when events last longer than a day, hence the appropriate label, i.e. INCLUDES, is chosen.

The same phenomena are also observed in the evaluation on test data, shown in Table 3: (i) concatenation $(\vec{w}_1 \oplus \vec{w}_2)$ is the best vector combination scheme for E-E edges and (ii) using word embeddings increases the performance on INCLUDES and AFTER labelling of E-D pairs. Pairs labelled as SIMULTANEOUS are highly under-represented in the dataset for all types of edges, composing only around 1.1% of the training data and 1.3% of the test data. This explains the failure of the classification models in capturing this particular relation in the test data.

Overall, using word vectors is better than using carefully crafted traditional features, particularly on the 10-fold cross-validation evaluation over the training data. However, on the evaluation over the test set, the improvements on the overall F1-scores are not significant ($p > .05$), even though we observe statistically significant improvements on specific TLINK types (e.g., INCLUDES for E-D pairs and IS_INCLUDED for E-E pairs).

TLINK type	E-D			E-E		
		S1	S2		S1	S2
	$\vec{w}$	$(\vec{w} \oplus \vec{f})$	$(\vec{p} \oplus \vec{f})$	$(\vec{w_1} \oplus \vec{w_2})$	$((\vec{w_1} \oplus \vec{w_2}) \oplus \vec{f})$	$(\vec{p} \oplus \vec{f})$
BEFORE	.627	.667	**.671**	.443	**.501**	.471
AFTER	.444	**.518**	.466	.412	**.435**	.430
SIMULTANEOUS	-	-	-	-	**.154**	-
INCLUDES	.217	**.333**	.250	.068	**.085**	.049
IS_INCLUDED	.524	.554	**.600**	**.435**	.288	.250
VAGUE	.424	.449	**.502**	.592	.596	**.613**
Overall	.479	.524*	**.534***	.496	.512*	**.519****

Table 4: F1-scores on both experimental settings S1 (concatenating word vectors and traditional features) and S2 (stacking), evaluated on the test set. * denotes significance at $p < .05$, ** denotes $p < .01$.

4.3 Experiment 2: Combining Traditional Features and Word Embeddings

To assess whether traditional features are still relevant in the presence of word vectors, two experimental settings are considered:

S1 We simply concatenate word vectors and traditional feature vectors together as the feature sets for the classifiers: $(\vec{w} \oplus \vec{f})$ is taken as the feature set for E-D and $((\vec{w_1} \oplus \vec{w_2}) \oplus \vec{f})$ for E-E pairs.

S2 We employ an ensemble learning technique, namely *stacking*, to combine both sets of features. First, the classifiers with word vectors as features, i.e., $\vec{w}$ for E-D and $(\vec{w_1} \oplus \vec{w_2})$ for E-E, are trained on the training data in a 10-fold cross-validation scheme, producing prediction vectors $\vec{p}$, i.e., one-hot representation of predicted labels, for the whole training data. Then, a combined classifier is trained to make a final prediction using $(\vec{p} \oplus \vec{f})$ as the feature set.

Again, LIBLINEAR (L2-regularized logistic regression) is used to build all the classifiers, which are then evaluated on the test data. Table 4 gives an overview of system performances on both settings, along with F1-scores when using the best feature vectors according to previous experiments, i.e., $\vec{w}$ for E-D and $(\vec{w_1} \oplus \vec{w_2})$ for E-E pairs.

For both E-D and E-E pairs, a super classifier trained using $\vec{p} \oplus \vec{f}$ as the feature vector performs better than a classifier trained with concatenated word vectors and traditional features. Furthermore, the super classifiers significantly outperform classifiers trained with the best single feature sets (word embeddings for E-D and E-E pairs), i.e., .534 vs. .479 F1-scores for E-D ($p < .05$) and .519 vs. .496 F1-scores for E-E pairs ($p < .01$).

5 Discussion

Our final system is composed of a rule set for T-T pairs, and three LIBLINEAR (L2-regularized logistic regression) classifiers for E-D, E-T and E-E pairs. Following the results from our experiments detailed in Section 4, we consider the best feature set for E-D and E-E pairs, i.e., the combination of word embeddings and traditional features via stacked learning $(\vec{p} \oplus \vec{f})$. Meanwhile, for E-T edges, we use the traditional feature vector $\vec{f}$ as the feature set.

Comparison of system performances We compare our system performance with two baseline systems. The first baseline labels all edges as VAGUE, which is the baseline system reported in Chambers et al. (2014). The second baseline system chooses the majority labels (non-VAGUE) for each type of edges, i.e., BEFORE for T-T, E-D and E-E, and AFTER for E-T pairs. As reported in Table 5, our system outperforms both baselines.

We also report in Table 5 our system performance in comparison with CAEVO (Chambers et al., 2014), the only existing temporal ordering system evaluated on the TimeBank-Dense corpus. Note that CAEVO is a hybrid system combining several rule-based and machine-learned classifiers in a sieve-based architecture, which includes transitive reasoning after each classifier labels the entity pairs. Our

System	T-T	E-D	E-T	E-E	Overall
Baseline: All VAGUE	.203	.277	.388	.447	.409
Baseline: Majority (non-VAGUE)	.508	.241	.305	.269	.278
Our system	**.780**	.534	.468	**.519**	**.518**
CAEVO	.712	**.553**	**.494**	.494	.507

Table 5: The comparison of system performances (F1-scores) for each edge type and the overall entity pairs.

	Our system			CAEVO		
Relation	**P**	**R**	**F1**	**P**	**R**	**F1**
BEFORE	.58	.46	**.51**	.52	.45	.49
AFTER	.59	.35	.44	.55	.38	**.45**
SIMULTANEOUS	.92	.28	**.43**	.71	.31	**.43**
INCLUDES	.15	.09	.11	.44	.21	**.28**
IS_INCLUDED	.51	.44	.47	.57	.43	**.49**
VAGUE	.49	.71	**.58**	.48	.66	.56

Table 6: The comparison of system performances on individual relation types.

system, composed of only one rule-set and three supervised-classifiers, is marginally better than CAEVO, particularly for T-T and E-E pairs, with the overall F1-scores of .518 vs. .507 (CAEVO).

CAEVO includes hand-crafted rules for E-D and E-T pairs, for instance rules to label edges between reporting events and DCTs (as IS_INCLUDED) and rules for edges between one verbal event and one timex based on temporal prepositions connecting the two (e.g. prepositions *for*, *at* and *throughout* signal a SIMULTANEOUS relation). It is very likely that our system based on general-purpose features cannot beat these very specific and carefully designed rules.

Finally, we present our system performance (in terms of precision, recall and F1-score) per-relation in Table 6, which is again compared to CAEVO. Overall, the two systems have comparable performances for all relation types, with the exception of the INCLUDES relation, in which CAEVO clearly outperforms our system.

Intra- vs. inter-sentential entity pairs Since we argue that embedding-based features may be particularly beneficial when no overt clues to express the temporal relation are present, as is often the case in inter-sentential relations, we compare the performance of the different feature vectors on inter- and intra-sentential relations in the test set. Results are reported in Table 7.

Feature vector	E-D		E-E	
	same	**diff**	**same**	**diff**
$\vec{f}$	-	.476	.466	.507
$\vec{w}$ or $(\vec{w_1} \oplus \vec{w_2})$	-	.479	.488	.501
S1: $(\vec{w} \oplus \vec{f})$ or $((\vec{w_1} \oplus \vec{w_1}) \oplus \vec{f})$	-	.524	**.516**	.509
S2: $(\vec{p} \oplus \vec{f})$	-	**.534**	.492	**.533**

Table 7: F1-scores for different feature vectors, evaluated on pairs in the test set belonging to the same sentence (same) and different sentences (diff).

Combining word embeddings and traditional features in *stacking* setting (S2) is shown to be beneficial for entity pairs occurring in different sentences. Interestingly, the combination of word embeddings and traditional features in the concatenated setting (S1) is quite beneficial to the classification of E-E pairs in the same sentence.

6 Conclusions

We have analysed the contribution of word embeddings to temporal relation type classification, specifically for E-D and E-E edges. The evaluation results shed some light on how word embeddings can potentially improve a classifier performance for this particular task, i.e., in combination with traditional features in the stacked learning scheme. These results confirm that word embeddings can become effective features when there are no overt markers of temporal relations.

Compared with the state-of-the-art system evaluated on the same corpus, CAEVO, our system achieves quite similar performances, even though it is based on a much simpler architecture. We believe that, integrating our rule set (for T-T pairs) and classifiers (for E-D, E-T and E-E pairs) in CAEVO's sieve-based architecture completed with transitive reasoning, may result in an improvement of the state-of-the-art on the task, which we plan to evaluate soon.

Several works have recently presented methods for building task-specific word embeddings (Hashimoto et al., 2015; Boros et al., 2014; Nguyen and Grishman, 2014; Tang et al., 2014). We believe that this may be beneficial also for temporal ordering, and we plan to build this kind of embeddings in the future, instead of using general-purpose vectors.

Acknowledgments

The research leading to this paper was partially supported by the European Union's 7th Framework Programme via the NewsReader Project (ICT-316404) and the National University of Singapore. We thank Ilija Ilievski, Min-Yen Kan and Hwee Tou Ng, who provided insight and expertise that greatly assisted the research.

References

James F. Allen. 1984. Towards a general theory of action and time. *Artif. Intell.*, 23(2):123–154, July.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland, June. Association for Computational Linguistics.

Steven Bethard. 2013. ClearTK-TimeML: A minimalist approach to TempEval 2013. In *Proceedings of the Seventh International Workshop on Semantic Evaluation*, SemEval '13, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Anders Bjorkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstrations*, pages 33–36, Beijing, China, August. Coling 2010 Organizing Committee.

Emanuela Boros, Romaric Besançon, Olivier Ferret, and Brigitte Grau. 2014. Event role extraction using domain-relevant word representations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1852–1857, Doha, Qatar, October. Association for Computational Linguistics.

Chloé Braud and Pascal Denis. 2015. Comparing word representations for implicit discourse relation classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2201–2211, Lisbon, Portugal, September. Association for Computational Linguistics.

Peter F. Brown, Peter V. de Souza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, December.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 173–176, Prague, Czech Republic, June. Association for Computational Linguistics.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 33–40, Barcelona, Spain, July. Association for Computational Linguistics.

Leon Derczynski and Robert Gaizauskas. 2013. Temporal signals help label temporal relations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 645–650, Sofia, Bulgaria, August. Association for Computational Linguistics.

Jennifer D'Souza and Vincent Ng. 2013. Classifying temporal relations with rich linguistic knowledge. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 918–927, Atlanta, Georgia, June. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of NAACL-HLT 2013*, pages 758–764, Atlanta, Georgia, June. ACL.

Andrey Gusev, Nathanael Chambers, Pranav Khaitan, Divye Khilnani, Steven Bethard, and Dan Jurafsky. 2011. Using query patterns to learn the duration of events. In *Proceedings of the Ninth International Conference on Computational Semantics*, IWCS '11, pages 145–154, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zellig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2015. Task-oriented learning of word embeddings for semantic relation classification. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 268–278, Beijing, China, July. Association for Computational Linguistics.

Natsuda Laokulrat, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Uttime: Temporal relation classification using deep syntactic features. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 88–92, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, ICML '98, pages 296–304, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Hector Llorens, Estela Saquete, and Borja Navarro. 2010. TIPSem (English and Spanish): Evaluating CRFs and Semantic Roles in TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 284–291, Uppsala, Sweden, July. Association for Computational Linguistics.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 753–760, Sydney, Australia, July. Association for Computational Linguistics.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 368–375, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Paramita Mirza and Sara Tonelli. 2014. Classifying temporal relations with simple features. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 308–317, Gothenburg, Sweden, April. Association for Computational Linguistics.

Paramita Mirza and Sara Tonelli. 2016. CATENA: Causal and Temporal relation Extraction from Natural language texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*, Osaka, Japan, December. Association for Computational Linguistics.

Marc Moens and Mark Steedman. 1987. Temporal ontology in natural language. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 1–7, Stanford, California, USA, July. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2014. Employing word representations and regularization for domain adaptation of relation extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 68–74, Baltimore, Maryland, June. Association for Computational Linguistics.

Emanuele Pianta, Christian Girardi, and Roberto Zanoli. 2008. The TextPro Tool Suite. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May. European Language Resources Association (ELRA).

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May. European Language Resources Association (ELRA).

James Pustejovsky, José Castaño, Robert Ingria, Roser Saurí, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003. TimeML: Robust specification of event and temporal expressions in text. In *Proceedings of the Fifth International Workshop on Computational Semantics (IWCS-5)*.

Andrea Setzer. 2001. *Temporal Information in Newswire Articles: An Annotation Scheme and Corpus Study*. University of Sheffield.

Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. 2014. Coooolll: A Deep Learning System for Twitter Sentiment Classification. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 208–212, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. SemEval-2007 Task 15: TempEval Temporal Relation Identification. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 75–80, Prague, Czech Republic, June. Association for Computational Linguistics.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 Task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 57–62, Uppsala, Sweden, July. Association for Computational Linguistics.

Modeling Context-sensitive Selectional Preference with Distributed Representations

Naoya Inoue Yuichiro Matsubayashi Masayuki Ono* Naoaki Okazaki Kentaro Inui
Graduate School of Information Sciences
Tohoku University
6-6-05 Aramaki Aza Aoba, Aobaku, Sendai, Miyagi 980-8579, Japan
{naoya-i,y-matsu,masayuki.ono,okazaki,inui}@ecei.tohoku.ac.jp

Abstract

This paper proposes a novel problem setting of selectional preference (SP) between a predicate and its arguments, called as *context-sensitive* SP (CSP). CSP models the narrative consistency between the predicate and preceding contexts of its arguments, in addition to the conventional SP based on semantic types. Furthermore, we present a novel CSP model that extends the neural SP model (Van de Cruys, 2014) to incorporate contextual information into the distributed representations of arguments. Experimental results demonstrate that the proposed CSP model successfully learns CSP and outperforms the conventional SP model in coreference cluster ranking.

1 Introduction

Selectional Preference (SP) of predicates is a term denoting a bias in co-occurrence of a predicate and its argument. Predicates tend to take a particular semantic type of phrase as an argument. For example, the object slot of *eat* is generally filled by a noun phrase denoting food such as *an apple*; it is rarely filled by a phrase that is not food such as *a watch*. As the knowledge of SP has been recognized as key for many natural language processing tasks, including semantic role labeling and anaphora resolution, automatic acquisition of SP knowledge has persisted as a popular research topic. In literature, a variety of computational models for SP have been proposed, ranging from thesaurus-based approaches (Resnik, 1996), to probabilistic latent variable models (Rooth et al., 1999; Séaghdha and Korhonen, 2014), and distributed approaches (Van de Cruys, 2014).

Conventionally, SP is defined as the context-independent acceptability of a word as a filler of a predicate *in the sense of a semantic type*. Suppose that we must identify the referent of $him_{(j)}$:

(1) $John_{(i)}$ *beat* $Bob_{(j)}$. *Mary comforted* $him_{(j)}$.

Henceforth, we call a predicate (e.g., *comfort*) and an argument (e.g., *John* and *Bob*) to be examined as a *query predicate* and *query argument*, respectively. Conventional SP models judge the appropriateness of $John_{(i)}$ and $Bob_{(j)}$ in terms of whether *comfort* can take each noun as its object. However, it ignores the information signified by the preceding context, namely $John_{(i)}$ *beat Bob* and $Bob_{(j)}$, *whom John beat*. Therefore, conventional approaches cannot determine the preference between *John* and *Bob*, both of whom can fill the object of *comfort*, with the same semantic type.

In this paper, we propose a *context-sensitive* version of SP (CSP), a novel task setting in which SP is considered in discourse. In text (1), for instance, $Bob_{(j)}$ is considered to be a more plausible filler than $John_{(i)}$ in terms of contextual relatedness: *one who was beaten* is more likely to be *comforted* than *one who beat someone*. In this paper, we want to discriminate (2a) from (2b) in addition to the conventional SP-based discrimination (e.g., *Mary comforted John* versus *Mary comforted a banana*):

(2) a. *Mary comforted John, who beat Bob.*
b. *Mary comforted Bob, whom John beat.*

*Present affiliation: FUJITSU FIP CORPORATION

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2829–2838, Osaka, Japan, December 11-17 2016.

The goal of this paper is to develop a CSP model that jointly considers the following aspects: (i) the conventional acceptability based on the semantic type of a query argument, and (ii) the narrative consistency between events denoted by a query predicate and preceding context of the query argument (as seen in (3a) with its counter example (3b)).

(3) a. *Mary comforted X who beat Bob.*

b. *Mary comforted X whom John beat.*

The joint modeling has an advantage in applications, such as predicate argument structure analysis and coreference resolution because the preceding context of a given query argument may not always be available in these tasks. For example, in pronoun resolution, some candidate antecedents may have preceding contexts relevant to narrative consistency but other candidates may not.

The challenges in modeling a CSP are as follows: (i) data sparseness caused by the incorporation of context words, and (ii) an effective means of incorporating context-sensitivity into SP. To address these issues, we propose to extend the state-of-the-art SP model by using a distributed representation (Van de Cruys, 2014). The distributed framework alleviates the data sparseness problem and naturally injects the contextual information of a query argument into its word vector based on compositional distributional semantics (Socher et al., 2012; Socher et al., 2013; Muraoka et al., 2014; Hashimoto et al., 2014, etc.).

We empirically evaluate the impacts of incorporating context-sensitivity into SP for two tasks: (i) context-sensitive pseudo-disambiguation, a novel benchmark tailored for evaluating CSP models, and (ii) coreference cluster ranking for pronominal anaphora resolution. The results demonstrate that our approach achieves considerable improvements. Moreover, the results suggest that CSP is a meaningful problem setting and that our model captures the context-sensitivity of SP.

2 Related Work

A fundamental approach to modeling SP is to count the co-occurrences of predicates and their arguments on a large corpus. As simply counting a predicate-argument pair causes data sparseness problem, previous SP models adopted methods for smoothing co-occurrence counts. Earlier efforts combined a manually crafted thesaurus with the acquired distribution (Resnik, 1996; Li and Abe, 1998). Another approach used a latent probabilistic model to obtain a semantically smoothed probability distribution (Rooth et al., 1999; Séaghdha and Korhonen, 2014). Other directions include example-based approaches (Erk, 2007). However, these studies differ from ours in that they do not consider the context-sensitivity.

Some previous studies (Ritter et al., 2010; Van de Cruys, 2014; Kawahara et al., 2014) estimate the plausibility of a subject–verb–object (SVO) tuple. These studies model a type of CSP: a subject or an object can be regarded as an additional context to restrict a set of possible fillers of a query predicate. However, the context captured in our study is not a local context of a query *predicate* but that of a query *argument*, working as a *validator* of the narrative consistency between a query predicate and events in which a query argument participates (see Section 4).

In addition, the modeling of a narrative consistency between events has been studied extensively (Chambers and Jurafsky, 2009; Modi and Titov, 2014; Granroth-wilding and Clark, 2016, etc.). Chambers and Jurafsky (2009) acquired sets of narratively related events sharing at least one entity (e.g., $\{X$ *commit a crime, police arrest* X*,* X *convict, ...*$\}$) by collecting a set of verbal mentions sharing coreferring arguments in a large corpus. The relatedness between two events was then estimated statistically through pointwise mutual information (Church and Hanks, 1990).

To address the data sparseness problem of Chambers and Jurafsky (2009), Granroth-wilding and Clark (2016) proposed an architecture based on distributed representation to judge the narrative coherence between two events. They trained a neural network (NN) model based on event chain instances acquired in the same strategy as that of Chambers and Jurafsky (2009) and reported that the NN model outperformed their approach. As argued in Section 4, our approach can be regarded as an integrated framework of the state-of-the-art approaches of conventional SP and narrative consistency.

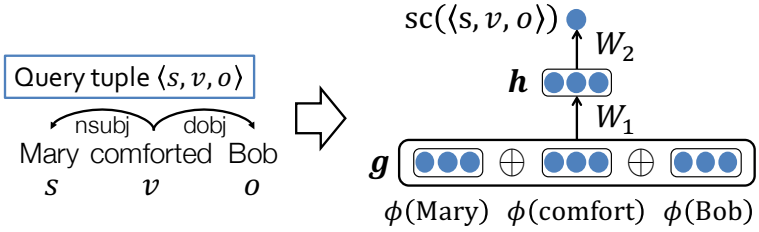


Figure 1: Van de Cruys' SVO model

3 Van de Cruys' SVO model

We adopted the SVO model by Van de Cruys (2014) as a baseline model and extended it to capture narrative consistency. Van de Cruys' SVO model, based on an NN architecture, estimates a preference score for a tuple of words $\langle s, v, o\rangle$ (referred to as a *query tuple*), in which s and o respectively correspond to the subject and object of a transitive verb v. Figure 1 presents the NN structure. The SP score $\mathrm{sc}(\cdot)$ of $\langle s, v, o\rangle$ is calculated using a two-layer NN:

$$\mathrm{sc}(\langle s, v, o\rangle) = W_2 \boldsymbol{h}_{s,v,o}, \tag{1}$$

$$\boldsymbol{h}_{s,v,o} = f(W_1 \boldsymbol{g}_{s,v,o} + \boldsymbol{b}), \tag{2}$$

$$\boldsymbol{g}_{s,v,o} = \phi(s) \oplus \phi(v) \oplus \phi(o) \tag{3}$$

where $\phi(w) \in \mathbb{R}^d$ is the vector representation[1] of word w, $\boldsymbol{g} \in \mathbb{R}^{3d}$ presents an input layer concatenating word vectors of $\langle s, v, o\rangle$ by using the operator $\oplus$, and $\boldsymbol{h} \in \mathbb{R}^h$ is a hidden layer. $W_1 \in \mathbb{R}^{h \times 3d}$ and $W_2 \in \mathbb{R}^{1 \times h}$ are respectively the weight matrices of the first and second layers, and $\boldsymbol{b} \in \mathbb{R}^h$ is a bias on the first layer. $f(\cdot)$ is an element-wise activation function using $\tanh$.

The model simultaneously learns word embedding and scoring function of SP based on the framework proposed by Collobert et al. (2011), who employed a ranking-type loss function that discriminates between positive and negative examples. Positive training examples include $\langle s, v, o\rangle$ tuples observed in a corpus. Negative examples are generated from the positive examples by replacing arguments in correct tuples with randomly selected words. This procedure generates the following three types of negative examples from a positive example $\langle s, v, o\rangle$: $\langle \tilde{s}, v, o\rangle$, $\langle s, v, \tilde{o}\rangle$, and $\langle \tilde{s}, v, \tilde{o}\rangle$. The loss function is then defined as:

$$\sum_{(s,v,o)} \big\{ \max(0, 1 - \mathrm{sc}(\langle s, v, o\rangle) + \mathrm{sc}(\langle \tilde{s}, v, o\rangle)) + \max(0, 1 - \mathrm{sc}(\langle s, v, o\rangle) + \mathrm{sc}(\langle s, v, \tilde{o}\rangle)) \\ + \max(0, 1 - \mathrm{sc}(\langle s, v, o\rangle) + \mathrm{sc}(\langle \tilde{s}, v, \tilde{o}\rangle)) \big\}. \tag{4}$$

Following Collobert et al. (2011), Van de Cruys (2014) calculates the gradient of the loss *online* by sampling a single corrupted subject $\tilde{s}$ and object $\tilde{o}$ for each correct tuple.

4 Context-sensitive SP Model

We propose a context-sensitive selectional preference (CSP) model by extending the SVO model. The advantage of using the model by Van de Cruys (2014) is that we can naturally represent the attachment of the contextual information into a query argument with compositional distribution semantics (Socher et al., 2012; Hashimoto et al., 2014, etc.). We incorporate both the conventional SP and narrative consistency described in Section 1 into a single model by learning the vector representation of *an argument with its context* in the same vector space as word vectors in the SVO model. To realize this, we inserted an additional layer for calculating a context-injected word vector under the input layer.

Figure 2 presents the network structure of our model with the following text as an input.

[1]The original paper differentiates vectors for a word w depending on whether it is used as a subject, object, or verb. However, we used the same vector for a word because we obtained higher performance than that setting.

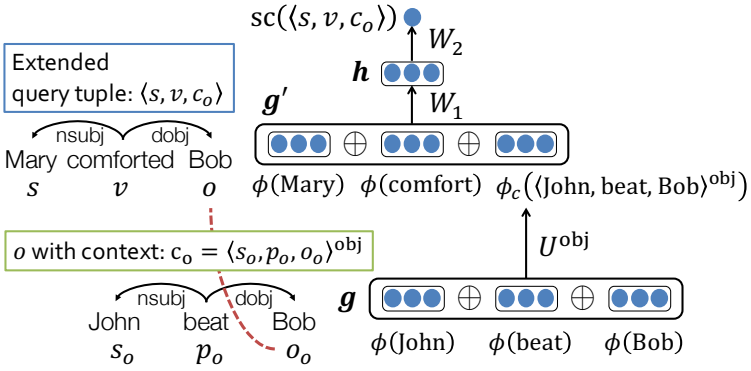


Figure 2: Network structure of proposed model

(4) *John beat Bob*$_{(i)}$. *Mary comforted Bob*$_{(i)}$.

Here, the query tuple $\langle s, v, o\rangle$ is $\langle$*Mary, comfort, Bob*$\rangle$ and the context for the query argument *Bob*$_{(i)}$ is "*John beat Bob*." From this context, we calculate a context-injected word vector representing "*Bob, whom John beat*" by combining the vectors of the words in its predicate-argument structure (PAS). We use the resulting vector as the input to the SVO model, instead of the vanilla word vector.

Formally, we extend the representation of query tuple $\langle s, v, o\rangle$ so that s and o can accompany their contexts. We represent such cases as $\langle c_s, v, o\rangle$, $\langle s, v, c_o\rangle$, and $\langle c_s, v, c_o\rangle$, where c_w denotes *a word w with its context*. Then, we extend Equations (1), (2), and (3) for $\langle s, v, c_o\rangle$ as

$$\mathrm{sc}(\langle s, v, c_o\rangle) = W_2 \boldsymbol{h}'_{s,v,c_o}, \tag{5}$$

$$\boldsymbol{h}'_{s,v,c_o} = f(W_1 \boldsymbol{g}'_{s,v,c_o} + \boldsymbol{b}), \tag{6}$$

$$\boldsymbol{g}'_{s,v,c_o} = f(\phi(s) \oplus \phi(v) \oplus \phi_c(c_o)), \tag{7}$$

where $\phi_c(c_w)$ is a context-injected word vector, which is explained in the rest of this section. Similarly, for $\langle c_s, v, o\rangle$ and $\langle c_s, v, c_o\rangle$, we extend Equation (3) as follows: $\boldsymbol{g}'_{c_s,v,o} = f(\phi_c(c_s) \oplus \phi(v) \oplus \phi(o))$, and $\boldsymbol{g}'_{c_s,v,c_o} = f(\phi_c(c_s) \oplus \phi(v) \oplus \phi_c(c_o))$.

As a context of w, we can potentially consider various types of modifiers, such as predicates, adverbs, appositives, and genitives, that affects the preference score of a query argument. In this study, as a first step, we restrict the context information to PASs along the lines of the previous studies that utilize event-to-event relations in an anaphora resolution (Inoue et al., 2012; Peng et al., 2015).

Specifically, we first assume a context of a query argument be a single PAS which takes the query argument as its argument (referred to as a *context-PAS*). Then we represent w *with its context-PAS* as $c_w = \langle s_w, p_w, o_w\rangle^r$, where s_w and o_w are respectively the subject and object of a predicate p_w that syntactically governs the word w (i.e., either s_w or o_w is w). r indicates the grammatical role of the query argument w in the context-PAS, whose value is either subj (when $w = s_w$) or obj (when $w = o_w$). For example, for text (4), the context for the query object *Bob*, modified by the PAS of the transitive verb *beat*, is represented as $c_o = \langle$*John, beat, Bob*$\rangle^{\mathrm{obj}}$. The grammatical role obj of the query argument *Bob* is indicated to discriminate "*Bob, whom John beat*" from "*John, who beat Bob*." Note that some context-PASs (e.g., intransitive verbs and adjectives) do not take an object argument. r and o_w for these ASs are thus ignored.

To compute a context-injected vector of a query argument, we composed a word vector in a context-PAS by using methods similar to those for building phrase vectors (Socher et al., 2012; Socher et al., 2013; Muraoka et al., 2014; Hashimoto et al., 2014). In this study, we adopted a simple compositional operation, keeping comparisons with other composition methods for future studies. We compute the context-injected word vector $\phi_c(c_w)$ as

$$\phi_c(\langle s_w, p_w, o_w\rangle^r) = U^r \boldsymbol{g}_{s_w,p_w,o_w}, \tag{8}$$

$$\boldsymbol{g}_{s_w,p_w,o_w} = f(\phi(s_w) \oplus \phi(p_w) \oplus \phi(o_w)), \tag{9}$$

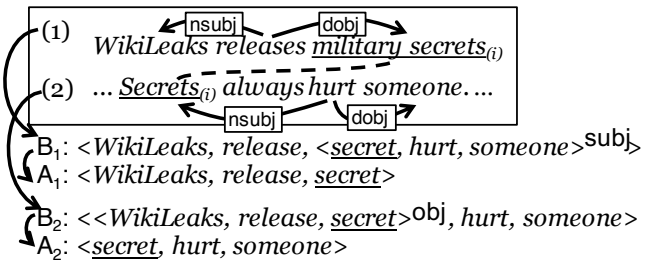


Figure 3: Generation of training instances.

where $U^r \in \mathbb{R}^{h \times 3d}$ is a weight matrix used for building the context-injected vector for the grammatical role r. We set a word vector $\phi(o_w) = \mathbf{0} \in \mathbb{R}^d$ if o_w does not exist (when the context-PAS does not have o_w).

5 Training

To model conventional SP and CSP jointly, we simultaneously train matrices U^r, W_1, and W_2, and vectors $\phi(\cdot)$. We minimize the same loss function introduced in Section 3, except that (i) we replace the score function $\mathrm{sc}(\cdot)$ with Equation (5); and (ii) we use two types of tuples (TYPE A and TYPE B) as training instances. TYPE A is a tuple whose subject and object are bare nouns (i.e., $\langle s, v, o \rangle$). TYPE B is a tuple with either its subject or object including a context-PAS (i.e., $\langle c_s, v, o \rangle$ and $\langle s, v, c_o \rangle$). Hereafter, we describe the method to obtain these instances from a corpus.

5.1 TYPE B instance generation

Positive instance. We assume that a corpus is parsed using a syntactic dependency parser and a coreference resolver. We extract a collection of TYPE B positive instances from the dependency and coreference relations, where we include only a head word for each predicate/argument slot. Figure 3 illustrates the extraction procedure. From sentence (1), we obtain $\langle$*WikiLeaks, release,* $\langle$*secret, hurt, someone*$\rangle^{\mathrm{subj}}\rangle$ (B_1), where "*hurt someone*", the context of *secret*, is attached via the coreference link between *military secrets* and *Secrets*. Similarly, from sentence (2), we obtain $\langle\langle$*WikiLeaks, release, secret*$\rangle^{\mathrm{obj}}$*, hurt, someone*$\rangle$ (B_2).

As context-PASs, we used a *non-negated* transitive verb, intransitive verb, adjective, and copula. Thus, we do *not* extract tuples including one or more negations (e.g., $\langle$John, *not* eat, apple$\rangle$). A nontrivial issue of managing negations in compositional semantics has not been explored much in distributional compositional semantics. Furthermore, we removed tuples where the predicates of coreferent mentions are connected via an *adversative* connective (e.g., *John beat Bob but Bob was happy*), which cannot be handled by the CSP model.

Negative instance. Next, negative training instances are generated by considering positive instances as counterparts. We generate $\langle \tilde{c_s}, v, o \rangle$, $\langle c_s, v, \tilde{o} \rangle$, and $\langle \tilde{c_s}, v, \tilde{o} \rangle$ for positive instance $\langle c_s, v, o \rangle$ and $\langle \tilde{s}, v, c_o \rangle$, $\langle s, v, \tilde{c_o} \rangle$, and $\langle \tilde{s}, v, \tilde{c_o} \rangle$ for positive instance $\langle s, v, c_o \rangle$. Here, $\tilde{s}$ and $\tilde{o}$ are sampled from all the bare subjects and objects in the positive instances, respectively. In addition, $\tilde{c_s}$ and $\tilde{c_o}$ are sampled from all the subjects and objects respectively, with contexts attached. For example, we generate $\tilde{c_s}$=$\langle$*John, eat, apple*$\rangle^{\mathrm{obj}}$ from c_s=$\langle$*WikiLeaks, release, secret*$\rangle^{\mathrm{obj}}$. We use the probabilistic negative sampling (Mikolov et al., 2013) based on the frequency of arguments in the positive instances[2].

5.2 TYPE A instance generation

We then created a TYPE A positive instance from each TYPE B positive instance by replacing the context-attached argument with the original bare argument. In Figure 3, we obtain $\langle$*WikiLeaks, release, secret*$\rangle$ and $\langle$*secret, hurt, someone*$\rangle$ from B_1 and B_2, respectively. To generate TYPE A negative instance, we follow the same procedure described in Section 3 but use probabilistic negative sampling instead.

[2]Although Van de Cruys (2014) used random sampling to generate negative instances, we used probabilistic sampling because our preliminary experiment shows a better performance.

5.3 Dataset

We identified syntactic dependency relations and coreference relations in 4.5 billion sentences extracted from the ClueWeb12 corpus[3], that is, a large collection of Web documents, by applying Stanford CoreNLP (Manning et al., 2014). To reduce noises from the obtained coreference relations, we applied the following heuristics to skim only highly plausible coreference relations off from the pool: (i) the coreference relation must be *intrasentential*, (ii) the head words of the coreferent mentions must be *identical* and *non*pronominal (e.g. *John–John* but not *John–boy*)[4]. Furthermore, we discarded TYPE A and TYPE B instances containing low-frequency words so that all our training instances include only the top 50k frequent verbs, 50k frequent nouns, and 50k frequent adjectives. We replaced all the rare words (occurring less than four times) with the special symbol `OOV` (implying out of vocabulary) to facilitate the calculation of the SP of unseen words appearing in the test set.

As a result, we obtained a collection of 4,824,394 TYPE B positive instances (2,912,624 unique tuples; $\mathcal{B}$ hereafter) and 4,824,394 TYPE A positive instances (1,500,990 unique tuples; $\mathcal{A}$ hereafter).

6 Evaluation

To check whether the CSP model can properly learn the conventional SP and narrative consistency, we first evaluated the CSP model against Van de Cruys' model by using a pseudo-disambiguation test, a binary classification task of discriminating a positive SVO tuple from its pseudo-negative counterpart. We then evaluated the effectiveness of the CSP model in a realistic problem setting, in which the disambiguation test is created from coreference annotations of the OntoNotes corpus (Hovy et al., 2006).

6.1 Parameters

We set the dimension of word embedding $d = 50$ and the dimension of hidden layer $h = 50$. The word embeddings are initialized with the publicly available word vectors trained through GloVe (Pennington et al., 2014)[5] and updated through back propagation. We updated weights by using Adam (Kingma and Ba, 2014) with a mini-batch size of 1,000 and 30 epochs[6].

To evaluate the effectiveness of the CSP model, we replicated the SVO model of Van de Cruys (2014) by training the CSP model only with TYPE A instances (henceforth, SP).

6.2 Pseudo-disambiguation test

Inspired by the conventional SP model (Erk, 2007; Van de Cruys, 2014, etc.), we set up three binary classification tasks. We performed hold-out validation on datasets $\mathcal{A}$ and $\mathcal{B}$.

6.2.1 Tasks

Pseudo-disambiguation (PD) discriminates a positive *non*-context-injected tuple (e.g., $\langle$*Mary, eat, banana*$\rangle$) from its pseudo-negative counterpart ($\langle$*Mary, eat, watch*$\rangle$) without any context information. This task setting has been employed in the previous SP models, including in Van de Cruys (2014).

Context-sensitive PD (CSPD) discriminates a positive context-attached tuple (e.g., $\langle$*Mary, eat,* $\langle$*banana, delicious*$\rangle^{\mathrm{subj}}\rangle$) from its pseudo-negative counterpart ($\langle$*Mary, eat,* $\langle$*watch, new*$\rangle^{\mathrm{subj}}\rangle$). This is a novel task setting designed to highlight the ability of modeling both conventional SP and narrative consistency.

CSPD-X is the same as CSPD except that the coreferent arguments are masked; namely, the task is to discriminate a masked context-attached tuple (e.g. $\langle$*Mary, eat,* $\langle$*X, delicious*$\rangle^{\mathrm{subj}}\rangle$, where *X* denotes the special symbol for masked arguments) from its masked pseudo-negative counterpart. In this task, we set the word vector for the masked argument $\phi(X) = \mathbf{0}$. Forcing the ignorance of the meaning of an argument, this task can assess how precisely the proposed models can predict a query argument through narrative consistency only (i.e., by relying only on the context information).

[3] `http://lemurproject.org/clueweb12/`

[4] A manual inspection revealed that the filtering improved the accuracy of the coreference relations from 71% to 87%.

[5] `http://nlp.stanford.edu/projects/glove/`

[6] All the parameters were determined through our preliminary experiments; we found that all the models were insensitive to the dimension parameters (25, 50, and 100 were explored). An initialization with GloVe performed better than random initialization, and the update of a word vector had a positive impact.

Model	PD	CSPD	CSPD-X
RANDOM	0.5000†	0.5000†	0.5000†
SP	**0.8635**	0.8635	0.5000†
CSP	0.8623	**0.8947***	**0.7856**

Table 1: Accuracy for pseudo-disambiguation tasks. '*' denotes a statistical significance against SP (McNemar test, $p < .05$). '†' indicates the accuracy on random guesses (no clue for discrimination).

Model	MQ	MQ_{no_pr}
SP	0.7420	0.7125
CSP	**0.8265***	0.7586*

Table 2: Model performance on entity ranking. '*' indicates statistical significance against SP (Wilcoxon signed-rank test, $p < .05$).

6.2.2 Dataset

To perform hold-out validation, we first randomly divided the dataset $\mathcal{B}$ into a training set (90%, $\mathcal{B}_{\text{train}}$) and a test set (10%, $\mathcal{B}_{\text{test}}$)[7]. For the PD task, we extracted $\mathcal{A}_{\text{train}}$ from $\mathcal{B}_{\text{train}}$ and $\mathcal{A}_{\text{test}}$ from $\mathcal{B}_{\text{test}}$ by using the procedure described in Section 5.1. For the CSPD and CSPD-X tasks, we used $\mathcal{B}_{\text{train}}$ and $\mathcal{B}_{\text{test}}$. Note that $\mathcal{A}_{\text{train}}$ includes all the SVO instances included in $\mathcal{B}_{\text{train}}$.

6.2.3 Results

Table 1 reports the accuracy of each model in this task. A subtle performance drop (≤ 0.0013 point) is observed in our CSP compared to SP; this was not statistically significant (the statistical significance test by McNemar (1947) showed $p < .05$). This indicates that our joint modeling does not degrade the ability for modeling a conventional SP. In contrast, the CSP model significantly outperformed SP (McNemar test, $p < .05$) in the CSPD task, capturing the context-sensitivity of SP successfully. The results of CSPD-X imply that our joint modeling can properly learn narrative consistency.

6.3 Ranking coreference clusters

In Section 6.2, we reported the results of a binary classification task in which negative instances were artificially generated. In contrast, this section describes a more realistic task setting: *ranking* coreference clusters in the OntoNotes corpus (Hovy et al., 2006).

6.3.1 Task

Given a target pronoun, our task is to determine the coreference cluster (*entity*) that is the most likely to be coreferent with the pronoun. Let us consider text (5) as an example.

(5) *In* his$_{(i)}$ *40-minute* speech$_{(j)}$, Chen$_{(i)}$ *declared the* determination$_{(k)}$ *of the* people$_{(l)}$ ... *against* Chen$_{(i)}$..., *and* he$_{(?)}$ *made a statement...*

Given a target pronoun *he*$_{(?)}$, four coreference clusters C_i, C_j, C_k, C_l are used as candidates for antecedents. As *he*$_{(?)}$ is a subject of the predicate *made a statement*, $\text{sc}(\langle c_s, v, o\rangle)$ gives the preference of c_s as an antecedent of the pronoun, where $v =$ *made* and $o =$ *statement*. In the example, c_s can be a preceding noun with its context attached (if any) or without its context: $\langle$*Chen*, *declare*, *determination*$\rangle^{\text{subj}}$, $\langle$*Chen*, *declare*, *determination*$\rangle^{\text{obj}}$, *speech*, or *people*. We expect that an SP model prefers the correct antecedent $\langle$*Chen*, *declare*, *determination*$\rangle^{\text{subj}}$ over the others.

We measure the ability of a model for selecting the correct cluster by using a *mean quantile* (MQ) (Guu et al., 2015). Let p be the target pronoun, C_p^+ the correct cluster for the pronoun, and $\mathcal{N}_p$ the set of negative (incorrect) clusters. MQ for the pronoun p is defined as:

$$\text{MQ}(p) = \frac{|\{C^- \in \mathcal{N}_p | \text{sp}(C^-, p) < \text{sp}(C_p^+, p)\}|}{|\mathcal{N}_p|}. \tag{10}$$

Intuitively, $\text{MQ}(p)$ represents the ratio where a correct cluster C_p^+ is preferred to incorrect clusters $C^- \in \mathcal{N}_p$ by the model. In general, a cluster contains multiple mentions; thus, we simply consider the maximal of the scores in a cluster C: for example, $\text{sp}(C, p) = \max_{m \in C} \text{sc}(m, v, o)$.

[7]To ensure that the two subsets are strictly disjoint, we prohibited a test instance from being an inverse of any instance in the training set; more concretely, the instance $\langle$*WikiLeaks, release,* $\langle$*secret, hurt, someone*$\rangle^{\text{subj}}\rangle$ must not exist in the test set when the training set includes its inverse $\langle\langle$*WikiLeaks, release, secret*$\rangle^{\text{obj}}$*, hurt, someone*$\rangle$.

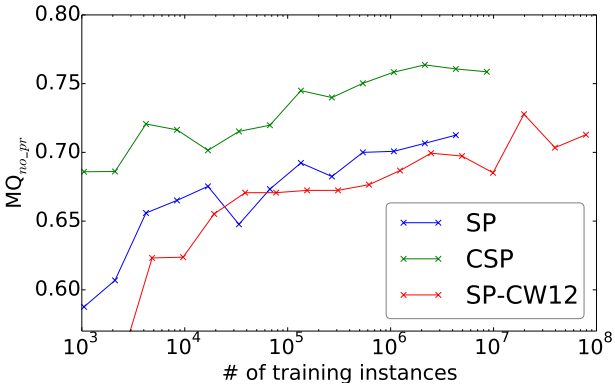


Figure 4: Learning curves of SP and CSP.

When a target pronoun appears at an object position, we use $\mathrm{sc}(\langle s, v, c_o \rangle)$ instead of $\mathrm{sc}(\langle c_s, v, o \rangle)$. In this experiment, we targeted only pronouns filling the subject or object slot of a *non-negated* transitive verb (see Section 5.3); we extracted *he*$_{(i)}$ but not *his*$_{(i)}$ in text (5).

6.3.2 Test set

We used the coreference annotations in OntoNotes Corpus 5.0 (Hovy et al., 2006). The corpus includes 625k newswire and 400k broadcast articles annotated with several layers of syntactic and semantic annotations. We obtained 16,414 test pronouns (16.6% of the total pronouns) with the average number of candidate coreference clusters of 75.9. Further, we used semantic roles to extract PASs.

Although pronouns are not informative to SP and CSP, CSP is expected to benefit from context-attached tuples. To test this, we enhanced the training dataset described in Section 5.1 by allowing pronominal coreferent mentions to be extracted because pronouns might appear in coreference clusters. Thus, we obtained a collection of 8,603,782 TYPE B positive instances (4,952,462 unique tuples; $\mathcal{B}_{pro}$ hereafter) and 8,603,782 TYPE A positive instances (2,178,540 unique tuples; $\mathcal{A}_{pro}$ hereafter). We trained SP with $\mathcal{A}_{pro}$, and CSP with $\mathcal{A}_{pro}$ and $\mathcal{B}_{pro}$.

6.3.3 Results

The MQ column of Table 2 shows the mean of the MQ scores for all target pronouns. The proposed model (CSP) outperformed the baseline model (SP) (Van de Cruys, 2014) (Wilcoxon signed-rank test, $p < .05$). The results indicate that the CSP model captures the SPs of predicates more precisely by exploiting the context information of coreference clusters. In addition, our joint models are demonstrated to be capable of comparing query arguments regardless of the existence of context information.

It may be presumed that this improvement is due to the task setting: pronominal coreferent clusters are relatively difficult for SP to discriminate because SP cannot exploit contextual information. Thus, we also report MQ$_{no_pr}$ in Table 2, which is the MQ of the coreference cluster ranking task including only nonpronominal nouns as candidate coreference clusters. This evaluation also shows that our CSP model outperformed the model by Van de Cruys (2014) (Wilcoxon signed-rank test, $p < .05$). The margin in MQ was found to be larger than in MQ$_{no_pr}$. This indicates that the CSP model successfully captures the narrative consistency-based SP of a pronoun that is difficult to be captured by SP. We leave the effectiveness of context-sensitivity on other (non-neural) types of conventional SP models (e.g., probabilistic latent models (Séaghdha and Korhonen, 2014, etc.)) as an open question. The primary goal of this study is to check the effectiveness of context information; this is proven by the aforementioned results.

6.3.4 Analysis

Although CSP outperformed SP for two tasks, the superiority of CSP may be argued to be due to the larger number of training instances; SP is trained on $\mathcal{A}_{pro}$ but CSP is trained on both $\mathcal{A}_{pro}$ and $\mathcal{B}_{pro}$. Therefore, we plotted the learning curves (`SP` and `CSP` in Figure 4), trained SP and CSP on subsets of

$\mathcal{A}_{pro}$ and $\mathcal{B}_{pro}$ sampled randomly, and measured MQs on the models. As we can generate much a larger size of TYPE A training instance from dependency parses, we also plot the learning curve of the SP model trained using extra SVO tuples extracted from the dependency parse of ClueWeb12 (SP-CW12). We extracted 316,063,648 SVO instances and trained the SP model by using up to 25% of all the extracted SVO tuples (77,011,125 instances) because of the computational cost of training. For fair comparison, we used MQ_{no_pr} as an evaluation measure.

The results show that the MQs of SP (SP, SP-CW12), and CSP (CSP) increase with the size of training data, and both models grow together, keeping a large margin. Based on the growth rate of SP and SP-CW12, we conjectured that we needed 10^3 times more instances of TYPE A for SP to reach the same performance as that of CSP. However, it is inefficient and unrealistic to increase the number of training instances of TYPE A in terms of the training time and availability of training data. In contrast, the CSP model leverages narrative consistency to SP, which is never addressed in previous studies.

To deeply analyze the CSP model, we investigated how the ranks of correct coreference cluster changed from SP to CSP. We found that MQ changed by 0.5 or more in 768 test instances, including 538 improvements and 230 degradations. For 75.7% of the improvements, a context was found to be attached to the candidate antecedent, which is maximally scored among a correct coreference cluster. For example, for the test pronoun *it* in $\langle$*you, own,* $\underline{it}\rangle$, the CSP model can rank the correct antecedent $\langle$*you, buy, something*$\rangle^{\text{obj}}$ at the top by capturing the narrative consistency between *buy X–own X*. In contrast, the context is attached to a correct coreference cluster in only 31.5% of the degradations. This indicates that the CSP model improves the conventional SP via narrative consistency.

7 Conclusion

We addressed the problem of CSP, the novel problem setting of SP. By extending the state-of-the-art SP model (Van de Cruys, 2014), we proposed the novel model that jointly learns both the conventional SP and narrative consistency between a query predicate and its context predicate. The experiments on the PD task demonstrated that the CSP model could leverage narrative consistency for predicting preferences of predicates. Furthermore, the CSP model is effective in a more realistic task setting, ranking coreference clusters.

In the immediate future, we will explore broader contextual information (e.g., prepositional attachment), which can be implemented in our framework naturally. We are interested in applying recent advances in NN, for example, Long Short-Term Memory, Gated Recurrent Unit, and attention mechanism, to compute the vector representation integrating multiple pieces of contextual information. In the experiments, we reserved a downstream application-oriented evaluation for future study so as to exclude various factors specific to a downstream application from the modeling of the CSP. We will explore how to integrate the CSP model with a neural coreference resolver (e.g., Wiseman et al. (2016)) together with conventional coreference features (e.g., distance between a candidate antecedent and a query predicate).

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Numbers 15H01702, 15H05318, 15K16045, and CREST, JST.

References

Nathanael Chambers and Daniel Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *ACL*, pages 602–610.

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

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537.

Katrin Erk. 2007. A simple, similarity-based model for selectional preferences. In *ACL*, pages 216–223.

Mark Granroth-wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *AAAI*.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *EMNLP*, pages 318–327.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2014. Jointly learning word representations and composition functions using predicate-argument structures. In *EMNLP*, pages 1544–1555.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% solution. In *NAACL (short papers)*, pages 57–60.

Naoya Inoue, Ekaterina Ovchinnikova, Kentaro Inui, and Jerry Hobbs. 2012. Coreference resolution with ILP-based weighted abduction. In *COLING*, pages 1291–1308.

Daisuke Kawahara, Daniel W Peterson, and Martha Palmer. 2014. A step-wise usage-based method for inducing polysemy-aware verb classes. In *ACL*, pages 1030–1040.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Hang Li and Naoki Abe. 1998. Generalizing case frames using a thesaurus and the mdl principle. *CL*, 24(2):11.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL*, pages 55–60.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. *CoNLL*, pages 49–57.

Masayasu Muraoka, Sonse Shimaoka, Kazeto Yamamoto, Yotaro Watanabe, Naoaki Okazaki, and Kentaro Inui. 2014. Finding the best model among representative compositional models. In *PACLIC 28*, pages 65–74.

Haoruo Peng, Daniel Khashabi, and Dan Roth. 2015. Solving hard coreference problems. In *NAACL*, pages 809–819.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Philip Resnik. 1996. Selectional constraints: An information-theoretic model and its computational realization. *Cognition*, 61(1):127–159.

Alan Ritter, Oren Etzioni, et al. 2010. A latent dirichlet allocation method for selectional preferences. In *ACL*, pages 424–434.

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via em-based clustering. In *ACL*, pages 104–111.

Diarmuid Séaghdha and Anna Korhonen. 2014. Probabilistic distributional semantics with latent variable models. *CL*, 40(3):587–631, September.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*, pages 1201–1211.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *ACL*, pages 455–465.

Tim Van de Cruys. 2014. A neural network approach to selectional preference acquisition. In *EMNLP*, pages 26–35.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *NAACL*, pages 994–1004.

Exploring the value space of attributes: Unsupervised bidirectional clustering of adjectives in German

Wiebke Petersen
Düsseldorf University, SFB 991
petersen@
phil.uni-duesseldorf.de

Oliver Hellwig
Düsseldorf University, SFB 991
ohellwig@
phil-fak.uni-duesseldorf.de

Abstract

The paper presents an iterative bidirectional clustering of adjectives and nouns based on a co-occurrence matrix. The clustering method combines a Vector Space Models (VSM) and the results of a Latent Dirichlet Allocation (LDA), whose results are merged in each iterative step. The aim is to derive a clustering of German adjectives that reflects latent semantic classes of adjectives, and that can be used to induce frame-based representations of nouns in a later step. We are able to show that the method induces meaningful groups of adjectives, and that it outperforms a baseline k-means algorithm.

1 Introduction

The research presented in this paper is part of a larger project which aims at the semantic analysis of adjectival modification of nouns in German in a frame-based approach. Its approach is to model the conceptual mechanisms underlying adjectival modification by decomposing the meanings of adjectives (A) and nouns (N) in frames, i.e. in recursive attribute-value structures. The most common modification process is that the noun concept bears an attribute whose value is restricted by the adjective. The examples in (1) show that some adjectives are strongly associated with an attribute like the color attributes or the attributes of size or age. Some of these adjectives do not only require a special attribute but also a special noun concept (like the color adjective 'blond' that only applies to 'hair'). For other adjectives like *städtisch* 'urban' it is less clear the value of which attribute they specify.

(1) a. schwarzer Ball / Stift
black ball / pen
b. blondes Haar
blond hair
c. fröhlicher Junge / Abend
happy boy / evening
d. städtische Schule
city$_{adj}$ school
e. kindlicher Organismus
child$_{adj}$ organism
f. schneller Fahrer
fast driver

As adjectives often merely restrict the value of an attribute that is associated with the adjective, as 'black' modifying the attribute color in (1-a) such that $\text{COLOR}(ball) = black$, attribute-value structures are well-suited for the task of modeling adjectival modification (cf. Pustejovsky, 1995). However, flat attribute-value lists are not sufficient, because 'black pen' can denote a pen that is black, or a pen that writes in black, $\text{COLOR}(\textit{writing of the pen}) = black$. Similarly in (1-c), it cannot be the

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2839–2848, Osaka, Japan, December 11-17 2016.

same HAPPINESS-attribute applying to both 'boy' and 'evening' in the same way. Relational adjectives like 'städtisch' in (1-d) express a relation between the denotation of N and of the root of A, although the kind of relation is rather vague, a.o. OWNER(*school*) = *city*, OPERATOR(*school*) = *city*, LOCATION(*school*) = *city*. In cases like (1-e) where the noun is relational (cf. de Bruin and Scha, 1988; Löbner, 2011), the adjective fills in the argument slot, POSSESSOR(*organism*) = *child*. Finally, in an event-related A+N phrase as in (1-f) an event frame is activated and modified by A, SPEED(*driving event*) = *fast*.[1]

The examples in (1) show that the compositional mechanisms active in A+N phrases are based on an interplay of the adjective and the noun meaning. Our aim is to cluster adjectives and nouns by these mechanisms, that means by the attributes they exhibit or modify and by the structural position of the modified attribute in the A+N frame.

2 Related research

Distributional approaches to compositionality have been the subject of several recent publications (Blacoe and Lapata, 2012). Regarding A+N phrases, research concentrates on modeling their compositional meaning (Baroni and Zamparelli, 2010; Guevara, 2010), predicting the acceptability rates of novel A+N phrases (Vecchi et al., in press), and on connecting linguistic theory with computational models (Boleda et al., 2013). As an alternative to classical Vector Space Models (VSM) (Turney and Pantel, 2010; Erk, 2012), Latent Dirichlet Allocation (LDA) and related Dirichlet process mixture models are increasingly applied to questions in lexical semantics. Séaghdha and Korhonen (2014) model selectional preferences of verbs in a Bayesian framework, and learn semantic classes of arguments from the latent variables of the model.

Our approach is closely related to recent work in distributional semantics by Hartung and Frank (2010, 2011) on selecting attributes in A+N phrases. However, their task differs from ours in that they explicitly restrict themselves to property-denoting adjectives for which an associated attribute is explicitly assigned in WordNet (Fellbaum, 1998). Hartung and Frank (2011) apply supervised variants of LDA to the problem of attribute prediction for A+N phrases. The authors filter candidate phrases from synsets in WordNet, construct separate pseudo-documents for As and Ns, and compose the output vectors of the LDA with different arithmetic operations following Baroni and Zamparelli (2010), Guevara (2010), and Hartung and Frank (2010).

Unsupervised detection of semantic classes of adjectives has been the topic of several studies. Boleda et al. (2004) annotate Catalan adjectives with two types of coarse semantic labels (unary/binary, basic/object/event), and cluster morpho-syntactic sentence patterns in which the adjectives typically occur. When setting the number of expected clusters to 2 (unary/binary) and 3 (basic/object/event), the authors observe a high correlation between their semantic labelings and the detected clusters. Furthermore, our task is related to detecting synsets and gradability of adjectives using a corpus based approach (Schulam and Fellbaum, 2010). We are interested in detecting groups of adjectives that describe the full range of a (latent) attribute for a group of nouns. So, a group of nouns that denote motorized vehicles may be characterized by the adjectives *geparkt* 'parked', *gestartet* 'started', and *fahrend* 'driving', all of which describe an attribute 'motion state' in a vehicle frame, but don't necessarily belong to a single synset, or are part of a scale of values in a traditional definition.

3 Method

We aim at deriving an adjective and noun clustering that reflects all modificational mechanisms possible in A+N phrases. Due to the explorative nature of our approach, we need a full picture of the combinatorial possibilities of As and Ns, so that we need to investigate a huge corpus. Moreover, we cannot apply supervised methods, because we do not want to restrict ourselves to a subclass of adjectives. The basic idea is to extract as many attested A+N phrases as possible and to apply an iterative bidirectional clustering algorithm based on the co-occurrences of adjectives and nouns. First, the adjectives are clustered on the basis of co-occurring nouns, then the nouns are clustered on the basis of co-occurrences with

[1] For a recent overview on adjectival modification refer to Morzycki (2015).

	distinct A+N pairs	A types	N types	density
Google n-grams	894,743	4,460	10,996	0.0182
Wikipedia	82,022	12,616	19,849	0.0003
newspaper	261,327	4,507	7,955	0,0073

Table 1: Number of A+N pairs and A and N types extracted from the corpora

adjectives from the just gained clusters. The process is iterated, until it reaches a stable clustering or a point at which new clusters would become too diverse.

As detected clusters are reinserted into the vector space matrix, our clustering produces hierarchically structured representations of As and Ns. An example will be discussed in Section 4.

3.1 Data

3.1.1 Co-occurrence data

Data for the A+N co-occurrences are extracted from the Google n-gram corpus (Michel et al., 2011), from a corpus of German newspaper texts, and from a dump of the German Wikipedia, including the Wikisources.[2] The Google and Wikipedia corpora are chosen for their sizes, but also because they contain texts from literary domains that may display other patterns of A+N usage than newspapers. Note that our focus is on the binary question of whether an adjective can modify a noun or not. Thus, throughout the A+N pair extraction process, we prioritize precision above recall. Furthermore, we are not interested in the co-occurrence frequencies (as long as each A+N pair occurs at least 5 times in the corpus), which allows us to merge the data received from the two text corpora ('newspaper' and 'wikipedia') with the Google n-gram corpus.

German marks most A+N pairs in singular number and all definite A+N pairs in plural by using articles. In addition, all nouns are capitalized, while adjectives start with lowercase letters, such that A+N pairs can be extracted from the Google n-grams by applying a regular expression (remember that we do not aim at the extraction of all A+N pairs). Raw A+N pairs detected in this way are grouped, counted, lemmatized with a full form lexicon and normalized[3], resulting in a total of 894,743 distinct A+N pairs with 4,460 A and 10,996 N types. We process the Wikipedia data by removing Wiki specific formatting, and splitting the resulting text into sentences. After POS-tagging each sentence with MATE (Bohnet and Nivre, 2012), A+N pairs are extracted by searching for the POS tag sequence ADJA + [NN|NE]. After removing AN pairs that occur less than 5 times and applying the spelling normalization, this corpus contains 82,022 distinct A+N pairs with 12,616 A and 19,849 N types. The newspaper corpus is lemmatized and POS tagged in the same way as the Wikipedia corpus. This corpus yields 261,327 distinct lemmatized and counted A+N combinations with 4,507 A and 7,955 N types.

Table 1 summarizes the results of the A+N pair extraction process for the three corpora. The small number of distinct A+N pairs found in the Wikipedia corpus compared to the large number of A and N types can be explained by the characteristics of the texts in this corpus. Many Wikipedia entries deal with highly specified topics and introduce a professional terminology; especially the introduced adjectives are mainly used in fixed expressions. Furthermore, the table shows the relevance of the Google n-gram corpus, although it is built from slightly outdated texts and contains many OCR errors (Pechenick et al., 2015). Compared to the newspaper corpus the number of A and N types in the Google corpus is similar, while the number of distinct A+N combinations is much higher. This is due to the fact that the Google corpus is extracted from a huge corpus exhibiting more of the combinatorial possibilities of As and Ns. The density values illustrate the differences between the corpora (number of A+N pairs divided by the product of the number of noun types and number of adjective types).

[2]We use Version 20120701 of the German Google n-grams, the 2013 collection from `http://www.statmt.org/wmt14/training-monolingual-news-crawl/` and the Wikipedia dump from 1st of June 2016.

[3]Because the Google and Wikipedia corpora contain several texts in old German orthography, we apply a spelling normalization that transforms, for example, *th* into *t* (*Alterthum* 'antiquity' → *Altertum*), and corrects a few obvious OCR errors (*Iahr* 'year' → *Jahr*).

3.1.2 Gold data

A gold clustering is required for the evaluation of our clustering solutions. We tested the adjective classification given in GermaNet (Hamp and Feldweg, 1997; Henrich and Hinrichs, 2010) as a gold standard for our purpose. It turned out that the data was not ideal for our task for two reasons: First, GermaNet is hierarchically structured by the hyperonymy relation. That makes it difficult to automatically detect the appropriate granularity level which corresponds to the level of attribute value specifications. Second, GermaNet is constructed on the basis of binary semantic relations like 'synonymy', 'antinonymy' and 'hyperonymy' and not on the basis of attributes. For our purpose, a classification based on attributes and their potential values and bearers would be more suitable.

As an alternative, we used a dataset consisting of all simple adjectives from a German dictionary (Duden, 2004) that are neither loanwords nor derived from other words.[4] The 278 extracted adjectives have been manually classified by 45 attributes. Attributes that take characteristic values if applied to special noun classes are separated from each other. Hence, the attribute *Haarfarbe* 'hair color' is separated from the general attribute *Farbe* 'color' as it allows for hair specific values like *blond* 'blond'. Similarly, *Körpergestalt* 'shape of a human body' is separated from *Gestalt* 'shape', because it takes values like *hager* 'lean' or *mager* 'skinny' that cannot be used for non-human bodies. The main drawback of this data set is its restriction on non-derived adjectives, which are relatively rare in German. As a result, the data set is fairly small, and it contains several outdated and rarely used adjectives.

3.2 Algorithm

The proposed clustering is run in two configurations. The configuration **bin** operates with Jaccard distances calculated from the binarized vector space matrix (VSM) only. The configuration **lda** reweights these Jaccard distances with the output of an LDA topic model.

Both configurations start by constructing a vector space matrix V in which each A is described by the frequencies of the nouns with which it occurs in the corpus, such that rows represent adjectives, and columns represent nouns. This matrix is a structured distributional semantic model in the sense of Baroni and Lenci (2010), because the context words are defined by the syntactic relation between A and N. Sparsely populated rows (A) and columns (N) are removed by an iterative thinning step. After desparsification, a binary matrix V_B is derived from V by setting all non-zero values in V to 1.[5]

In the following, we describe the first iteration step in which the rows correspond to nouns and the columns to adjectives. Note, that after each iteration the matrix V_B is transposed, such that rows correspond to adjectives in even iterations and to nouns in odd iterations. Let $\vec{r_i}$ denote a logical row vector of V_B.

In the configuration **bin**, pairwise distance measures d_{ij} between all rows in V_B are calculated based on the Jaccard distances of the rows.

$$d_{ij} = 1 - \frac{|\vec{r_i} \wedge \vec{r_j}|}{|\vec{r_i} \vee \vec{r_j}|}$$

In order to avoid the clustering of nouns that share only a few adjectives, we set the distance measure to zero if the number of shared nouns is less than a given parameter t_J:

$$d_{ij}^{\text{bin}} = \begin{cases} d_{ij} & \text{if } |\vec{r_i} \wedge \vec{r_j}| \geq t_J \\ 0 & \text{else} \end{cases}$$

In the current configuration the parameter t_J is set to 3 to avoid the clustering of nouns sharing only two or less adjectives and vice versa.

In the configuration **lda**, LDA[6] is applied to the V_B, and another list of pairwise distance tuples

[4] These adjectives were collected and annotated by Sebastian Löbner and Thomas Gamerschlag in the project B02 "Dimensional Verbs", SFB 991, HHU Düsseldorf. Our special thanks to Thomas Gamerschlag for providing the data.

[5] The binarization step is performed, because we are only interested in whether a noun can be modified by an adjective, thus whether it bears an attribute the value of which can be restricted by the adjective. We could not rely on the frequency counts, as our corpus of A+N pairs results from an unbalanced merge of different corpora.

[6] LDA is performed with Gibbs Sampling using the library `GibbsLDA++`, `https://sourceforge.net/projects/gibbslda/`.

$(\vec{r_i}, \vec{r_j}, \theta_{ij})$ is created from the Θ values obtained from the LDA. For row vectors $\vec{r_i}$ and $\vec{r_j}$, θ_{ij} is given as

$$\theta_{ij} = \left(\sum_{k=1}^{K=15} (\Theta_{ik} - \Theta_{jk})^2 \right)^{\frac{1}{2}}$$

The number of latent topics $K = 15$ is intentionally kept low, because we don't derive semantic classes from the topic model of the LDA, but rather use the similarities between the Θ distributions for reweighting the Jaccard distances. In configuration **lda**, the Jaccard distances d_{ij} are reweighted with the distances from the LDA topic model, resulting in the final score d_{ij}^{lda} for each pair of words:

$$d_{ij}^{\text{lda}} = d_{ij}^{\text{bin}} \cdot \theta_{ij}$$

When the pairs of candidates $(\vec{r_i}, \vec{r_j})$ are reordered using this score, the top scoring pairs have high similarities in the binary vector space and in the topic model induced by the LDA.

Depending on the configuration mode, new word clusters are created either from the top 5% of the top scoring pairs with respect to the d^{bin} or to the d^{lda} distance measure. Clusters are built by constructing the transitive closures. Thus, if d_{ij} and d_{ik} belong to the 5% lowest distances measured, the words belonging to the rows $\vec{r_i}$, $\vec{r_j}$ and $\vec{r_k}$ are clustered together. Let R denote all rows that belong to one transitive closure and should thus be clustered together. The distributional representation of the new cluster, i.e. of the new row vector $\vec{R}$, is built by merging R with a majority based binary operator:

$$\vec{R_k} = \begin{cases} 1 & \text{if } \sum_{\vec{r} \in R} r_k \geq \frac{|R|}{2} \\ 0 & \text{else} \end{cases}$$

The rows in V which belong to R are deleted and the new row $\vec{R}$ is added, shrinking the matrix dimension in this way by $|R| - 1$.

The algorithm terminates, when no new classes are detected, because no pair of rows has more than t_J non-zero columns in common. Else, V_B is transposed, and the clustering method is applied to the other word class.

4 Evaluation and Results

We will restrict the evaluation of the obtained clusters to the adjective clusters, as we do not have access to an attribute based clustering of nouns that could be used as our gold data. We evaluate the results of the bidirectional clustering against the gold data described in Section 3.1, and against the results of k-means as a baseline clustering algorithm. As the adjective noun co-occurrence matrix is too sparse to run k-means successfully on it, we use information from context windows of the adjectives instead. For running the baseline k-means, we create neural embeddings of all words that occur at least 20 times in the merged corpus (newspaper, Wikipedia) using the `word2vec` tool (Mikolov et al., 2011),[7] and extract the embeddings of the adjectives that are processed in the bidirectional clustering. Remember that one of the main challenges in this task consists in determining the number of semantic classes from the raw data. In order to find this number, we calculate the gap statistic (Tibshirani et al., 2001) for cluster sizes between 40 (and thus just below the number of semantic classes in the gold data) and an – arbitrarily chosen – upper limit of 300, advancing in steps of 10 classes. Figure 1 shows that $\hat{k}$ as defined in Tibshirani et al. (2001, 415) changes from negative to positive values, when k-means is performed with approximately 160 classes, so that we use $k = 160$ for the baseline k-means clustering.

Since k-means and the gold standard provide hard non-hierarchical clusterings, we need to extract a hard non-hierarchical clustering from our bidirectional clustering as well, in order to compare them by standard measures as the Rand index (RI; Rand, 1971). The bidirectional clustering can produce deeply nested hierarchical clusterings, when later iterations of the method merge detected clusters and adjectives

[7] We cannot use the Google corpus for `word2vec` as it does not provide us with the necessary context of the A+N pairs. The settings for the `word2vec` tools are as follows: window size: 6 words, embedding size: 300, bow. All other parameters are set to their default values.

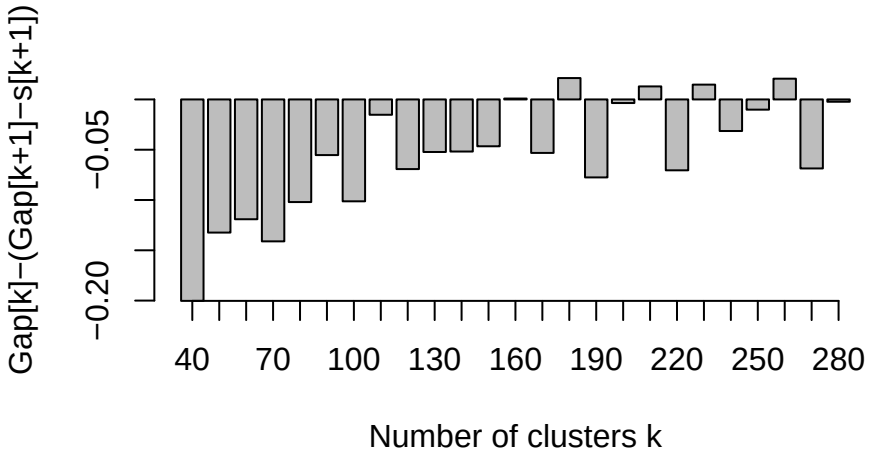


Figure 1: Gap statistic ($\hat{k}$) for neural embeddings of adjectives, showing a switch from negative to positive values at about 160 classes.

268 besonderer « 'topmost'

216 großartig außergewöhnlich

211 hervorragend

196 herausragend ausgezeichnet « 'leaf'

213 wunderschön wunderbar

Figure 2: Example demonstrating the effect of the two configurations 'leaf' and 'topmost'. Indentation indicates at which iteration a word cluster has been created. The configuration 'jaccard' cannot be displayed as its cannot be computed locally

into new classes. As such hierarchical representations provide no unambiguous hard clusters, we present three different modes of deriving hard clusterings from our representation. The following rules apply only to those adjectives that belong to at least one singleton cluster: In the configuration 'leaf' each such adjective is assigned to the smallest non-singleton cluster it belongs to; in the configuration 'topmost' it is assigned to the largest cluster it belongs to. Figure 2 illustrates the difference between these two configurations. The adjective *ausgezeichnet* 'excellent' is labeled with ID 196 in the 'leaf' and with ID 268 in the 'topmost' configuration. In addition, we derive a third hard clustering 'jaccard' that takes the homogeneity of the clusters into account. For this sake, we obtain pairwise Jaccard indexes between all adjectives that are subsumed under each node of the hierarchical output, and calculate their means for each node. To make this evaluation comparable with the result of k-means, the 160 nodes with the highest average Jaccard indexes are taken as hard cluster labels. All leaves that are not subsumed under one of these nodes are assigned to singleton clusters.

For the evaluation of the clustering we compute the Rand index (RI; Rand, 1971) and the Adjusted Rand index (Hubert and Arabie, 1985, ARI;). The Rand index of two clusterings C and C' is defined by

$$\mathcal{R}(C, C') = \frac{n_{11} + n_{00}}{\binom{n}{2}}$$

where n_{11} is the number of adjective pairs which belong to the same cluster in both clusterings, n_{00} is the number of adjective pairs that are assigned to different clusters in both clusterings, and n is the number of adjectives. Thus RI measures the proportion of the pairs which are clustered in the same

	RI	ARI
k-means	0.9499	0.0000

Table 2: (Adjusted) Rand index for the k-means clustering

configuration	'topmost'		'jaccard'		'leaf'	
	RI	ARI	RI	ARI	RI	ARI
newspaper (**bin**)	**0.9561**	0.0557	0.9568	0.0265	0.9583	0.0250
newspaper (**lda**)	0.9532	**0.0880**	0.9589	0.0263	0.9602	0.0269
google (**bin**)	0.9486	0.0687	0.9599	**0.0514**	0.9597	**0.0438**
google (**lda**)	0.9513	0.0759	**0.9607**	0.0442	**0.9607**	0.0370
merged (**bin**)	0.9382	0.0467	0.9566	0.0456	0.9571	**0.0454**
merged (**lda**)	**0.9534**	**0.0931**	**0.9576**	**0.0549**	**0.9581**	0.0386

Table 3: (Adjusted) Rand index for the bidirectional clustering

way in both clusterings. The adjusted Rand index corrects the Rand index by agreements that are solely due to chance. Note that while RI takes values between 0 and 1, ARI can take negative values as well (Meila, 2007). Table 2 shows the RI and ARI values for the k-means clusterings. Table 3 compares the clusterings resulting from the bidirectional clustering in different configurations and for different corpora.

While the RI values are high, ARI values are very low. This outcome is typical for sparse data and a large number of clusters. Remember that we have put each adjective that was not clustered by the algorithm into a singleton cluster. No clear favorite emerges when comparing the clusterings with and without LDA (Table 3, **bin** and **lda**). Only for the merged corpus which contains the data of all three corpora (newspaper, Google and Wikipedia), there is a tendency that **lda** outperforms **bin**. The different granularity levels ('topmost', 'jaccard' and 'leaf') by which the hierarchy is cut into a non-hierarchical clustering influence the RI and the ARI values as expected: a coarser clustering leads to higher ARI but slightly lower RI values. The most obvious result is that the bidirectional clustering outperforms the k-means clustering with respect to the more relevant ARI (independent of the chosen configuration). This result is remarkable, as the k-means clustering is based on textual context windows, while the bidirectional clustering only considers isolated A+N combinations. Thus, while for many tasks which aim at a semantic clustering it is better to look at the distribution of words in a larger context, the task of clustering adjectives by attributes seems to benefit from using a structured VSM.

To conclude this section, we will discuss some of the received clusters to give an impression of what kind of clusters can be expected. Many property-denoting adjectives turn out to be clustered very well, especially those which are derived from a numeral like *X-stellig* 'X place', *X-geschossig / X-stöckig* 'X floor', *X-malig* 'X times', *X-spurig* 'X lane', *X-jährig* 'X year', *X-tägig* 'X day' .... Although these clusters could have been easily identified by using morphological features, they are good candidates for a first proof of concept of our approach. Furthermore, we have gained satisfactory clusters of relational adjectives derived from concepts such as countries, languages, religions, cities, or territories.

A final example will be discussed in order to show the strengths and weaknesses of the approach. We get a cluster consisting of twenty adjectives describing properties of curves of some measure (temperature, price, ...), namely

5911 [*gleichbleibend* 'stable', *nachlassend* 'decreasing', *verringert* 'reduced', *gesunken* 'fallen']

5297 [*abnehmend* 'decreasing']

3163 [*vermindert* 'reduced', *vermehrt* 'increased']

2662 [*gesteigert* 'increased', *erhöht* 'raised']

1939 [*steigend* 'climbing']

1168 [*zunehmend* 'increasing / more and more', *wachsend* 'growing']

4541 [*höchstmöglich* 'highest possible']

3980 [*maximal* 'maximal', *größtmöglich* 'biggest possible']

5347 [*rückläufig* 'declining']

4687 [*sinkend* 'sinking', *gestiegen* 'climbed']

3331 [*stagnierend* 'stagnating', *schrumpfend* 'shrinking']

Most of the adjectives in this cluster denote some form of 'increasing', 'decreasing' or 'staying stable', thus a dynamic progression along a curve, and can be interpreted as values of the attribute 'progression' applied to curves. Others like *vermindert* 'decreased' or *vermehrt* 'increased' describe the general height of the curve. Only in subcluster 4541 one finds adjectives denoting extreme points of a curve like 'maximum' or 'highest possible'. Although the whole cluster 5911 is quite satisfactory, it could be improved in two respects. First, it obviously lacks many antinomies like *minimal* 'minimal' (to *maximal* 'maximal') or *fallend* 'decreasing' (to *steigend* 'increasing') which have been clustered elsewhere. Other adjectives describing the curve progression like *schwankend* 'floating' are missing as well. Second, from our frame-based perspective it would be desirable to receive clusters that reflect the different attributes by which a curve can be described like 'progression' and 'height' and which strictly separates adjectives that specify curves from others that specify points on a curve.

5 Conclusion

We have presented an iterative bidirectional clustering of adjectives and nouns by co-occurrences. The aim has been to derive adjective clusters which correspond to the value spaces of attributes. It has turned out that only some of the received clusters are perfect in that respect. Most miss some adjectives or mix adjectives belonging to different by familiar attributes. However, it has turned out that the clusters are a useful starting point for a manual analysis of the attribute value space.

In the quantitative evaluation the presented iterative bidirectional clustering outperformed the k-means clustering on word vectors. That indicates that for our task, the approach of only looking at individual adjective noun pairs instead of adjectives in bigger contexts is promising. By iteratively clustering adjectives on the basis of co-occurring nouns and vice versa, the hidden attributes connecting both can be crystallized out. However, bigger gold clusterings and more reliable evaluation measures are still missing.

The next steps to be taken are the following: The algorithm should be adapted to allow for overlapping clusters in order to account for polysemy. Better evaluation measures and more gold clusters which are specific to the task are needed. Ideally one would develop a frame-specific evaluation measure. One way could be to automatically induce frames from the derived adjective and noun clusters and to evaluate the resulting frames.

References

M. Baroni and R. Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*. Boston, pages 1183–1193.

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

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 546–556.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *EMNLP-CoNLL*. pages 1455–1465.

Gemma Boleda, Toni Badia, and Eloi Batlle. 2004. Acquisition of semantic classes for adjectives from distributional evidence. In *Proceedings of the 20th international conference on Computational Linguistics*. page 1119.

Gemma Boleda, Marco Baroni, Louise McNally, and Nghia Pham. 2013. Intensionality was only alleged: On adjective-noun composition in distributional semantics. In *Proceedings of the 10th International Workshop on Computational Semantics*. pages 35–46.

Jos de Bruin and Remko Scha. 1988. The interpretation of relational nouns. In *Proceedings of the 26th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '88, pages 25–32.

Duden. 2004. *Duden - Die deutsche Rechtschreibung*. Bibliographisches Institut & F. A. Brockhaus AG Mannheim, 23rd edition. Electronic version.

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

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of the 2010 Workshop on Geometrical Models of Natural Language Semantics*. pages 33–37.

Birgit Hamp and Helmut Feldweg. 1997. Germanet - a lexical-semantic net for german. In *Proceedings of the ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*.

M. Hartung and A. Frank. 2010. A structured Vector Space Model for hidden attribute meaning in adjective-noun phrases. In *Proceedings of the 23rd International Conference on Computational Linguistics*. pages 430–438.

Matthias Hartung and Anette Frank. 2011. Exploring supervised LDA models for assigning attributes to adjective-noun phrases. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. pages 540–551.

Verena Henrich and Erhard Hinrichs. 2010. GernEdiT - the GermaNet editing tool. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. pages 2228–2235.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification* 2(1):193–218.

Sebastian Löbner. 2011. Concept types and determination. *Journal of Semantics* 28(3):279–333.

Marina Meila. 2007. Comparing clusterings – an information based distance. *Journal of Multivariate Analysis* 98:873–895.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, , Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science* 331(6014):176–182.

Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. 2011. Strategies for training large scale neural network language models. In *2011 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. pages 196–201.

Marcin Morzycki. 2015. *Modification*. Cambridge University Press.

Eitan Adam Pechenick, Christopher M. Danforth, and Peter Sheridan Dodds. 2015. Characterizing the Google Books Corpus: Strong limits to inferences of socio-cultural and linguistic evolution. *PLoS ONE* 10(10).

James Pustejovsky. 1995. *The Generative Lexicon*. MIT Press, Cambridge, MA.

W.M. Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association* 66:846–850.

Peter F Schulam and Christiane Fellbaum. 2010. Automatically determining the semantic gradation of German adjectives. In *KONVENS*. pages 163–167.

Diarmuid Séaghdha and Anna Korhonen. 2014. Probabilistic distributional semantics with latent variable models. *Compututaional Linguistics* 40(3):587–631.

R. Tibshirani, G. Walther, and T. Hastie. 2001. Estimating the number of data clusters via the Gap statistic. *Journal of the Royal Statistical Society B* 63:411–423.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research* 37(1):141–188.

E.M. Vecchi, M. Marelli, R. Zamparelli, and M. Baroni. in press. Spicy adjectives and nominal donkeys: Capturing semantic deviance using compositionality in distributional spaces. *Cognitive Science.* .

Distributional Inclusion Hypothesis for Tensor-based Composition

Dimitri Kartsaklis and **Mehrnoosh Sadrzadeh**
Queen Mary University of London
School of Electronic Engineering and Computer Science
Mile End Road, London, UK
{d.kartsaklis;mehrnoosh.sadrzadeh}@qmul.ac.uk

Abstract

According to the distributional inclusion hypothesis, entailment between words can be measured via the feature inclusions of their distributional vectors. In recent work, we showed how this hypothesis can be extended from words to phrases and sentences in the setting of compositional distributional semantics. This paper focuses on inclusion properties of tensors; its main contribution is a theoretical and experimental analysis of how feature inclusion works in different concrete models of verb tensors. We present results for relational, Frobenius, projective, and holistic methods and compare them to the simple vector addition, multiplication, min, and max models. The degrees of entailment thus obtained are evaluated via a variety of existing word-based measures, such as Weed's and Clarke's, KL-divergence, APinc, balAPinc, and two of our previously proposed metrics at the phrase/sentence level. We perform experiments on three entailment datasets, investigating which version of tensor-based composition achieves the highest performance when combined with the sentence-level measures.

1 Introduction

Distributional hypothesis asserts that words that often occur in the same contexts have similar meanings (Firth, 1957). Naturally these models are used extensively to measure the semantic similarity of words (Turney and Pantel, 2010). Similarity is an a-directional relationship and computational linguists are also interested in measuring degrees of directional relationships between words. Distributional inclusion hypothesis is exactly about such relationships, and particularly, about entailment (Dagan et al., 1999; Geffet and Dagan, 2005; Herbelot and Ganesalingam, 2013). It states that a word u entails a word v if in any context that word u is used so can be word v. For example, in a corpus of sentences 'a boy runs', 'a person runs', 'a person sleeps', according to this hypothesis, *boy* $\vdash$ *person*, since wherever 'boy' is used, so is 'person'. Formally, we have that u entails v if features of u are included in features of v, where features are non-zero contexts. In this example, *boy* $\vdash$ *person*, since $\{\text{run}\} \subset \{\text{run, sleep}\}$.

For the same reasons that the distributional hypothesis is not directly applicable to phrases and sentences, the distributional inclusion hypothesis does not scale up from words to larger language constituents. In a nutshell, this is because the majority of phrases and sentences of language do not frequently occur in corpora of text, thus reliable statistics cannot be collected for them. In distributional models, this problem is addressed with the provision of composition operators, the purpose of which is to combine the statistics of words and obtain statistics for phrases and sentences. In recent work, we have applied the same compositionality principles to lift the entailment relation from word level to phrase/sentence level (Kartsaklis and Sadrzadeh, 2016; Balkır et al., 2016a; Balkır et al., 2016b; Balkır, 2014). The work in (Balkır, 2014; Balkır et al., 2016b) was focused on the use of entropic measures on density matrices and compositional operators thereof, but no experimental results were considered; similarly, Bankova et al. (2016) use a specific form of density matrices to represent words for entailment purposes, focusing only on theory. In (Balkır et al., 2016a), we showed how entropic and other measures can be used on vectors as well as on density matrices and supported this claim with experimental results. In (Kartsaklis and Sadrzadeh, 2016), we focused on making the distributional inclusion hypothesis compositional, worked

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2849–2860, Osaka, Japan, December 11-17 2016.

out how feature inclusion lifts from words to compositional operators on them, and based on experimental results showed that intersective composition operators result in more reliable entailments. This paper takes a more concrete perspective and focuses on the feature inclusion properties of tensors constructed in different ways and composition operators applied to the tensors and vectors.

One can broadly classify the compositional distributional models to three categories: ones that are based on simple element-wise operations between vectors, such as addition and multiplication (Mitchell and Lapata, 2010); tensor-based models in which relational words such as verbs and adjectives are tensors and matrices contracting and multiplying with noun (and noun-phrase) vectors (Coecke et al., 2010; Grefenstette and Sadrzadeh, 2011; Baroni and Zamparelli, 2010); and models in which the compositional operator is part of a neural network (Socher et al., 2012; Kalchbrenner et al., 2014) and is usually optimized against a specific objective.

Tensor-based models stand in between the two extremes of element-wise vector mixing and neural net-based methods, offering a sufficiently powerful alternative that allows for theoretical reasoning at a level deeper than it is usually possible with black-box statistical approaches. Models of this form have been used in the past with success in a number of NLP tasks, such as head-verb disambiguation (Grefenstette and Sadrzadeh, 2011), term-definition classification (Kartsaklis et al., 2012), and generic sentence similarity (Kartsaklis and Sadrzadeh, 2014). In this paper we extend this work to entailment, by investigating (theoretically and experimentally) the properties of feature inclusion on the phrase and sentence vectors produced in four different tensor-based compositional distributional models: the relational models of (Grefenstette and Sadrzadeh, 2011), the Frobenius models of (Kartsaklis et al., 2012; Milajevs et al., 2014), the projective models of (Kartsaklis and Sadrzadeh, 2016), and the holistic linear regression model of (Baroni and Zamparelli, 2010).

Contrary to formal semantic models, and customary to distributional models, our entailments are non-boolean and come equipped with degrees. We review a number of measures that have been developed for evaluating degrees of entailment at the lexical level, such as the *APinc* measure and its newer version, *balAPinc* of Kotlerman et al. (2010), which is considered as state-of-the-art for word-level entailment. A newly proposed adaptation of these metrics, recently introduced by the authors in (Kartsaklis and Sadrzadeh, 2016), is also detailed. This measure takes into account the specificities introduced by the use of a compositional operator and lifts the measures from words to phrase/sentence level.

We experiment with these models and evaluate them on entailment relations between simple intransitive sentences, verb phrases, and transitive sentences on the datasets of (Kartsaklis and Sadrzadeh, 2016). Our findings suggest that the Frobenius models provide the highest performance, especially when combined with our sentence-level measures. On a more general note, the experimental results of this paper support that of previous work (Kartsaklis and Sadrzadeh, 2016) and strongly indicate that compositional models employing some form of intersective feature selection, i.e. point-wise vector multiplication or tensor-based models with an element of element-wise mixing (such as the Frobenius constructions), are more appropriate for entailment tasks in distributional settings.

2 Compositional distributional semantics

The purpose of a compositional distributional model is to produce a vector representing the meaning of a phrase or a sentence by combining the vectors of its words. In the simplest case, this is done by element-wise operations on the vectors of the words (Mitchell and Lapata, 2010). Specifically, the vector representation $\overrightarrow{w}$ of a sequence of words $w_1, \ldots, w_n$ is defined to be:

$$\overrightarrow{w} := \overrightarrow{w_1} + \overrightarrow{w_2} + \cdots + \overrightarrow{w_n} \qquad\qquad \overrightarrow{w} := \overrightarrow{w_1} \odot \overrightarrow{w_2} \odot \cdots \odot \overrightarrow{w_n}$$

In a more linguistically motivated approach, relational words such as verbs and adjectives are treated as linear or multi-linear maps. These are then applied to the vectors of their arguments by following the rules of the grammar (Coecke et al., 2010; Grefenstette and Sadrzadeh, 2011; Baroni and Zamparelli, 2010). For example, an adjective is a map $N \to N$, for N a basic noun space of the model. Equivalently, this map can be represented as a matrix living in the space $N \otimes N$. In a similar way, a transitive verb is a map $N \times N \to S$, or equivalently, a "cube" or a tensor of order 3 in the space $N \otimes N \otimes S$, for S a basic sentence space of the model. Composition takes place by tensor contraction, which is a generalization of

matrix multiplication to higher order tensors. For the case of an adjective-noun compound, this simplifies to matrix multiplication between the adjective matrix and the vector of its noun, while for a transitive sentence it takes the following form, where $\overline{\text{verb}}$ is a tensor of order 3 and $\times$ is tensor contraction:

$$\overrightarrow{svo} = (\overline{\text{verb}} \times \overrightarrow{\text{obj}}) \times \overrightarrow{\text{subj}}$$

Finally, phrase and sentence vectors have been also produced by the application of neural architectures, such as recursive or recurrent neural networks (Socher et al., 2012; Cheng and Kartsaklis, 2015) and convolutional neural networks (Kalchbrenner et al., 2014). These models have been shown to perform well on large scale entailment tasks such as the ones introduced by the RTE challenge.

3 Distributional inclusion hypothesis

The distributional inclusion hypothesis (DIH) (Dagan et al., 1999; Geffet and Dagan, 2005; Herbelot and Ganesalingam, 2013) is based on the fact that whenever a word u entails a word v, then it makes sense to replace instances of u with v. For example, 'cat' entails 'animal', hence in the sentence 'a cat is asleep', it makes sense to replace 'cat' with 'animal' and obtain 'an animal is asleep'. On the other hand, 'cat' does not entail 'butterfly', and indeed it does not make sense to do a similar substitution and obtain the sentence 'a butterfly is asleep'. This hypothesis has limitations, the main one being that it only makes sense in upward monotone contexts. For instance, the substitution of u for v would not work for sentences that have negations or quantifiers such as 'all' and 'none'. As a result, one cannot replace 'cat' with 'animal' in sentences such as 'all cats are asleep' or 'a cat is not asleep'. Despite this and other limitations, the DIH has been subject to a good amount of study in the distributional semantics community and its predictions have been validated (Geffet and Dagan, 2005; Kotlerman et al., 2010).

Formally, the DIH says that if word u entails word v, then the set of features of u are included in the set of features of v. In the context of a distributional model of meaning, the term *feature* refers to a non-zero dimension of the distributional vector of a word. By denoting the features of a distributional vector $\overrightarrow{v}$ by $\mathcal{F}(\overrightarrow{v})$, one can symbolically express the DIH as follows:

$$u \vdash v \quad \text{whenever} \quad \mathcal{F}(\overrightarrow{u}) \subseteq \mathcal{F}(\overrightarrow{v}) \tag{1}$$

The research on the DIH can be categorised into two classes. In the first class, the degree of entailment between two words is based on the distance between the vector representations of the words. This distance must be measured by asymmetric means, since entailment is directional. Examples of measures used here are entropy-based measures such as KL-divergence (Chen and Goodman, 1996). KL-divergence is only defined when the support of $\overrightarrow{v}$ is included in the support of $\overrightarrow{u}$. To overcome this restriction, a variant referred to by α-skew (Lee, 1999) has been proposed (for $\alpha \in (0, 1]$ a smoothing parameter). *Representativeness* provides another way of smoothing the KL-divergence. The formulae for these are as follows, where abusing the notation we take $\overrightarrow{u}$ and $\overrightarrow{v}$ to also denote the probability distributions of u and v:

$$D_{\text{KL}}(\overrightarrow{v} \| \overrightarrow{u}) = \sum_i v_i(\ln v_i - \ln u_i) \qquad s_\alpha(\overrightarrow{u}, \overrightarrow{v}) = D_{\text{KL}}(\overrightarrow{v} \| \alpha\overrightarrow{u} + (1-\alpha)\overrightarrow{v})$$

$$R_{\text{D}}(\overrightarrow{v} \| \overrightarrow{u}) = \frac{1}{1 + D_{\text{KL}}(\overrightarrow{v} || \overrightarrow{u})}$$

Representativeness turns KL-divergence into a number in the unit interval $[0, 1]$. As a result we obtain $0 \leq R_{\text{D}}(\overrightarrow{v} \| \overrightarrow{u}) \leq 1$, with $R_{\text{D}}(\overrightarrow{v} \| \overrightarrow{u}) = 0$ when the support of $\overrightarrow{v}$ is not included in the support of $\overrightarrow{u}$ and $R_{\text{D}}(\overrightarrow{v} \| \overrightarrow{u}) = 1$, when $\overrightarrow{u}$ and $\overrightarrow{v}$ represent the same distribution.

The research done in the second class attempts a more direct measurement of the inclusion of features, with the simplest possible case returning a binary value for inclusion or lack thereof. Measures developed by Weeds et al. (2004) and Clarke (2009) advance this simple methods by arguing that not all features play an equal role in representing words and hence they should not be treated equally when it comes to measuring entailment. Some features are more "pertinent" than others and these features have to be given a higher weight when computing inclusion. For example, 'cat' can have a non-zero coordinate on

all of the features 'mammal, miaow, eat, drink, sleep'. But the amount of these coordinates differ, and one can say that, for example, the higher the coordinate the more pertinent the feature. Pertinence is computed by various different measures, the most recent of which is *balAPinc* (Kotlerman et al., 2010), where *LIN* is Lin's similarity (Lin, 1998) and *APinc* is an asymmetric measure:

$$\mathit{balAPinc}(u,v) = \sqrt{\mathit{LIN}(u,v) \cdot \mathit{APinc}(u,v)} \qquad \mathit{APinc}(u,v) = \frac{\sum_r [P(r) \cdot \mathit{rel}'(f_r)]}{|\mathcal{F}(\overrightarrow{u})|}$$

APinc applies the DIH via the idea that features with high values in $\mathcal{F}(\overrightarrow{u})$ must also have high values in $\mathcal{F}(\overrightarrow{v})$. In the above formula, f_r is the feature in $\mathcal{F}(\overrightarrow{u})$ with rank r; $P(r)$ is the precision at rank r; and $\mathit{rel}'(f_r)$ is a weight computed as follows:

$$rel'(f) = \begin{cases} 1 - \frac{\mathit{rank}(f, \mathcal{F}(\overrightarrow{v}))}{|\mathcal{F}(\overrightarrow{v})|+1} & f \in \mathcal{F}(\overrightarrow{v}) \\ 0 & o.w. \end{cases} \tag{2}$$

where $\mathit{rank}(f, \mathcal{F}(\overrightarrow{v}))$ shows the rank of feature f within the entailed vector. In general, *APinc* can be seen as a version of average precision that reflects lexical inclusion.

4 Measuring feature inclusion at the phrase/sentence level

In recent work, the authors of this paper introduced a variation of the *APinc* and *balAPinc* measures aiming to address the extra complications imposed when evaluating entailment at the phrase/sentence level (Kartsaklis and Sadrzadeh, 2016). The modified measures differ from the original ones in two aspects. Firstly, in a compositional distributional model, the practice of considering only non-zero elements of the vectors as features becomes too restrictive and thus suboptimal for evaluating entailment; indeed, depending on the form of the vector space and the applied compositional operator (especially in intersective models, see Sections 5 and 6), an element can get very low values without however ever reaching zero. The new measures exploit this blurring of the notion of "featureness" to the limit, by letting $\mathcal{F}(\overrightarrow{w})$ to include all the dimensions of $\overrightarrow{w}$.

Secondly, the continuous nature of distributional models is further exploited by providing a stronger realization of the idea that $u \vdash v$ whenever v occurs in all the contexts of u. Let $f_r^{(u)}$ be a feature in $\mathcal{F}(\overrightarrow{u})$ with rank r and $f_r^{(v)}$ the corresponding feature in $\mathcal{F}(\overrightarrow{v})$, we remind that Kotlerman et al. consider that feature inclusion holds at rank r whenever $f_r^{(u)} > 0$ and $f_r^{(v)} > 0$; the new measures strengthen this assumption by requiring that $f_r^{(u)} \leq f_r^{(v)}$. Incorporating these modifications in the *APinc* measure, $P(r)$ and $rel'(f_r)$ are redefined as follows:

$$P(r) = \frac{\left|\{f_r^{(u)} | f_r^{(u)} \leq f_r^{(v)}, 0 < r \leq |\overrightarrow{u}|\}\right|}{r} \qquad rel'(f_r) = \begin{cases} 1 & f_r^{(u)} \leq f_r^{(v)} \\ 0 & o.w. \end{cases} \tag{3}$$

The new relevance function subsumes the old one, as by definition high-valued features in $\mathcal{F}(\overrightarrow{u})$ must be even higher in $\mathcal{F}(\overrightarrow{v})$. The new *APinc* at the phrase/sentence level thus becomes as follows:

$$\mathit{SAPinc}(u,v) = \frac{\sum_r [P(r) \cdot \mathit{rel}'(f_r)]}{|\overrightarrow{u}|} \tag{4}$$

where $|\overrightarrow{u}|$ is the number of dimensions of $\overrightarrow{u}$. Further, we notice that *balAPinc* is the geometric average of *APinc* with Lin's similarity measure, which is symmetric. According to (Kotlerman et al., 2010), the rationale of including a symmetric measure was that *APinc* tends to return unjustifyingly high scores when the entailing word is infrequent, that is, when the feature vector of the entailing word is very short; the purpose of the symmetric measure was to penalize the result, since in this case the similarity of the narrower term with the broader one is usually low. However, now that all feature vectors have the same length, such a balancing action is unnecessary; more importantly, it introduces a strong element of symmetry in a measure that is intended to be strongly asymmetric. We cope with this issue by replacing Lin's measure with representativeness on KL-divergence,[1] obtaining the following new version of *balAPinc*:

[1]Using other asymmetric measures is also possible; the choice of representativeness on KL-divergence was based on informal experimentation which showed that this combination works better than other options in practice.

$$\textit{SBalAPinc}(u,v) = \sqrt{R_{\mathrm{D}}(\overrightarrow{u}\,\|\,\overrightarrow{v}) \cdot \textit{SAPinc}(\overrightarrow{u},\overrightarrow{v})} \tag{5}$$

Recall that $R_{\mathrm{D}}(p\|q)$ is asymmetric, measuring the extent to which q represents (i.e. is similar to) p. So the term $R_{\mathrm{D}}(\overrightarrow{u}\,\|\,\overrightarrow{v})$ in the above formula measures how well the *broader* term v represents the narrower one u; as an example, we can think that the term 'animal' is representative of 'cat', while the reverse is not true. The new measure aims at: (i) retaining a strongly asymmetric nature; and (ii) providing a more fine-grained element of entailment evaluation.

5 Generic feature inclusion in compositional models

In the presence of a compositional operator, features of phrases or sentences adhere to set-theoretic properties. For simple additive and multiplicative models, the set of features of a phrase/sentence is derived from the set of features of their words using union and intersection. It is slightly less apparent (and for reasons of space we will not give details here) that the features of point-wise minimum and maximum of vectors are also derived from the intersection and union of their features, respectively. That is:

$$\mathcal{F}(\overrightarrow{v_1} + \cdots + \overrightarrow{v_n}) = \mathcal{F}(\max(\overrightarrow{v_1},\cdots,\overrightarrow{v_n})) = \mathcal{F}(\overrightarrow{v_1}) \cup \cdots \cup \mathcal{F}(\overrightarrow{v_n})$$
$$\mathcal{F}(\overrightarrow{v_1} \odot \cdots \odot \overrightarrow{v_n}) = \mathcal{F}(\min(\overrightarrow{v_1},\cdots,\overrightarrow{v_n})) = \mathcal{F}(\overrightarrow{v_1}) \cap \cdots \cap \mathcal{F}(\overrightarrow{v_n})$$

As shown in (Kartsaklis and Sadrzadeh, 2016), element-wise composition of this form lifts naturally from the word level to phrase/sentence level; specifically, for two sentences $s_1 = u_1 \dots u_n$ and $s_2 = v_1 \dots v_n$ for which $u_i \vdash v_i\ \forall i \in [1,n]$, it is always the case that $s_1 \vdash s_2$. This kind of lifting of the entailment relationship from words to the phrase/sentence level also holds for tensor-based models (Balkır et al., 2016a).

In general, feature inclusion is a more complicated process for tensor-based settings, since in this case the composition operation is matrix multiplication and tensor contraction. As an example, consider the simple case of a matrix multiplication between a $m \times n$ matrix $\mathbf{M}$ and a $n \times 1$ vector $\overrightarrow{v}$. Matrix $\mathbf{M}$ can be seen as a list of column vectors $(\overrightarrow{w_1},\overrightarrow{w_2},\cdots,\overrightarrow{w_n})$, where $\overrightarrow{w_i} = (w_{1i},\cdots,w_{mi})^{\mathsf{T}}$. The result of the matrix multiplication is a combination of scalar multiplications of each element v_i of the vector $\overrightarrow{v}$ with the corresponding column vector $\overrightarrow{w_i}$ of the matrix $\mathbf{M}$. That is, we have:

$$\begin{pmatrix} w_{11} & \cdots & w_{1n} \\ w_{21} & \cdots & w_{2n} \\ \vdots & & \vdots \\ w_{m1} & \cdots & w_{mn} \end{pmatrix} \times \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = v_1\overrightarrow{w_1} + v_2\overrightarrow{w_2} + \cdots + v_n\overrightarrow{w_n}$$

Looking at $\mathbf{M} \times \overrightarrow{v}$ in this way enables us to describe $\mathcal{F}(\mathbf{M} \times \overrightarrow{v})$ in terms of the union of $\mathcal{F}(\overrightarrow{w_i})$'s where v_i is non zero, that is, we have:

$$\mathcal{F}(\mathbf{M} \times \overrightarrow{v}) = \bigcup_{v_i \neq 0} \mathcal{F}(\overrightarrow{w_i})$$

By denoting v_i a feature whenever it is non-zero, we obtain an equivalent form as follows:

$$\bigcup_i \mathcal{F}(\overrightarrow{w_i})\ |_{\mathcal{F}(v_i)} \tag{6}$$

The above notation says that we collect features of each $\overrightarrow{w_i}$ vector but only up to "featureness" of v_i, that is up to v_i being non-zero. This can be extended to tensors of higher order; a tensor of order 3, for example, can be seen as a list of matrices, a tensor of order 4 as a list of "cubes" and so on. For the case of this paper, we will not go beyond matrix multiplication and cube contraction.

6 Feature inclusion in concrete constructions of tensor-based models

While the previous section provided a generic analysis of the feature inclusion behaviour of tensor-based models, the exact feature inclusion properties of these models depend on the specific concrete constructions, and in principle get a form more refined than that of simple intersective or union-based composition. In this section we investigate a number of tensor-based models with regard to feature inclusion, and derive their properties.

6.1 Relational model

As a starting point we will use the model of Grefenstette and Sadrzadeh (2011), which adopts an extensional approach and builds the tensor of a relational word from the vectors of its arguments. More specifically, the tensors for adjectives, intransitive verbs, and transitive verbs are defined as below, respectively:

$$\overline{adj} = \sum_i \overrightarrow{Noun_i} \qquad \overline{verb}_{\text{IN}} = \sum_i \overrightarrow{Sbj}_i \qquad \overline{verb}_{\text{TR}} = \sum_i \overrightarrow{Sbj}_i \otimes \overrightarrow{Obj}_i \tag{7}$$

where $\overrightarrow{Noun_i}$, $\overrightarrow{Sbj}_i$ and $\overrightarrow{Obj}_i$ refer to the distributional vectors of the nouns, subjects and objects that occurred as arguments for the adjective and the verb across the training corpus. For the case of a subject-verb sentence and a verb-object phrase, composition reduces to element-wise multiplication of the two vectors, and the features of the resulting sentence/phrase vector get the following form (with $\overrightarrow{s}$ and $\overrightarrow{o}$ to denote the vectors of the subject/verb of the phrase/sentence):

$$\mathcal{F}(\overrightarrow{sv}) = \bigcup_i \mathcal{F}(\overrightarrow{Sbj}_i) \cap \mathcal{F}(\overrightarrow{s}) \qquad \mathcal{F}(\overrightarrow{vo}) = \bigcup_i \mathcal{F}(\overrightarrow{Obj}_i) \cap \mathcal{F}(\overrightarrow{o}) \tag{8}$$

For a transitive sentence, the model of Grefenstette and Sadrzadeh (2011) returns a matrix, computed in the following way:

$$\overline{svo}_{\text{Rel}} = \overline{verb} \odot (\overrightarrow{s} \otimes \overrightarrow{o})$$

where $\overline{verb}$ is defined as in Equation 7. By noticing that $\mathcal{F}(\overrightarrow{u} \otimes \overrightarrow{v}) = \mathcal{F}(\overrightarrow{u}) \times \mathcal{F}(\overrightarrow{v})$, with symbol $\times$ to denote in this case the *cartesian product* of the two feature sets, we define the feature set of a transitive sentence as follows:

$$\mathcal{F}(\overline{svo}_{\text{Rel}}) = \bigcup_i \mathcal{F}(\overrightarrow{Sbj}_i) \times \mathcal{F}(\overrightarrow{Obj}_i) \cap \mathcal{F}(\overrightarrow{s}) \times \mathcal{F}(\overrightarrow{o}) \tag{9}$$

Equation 9 shows that the features of this model are pairs (f_{sbj}, f_{obj}), with f_{sbj} a subject-related feature and f_{obj} an object-related feature, providing a fine-grained representation of the sentence. Throughout this paper, we refer to this model as *relational*.

6.2 Frobenius models

As pointed out in (Kartsaklis et al., 2012), the disadvantage of the relational model is that their resulting representations of verbs have one dimension less than what their types dictate. According to the type assignments, an intransitive verb has to be a matrix and a transitive verb a cube, where as in the above we have a vector and a matrix. A solution presented in (Kartsaklis et al., 2012) suggested the use of *Frobenius* operators in order to expand vectors and matrices into higher order tensors by embedding them into the the diagonals. For example, a vector is embedded into a matrix by putting it in the diagonal of a matrix and padding the off-diagonal elements with zeros. Similarly, one can embed a matrix into a cube by putting it into the main diagonal and pad the rest with zeros. Using this method, for example, one could transform a simple intersective model in tensor form by embedding the context vector of a verb $\overrightarrow{v}$ first into a matrix and then into a cube. For a transitive sentence, one could use the matrix defined in Equation 7 and derive a vector for the meaning of the sentence in two ways, each one corresponding to a different embedding of the matrix into a tensor of order 3:

$$\overrightarrow{svo}_{\text{CpSbj}} = \overrightarrow{s} \odot (\overline{verb} \times \overrightarrow{o}) \qquad \overrightarrow{svo}_{\text{CpObj}} = (\overrightarrow{s}^{\mathsf{T}} \times \overline{verb}) \odot \overrightarrow{o} \tag{10}$$

We refer to these models as Copy-Subject and Copy-Object, correspondingly. In order to derive their feature inclusion properties, we first examine the form of the sentence vector produced when the verb is composed with a new subject/object pair:

$$\overrightarrow{svo}_{\text{CpSbj}} = \overrightarrow{s} \odot (\overline{verb} \times \overrightarrow{o}) = \overrightarrow{s} \odot \sum_i \overrightarrow{Sbj_i} \langle \overrightarrow{Obj_i} | \overrightarrow{o} \rangle$$
$$\overrightarrow{svo}_{\text{CpObj}} = (\overrightarrow{s}^{\mathsf{T}} \times \overline{verb}) \odot \overrightarrow{o} = \overrightarrow{o} \odot \sum_i \overrightarrow{Obj_i} \langle \overrightarrow{s} | \overrightarrow{Sbj_i} \rangle$$

We can now define the feature sets of the two models using notation similar to that of Equation 6:

$$\begin{aligned} \mathcal{F}(\overrightarrow{svo}_{\text{CpSbj}}) &= \mathcal{F}(\overrightarrow{s}) \cap \bigcup_i \mathcal{F}(\overrightarrow{Sbj_i}) \mid_{\mathcal{F}\left(\langle \overrightarrow{Obj_i} | \overrightarrow{o} \rangle\right)} \\ \mathcal{F}(\overrightarrow{svo}_{\text{CpObj}}) &= \mathcal{F}(\overrightarrow{o}) \cap \bigcup_i \mathcal{F}(\overrightarrow{Obj_i}) \mid_{\mathcal{F}\left(\langle \overrightarrow{s} | \overrightarrow{Sbj_i} \rangle\right)} \end{aligned} \tag{11}$$

The symbol | defines a restriction on feature inclusion based on how well the arguments of the sentence fit to the arguments of the verb. For a subject-object pair (Sbj, Obj) that has occured with the verb in the corpus, this translates to the following:

- *Copy-Subject:* Include the features of Sbj up to similarity of Obj with the sentence object
- *Copy-Object:* Include the features of Obj up to similarity of Sbj with the sentence subject

Note that each of the Frobenius models puts emphasis on a different argument of a sentence; the Copy-Subject model collects features of the subjects that occured with the verb, while the Copy-Object model collects features from the verb objects. It is reasonable then to further combine the two models in order to get a more complete representation of the sentence meaning, and hence its feature inclusion properties. Below we define the feature sets of two variations, where this combination is achieved via addition (we refer to this model as *Frobenius additive*) and element-wise multiplication (*Frobenius multiplicative*) of the vectors produced by the individual models (Kartsaklis and Sadrzadeh, 2014):

$$\begin{aligned} \mathcal{F}(\overrightarrow{svo}_{\text{FrAdd}}) &= \mathcal{F}(\overrightarrow{svo}_{\text{CpSbj}}) \cup \mathcal{F}(\overrightarrow{svo}_{\text{CpObj}}) \\ \mathcal{F}(\overrightarrow{svo}_{\text{FrMul}}) &= \mathcal{F}(\overrightarrow{svo}_{\text{CpSbj}}) \cap \mathcal{F}(\overrightarrow{svo}_{\text{CpObj}}) \end{aligned} \tag{12}$$

where $\mathcal{F}(\overrightarrow{svo}_{\text{CpSbj}})$ and $\mathcal{F}(\overrightarrow{svo}_{\text{CpObj}})$ are defined as in Equation 11.

6.3 Projective models

In this section we provide an alternative solution and remedy the problem of having lower dimensions than the required by arguing that the sentence/phrase space should be spanned by the vectors of the arguments of the verb across the corpus. Thus we create verb matrices for intransitive sentences and verb phrases by summing up *projectors* of the argument vectors, in the following way:

$$\overline{v}_{itv} := \sum_i \overrightarrow{Sbj_i} \otimes \overrightarrow{Sbj_i} \qquad \overline{v}_{vp} := \sum_i \overrightarrow{Obj_i} \otimes \overrightarrow{Obj_i} \tag{13}$$

When these verbs are composed with some subject/object to form a phrase/sentence, each vector in the spanning space is weighted by its similarity (assuming normalized vectors) with the vector of that subject/object, that is:

$$\overrightarrow{sv}_{\text{Prj}} = \overrightarrow{s}^{\mathsf{T}} \times \overline{v}_{itv} = \sum_i \langle \overrightarrow{Sbj_i} | \overrightarrow{s} \rangle \overrightarrow{Sbj_i} \qquad \overrightarrow{vo}_{\text{Prj}} = \overline{v}_{vp} \times \overrightarrow{o} = \sum_i \langle \overrightarrow{Obj_i} | \overrightarrow{o} \rangle \overrightarrow{Obj_i} \tag{14}$$

Translating the above equations to feature inclusion representations will give:

$$\mathcal{F}(\overrightarrow{s\bar{v}}_{\mathrm{Prj}}) = \bigcup_i \mathcal{F}(\overrightarrow{Sbj}_i) \mid_{\mathcal{F}\left(\langle \overrightarrow{Sbj}_i | \overrightarrow{s} \rangle\right)} \qquad \mathcal{F}(\overrightarrow{v\bar{o}}_{\mathrm{Prj}}) = \bigcup_i \mathcal{F}(\overrightarrow{Obj}_i) \mid_{\mathcal{F}\left(\langle \overrightarrow{Obj}_i | \overrightarrow{o} \rangle\right)} \tag{15}$$

with symbol | to define again a restriction on feature inclusion based on the similarity of the arguments with the subject or object of the sentence/phrase. For the subject-verb case, this reads: "include a subject that occured with the verb, up to its similarity with the subject of the sentence". For the case of a transitive verb (a function of two arguments), we define the sentence space to be spanned by the average of the argument vectors, obtaining:

$$\overline{v}_{trv} := \sum_i \overrightarrow{Sbj}_i \otimes \left(\frac{\overrightarrow{Sbj}_i + \overrightarrow{Obj}_i}{2} \right) \otimes \overrightarrow{Obj}_i$$

The meaning of a transitive sentence then is computed as:

$$\overrightarrow{s\bar{v}o}_{\mathrm{Prj}} = \overrightarrow{s}^{\mathsf{T}} \times \overline{v}_{trv} \times \overrightarrow{o} = \sum_i \left[\langle \overrightarrow{s} \mid \overrightarrow{Sbj}_i \rangle \langle \overrightarrow{Obj}_i \mid \overrightarrow{o} \rangle \left(\frac{\overrightarrow{Sbj}_i + \overrightarrow{Obj}_i}{2} \right) \right] \tag{16}$$

Feature-wise, the above translates to the following:

$$\mathcal{F}(\overrightarrow{s\bar{v}o}_{\mathrm{Prj}}) = \bigcup_i \left(\mathcal{F}(\overrightarrow{Sbj}_i) \cup \mathcal{F}(\overrightarrow{Obj}_i) \right) \mid_{\mathcal{F}\left(\langle \overrightarrow{s} | \overrightarrow{Sbj}_i \rangle\right) \mathcal{F}\left(\langle \overrightarrow{Obj}_i | \overrightarrow{o} \rangle\right)} \tag{17}$$

Note that in contrast with the relational and Frobenius models, which all employ an element of intersective feature selection, the projective models presented in this section are purely union-based.

6.4 Inclusion of verb vectors

The models of the previous sections provide a variety of options for representing the meaning of a verb from its arguments. However, none of these constructions takes into account the distributional vector of the verb itself, which includes valuable information that could further help in entailment tasks. We remedy this problem by embedding the missing information into the existing tensors; for example, we can amend the tensors of the projective model as follows:

$$\overline{v}_{itv} = \sum_i \overrightarrow{Sbj}_i \otimes \left(\overrightarrow{Sbj}_i \odot \overrightarrow{v} \right) \qquad \overline{v}_{vp} = \sum_i \left(\overrightarrow{Obj}_i \odot \overrightarrow{v} \right) \otimes \overrightarrow{Obj}_i \tag{18}$$

with $\overrightarrow{v}$ denoting the distributional vector of the verb. In the context of an intransitive sentence, now we have the following interaction:

$$\overrightarrow{s\bar{v}} = \overrightarrow{s}^{\mathsf{T}} \times \overline{v}_{itv} = \overrightarrow{s}^{\mathsf{T}} \times \sum_i \overrightarrow{Sbj}_i \otimes \left(\overrightarrow{Sbj}_i \odot \overrightarrow{v} \right) = \overrightarrow{v} \odot \sum_i \langle \overrightarrow{s} | \overrightarrow{Sbj}_i \rangle \overrightarrow{Sbj}_i \tag{19}$$

We see that the result of the standard projective model (Equation 14) is now enchanced with an additional step of interesective feature selection. In feature inclusion terms, we get:

$$\mathcal{F}(\overrightarrow{s\bar{v}}) = \mathcal{F}(\overrightarrow{v}) \cap \mathcal{F}(\overrightarrow{s\bar{v}}_{\mathrm{Prj}}) \qquad \mathcal{F}(\overrightarrow{v\bar{o}}) = \mathcal{F}(\overrightarrow{v}) \cap \mathcal{F}(\overrightarrow{v\bar{o}}_{\mathrm{Prj}}) \tag{20}$$

It is easy to show that similar formulae hold for the relational and Frobenius models.

7 Experimental setting

We evaluate the feature inclusion behaviour of the tensor-based models of Section 6 in three different tasks; specifically, we measure upward-monotone entailment between (a) intransitive sentences; (b) verb phrases; and (c) transitive sentences. We use the entailment datasets introduced in (Kartsaklis and Sadrzadeh, 2016), which consist of 135 subject-verb pairs, 218 verb-object pairs, and 70 subject-verb-object pairs, the phrases/sentences of which stand in a fairly clear entailment relationship. Each dataset has been created using hypernym-hyponym relationships from WordNet, and it was extended with the reverse direction of the entailments as negative examples, creating three strictly directional entailment

datasets of 270 (subject-verb), 436 (verb-object) and 140 (subject-verb-object) entries. Some examples of positive entailments from all three categories include:[2]

Subject-verb	Verb-object	Subject-verb-object
evidence suggests ⊢ information expresses	sign contract ⊢ write agreement	book presents account ⊢ work shows evidence
survey reveals ⊢ work shows	publish book ⊢ produce publication	woman marries man ⊢ female joins male
player plays ⊢ contestant compete	sing song ⊢ perform music	author retains house ⊢ person holds property
study demonstrates ⊢ examination shows	reduce number ⊢ decrease amount	study demonstrates importance ⊢ work shows value
summer finishes ⊢ season ends	promote development ⊢ support event	experiment tests hypothesis ⊢ research evaluates proposal

In all cases, we first apply a compositional model to the phrases/sentences of each pair in order to create vectors representing their meaning, and then we evaluate the entailment relation between the phrases/sentences by using these composite vectors as input to a number of entailment measures. The goal is to see which combination of compositional model/entailment measure is capable of better recognizing strictly directional entailment relationships between phrases and sentences.

We experimented with a variety of entailment measures, including *SAPinc* and *SBalAPinc* as in (Kartsaklis and Sadrzadeh, 2016), their word-level counterparts (Kotlerman et al., 2010), KL-divergence (applied to smoothed vectors as in Chen and Goodman (1996)), α-skew with $\alpha = 0.99$ as in Kotlerman et al. (2010), *WeedsPrec* as in Weeds et al. (2004), and *ClarkeDE* as in Clarke (2009). We use strict feature inclusion as a baseline; in this case, entailment holds only when $\mathcal{F}(\overrightarrow{phrase_1}) \subseteq \mathcal{F}(\overrightarrow{phrase_2})$. For compositional operators, we experimented with element-wise vector multiplication and MIN, vector addition and MAX, and the tensor models presented in Section 6. Informal experimentation showed that directly embedding distributional information from verb vectors in the tensors (Section 6.4) works considerably better than the simple versions, so the results we report here are based on this approach. We also present results for a least squares fitting model, which approximates the distributional behaviour of holistic phrase/sentence vectors along the lines of (Baroni and Zamparelli, 2010). Specifically, for each verb, we compute an estimator that predicts the ith element of the resulting vector as follows:

$$\overrightarrow{w_i} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\overrightarrow{y_i}$$

Here, the rows of matrix $\mathbf{X}$ are the vectors of the subjects (or objects) that occur with our verb, and $\overrightarrow{y_i}$ is a vector containing the ith elements of the holistic phrase vectors across all training instances; the resulting $\overrightarrow{w_i}$'s form the rows of our verb matrix. This model could be only implemented for verb-object and subject-verb phrases due to data sparsity problems. As our focus is on analytic properties of features, we did not experiment with any neural model.

Regarding evaluation, since the tasks follow a binary classification objective and our models return a continuous value, we report *area under curve* (AUC). This reflects the generic discriminating power of a binary classifier by evaluating the task at every possible threshold. In all the experiments, we used a 300-dimensional PPMI vector space trained on the concatenation of UKWAC and Wikipedia corpora. The context was defined as a 5-word window around the target word.

8 Results and discussion

The results are presented in Table 1 (subject-verb and verb-object task) and Table 2 (subject-verb-object task). In all cases, a combination of a Frobenius tensor model with one of the sentence-level measures (*SAPinc*) gives the highest performance. In general, *SAPinc* and *SBalAPinc* work very well with all the tested compositional models, achieving a cross-model performance higher than that of any other metric, for all three tasks. From a feature inclusion perspective, we see that models employing an element of interesective composition (vector multiplication, MIN, relational and Frobenius tensor models) have consistent high performances across all the tested measures. The reason may be that the intersective filtering avoids generation of very dense vectors and thus facilitates entailment judgements based on the DIH. On the other hand, union-based compositional models, such as vector addition, MAX, and the projective tensor models, produce dense vectors for even very short sentences, which affects negatively the evaluation of entailment. The non-compositional verb-only baseline was worse than any compositional model other than the least-squares model, which is the only tensor model that did not perform well; this indicates that our algebraic tensor-based constructions are more robust against data sparsity problems than statistical models based on holistic vectors of phrases and sentences.

[2]The datasets are available at `http://compling.eecs.qmul.ac.uk/resources/`.

Subject-verb task									
Model	Inclusion	KL-div	αSkew	WeedsPrec	ClarkeDE	APinc	balAPinc	SAPinc	SBalAPinc
Verb	0.59	0.59	0.63	0.67	0.57	0.69	0.65	0.65	0.65
$\odot$	0.54	0.66	0.75	0.75	0.66	0.78	0.72	**0.81**	**0.81**
min	0.54	0.68	0.72	0.75	0.63	0.78	0.71	0.74	0.75
$+$	0.63	0.57	0.74	0.65	0.62	0.72	0.70	0.72	0.72
max	0.63	0.57	0.70	0.65	0.60	0.71	0.65	0.71	0.71
Least-Sqr	0.50	0.59	0.62	0.59	0.56	0.60	0.58	0.63	0.64
$\otimes_{\mathrm{proj}}$	0.59	0.59	0.65	0.67	0.59	0.70	0.67	0.71	0.69
$\otimes_{\mathrm{rel/frob}}$	0.54	0.64	0.77	0.74	0.68	0.78	0.73	**0.84**	**0.83**
Verb-object task									
Model	Inclusion	KL-div	αSkew	WeedsPrec	ClarkeDE	APinc	balAPinc	SAPinc	SBalAPinc
Verb	0.58	0.62	0.65	0.67	0.58	0.69	0.66	0.62	0.66
$\odot$	0.52	0.64	0.74	0.70	0.67	0.75	0.70	**0.82**	**0.79**
min	0.52	0.66	0.70	0.71	0.63	0.75	0.69	0.74	0.74
$+$	0.64	0.61	0.75	0.68	0.63	0.74	0.71	0.72	0.73
max	0.64	0.62	0.73	0.68	0.62	0.72	0.68	0.62	0.66
Least-Sqr	0.50	0.58	0.57	0.56	0.53	0.56	0.55	0.58	0.59
$\otimes_{\mathrm{proj}}$	0.58	0.60	0.66	0.67	0.60	0.70	0.67	0.68	0.68
$\otimes_{\mathrm{rel/frob}}$	0.52	0.63	0.75	0.71	0.67	0.75	0.70	**0.82**	**0.79**

Table 1: AUC results for the subject-verb and verb-object tasks. 'Verb' refers to a non-compositional baseline, where the vector/tensor of the phrase is taken to be the vector/tensor of the head verb. $\odot$, $+$ refer to vector element-wise multiplication and addition, respectively, $\otimes_{\mathrm{proj}}$ to the projective tensor models of Section 6.3, and $\otimes_{\mathrm{rel/frob}}$ to the construction of Section 6.1. The tensor models (except the least squares one) are further enhanced with information from the distributional vector of the verb, as detailed in Section 6.4.

Model	Inclusion	KL-div	αSkew	WeedsPrec	ClarkeDE	APinc	balAPinc	SAPinc	SBalAPinc
Verb	0.61	0.61	0.66	0.69	0.58	0.74	0.67	0.59	0.63
$\odot$	0.55	0.65	0.74	0.79	0.67	0.76	0.71	**0.80**	**0.80**
min	0.55	0.71	0.74	0.78	0.63	0.77	0.71	0.73	0.76
$+$	0.58	0.54	0.71	0.59	0.60	0.65	0.64	0.67	0.67
max	0.58	0.55	0.68	0.58	0.58	0.63	0.61	0.60	0.61
Least-Sqr	–	–	–	–	–	–	–	–	–
$\otimes_{\mathrm{rel}}$	0.51	0.64	0.78	0.79	0.69	0.79	0.72	0.84	0.83
$\otimes_{\mathrm{proj}}$	0.64	0.60	0.70	0.69	0.61	0.74	0.70	0.75	0.76
$\otimes_{\mathrm{CpSbj}}$	0.57	0.65	0.73	0.77	0.63	0.73	0.68	0.79	0.78
$\otimes_{\mathrm{CpObj}}$	0.54	0.62	0.73	0.72	0.64	0.76	0.71	0.81	0.79
$\otimes_{\mathrm{FrAdd}}$	0.60	0.60	0.75	0.72	0.67	0.77	0.75	0.84	0.82
$\otimes_{\mathrm{FrMul}}$	0.55	0.62	0.76	0.81	0.68	0.78	0.73	**0.86**	**0.83**

Table 2: AUC results for the subject-verb-object task. $\otimes_{\mathrm{Rel}}$ refers to the relational tensor model of Section 6.1, while $\otimes_{\mathrm{CpSbj}}$, $\otimes_{\mathrm{CpObj}}$, $\otimes_{\mathrm{FrAdd}}$, and $\otimes_{\mathrm{FrMul}}$ to the Frobenius models of Section 6.2. As in the other tasks, the distributional vector of the verb has been taken into account in all tensor-based models except Least-Sqr.

9 Conclusion and future work

In this paper we investigated the application of the distributional inclusion hypothesis on evaluating entailment between phrase and sentence vectors produced by compositional operators with a focus on tensor-based models. Our results showed that intersective composition in general, and the Frobenius tensor models in particular, achieve the best performance when evaluating upward monotone entailment, especially when combined with the sentence-level measures of (Kartsaklis and Sadrzadeh, 2016). Experimenting with different versions of tensor models for entailment is an interesting topic that we plan to pursue further in a future paper. Furthermore, the extension of word-level entailment to phrases and sentences provides connections with natural logic (MacCartney and Manning, 2007), a topic that is worth a separate treatment and constitutes a future direction.

Acknowledgments

The authors gratefully acknowledge support by EPSRC for Career Acceleration Fellowship EP/J002-607/1 and AFOSR International Scientific Collaboration Grant FA9550-14-1-0079.

References

E. Balkır, D. Kartsaklis, and M. Sadrzadeh. 2016a. Sentence Entailment in Compositional Distributional Semantics. In *Proceedings of the International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, Fort Lauderdale, FL, January.

Esma Balkır, Mehrnoosh Sadrzadeh, and Bob Coecke, 2016b. *Topics in Theoretical Computer Science: The First IFIP WG 1.8 International Conference, TTCS 2015, Tehran, Iran, August 26-28, 2015, Revised Selected Papers*, chapter Distributional Sentence Entailment Using Density Matrices, pages 1–22. Springer International Publishing, Cham.

Esma Balkır. 2014. Using density matrices in a compositional distributional model of meaning. Master's thesis, University of Oxford.

Desislava Bankova, Bob Coecke, Martha Lewis, and Daniel Marsden. 2016. Graded entailment for compositional distributional semantics. *arXiv preprint arXiv:1601.04908*.

M. Baroni and R. Zamparelli. 2010. Nouns are Vectors, Adjectives are Matrices. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jianpeng Cheng and Dimitri Kartsaklis. 2015. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1531–1542, Lisbon, Portugal, September. Association for Computational Linguistics.

Daoud Clarke. 2009. Context-theoretic semantics for natural language: an overview. In *Proceedings of the workshop on geometrical models of natural language semantics*, pages 112–119. Association for Computational Linguistics.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36.

Ido Dagan, Lillian Lee, and Fernando C. N. Pereira. 1999. Similarity-based models of word cooccurrence probabilities. *Mach. Learn.*, 34(1-3):43–69.

John R. Firth. 1957. A Synopsis of Linguistic Theory, 1930-1955. *Studies in Linguistic Analysis*, pages 1–32.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 107–114, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404. Association for Computational Linguistics.

Aurélie Herbelot and Mohan Ganesalingam. 2013. Measuring semantic content in distributional vectors. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 440–445. Association for Computational Linguistics.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June. Association for Computational Linguistics.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2014. A study of entanglement in a categorical framework of natural language. In B. Coecke, I. Hasuo, and P. Panangaden, editors, *Quantum Physics and Logic 2014 (QPL 2014). EPTSC 172*, pages 249–261.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2016. A Compositional Distributional Inclusion Hypothesis. In *Proceedings of the 9th Conference on Logical Aspects of Computational Linguistics (LACL)*, Lecture Notes in Computer Science. Springer. To appear.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2012. A unified sentence space for categorical distributional-compositional semantics: Theory and experiments. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, pages 549–558.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(04):359–389.

Lillian Lee. 1999. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 25–32.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the International Conference on Machine Learning*, pages 296–304.

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *ACL Workshop on Textual Entailment and Paraphrasing*. Association for Computational Linguistics.

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 708–719, Doha, Qatar, October. Association for Computational Linguistics.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1439.

R. Socher, B. Huval, C. Manning, and Ng. A. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Conference on Empirical Methods in Natural Language Processing 2012*.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th international conference on Computational Linguistics*, number 1015. Association for Computational Linguistics.

Parameter estimation of Japanese predicate argument structure analysis model using eye gaze information

Ryosuke Maki **Hitoshi Nishikawa** **Takenobu Tokunaga**
Department of Computer Science
Tokyo Institute of Technology
{maki.r.aa@m, {hitoshi, take}@c}.titech.ac.jp

Abstract

In this paper, we propose utilising eye gaze information for estimating parameters of a Japanese predicate argument structure (PAS) analysis model. We employ not only linguistic information in the text, but also the information of annotator eye gaze during their annotation process. We hypothesise that annotator's frequent looks at certain candidates imply their plausibility of being the argument of the predicate. Based on this hypothesis, we consider annotator eye gaze for estimating the model parameters of the PAS analysis. The evaluation experiment showed that introducing eye gaze information increased the accuracy of the PAS analysis by 0.05 compared with the conventional methods.

1 Introduction

In recent years, there have been many attempts of annotating corpora with various kinds of information, as supervised machine learning (ML) techniques had been a significant device for natural language processing (NLP) (Pustejovsky and Stubbs, 2012). The annotated corpus is used as training data for constructing a task model, considering the annotated information as expected output of the model. In the current framework, however, only annotated information in the corpus is used for training. In this paper, we propose utilising the information of annotator behaviour during the annotation process as well as resulting annotated information for training the task model (Tokunaga et al., 2013).

We take the predicate argument structure (PAS) analysis of Japanese texts as the target task in the present work. Predicate argument relations are usually marked by case particles denoting grammatical cases in Japanese, therefore identifying dependencies marked by the major obligatory cases, *ga* (nominative), *wo* (accusative) and *ni* (dative) is the main task. However, since ellipses are ubiquitous in Japanese texts, arguments might be identified beyond the sentence including the target predicates (inter-sentence arguments) as well as within the sentence (intra-sentence arguments). This feature is different from the PAS analysis in English in which arguments can be found in the same sentence of the target predicate in most cases. Another complication is the case alternation caused by certain types of auxiliary verbs such as causative and passivisation verbs. In such cases, the original case should be recovered in the PAS analysis. In this respect, the Japanese PAS analysis shares the similarity with semantic role labelling in English (Gildea and Jurafsky, 2002). Furthermore, treating "event nouns" (Komachi et al., 2007) as predicates, we identify their arguments as well as the arguments of verbs and adjectives. Currently, several PAS annotated Japanese corpora such as NAIST Text Corpus (NTC) (Iida et al., 2007b) and BCCWJ-DepParaPAS (Ueda et al., 2015; Maekawa et al., 2014) are available. We use the latter in this study.

In order to improve the PAS analysis performance, we propose to utilise the information of annotator behaviour, particularly eye gaze information, during their annotating predicate argument relations in texts. In the past PAS analysis, a model has been constructed by utilising a certain ML technique regarding the human annotated argument as the correct argument for a given predicate, i.e. it is considered the positive example for the predicate and all other argument candidates are negative examples. Observing

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2861–2869, Osaka, Japan, December 11-17 2016.

the annotator eye movement during their annotation, however, they look at other argument candidates as well until their final decision. Although the not-selected candidates are not the argument of the target predicate in the text, frequently-looked candidates could be an argument of the predicate in similar but different texts. Our idea is to utilise the information gained from those "near miss" candidates in the training phase of the model.

Consider the following two example texts.

(1) *watasi-wa kinou tomodati-to ranti-wo tabeni ikimasita.*
I-TOPIC yesterday friend-with lunch-ACC to eat went

nedan-ni mo azi-ni mo manzokusimasita.
price-DAT also taste-DAT also was satisfied

'Yesterday, I went to have lunch with my friend. I was satisfied with its price and taste.'

(2) *watasi-wa kinou tomodati-to ranti-wo tabeni ikimasita.*
I-TOPIC yesterday friend-with lunch-ACC to eat went

nedan-ni mo azi-ni mo manzokusita youdesu.
price-DAT also taste-DAT also was satisfied seem

'Yesterday, I went to have lunch with my friend. He seemed to be satisfied with its price and taste.'

Although these texts are almost the same on the surface except for the verb ending in the second sentences, the *ga* (nominative) arguments of the predicate '*manzokusuru* (be satisfied)' are different; the *ga* argument in (1) is '*watasi* (I)' in the first sentence while that in (2) is '*tomodati* (friend)'. This difference is caused by the modality auxiliary '*youdesu* (seem)' in the second sentence of (2). Treating these texts as a part of the training examples for binary classification, '*watasi* (I)' is regarded as a positive example, and other candidates including '*tomodati* (friend)' as negative examples in the text (1). However, '*tomodati* (friend)' looks a better negative example than others in (1). This is also the case for (2) but in the opposite way. Our proposal treats '*tomodati* (friend)' in (1) and '*watasi* (I)' in (2) as "near miss" candidates, i.e. better negative examples than others in each text, and takes them into account in the training. To salvage the "near miss" candidates that were merely discarded in the existing PAS analysis based on binary classification, we build our PAS analyser with a learning-to-rank framework; we make a ranking of candidates instead of their binary positive-negative distinction.

It has been reported that the eye gaze information contributes to detecting annotation disagreement between annotators (Mitsuda et al., 2013), but there has been no study on the Japanese PAS analysis using eye gaze information. We hypothesise that frequent looks at argument candidates imply their plausibility of being the argument of the predicate. In this study, we make a candidate ranking based on the frequency of the annotator gaze at the candidates for training a model with the ranking SVM (Joachims, 2002)

This paper is organised as follows. Section 2 overviews the related work, and in Section 3, we define the task setting and propose a method for Japanese PAS analysis. Section 4 discusses the evaluation results and Section 5 concludes the paper.

2 Related work

Basic features for the PAS analysis were proposed in the studies in English (Gildea and Jurafsky, 2002). Unlike English, subject ellipses frequently occur in Japanese. Thus dealing with ellipses, i.e. zero anaphora resolution, is fundamental in processing Japanese texts, and linguistic features for Japanese zero anaphora resolution have been proposed in the past research (Iida et al., 2007a; Sasano and Kurohashi, 2011; Komachi et al., 2007).

The past Japanese PAS analysis methods can be categorised into two classes: one constructs an individual model for predicting each case (*ga* , *wo* , and *ni*) (Taira et al., 2008; Imamura et al., 2009; Hayashibe et al., 2011) and another constructs a single model for predicting all three cases at the same time. The

latter is further divided into two types: (i) identifying all the three cases of one predicate (Sasano and Kurohashi, 2011; Yoshikawa et al., 2011; Hangyo et al., 2013), and (ii) identifying all the three cases of all predicates in the same sentence (Ouchi et al., 2015; Shibata et al., 2016). Since *ga* arguments tend to be omitted more than other cases, we focus on identifying the *ga* case in the present study, thus our proposing method belongs to the former class.

Table 1: Features for PAS analysis

category	ID	feature name	description
predicate	1	lemma	a lemma of the predicate
	2	word origin	originated in Japanese, Chinese, other language or compound
	3	parts of speech	POS of the predicate
	4	conjugation form	a conjugation form of the predicate
	5	conjugation type	a conjugation type of the predicate
	6	surface form	a surface form of the predicate
argument	7	lemma	a lemma of the argument candidate
	8	word origin	originated in Japanese, Chinese, other language or compound
	9	parts of speech	POS of the argument candidate
	10	surface form	a surface form of the argument candidate
	11	case marker	a case particle following the argument candidate
	12	semantic category	a semantic category of the argument candidate
predicate and argument	13	lemma pair	a pair of lemmas of the predicate and the argument candidate
	14	distance	distance between the predicate and the argument candidate
	15	intra/inter	a binary value indicating if the predicate and the argument candidate are in the same sentence or not
	16	intra/inter+case marker	combination of the feature 11 and 15
	17	semantic category pair	a pair of semantic categories of the predicate and the argument candidate
	18	dependency	dependency type: direct dependency, indirect dependency and no dependency

3 Introducing eye gaze information into PAS analysis

3.1 Task setting

The task in this study is identifying the *ga* argument of a specified predicate in a text. We particularly focus on the *ga* case since it tends to be omitted more than other cases in Japanese. Given a target predicate and argument candidates preceding the predicate in a text as an input, the learnt model is expected to select a candidate as the correct *ga* argument as the output. The goal of this study is to show that eye gaze information is useful for training the PAS analysis model.

3.2 Detecting fixations

As detailed below, we detect fixations from the recorded gaze sequence by using the Dispersion-Threshold Identification Algorithm (Salvucci and Goldberg, 2000). A fixation on a word in texts is widely believed to have some relation with the cognitive process on that word (Just and Carpenter, 1980). The overview of the algorithm is described below.

1. A gaze point is added to a set one by one as far as the following conditions hold.
   - All gaze points in the set resides within the distance threshold D from the centre-of-gravity of the set.
   - No tracking error was flagged in the set.

   Just before violating the above conditions, the centre-of-gravity of the set is identified as a fixation candidate.
2. Repeat Step 1. to the rest of the gaze sequence.
3. For each fixation candidate, if its duration time is T or longer, we identify the candidate as a fixation.

Following Mitsuda et al. (2013), we set the space and time threshold D and T as 16 pixels and 100 msec respectively. By making the correspondence between the fixation points and the bounding boxes of argument candidates, we obtain the data indicating what argument candidates the annotators looked at at a certain duration at a certain time point.

3.3 Features

We use the features proposed in the past studies (Taira et al., 2008; Imamura et al., 2009) to represent each argument candidate as shown in Table 1. The distance feature d is calculated by the number of words between the predicate and the candidate, which is normalised to $d \in [0, 1]$. The semantic category feature adopts the categories defined in the Japanese thesaurus 'Nihongo Goi Taikei' (Ikehara et al., 1997).

3.4 Ranking argument candidates

We make the ranking of argument candidates for each training instance represented in terms of the features in Table 1 by using the fixations in the following way.

1. The correct candidate is placed at the top of the ranking.
2. The most frequently fixated candidate other than the correct one follows the correct candidate in the ranking.
3. The candidates with no fixation are placed at the bottom of the ranking with equal rank.
4. Any other candidates are excluded from the ranking.

Using these rankings of argument candidates as training data for the ranking SVM (Joachims, 2002), we estimate the model parameter. In the test phase, the most highly ranked candidate by the model is considered as the model output. Note that we do not require any eye gaze information in the test phase. The eye gaze information is required only for estimating the model parameters.

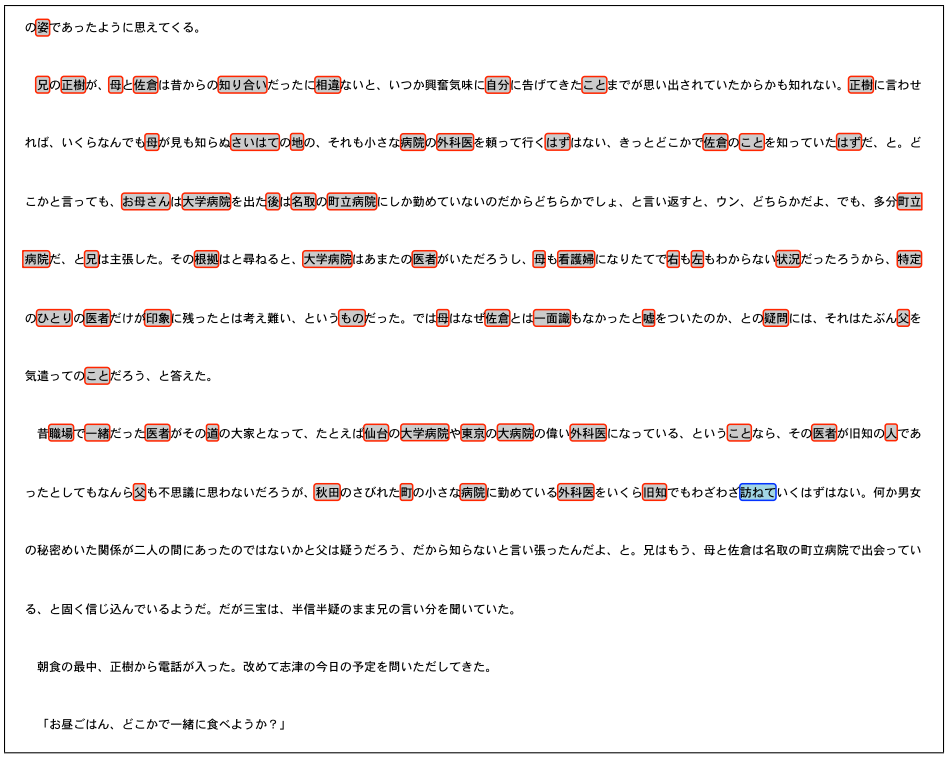


Figure 1: Screenshot of annotation interface

4 Evaluation

4.1 Data

We use the data collected by Mitsuda et al. (2014) for evaluation. Mitsuda et al. (2014) conducted an experiment for collecting annotator's behavioural data during the PAS annotation in Japanese texts. Given a single predicate in a text in which its argument candidates were marked on the screen, the annotators were instructed to identify the *ga* (nominative) argument of the target predicate by clicking it with a mouse. Figure 1 shows a screenshot of the annotation interface, in which a single target predicate in the text is highlighted in blue and *ga* argument candidates are highlighted in gray.

The texts used in their experiment were sampled from Balanced Corpus of Contemporary Written Japanese (BCCWJ) (Maekawa et al., 2014). BCCWJ contains approximate 100 million words collected from around 170 thousand texts of various domains. The texts are annotated with various kinds of information at bibliographic and morphological levels. BCCWJ has two types of schema for dividing a sentence into words: Long Unit Word (LUW) and Short Unit Word (SUW). Roughly speaking, an LUW corresponds to a compound noun, whereas an SUW corresponds to a component word of the compound. The LUW were used as argument candidates in their experiment.

BCCWJ consists of three sub-corpora (Publication, Library, and Special-purpose) and each sub-corpus has several registers. For example, PB register, whose texts are sampled from books, is a part of the publication sub-corpus.

About one percent of BCCWJ is defined as the "core data", and the core data is manually annotated with richer linguistic information. The core data annotated with dependency structures, coordinate structures, coreference relations and predicate argument structures is particularly called BCCWJ-DepParaPAS (Ueda et al., 2015; Maekawa et al., 2014). BCCWJ-DepParaPAS adopts the annotation schema of NAIST Text Corpus (Iida et al., 2007b) where three obligatory arguments, *ga* (nominative), *wo* (accusative) and *ni* (dative) are annotated at the SUW level.

The 221 texts used in their experiment were sampled from the PB register of BCCWJ-DepParaPAS, thus they have rich linguistic information including the PAS information. Among these 221 texts we discarded 37 texts because of the discrepancy in the annotation schema between BCCWJ-DepParaPAS and the experiment conducted by Mitsuda et al. (2014). The remaining 184 texts comprise 107 texts with intra-sentence *ga* arguments and 77 texts with inter-sentence ones. The number of candidates including the correct one in a text ranges from 12 to 109, with the median being 60.

In the experiment by Mitsuda et al. (2014), the texts were independently annotated by 20 annotators. During the annotation, annotator eye gaze was captured by the eye tracker Tobii T60 at intervals of 1/60 second. Although recent development of the eye-tracking technology enables us to capture eye gaze handily and precisely, we have still errors in the eye tracking data due to various factors such as the ambient lighting conditions, the annotator's glasses, and the characteristics of annotator's eyes. Among data from 20 annotators, we discarded the data from 13 annotators because their data included a session the tracking error rate of which exceeded 30%. Here "a session" stands for a single annotation session in which an annotator chooses a *ga* argument of the specified predicate in a single text. In summary, we utilise the data consisting of the 184 texts annotated by seven annotators.

Training data We prepare the following three types of models for evaluation and construct the training data for each model from the data explained above.

- **Binary Regression (BiReg) model**:
  This model chooses the argument candidate with the highest regression value as the correct *ga* argument. In the training examples, only the correct argument, i.e. the manually annotated *ga* argument, is considered a positive example, and all other candidates in the text are negative examples.

- **Distance Ranking (DRank) model**:
  This model ranks the argument candidates according to their plausibility of being the *ga* argument of the specified predicate. The distance between the predicate and the candidate is used for calculating the rank. The training examples are created as follows. For each annotation instance, the correct

argument is placed at the top of the ranking, and the other candidates are ranked in the ascending order of their distance feature, i.e. the normalised number of LUWs between the predicate and the candidate. We prepare this model for verifying the effectiveness of the ranking model based on the other metric than the fixation frequency.

- **Fixation Ranking (FixRank) model**:
  This model also ranks the argument candidates according to their plausibility of being the *ga* argument of the specified predicate. Training examples are created based on the number of fixations on the candidates as described in 3.4.

Test data We sampled 29,519 predicate instances from the PB register in the BCCWJ core data: 21,816 intra-sentence instances and 7,703 inter-sentence instances. We kept the range of the number of candidates in a text of the test data as the same as that of the training data, ranging from 12 to 109 with the median being 48.

4.2 Results and discussion

For the training of the BiReg model, we utilised Classias (Okazaki, 2009). As a learning algorithm, L2 regularised logistic regression was adopted. For the training of the DRank and FixRank models, we utilised SVM$^{\text{rank}}$ (Joachims, 2006). As a slack variable, L1-norm was adopted. We prepared four variations of the feature set for model training: a base set (F_{base}), the base set with semantic category features (F_{sem}), the base set with syntactic dependency features (F_{syn}), and the base set with both semantic and syntactic features (F_{synsem}). These feature sets are summarised in Table 2.

Table 2: Variations of feature set

feature set	feature ID defined in Table 1
F_{base}	1–11, 13–16
F_{sem}	1–17
F_{syn}	1–11, 13–16, 18
F_{synsem}	1–18

The all models estimated *ga* likeliness of each candidate, and then selected the candidate with the maximum score as the *ga* argument. If the selected candidate agreed with the answer, it was judged to be correct. Even though the selected candidates were in the same coreference chain as the answer argument, we judged it as wrong selection. We adopted a strict criterion in correctness.

Table 3: Evaluation result (Accuracy)

model	feature set	intra	inter	total
BiReg	F_{base}	0.56	0.04	0.42
	F_{sem}	0.48	**0.06**	0.37
	F_{syn}	0.58	0.03	0.44
	F_{synsem}	0.52	0.05	0.40
DRank	F_{base}	0.47	0.01	0.35
	F_{sem}	0.47	0.01	0.35
	F_{syn}	0.50	0.01	0.37
	F_{synsem}	0.51	0.01	0.38
FixRank	F_{base}	0.55	0.02	0.41
	F_{sem}	0.49	0.02	0.37
	F_{syn}	**0.63**	0.02	**0.47**
	F_{synsem}	0.58	0.02	0.43

Table 3 shows the accuracy of each model and feature set combination. We calculated the accuracy for two groups: intra-sentence cases, i.e. a predicate and its *ga* argument are in the same sentence, and inter-sentence cases, i.e. they are not in the same sentence. Table 3 shows that FixRank+F_{syn} shows the

highest accuracy in the 'intra-sentence' and 'total' columns, while the BiReg model, particularly with F_{sem}, shows the better accuracy in the 'inter-sentence' column.

As far as this result is concerned, the FixRank+F_{syn} model predicts intra *ga* arguments the best of all the models. This supports our hypothesis that frequent looks at argument candidates imply their plausibility of being the argument of the predicate, and utilising eye gaze information contributes to the improvement in parameter estimation of the PAS analysis model.

The table also showed that the semantic feature set F_{sem} does not work well except for the inter-sentence cases with the BiReg model. We automatically assigned the semantic categories to the arguments. As Taira et al. (2008) did, it is worthwhile to see if manual assignment of the semantic categories would improve the effectiveness of F_{sem}.

We need to mention that the DRank model showed the worst performance among three models. The training examples for the DRank model would contain more superfluous information than the FixRank model because the candidates were rigidly ordered in the ascending order of the distance feature, thus there was no candidate of equal rank. This rigid ranking could make the DRank model fail to capture better negative examples.

Table 4: Comparison between BiReg+F_{syn} model and FixRank+F_{syn} model

(15) intra/inter	(18) dependency	(11) case marker	BiReg+F_{syn} P	BiReg+F_{syn} R	BiReg+F_{syn} F	FixRank+F_{syn} P	FixRank+F_{syn} R	FixRank+F_{syn} F
intra	direct	*ga*	0.949	0.884	0.915	0.948	0.975	0.962
		others	0.513	0.463	0.487	0.403	0.693	0.509
	indirect	*ga*	0.227	0.602	0.330	0.242	0.561	0.338
		others	0.184	0.166	0.175	0.156	0.103	0.124
	no	*ga*	0.294	0.892	0.442	0.352	0.741	0.477
		others	0.239	0.469	0.316	0.306	0.290	0.297
inter	no	*ga*	0.232	0.050	0.082	0.284	0.025	0.046
		others	0.247	0.031	0.055	0.172	0.014	0.025

"others" in the case marker column includes arguments without case markers.
P, R, and F denote precision, recall, and F-measure, respectively.

For further analysis, we focused on the specific three features: (11) case marker, (15) intra/inter, and (18) dependency, and calculated precision (P), recall (R), and F-measure (F) for the test cases with each combination of these features. Table 4 shows the comparison between the BiReg+F_{syn} model and the FixRank+F_{syn} model that show the highest accuracy in total in Table 3.

The notable difference between these two model is that the FixRank model shows the better F-measures for directly dependent arguments, particularly in *ga* arguments. Concerning the directly dependent *ga* argument cases, the FixRank shows the better recall value with the similar precision value. In contrast, in the cases of directly dependent 'other' case markers, the BiReg shows the better precision value. We investigated the number of directly dependent arguments that each model selected as the answer. The FixRank model selected 19,612 directly dependent arguments, and 13,751 *ga* argument among them, while the BiReg model selected 12,520 directly dependent arguments with 7,208 *ga* arguments. These numbers suggest that the FixRank model tends to choose directly dependent arguments regardless of their case markers.

Concerning the FixRank model, we further investigated the number of the second rank training candidates that directly depend on the target predicate, i.e. the directly dependent "better negative examples", to obtain 428 candidates out of 1,288 candidates ($= 184$ texts $\times$ 7 annotators) that make 33% of the total training examples. On the other hand, the number of the correct arguments that directly depend on the target predicate is 35 out of 184 examples, making 19% of the texts. These numbers suggest that the annotators tend to look at directly dependent arguments more during the annotation, the FixRank model tends to select directly dependent arguments as our method proposes frequently fixated arguments as "better negative candidates". This tendency would explain the poor performance of the FixRank model in the inter-sentence cases.

5 Conclusion

We proposed utilising the information of annotator eye gaze during their annotation for estimating the parameters of the Japanese PAS analysis model. Through the evaluation in which the *ga* argument of a specified predicate was identified, we confirmed the effectiveness of the eye gaze information for the PAS analysis. We gained 0.05 point increase in accuracy by introducing the eye gaze information into the parameter estimation. The accuracy for the inter-sentence arguments, however, still remains very low. Moreover, the eye gaze information does not work well on the inter-sentence arguments. We need to refine the usage of the eye gaze information for further improvement of the PAS analysis. Currently we use only the fixation frequency, but the fixation duration might provide the information from different aspects. Also instead of utilising all fixations, their selective usage would be another exploring direction.

In the current evaluation, we used the DRank model as the baseline of the ranking model, but its performance is too poor to be a fair baseline. It would be necessary to adopt a state-of-the-art ranking model such as Hangyo et al. (2013) for further evaluation.

The eye gaze information has attracted much attention in various NLP tasks in recent years such as dialogue systems (Prasov et al., 2007; Qu and Chai, 2007), reference resolution (Prasov and Chai, 2008), dependency parsing (Barrett and Søgaard, 2015), coreference resolution (Ross et al., 2016), named entity recognition (Tomanek et al., 2010). Exploring the effectiveness of eye gaze information during annotation in other NLP tasks would be another important research direction.

References

Maria Barrett and Anders Søgaard. 2015. Using reading behavior to predict grammatical functions. In *Proceedings of the Sixth Workshop on Cognitive Aspects of Computational Language Learning*, pages 1–5.

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

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2013. Japanese zero reference resolution considering exophora and author/reader mentions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 924–934.

Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2011. Japanese predicate argument structure analysis exploiting argument position and type. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 201–209.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2007a. Zero-anaphora resolution by learning rich syntactic pattern features. *ACM Transactions on Asian Language Information Processing (TALIP)*, 6(4):1:1–1:22.

Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007b. Annotating a Japanese text corpus with predicate-argument and coreference relations. In *Proceedings of ACL 2007 Workshop on Linguistic Annotation*, pages 132–139.

Satoru Ikehara, Masahiro Miyazaki, Satoshi Shirai, Akio Yokoo, Hiromi Nakaiwa, Kentaro Ogura, Yoshifumi Ooyama, and Yoshihiko Hayashi. 1997. *Nihongo Goi Taikei, A Japanese Lexicon*. Iwanami Shoten, Tokyo.

Kenji Imamura, Kuniko Saito, and Tomoko Izumi. 2009. Discriminative Approach to Predicate-argument Structure Analysis with Zero-anaphora Resolution. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 85–88.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD 2002)*, pages 133–142.

Thorsten Joachims. 2006. Training Linear SVMs in Linear Time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Marcel Adam Just and Patricia A. Carpenter. 1980. A theory of reading: From eye fixations to comprehension. *Psychological Review*, 87(4):329–354.

Mamoru Komachi, Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2007. Learning-based argument structure analysis of event-nouns in Japanese. In *Proceedings of the Conference of the Pacific Association for Computational Linguistics*, pages 120–128.

Kikuo Maekawa, Makoto Yamazaki, Toshinobu Ogiso, Takehiko Maruyama, Hideki Ogura, Wakako Kashino, Hanae Koiso, Masaya Yamaguchi, Makiro Tanaka, and Yasuharu Den. 2014. Balanced corpus of contemporary written Japanese. *Language resources and evaluation*, 48(2):345–371.

Koh Mitsuda, Ryu Iida, and Takenobu Tokunaga. 2013. Detecting missing annotation disagreement using eye gaze information. In *Proceedings of the 11th Workshop on Asian Language Resources*, pages 19–26.

Koh Mitsuda, Ryu Iida, and Takenobu Tokunaga. 2014. Collection and analysis of eye gaze information in single predicate-argument relation annotation (in Japanese). In *Proceedings of Special Interest Group on Natural Language Processing (IPSJ-SIGNL)*, volume 2014-NL-217-2, pages 1–9. Information Processing Society of Japan.

Naoaki Okazaki. 2009. Classias: a collection of machine-learning algorithms for classification. `http://www.chokkan.org/software/classias/`.

Hiroki Ouchi, Hiroyuki Shindo, Kevin Duh, and Yuji Matsumoto. 2015. Joint case argument identification for Japanese predicate argument structure analysis. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 961–970.

Zahar Prasov and Joyce Y. Chai. 2008. What's in a gaze?: The role of eye-gaze in reference resolution in multimodal conversational interfaces. In *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 20–29.

Zahar Prasov, Joyce Y. Chai, and Hogyeong Jeong. 2007. Eye gaze for attention prediction in multimodal human-machine conversation. In *Proceedings of the AAAI Spring Symposium on Interaction Challenges for Artificial Assistants*, pages 102–110.

James Pustejovsky and Amber Stubbs. 2012. *Natural Language Annotation for Machine Learning*. O'Reilly.

Shaolin Qu and Joyce Y. Chai. 2007. An exploration of eye gaze in spoken language processing for multimodal conversational interfaces. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 284–291.

Joe Cheri Ross, Abhijit Mishra, and Pushpak Bhattacharyya. 2016. Leveraging annotators' gaze behaviour for coreference resolution. In *Proceedings of the 7th Workshop on Cognitive Aspects of Computational Language Learning*, pages 22–26.

Dario D. Salvucci and Joseph H. Goldberg. 2000. Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications (ETRA 2000)*, pages 71–78.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to Japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 758–766.

Tomohide Shibata, Daisuke Kawahara, and Sadao Kurohashi. 2016. Neural network-based model for Japanese predicate argument structure analysis. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1235–1244.

Hirotoshi Taira, Sanae Fujita, and Masaaki Nagata. 2008. A Japanese predicate argument structure analysis using decision lists. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 523–532.

Takenobu Tokunaga, Ryu Iida, and Koh Mitsuda. 2013. Annotation for annotation – toward eliciting implicit linguistic knowledge through annotation –. In *Proceedings of the 9th Joint ISO - ACL SIGSEM Workshop on Interoperable Semantic Annotation (ISA-9)*, pages 79–83.

Katrin Tomanek, Udo Hahn, Steffen Lohmann, and Jürgen Ziegler. 2010. A cognitive cost model of annotations based on eye-tracking data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, pages 1158–1167.

Yoshiko Ueda, Ryu Iida, Masayuki Asahara, Yuji Matsumoto, and Takenobu Tokunaga. 2015. Predicate-argument structure and coreference relation annotation on 'balanced corpus of contemporary written Japanese' (in Japanese). In *Proceedings of the 8th Japanese corpus linguistics workshop*, pages 205–214.

Katsumasa Yoshikawa, Masayuki Asahara, and Yuji Matsumoto. 2011. Jointly extracting Japanese predicate-argument relation with markov logic. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1125–1133.

Reading and Thinking: Re-read LSTM Unit for Textual Entailment Recognition

Lei Sha, Baobao Chang, Zhifang Sui, Sujian Li
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
Collaborative Innovation Center for Language Ability, Xuzhou 221009 China
{shalei, chbb, szf, lisujian}@pku.edu.cn

Abstract

Recognizing Textual Entailment (RTE) is a fundamentally important task in natural language processing that has many applications. The recently released Stanford Natural Language Inference (SNLI[1]) corpus has made it possible to develop and evaluate deep neural network methods for the RTE task. Previous neural network based methods usually try to encode the two sentences and send them together into multi-layer perceptron, or use LSTM-RNN to link two sentence together while using attention mechanic to enhance the model's ability. In this paper, we propose to use the intensive reading mechanic, which means to re-read the sentence (read the sentence again) according to the memory of the other sentence for a better understanding of the sentence pair. The re-read process can be applied alternatively between the two sentences. Experiments show that we achieve results better than current state-of-art equivalents.

1 Introduction

For the natural language, a common phenomenon is that there exist a lot of ways to express the same or similar meaning. To discover such different expressions, the Recognizing Textual Entailment (RTE) task is proposed to judge whether the meaning of one text (denoted as *hypothesis*) can be inferred (entailed) from the other one (*premise*) (Dagan et al., 2006). A simple example is shown in Table 1. For many natural language processing applications like question answering, information retrieval which need to deal with the diversity of natural language, recognizing textual entailments is a critical step.

Most previous neural network based methods are sentence encoding-based models. They applied a large variety of methods to encode the two sentences (premise and hypothesis), such as LSTM encoder, GRU encoder and tree-based CNN encoder. Then combine them as the feature of this sentence pair and send into a deep neural network for classification. However, in the sentence encoding process, the premise and the hypothesis cannot affect each other. It is well known that the encoding procedure is just automatically learning useful features. Without the impact between the two sentences, it is difficult for the encoder to extract the sentence-relationship-specific features. Other methods mainly make use of attention mechanism to capture the word-by word alignment information while training (Rocktäschel et al., 2015) or just integrate memory network into LSTM to make the model remember more information (Cheng et al., 2016). Among them, only the attention mechanism can make the two sentences contact with each other. However, the word-by-word attention does not represent a better understanding of the sentences.

When deciding the entailment relationship between a pair of sentences, what is really matters? Unlike paraphrasing and machine translation, entailment relationship does not force the two sentences have the same meaning. Instead, as long as the premise can cover the meaning of the hypothesis, the entailment stands. Therefore, if the premise entails the hypothesis, that doesn't really mean that the words in hypothesis can be totally entailed by the words in premise. For example, "these girls are having a great

[1] http://nlp.stanford.edu/projects/snli/

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2870–2879, Osaka, Japan, December 11-17 2016.

Relationship	Premise & Hypothesis	
Entailment	Premise:	This church choir sings to the masses as they sing joyous songs from the book at a church.
	Hypothesis:	The church is filled with song.
Neutral	Premise:	This church choir sings to the masses as they sing joyous songs from the book at a church.
	Hypothesis:	The church has cracks in the ceiling.
Contradict	Premise:	This church choir sings to the masses as they sing joyous songs from the book at a church.
	Hypothesis:	A choir singing at a baseball game.

Table 1: Example of entailment / neutral / contradict eases.

time looking for seashells" can entail "the girls are outside". These entailment cases certainly cannot be solved by word-by-word alignment based methods since "outside" cannot be aligned to any of the words in the premise. Intuitively, when a human is judging the relationship of the two sentences, he/she would first read the premise, and then read the hypothesis while considering whether it can be entailed by the premise. Therefore, we intend to make the model more like human, namely, we require the model be capable of reading and thinking.

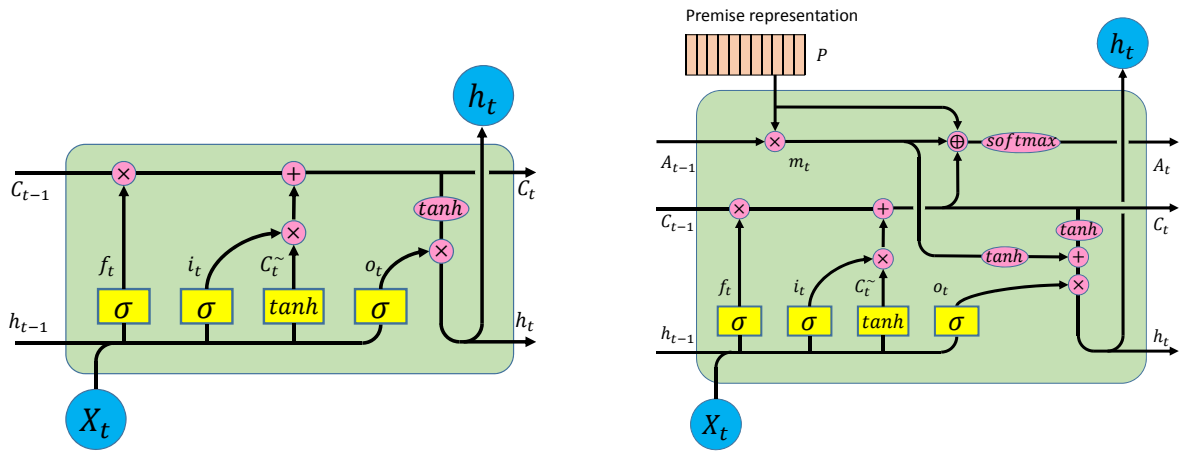
In this paper, we propose a new LSTM variant called re-read LSTM unit (rLSTM), which also take the attention vector of one sentence as an inner state while reading the other sentence. Therefore, this kind of unit is specially designed for dual sentence modeling. Then we use a standard bidirectional LSTM to read the premise, and use a bidirectional rLSTM to read the hypothesis. The output of the standard BiLSTM is taken as the general input of the bidirectional rLSTM. Experiments show that our method has outperformed the state-of-the-art approaches.

2 Related Work

Textual Entailment Recognizing (RTE) task has been widely studied by many previous work. Firstly, the methods use statistical classifiers which leverage a wide variety of features, including hand-engineered features derived from complex NLP pipelines and similarity between sentences (T and H) and sentence pairs ((T', H') and (T'', H''))(Malakasiotis and Androutsopoulos, 2007; Jijkoun and de Rijke, 2005; Wan et al., 2006; Zanzotto and Moschitti, 2006; Wang and Neumann, 2007; Dinu and Wang, 2009; Nielsen et al., 2009; Malakasiotis, 2011). This kind of methods are hard to generalize due to the complexity of feature engineering. Moreover, the hand-engineered features usually cannot represent implicit meanings of sentences.

Secondly, (Hickl, 2008; Sha et al., 2015; Shnarch et al., 2011b; Shnarch et al., 2011a; Beltagy et al., 2013; Rios et al., 2014) extract the structured information (discourse commitments or predicate-argument representations) in T-H pair and check if the information in T contains or can infer the information in H. Probabilistic methods are used for recognizing the entailment. However, these work are still based on hand-engineered features which is not easy to generalize.

Recently, neural network based methods start to show its effectiveness. Based on (Bowman et al., 2015), Rocktäschel et al. (2015) uses the attention-based technique to improve the performance of LSTM-based recurrent neural network. Then, Yin et al. (2015) applied attention mechanic to convolution neural network, Liu et al. (2016a) proposed coupled-LSTM, Vendrov et al. (2015) proposed ordered embedding, Mou et al. (2016) applied Tree-based CNN, Wang and Jiang (2015) proposed matching LSTM, Liu et al. (2016b) applied inner-attention, Cheng et al. (2016) proposed Long Short-Term Memory-Networks to improve the performance. To free the model from traditional parsing process, Bowman et al. (2016) combines parsing and interpretation within a single tree sequence hybrid model by integrating tree structured sentence interpretation into the linear sequential structure of a shift-reduce parser. Parikh et al. (2016) uses attention to decompose the problem into subproblems that can be solved separately, thus making it trivially parallelizable.



(a) LSTM Unit (b) rLSTM Unit

Figure 1: The inner architecture of the traditional LSTM unit and the re-read LSTM unit.

3 Model

3.1 Background

The LSTM architecture (Hochreiter and Schmidhuber, 1997) addresses this problem of learning long-term dependencies by introducing a memory cell that is able to preserve state over long periods of time. While numerous LSTM variants have been described, here we describe the most widely-used version.

Long short-term memory (LSTM) based recurrent neural networks (RNNs) have long been tried to apply to a wide range of NLP tasks, including RTE (Bowman et al., 2015). We define the LSTM unit at each time step t to be a collection of vectors in $\mathbb{R}^d$: an input gate i_t, a forget gate f_t, an output gate o_t, a memory cell c_t, candidate memory cell state $\widetilde{C}_t$ and a hidden state h_t. The entries of the gating vectors i_t, f_t and o_t are in $[0, 1]$. We refer to d as the memory dimension of the LSTM. The LSTM transition equations are listed in Eq 1.

$$
\begin{aligned}
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) & \widetilde{C}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c) \\
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) & c_t &= i_t \odot \widetilde{C}_t + f_t \odot c_{t-1} \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) & h_t &= o_t \odot \tanh(c_t)
\end{aligned} \tag{1}
$$

where x_t is the input at the current time step, σ denotes the logistic sigmoid function and $\odot$ denotes element-wise multiplication. Intuitively, the forget gate controls the extent to which the previous memory cell is forgotten, the input gate controls how much information is input to each unit, and the output gate controls the exposure of the internal memory state. The hidden state vector in an LSTM unit is therefore a gated, partial view of the state of the unit's internal memory cell. Since the value of the gating variables vary for each vector element, the model can learn to represent information over multiple time scales.

3.2 Re-read LSTM (rLSTM)

In the textual entailment recognition problem, we need to model the relationship of two sentences and judge whether the premise can entail the hypothesis. As for human beings, maybe the most nature way for the judging process is first read the premise, remember it, and then read the hypothesis while thinking whether the premise can entail the hypothesis. Intuitively, we can improve the performance of RTE by adding some human nature to deep neural network models. In this paper, we intend to make the LSTM unit capable of thinking the relation between premise and hypothesis while reading them. Our proposed LSTM architecture is shown in Figure 1b, we call it re-read LSTM (rLSTM). The rLSTM unit is specially designed for dual sentence modeling, it is applied only when dealing with the second sentence. Therefore, in Figure 1b, there is a general input to rLSTM: the premise representation. This representation is composed of each word's representation in the premise. The word's representation can

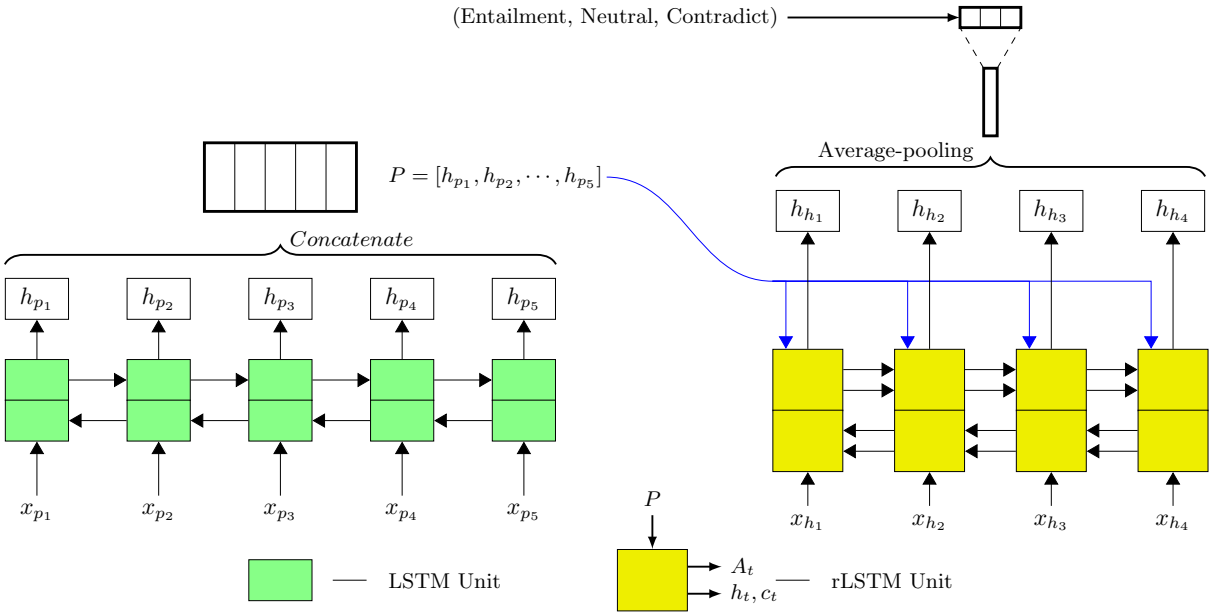


Figure 2: The architecture of our model.

be the word embedding or another plain LSTM's output. We add an input and an output to rLSTM unit which represents the attention over the words in the premise at time step $t-1$ (A_{t-1}) and $t(A_t)$. Given the attention at time $t-1$ and the premise representation P, we can get the current understanding of the premise, or the memory m_t:

$$m_t = A_{t-1}P \tag{2}$$

where $A_{t-1} \in \mathbb{R}^L$, $P \in \mathbb{R}^{d \times L}$, L represents the number of words in the premise.

The model needs to consider the entailment relationship. Therefore, the memory of the premise should also affect the hidden state of the hypothesis. With the information of premise, the hidden state of the hypothesis can learn more specific information for their relationship. So we add the memory of the premise and the memory cell to affect the hidden state vector as follows:

$$h_t = o_t \odot \Big(\tanh(c_t) + \tanh(m_t) \Big) \tag{3}$$

Intuitively, each time the model reads one more word in the hypothesis, it should be clearer about what information in the premise is more important respected to the hypothesis. So the A_{t-1} is the premise's attention in time step $t-1$, after read one word, it became A_t, the information represented by which is more focus on the relationship between the premise and the hypothesis. Therefore, the origin LSTM's memory cell can affect the attention in time step t:

$$\begin{aligned} \alpha &= W_p P + W_m m_t + W_c C_{t-1} \\ A_t &= softmax(w\alpha) \end{aligned} \tag{4}$$

where $W_p \in \mathbb{R}^{d \times d}$, $W_m \in \mathbb{R}^{d \times d}$, $W_c \in \mathbb{R}^{d \times d}$, $w \in \mathbb{R}^d$ are weight matrices or vectors.

3.3 Our Model

Our model is shown in Figure 2. We use the standard LSTM to deal with the premise. The output of the standard LSTM $h_{p_1}, h_{p_2}, \cdots, h_{p_n}$ are concatenated as a matrix P:

$$P = [h_{p_1}, h_{p_2}, \cdots, h_{p_L}] \tag{5}$$

where $P \in \mathbb{R}^{d \times n}$, L represents the number of words in the premise.

	Train	Dev	Test
Entailment	183416	3329	3368
Neutral	182764	3235	3219
Contradict	183187	3278	3237
Total	549367	9842	9824

Table 2: Distribution of Entailment Classes in SNLI

And then, we take P as the general input of re-read LSTM (rLSTM), which is used to deal with the hypothesis as is shown in Formula 6.

$$\begin{aligned} h_{h_i}^{\rightarrow} &= rLSTM(P, A_{t-1}, h_{h_{i-1}}, c_{t-1}) \\ h_{h_i}^{\leftarrow} &= rLSTM(P, A_{t+1}, h_{h_{i+1}}, c_{t+1}) \\ h_{h_i} &= [h_{h_i}^{\rightarrow}, h_{h_i}^{\leftarrow}] \end{aligned} \tag{6}$$

where the forward output $h_{h_i}^{\rightarrow}$ and the backward output $h_{h_i}^{\leftarrow}$ are all calculated by re-read LSTM unit. We concatenate the forward output $h_{h_i}^{\rightarrow}$ and the backward output $h_{h_i}^{\leftarrow}$ as the final output of the bidirectional rLSTM: h_{h_i}.

Then, we average the outputs of rLSTM as the final representation of the premise-hypothesis sentence pair:

$$S = \frac{1}{n}\sum_{i=1}^{n} h_{h_i} \tag{7}$$

where $S \in \mathbb{R}^d$ is the final representation. Then S is fed into a logistic classifier:

$$O = WS + b \tag{8}$$

where $W \in \mathbb{R}^{3\times d}$, $b \in \mathbb{R}^3$ are the parameters, $O \in \mathbb{R}^3$ is the final output of this premise-hypothesis pair, each entry contains the score of an entailment class (entailment, neutral, contradiction).

3.4 Training

We define the ground-truth label vector y for each premise-hypothesis pair as a binary vector. If this premise-hypothesis pair belongs to class i, only the i-th dimension $y(i)$ is 1 and the other dimensions are set to 0. In our model, the RTE task is classification problem and we adopt cross entropy loss as the objective function. Given the parameters set $\theta = \{\theta_{LSTM}, \theta_{rLSTM}, W, b\}$, where θ_{LSTM} represents the LSTM neural network parameters, θ_{rLSTM} represents the rLSTM neural network parameters. the objective function for a premise-hypothesis pair can be written as,

$$J(\theta) = -\sum_{i} y(i)\log(O(i)) + \frac{\lambda}{2}\|\theta\|^2 \tag{9}$$

To compute the network parameter θ, we maximize the log likelihood $J(\theta)$ through stochastic gradient descent over shuffled mini-batches with the Adadelta (Zeiler, 2012) update rule.

4 Experiment

In this section, we present the evaluation of our model. We first perform quantitative evaluation, comparing our model with previous works. We then conduct some qualitative analyses to understand how our rLSTM model works in matching the premise and the hypothesis.

4.1 Datasets and Model Configuration

We conduct experiments on the Stanford Natural Language Inference corpus (SNLI) (Bowman et al., 2015). The original data set contains 570,152 sentence pairs, each labeled with one of the following relationships: entailment, contradiction, neutral and $-$, where $-$ indicates a lack of consensus from the

human annotators. We discard the sentence pairs labeled with $-$ and keep the remaining ones for our experiments. Table 2 summarizes the statistics of the three entailment classes in SNLI.

We use 300 dimensional GloVe embeddings (Pennington et al., 2014) to represent words, which is trained on the Wikipedia+Gigaword dataset. The embeddings of unknown tokens are initialized by random vectors.

4.2 Methods for Comparison

Although we list all of the approaches designed for recognizing textural entailment task on the SNLI dataset in Table 3, we mainly want to compare our model with the word-by-word attention model by Rocktäschel et al. (2015), long short term memory network model by Cheng et al. (2016) and the decomposable attention model by Parikh et al. (2016) since they are either related to our work or achieved the state-of-the-art performance on the SNLI corpus.

We did the following ablation experiments:

- rLSTM: our final model.
- rLSTM - C info: in this model, in the rLSTM unit, the origin memory cell C_{t-1} cannot affect the next attention A_t. Then the Formula 4 is changed into Formula 10. We intend to see whether the origin memory cell's information can help the model to focus on more important information in the premise.

$$\begin{aligned} \alpha &= W_p P + W_m m_t \\ A_t &= softmax(w\alpha) \end{aligned} \tag{10}$$

- rLSTM - A info: in this model, in the rLSTM unit, the premise's memory m_t cannot affect the next hidden state h_t. Then the Formula 3 is changed into Formula 11, which is the origin formula of LSTM. We intend to see whether the current understanding of the premise can help the model to make better decision of the relation between the premise and the hypothesis.

$$h_t = o_t \odot \tanh(c_t) \tag{11}$$

- rLSTM - A&C info: in this model, in the rLSTM unit, the origin memory cell C_{t-1} cannot affect the next attention A_t and the premise's memory m_t cannot affect the next hidden state h_t. Then the LSTM part in the rLSTM is the same as the origin LSTM, and the attention part is updated each step only based on the model's understanding of the premise itself.

4.3 Quantitative Results

The experiment results are listed in Table 3. The experiment results are listed in Table 3. We can see that when we set d to 300 our final model (rLSTM) achieves an accuracy of 87.5% on the test data, which to the best of our knowledge is the highest on this data set. When we remove the effect of the origin memory cell C_{t-1} on the next attention A_t (rLSTM $-$ C info), we found that the performance has dropped 2.7 percent. This phenomenon shows that the memory of the hypothesis can indeed contribute to a better understanding of the premise. When we remove the effect of the premise's memory m_t on the next hidden state h_t, the performance dropped again. That means the premise's memory can indeed help understand the hypothesis.

When comparing our rLSTM model with Rocktäschel et al. (2015)'s LSTM word-by-word attention model under the same setting with $d = 100$, we can see that our performance on the test dataset is still higher than that of Rocktäschel et al. (2015)'s (**student t-test,** $p < 0.05$). One advantage of our model compared to Rocktäschel et al. (2015)'s model is that our attention is inside the LSTM unit, so that it can be affected by the inner memory of LSTM and can also affect the LSTM's inner state. This mechanic can provide our model more flexibility to achieve a better result.

Our result has also outperformed Cheng et al. (2016)'s 300d LSTMN model (**student t-test,** $p < 0.05$). LSTMN tends to use memory network to remember the sentence while making intra-attention (within a sentence) and inter-attention (between two sentences) complement each other to get good results.

Feature-based models				
Publication	Model	Params	Train	Test
Bowman et al. (2015)	Unlexicalized features	-	49.4	50.4
Bowman et al. (2015)	+ Unigram and bigram features	-	99.7	78.2
Sentence encoding-based models				
Bowman et al. (2015)	100D LSTM encoders	221k	84.8	77.6
Bowman et al. (2016)	300D LSTM encoders	3.0m	83.9	80.6
Vendrov et al. (2015)	1024D GRU encoders w/ unsupervised 'skip-thoughts' pre-training	15m	98.8	81.4
Mou et al. (2016)	300D Tree-based CNN encoders	3.5m	83.3	82.1
Bowman et al. (2016)	300D SPINN-PI encoders	3.7m	89.2	83.2
Liu et al. (2016b)	600D (300+300) BiLSTM encoders	2.0m	86.4	83.3
Liu et al. (2016b)	600D (300+300) BiLSTM encoders with intra-attention and symbolic preproc.	2.8m	85.9	85.0
Other neural network models				
Rocktäschel et al. (2015)	100D LSTMs w/ word-by-word attention	252k	85.3	83.5
Wang and Jiang (2015)	300D mLSTM word-by-word attention model	1.9m	92.0	86.1
Cheng et al. (2016)	300D LSTMN with deep attention fusion	1.7m	87.3	85.7
Cheng et al. (2016)	450D LSTMN with deep attention fusion	3.4m	88.5	86.3
Parikh et al. (2016)	200D decomposable attention model	382k	89.5	86.3
Parikh et al. (2016)	200D decomposable attention model with intra-sentence attention	582k	90.5	86.8
Re-read LSTM models				
This paper	300D rLSTM	2.0m	90.7	**87.5**
This paper	300D rLSTM - C info	1.9m	88.9	84.8
This paper	300D rLSTM - A info	2.0m	90.2	87.3
This paper	300D rLSTM - A&C info	1.9m	88.6	84.7

Table 3: Train/test accuracies on the SNLI dataset and number of parameters (excluding embeddings) for each approach.

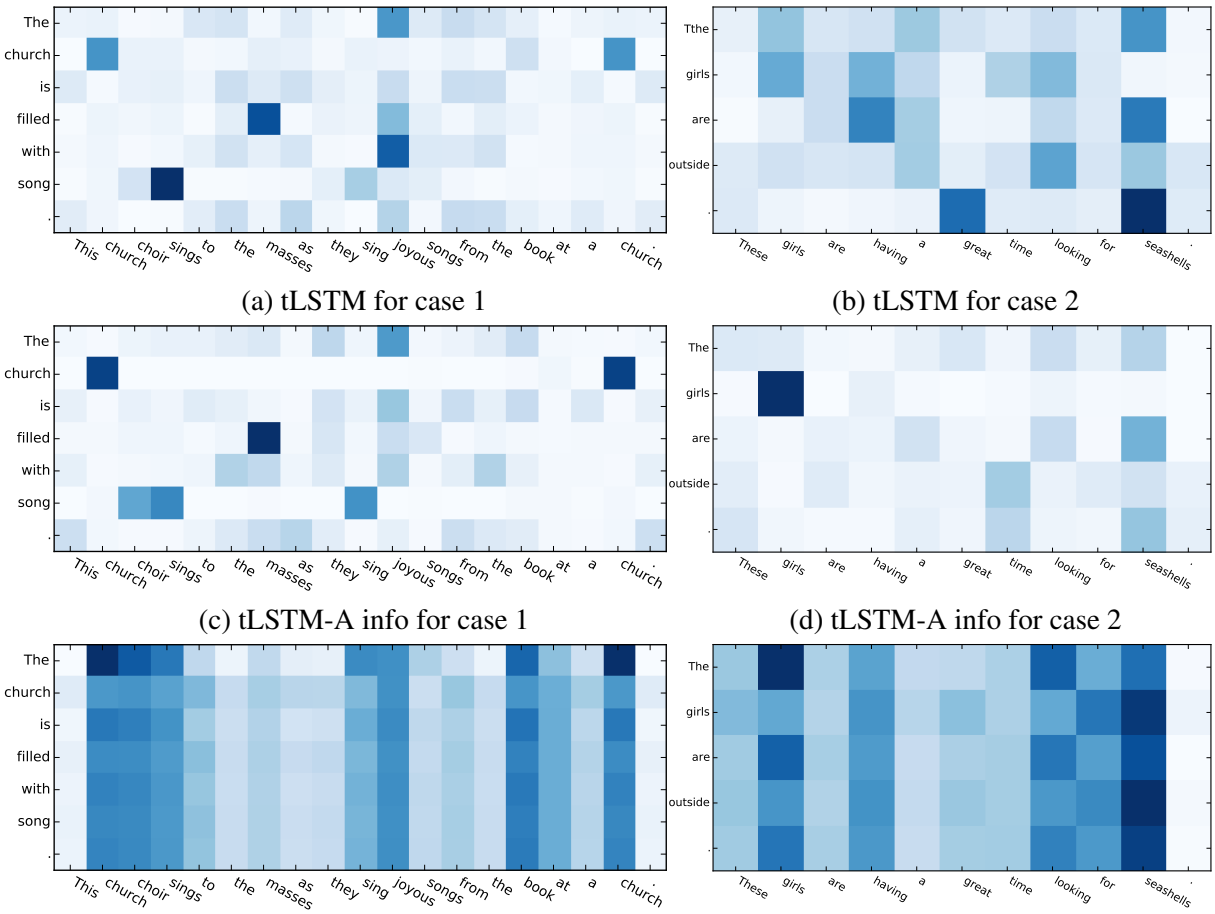
However, the intra-attention is generated just according to the information of the current sentence itself and the inter-attention focus on generating a better alignment of the two sentences without considering a better understanding of the sentence relationship. Our model intends to better understand the relation between two sentences by making the two sentence's information affect each other, which helps our model achieves good result.

4.4 Qualitative Analyses

Figure 3 listed the visualizations of the attention changing process. Figure 3a, Figure 3c, Figure 3e are the visualization of the entailment case in Table 1. Figure 3b, Figure 3d, Figure 3f are the visualization of the case in Section 1.

In Figure 3a, we can see that as the tLSTM is dealing with the hypothesis, the model pays more attention on the real meaning of the two sentences instead of simple word alignment. For example, when dealing with the word "song" in the hypothesis, the model focus more on the word "sing" in the main clause instead of the word "sing" in the subordinate clause (Figure 3c). Figure 3b has the similar phenomenon. After dealing with "outside", the model tend to focus on the words related to "outside" such as "seashells". Instead, in Figure 3d, after remove the "A info", the model cannot focus on useful information any more.

In addition, when we stop allowing the hypothesis's memory to affect the attention of the premise,



(a) tLSTM for case 1

(b) tLSTM for case 2

(c) tLSTM-A info for case 1

(d) tLSTM-A info for case 2

(e) tLSTM-C info for case 1

(f) tLSTM-C info for case 2

Figure 3: The attention visualization analysis of tLSTM, tLSTM-A info and tLSTM-C info. The premise is on the x axis, the hypothesis is on the y axis.

in Figure 3e and Figure 3f, the model's attention performs very poor. Therefore, the test accuracy of tLSTM−C info is much lower than tLSTM.

5 Conclusion

In this paper, we proposed a special LSTM architecture for the task of textural entailment recognizing. Inspired by the process of human reading, we bring re-read mechanic into the LSTM unit and call it re-read LSTM (rLSTM). Re-read LSTM is specially designed for dual sentence modeling and it takes the representation of the premise as general input when dealing with the hypothesis. There are two main differences between rLSTM and LSTM. First, in rLSTM, the memory of the hypothesis can affect the attention of the premise, which means the hypothesis can help the model better understanding the premise. Second, the premise's memory can affect the hidden state of the hypothesis, which means the premise can provide useful information to help the model make better decision of the relation between the premise and the hypothesis. And then, we designed an architecture for the RTE task, we use a traditional bidirectional LSTM to deal with the premise and use bidirectional rLSTM to deal with the hypothesis. Finally, the average of the bidirectional rLSTM's outputs can be used for predicting the relationship between the premise and the hypothesis.

Experiments on the SNLI corpus showed that the rLSTM model outperformed the state-of-the-art performance reported so far on this data set. Moreover, closer analyses on the attention vectors revealed that our rLSTM mechanics indeed provide and generate better attention on the premise, which represents a better understanding of the sentences.

Acknowledgements

We would like to thank our three anonymous reviewers for their helpful advice on various aspects of this work. This research was supported by the National Key Basic Research Program of China (No.2014CB340504) and the National Natural Science Foundation of China (No.61375074,61273318). The contact authors for this paper is Baobao Chang and Zhifang Sui.

References

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Deep semantics with probabilistic logical form. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (* SEM-13)*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. *arXiv preprint arXiv:1603.06021*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Georgiana Dinu and Rui Wang. 2009. Inference rules and their application to recognizing textual entailment. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 211–219. Association for Computational Linguistics.

Andrew Hickl. 2008. Using discourse commitments to recognize textual entailment. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 337–344. Association for Computational Linguistics.

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

Valentin Jijkoun and Maarten de Rijke. 2005. Recognizing textual entailment using lexical similarity. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 73–76.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016a. Modelling interaction of sentence pair with coupled-lstms. *arXiv preprint arXiv:1605.05573*.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016b. Learning natural language inference using bidirectional lstm model and inner-attention. *arXiv preprint arXiv:1605.09090*.

Prodromos Malakasiotis and Ion Androutsopoulos. 2007. Learning textual entailment using svms and string similarity measures. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 42–47. Association for Computational Linguistics.

Prodromos Malakasiotis. 2011. *Paraphrase and Textual Entailment Recognition and Generation*. Ph.D. thesis, Ph. D. thesis, Department of Informatics, Athens University of Economics and Business, Greece.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 2016 Conference on Association for Computational Linguistics*, Lisbon, Portugal, August. Association for Computational Linguistics.

Rodney D Nielsen, Wayne Ward, and James H Martin. 2009. Recognizing entailment in intelligent tutoring systems. *Natural Language Engineering*, 15(04):479–501.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Miguel Rios, Lucia Specia, Alexander Gelbukh, and Ruslan Mitkov. 2014. Statistical relational learning to recognise textual entailment. In *Computational Linguistics and Intelligent Text Processing*, pages 330–339. Springer.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Lei Sha, Sujian Li, Baobao Chang, Zhifang Sui, and Tingsong Jiang. 2015. Recognizing textual entailment using probabilistic inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1620–1625, Lisbon, Portugal, September. Association for Computational Linguistics.

Eyal Shnarch, Jacob Goldberger, and Ido Dagan. 2011a. A probabilistic modeling framework for lexical entailment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 558–563. Association for Computational Linguistics.

Eyal Shnarch, Jacob Goldberger, and Ido Dagan. 2011b. Towards a probabilistic model for lexical entailment. In *Proceedings of the TextInfer 2011 Workshop on Textual Entailment*, pages 10–19. Association for Computational Linguistics.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2015. Order-embeddings of images and language. *arXiv preprint arXiv:1511.06361*.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2006. Using dependency-based features to take the para-farce out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop*, volume 2006.

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*.

Rui Wang and Günter Neumann. 2007. Recognizing textual entailment using sentence similarity based on dependency tree skeletons. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 36–41. Association for Computational Linguistics.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Fabio Massimo Zanzotto and Alessandro Moschitti. 2006. Automatic learning of textual entailments with cross-pair similarities. In *International conference on computational linguistics and the 44th Annual meeting of the Association for computational linguistics*, volume 1, pages 401–408. Association for Computational Linguistics.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

A Paraphrase and Semantic Similarity Detection System for User Generated Short-Text Content on Microblogs

Kuntal Dey
IBM Research India
New Delhi, India
kuntadey
@in.ibm.com

Ritvik Shrivastava
NSIT Delhi
New Delhi, India
ritviks.it
@nsit.net.in

Saroj Kaushik
IIT Delhi
New Delhi, India
saroj
@cse.iitd.ac.in

Abstract

Existing systems deliver high accuracy and F1-scores for detecting paraphrase and semantic similarity on traditional clean-text corpus. For instance, on the clean-text Microsoft Paraphrase benchmark database, the existing systems attain an accuracy as high as 0.8596. However, existing systems for detecting paraphrases and semantic similarity on user-generated short-text content on microblogs such as Twitter, comprising of noisy and ad hoc short-text, needs significant research attention. In this paper, we propose a machine learning based approach towards this. We propose a set of features that, although well-known in the NLP literature for solving other problems, have not been explored for detecting paraphrase or semantic similarity, on noisy user-generated short-text data such as Twitter. We apply support vector machine (SVM) based learning. We use the benchmark Twitter paraphrase data, released as a part of SemEval 2015, for experiments. Our system delivers a paraphrase detection F1-score of 0.717 and semantic similarity detection F1-score of 0.741, thereby significantly outperforming the existing systems, that deliver F1-scores of 0.696 and 0.724 for the two problems respectively. Our features also allow us to obtain a rank among the top-10, when trained on the Microsoft Paraphrase corpus and tested on the corresponding test data, thereby empirically establishing our approach as ubiquitous across the different paraphrase detection databases.

1 Introduction

Detecting paraphrase, and more specifically, given a pair of input natural language texts identifying whether one is a paraphrase of the other, has been a challenging research problem. Over a number of years and volumes of research, the paraphrase detection systems have emerged as robust and well-performing ones, especially for "clean text corpus", such as the Microsoft Paraphrase Corpus (Dolan et al., 2004). Semantic similarity detection has been another related problem of interest, where the objective is to identify how similar is one text with respect to another (Harispe et al., 2015). Since, paraphrases are expected to have significant semantic similarity (Corley and Mihalcea, 2005) (Mihalcea et al., 2006), systems performing well for paraphrase detection can also be intuitively expected to perform well for semantic similarity detection of a pair of text inputs. It is worth noting that, for benchmark "clean text corpus" data such as the Microsoft Paraphrase Corpus database, the paraphrase detection systems deliver an impressive performance, with a high F-score of 0.8596.

While the literature around paraphrase and semantic similarity detection has matured along the directions of clean text corpus, much work remains to be done to attain commendable performances in the paradigm of noisy short-text inputs, such as user-generated short-text content found on Twitter. Some initial work has been recently carried out on the benchmark SemEval 2015 data (Xu, 2014) (Xu et al., 2015); however, the performance of the systems developed till date leave much desired. Detecting paraphrases and semantic similarity on such text is inherently difficult; however, we believe there is scope for improvement over the current 0.724 performance, that the current state of the art delivers (Xu et al., 2014). Given the well-understood importance of paraphrase detection in multiple fields of NLP, such as

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2880–2890, Osaka, Japan, December 11-17 2016.

review summarization, opinion mining and information matching, to name a few, and practical applications such as first-story detection (Petrović et al., 2012), it is of paramount importance to develop highly accurate paraphrase detectors for such user-generated noisy short-text corpus.

This motivates us to attempt to improve the performance of detecting paraphrases and semantic similarity on user-generated short-text on noisy platforms such as Twitter. We propose a feature-set driven approach, and use support vector machine (SVM) based machine learning. We rely upon lexical, syntactic, semantic and pragmatic features of a given pair of tweets, to label whether these two tweets are paraphrases of each other. We further obtain the semantic similarity scores of the pairs, and thereby assign semantic similarity labels. We validate our work against the benchmark SemEval 2015 data, and find our system (F1-score 0.741) to outperform the known state of the art (best known F1-score 0.724), that includes all of the feature-based approaches as well as deep-learning based approaches.

The main contributions of our work are the following.

- We provide a machine learning based model, and a set of lexical, syntactic, semantic and pragmatic features, for detecting paraphrases on user-generated noisy Twitter short-text content data.
- We empirically show the goodness of the proposed features, on a benchmark Twitter paraphrase corpus. Our system outperforms all the systems that exist in the state of the art.
- We further demonstrate the effectiveness of our feature set on benchmark clean-corpus text data, namely Microsoft Paraphrase dataset, and obtain a rank within the top 10 existing systems, by training our system (our features) on their dataset, and testing on their dataset as well. This empirically establishes the goodness of our proposition across different types of text.

2 Related Work

The importance of detecting paraphrases for different information processing applications, has been well-accepted by researchers. Over the long-standing history of research on paraphrase detection, a volume of significant work has been carried out.

Substantial work has been carried out in the space of detecting paraphrases on traditional clean-text corpus, by several researchers. The Microsoft Paraphrase corpus (Dolan et al., 2004), that presents sentence pairs from newswire text, serves as the long-standing baseline for such text. As observed by (Ji and Eisenstein, 2013), there are three high-level approaches. (1) String similarity metrics, comprising of n-gram overlaps and BLEU features, that have been proposed by (Wan et al., 2006), and (Madnani et al., 2012). (2) Parse structure based syntactic operations, such as by (Wu, 2005), and (Das and Smith, 2009). (3) Distributional methods, such as latent semantic analysis (LSA), by (Landauer et al., 1998). (Kauchak and Barzilay, 2006) propose a distributionally similar alternative based approach. (Socher et al., 2011) propose a feature auto-encoder based approach, combining word representations, building upon recursive auto-encoding. (Blacoe and Lapata, 2012) demonstrate the effectiveness of combining latent representations with simple element-wise operations, for the purpose of identifying semantic similarity amongst larger text units. (Ji and Eisenstein, 2013) propose a discriminative term weighting metric, namely TF-KL_D to outperform TF-IDF, and show that "using the latent representation from matrix factorization as features in a classification algorithm substantially improves accuracy". They combine the latent features with fine-grained n-gram overlap features, improving the state of the art significantly.

Recently, a few initial works have been carried out for detecting paraphrases and semantic similarities on user-generated noisy social network and microblog short-text, such as Twitter. In a recent seminal paper, (Xu et al., 2014) proposed a joint word-sentence approach based model, and applied a multi-instance learning assumption (Dietterich et al., 1997) that "two sentences under the same topic are paraphrases if they contain at least one word pair that is indicative of sentential paraphrase". They observe that the "at least one anchor" overlap works well for short text situations, such as on Twitter, though they are not necessarily effective for traditional long texts, and thereby design a graphical model to discriminatively determine whether a given pair of words are paraphrases. They create a dataset (the collection of which is found in further detail in (Xu, 2014)), and empirically demonstrate their system. They obtain an F1-score

for semantic similarity of 0.724 by their system, and claim the human upper bound to have an F1-score of 0.823. While they do not report their paraphrase detection score, our experimentations using the data[1] and the corresponding released code[2] finds their system to yield an F1-score of 0.696.

Subsequently, a number of researchers attempted to solve the problem of paraphrase detection, as part of the task proposed by (Xu et al., 2015). The model by (Eyecioglu and Keller, 2015) proposes to have unigram and bigram features for characters and words, and explores the overlap (and non-overlap) of these features for determining whether one tweet is a paraphrase of the other. While the authors do not address the semantic similarity detection problem, the paraphrase detection problem yields an F1-score of 0.674. In another work applied on the same data, (Zarrella et al., 2015) obtain a paraphrase detection F1-score of 0.667, and a semantic similarity score of 0.724. They use "mixtures of string-matching metrics, tweet-specific distributed word representation alignments, recurrent neural networks to model similarity between those alignments, and distance measurements on pooled latent semantic features", and tie these systems together using logistic regression. In addition to using string based, corpus based and syntactic features, (Zhao and Lan, 2015) propose "novel features based on distributed word representations, learned using deep learning paradigms". They attained an F1-score of 0.662 for paraphrase detection on the given dataset.

As apparent from the above discussion, the state of the art for paraphrase detection on noisy short-text can potentially be significantly improved. Different classes of features, with different implications, need to be applied for making such improvement. Our work addresses this, and also attempts to explore the applicability of the proposed model across other, clean-text corpus. Our system, with its feature set that have not been used on noisy Twitter data before, outperforms the state of the art systems for both paraphrase detection as well as semantic similarity identification.

3 Methodology

We choose a machine-learning based approach to solve the problem at hand. We identify several features of different types, and apply Support Vector Machine (SVM) based learning.

3.1 Preprocessing

While by itself we do not use any *preprocessing feature* per se, we carry out preprocessing steps that impact the performance of our overall system. The preprocessing steps include the following.

- **Topic phrase removal:** We remove the topic phrase from the tweet pairs, as this does not provide any information to distinguish given tweet pairs, but causes unnecessary gram overlaps. To illustrate with an example, in the test dataset, a given pair of texts are: *"Which Star Wars episode should I watch"* and *"I have so much of a life that Im at home watching Star Wars"*. The topic given here, explicitly noted as part of the dataset, is *Star Wars*.

- **Tweet normalization using net slang and Han-Baldwin dictionary:** We normalize the tweets, using net slang and Han-Baldwin normalization dictionary knowledge (Han and Baldwin, 2011). This would help resolve non-dictionary colloquial expressions. E.g.: *aaf* maps to *as a friend*. We use an online version of one of the many net slang dictionaries that are available today. For experiments, we use an online net slang dictionary.

- **Named Entity (NE) boundary correction:** We align NE boundaries across tweets, for appropriate matching. For example, for a given pair of tweets having the same topic, words like *Colorado Ravens* and *Ravens* probably convey the same concept, but would create different gram features. For experimentation, we use the NE tags provided in the dataset, in the CoNLL IOB format.

- **NE tag cleaning:** Inconsistent capitalization of tweets lead to named entity recognition (NER) errors, jeopardizing system performance. For example, if within a given pair of tweets with the

[1] https://github.com/cocoxu/SemEval-PIT2015, crawled on July 10, 2016
[2] https://github.com/cocoxu/multip, crawled on July 10, 2016

same topic, one tweet consists of a named entity *Colorado Ravens* and the other one has the word *ravens* with a NE-tag **O**, they are likely to be discussing the same Raven, with the inconsistency of capitalization confusing the named entity recognizer. We clean NE tags by cross-checking the given pair of tweets, and matching unigrams and bigrams in a case-insensitive manner.

- **Synonym and hypernym replacement using Wordnet:** We use Wordnet (Miller, 1995) to carry out a synonym and hypernym replacement process. For a pair of given tweets, we aim to unify the diverse words appearing in the pair of input texts (tweets), by replacing a word with its synonym (or hypernym). For instance, if *Tweet 1* contains the text *"He made a fast move"* and if *Tweet 2* contains the text *"He performed a quick move"*, then the content of the second tweet (*tweet 2*) gets updated to *"He performed a fast move"*, as long as "fast" and "quick" appear as synonyms in Wordnet. This in turn is fed to the main processing stream for feature selection. This replacement process proves to be useful for adjectives and verbs, so the synonym replacement process considers all the words having adjective and verb POS tags.

3.2 Feature Selection

The challenges to perform the task of paraphrase and semantic similarity detection, exist at different layers of NLP. These include several challenges at (a) lexical, (b) syntactic, (c) semantic and (d) pragmatic levels. We observe that the existing systems, that have specifically attempted to solve the paraphrase and/or semantic similarity detection problem for noisy short-text user generated content platform as Twitter, such as (Xu et al., 2014) and (Eyecioglu and Keller, 2015) *etc.*, have opportunities to be improved, by enhancing the features to solve for all these levels. With this observation, we attempt to explore features across these dimensions, as described below.

3.2.1 Lexical Features

Character-level gram features: We create character-level gram features, notionally borrowing from the features described by (Eyecioglu and Keller, 2015), and thereby include the number of overlaps of character trigrams as features. We further include the number of character trigrams in each of the two input texts (tweets), the size of union of character trigrams over the pair of input texts, the size of intersection, and the size of difference of number of character trigrams - in the form of (a) the number of trigrams that occur in the first text but not the second, and (b) those which occur in the second text but not the first - as features.
Word-level gram features: Apart from character-grams, we also construct word-level gram features, again borrowing in principle from (Eyecioglu and Keller, 2015). We construct features for word-level unigrams, bigrams and trigrams, that are similar in nature with the character gram features, in that, we consider the number of grams in each of the two input texts, and the union, intersection and difference sizes. Further, we also consider **unordered gram overlap** features for word grams, rewarding the system for unordered match in word bigrams and trigrams. This significantly helps the system performance.
Stemming: We repeat the use of the word-level gram features, after having stemmed the text (tweet content). We use the Porter stemmer (Porter, 2001) for performing the stemming in our experimentation.
Stopword removal: We perform stopword removal, using a list of well-accepted set of stopwords. We repeat the use of the word-level gram features after stopword removal.
String-level features: Stemming and Han-Baldwin normalization, done as part of the earlier steps, are two of the string-level features used by our methodology. Further, motivated by (Xu et al., 2014), we use the Jaro-Winkler string similarity (Winkler, 1999) of each given pair of input texts, as a feature.

3.2.2 Syntactic Features

Part-of-Speech (POS) agreement features: We construct two features under this subcategory, that attempt to capture the POS agreement features across the pair of input texts (tweets).

- **Number of matching words with matching POS:** We count the number of word unigrams across the pair of texts, where both the actual word as well as the POS tags of the pair of words, match with each other. We use this count as a feature. We empirically observe that the preprocessing step

of performing synonym replacement with Wordnet, significantly improves the system performance, when deployed along with this feature.

- **Verb similarity:** We count the number of verbs in each of the two texts, where the verb tag (such as VB, VBP, VBZ, VBN *etc.*) match with each other, and use this count as a feature.

Named entity (NE) features: Using the preprocessed named entities, we explore the degree of overlap of NEs across the tweet pairs, over and beyond the core topic phrase. Note that the core topic words are often named entities in our case, and hence they are removed earlier in the preprocessing phase.

3.2.3 Semantic Features

We extensively use Wordnet (Miller, 1995), to enable the construction of our semantic features.
Word overlap features: We compute the degree of overlap between words that appear across the pair of input texts, considering their semantics. Note that, this is inherently different from the word gram features computed as part of the lexical features, as the lexical features are agnostic of semantic attributes of words. Using Wordnet, we construct two features under this subcategory, as given below.

- **Best word overlap:** For each pair of words of a given POS that appear in each of the input texts (tweets), we compute the overlap (intersection) of the sets of Wordnet synonyms and hypernyms of each word of the pair, as well as the union of the synonym and hypernym set. We repeat the above for the stemmed words. If one word is a part of the synonym or hypernym set of the other word, then we count the word as a whole (with $weight = 1$), and the value of this feature becomes unity (1). Otherwise, if none of the two words is included in any of the synonym or hypernym sets of the other, then we use the sum of Jaccard coefficients of the above-computed overlap, of the combination of the given words and stemmed words, as the value of the best word overlap feature. Thus, for the four different POS tags we consider, namely noun, verb, adjective and adverb, this gives us a set of four different features.

- **Adjective overlap:** For each pair of words tagged as adjectives by the POS tagging process, we compute the overlap (intersection) of the sets of Wordnet synonyms and hypernyms of each word of the pair, as well as the union of the synonym and hypernym set. Just like in the case of finding the best word overlap feature, we repeat the above for the stemmed words. We use the sum of Jaccard coefficients of the above-computed overlap, of the combination of the given words and stemmed words, as a feature. Note that, in this feature, even if an adjective from one text appears directly as a synonym or hypernym of the other adjective from the other text, we never consider the feature value to be unity (which is captured by the best word overlap feature separately). While intuitively this feature is extremely "close" to the best word overlap feature, we still retain this feature as during the feature construction (development) process, we observe this feature to be beneficial.

Phrase overlap features: We compute the degree of direct overlap of (a) noun phrases (b) and verb phrases across tweet pairs. We also use the stemmed versions to compute stemmed phrase overlaps. Treating each input text (tweet) as a disconnected graph and each phrase as a vertex in the graph, we connect the pairs of graphs (one graph for each tweet) using the best matching phrase. The matching of a pair of phrases is carried out, by computing the overlap (intersection) of the sets of Wordnet synonyms and hypernyms of the individual words that appear in the phrase pair. Similar to the case of the best word overlap feature, we set the value of this feature of unity if, any one word belonging to one phrase, is directly included in the set union of synonyms and hypernyms of any one word belonging to the other phrase. Otherwise, we compute the sum of Jaccard coefficients of the above-computed overlap for all the word pairs across the text (tweet) pairs, and use this sum as the value of the feature. This is constructed for noun and verb phrase types, thereby adding two features to the overall set of features. Effectively, this is a manifestation of MAXSIM (Chan and Ng, 2008), that is used in the machine translation paradigm.

3.2.4 Pragmatic Features

Subjectivity/objectivity agreement feature: In one implementation, we include the agreement of the nature of the tweet pair with respect to subjectivity/objectivity, as a boolean feature. We make use of the MPQA dictionary, that indicates strong and weak subjectivity and objectivity of words, to construct this feature. The final experimental results we present for the Twitter, does not include this feature, when we attain an F1-score of 0.741; however, if we compromise our F1-score on the Twitter noisy short-text data to 0.740, we obtain a marginal lift from 0.824 to 0.825 on the Microsoft Paraphrase data, helping our system outperform other systems that also yield an F1-score of 0.824 on the Microsoft Paraphrase data.
Close attachment of negations: We also perform close attachment of negations, by attaching the semantic sense of a negation (not *etc.*) with an appropriate term, using well-known techniques from the literature. This in turn assigns appropriate polarity to the subjectivity features.

4 Experiments

In this section, we provide the details of the experiments we conducted.

4.1 Data Description and Tools Used

For experimentation, we select the benchmark dataset by (Xu, 2014) provided as part of the SemEval 2015 task (Xu et al., 2015). This dataset has been used by all the recent works in this space, and further, existing paraphrase detection algorithms, developed optimizing for clean corpus and tested with other clean-text datasets such as the Microsoft Paraphrase database, have also been tested on this dataset. Thus, it provides a well-tested foundation for empirically observing the performance of our system, and benchmarking the performance of our system against the other existing systems.

The original dataset of (Xu et al., 2014) is shown on Table 1. Note that, the dataset based on which the final scores are reported by (Xu et al., 2014), which is also released as part of SemEval 2015 (Xu et al., 2015), consists of 838 tweets in the test set, as opposed to the 972 that were present in the original dataset. This is arrived at, by ignoring the 134 "debatable" entries, that were marked in (Xu et al., 2015), All the systems that use this dataset to report their scores, are against the modified test dataset with 838 test entries ignoring the "debatable" ones, and our reports are also based upon this modified dataset. We used Weka (Hall et al., 2009) for machine learning, and used the POS and NE tags already provided by the SemEval 2015 (Xu et al., 2015) dataset.

	Num. Unique Sentences	**Num. Pair of Sentences**	**Num. Paraphrases**	**Num. Non-Paraphrases**	**Num. Debatable**
Train	$13,231$	$13,063$	$3,996$ (30.6%)	$7,534$ (57.7%)	$1,533$ (11.7%)
Dev	$4,772$	$4,727$	$1,470$ (31.1%)	$2,672$ (56.5%)	585 (12.4%)
Test	$1,295$	972	175 (18.0%)	663 (68.2%)	134 (13.8%)

Table 1: The benchmark dataset released by SemEval 2015 (Xu et al., 2015).

4.2 Performance of Our System

As detailed in Section 3, we train our SVM model on the training data provided, using the identified features. After identifying the effective features, we train our final model on the combination of training and development data provided, and subsequently execute the model to benchmark our system on the given test data. The baseline features comprise of word unigrams and bigrams, and Jaro-Winkler string similarity features. The performance, as well as the impact of each feature (as observed by feature ablation tests) of our system, are provided in Table 2.

Some example cases have been illustrated in Table 3, including cases of correct and incorrect detections by our system. One glance at the examples make it obvious that, in most cases, it is challenging to label the text pairs as paraphrases versus otherwise. Thus, these examples not only illustrate instances of successes and failures of our system, but also provides an instinctive view of the magnitude of challenge, that a solution to the current problem needs to overcome.

Features Used in Model	Semantic Similarity			Paraphrase		
	F1	P	R	F1	P	R
Baseline	0.641	0.667	0.617	0.523	0.67	0.429
+Trigrams	0.663	0.701	0.629	0.533	0.675	0.44
+Topic Removal	0.708	0.739	0.68	0.705	0.707	0.703
+WordNet Synonym Replacement	0.718	0.747	0.691	0.709	0.683	0.737
+POS features	0.721	0.734	0.709	0.709	0.683	0.737
+Phrase Overlap features	0.735	0.758	0.714	0.717	0.697	0.737
+NE features (Final Model)	**0.741**	0.756	0.726	**0.717**	0.697	0.737

Table 2: Performance of our system and impact of features (based upon feature ablation tests). F1 ← F1-score. P ← Precision. R ← Recall.

4.3 Comparing with existing systems

We compare the performance of our system with the existing systems. (Xu et al., 2014) summarize the performances of prior existing paraphrase and semantic similarity detection systems, trained on clean-text corpus, when tested on noisy user-generated short-text data on Twitter. We further compare our results with the existing systems in the literature that are specifically trained on the Twitter paraphrase corpus, and benchmark our performance. Table 4 provides a summary of the results that are attained by the existing systems, trained and tested on the same (SemEval 2015) dataset as ours, as well as our by our methodology. Clearly, our system outperforms all the others in terms of the F1-score it achieves. Although (Zhao and Lan, 2015) obtain a higher precision and (Zarrella et al., 2015) obtain a higher recall compared to ours, our system ranks #2 in terms of precision as well as in terms of recall, and #1 when the precision and recall are combined to obtain an F1-score.

4.4 Applying our feature set on clean-corpus paraphrase detection text

In order to obtain empirical insights about the goodness of the proposed feature set, for the task of paraphrase detection on clean-corpus data, we port the features to the Microsoft Paraphrase dataset (Dolan et al., 2004). Performing training on the Microsoft Paraphrase dataset with our features, and testing on the corresponding test dataset, we obtain an F1-score of 0.824. We further note that, including the subjectivity feature described in Section 3.2.4, lifts the score to 0.825 on this dataset. According to the ACL wiki for paraphrase identification [3], and also factoring for (Eyecioglu and Keller, 2015) that delivers a higher performance on the Microsoft Paraphrase database[4] compared to our system, this gives us the 10^{th} rank overall for the clean-corpus text, using no additional feature. We note the following.

- The best known system, designed by (Ji and Eisenstein, 2013), currently delivers an F1-score of 0.8596 on the Microsoft Paraphrase dataset, while ours delivers 0.825, without any adaptation. On the other hand, our system at an F1-score of 0.741, delivers a significantly better performance compared to their system with an F1-score of 0.641 on the Twitter data, also without any adaptation. Clearly, our system performs (adapts) "better" when observed in a cross-data-type (clean and noisy text) scenario, compared to theirs. This argument holds for all the other existing systems as well, including (Eyecioglu and Keller, 2015), that delivers an F1-score of 0.674 on the noisy Twitter data and an F1-score of 0.828 on the clean Microsoft Paraphrase data.

- Topic removal, which plays a major in helping our system deliver its high performance (as clear from Table 2), cannot be performed on the Microsoft Paraphrase corpus, since, unlike the Twitter corpus, the topics are not explicitly given, and topic-detection is not a focus of the current work. Detecting and removing topics from the Microsoft Paraphrase dataset, would completely unify the

[3] $http://aclweb.org/aclwiki/index.php?title = Paraphrase\_Identification\_(State\_of\_the\_art)$, crawled on July 10, 2016

[4] Yet to be added to the ACL wiki, as found at the time of writing this paper

Sentence Pair from Twitter	Gold Label	System Label	Confidence	Remark
And the Mets and Marlins go 20 Miami Marlins beat the NY Mets 21 in 20 innings	Paraphrase	Paraphrase	0.8971	Correct
Thank you very much Rafa Benitez Why do liverpool fans love benitez so much	Not Paraphrase	Paraphrase	0.6192	Incorrect
chris davis is 44 with two bombs Chris Davis has 2 home runs tonight	Paraphrase	Not Paraphrase	0.2318	Incorrect
Which Star Wars episode should I watch I have so much of a life that Im at home watching Star Wars	Not Paraphrase	Not Paraphrase	0.2317	Correct
So Roberto Mancini has been officially sacked as Man City s manager UK football manager Roberto Mancini sacked by Manchester City	Paraphrase	Paraphrase	0.9791	Correct
I wanna see the movie after earth NOW YOU SEE ME and AFTER EARTH Cant Outpace FAST FURIOUS 6	Not Paraphrase	Paraphrase	0.6180	Incorrect
Classy gesture by the Mets for Mariano real class shown by The Mets Mo Rivera is a legend	Paraphrase	Not Paraphrase	0.2316	Incorrect
Anyone wanting to go to the JT concert 722 in Chicago just watched season finale of chicago fire and cried	Not Paraphrase	Not Paraphrase	0.2312	Correct

Table 3: Examples of outputs of our system, and its correctness with respect to gold labels. "Confidence" denotes the confidence score, between 0 and 1, of a given pair of tweets being paraphrases.

feature set across the two types of data (clean and noisy), and also might potentially improve the system performance (open for exploration). We propose to explore this in future.

5 Discussion

Choice of Approach: Why not Deep Learning?
We use the traditional methodology of identifying features, and a traditional SVM classifier. The choice of modeling in the given manner is, in the known state of the art, traditional feature-based systems, such as (Xu et al., 2014) and (Eyecioglu and Keller, 2015), have so far outperformed deep learning systems, such as the recurrent neural network based approaches by (Zarrella et al., 2015) and (Zhao and Lan, 2015), for paraphrase detection. A deeper subsequent study by (Sanborn and Skryzalin, 2015) also models deep neural learning systems, and observes a similar shortcoming of such systems. A likely cause of this anomaly is the simple absence of a sufficiently large paraphrase dataset in the domain of user-generated noisy texts on microblogs such as Twitter. As and when a much-larger noisy short-text paraphrase corpus for user generated content on microblogs, such as Twitter, becomes available, a deep-learning model will be likely to deliver stronger performances, and will require a revisit.

The Human Upper Bound and the Practical Improvement Delivered by Our System
We further note that, (Xu et al., 2014) observe the human upper bound for detecting paraphrases (computed as semantic similarity, as per the numerical values they report and the corresponding program code they make available) on noisy short-text of Twitter to be limited to 0.823. Given this human upper bound, improving the F1-score for semantic similarity from the reported 0.724 to the current

Method	F1	P	R	Rank		
				F1-Based	P-Based	R-Based
Random	0.294	0.208	0.500	10	10	10
WTMF (Guo and Diab, 2012)	0.583	0.525	0.655	9	9	5
LR (Das and Smith, 2009)	0.630	0.629	0.632	8	7	6
LEXLATENT (Xu et al., 2014)	0.641	0.663	0.621	7	6	8
LEXDISCRIM (Ji and Eisenstein, 2013)	0.645	0.664	0.628	6	5	7
ENCU (Zhao and Lan, 2015)	0.662	**0.767**	0.583	5	1	9
MITRE (Zarrella et al., 2015)	0.667	0.569	**0.806**	4	8	1
ASOBEK (Eyecioglu and Keller, 2015)	0.674	0.680	0.669	3	3	4
MultiP (Xu et al., 2014)	0.724	0.722	0.726	2	3	2
Our Methodology	**0.741**	0.756	0.726	**1**	2	2

Table 4: A comparison of the performance of different systems on the given Twitter data. F1 ← F1-score. P ← Precision. R ← Recall. Note: a part of the table has been reprinted from (Xu et al., 2014).

0.741 delivers an effective improvement of not $(0.741 - 0.724) * 100\% = 1.7\%$, but a much-higher $((0.741 - 0.724)/0.823) * 100\% = 2.066\%$. This enhances the significance of our work.

6 Conclusion

In this work, we proposed a rich feature set, and performed simplistic SVM-based learning, for performing paraphrase and semantic similarity detection on user generated noisy short-text on social microblogs such as Twitter. We demonstrated the goodness of our system on the benchmark SemEval 2015 database, obtaining an F1-score of 0.717 for paraphrase detection where the known state of the art attains 0.696, and an F1-score of 0.741 for semantic similarity detection where the known state of the art attains 0.724, thus outperforming all the known systems. We show our system to also work well on the benchmark Microsoft Paraphrase database for clean text, and thereby empirically establish our approach as the most ubiquitous system available today, across the different types of paraphrase detection datasets (noisy and clean-text). Our system can be used on applications that need to identify paraphrases and semantic similarities, such as social information spread modeling.

References

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics.

Yee Seng Chan and Hwee Tou Ng. 2008. Maxsim: A maximum similarity metric for machine translation evaluation. In *ACL*, pages 55–62. Citeseer.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment*, pages 13–18. Association for Computational Linguistics.

Dipanjan Das and Noah A Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 468–476. Association for Computational Linguistics.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1):31–71.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics.

Asli Eyecioglu and Bill Keller. 2015. Asobek: Twitter paraphrase identification with simple overlap features and svms. *Proceedings of SemEval*.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 864–872. Association for Computational Linguistics.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 368–378. Association for Computational Linguistics.

Sebastien Harispe, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain. 2015. Semantic similarity from natural language and ontology analysis. *Synthesis Lectures on Human Language Technologies*, 8(1):1–254.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *EMNLP*, pages 891–896.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 455–462. Association for Computational Linguistics.

Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190. Association for Computational Linguistics.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2012. Using paraphrases for improving first story detection in news and twitter. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 338–346. Association for Computational Linguistics.

Martin F Porter. 2001. Snowball: A language for stemming algorithms.

Adrian Sanborn and Jacek Skryzalin. 2015. Deep learning for semantic similarity. *CS224d: Deep Learning for Natural Language Processing. Stanford, CA, USA: Stanford University*.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2006. Using dependency-based features to take the para-farce out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop*, volume 2006.

William E Winkler. 1999. The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau*. Citeseer.

Dekai Wu. 2005. Recognizing paraphrases and textual entailment using inversion transduction grammars. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 25–30. Association for Computational Linguistics.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from twitter. *Transactions of the Association for Computational Linguistics*, 2:435–448.

Wei Xu, Chris Callison-Burch, and William B Dolan. 2015. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). *Proceedings of SemEval*.

Wei Xu. 2014. *Data-driven approaches for paraphrasing across language variations*. Ph.D. thesis, New York University.

Guido Zarrella, John Henderson, Elizabeth M Merkhofer, and Laura Strickhart. 2015. Mitre: Seven systems for semantic similarity in tweets. *Proceedings of SemEval*.

Jiang Zhao and Man Lan. 2015. Ecnu: Leveraging word embeddings to boost performance for paraphrase in twitter. *Proceedings of SemEval*, page 34.

Modeling Extractive Sentence Intersection via Subtree Entailment

Omer Levy* **Ido Dagan*** **Gabriel Stanovsky*** **Judith Eckle-Kohler†** **Iryna Gurevych†‡**

*Computer Science Department, Bar-Ilan University, Israel
†Ubiquitous Knowledge Processing Lab, Technische Universität Darmstadt, Germany
‡Ubiquitous Knowledge Processing Lab, German Institute for Educational Research, Germany

Abstract

Sentence intersection captures the semantic overlap of two texts, generalizing over paradigms such as textual entailment and semantic text similarity. Despite its modeling power, it has received little attention because it is difficult for non-experts to annotate. We analyze 200 pairs of similar sentences and identify several underlying properties of sentence intersection. We leverage these insights to design an algorithm that decomposes the sentence intersection task into several simpler annotation tasks, facilitating the construction of a high quality dataset via crowdsourcing. We implement this approach and provide an annotated dataset of 1,764 sentence intersections.

1 Introduction

Various paradigms exist for comparing the meanings of two texts and modeling their semantic overlap. Paraphrase detection (Dolan et al., 2004), for example, tries to identify whether two texts express the same information. It cannot, however, capture cases where there is only partial information overlap. One paradigm that addresses this issue is textual entailment (Dagan et al., 2006), which asks whether one text can be inferred from another. While textual entailment models when the information of one text is subsumed by another, it falls short when neither text strictly subsumes the other, yet a significant amount of information is common to both. A more recent paradigm, semantic text similarity (Agirre et al., 2012), allows for these cases by measuring "how much" similar two sentences are.

In this paper, we consider an additional paradigm for modeling the semantic overlap between two texts. The task of *sentence intersection*, as defined in several variants (Marsi and Krahmer, 2005; McKeown et al., 2010; Thadani and McKeown, 2011), is to construct a sentence containing all the information shared by the two input sentences (Figure 1). While sentence intersection was originally proposed for abstractive summarization, we argue that it is an interesting generic paradigm for modeling information overlap, which generalizes over several previous approaches. In particular, it is expressive enough to capture partial semantic overlaps that are not modeled by textual entailment or even partial textual entailment (Nielsen et al., 2009; Levy et al., 2013). Furthermore, rather than quantifying the *amount* of shared information as in semantic text similarity, sentence intersection captures *what* this shared information is.

Although sentence intersection has existed for over a decade, it has received little attention due to a lack of annotated data. Previous annotation attempts have either used experts, which did not scale, or crowdsourcing, which yielded unreliable annotations (McKeown et al., 2010). We also observe that annotating sentence intersection is difficult for non-experts. We hypothesize that this difficulty stems from the task's requirement that the annotator indicate *all* the shared information when constructing the intersection, necessitating a high level of attention to fine details in two different sentences simultaneously.

This paper addresses the issue of annotating sentence intersection. We clarify some ambiguous aspects of sentence intersection by defining *extractive sentence intersection* (§3): the *set* of all intersections that can be composed only from words in the input sentences. Though less ambiguous, this task is still challenging for non-experts to annotate correctly (§4). Our main observation is that extractive intersections have a common underlying structure – *an aligned partition* (§6). In a nutshell, the dependency tree of

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2891–2901, Osaka, Japan, December 11-17 2016.

s_1	Obama visited Canada yesterday.
s_2	The president flew to Ottawa to meet Trudeau.
$\cap$	The president visited Canada.
$\cup$	Obama flew to Ottawa yesterday to meet Trudeau.

Figure 1: Sentence intersection ($\cap$) versus union ($\cup$). Union captures all the information in the input sentences, whereas intersection captures all the *shared* information.

s_1	Sanders disagrees with HRC on Super PACs.
s_2	Bernie criticizes Hillary on campaign funding.
$\cap$	Sanders disagrees with HRC on campaign funding. Sanders disagrees with Hillary on campaign funding. Bernie disagrees with HRC on campaign funding. Bernie disagrees with Hillary on campaign funding.

Figure 2: Extractive intersection is a combinatorial problem, because the common information between two sentences can often be expressed in various equivalent ways.

an intersection sentence can be partitioned into subtrees, each one originating from one of the input sentences. The interesting property is that for every subtree originating from one input sentence, there exists another subtree in the other input sentence, which entails the first. For this purpose, we define *subtree entailment* (§5), an extension of the well-studied inference rules paradigm (Lin and Pantel, 2001).

We then design a semi-automatic algorithm that helps the annotator construct an aligned partition and eventually the extractive intersection set (§7). Our process decomposes the complex task of extractive sentence intersection into a collection of simpler local subtree entailment queries. The entailment queries (which are easy for humans but hard for machines) are annotated manually, while the combinatorial aspect of generating intersections is completely automated. Finally, we apply our process and annotate 1,764 examples via crowdsourcing (§8). These annotations are superior to direct manual crowdsourced annotations.[1]

2 Background: Sentence Intersection

The ability to combine two (or more) semantically-similar sentences into one is a core requirement of abstractive summarization. Various flavors of this combination task, dubbed *sentence fusion*, have been studied. Barzilay et al. (1999; 2003; 2005) focused on generating a single sentence from the most important parts of the input sentences. The importance criterion was argued to be too vague for consistent human annotation (Daume III and Marcu, 2004), giving rise to the more accurately-defined notions of sentence *union* and *intersection* (Marsi and Krahmer, 2005). Sentence union aspires to create a single sentence containing *all* the information in the input sentences, while sentence intersection tries to isolate all the information that they have *in common* – i.e. the consensus. Figure 1 demonstrates this difference.

Annotation Efforts Perhaps the most pressing issue in making sentence intersection an appealing semantic task is creating enough high-quality annotated data. However, prior art appears to lack a method for annotating sentence intersection that is both efficient (scalable) and accurate. The previous attempt to annotate sentence intersection via crowdsourcing (McKeown et al., 2010) resulted in unreliable annotations. Annotators were presented with two similar sentences, and asked to combine them "into a single sentence conveying only the information they have in common". Each pair of input sentences was given to 5 different workers, and the most agreed-upon (centroid) annotation was selected. Results showed that only 54% of the annotation were indeed valid intersections. On the other hand, sentence *union* annotations over the same sentence pairs had 95% accuracy. Krahmer et al. (2008) also observed intersection annotations to be less consistent than unions while studying query-based sentence fusion.

More recently, Thadani and McKeown (2013) proposed a different dataset, which relied on pre-existing pyramid annotated data (Nenkova and Passonneau, 2004). This dataset is more consistent, but does not comply with the definition of sentence intersection. In practice, it models a simpler task: generating a sentence that contains only shared information, not necessarily *all* of the shared information. It seems that this distinction – the need to identify *every* piece of common information at once – is what makes sentence intersection a combinatorial problem, explaining why it is so difficult for humans to perform this task consistently.

Algorithms The sentence fusion literature typically takes an alignment-based algorithmic approach (Barzilay and McKeown, 2005; Filippova and Strube, 2008), which was also adopted by Thadani and

[1] Our code and dataset are available online: `bitbucket.org/omerlevy/extractive_intersection`

s_1	Bill traveled back in time.
s_2	Ted traveled back in time.
	A person traveled back in time.
$\cap$	A man traveled back in time.
	Someone traveled back in time.

Figure 3: Possible lexical abstractions. The best choice of words is not always obvious.

s_1	Akiko ate cake.
s_2	Akiko ate pudding.
$\cap$	Akiko ate **[something]**.

Figure 4: An intersection containing a placeholder.

McKeown (2011) for sentence intersection. In general, they (lexically) align the input sentences, and then select a subset of the aligned components when generating the fused sentence. Our annotation method is inspired by this approach, though it differs in various ways, such as the kind of alignments it creates (subtree entailments), the output it generates (a *set* of sentences), and the fact that it is interactive.

3 Extractive Sentence Intersection

We define a new variant of sentence intersection, *extractive sentence intersection*, which departs from the original task in two aspects: specifying a set of intersection sentences, while limiting their scope to be based on the lexical elements in the input sentences.

First, we observe that several different output sentences may satisfy the original definition of sentence intersection. In Figure 2, we see two input sentences conveying very similar information that are expressed by completely different words. Since many of the terms are synonymous, choosing one over another is arbitrary (e.g. "Bernie" = "Sanders" in this context). Combining different words from the input sentences creates several intersections – each one as valid as the other. Therefore, instead of defining sentence intersection as a single sentence that contains all the information shared by the input sentences, we define it as the *set* of all such sentences.

Second, we observe that, according to its original definition, an intersection sentence might include words that do not appear in any of the two input sentences. This may happen in two situations: (a) lexical variability – when introducing an unmentioned synonym (e.g. using "Clinton" instead of "HRC" or "Hillary"); (b) lexical abstraction – where creating an intersection requires introducing a new word that subsumes two terms from the input sentences (Figure 3). Intersections that require lexical abstraction are difficult to define and annotate consistently because the desired level of abstraction is often unclear. We avoid the issues of lexical variability and abstraction by focusing on intersections that are *extractive*, i.e. intersections that *only contain lexical elements from the input sentences*.

To maintain grammaticality, we make an exception to this rule. Placeholders, such as *[something]*, *[somewhere]*, and *[sometime]*, can be used when the input sentences provide different information on an entity that cannot be omitted for syntactic reasons (e.g. subjects). In many cases, there is a specific word that could fit instead of a placeholder, but this word does not appear in the input sentences (Figure 4). The task of finding such words requires lexical abstraction, which is beyond our current scope.

Combining both these criteria – output set and extractiveness – we define *extractive sentence intersection* as the set of all sentences that each contains all the information common to the input sentences, while being composed only of words that appeared in the input sentences and placeholders.[2]

The entire set of extractive intersections allows for more accurate automatic evaluation, since the evaluation mechanism does not need to overcome issues in lexical variability; instead, it can simply select the most similar expert-crafted sentence from the set using simple metrics such as BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004). Having multiple possible solutions is not a foreign concept to NLP, and is widely used in translation and summarization.

[2]While this paper discusses intersections between two input sentences, one can theoretically extend this setting to multiple input sentences by consecutively applying the intersection operation. For example, if we have three sentences s_1, s_2, s_3, we could first find the intersection between s_1 and s_2, and then for each sentence s' in $s_1 \cap s_2$, intersect that with s_3. We would essentially take the union of the latter set of intersection sets, i.e. $s_1 \cap s_2 \cap s_3 = (s_1 \cap s_2) \cap s_3 = \bigcup\{s' \cap s_3 | s' \in s_1 \cap s_2\}$.

s_1	It was Rehnquist who presided over the swearing-in ceremony when Roberts took his seat on the U.S. Court of Appeals for the District of Columbia.
s_2	Roberts, nominated to succeed retiring justice Sandra Day O'Connor, easily won senate confirmation to the U.S. Court of Appeals for the District of Columbia two years ago.
$\cap$	Roberts won senate confirmation to the U.S. Court of Appeals for the District of Columbia.

Figure 5: An example pair of sentences from our dataset and their (expert-annotated) intersection.

4 Annotation Feasibility

We wish to create a large gold-standard annotated dataset of extractive intersection via crowdsourcing. Since the previous crowdsourced intersection dataset was of mixed quality (McKeown et al., 2010), we annotate the same data with both experts and crowdsourcing to determine the annotation's feasibility.

Evaluation Metrics We present two complementary methods for evaluating extractive sentence intersection: a strict method based on sentence equality, and a softer measure similar to ROUGE (Lin, 2004). The methods essentially compare a candidate set of sentences C to a gold-standard set G, yielding precision and recall scores that are averaged across the dataset.

The sentence equality method follows directly from the task's definition: $Precision = |C \cap G|/|C|$, $Recall = |C \cap G|/|G|$. Note that the underlying sentence equality measure is simple string equality.

Sentence equality can sometimes be too harsh for measuring intersection. For example, if an intersection sentence contains exactly all the shared information except for one minor detail, we would want to partially penalize it rather than completely rejecting it. We therefore adopted a more lenient evaluation measure, based on ROUGE (Lin, 2004). In automatic summarization, ROUGE measures the n-gram overlap between a single candidate summary and a set of several gold-standard summaries. Whereas ROUGE tries to maximize agreement with *all* gold summaries, we are satisfied if there exists even one gold intersection sentence that is sufficiently similar to the given candidate sentence. Therefore, to calculate the precision of a given candidate intersection sentence $c \in C$, we take the best-matching gold sentence g, and calculate the portion of n-grams in c that are covered by g. Similarly, a gold sentence's recall is measured versus the best-matching $c \in C$. We define an intersection set's precision and recall by taking the averages. Formally: $Precision = \frac{1}{|C|}\sum_{c \in C} \max_{g \in G}(Cover(c, g))$, $Recall = \frac{1}{|G|}\sum_{g \in G} \max_{c \in C}(Cover(g, c))$, where the portion of n-grams covered is $Cover(a, b) = |ngrams(a) \cap ngrams(b)|\,/|ngrams(a)|$. To avoid deviating too much from the sentence equality measure, we select a conservative definition of n-grams ($n = 4$). Note that replacing $Cover$ with equality in the equations above is exactly the sentence equality method.

Annotation We annotated a random sample of 200 sentence pairs, half from Columbia's sentence fusion dataset (McKeown et al., 2010) and half from MSR's paraphrase dataset (Dolan et al., 2004). Columbia's dataset contains 297 pairs of related sentences, annotated for sentence intersection and union. MSR's dataset contains 1,500 pairs of related sentences, originally annotated for a paraphrase detection task, and later with semantic text similarity scores (Agirre et al., 2012). Existing annotations are ignored, and replaced by our new annotations (see Figure 5).

The first author annotated the data (as an expert), and Mechanical Turk workers were hired to annotate the same sentence pairs. Workers were told to "extract" a new sentence by selecting words from the input sentences or a bank of placeholders, and encouraged to create as many intersections as possible. Each sentence pair was given to 5 workers, of which the most consistent 3 were selected to reduce the effect of poor annotations (agreement between two workers was measured using the ROUGE-like F_1). We also used the set of both input sentences as a simple annotation baseline (the "input" baseline), which captures (with some error) the cases in which one sentence is completely entailed by another.

Results Table 1 shows very poor crowd-expert agreement rates when measured using sentence equality. The crowdsourced annotation is slightly better than the input baseline (precision-wise), and does not seem to improve coverage. Even using the softer ROUGE-like metric, the crowdsourced data does not seem to provide enough added value beyond the simple input baseline (its F_1 is only 0.014 higher).

	Sentence Equality			ROUGE-like		
	P	R	F_1	P	R	F_1
Input	0.080	0.124	0.097	0.611	0.952	0.744
Direct	0.130	0.124	0.127	0.735	0.781	0.758

Table 1: The agreement between crowd ("Direct") and expert annotations. "Input" is a baseline where the two input sentences are also the output.

s	Oswald killed JFK.
$\sigma(s, \text{JFK}, \text{Kennedy})$	Oswald killed Kennedy.
$\sigma(s, \text{killed}, \text{assassinated})$	Oswald assassinated JFK.
$\sigma(s, \text{killed}, \text{was shot by})$	JFK was shot by Oswald.
$\sigma(s, \text{killed}, \text{died})$	JFK died.

Figure 6: Four examples of the substitution function.

Extractive intersection is clearly a difficult task for non-expert annotators. We suggest that this stems from the many nuanced pieces of information in each sentence (precision) and the many possible combinations that reflect valid extractive intersections (recall). From our own annotation effort, we noticed that these two difficulties pertain to an underlying structure that consistently appears in extractive intersections. In the following sections, we describe this structure (§6) and propose a way of exploiting it to generate high-quality annotations (§7).

5 Subtree Entailment in Context

In this section, we describe a new textual relation, *subtree entailment in context*, which will assist in defining the underlying structure of extractive intersection. Subtree entailment in context models inference between rich lexical-syntactic patterns (subtrees of syntactic dependency trees), while considering internal context (instantiated arguments) and external context (substitution in a complete sentence). As such, it generalizes over previous ideas presented separately in prior art; subtrees were used in TEASE (Szpektor et al., 2015) and PPDB (Pavlick et al., 2015), internal context was considered in context-sensitive relation inference (Zeichner et al., 2012; Melamud et al., 2013; Levy and Dagan, 2016), and external context is studied in the lexical substitution task (McCarthy and Navigli, 2007; Biemann, 2013; Kremer et al., 2014). Subtree entailment in context is the first notion that combines all these traits. In addition to these advantages, we also introduce a new mechanism for handling changes in arity, in case one subtree contains more/less arguments than another.

The Substitution Function To define subtree entailment in context, we must first define an auxiliary operation – subtree substitution. The substitution function σ is given a sentence tree s, a subtree within that sentence t, and another subtree t', which is not necessarily part of s. It creates a new sentence s' by replacing t with t' (see Figure 6): $s' = \sigma(s, t, t')$.

Slot Assignments The substitution function does not specify how the child nodes of t connect to t'. This is not always trivial; e.g. in the third example in Figure 6, the *subject* of "killed" is attached as the *object* of "was shot by". Moreover, the arity of t (the number of expected child nodes) might be different from that of t', as in the fourth example.

The missing piece of the puzzle is *slot assignments*. A *slot* is a dangling edge whose head is part of a subtree (either t or t') but its modifier is not. Slot assignments match these outgoing edges, and are especially useful when going from passive to active and when there is a change in arity. We denote slot assignment as a function α from the slots of t' to those of t, and extend the definition of the substitution function to include it: $s' = \sigma(s, t, t', \alpha)$.

The fact that α is a function means that not all of t's child nodes will necessarily appear in the new sentence s'. In addition, the slots of t' can be assigned a null value ($\emptyset$), which means that none of t's child nodes will fill this slot, leaving it empty. When generating a natural language expression with an empty slot, it is often the case that a placeholder will fill it (see §3). For example, let us invert the fourth example in Figure 6: $s' = \sigma(\text{"JFK died."}, \text{died}, \text{killed}, \alpha)$. If $\alpha(subject) = \emptyset$, then s' will be "[someone] killed JFK", where "[someone]" is a placeholder.

Definition Given a sentence s, we can say that t_p (p for premise), a subtree of s, *entails* some other subtree t_h (h for hypothesis) in the context of s, if: $s \vDash \sigma(s, t_p, t_h, \alpha)$, where α assigns the slots of t_h, and $\vDash$ is textual entailment. In other words, if s is true, then replacing t_p with t_h in s should create a new sentence that is also true. Considering the second example from Figure 6, "killed" entails "assassinated"

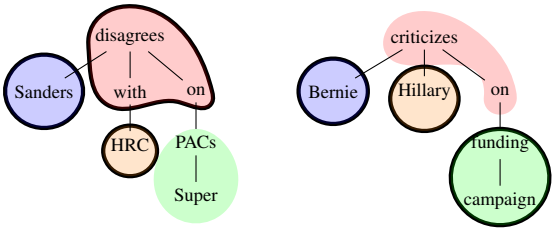


Figure 7: The aligned partition of the example in Figure 2. Subtrees marked by the same color are aligned, and those surrounded by dark borders are entailed by their counterpart.

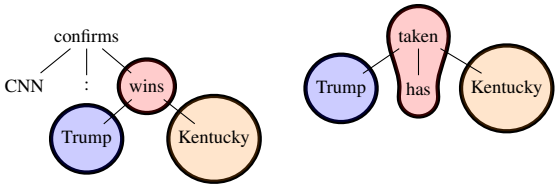


Figure 8: Two input sentences whose aligned partition does not include s_1's root.

(in the context of s) if given that "Oswald killed JFK", it is also true that "Oswald assassinated JFK". Subtree entailment is always in the context of a full premise s; we therefore use the shorthand notation $t_p \models_s t_h$.

Notice that, in fact, we are reducing the subtree entailment question to a textual entailment question, where both premise and hypothesis are full sentences (instantiated predicates). This is essential, since one cannot ask "given that 'killed' is true, is 'assassinated' true too?"; an uninstantiated predicate (or worse, a noun) does not yield a truth value.

6 Properties of Extractive Intersections

When examining the dependency trees of sentences created by extractive intersection, we observe that almost all of them (93%–99.5%, see footnote 5 in §8) exhibit three particular properties, which we define using subtree entailment in context.

Aligned Partition The dependency tree of a sentence created by extractive intersection can be partitioned into subtrees, each one originating from one of the input sentences. Interestingly, for every subtree t_1 originating from s_1 (without loss of generality), there exists another subtree t_2 in the other input sentence s_2, which entails t_1 ($t_2 \models_{s_2} t_1$). The entailment relation between t_1 and t_2 can be mutual or asymmetric, but it is always one-to-one, i.e. no other subtree within s_2 entails t_1, and vice versa. This induces a partition of the input sentences' dependency trees (seen in Figure 7), where each subtree t_1 is either unaligned or in an entailment relation with exactly one subtree from s_2. This also means that each aligned subtree (except for the root) has exactly one parent, which we denote as $\pi(t, s)$ (t being a subtree in sentence s).

We observe that the vast majority of output sentences can be easily created from such an aligned partition of the input sentences. This is not surprising, because every intersection sentence is entailed by both input sentences (Marsi and Krahmer, 2005). It makes sense that the same quality exists at the subtree level, which we model with aligned partitions.

Inverse Inheritance of Alignment The second property, *inverse inheritance of alignment*, describes how subtrees in the output sentence connect. Given a subtree t_1, its parent in the output sentence's tree is either its original parent $\pi(t_1, s_1)$ or its "parent-in-law" $\pi(t_2, s_2)$ (the parent of its aligned subtree). Looking back at the input sentences, this means that every pair of aligned subtrees also has aligned parents, i.e. the parents "inherit" the alignment from their children. However, this does not mean that the direction of entailment needs to be the same; in Figure 7, "Super PACs" entails "campaign funding" ($t_1 \models_{s_1} t_2$), while "X criticizes Y on Z" entails "X disagrees with Y over Z" ($\pi(t_2, s_2) \models_{s_2} \pi(t_1, s_1)$).

Entailed Embedded Clauses Figure 8 shows that the input sentences are not necessarily aligned at the root, and that the common information might be contained in an embedded clause. In such cases, if there is a valid (non-empty) intersection, we typically observe that the embedded clause is entailed by the original sentence. In terms of subtree entailment, this means that the subtree wrapping the embedded clause entails the empty subtree (a single slot with no nodes); e.g. if "CNN confirms: X", then "X" is typically true. Conversely, if the embedded clause is not entailed by the original sentence (e.g. s_3 ="Carson will

quit if Trump wins Kentucky") then there is no intersection ($s_1 \cap s_3 = s_2 \cap s_3 = \emptyset$, assuming s_1 and s_2 from Figure 8).

7 A Semi-Automatic Algorithm

Based on the properties discussed in §6, we describe a semi-automatic algorithm for annotating extractive intersection. It requires relatively simple human interaction throughout the process, while automating the combinatorial aspect, which is difficult for humans. Our algorithm also masks the underlying syntactic tree from the annotator, making the entire process crowdsourcable. The core of our algorithm is creating an aligned partition of the input sentences. Once the aligned partition is obtained, we can deterministically create the set of output dependency trees, from which natural language sentences are generated. Some minor technical details are omitted from this description, and will be made available with our code and its documentation upon publication.

7.1 Creating the Aligned Partition

Given the dependency trees[3] of two sentences, we create their aligned partition over 4 steps. We first elicit lexical alignments from the annotators; these are used to detect the aligned partition's root in each sentence, and then as a seed for automatically deriving subtree alignments. The final step validates that the subtrees are indeed in an entailment relation, and also annotates the entailment's direction.

Step 1: Lexical Alignment We present both sentences in natural language, and ask the annotators to mark words (or phrases) that appear in different sentences yet have similar meanings. This phase is recall-oriented, and its goal is to capture as many potential entailments by annotating a slightly looser notion. We limit the number of alignments to one per token; this constraint is essential for inducing an aligned *partition*. Despite detailed guidelines and examples, non-expert annotators mainly mark nouns, typically ignoring verbs and other predicates unless they are identical. We address this issue in Step 3.

Step 2: Root Detection As discussed in §6, the root of an aligned partition does not always contain the roots of both input sentences. We therefore apply a heuristic, on each sentence separately, which finds the root of the embedded clause participating in the aligned partition: the lowest common ancestor of nodes that have a lexical alignment. This technique is motivated by our observation that, for intersection purposes, the original sentence can be replaced with the embedded clause since it contains all the aligned components. In Figure 8, for example, if an annotator were to only align "Trump" and "Kentucky" with themselves, the heuristic would detect "wins" and "taken" as the roots of s_1 and s_2, respectively.

Step 3: Subtree Alignment Since annotators rarely align predicates, we introduce a heuristic method for automatically deriving subtree alignments from the lexical ones provided manually (Step 1). Figure 9 shows a typical lexical alignment. Our heuristic makes greedy local changes until the following four constraints are satisfied:

Inverse Inheritance of Alignment: In §6 we saw that if two subtrees are aligned, then so are their parents. To enforce this property, we create new alignments between parents of already-aligned subtrees. In our example (Figure 9), this yields two new alignments: "primary" – "wins" and "to" – "wins".

Include Function Modifiers: Annotators also tend to ignore function words. We observe that for certain dependency types[4] that connect function words to content words as modifiers, the modifiers should always be part of the same alignment as their heads. Therefore, if the head participates in an alignment, the modifier is added as well; e.g. "the" modifies "primary" and is therefore added to its alignment.

No Overlapping Alignments: Since aligned partitions induce a *partition* of each input sentence, each node take part in one alignment at most. Therefore, if a node participates in more than one alignment, those two alignments are merged. In our example, "wins" participates in two alignments. After merging, we are left with a single alignment (excluding the initial ones): "the", "primary", "to" – "wins".

[3] We used the Stanford converter on top of the Berkeley parser (Petrov and Klein, 2007), which produced correct dependency trees for most of our data.

[4] The following Stanford dependencies: `det`, `aux`, `auxpass`, `neg`, `prt`, `attr`.

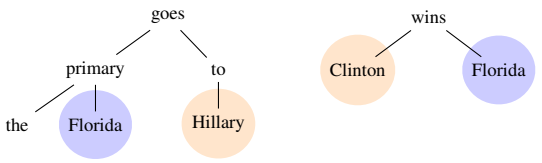


Figure 9: The information provided by a typical annotator in the lexical alignment phase is usually limited to nouns.

	Sentence Equality			ROUGE-like		
	P	R	F_1	P	R	F_1
Input	0.080	0.124	0.097	0.611	0.952	0.744
Direct	0.130	0.124	0.127	0.735	0.781	0.758
Semi-Auto	0.360	0.305	0.330	0.820	0.902	0.859

Table 2: The agreement between crowd and expert annotations. "Semi-Auto" is the semi-automatic annotation process, "Direct" is the direct annotation process described in §4, and "Input" is a baseline where the two input sentences are also the output.

Subtrees Only: Subtree entailment in context (§5) requires every alignment to be between two subtrees. If an aligned component is not a subtree, (i.e. is non-contiguous in the dependency tree) that subgraph will be turned into a subtree by adding the nodes all the way up to the lowest common ancestor. In our case, ("the", "primary", "to") is non-contiguous; adding the lowest common ancestor to this alignment ("goes") yields the subtree alignment: "the", "primary", "goes", "to" – "wins".

Step 4: Entailment Questions The previous step yields a *potential* aligned partition, where aligned subtrees have not yet been tested for entailment. Based on this structure, we generate a set of yes/no subtree entailment questions. Starting at the root, we traverse the aligned partition, and for each alignment (t_1, t_2), two subtree entailment questions are generated: $t_1 \vDash_{s_1} t_2$ and $t_2 \vDash_{s_2} t_1$.

Reusing the example from Figure 9, let t_1 = "the", "primary", "goes", "to" and t_2 = "wins". In the entailment question $t_1 \vDash_{s_1} t_2$, the premise is the original sentence in which t_1 appeared ("The Florida primary goes to Hillary."). The hypothesis is created by substituting t_1 with t_2: $\sigma(s_1, t_1, t_2, \alpha)$. The slot assignment α is deduced directly from the existing subtree alignments; if a child node of t_2 is aligned to a child node of t_1, their slots will be assigned accordingly. If no such alignment exists for a given slot, it will be assigned $\emptyset$ and manifest as a placeholder. In our example, this yields the assignment: "the X primary goes to Y" – "Y wins X", because (X) "Florida" is aligned to itself, and (Y) "Hillary" is aligned to "Clinton". Therefore, the hypothesis we generate is: "Hillary wins Florida."

Annotators are asked if given that the premise is true, the hypothesis is true too. This step determines both the validity and direction of each alignment, creating the final aligned partition.

7.2 Creating the Intersection Set

The aligned partition contains all the information necessary for constructing the intersection set. We create this set by traversing the aligned partition and generating every combination of subtrees, in which only one subtree from each alignment is chosen. Given an alignment between two subtrees, if only subtree is entailed, this subtree is always chosen. If the subtrees are mutually entailing, the number of generated trees is doubled, each one containing a different subtree. If there is no entailment, the subtrees are not aligned, and will not participate in any of the generated sentences.

Overall, this process creates 2^n sentences, where n is the number of mutually entailing subtrees in the aligned partition. For example, the aligned partition in Figure 7 contains 2 mutually-entailing subtrees ("Bernie" = "Sanders", "Hillary" = "HRC"), resulting in 4 intersection sentences (Figure 2). Finally, we generate natural language sentences from the intersection sentences' trees.

8 Annotated Dataset

We hired Mechanical Turk workers to annotate the entire set of 1,800 examples from the sentence fusion (McKeown et al., 2010) and paraphrase (Dolan et al., 2004) datasets presented in §4 using our semi-automatic algorithm. After removing spam annotations, we had an annotated dataset of 1,764 extractive sentence intersections. Again, we retained the 3 most consistent annotations out of 5, per example. For evaluation, we compared the same 200 examples we analyzed in §4. We call this annotation "Semi-Auto", the expert annotation "Expert", and the manual annotation in §4 "Direct".

In terms of measured agreement with Expert, Table 2 shows that Semi-Auto substantially improves over Direct in both the sentence equality and ROUGE-like metrics. Semi-Auto has much higher precision

than both Direct and the input baseline. We hypothesize that explicitly decomposing the intersection task into several local subtree entailment queries allows the annotators to focus on the finer details and nuances of each sentence. Semi-Auto's recall is also better than Direct's because the combinatorial component of creating the entire intersection set is automated.

Although Semi-Auto significantly increases agreement with experts, the agreement is still not perfect. To determine whether these errors stem from annotator quality or algorithmic issues, we sampled 25 input sentence pairs and examined the intersection sentences constructed by Semi-Auto that were not produced by Expert. From this sample of false-positives, 42.2% were actually valid intersections, which were not covered by Expert because of the task's subjective nature. While we used only a single expert, multiple expert annotators could potentially reduce the subjectivity factor. An additional 51.1% stem from real annotation errors by Mechanical Turk workers. A deeper look reveals that these errors result from low-quality annotators who did not follow our instructions. Having said that, the vast majority of these errors manifest as a single missing or superfluous word (in some cases, it is even a function word), while getting the core aspects of the intersection correct. Only 6.7% of the cases were traced to algorithmic errors, most of which are caused by the way English dependency parsers represent coordinations.[5] Overall, it appears that the major cause of error is a portion of Mechanical Turk annotators who failed to follow our instructions accurately. Our semi-automatic algorithm, on the other hand, seems relatively robust.

9 Conclusions

This work resurrects the task of sentence intersection, and addresses the main reason it has remained dormant: lack of annotated data. We show that our new variant of the task, extractive sentence intersection, can be decomposed into several simpler tasks, allowing for high-quality annotation via crowdsourcing. We hope our insights, data, and algorithmic framework provide a foundation for future work on sentence intersection and semantic overlap.

Acknowledgements

This work was supported by the German Research Foundation via the German-Israeli Project Cooperation (grant DA 1600/1-1), the Israel Science Foundation grant 880/12, and by grants from the MAGNET program of the Israeli Office of the Chief Scientist (OCS).

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 550–557, College Park, Maryland, USA, June. Association for Computational Linguistics.

Regina Barzilay. 2003. *Information Fusion for Multidocument Summarization: Paraphrasing and Generation*. Ph.D. thesis, Columbia University.

Chris Biemann. 2013. Creating a system for lexical substitutions from scratch using crowdsourcing. *Language Resources and Evaluation*, 47(1):97–122.

[5] A word on coordinations: English dependency parsers typically represent coordinations ("Bill and Ted") with the first conjunct ("Bill") as the head, and the other conjuncts ("Ted") and coordinators ("and") as its modifiers. This prevents us from modeling entailments between reflexive verbs, such as "X met Y"="X and Y met", using subtrees. To correctly capture the pattern "X and Y met", we need to use Prague-style coordinations (Popel et al., 2013), which place the coordinator ("and") at the head of the structure. This modification, which we did not apply to Semi-Auto, fixes 13 out of 14 cases in which we did not observe a perfect aligned partition during our expert annotation, leaving only 1 exception in 200 examples.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In Joaquin Quiñonero Candela, Ido Dagan, Bernardo Magnini, and Florence d'Alché Buc, editors, *Machine Learning Challenges*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer.

Hal Daume III and Daniel Marcu. 2004. Generic sentence fusion is an ill-defined summarization task. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 96–103, Barcelona, Spain, July. Association for Computational Linguistics.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of Coling 2004*, pages 350–356, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Katja Filippova and Michael Strube. 2008. Sentence fusion via dependency graph compression. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 177–185, Honolulu, Hawaii, October. Association for Computational Linguistics.

Emiel Krahmer, Erwin Marsi, and Paul van Pelt. 2008. Query-based sentence fusion is better defined and leads to more preferred results than generic sentence fusion. In *Proceedings of ACL-08: HLT, Short Papers*, pages 193–196, Columbus, Ohio, June. Association for Computational Linguistics.

Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. 2014. What substitutes tell us - analysis of an "all-words" lexical substitution corpus. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 540–549, Gothenburg, Sweden, April. Association for Computational Linguistics.

Omer Levy and Ido Dagan. 2016. Annotating relation inference in context via question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Berlin, Germany, August. Association for Computational Linguistics.

Omer Levy, Torsten Zesch, Ido Dagan, and Iryna Gurevych. 2013. Recognizing partial textual entailment. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 451–455, Sofia, Bulgaria, August. Association for Computational Linguistics.

Dekang Lin and Patrick Pantel. 2001. Dirt: Discovery of inference rules from text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–328. ACM.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July. Association for Computational Linguistics.

Erwin Marsi and Emiel Krahmer. 2005. Explorations in sentence fusion. In *Proceedings of the European Workshop on Natural Language Generation*, pages 109–117. Citeseer.

Diana McCarthy and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 48–53, Prague, Czech Republic, June. Association for Computational Linguistics.

Kathleen McKeown, Sara Rosenthal, Kapil Thadani, and Coleman Moore. 2010. Time-efficient creation of an accurate sentence fusion corpus. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–320, Los Angeles, California, June. Association for Computational Linguistics.

Oren Melamud, Jonathan Berant, Ido Dagan, Jacob Goldberger, and Idan Szpektor. 2013. A two level model for context sensitive inference rules. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1331–1340, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 145–152, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Rodney D Nielsen, Wayne Ward, and James H Martin. 2009. Recognizing entailment in intelligent tutoring systems. *Natural Language Engineering*, 15(04):479–501.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China, July. Association for Computational Linguistics.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

Martin Popel, David Mareček, Jan Št?pánek, Daniel Zeman, and Zd?n?k Žabokrtský. 2013. Coordination structures in dependency treebanks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 517–527, Sofia, Bulgaria, August. Association for Computational Linguistics.

Idan Szpektor, Hristo Tanev, Ido Dagan, Bonaventura Coppola, and Milen Kouylekov. 2015. Unsupervised acquisition of entailment relations from the web. *Natural Language Engineering*, 21:3–47, 1.

Kapil Thadani and Kathleen McKeown. 2011. Towards strict sentence intersection: Decoding and evaluation strategies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 43–53, Portland, Oregon, June. Association for Computational Linguistics.

Kapil Thadani and Kathleen McKeown. 2013. Supervised sentence fusion with single-stage inference. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1410–1418, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Naomi Zeichner, Jonathan Berant, and Ido Dagan. 2012. Crowdsourcing inference-rule evaluation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 156–160, Jeju Island, Korea, July. Association for Computational Linguistics.

Context-Sensitive Inference Rule Discovery: A Graph-Based Method

Xianpei Han Le Sun
State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Beijing, China
{xianpei, sunle}@nfs.iscas.ac.cn

Abstract

Inference rule discovery aims to identify entailment relations between predicates, e.g., '*X acquire Y → X purchase Y*' and '*X is author of Y → X write Y*'. Traditional methods discover inference rules by computing distributional similarities between predicates, with each predicate is represented as one or more feature vectors of its instantiations. These methods, however, have two main drawbacks. Firstly, these methods are mostly *context-insensitive*, cannot accurately measure the similarity between two predicates in a specific context. Secondly, traditional methods usually model predicates *independently*, ignore the rich inter-dependencies between predicates. To address the above two issues, this paper proposes a graph-based method, which can discover inference rules by effectively modelling and exploiting both the context and the inter-dependencies between predicates. Specifically, we propose a graph-based representation—*Predicate Graph*, which can capture the semantic relevance between predicates using both the predicate-feature co-occurrence statistics and the inter-dependencies between predicates. Based on the predicate graph, we propose a context-sensitive random walk algorithm, which can learn context-specific predicate representations by distinguishing context-relevant information from context-irrelevant information. Experimental results show that our method significantly outperforms traditional inference rule discovery methods.

1 Introduction

Inference rule discovery aims to identify entailment relations between predicates, such as '*X acquire Y → X purchase Y*' and '*X is author of Y → X write Y*', with each predicate is a textual pattern with (two) variable slots (*X* and *Y* in above). Inference rules are important in many fields such as Question Answering (Ravichandran and Hovy, 2002), Textual Entailment (Dagan et al., 2006) and Information Extraction (Hearst, 1992). For example, given the problem "*Which company purchases WhatsApp?*", a QA system can extract the answer "*Facebook*" from the sentence "*Facebook acquires WhatsApp for $19 billion*" based on the inference rule '*X acquire Y → X purchase Y*'.

Given a set of predicates and their instantiations in a large corpus, most traditional methods identify inference rules by computing distributional similarities between predicates, where each predicate is represented as one or more feature vectors of its variable instantiations. For example, given the predicates and their instantiations in Figure 1, we can represent '*X acquire Y*' as {*X='Google', Y='YouTube', X='children', Y='skill'*} and measure the similarity between '*X acquire Y*' and '*X purchase Y*' based on their common features {*X='Google', Y='YouTube'*}. To achieve the above goal, many similarity measures have been proposed for inference rule discovery, such as *DIRT* Similarity (Lin and Pantel, 2001), *Balanced-Inclusion* similarity (Szpektor and Dagan, 2008) and *Soft Set Inclusion* similarity (Nakashole et al., 2012), etc.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2902–2911, Osaka, Japan, December 11-17 2016.

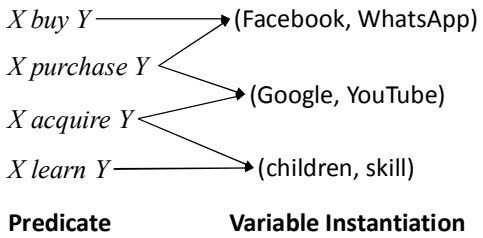


Figure 1. Some predicates and their variable instantiations

These distributional similarity based methods, however, have two main drawbacks:

Firstly, these methods are mostly *context-insensitive*, cannot accurately measure the similarity between two predicates in a specific context. Due to the ambiguity of predicates, a predicate may have different meanings under different contexts (In this paper, as the same as Melamud et al. (2013), the context of a predicate is specified by the predicate's given arguments). For example, the predicate '*X acquire Y*' should have different meanings under context (*Google, YouTube*) and context (*children, skill*), because it corresponds to two different senses of *acquire* in these two contexts. Unfortunately, traditional methods mostly use the same representation to represent a predicate in different contexts, therefore may learn invalid inference rules. For example, given two predicates '*X acquire Y*' and '*X purchase Y*', traditional context-insensitive methods will return the same similarity between them in contexts (*Google, YouTube*) and (*children, skill*). However, '*X acquire Y* → *X purchase Y*' is not a valid rule in context (*children, skill*). Based on the above discussion, we believe that context-specific predicate representation is critical to the success of inference rule discovery.

Secondly, traditional methods usually model predicates *independently*, ignore the rich inter-dependencies between predicates. It is clear though, that there are rich inter-dependencies between predicates. For example, '*X buy Y*' is a synonym of '*X purchase Y*', and '*Y be acquired by X*' is the passive form of '*X acquire Y*'. These dependencies can be exploited to enhance inference rule discovery in many ways. For instance, we can collect richer instantiation co-occurrence statistics per predicate by combining the statistics of semantically similar predicates, or enforce global coherence between the representations of semantically similar predicates. Ignoring these useful inter-dependencies, traditional methods often suffer from the data sparsity problem. For example, if we represent predicates using only their instantiations, we cannot identify the inference rule '*X acquire Y* → *X buy Y*' in Figure 1, because '*X acquire Y*' and '*X buy Y*' don't share any common features.

To address the above two problems, this paper proposes a graph-based method, which can effectively exploit both the context of a predicate and the inter-dependencies between predicates for accurate inference rule discovery. Specifically, we propose a graph-based representation, called *Predicate Graph*, which can capture the semantic relevance between predicates and features by encoding both the predicate-feature co-occurrence statistics and the rich inter-dependencies between predicates. For example, the predicate graph will model the semantic relevance between the predicate '*X buy Y*' and the feature *X*='*Google*' in Figure 1 by taking advantage of the synonym relation between '*X buy Y*' and '*X purchase Y*'. Based on the predicate graph, we propose a context-sensitive random walk algorithm, which can learn context-specific predicate representations by distinguishing context-relevant information from context-irrelevant information. For example, to learn the representation of '*X acquire Y*' under context (*people, language*), our method will identify (*Google, YouTube*) and (*Facebook, WhatsApp*) in Figure 1 as context-irrelevant and will identify (*children, skill*) as context-relevant.

We have evaluated our method on a publicly available dataset. The experimental results show that, using context-specific predicate representations and taking advantage of inter-dependencies between predicates, our method can significantly outperform traditional inference rule discovery methods.

This paper is structured as follows. Section 2 briefly reviews related work. Section 3 describes the proposed method. Section 4 presents and discusses experimental results. Finally we conclude this paper in Section 5.

2 Related Work

Many approaches have been proposed for inference rule discovery, and most of them are distributional similarity based methods. Based on the distributional hypothesis, traditional methods differ in their feature representations and their similarity measures. For predicate representation, some methods represent predicates using one feature vector, where each feature is a pair of argument instantiations such as *X*='*children*'-*Y*='*skill*'(Szpektor et al., 2004; Sekine, 2005; Nakashole et al., 2012; Dutta et al., 2015); some methods represent predicates using two or more feature vectors, one for each argument slot (Lin and Pantel, 2001; Bhagat et al., 2007), e.g., one feature vector for slot *X* and one for slot *Y*. To compute the similarity between predicates, many similarity measures have been proposed, such as *DIRT* Similarity (Lin and Pantel, 2001), *Balanced-Inclusion* similarity (Szpektor and Dagan, 2008) and *Soften Set Inclusion* similarity (Nakashole et al., 2012), etc. Hashimoto et al. (2009) proposed a conditional probability based directional similarity measure to acquire verb entailment pairs on a large scale corpus. As discussed in above, the main drawbacks of these methods are that they are *context-insensitive* and model predicates *independently*.

Having observed that the meaning of a predicate is context-sensitive, several recent methods try to model the context of a predicate using class-based model or latent topic model. The class-based models represent the context of a predicate using ontological type signatures (Pantel et al., 2007; Nakashole et al., 2012), e.g., <*singer, song*> for '*X sing Y*', based on the assumption that two predicates in a rule must have the same type signature. The shortcomings of the class-based context models are that they need a fine-grained ontology and it is often very challenging to determine the fine-grained types of arguments. The latent topic based model represents the context of a predicate as a vector in a low dimensional space, such as the LSA-based model (Szpektor et al., 2008) and the LDA based model (Ritter et al., 2010; Dinu and Lapata, 2010). Based on the context vector, the similarity between two predicates are computed by combining both the context vector similarity and the feature vector similarity (Szpektor et al., 2008), or by first learning predicate similarity per topic, then combining the per-topic similarities using context vector (Melamud et al., 2013). Currently, most of the context-sensitive methods focus on developing an extra context model, by contrast our method focuses on the learning of context-specific predicate representations, without the need of an extra context model.

Recent research has also investigated the jointly learning of inference rules. Kok and Domingos (2008) and Yates and Etzioni (2009) learned inference rules by clustering predicates using relational clustering algorithms. Berant et al.(2010) and Berant et al.(2011) proposed two global learning methods, which first classify each pair of predicates using a local classifier, then these local results are globally rescored using *Integer Linear Programming(ILP)* algorithm. Nakashole et al. (2012) proposed a prefix-tree mining algorithm, which can arrange predicates into a semantic taxonomy. The current joint learning methods mostly employ a meta-classification schema, i.e., the inter-dependencies between predicates are used to adjust the pair-wise predicate similarities, therefore their predicate representations still suffer from the data sparsity problem. In contrast our method exploits the inter-dependencies for better predicate representation, which can effectively resolve the data sparsity problem.

3 Graph-Based Context-Sensitive Inference Rule Discovery

This section describes our graph-based method for context-sensitive inference rule discovery. We first construct a graph, which can effectively capture the semantic relevance between predicates and features. Then we propose a context-sensitive random walk algorithm, which can learn accurate, context-sensitive predicate representations. Finally, we discover inference rules by computing similarities between context-sensitive predicate representations.

3.1 The Predicate Graph Representation

Generally, there are two kinds of information which can be exploited to represent a predicate: 1) its variable instantiations in a corpus, such as the instantiations (*Google, YouTube*) and (*children, skill*) in Figure 1 for representing predicate '*X acquire Y*'; 2) the information from semantically similar predicates, for example, the instantiation (*Google, YouTube*) of '*X purchase Y*' can be used to enrich the representation of '*X buy Y*'. In this paper, we uniformly encode the above two kinds of information using a graph representation, named *Predicate Graph*, which is defined as follows:

*A **Predicate Graph** is a weighted graph **G=(V, E)**, where the node set **V** contains all predicates and all features of these predicates; each edge between a predicate and a feature represents a co-occurrence relation between them; each edge between two predicates represents a semantic-dependent relation between them.*

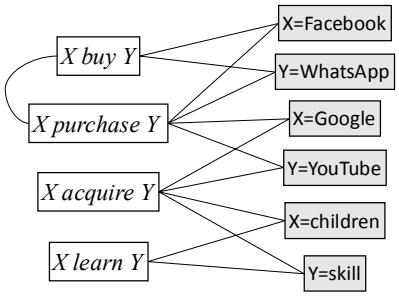


Figure 2. A predicate graph demo

Figure 2 demonstrates a predicate graph, which is constructed using the information in Figure 1. We can see that, the instantiation information of predicates is modelled by co-occurrence edges between (*predicate, feature*), such as the edges between (‘*X buy Y*’, *Y*=‘*WhatsApp*’) and between (‘*X buy Y*’, *X*=‘*Facebook*’). The inter-dependencies between predicates are modelled by semantic-dependent edges between predicates, e.g., the edge between (‘*X buy Y*’, ‘*X purchase Y*’). Based on the co-occurrence and the semantic-dependent edges, both the explicit and the implicit semantic relevance between predicates and features can be captured using the paths between them. For example, the implicit semantic relevance between the feature *X*=‘*Google*’ and the predicate ‘*X buy Y*’ can be modelled through the path *X*=‘*Google*’–‘*X purchase Y*’–‘*X buy Y*’.

The Construction of Predicate Graph. Given a set of predicates and their instantiations in a large corpus, we construct predicate graph by first adding all predicates and all features as nodes, then we link these nodes using the following two types of edges:

- *Co-occurrence Edge*. We take each argument instantiation of a predicate *p* as a feature *f* of *p* and add a co-occurrence edge between them, the pointwise mutual information (PMI) between *p* and *f* is used as the edge’s weight;
- *Semantic-Dependent Edge*. To encode inter-dependencies between predicates, we add a semantic-dependent edge between a predicate *p* and each of its semantically similar predicates. We use the same edge weight α for all semantic-dependent edges, and which will be empirically tuned. Specifically, given a predicate *p*, we find its semantically similar predicates as follows: 1) we identify its active/passive verb form as its semantically similar predicate, e.g., ‘*Y be acquired by X*’ will be identified as a semantically similar predicate of ‘*X acquire Y*’; 2) we generate semantically similar predicate candidates by replacing each verb/noun in the predicate *p* with its synonyms/hypernyms in WordNet 3.0. If a predicate candidate is a valid predicate (i.e., it is one of the given predicates), we take it as a semantically similar predicate of *p*. For example, (‘*X buy Y*’, ‘*X purchase Y*’) and (‘*X be maker of Y*’, ‘*X be creator of Y*’) will be identified semantically similar using the synonym relations between (*buy*, *purchase*) and between (*maker*, *creator*).

3.2 Context-Sensitive Random Walk Algorithm

In this section, we describe how to accurately represent a predicate in a specific context. Specifically, given a predicate *p*, its context *c* and all features $\{f_1, f_2, \ldots, f_n\}$, we represent predicate *p* as a vector:

$$\vec{v}_p^c = (w_{p1}^c, w_{p2}^c, \ldots, w_{pn}^c)$$

where w_{pi}^c is the relevance score between predicate *p* and feature f_i under context *c*. In following we first develop a context-insensitive random walk algorithm which can estimate context-insensitive relevance score between a predicate *p* and a feature *f*, then we extend the algorithm by taking context into consideration. For simplicity, we assign each node in predicate graph *G=(V, E)* an integer index from 1 to $|V|$, and use it to represent the node.

Context-Insensitive Random Walk. Given a predicate graph *G=(V, E)*, the relevance score between a predicate *p* and a feature *f* can be naturally modelled as the relevance score between the two nodes in *G* corresponding to *p* and *f*. Estimating relevance score between two nodes in a graph is one of the

fundamental tasks in graph mining, and many algorithms have been developed. In this paper we estimate the context-insensitive relevance score between two nodes using one of the most widely used algorithm – *Random Walk with Restart (RWR)* (Tong et al., 2006), which can be fast computed and has been successfully used in many applications, like personalized PageRank (Haveliwala, 2003), image retrieval (He et al., 2004), etc.

Specifically, *RWR* models the relevance score between node *i* and node *j* in a graph *G* as the steady-state probability $r_{i,j}$, i.e., the probability of a random walk starts from node *i* will end at node *j*. For example, the relevance between ('*X acquire Y*', *X*='*Facebook*') in Figure 2 will be computed by starting random walks from the predicate node '*X acquire Y*', then estimate the probability of these random walks ending at the feature node *X*='*Facebook*'.

The random walk used in *RWR* is specified as follows: consider a random particle that starts from node *s* that indicates predicate *p*, at each step the particle iteratively transmits to its neighbourhood with probability that is proportional to their edge weights, and it also has a restart probability $\lambda \in [0, 1]$ to return to the start node *s*:

$$\mathrm{P}(\mathrm{i} \rightarrow \mathrm{j}) = \begin{cases} (1-\lambda)\dfrac{w_{ij}}{\sum_k w_{ik}} & \text{transmit to neighorhood } j \\ \lambda & \text{restart to start node } s \end{cases}$$

where $\mathrm{P}(\mathrm{i} \rightarrow \mathrm{j})$ is the probability of transmit from node *i* to node *j* at each step, and w_{ij} is the edge weight between node *i* and node *j*. *RWR* can also be written in matrix form:

$$\vec{r}_s = (1-\lambda)\mathbf{M}\vec{r}_s + \lambda\vec{e}_s$$

where $\vec{r}_s$ is the n×1 relevance score vector, with $r_{s,j}$ is the relevance score of node *j* with respect to start node *s*, and $\vec{e}_s$ is n×1 starting vector with the s^{th} element 1 and 0 for others; **M** is the neighbourhood transition matrix with $\mathrm{M_{ij}} = w_{ji}/\sum_k w_{jk}$.

Using *RWR*, the relevance score between a predicate *p* and a feature *f* can effectively summarize the semantic relevance information between them by exploiting the global structure of predicate graph. For example, in Figure 2 all the paths between '*X buy Y*' and *X*='*Facebook*' will be used to estimate the relevance score between them, such as the direct edge '*X buy Y*'—*X*='*Facebook*' and the indirect path '*X buy Y*'—'*X purchase Y*'—*X*='*Facebook*'. To demonstrate the effect of *RWR*, Table 1 shows the state-steady probability of the random walk starting from '*X acquire Y*'. We can see that *RWR* can effectively exploit both the inter-dependencies between predicates and the predicate-feature co-occurrence information. For example, although '*X acquire Y*' doesn't co-occur with *X*='*Facebook*' in Figure 2, *RWR* can still estimate the relevance score between them as *0.045*.

Context / Feature	No Context	X=Microsoft Y=Nokia	X=people Y=language
X=Facebook	*0.045*	*0.055*	*0.003*
Y=WhatsApp	*0.045*	*0.055*	*0.003*
X=Google	*0.064*	*0.092*	*0.073*
Y=YouTube	*0.064*	*0.092*	*0.073*
X=children	*0.119*	*0.080*	*0.163*
Y=skill	*0.119*	*0.080*	*0.163*

Table 1. The representations of 'X acquire Y' in different contexts (λ=0.1 and semantic-dependent edge weight = 0.5)

Context-Sensitive Random Walk. The main problem of the above random walk algorithm is that it is context-insensitive, cannot accurately represent a predicate in different contexts. For example, the above algorithm will return the same representation for '*X acquire Y*' in contexts (*Microsoft, Nokia*) and (*people, language*), although it corresponds to different senses of *acquire*.

To learn context-specific predicate representations, we extend *RWR* algorithm by also taking context into consideration. Specifically, the start point of our algorithm is to distinguish context relevant information from context irrelevant information. For example, to represent '*X acquire Y*' in the context (*peo-*

ple, language), the features *X*='*Facebook*', *X*='*Google*', *Y*='*WhatsApp*' and *Y*='*YouTube*' will be identified as context-irrelevant and their relevance scores will be reduced, meanwhile the features *X*='*children*' and *Y*='*skill*' will be identified as context-relevant and their relevance scores will be increased. To achieve the above goal, we revise the transition probability of *RWR* using a context-sensitive node-dependent restart probability:

$$\mathrm{P}(\mathrm{i} \rightarrow \mathrm{j}|\mathrm{c}) = \begin{cases} (1-\lambda_{c,i})\dfrac{w_{ij}}{\sum_i w_{ik}} & \text{transmit to neighorhood } j \\ \lambda_{c,i} & \text{restart to start node } s \end{cases}$$

where $\lambda_{c,i}$ is the restart probability at node *i* in context *c*, which depends on the context relevance between node *i* and context *c*. For instance, in Figure 1, to learn the representation of '*X acquire Y*' in context (*people, language*), our method will set a high restart probability to context-irrelevant nodes *X='Facebook', X='Google', Y='WhatsApp'* and *Y='YouTube'*, in contrast our method will set a low restart probability to context-relevant nodes *X='children'* and *Y='skill'*. Based on the context-sensitive random walk, we can easily identify context-relevant information: once a random walk hits a context-irrelevant node, it will jump to the start node, then the relevance scores of all nodes which are semantically similar to the context-irrelevant node will be reduced. The context-sensitive random walk algorithm can also be written in matrix form:

$$\vec{r}_s^{\,c} = \mathbf{M}(\mathbf{I}-\mathbf{\Lambda})\vec{r}_s^{\,c} + (\vec{1}\,\mathbf{\Lambda}\,\vec{r}_s^{\,c})\vec{e}_s$$

where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_{c,1}, \lambda_{c,2}, \ldots, \lambda_{c,n})$ is the diagonal matrix of node-dependent restart probabilities, **I** is the identity matrix and $\vec{1}$ is a 1×n vector with all entries 1.

To compute the context-sensitive node-dependent restart probability $\lambda_{c,i}$, we first measure the context relevance between a feature *f* and context *c*. In this paper, the context of a predicate *p* is its variable instantiation (*X=x, Y=y*), such as (*X*='*Microsoft*', *Y*='*Nokia*') for '*X acquire Y*'. Then we measure the context relevance using the word similarity between feature *f* and the corresponding argument of context *c*:

$$\mathrm{CR}(\mathrm{f},\mathrm{c}) = \mathrm{Sim}(\mathrm{f_w}, \mathrm{c_{fs}})$$

where f_w is the word content of feature *f* (e.g., *people* for *X*='*people*'), *fs* is the slot signature of feature *f* (e.g., *X* for *X*='*people*'), and c_{fs} is the word in the slot *fs* of context *c*. In this paper, the similarity between two words is the cosine similarity between their word vectors (Pennington et al., 2014), using a publicly available pre-trained word vectors[1].

Finally, the context-sensitive node-dependent restart probability of node *i* is computed as:

$$\lambda_{\mathrm{i,c}} = \begin{cases} \lambda + \beta\,(1-\lambda)\big(1.0 - \mathrm{CR}(\mathrm{i},\mathrm{c})\big) & \text{if i is a feature} \\ \lambda & \text{if i is a predicate} \end{cases}$$

where λ is the global restart probability used for smoothing, β is used to control the impact of context relevance in context-sensitive random walk, which will be empirically tuned.

Table 2 shows the learned context-specific representations of '*X acquire Y*' in different contexts. We can see that our algorithm can effectively learn context-specific representations: the most important features are *X='Google'* and *Y='Youtube'* in context (*X*='*Microsoft, Y*='*Nokia*'), by contrast the most important features are *X='children'* and *Y='skill'* in context (*X*='*people*', *Y*='*language*').

3.3 Context-Sensitive Inference Rule Discovery

Based on the above algorithm, each predicate in a specific context is represented as the context-specific steady-state probability vector $\vec{r}_s^{\,c}$. To discover inference rules, we first compute similarities between predicates, then two predicates *p* and *q* in context *c* will form an inference rule if their similarity is above a threshold. Specifically, because each representation $\vec{r}_s^{\,c}$ can be viewed as a distribution over nodes, we measure the similarity between two predicates using the Kullback–Leibler divergence between $\vec{r}_p^{\,c}$ and $\vec{r}_q^{\,c}$ (Kullback & Leibler, 1951):

[1] http://www-nlp.stanford.edu/data/glove.840B.300d.txt.gz

$$\mathrm{KL}(\vec{r}_p^c|\vec{r}_q^c) = \sum_i r_{p,i}^c \times \ln(\frac{r_{p,i}^c}{r_{q,i}^c})$$

Notice that KL divergence is a distance measure: the smaller the KL divergence between $\vec{r}_p^c$ and $\vec{r}_q^c$, the more similar the two predicates p and q.

4 Experiments

In this section, we evaluate the performance of our method and compare it with traditional methods.

4.1 Experimental Settings

Corpus. In this paper, we use the ReVerb corpus (Fader et al., 2011) as the inference rule discovery corpus, which contains about 15 million publicly available unique open extractions. Each extraction in ReVerb is an instantiation of a predicate in the form (x, predicate, y), such as (*Facebook, acquire, Instagram*) and (*Paris, is capital of, France*). Before inference rule discovery, we apply some clean-up preprocessing to the ReVerb extractions: we remove all predicates occurring in less than 50 times and all arguments occurring in less than 10 times.

Evaluation. For evaluation, we use the publicly available dataset constructed by Zeichner et al. (2015)[2]. The dataset contains 6567 instantiated inference rules, where each one is manually labeled as correct or incorrect. For example, '*X be crucial to Y → X be important in Y*' is labeled as correct with instantiation (*oil prices, decisions*), and '*X own Y → X purchase Y*' is labeled as incorrect with instantiation (*we, these items*). For evaluation, we remove all inference rules whose predicates are not within the *ReVerb* corpus. Finally the evaluation dataset contains 5688 inference rules (2213 are correct and 3475 are incorrect). We split the dataset randomly in 2 subsets: 80% for testing and 20% for validating.

To assess the performance of different methods, we compute similarity scores for all annotated testing inference rules using different methods, and outputted the ranked inference rules of different methods using their similarity scores.

As the same as Melamud et al. (2013), we compare different methods by measuring Mean Average Precision (MAP) (Manning et al., 2008) of the inference rule ranking outputted by different methods. To compute MAP values and corresponding statistical significance, we randomly split test set into 30 subsets and computed Average Precision on every subset, the average over all subsets are used as the final MAP value.

Baselines. We compare our method with three types of inference rule discovery methods:

1) We evaluate two distributional similarity based context-insensitive baselines. One follows the DIRT similarity in (Lin and Pantel, 2001), we denote it as *DIRT*. The other uses the *Balanced-Inclusion* similarity in (Szpektor and Dagan, 2008), we denote it as *BINC*.
2) We evaluate a latent topic model based context-sensitive method. We follow the method described in Melamud et al. (2013), a two level model which computes context-sensitive similarity using two predicates' word-level vectors biased by topic-level context representations. We apply their method on two base word-level similarities, the *LIN* similarity and the *BINC* similarity, correspondingly denoted as *WT-LIN* and *WT-BINC*.
3) We evaluate the global learning method proposed in Berant et al. (2011), which use ILP solvers to performance global optimization over local classification results—We denote it as *ILP*. For comparison, we directly use the inference rule resource[3] released by Berant et al. (2011), which was also learned from the *ReVerb* corpus.

For our graph-based method, we tune its parameters on the validating dataset, and the final parameters used in our method are as follows: the global restart probability λ=0.1, the weight of the semantic dependent edge α = 4.0, and the context relevance restart weight β=0.7.

[2] http://u.cs.biu.ac.il/~nlp/resources/downloads/annotation-of-rule-applications/
[3]http://www-nlp.stanford.edu/joberant/homepage_files/resources/ACL2011Resource.zip

4.2 Experimental Results and Discussions

We conduct experiments on the test dataset using all baselines. For our method, we use two different settings: one uses context-insensitive random walk – we denote it as *RWR-CI*, and the other uses context-sensitive random walk—we denote it as *RWR-CS*. The overall results are presented in Table 2.

System	MAP
DIRT	0.401
BINC	0.424
WT-LIN	0.482
WT-BINC	0.500
ILP	0.513
RWR-CI	0.511
RWR-CS	**0.576**

Table 2. The overall results of different methods

From Table 2, we can see that:

1) By taking both the context and the inter-dependencies between predicates into consideration, our method can achieve significant performance improvement over traditional methods. Compared with the distributional similarity based baselines *DIRT* and *BINC*, *RWR-CS* achieved 44% and 36% MAP improvements. Compared with the latent topic model based context-sensitive baselines *WT-LIN* and *WT-BINC*, *RWR-CS* achieved 20% and 15% MAP improvements. Compared with the global learning baseline *ILP*, *RWR-CS* achieved 12% MAP improvement.
2) Context-sensitive similarity is critical for inference rule discovery. By taking the context into consideration, *WT-LIN*, *WT-BINC* and *RWR-CS* correspondingly achieved 20%, 18% and 13% MAP improvements over their context-insensitive counterparts—*DIRT*, *BINC* and *RWR-CI*.
3) The predicate inter-dependency can enhance the performance of inference rule discovery. By taking advantage of the rich inter-dependencies, both *ILP* and *RWR-CI* achieve performance improvements over the two baselines which model predicates independently: *DIRT* and *BINC*.

To better understand the reasons why and how the graph-based method works well, we evaluate our method using different settings. The results are presented in Table 3.

	Context-Insensitive Random Walk	Context-Sensitive Random Walk
Co-occurrence Edges	0.506	0.547
+ Semantic-Dependent Edges	0.511	0.576

Table 3. The results of the different settings of our method

From Table 3, we can see that:

1) The context-sensitive random walk algorithm can effectively capture the semantics of a predicate in a specific context: Using context-sensitive random walk algorithm, our method achieves MAP improvements on both predicate graph settings (co-occurrence edges only and all edges).
2) The predicate inter-dependency and the context-sensitive random walk can reinforce each other: our method can achieve a 14% MAP improvement by both adding semantic-dependent edges and performing context-sensitive random walk, which is larger than the sum of the performance improvements by only adding semantic-dependent edges (1% improvement) and by only performing context-sensitive random walk (8% improvement). We believe this is because although the inter-dependencies between predicates can enrich predicate representation with more information, it may also introduce some irrelevant information. As a complement, the context-sensitive random walk can filter out irrelevant information and retain only relevant information.

5 Conclusions and Future Work

This paper proposes a graph-based method for context-sensitive inference rule discovery. The advantages of our method are: 1) our method is context-sensitive, it can accurately represent the semantics of a predicate in a specific context; 2) our method can take advantage of the inter-dependencies between predicates for better predicate representation. Experiments verified the effectiveness of our method.

In future work, we aim to jointly model inference rule discovery and knowledge base completion, so that inference rules can be exploited to complete a knowledge base and the semantic knowledge in the given knowledge base can be used to enhance inference rule discovery. Furthermore, we also want to learn the distributed representations of predicates using deep neural networks.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants no. 61572477, 61433015 and 61272324, and the National High Technology Development 863 Program of China under Grants no. 2015AA015405. Moreover, we sincerely thank the reviewers for their valuable comments.

Reference

Berant, J., Dagan, I. and Goldberger, J. 2010. *Global learning of focused entailment graphs*. In: Proceedings of ACL 2010.

Berant, J., Dagan, I. and Goldberger, J. 2011. *Global learning of typed entailment rules*. In: Proceedings of ACL 2011.

Bhagat, R., Pantel, P., Hovy, E. and Rey, M. 2007. *LEDIR: An unsupervised algorithm for learning directionality of inference rules*. In: Proceedings of EMNLP-CoNLL 2007.

Dagan, I., Glickman, O. and Magnini, B. 2006. *The pascal recognizing textual entailment challenge*. In: Lecture Notes in Computer Science, 3944:177-190.

Dinu, G. and Lapata, M. 2010. *Measuring distributional similarity in context*. In: Proceedings of EMNLP 2010.

Dutta, A., Meilicke, C. and Stuckenschmidt, H. 2015. *Enriching Structured Knowledge with Open Information*. In: Proceedings of WWW 2015.

Fader, A., Soderland, S. and Etzioni, O. 2011. *Identifying relations for open information extraction*. In: Proceedings of EMNLP 2011.

Hashimoto, C. and Torisawa, K. and Kuroda, K. and De Saeger, S. and Murata, M. and Kazama, J. 2009. *Large-Scale Verb Entailment Acquisition from the Web*. In: Proceedings of EMNLP 2009.

Haveliwala, T. H. 2003. *Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search*. In: IEEE Transactions on Knowledge and Data Engineering.

He, J., Li, M., Zhang, H., Tong, H. and Zhang, C. 2004. *Manifoldranking based image retrieval*. In: Proceedings of ACM Multimedia 2004.

Hearst, M. A. 1992. *Automatic acquisition of hyponyms from large text corpora*. In: Proceedings of COLING 1992.

Kullback, S. and Leibler, R.A. 1951. On information and sufficiency. In: Annals of Mathematical Statistics 22 (1): 79–86.

Lin, D. and Pantel, P. 2001. *DIRT—discovery of inference rules from text*. In: Proceedings of ACM SIGKDD 2001.

Kok, S. and Domingos, P. 2008. *Extracting semantic networks from text via relational clustering*. In: Machine Learning and Knowledge Discovery in Databases, pp. 624-639. Springer Berlin Heidelberg, 2008.

Manning, C., Raghavan, P. and Sch¨utze, H. 2008. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge.

Melamud, O., Berant, J., Dagan, I., Goldberger, J. and Szpektor, I. 2013. *A Two Level Model for Context Sensitive Inference Rules*. In: Proceedings of ACL 2013.

Nakashole, N., Weikum, G. and Suchanek, F. 2012. *Patty: A taxonomy of relational patterns with semantic types*. In: Proceedings of EMNLP 2012.

Pantel, P., Bhagat, R., Coppola, B., Chklovski, T. and Hovy, E. 2007. *ISP: Learning inferential selectional preferences*. In: Proceedings of NAACL-HLT 2007.

Pennington, J., Socher, R. and Manning C. 2014. *Glove: Global Vectors for Word Representation*. In: Proceedings of EMNLP 2014.

Ritter, A. and Etzioni, O. 2010. *A latent dirichlet allocation method for selectional preferences*. In: Proceedings of ACL 2010.

Ravichandran, D. and Hovy, E. 2002. *Learning surface text patterns for a question answering system*. In: Proceedings of ACL 2002.

Sekine, S. 2005. *Automatic paraphrase discovery based on context and keywords between NE pairs*. In: Proceedings of IWP 2005.

Szpektor, I., and Dagan, I. 2008. *Learning entailment rules for unary templates*. In: Proceedings of COLING 2008.

Szpektor, I., Dagan, I., Bar-Haim, R. and Goldberger, J. 2008. *Contextual preferences*. In: Proceedings of ACL-HLT 2008.

Szpektor, I., Tanev, H., Dagan, I. and Coppola, B. 2004. *Scaling Web-based Acquisition of Entailment Relations*. In: Proceedings of EMNLP 2004.

Szpektor, I., Shnarch, E. and Dagan, I. 2007. *Instance-based evaluation of entailment rule acquisition*. In: Proceedings of ACL 2007.

Tong, H., Faloutsos, C. and Pan, J.Y. 2006. *Fast random walk with restart and its applications*. In: Proceedings of ICDM 2006.

Yates, A. and Etzioni, O. 2009. *Unsupervised methods for determining object and relation synonyms on the web*. In: Journal of Artificial Intelligence Research 34, no. 1 (2009): 255.

Zeichner, N., Berant, J. and Dagan, I. 2012. *Crowdsourcing inference-rule evaluation*. In Proceedings of ACL 2012.

Modelling Sentence Pairs with Tree-structured Attentive Encoder

Yao Zhou, Cong Liu* and **Yan Pan**
School of Data and Computer Science, Sun Yat-sen University
yoosan.zhou@gmail.com
{liucong3, panyan5}@mail.sysu.edu.cn

Abstract

We describe an attentive encoder that combines tree-structured recursive neural networks and sequential recurrent neural networks for modelling sentence pairs. Since existing attentive models exert attention on the sequential structure, we propose a way to incorporate attention into the tree topology. Specially, given a pair of sentences, our attentive encoder uses the representation of one sentence, which generated via an RNN, to guide the structural encoding of the other sentence on the dependency parse tree. We evaluate the proposed attentive encoder on three tasks: semantic similarity, paraphrase identification and true-false question selection. Experimental results show that our encoder outperforms all baselines and achieves state-of-the-art results on two tasks.

1 Introduction

Modelling a sentence pair is to score two pieces of sentences in terms of their semantic relationship. The applications include measuring the semantic relatedness of two sentences (Marelli et al., 2014), recognizing the textual entailment (Bowman et al., 2015) between the premise and hypothesis sentences, paraphrase identification (He et al., 2015), answer selection and query ranking (Yin et al., 2015) etc.

The approach of modelling a sentence pair based on neural networks usually consist of two steps. First, a sentence encoder transforms each sentence into a vector representation. Second, a classifier receives two sentence representations as features to make the classification. The sentence encoder can be regarded as a semantic compositional function which maps a sequence of word vectors to a sentence vector. This compositional function takes a range of different forms, including (but not limited to) sequential recurrent neural networks (Seq-RNNs) (Mikolov, 2012), tree-structured recursive neural networks (Tree-RNNs) (Socher et al., 2014; Tai et al., 2015) and convolutional neural networks (CNNs) (Kim, 2014).

We introduce an approach that combines recursive neural networks and recurrent neural networks with the attention mechanism, which has been widely used in the sequence to sequence learning (*seq2seq*) framework whose applications ranges from machine translation (Bahdanau et al., 2015; Luong et al., 2015), text summarization (Rush et al., 2015) to natural language conversation (Shang et al., 2015) and other NLP tasks such as question answering (Sukhbaatar et al., 2015; Hermann et al., 2015), classification (Rocktäschel et al., 2016; Shimaoka et al., 2016). In the machine translation, the attention mechanism is used to learn the alignments between source words and target words in the decoding phase. More generally, we consider that the motivation of attention mechanism is to allow the model to attend over a set of elements with the intention of attaching different emphases to each element. We argue that the attention mechanism used in a tree-structured model is different from a sequential model. Our idea is inspired by Rocktäschel et al. (2016) and Hermann et al. (2015). In this paper, we utilise the attention mechanism to select semantically more relevant child by the representation of one sentence learned by a Seq-RNNs, when constructing the head representation of the other sentence in the pair on a dependency tree. Since our model adopts the attention in the sentence encoding phase, we refer to it as an attentive

* indicates the corresponding author.
Code is available at https://github.com/yoosan/sentpair

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2912–2922, Osaka, Japan, December 11-17 2016.

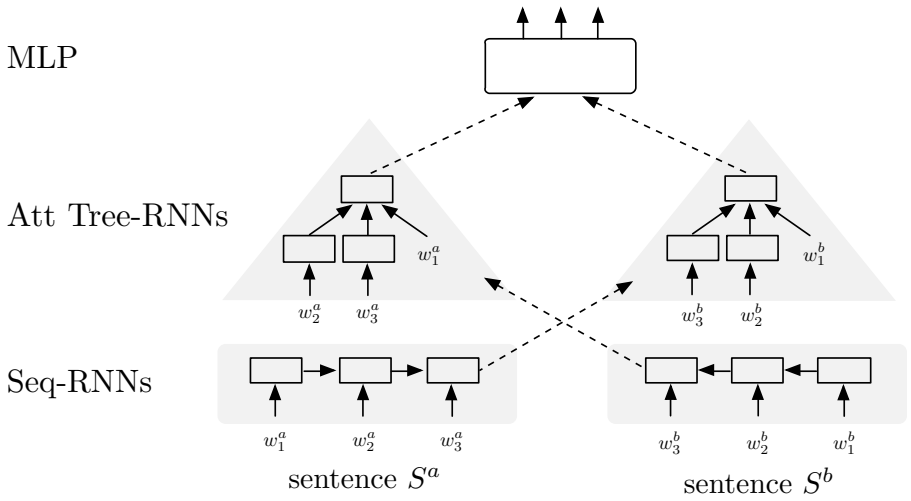


Figure 1: An overview of our tree-structured attentive encoder.

encoder. In this work, we implement this attentive encoder with two architectures: tree-structured LSTM and tree-structured GRU.

We evaluate the proposed encoder on three sentence pair modelling tasks: semantic similarity on the SICK dataset, paraphrase identification on the MSRP dataset and true-false question selection on the AI2-8grade science questions dataset. Experimental results demonstrate that our attentive encoder is able to outperform all non-attentional counterparts and achieves the state-of-the-art performance on the SICK dataset and AI2-8grade dataset.

2 Models

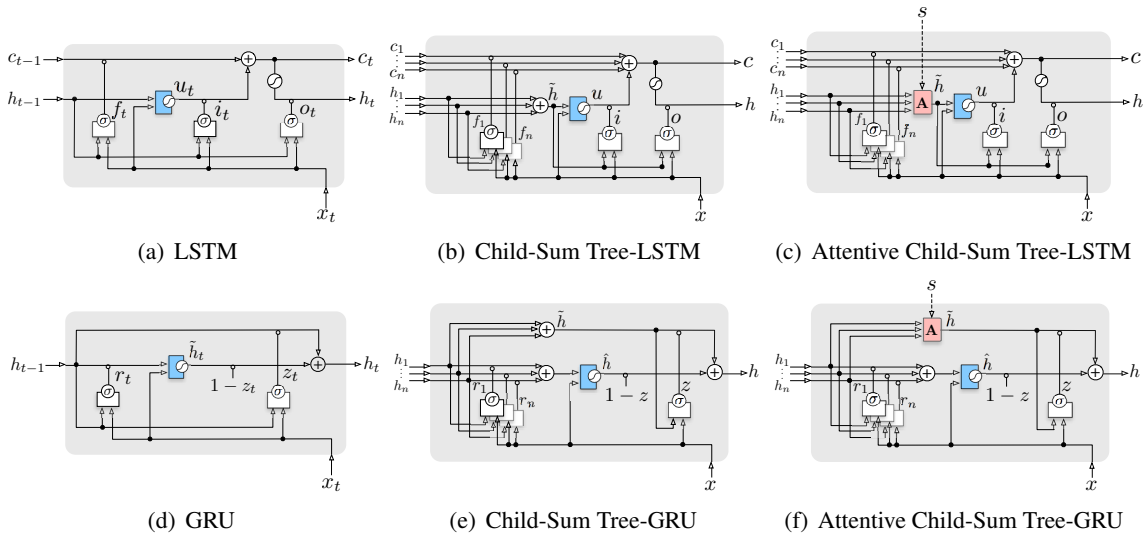
Let's begin with a high-level discussion of our tree-structured attentive encoder. As shown in Figure 1An overview of our tree-structured attentive encoderfigure.1, given a sentence pair (S^a, S^b), our goal is to score this sentence pair. Our tree-structured attentive model has two components. In the first component, a pair of sentences is fed to a Seq-RNNs, which encodes each sentence and results in a pair of sentence representations. In second component, the Attentive Tree-RNNs encodes a sentence again, aimed by the representation of the other sentence generated by the first component. Compared with the existing approaches of modelling sentence pairs, our attentive encoder consider not only the sentence itself but also the other sentence in the pair. Finally, the two sentence vectors produced by the second component are fed to the multilayer perceptron network to produce a distribution over possible values. These components will be detailed in the following sections.

2.1 Seq-RNNs

We first describe the RNN composer, which is the basic unit of Seq-RNNs. Given an input sequence of arbitrary length, an RNN composer iteratively computes a hidden state h_t using the input vector x_t and its previous hidden state h_{t-1}. In this paper, the input vector x_t is a word vector of the t-th word in a sentence. The hidden state h_t can be interpreted as a distributed representation of the sequence of tokens observed up to time t. Commonly, the RNN transition function is the following:

$$h_t = \tanh(Wx_t + Uh_{t-1} + b) \tag{1}$$

We refer to the model that recursively apply the RNN composer to a sequence as the Seq-RNNs. Unfortunately, standard Seq-RNNs suffers from the problem that the gradients of the hidden states of earlier part of the sequence vanishes in long sequences (Hochreiter, 1998). Long Short-term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Chung et al., 2014) are two powerful and popular architectures that address this problem by introducing gates and memory. In this paper, we only show the illustrations of LSTM (Figure 2(a)Subfigure 2(a)subfigure.2.1) and



(a) LSTM (b) Child-Sum Tree-LSTM (c) Attentive Child-Sum Tree-LSTM

(d) GRU (e) Child-Sum Tree-GRU (f) Attentive Child-Sum Tree-GRU

Figure 2: Illustrations of different recurrent architectures.

GRU (Figure 2(d)Subfigure 2(d)subfigure.2.4). The implementations of standard LSTM and GRU in this paper are same as (Luong et al., 2015) and (Chung et al., 2014). When we replace the standard RNN composer with LSTM or GRU, the Seq-RNNs becomes Seq-LSTMs or Seq-GRUs.

2.2 Standard Tree-RNNs

Compared with standard RNN composer, which computes its hidden state from the input at the current time step and the hidden state of previous time step, the Tree-RNN composer computes its hidden state from an input and the hidden states of arbitrarily many child units. We now describe the *Child-Sum Tree-LSTM* and *Child-Sum Tree-GRU* architectures which are formed by applying the *Child-Sum* algorithm to LSTM and GRU respectively.

Child-Sum Tree-LSTM. In this paper, the implementation of Child-Sum Tree-LSTM is same as (Tai et al., 2015). We consider that a Child-Sum Tree-LSTM composer contains two parts: the external part and internal part. The external part consists of the inputs and outputs, and the internal part is the controllers and memory of the composer. As shown in Figure 2(b)Subfigure 2(b)subfigure.2.2, the inputs of the composer are: a input vector x, multiple hidden states $h_1, h_2, \ldots, h_n$ and multiple memory cells $c_1, c_2, \ldots, c_n$, where n is the number of child units. The outputs consist of a memory cell c and a hidden state h which can be interpreted as the representation of a phrase. The internal part aims at controlling the flow of information by an input gate i, an output gate o and multiple forget gates $f_1, f_2, \ldots, f_n$. The gating mechanisms used in the Child-Sum Tree-LSTM are similar to sequential LSTM. Intuitively, the sum of children's hidden states $\tilde{h}$ is the previous hidden state, the forget gate f_k controls the degree of memory kept from that of the child k, the input gate i controls how much the internal input u is updated and the output gate controls the exposure of internal memory c. We define the transition equations as follows:

$$
\begin{aligned}
&\tilde{h} = \textstyle\sum_{1 \le k \le n} h_k, && i = \sigma(W^{(i)}x + U^{(i)}\tilde{h} + b^{(i)}),\\
&o = \sigma(W^{(o)}x + U^{(o)}\tilde{h} + b^{(o)}), && u = \tanh(W^{(u)}x + U^{(u)}\tilde{h} + b^{(u)}),\\
&f_k = \sigma(W^{(f)}x + U^{(f)}h_k + b^{(f)}), && c = i \odot u + \textstyle\sum_{1 \le k \le n} f_k \odot c_k,\\
&h = o \odot \tanh(c), &&
\end{aligned}
\tag{2}
$$

Child-Sum Tree-GRU. The way of Child-Sum Tree-GRU extending to the standard GRU is similar to the way of Child-Sum Tree-LSTM extending to the standard LSTM. Since we only introduce the Child-Sum algorithm applied to the LSTM and GRU, in the following, we omit the "Child-Sum" prefix of Child-Sum Tree-LSTM and Child-Sum Tree-GRU for simplicity. Compared with the Tree-LSTM,

the Tree-GRU removes the memory cell c and introduces an update gate z and multiple reset gates $r_1, r_2, \ldots, r_n$ that allow the composer to reset the hidden states of the child units. The candidate hidden state $\hat{h}$ is computed similarly to the standard RNN (Equation 1Seq-RNNsequation.2.1) and the update gate z is used to control how much the previous hidden state $\tilde{h}$ and the candidate $\hat{h}$ should be passed. The transition equations of Tree-GRU are in the following:

$$\begin{array}{ll} \tilde{h} = \sum_{1 \leq k \leq n} h_k, & r_k = \sigma(W^{(r)}x + U^{(r)}h_k + b^{(r)}), \\ \hat{h} = \tanh(W^{(h)}x + U^{(h)} \sum_{k=1}^{n} r_k \odot h_k), & z = \sigma(W^{(z)}x + U^{(z)}\tilde{h} + b^{(z)}) \\ h = z \odot \tilde{h} + (1 - z) \odot \hat{h}, & \end{array} \tag{3}$$

where σ denotes the sigmoid function and $\odot$ denotes element-wise multiplication.

We can easily apply the Child-Sum Tree-RNN to the dependency trees that have branching factors of arbitrary number and order-insensitive nodes. We refer to the model adopting the Tree-LSTM and Tree-GRU composer to the dependency tree as the *Dependency Tree-LSTMs* and *Dependency Tree-GRUs*. For simplicity, we also omit the prefix "Dependency" in the following sections.

2.3 Attentive Tree-RNNs

We now details how we extend the standard Tree-RNN. The idea that we incorporate the attention into the standard Tree-RNN comes from: (1) there will be semantic relevance between two sentences in the sentence pair modelling tasks; (2) the effect of semantic relevance could be implemented in the process of constructing the sentence representation by Tree-RNN where each child should be assigned a different weight; and (3) the attention mechanism is well suited for learning weights on a contextual collection where a guided vector is attending over.

Soft Attention Layer In this work, the attention mechanism is implemented by a *soft attention layer* A. Given a collection of hidden states $h_1, h_2, \ldots, h_n$ and an external vector s, the soft attention layer produce a weight α_k for each hidden state as well as a weighted vector g via the Equations 4Soft Attention Layerequation.2.4:

$$\begin{array}{ll} m_k = \tanh(W^{(m)}h_k + U^{(m)}s), & \alpha_k = \frac{\exp(w^\intercal m_k)}{\sum_{j=1}^{n} \exp(w^\intercal m_j)}, \\ g = \sum_{1 \leq k \leq n} \alpha_k h_k, & \end{array} \tag{4}$$

Attentive Tree-LSTM and -GRU As illustrated in Figure 2(c)Subfigure 2(c)subfigure.2.3 and Figure 2(f)Subfigure 2(f)subfigure.2.6, when the attention layer is embedded to the standard Tree-LSTM and Tree-GRU composer, these composers become the Attentive Tree-LSTM and Attentive Tree-GRU. The attention layer (notated with A) receives the children's hidden states and an external vector s, producing a weighted representation g (Equation 4Soft Attention Layerequation.2.4). In our implementation, the external vector s is a vector representation of sentence learned by a Seq-RNNs. Specifically, if the tree composer is Attentive Tree-LSTM, then the Seq-RNNs is Seq-LSTMs. Instead of taking the sum of children's hidden states as the previous hidden state $\tilde{h}$ in the standard Tree-RNN, we compute a new hidden state by a transformation $\tilde{h} = \tanh(W^{(a)}g + b^{(a)})$. Similar to the standard Tree-RNN, the attentive composers can also be easily applied to dependency trees. We refer to the model which applies the Attentive Tree-RNN composer to the dependency tree as the Attentive (Dependency) Tree-RNNs.

2.4 MLP

The multilayer perceptron network (MLP) receives a pair of vectors produced by the sentence encoder to compute a multinomial distribution over possible values. Given two sentence representations h_L and h_R, we compute their componentwise product $h_L \odot h_R$ and their absolute difference $|h_L - h_R|$. These features are also used by Tai et al.(2015). We then compress these features into a low dimensional vector

h_s, which is used to compute the probability distribution $\hat{p}_\theta$. The equations are the following:

$$
\begin{aligned}
h_\times &= h_L \odot h_R, \quad h_+ = |h_L - h_R|, \\
h_s &= \sigma(W^{(\times)} h_\times + W^{(+)} h_+ + b^{(h)}), \\
\hat{p}_\theta &= \text{softmax}(W^{(p)} h_s + b^{(p)}),
\end{aligned} \tag{5}
$$

3 Experiments and Results

Config	Value	Config	Value
Word vectors	Glove (Pennington et al., 2014)	Dims of word vectors	300
OOV word vectors	uniform(-0.05, 0.05)	Dims of hidden state	150
Learning rate	0.05	Batch size	25
Regularization	L2 with $\lambda = 10^{-4}$	Dropout rate	0.5
Optim method	Adagrad (Duchi et al., 2011)	Num of epoch	10

Table 1: The training configs. We use the 300D Glove vectors as the initial word vectors. Out-of-vocabulary words are initialized with a uniform distribution. The model parameters are regularized with a per-minibatch L2 regularization strength of 10^{-4}. The dropout is used at the classifier with a dropout rate 0.5. All models are trained using Adagrad with a learning rate of 0.05. We train our models for 10 epochs, and pick the model that has the best results on the devlopment set to evaluate on the test set.

Our baselines In order to make a meaningful comparison between the sequential models, tree-structured models and attentive models, we present four baselines. They are: (i) **Seq-LSTMs**, learning two sentence representations by the sequential LSTMs; (ii) **Seq-GRUs**, like Seq-LSTMs but using GRU composer; (iii) **Tree-LSTMs**, learning two sentence representations by the Dependency Tree-LSTMs; and (iv) **Tree-GRUs**, like Tree-LSTMs but using Child-Sum Tree-GRU composer. The two sentence representations are fed to the MLP to produce a probability distribution.

3.1 Task 1: Semantic Similarity

First we conduct our semantic similarity experiment on the Sentences Involving Compositional Knowledge(SICK) dataset (Marelli et al., 2014)[12]. This task is to predict a similarity score of a pair of sentences, based on human generated scores. The SICK dataset consists of 9927 sentence pairs with the split of 4500 training pairs, 500 development pairs and 4927 testing pairs. Each sentence pair is annotated with a similarity score ranging from 1 to 5. A high score indicates that the sentence pair is highly related. All sentences are derived from existing image and video annotation dataset. The evaluation metrics are Pearson's r, Spearman's ρ and mean squared error (MSE).

Recall that the output of MLP (Section 2.4MLPsubsection.2.4) is a probability distribution $\hat{p}_\theta$. Our goal in this task is to predict a similarity score of two sentences. Let $r^\intercal = [1, \ldots, 5]$ be an integer vector, the similarity score $\hat{y}$ is computed by $\hat{y} = r^\intercal \hat{p}_\theta$. We take the same setup as (Tai et al., 2015) that computes a target distribution p as a function of prediction score y given by:

$$
p_i = \begin{cases} y - \lfloor y \rfloor, & i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & i = \lfloor y \rfloor \\ 0 & \text{otherwise} \end{cases}
$$

The loss function of semantic similarity is the KL-divergence that measures the continuous distance between the predicted distribution $\hat{p}_\theta$ and the distribution of the ground truth p:

$$
J(\theta) = \frac{1}{N} \sum_{k=1}^{N} \text{KL}(p^{(k)} \Big|\Big| \hat{p}_\theta^{(k)}) + \frac{\lambda}{2} ||\theta||_2^2 \tag{6}
$$

Method	r	ρ	MSE
ECNU (Zhao et al., 2014)	0.8414	-	-
Dependency Tree-LSTMs (Tai et al., 2015)	0.8676	0.8083	0.2532
combine-skip+COCO (Kiros et al., 2015)	0.8655	0.7995	0.2561
ConvNet (He et al., 2015)	0.8686	0.8047	0.2606
Seq-GRUs	0.8595	0.7974	0.2689
Seq-LSTMs	0.8528	0.7911	0.2831
(Dependency) Tree-GRUs	0.8672	0.8116	0.2573
(Dependency) Tree-LSTMs(ours)	0.8664	0.8068	0.2610
+Attention			
Attentive (Dependency) Tree-GRUs	0.8701	0.8085	0.2524
Attentive (Dependency) Tree-LSTMs	**0.8730**	**0.8117**	**0.2426**

Table 2: Test set results on the SICK dataset. The first group is previous results, and remaining is ours.

The results are summarized in Table 2Test set results on the SICK dataset. The first group is previous results, and remaining is ourstable.2. We first compare our results against the previous results. ECNU (Zhao et al., 2014), the best result of SemEval 2014 submissions, achieves a 0.8414 r score by a heavily feature-engineered approach. Kiros et al. (2015) presents an unsupervised approach to learn the universal sentence vectors without depending on a specific task. Their Combine–skip+COCO model improve the Pearson's r to 0.8655, but a weakness is that their sentence vectors are high-dimensional vectors (2400D). Training the skip-thoughts vectors needs a lot of time and space. He et al. (2015) show the effectiveness of convolutional nets with the similarity measurement layer for modelling sentence similarity. Their ConvNet outperforms ECNU with +0.027 Pearson's r. We can observe that dependency Tree-LSTM, combine-skip+COCO and ConvNet almost achieve the same performance and our Attentive Tree-LSTMs outperforms these three methods around +0.005 points. Comparison to ECNU, our Attentive Tree-LSTMs gains an improvement of +0.032 and achieves the state-of-the-art performance. We find a phenomenon also appeared in (Tai et al., 2015) that tree-structured models can outperform sequential counterparts. Comparison to the non-attentional baselines (such as Tree-LSTMs), the attention mechanism (such as Attentive Tree-LSTMs) gives us a boost of around +0.007. All results highlight that our attentive Tree-RNNs are well suited for the semantic similarity task.

3.2 Task 2: Paraphrase Identification

The next task we evaluate is paraphrase identification on the Microsoft Research Paraphrase Corpus (MSRP) (Dolan et al., 2004). Given two sentences, this task is to predict whether or not they are paraphrases. The dataset is collected from news sources and contains 5801 pairs of sentences, with 4076 for training and the remaining 1725 for testing. We randomly select 10% of training set and use them as our dev set. This task is a binary classification task, therefore we report the accuracy and F1 score.

Since that the $\hat{p}_\theta$ indicates the distribution over the possible labels, we take $argmax(\hat{p}_\theta)$ as the predicted label in the testing phase. The loss function for the binary classification is the *binary cross-entropy*:

$$J(\theta) = -\frac{1}{N}\sum_{k=1}^{N}(y^{(k)}\log\hat{p}_\theta^{(k)} + (1-y^{(k)})\log(1-\hat{p}_\theta^{(k)})) + \frac{\lambda}{2}||\theta||_2^2 \quad (7)$$

Table 3The test results of paraphrase identification on the Microsoft Paraphrase Corpus (**Left**) and true-false selection on the AI2-8grade dataset (**Right**)table.3 (**left**) presents our results on the MSRP dataset. The previous approaches are: (1) Baseline, cosine similarity with tf-idf weighting; (2) RAE,

[1]Dependency trees are parsed by the Stanford Parser package, `http://nlp.stanford.edu/software/lex-parser.html`

[2]Glove vectors are available at `http://nlp.stanford.edu/projects/glove/`

Method	Acc(%)	F1(%)
Baseline (Mihalcea et al., 2006)	65.4	75.3
RAE (Socher et al., 2011)	76.8	83.6
combine-skip+feats (Kiros et al., 2015)	75.8	83.0
ABCNN-3 (Yin et al., 2015)	78.9	84.8
TF-KLD (Ji and Eisenstein, 2013)	**80.4**	**85.9**
Seq-GRUs	71.8	80.2
Seq-LSTMs	71.7	80.6
Tree-GRUs	73.6	81.8
Tree-LSTMs	73.5	82.1
+Attention		
Attentive Tree-GRUs	74.8	82.3
Attentive Tree-LSTMs	**75.8**	**83.7**

Method	Dev Acc(%)	Test Acc(%)
RNN (Baudis et al., 2016)	38.1	36.1
CNN (Baudis et al., 2016)	44.2	38.4
RNN-CNN (Baudis et al., 2016)	43.9	37.6
attn1511 (Baudis et al., 2016)	38.4	35.8
Ubu.RNN (Baudis et al., 2016)	49.4	44.1
Seq-GRUs	72.1	62.4
Seq-LSTMs	71.8	63.3
Tree-GRUs	75.2	70.6
Tree-LSTMs	74.6	69.1
+Attention		
Attentive Tree-GRUs	**76.4**	72.1
Attentive Tree-LSTMs	76.2	**72.5**

Table 3: The test results of paraphrase identification on the Microsoft Paraphrase Corpus (**Left**) and true-false selection on the AI2-8grade dataset (**Right**).

recursive autoencoder with dynamic pooling; (3) combine-skip+feats, skip-thought vectors with features; (4) ABCNN-3, attention-based convolutional nets; and (5) TF-KLD, matrix factorization with supervised reweighting. First, all our models are able to outperform the baseline. We only compare our models with the neural networks-based approaches, including RAE and ABCNN-3 for a fair comparison. We find that our models do not prove to be very competitive. After a careful analysis, we conclude that the reasons are (1) our models are pure neural networks-based, we don't add any features to identify paraphrases while the other methods have used additional features; (2) The MLP is not very suitable in this task. We attempt to replace the MLP with the cosine distance and euclidean distance in our future work. Although our models have not yet matched the SOTA performance, we obtain an improvement of +2.3 accuracy by Attentive Tree-LSTMs when we incorporate the attention into the standard Tree-LSTM.

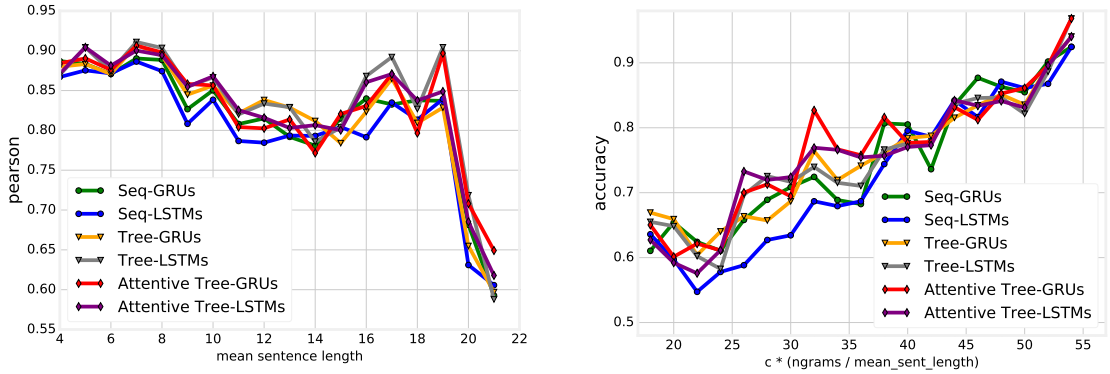
3.3 Task 3: True-False Question Selection

We last consider a challenging task: selecting true or false given a scientific question and its evidence. In this task, we use the AI2-8grade dataset built by (Baudis et al., 2016). This dataset is derived from the *AI2 Elementary School Science Questions* released by Allen Institute. Each sentence pair consists of a hypothesis sentence processed by substituting the wh-word in the question by answer and its evidence sentence extracted from a collection of CK12 textbooks. The number of sample pairs in the training, development, and test set are 12689, 2483 and 11359 respectively. This dataset contains 626 words not appearing in *Glove vectors*, most of which are named entities and scientific jargons.

The loss function is the same as the paraphrase identification since this task is also a binary classification task. We reports the accuracy on development set and test set shown in Table 3The test results of paraphrase identification on the Microsoft Paraphrase Corpus (**Left**) and true-false selection on the AI2-8grade dataset (**Right**)table.3 (**right**). Since this dataset is a fresh and uncompleted dataset, we only compare our models with Baudis et al. (2016) who have evaluated several models on it. Comparison to (Baudis et al., 2016), all of our models gain a significant improvement. Specially, our best result achieved by the Attentive Tree-LSTMs is higher than the best of (Baudis et al., 2016) by +28 percents. It is observed that tree-structured models are more competitive than the sequential counterparts. As we expected, the attentive models can outperform all non-attentional counterparts.

4 Quantitative Analysis

Example Analysis Table 4Example predictions from the test set. **GT**: ground truth, **Pred**: predicted scoretable.4 presents example predictions that are produced by our Attentive Tree-LSTMs. The first group shows that our model is able to predict semantic similarity score nearly perfectly on the SICK dataset. We argue the reason is that the sentences of SICK dataset are image and video descriptions whose sentence structure is relatively simple and there are less uncommon words and named entities in the vocabulary. The second group gives us three examples on the MSRP test set. We find that our model can identify whether two fact statements are paraphrases, but fails to recognize the numbers (in group 2, line 3). We presents the examples on AI2-8grade dataset in the last group. We can observe that our model is efficient to select the false questions, while our model is difficult to select the true answers,



(a) SICK dataset (b) MSRP dataset

Figure 3: Qualities of different models based on mean sentence length and n-grams overlap

Dataset	Sentence 1	Sentence 2	GT	Pred
SICK	The black dog is playing with the brown dog on the sand	A black dog is playing with a brown dog on the sand	4.8	4.8
	A brown dog and a black dog are playing in the sand	The black dog is playing with the brown dog on the sand	5.0	4.2
	A brown dog and a black dog are playing in the sand	A black dog is attacking a brown dog on the sand	3.5	3.4
MSRP	The study is being published today in the journal Science.	Their findings were published today in Science.	1	1
	The launch marks the start of a new golden age in Mars exploration.	The launch marks the start of a race to find life on another planet.	1	1
	Last year, Comcast signed 1.5 million new digital cable subscribers.	Comcast has about 21.3 million cable subscribers, many in the largest U.S. cities.	0	1
AI2	Sunlight is the nutrient source for some fungi ?	The main difference between plants and fungi is how they obtain energy.	0	0
	Sunlight is the nutrient source for some fungi ?	Plants are autotrophs, meaning that they make their own "food" using the energy from sunlight.	0	0
	Sunlight is the nutrient source for some fungi ?	Fungi are heterotrophs, which means that they obtain their "food" from outside of themselves.	0	0
	Dead organisms is the nutrient source for some fungi ?	Most fungi live in soil or dead matter, and in symbiotic relationships with plants, animals, or other fungi.	1	0
	Dead organisms is the nutrient source for some fungi ?	Relate the structure of fungi to how they obtain nutrients.	1	0
	Dead organisms is the nutrient source for some fungi ?	From dead plants to rotting fruit.	1	1

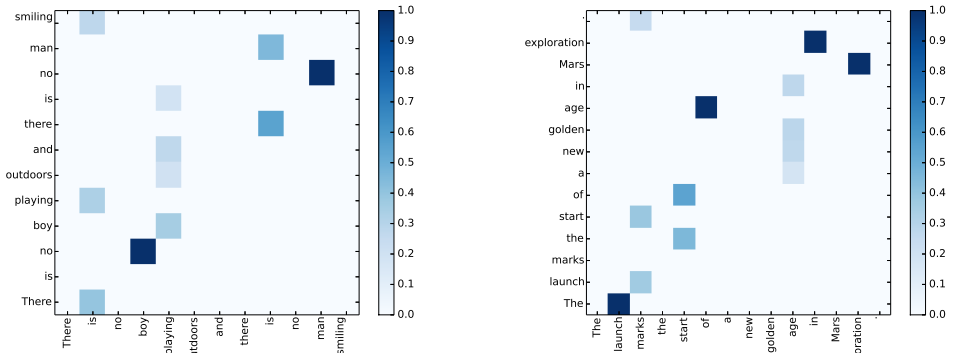
Table 4: Example predictions from the test set. **GT**: ground truth, **Pred**: predicted score.

unless the evidence of question is very strong.

Effect of Sentence Length In order to analyse the effect of mean sentence length on the SICK dataset, we draw the Figure 3(a)Subfigure 3(a)subfigure.3.1. We observe that the Pearson score become lower as sentence become longer. Compared with the Seq-RNNs, the Tree-RNNs obtain a little improvements. Specially, the Attentive Tree-GRUs proves to be more effective than Tree-GRUs when the mean sentence length reaches to 20.

Effect of N-grams In the MSR paraphrase corpus, a hypothesis is that two sentence tend to be paraphrases when the value of their n-gram overlap is high. As a result we present the Figure 3(b)Subfigure 3(b)subfigure.3.2, x-axis is the normalized n-grams overlap whose value is computed by $c* \frac{(unigram+bigram+trigram)}{mean\_sent\_length}$, where c equals to 50, and y-axis is the accuracy. We can observe that the Attentive Tree-GRUs are more effective than Tree-GRUs when the value of normalized n-grams overalp is less than 40. The results suggest that our attentive models are more general.

Attention Visualization It is instructive to analyse which child the attentive model is attending over when constructing the head representation. We visualize the heatmaps of attention weights shown in Figure 4Heatmap of attention weightsfigure.4. The words at x-axis are modified by the words at y-axis with a weight (greater than *zero*). For example in Figure 4(a)Subfigure 4(a)subfigure.4.1, the 5th word at x-axis is "playing" whose children are "boy", "outdoors", "and" and "is". We can observe that the word "boy" holds a higher weight among all the modifiers. It means that the branch rooted with "boy" contributes more when constructing the representation of subtree whose root node is "playing". This phenomenon is very reasonable because the sentence is describing a image of "a **boy** is **playing**



(a) (b)

Figure 4: Heatmap of attention weights.

something".

5 Conclusion

In this paper, we introduced a way of incorporating attention into the Child-Sum Tree-LSTM and Tree-GRU that can be applied to the dependency tree. We evaluate the proposed models on three sentence pair modelling tasks and achieve state-of-the-art performance on two of them. Experiment results show that our attentive models are effective for modelling sentence pairs and can outperform all non-attentional counterparts. In the future, we will evaluate our models on the other sentence pair modelling tasks (such as RTE) and extend them to the $seq2seq$ learning framework.

Acknowledgements

This work was funded in part by the National Key Research and Development Program of China (2016YFB0201900), the National Science Foundation of China (grant 61472459, 61370021, U1401256, 61472453), Natural Science Foundation of Guangdong Province under Grant S2013010011905.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Petr Baudis, Silvestr Stanko, and Jan Sedivy. 2016. Joint learning of sentence embeddings for relevance and entailment. *arXiv preprint arXiv:1605.04655*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Junyoung Chung, Çalar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. Technical Report Arxiv report 1412.3555, Université de Montréal. Presented at the Deep Learning workshop at NIPS2014.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

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

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 891–896.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval-2014*.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780.

Tomáš Mikolov. 2012. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, volume 14, pages 1532–43.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *International Conference on Learning Representations*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1577–1586.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, San Diego, California, USA, June. to appear.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc.

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

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Jiang Zhao, Tian Tian Zhu, and Man Lan. 2014. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. *Proceedings of the SemEval*, pages 271–277.

Neural Paraphrase Generation with Stacked Residual LSTM Networks

Aaditya Prakash[1,2], **Sadid A. Hasan**[2], **Kathy Lee**[2], **Vivek Datla**[2],
Ashequl Qadir[2], **Joey Liu**[2], **Oladimeji Farri**[2]
[1]Brandeis University, Waltham, MA, USA
[2]Artificial Intelligence Laboratory, Philips Research North America, Cambridge, MA, USA
{aprakash,aaditya.prakash}@{brandeis.edu,philips.com}
{sadid.hasan,kathy.lee_1,vivek.datla}@philips.com
{ashequl.qadir,joey.liu,dimeji.farri}@philips.com

Abstract

In this paper, we propose a novel neural approach for paraphrase generation. Conventional paraphrase generation methods either leverage hand-written rules and thesauri-based alignments, or use statistical machine learning principles. To the best of our knowledge, this work is the first to explore deep learning models for paraphrase generation. Our primary contribution is a stacked residual LSTM network, where we add residual connections between LSTM layers. This allows for efficient training of deep LSTMs. We evaluate our model and other state-of-the-art deep learning models on three different datasets: PPDB, WikiAnswers, and MSCOCO. Evaluation results demonstrate that our model outperforms sequence to sequence, attention-based, and bidirectional LSTM models on BLEU, METEOR, TER, and an *embedding*-based sentence similarity metric.

1 Introduction

Paraphrasing, the act to express the same meaning in different possible ways, is an important subtask in various Natural Language Processing (NLP) applications such as question answering, information extraction, information retrieval, summarization and natural language generation. Research on paraphrasing methods typically aims at solving three related problems: (1) recognition (i.e. to identify if two textual units are paraphrases of each other), (2) extraction (i.e. to extract paraphrase instances from a thesaurus or a corpus), and (3) generation (i.e. to generate a reference paraphrase given a source text) (Madnani and Dorr, 2010). In this paper, we focus on the paraphrase generation problem.

Paraphrase generation has been used to gain performance improvements in several NLP applications, for example, by generating query variants or pattern alternatives for information retrieval, information extraction or question answering systems, by creating reference paraphrases for automatic evaluation of machine translation and document summarization systems, and by generating concise or simplified information for sentence compression or sentence simplification systems (Madnani and Dorr, 2010). Traditional paraphrase generation methods exploit hand-crafted rules (McKeown, 1983) or automatically learned complex paraphrase patterns (Zhao et al., 2009), use thesaurus-based (Hassan et al., 2007) or semantic analysis driven natural language generation approaches (Kozlowski et al., 2003), or leverage statistical machine learning theory (Quirk et al., 2004; Wubben et al., 2010). In this paper, we propose to use deep learning principles to address the paraphrase generation problem.

Recently, techniques like sequence to sequence learning (Sutskever et al., 2014) have been applied to various NLP tasks with promising results, for example, in the areas of machine translation (Cho et al., 2014; Bahdanau et al., 2015), speech recognition (Li and Wu, 2015), language modeling (Vinyals et al., 2015), and dialogue systems (Serban et al., 2016). Although paraphrase generation can be formulated as a sequence to sequence learning task, not much work has been done in this area with regard to applications of state-of-the-art deep neural networks. There are several works on paraphrase recognition (Socher et al., 2011; Yin and Schütze, 2015; Kiros et al., 2015), but those employ classification techniques and do not attempt to generate paraphrases. More recently, attention-based Long Short-Term Memory (LSTM) networks have been used for textual entailment generation (Kolesnyk et al., 2016); however, paraphrase

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,
pages 2923–2934, Osaka, Japan, December 11-17 2016.

generation is a type of bi-directional textual entailment generation and no prior work has proposed a deep learning-based formulation of this task.

To address this gap in the literature, we explore various types of sequence to sequence models for paraphrase generation. We test these models on three different datasets and evaluate them using well recognized metrics. Along with the application of various existing sequence to sequence models for the paraphrase generation task, in this paper we also propose a new model that allows for training multiple stacked LSTM networks by introducing a residual connection between the layers. This is inspired by the recent success of such connections in a deep Convolutional Neural Network (CNN) for the image recognition task (He et al., 2015). Our experiments demonstrate that the proposed model can outperform other techniques we have explored.

Most of the deep learning models for NLP use Recurrent Neural Networks (RNNs). RNNs differ from normal perceptrons as they allow gradient propagation in time to model sequential data with variable-length input and output (Sutskever et al., 2011). In practice, RNNs often suffer from the vanishing/exploding gradient problems while learning long-range dependencies (Bengio et al., 1994). LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014) are known to be successful remedies to these problems.

It has been observed that increasing the depth of a deep neural network can improve the performance of the model (Simonyan and Zisserman, 2014; He et al., 2015) as deeper networks learn better representations of features (Farabet et al., 2013). In the vision-related tasks where CNNs are more widely used, adding many layers of neurons is a common practice. For tasks like speech recognition (Li and Wu, 2015) and also in machine translation, it is useful to stack layers of LSTM or other variants of RNN. So far this has been limited to only a few layers due to the difficulty in training deep RNN networks. We propose to add residual connections between multiple stacked LSTM networks and show that this allows us to stack more layers of LSTM successfully.

The rest of the paper is organized as follows: Section 2 presents a brief overview of the sequence to sequence models followed by a description of our proposed residual deep LSTM model, Section 3 describes the datasets used in this work, Section 4 explains the experimental setup, Section 5 presents the evaluation results and analyses, Section 6 discusses the related work, and in Section 7 we conclude and discuss future work.

2 Model Description

2.1 Encoder-Decoder Model

A neural approach to sequence to sequence modeling proposed by Sutskever et al. (2014) is a two-component model, where a source sequence is first encoded into some low dimensional representation (Figure 1) that is later used to reproduce the sequence back to a high dimensional target sequence (i.e. decoding). In machine translation, an encoder operates on a sentence written in the source language and encodes its meaning to a vector representation before the decoder can take that vector (which represents the meaning) and generate a sentence in the target language. These encoder-decoder blocks can be either a vanilla RNN or its variants. While producing the target sequence, the generation of each new word depends on the model and the preceding generated word. Generation of the first word in the target sequence depends on the special '`EOS`' (end-of-sentence) token appended to the source sequence.

The training objective is to maximize the log probability of the target sequence given the source sequence. Therefore, the best possible decoded target is the one that has the maximum score over the length of the sequence. To find this, a small set of hypotheses (candidate set) called *beam size* is used and the total score for all these hypotheses are computed. In the original work by Sutskever et al. (2014), they observe that although a beam size of 1 achieves good results, a higher beam size is always better. This is because for some of the hypotheses, the first word may not always have the highest score.

2.2 Deep LSTM

LSTM (Figure 2) is a variant of RNN, which computes the hidden state h_t using a different approach by adding an internal memory cell $c_t \in \mathbb{R}^n$ at every time step t. In particular, an LSTM unit considers the input state x_t at time step t, the hidden state h_{t-1}, and the internal memory state c_{t-1} at time step

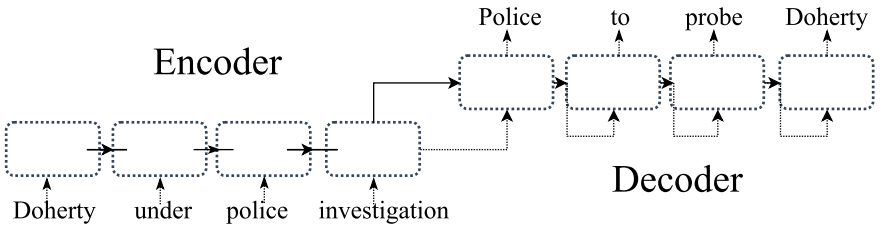


Figure 1: Encoder-Decoder framework for sequence to sequence learning.

1. Gates

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

2. Input transform

$$c\_in_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_{c\_in})$$

3. State Update

$$c_t = f_t \odot c_{t-1} + i_t \odot c\_in_t$$
$$h_t = o_t \odot \tanh(c_t)$$

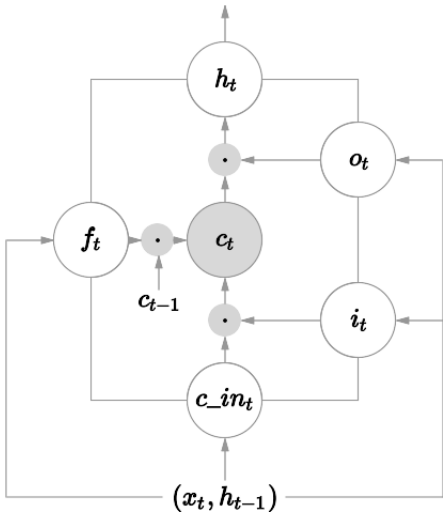


Figure 2: LSTM cell (Paszke, 2015).

$t-1$ to produce the hidden state h_t and the internal memory state c_t at time step t. The memory cell is controlled via three learned gates: input i, forget f, and output o. These memory cells use the addition of gradient with respect to time and thus minimize the gradient explosion. In most NLP tasks, LSTM outperforms vanilla RNN (Sundermeyer et al., 2012). Therefore, for our model we only explore LSTM as a basic unit in the encoder and decoder. Here, we describe the basic computations in an LSTM unit, which will provide the grounding to understand the residual connections between stacked LSTM layers later.

In the equations above, $W_{x\_}$, $W_{h\_}$ are the learned parameters for x and h respectively. $\sigma(.)$ and $\tanh(.)$ denote element-wise sigmoid and hyperbolic tangent functions respectively. $\odot$ is the element-wise multiplication operator and b denotes the added bias.

Graves (2013) explored the advantages of deep LSTMs for handwriting recognition and text generation. There are multiple ways of combining one layer of LSTM with another. For example, Pascanu et al. (2013) explored multiple ways of combining them and discussed various difficulties in training deep LSTMs. In this work, we employ vertical stacking where only the output of the previous layer of LSTM is fed to the input, as compared to the stacking technique used by Sutskever et al. (2014), where hidden states of all LSTM layers are fully connected. In our model, all but the first layer input at time step t is passed from the hidden state of the previous layer h_t^l, where l denotes the layer. This is similar to stacked RNN proposed by Bengio et al. (1994) but with LSTM units. Thus, for a layer l the activation is described by:

$$\boldsymbol{h}_t^{(l)} = f_h^l(\boldsymbol{h}_t^{(l-1)}, \boldsymbol{h}_{t-1}^{(l)})$$

where hidden states $\boldsymbol{h}$ are recursively computed and $\boldsymbol{h}_t^{(l)}$ at $t = 0$ and $l = 0$ is given by the LSTM equation of h_t.

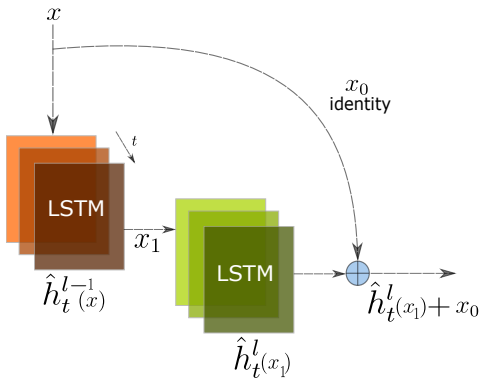


Figure 3: A unit of stacked residual LSTM.

2.3 Stacked Residual LSTM

We take inspiration from a very successful deep learning network *ResNet* (He et al., 2015) with regard to adding residue for the purpose of learning. With theoretical and empirical reasoning, He et al. (2015) have shown that the explicit addition of the residue x to the function being learned allows for deeper network training without overfitting the data.

When stacking multiple layers of neurons, the network often suffers through a *degradation* problem (He et al., 2015). The degradation problem arises due to the low convergence rate of training error and is different from the vanishing gradient problem. Residual connections can help overcome this issue. We experimented with four-layers of stacked LSTM for each of the model. Residual connections are added at layer two as the pointwise addition (see Figure 3), and thus it requires the input to be in the same dimension as the output of h_t. Principally because of this reason, we use a simple last hidden unit stacking of LSTM instead of a more intricate way as shown by Sutskever et al. (2014). This allowed us to clip the h_t to match the dimension of x_{t-2} where they were not the same. Similar results could be achieved by padding x to match the dimension instead. The function $\hat{\boldsymbol{h}}$ that is being learned for the layer with residual connection is therefore:

$$\hat{\boldsymbol{h}}_t^{(l)} = f_h^l(\boldsymbol{h}_t^{(l-1)}, \boldsymbol{h}_{t-1}^{(l)}) + x_{l-n}$$

where $\hat{h}$ for layer l is updated with residual value x_{l-n} and x_i represents the input to layer $i+1$. Residual connection is added after every n layers. However, for stacked LSTM, $n > 3$ is very expensive in terms of computation. In this paper we experimented with $n = 2$. Note that, when $n = 1$, the resulting function learned is a standard LSTM with bias that depends on the input x. That is why, it is not necessary to add the residual connection after every stacked layer of LSTM. The addition of residual connection does not add any learnable parameters. Therefore, this does not increase the complexity of the model unlike bi-directional models which double the number of LSTM units.

3 Datasets

We present the performance of our model on three datasets, which are significantly different in their characteristics. So, evaluating our paraphrase generation approach on these datasets demonstrates the versatility and robustness of our model.

PPDB (Pavlick et al., 2015) is a well known paraphrase dataset used for various NLP tasks. It comes in different sizes and the precision of the paraphrases degrades with the size of the dataset. We use the size L dataset from PPDB 2.0, which comes with over $18M$ paraphrases including lexical, phrasal and syntactic types. We have omitted the syntactic paraphrases and the instances which contain numbers, as they increase the vocabulary size significantly without giving any advantage of a larger dataset. This dataset contains relatively short paraphrases (86% of the data is less than four words), which makes it suitable for synonym generation and phrase substitution to address lexical and phrasal paraphrasing (Madnani and Dorr, 2010). For some phrases, PPDB has one-to-many paraphrases. We collect all such phrases to make a set of paraphrases and sampling without replacement was used to obtain the source

and reference phrases.

WikiAnswers (Fader et al., 2013) is a large question paraphrase corpus created by crawling the WikiAnswers website[1], where users can post questions and answers about any topic. The paraphrases are different questions, which were tagged by the users as similar questions. The dataset contains approximately 18M word-aligned question pairs. Sometimes, there occurs a loss of specialization between a given source question and its corresponding reference question when a paraphrase is tagged as similar to a reference question. For example, "prepare a *three month* cash budget" is tagged to "how to prepare a cash budget". This happens because general questions are typically more popular and get answered. So, specific questions are redirected to the general ones due to a comparative lack of interest in the very specific questions. It should be noted that this dataset comes preprocessed and lemmatized. We refer the reader to the original paper for more details.

MSCOCO (Lin et al., 2014) dataset contains human annotated captions of over $120K$ images. Each image contains five captions from five different annotators. While there is no guarantee that the human annotations are paraphrases, the nature of the images (which tends to focus on only a few objects and in most cases one prominent object or action) allows most annotators describe the most obvious things in an image. In fact, this is the main reason why neural networks for generating captions obtain better `BLEU` scores (Vinyals et al., 2014), which confirms the suitability of using this dataset for the paraphrase generation task.

4 Experimental Settings

4.1 Data Selection

For `PPDB` we remove the phrases that contain numbers including all syntactic phrases. This gives us a total of $5.3M$ paraphrases from which we randomly select 90% instances for training. For testing, we randomly select $20K$ pairs of paraphrases from the remaining 10% data. Although `WikiAnswers` comes with over $29M$ instances, we randomly select $4.8M$ for training to keep the training size similar to `PPDB` (see Table 1). $20K$ instances were randomly selected from the remaining data for testing. Note that, for the `WikiAnswers` dataset, we clip the vocabulary size[2] to $50K$ and use the special *UNK* symbol for the words outside the vocabulary. `MSCOCO` dataset has five captions for every image. This dataset comes with separate subsets for training and validation: *Train 2014* contains over $82K$ images and *Val 2014* contains over $40K$ images. From the five captions accompanying each image, we randomly omit one caption and use the other four as training instances (by creating two source-reference pairs). Thus, we obtain a collection of over $330K$ instances for training and $20K$ instances for testing. Because of the free form nature of the caption generation task (Vinyals et al., 2014), some captions were very long. We reduced those captions to the size of 15 words (by removing the words beyond the first 15) in order to reduce the training complexity of the models.

Dataset	Training	Test	Vocabulary Size
`PPDB`	4,826,492	20,000	38,279
`WikiAnswers`	4,826,492	20,000	50,000
`MSCOCO`	331,163	20,000	30,332

Table 1: Dataset details.

Models	Reference
Sequence to Sequence	(Sutskever et al., 2014)
With Attention	(Bahdanau et al., 2015)
Bi-directional LSTM	(Graves et al., 2013)
Residual LSTM	Our proposed model

Table 2: Models.

4.2 Models

We experimented with four different models (see Table 2). For each model, we experimented with two- and four-layers of stacked LSTMs. This was motivated by the state-of-the-art speech recognition systems that also use three to four layers of stacked LSTMs (Li and Wu, 2015). In encoder-decoder models, the size of the beam search used during inference is very important. Larger beam size always gives higher

[1] http://wiki.answers.com

[2] `WikiAnswers` dataset had many spelling errors yielding a very large vocabulary size (approximately $250K$). Hence, we selected the most frequent $50K$ words in the vocabulary to reduce the computational complexity.

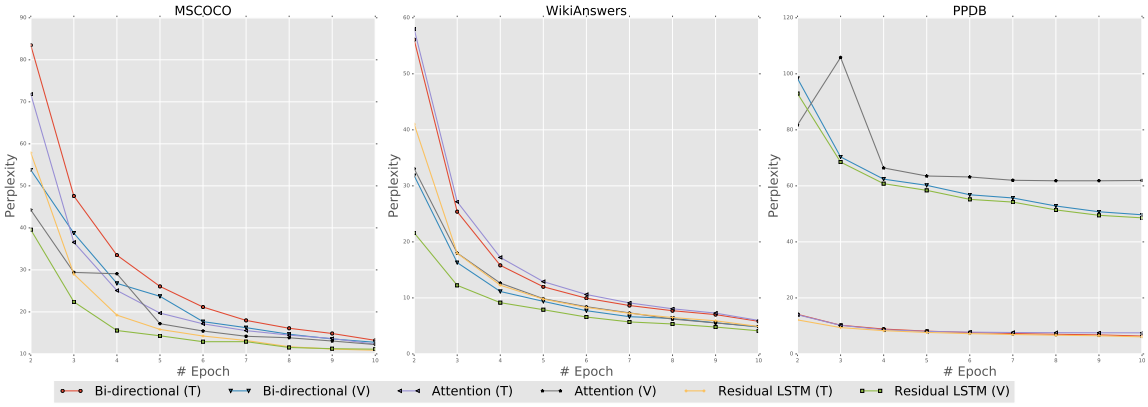


Figure 4: Perplexity during training (T) and validation (V) for various models [shared legend]. A lower perplexity represents a better model.

accuracy but is associated with a computational cost. We experimented with beam sizes of 5 and 10 to compare the models, as these are the most common beam sizes used in the literature (Sutskever et al., 2014). The bi-directional model used half of the number of layers shown for other models. This was done to ensure similar parameter sizes across the models.

4.3 Training

We used a one-hot vector approach to represent the words in all models. Models were trained with a stochastic gradient descent (SGD) algorithm. The learning rate began at 1.0, and was halved after every third training epoch. Each network was trained for ten epochs. In order to allow exploration of a wide variety of models, training was restricted to a limited number of epochs, and no hyper-parameter search was performed. A standard dropout (Srivastava et al., 2014) of 50% was applied after every LSTM layer. The number of LSTM units in each layer was fixed to 512 across all models. Training time ranged from 36 hours for `WikiAnswers` and `PPDB` to 14 hours for `MSCOCO` on a *Titan X* with *CuDNN 5* using Theano version $0.9.0dev1$ (Theano Development Team, 2016).

A beam search algorithm was used to generate optimal paraphrases by exploiting the trained models in the testing phase (Sutskever et al., 2014). We used perplexity as the loss function during training. Perplexity measures the uncertainty of the language model, corresponding to how many bits on average would be needed to encode each word given the language model. A lower perplexity indicates a better score. While `WikiAnswers` and `MSCOCO` had a very good correlation between training and validation perplexity, overfitting was observed with `PPDB` that yielded a worse validation perplexity (see Figure 4).

5 Evaluation

5.1 Metrics

To quantitatively evaluate the performance of our paraphrase generation models, we use the well-known automatic evaluation metrics[3] for comparing parallel corpora: `BLEU` (Papineni et al., 2002), `METEOR` (Lavie and Agarwal, 2007), and Translation Error Rate (`TER`) (Snover et al., 2006). Even though these metrics were designed for machine translation, previous works have shown that they can perform well for the paraphrase recognition task (Madnani et al., 2012) and correlate well with human judgments in evaluating generated paraphrases (Wubben et al., 2010).

Although there exists a few automatic evaluation metrics that are specifically designed for paraphrase generation, such as `PEM` (Paraphrase Evaluation Metric) (Liu et al., 2010) and `PINC` (Paraphrase In N-gram Changes) (Chen and Dolan, 2011), they have certain limitations. `PEM` relies on large in-domain bilingual parallel corpora along with sample human ratings for training while it can only model paraphrasing up to the phrase-level granularity. `PINC` attempts to solve these limitations by proposing a method that is essentially the inverse of `BLEU`, as it calculates the n-gram difference between the source

[3]We used the software available at `https://github.com/jhclark/multeval`

and the reference sentences. Although PINC correlates well with human judgments in lexical dissimilarity assessment, BLEU has been shown to correlate better for semantic equivalence agreements at the sentence-level when a sufficiently large number of reference sentences are available for each source sentence (Chen and Dolan, 2011).

BLEU considers exact matching between reference paraphrases and system generated paraphrases by considering n-gram overlaps while METEOR improves upon this measure via stemming and synonymy using WordNet. TER measures the number of edits required to change a system generated paraphrase into one of the reference paraphrases. As suggested in Clark et al. (2011), we used a stratified approximate randomization (AR) test. AR calculates the probability of a metric score providing the same reference sentence by chance. We report our *p*-values at 95% Confidence Intervals (CI).

The major limitation of these evaluation metrics is that they do not consider the meaning of the paraphrases, and hence, are not able to capture paraphrases of entities. For example, these metrics do not reward the paraphrasing of *"London"* to *"Capital of UK"*. Therefore, we also evaluate our models on a sentence similarity metric[4] proposed by Rus et al. (2012). This metric uses word embeddings to compare the phrases. In our experiments, we used Word2Vec embeddings pre-trained on the Google News Corpus (Mikolov et al., 2014). This is referred to as *'Emb Greedy'* in our results table.

5.2 Results

Table 3 presents the results from various models across different datasets. ↑ denotes that higher scores represent better models while ↓ means that a lower score yields a better model. Although our focus is on stacked residual LSTM, which is applicable only when there are more than two layers, we still present the scores from two-layer LSTM as a baseline. This provides a good comparison against deeper models. The results demonstrate that our proposed model outperforms other models on BLEU and TER for all datasets. On *Emb Greedy*, our model outperforms other models in all datasets except the Attention model when beam size is 10. On METEOR, our model outperforms other models on MSCOCO and WikiAnswers; however, for PPDB, the simple sequence to sequence model performs better. Note that these results were obtained by using single models and no ensemble of the models was used.

To calculate BLEU and METEOR, four references were used for MSCOCO, and five for PPDB and WikiAnswers. In some instances, WikiAnswers did not have up to five reference paraphrases for every source, hence, those were calculated on reduced references. In Table 4, we present the variance due to the test set selection. This is calculated using bootstrap re-sampling for each optimizer run (Clark et al., 2011). Variance due to optimizer instability was less than 0.1 in all cases. *p*-value of these tests are less than 0.05 in all cases. Thus, comparison between two models is significant at 95% CI if the difference in their score is more than the variance due to test set selection (Table 4).

5.3 Analysis

Scores on various metrics vary a lot across the datasets, which is understandable due to their inherent differences. PPDB contains very small phrases and thus does not score well with metrics like BLEU and METEOR which penalize shorter phrases. As shown in Figure 5, more than 50% of PPDB contains one or two words. This leads to a substantial difference between training and validation errors, as shown in Figure 4. The results demonstrate that deeper LSTMs consistently improve performance over shallow models. For beam size of 5 our model outperforms other models in all datasets. For beam size of 10, the attention-based model has a marginally better *Emb Greedy* score than our model. When we look at the qualitative results, we notice that the bias in the dataset is exploited by the system which is a side effect of any form of learning on a limited dataset. We can see this effect in Table 5. For example, *an OBJECT* is mostly paraphrased with *an OBJECT* (e.g. *bowl*, *motorcycle*). Shorter sentences mostly generate shorter paraphrases and the same is true for longer sequences. Based on our results, the embedding-based metric correlates well with statistical metrics. Figure 4 and the results from Table 5 suggest that perplexity is a good loss function for training paraphrase generation models. However, a more ideal metric to fully encode the fundamental objective of paraphrasing should also reward novelty and penalize redundancy during paraphrase generation, which is a notable limitation of the existing paraphrase evaluation metrics.

[4]We used the software available at https://github.com/julianser/hed-dlg-truncated/

#Layers	Model	Beam size = 5 BLEU↑	METEOR↑	Emb Greedy↑	TER↓	Beam size = 10 BLEU↑	METEOR↑	Emb Greedy↑	TER↓
				PPDB					
2	Sequence to Sequence	12.5	21.3	32.55	82.9	12.9	20.5	32.65	83.0
	With Attention	13.0	21.2	32.95	82.2	13.8	20.6	32.29	81.9
4	Sequence to Sequence	18.3	**23.5**	33.18	82.7	18.8	**23.5**	33.78	82.1
	Bi-directional	19.2	23.1	34.39	77.5	19.7	23.2	34.56	84.4
	With Attention	19.9	23.2	34.71	83.8	20.2	22.9	**34.90**	77.1
	Residual LSTM	**20.3**	23.1	**34.77**	**77.1**	**21.2**	23.0	34.78	**77.0**
				WikiAnswers					
2	Sequence to Sequence	19.2	26.1	62.65	35.1	19.5	26.2	62.95	34.8
	With Attention	21.2	22.9	63.22	37.1	21.2	23.0	63.50	37.0
4	Sequence to Sequence	33.2	29.6	73.17	28.3	33.5	29.6	73.19	28.3
	Bi-directional	34.0	30.8	73.80	27.3	34.3	30.7	73.95	27.0
	With Attention	34.7	31.2	73.45	27.1	34.9	31.2	73.50	27.1
	Residual LSTM	**37.0**	**32.2**	**75.13**	**27.0**	**37.2**	**32.2**	**75.19**	**26.8**
				MSCOCO					
2	Sequence to Sequence	15.9	14.8	54.11	66.9	16.5	15.4	55.81	67.1
	With Attention	17.5	16.6	58.92	63.9	18.6	16.8	59.26	63.0
4	Sequence to Sequence	28.2	23.0	67.22	56.7	28.9	23.2	67.10	56.3
	Bi-directional	32.6	24.5	68.62	53.8	32.8	24.9	68.91	53.7
	With Attention	33.1	25.4	69.10	54.3	33.4	25.2	**69.34**	53.8
	Residual LSTM	**36.7**	**27.3**	**69.69**	**52.3**	**37.0**	**27.0**	69.21	**51.6**

Table 3: Evaluation results on PPDB, WikiAnswers, and MSCOCO (Best results are in **bold**).

Dataset	σ^2[BLEU]	σ^2[METEOR]	σ^2[TER]	σ^2[Emb Greedy]
PPDB	2.8	0.2	0.4	0.000100
WikiAnswers	0.3	0.1	0.1	0.000017
MSCOCO	0.2	0.1	0.1	0.000013

Table 4: Variance due to test set selection.

	PPDB	WikiAnswers	MSCOCO
Source	south eastern	what be the symbol of magnesium sulphate	a small kitten is sitting in a bowl
Reference	the eastern part	chemical formulum for magnesium sulphate	a cat is curled up in a bowl
Generated	south east	do magnesium sulphate have a formulum	a cat that is sitting on a bowl
Source	organized	what be the bigggest galaxy know to man	an old couple at the beach during the day
Reference	managed	how many galaxy be there in you known universe	two people sitting on dock looking at the ocean
Generated	arranged	about how many galaxy do the universe contain	a couple standing on top of a sandy beach
Source	counselling	what do the ph of acid range to	a little baby is sitting on a huge motorcycle
Reference	be kept informed	a acid have ph range of what	a little boy sitting alone on a motorcycle
Generated	consultations	how do acid affect ph	a baby sitting on top of a motorcycle

Table 5: Example paraphrases generated using the 4-layer Residual LSTM with beam size 5.

6 Related Work

Prior approaches to paraphrase generation have applied relatively different methodologies, typically using knowledge-driven approaches or statistical machine translation (SMT) principles. Knowledge-driven methods for paraphrase generation (Madnani and Dorr, 2010) utilize hand-crafted rules (McKeown, 1983) or automatically learned complex paraphrase patterns (Zhao et al., 2009). Other paraphrase generation methods use thesaurus-based (Hassan et al., 2007) or semantic analysis-driven natural language generation approaches (Kozlowski et al., 2003) to generate paraphrases. In contrast, Quirk et al., (2004)

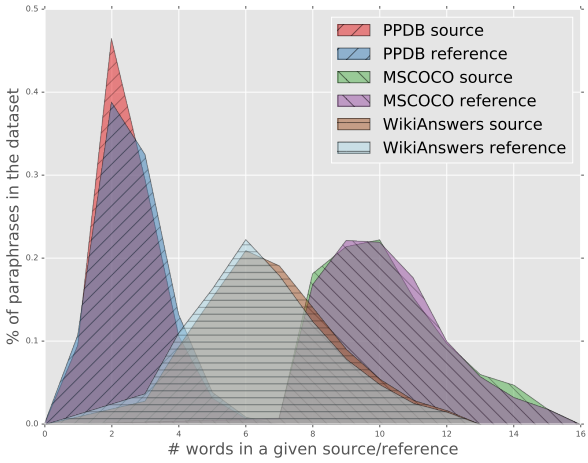


Figure 5: Distribution of sequence length (in number of words) across datasets.

show the effectiveness of SMT techniques for paraphrase generation given adequate monolingual parallel corpus extracted from comparable news articles. Wubben et al., (2010) propose a phrase-based SMT framework for sentential paraphrase generation by using a large aligned monolingual corpus of news headlines. Zhao et al., (2008) propose a combination of multiple resources to learn phrase-based paraphrase tables and corresponding feature functions to devise a log-linear SMT model. Other models generate application-specific paraphrases (Zhao et al., 2009), leverage bilingual parallel corpora (Bannard and Callison-Burch, 2005) or apply a multi-pivot approach to output candidate paraphrases (Zhao et al., 2010).

Applications of deep learning for paraphrase generation tasks have not been rigorously explored. We utilized several sources as potential large datasets. Recently, Weiting et al. (2015) took the PPDB dataset (size XL) and annotated phrases based on their paraphrasability. This dataset is called *Annotated*-PPDB and contains 3000 pairs in total. They also introduced another dataset called *ML-Paraphrase* for the purpose of evaluating bigram paraphrases. This dataset contains 327 instances. Microsoft Research Paraphrase Corpus (MSRP) (Dolan et al., 2005) is another widely used dataset for paraphrase detection. MSRP contains 5800 pairs of sentences (obtained from various news sources) accompanied with human annotations. These datasets are too small and therefore, we did not use them for training our deep learning models.

To the best of our knowledge, this is the first work on using residual connections with recurrent neural networks. Very recently, we found that Toderici et al. (2016) used residual GRU to show an improvement in image compression rates for a given quality over JPEG. Another variant of residual network called *DenseNet* (Huang et al., 2016), which uses dense connections over every layer, has been shown to be effective for image recognition tasks achieving state-of-the-art results in CIFAR and SVHN datasets. Such works further validate the efficacy of adding residual connections for training deep networks.

7 Conclusion and Future Work

In this paper, we described a novel technique to train stacked LSTM networks for paraphrase generation. This is an extension to sequence to sequence learning, which has been shown to be effective for various NLP tasks. Our model outperforms state-of-the-art models for sequence to sequence learning. We have shown that stacking of residual LSTM layers is useful for paraphrase generation, but it may not perform equally well for machine translation because not every word in a source sequence needs to be substituted for paraphrasing. Residual connections help retain important words in the generated paraphrases.

We experimented on three different large scale datasets and reported results using various automatic evaluation metrics. We showed the use of the well-known MSCOCO dataset for paraphrase generation and demonstrated that the models can be trained effectively without leveraging the images. The presented experiments should set strong baselines for neural paraphrase generation on these datasets, enabling future researchers to easily compare and evaluate subsequent works in paraphrase generation.

Recent advances in neural networks with regard to learnable memory (Sukhbaatar et al., 2015; Graves et al., 2014) have enabled models to get one step closer to learning comprehension. It may be helpful to explore such networks for the paraphrase generation task. Also, it remains to be explored how unsupervised deep learning could be harnessed for paraphrase generation. It would be interesting to see if researchers working on image-captioning can employ neural paraphrase generation to augment their dataset.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and feedback. The first author is especially grateful to Prof. James Storer, Brandeis University, for his guidance and Nick Moran, Brandeis University, for helpful discussions.

References

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*, pages 1–15.

C. Bannard and C. Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of ACL*, pages 597–604.

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

D. Chen and W. B. Dolan. 2011. Collecting Highly Parallel Data for Paraphrase Evaluation. In *Proceedings of ACL-HLT*, pages 190–200.

K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of EMNLP*, pages 1724–1734.

J. H. Clark, C. Dyer, A. Lavie, and N. A. Smith. 2011. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of ACL-HLT*, pages 176–181.

B. Dolan, C. Brockett, and C. Quirk. 2005. Microsoft Research Paraphrase Corpus. *Retrieved March*, 29:2008.

A. Fader, L. S Zettlemoyer, and O. Etzioni. 2013. Paraphrase-Driven Learning for Open Question Answering. In *ACL*, pages 1608–1618. ACL.

C. Farabet, C. Couprie, L. Najman, and Y. LeCun. 2013. Learning Hierarchical Features for Scene Labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929.

A. Graves, N. Jaitly, and A. Mohamed. 2013. Hybrid Speech Recognition with Deep Bidirectional LSTM. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

A. Graves, G. Wayne, and I. Danihelka. 2014. Neural Turing Machines. In *arXiv:1410.5401*.

A. Graves. 2013. Generating Sequences with Recurrent Neural Networks. *arXiv preprint arXiv:1308.0850*.

S. Hassan, A. Csomai, C. Banea, R. Sinha, and R. Mihalcea. 2007. UNT: SubFinder: Combining Knowledge Sources for Automatic Lexical Substitution. In *Proceedings of SemEval*, pages 410–413.

K. He, X. Zhang, S. Ren, and J. Sun. 2015. Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385*.

S. Hochreiter and J. Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

G. Huang, Z. Liu, and K. Q. Weinberger. 2016. Densely Connected Convolutional Networks. *arXiv preprint arXiv:1608.06993*.

R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. 2015. Skip-Thought Vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302.

V. Kolesnyk, T. Rocktäschel, and S. Riedel. 2016. Generating Natural Language Inference Chains. *CoRR*, abs/1606.01404.

R. Kozlowski, K. F. McCoy, and K. Vijay-Shanker. 2003. Generation of Single-sentence Paraphrases from Predicate/Argument Structure Using Lexico-grammatical Resources. In *Proceedings of the 2nd International Workshop on Paraphrasing*, pages 1–8.

A. Lavie and A. Agarwal. 2007. METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231.

X. Li and X. Wu. 2015. Constructing Long Short-term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4520–4524. IEEE.

T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*, pages 740–755. Springer.

C. Liu, D. Dahlmeier, and H. T. Ng. 2010. PEM: A Paraphrase Evaluation Metric Exploiting Parallel Texts. In *Proceedings of EMNLP*, pages 923–932.

N. Madnani and B. J. Dorr. 2010. Generating Phrasal and Sentential Paraphrases: A Survey of Data-driven Methods. *Computational Linguistics*, 36(3):341–387.

N. Madnani, J. Tetreault, and M. Chodorow. 2012. Re-examining Machine Translation Metrics for Paraphrase Identification. In *Proceedings of NAACL-HLT*, pages 182–190.

K. R. McKeown. 1983. Paraphrasing Questions Using Given and New Information. *Computational Linguistics*, 9(1):1–10.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2014. Word2Vec. https://code.google.com/p/word2vec. Online; accessed 2014-04–15.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL*, pages 311–318.

R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. 2013. How to Construct Deep Recurrent Neural Networks. *arXiv preprint arXiv:1312.6026*.

A. Paszke. 2015. LSTM Implementation Explained: apaszke.github.io/lstm-explained.html , 2016-07-15.

E. Pavlick, P. Rastogi, J. Ganitkevitch, B. Van Durme, and C. Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of ACL-IJCNLP)*, pages 425–430.

C. Quirk, C. Brockett, and W. Dolan. 2004. Monolingual Machine Translation for Paraphrase Generation. In *Proceedings of EMNLP*, pages 142–149.

V. Rus and M. Lintean. 2012. A Comparison of Greedy and Optimal Assessment of Natural Language Student Input using Word-to-Word Similarity Metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 157–162. ACL.

I. V. Serban, T. Klinger, G. Tesauro, K. Talamadupula, B. Zhou, Y. Bengio, and A. Courville. 2016. Multiresolution Recurrent Neural Networks: An Application to Dialogue Response Generation. *arXiv preprint arXiv:1606.00776*.

K. Simonyan and A. Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems*, pages 1–9.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

S. Sukhbaatar, J. Weston, R. Fergus, et al. 2015. End-to-End Memory Networks. In *Advances in neural information processing systems*, pages 2440–2448.

M. Sundermeyer, R. Schlüter, and H. Ney. 2012. LSTM Neural Networks for Language Modeling. In *Interspeech*, pages 194–197.

I. Sutskever, J. Martens, and G. E. Hinton. 2011. Generating Text with Recurrent Neural Networks. In *Proceedings of ICML*, pages 1017–1024.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Annual Conference on Neural Information Processing Systems*, pages 3104–3112.

Theano Development Team. 2016. Theano: A Python Framework for Fast Computation of Mathematical Expressions. *arXiv e-prints*, abs/1605.02688, May.

G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell. 2016. Full Resolution Image Compression with Recurrent Neural Networks. *arXiv preprint arXiv:1608.05148*.

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. 2014. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555.

O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton. 2015. Grammar as a Foreign Language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. 2015. From Paraphrase Database to Compositional Paraphrase Model and Back. *Transactions of the ACL (TACL)*.

S. Wubben, A. van den Bosch, and E. Krahmer. 2010. Paraphrase Generation As Monolingual Translation: Data and Evaluation. In *Proceedings of INLG*, pages 203–207.

W. Yin and H. Schütze. 2015. Convolutional Neural Network for Paraphrase Identification. In *Proceedings of NAACL-HLT*, pages 901–911.

S. Zhao, C. Niu, M. Zhou, T. Liu, and S. Li. 2008. Combining Multiple Resources to Improve SMT-based Paraphrasing Model. In *Proceedings of ACL-HLT*, pages 1021–1029.

S. Zhao, X. Lan, T. Liu, and S. Li. 2009. Application-driven Statistical Paraphrase Generation. In *Proceedings of ACL-IJCNLP*, pages 834–842.

S. Zhao, H. Wang, X. Lan, and T. Liu. 2010. Leveraging Multiple MT Engines for Paraphrase Generation. In *Proceedings of COLING*, pages 1326–1334.

English-Chinese Knowledge Base Translation with Neural Network

Xiaocheng Feng, Duyu Tang, Bing Qin, Ting Liu
Harbin Institute of Technology, Harbin, China
{*xcfeng, dytang, qinb, tliu*}@ir.hit.edu.cn

Abstract

Knowledge base (KB) such as Freebase plays an important role for many natural language processing tasks. English knowledge base is obviously larger and of higher quality than low resource language like Chinese. To expand Chinese KB by leveraging English KB resources, an effective way is to translate English KB (source) into Chinese (target). In this direction, two major challenges are to model triple semantics and to build a robust KB translator. We address these challenges by presenting a neural network approach, which learns continuous triple representation with a gated neural network. Accordingly, source triples and target triples are mapped in the same semantic vector space. We build a new dataset for English-Chinese KB translation from Freebase, and compare with several baselines on it. Experimental results show that the proposed method improves translation accuracy compared with baseline methods. We show that adaptive composition model improves standard solution such as neural tensor network in terms of translation accuracy.

1 Introduction

Knowledge base (KB) like Freebase[1] and Yago[2] has attracted a lot of attention in both research and industry communities. Knowledge Base contains massive triples (entries), each of which is a fact consisting of two arguments and one predicate, such as (*Una White, profession, Nurse*). A large, high-quality KB is valuable and can be applied to many natural language processing and information retrieval tasks (Graupmann et al., 2005; Hotho et al., 2006; Ferrández et al., 2009; Bouma et al., 2009; Shi et al., 2016; Feng et al., 2016). Let us take Freebase as an example, English KB contains 2.7 billion entries and the accuracy is higher than 80%. This is obviously larger and better than a KB with less amount of entries such as Chinese.

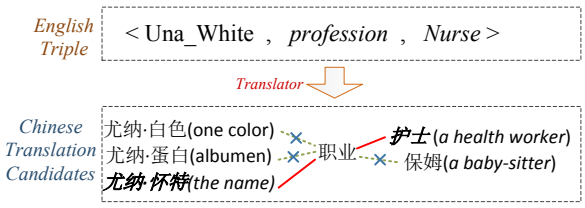


Figure 1: An example of translation ambiguity in English-Chinese KB translation.

A straightforward way to enrich Chinese KB is to directly translate English KB (source) to Chinese (target) based on the surface texts of a triple with existing machine translation system. However, we find that they suffer from the problem of ambiguity. An example is given in Figure 1. The argument "*nurse*" has two translation candidates namely "护士" (a person taking care of sick people in hospital) and "保姆" (baby sitter). "*Una White*" have three translation candidates, so that there are totally six ambiguous

[1]http://en.wikipedia.org/wiki/Freebase
[2]http://en.wikipedia.org/wiki/YAGO_(database)

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2935–2944, Osaka, Japan, December 11-17 2016.

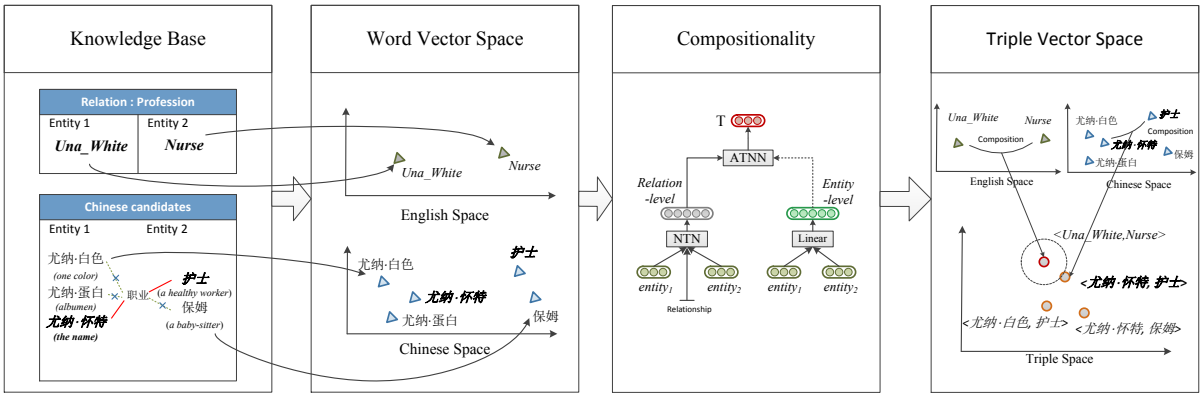


Figure 2: An illustration of the neural network approach for English-Chinese KB translation. It corresponds to the translation example as given in Figure 1.

candidates according to Cartesian product. A preliminary statistical analysis shows that more than half of Freebase translations (English→Chinese) are ambiguous.

There are two main challenges to effectively disambiguate these translated triples. The first challenge is how to effectively model the semantics of English triple and Chinese triple in a unified space. It is preferable to learn a projection function, which maps both English triples and Chinese triples in the same semantic space. The second challenge is how to build a robust KB translator without labor-intensive feature engineering.

In this paper, we address these two challenges by presenting an adaptive neural network to translate English KB into Chinese. Given an English triple and a list of translated Chinese triple candidates, the method assigns a scalar to each translation pair to represent their semantic relatedness. Specifically, we represent each KB triple from the embeddings of words it contains, and introduce an adaptive composition model to effectively capture the semantic composition between arguments of a triple. Compared to previous triple composition methods, the adaptive policy is inspired by highway network (Srivastava et al., 2015b), which is a dynamic calculating process based on different triples rather than a fixed model. In this way, source triple and target triple are naturally encoded in the same semantic vector space. We design a ranking-type hinge loss function to effectively train the parameters of neural networks.

We evaluate the effectiveness of our method on a manually created corpus. We conduct experiments in two settings. Empirical results show that the proposed method consistently outperforms baseline methods. We also show that the use of gated neural network improves strong composition models such as neural tensor network (Socher et al., 2013b) in terms of translation accuracy. The main contributions of this work are as follows:

- We introduce an approach based on representation learning for English-Chinese KB translation in this paper.
- We present a gated neural network to adaptively integrate entity and relational level evidences in triple representation.
- We build a dataset for English-Chinese KB translation, and report the superior performance of our method over baseline methods on it.

2 The Approach

In this section, we present our neural network method for KB Translation in detail. Figure 2 displays a high-level overview of the approach. Given an English triple as input, we first get the candidate Chinese triples (Section 2.1). Afterwards, the semantic representations of English triples and Chinese triples are modeled with neural network (Section 2.2), which are further used for triple ranking (Section 2.3).

2.1 Candidate Generation

In general, a triple in KB is composed of two entities and a relation. In this work, we use English triples from Freebase as the input source, which contains a small number of pre-defined relations. Therefore, we only translate the entities of a triple, while regarding the relation as given. To this end, it is intuitive to use existing text translator to directly translate entities for obtaining candidates. We try to feed English entities to Bing translator, however, a large portion of them can not be translated directly. This stems from the fact that majority of entities are names, proper nouns and named entities which are not well covered in existing translator. Therefore, we use a heuristic method to handle the entities not covered by Bing translator. We split an entity as individual words, and compose the translation results of words as its translation candidate. For example, "*Una White*" in Figure 2.

2.2 Semantic Composition for KB Triple

This section introduces a neural network approach to learn continuous representation for English and Chinese KB triple. We extend this principle in this paper and state that the meaning of a triple is composed from the meanings of entities, relations as well as their correlations.

We find that directly translating entity literally is not effective enough for English-Chinese triple translation. Let us take (Una White, Profession, Nurse) in Figure 1 as an example. The argument "*nurse*" has two translation candidates namely "护士" (a person taking care of sick people in hospital) and "保姆" (baby sitter). "*Una White*" has three translation candidates, so that there are totally six ambiguous candidates according to Cartesian product. To effectively handle this ambiguity problem, we develop an adaptive neural network approach to produce triple representations by effectively capturing entity semantics and the relations between them.

We first describe entity and relational representation. After that, an adaptive composition model is introduced to produce triple representations by automatically combining entity and relational semantics.

2.2.1 Entity Representation

We describe the method to learn continuous representation for each entity. Since an entity is typically a phrase consisting of 2 or 3 words, we average the continuous word representation as the entity representation (Socher et al., 2013b).

Formally, we represent each word as a distributed, continuous and dense vector, which is also known as word embedding (Bengio et al., 2003; Mikolov et al., 2013). These word vectors are stacked in an embedding matrix $L \in \mathbb{R}^{d \times |V|}$, where $|V|$ is word vocabulary size and d is the dimension of each word vector. These word vectors can be randomly initialized from a uniform distribution $U(-0.001, 0.001)$, regarded as parameters of neural networks, and jointly learned with task-specific objectives. Alternatively, the embedding of a word can be trained based on its context information in large-scale text corpus. We use the latter approach since it can make better use of the semantics of words. We learn word embeddings with word2vec[3], which is one of state-of-the-art embedding learning algorithms and widely used for many natural language processing tasks. We learn English word embedding from Wikipedia dump[4], and learn Chinese word embedding from Baike texts[5].

To compose the entity representation from the embedding of words it contains, we follow Socher et al. (2013b) and average the continuous word representation as entity representation. Recursive Neural Network is not suitable to represent entity because entities are typically people names which do not contain explicit compositional structure.

Since the English word vectors and Chinese word vectors are separately trained without using bilingual parallel corpus, these word vectors are mapped into different semantic spaces. This is not desirable for comparing the semantic relatedness between English triple and Chinese triple. We use linear layers to transform English and Chinese word vectors in a same semantic vector space. A simple linear layer is calculated as $v_e = We + b$, where W and b are the parameters. One could also learn bilingual

[3] code.google.com/p/word2vec/

[4] https://dumps.wikimedia.org/

[5] http://baike.baidu.com/

word vectors simultaneously from bilingual parallel corpus with tailored learning algorithm (Zou et al., 2013). We leave this as a future work and we believe our method could benefit from the bilingual word embeddings.

2.2.2 Relational Representation

We model the semantic relatedness between entities in this part. The basic idea is that the semantic relatedness between entities is determined by the semantics of entities and their relations. Based on this, we utilize neural tensor network, which is one of state-of-the-art semantic composition approach for natural language processing tasks (Mitchell and Lapata, 2010; Socher et al., 2013a; Jenatton et al., 2012).

A standard neural tensor with rank 3 is essentially a list of bilinear neural layers, each of which takes two vectors as inputs and outputs a real-valued scalar with element wise multiplication. Furthermore, relation-specific neural tensor can be exploited to make better use of the relation between entities, which is calculated as follows.

$$v_r = e_1^T W_R^{[1:k]} e_2 \tag{1}$$

where k is the length of output vector, e_1 and e_2 are the d-dimensional embeddings of two entities in a given triple, $W_R^{[1:k]} \in \mathbb{R}^{d \times d \times k}$ stands for the parameters of tensor. Each element in v_r is computed by one slice $i \in \{1, ..., k\}$ of tensor.

2.2.3 Adaptive Neural Network

We have previously obtained entity representation and relational representation, both of which play important roles for representing the meaning of a triple. Furthermore, a better approach should benefit from both aspects, and integrate them in triple semantic with an automatic method. To this end, we introduce a gated neural network in this part. It takes entity and relational vectors of a triple as input, and adaptively produces the composed continuous representation of them.

Given v_e and v_r as inputs, a traditional compositional function is to concatenate v_e and v_r and feed them to a linear layer (Socher et al., 2011), which is calculated as Equation 2. Despite its computational efficiency, tied parameters cannot easily capture the complex linguistic phenomena in natural language expressions.

$$\tilde{v} = tanh(W_e v_e + W_r v_r + b) \tag{2}$$

$$\alpha = \sigma(W_{eg} v_e + W_{rg} v_r + b_g) \tag{3}$$

$$v(t) = \alpha \cdot v_r + (1 - \alpha) \cdot \tilde{v} \tag{4}$$

Therefore, we add a neural gate to change parameter values for different input vectors v_e and v_r, which is partly inspired by the recent success of gated recurrent neural network (Cho et al., 2014; Chung et al., 2015) and Long Short-Term Memory (Hochreiter and Schmidhuber, 1997; Tai et al., 2015). And our gated neural network is inspired by highway network, which allow the model to suffer less from the vanishing gradient problem (Srivastava et al., 2015a; Srivastava et al., 2015b). The gate takes v_e and v_r as inputs, and outputs as a weight $\alpha \in [0, 1]$, which linearly weights the two parts. Specifically, the gate is calculated as Equation 3, where σ is standard sigmoid function, W_{eg}, W_{rg} and b_g are parameters. Triple representation $v(t)$ is calculated as given in Equation 4, which linearly weights the candidate composed representation $\tilde{v}$ and relational representation v_r. In this way, entity representation and relational representation are adaptively encoded in the semantic representation of a triple.

2.3 English-Chinese Triple Translation

Given an English triple t_e and a list of candidate Chinese triples $\{t_{c1}, ...t_{cj}, ... \ t_{ck}\}$, we select the most relevant Chinese candidate in terms of semantic as the translation answer. To this end, we need to formalize a scoring function $f(t_e, t_{cj})$, which is capable of measuring the semantic relatedness between an English triple t_e and a Chinese candidate triple t_{cj}. Specifically, we apply the continuous triple vector

learned in Section 2.2.3 as English and Chinese triple representations without any feature engineering. We use standard L1 and L2 norms as the dissimilarity measure f, namely:

$$f(t_e, t_{cj}) = ||v(t_e) - v(t_{cj})||_p \tag{5}$$

where $p = 1$ means L1 norm, and $p = 2$ stands for L2 norm.

To effectively estimate the parameters of the neural networks, we use a ranking type loss function based on the intuition of noise contrastive estimation (Gutmann and Hyvärinen, 2010), which has been exploited as an effective training objective in deep learning community (Socher et al., 2013b).

In this paper, the basic idea of the optimizing objective is that: the distance between an English triple and its correct translation candidate $f(t_e, t_c)$ should get a lower score than the distance between the source triple and a *corrupted* incorrect candidate triple $f(t_e, t'_c)$ by a margin of 1.

$$Loss = \sum_{i=1}^{N} max(0, 1 - f(t_e, t'_c) + f(t_e, t_c)) \tag{6}$$

where N is the number of training instances. Given a correct Chinese triple (e_1, r, e_2), we generate two corrupted triples by randomly replacing one entity at one time, resulting in (e_1, r, e'_2) and (e'_1, r, e_2). We train the neural networks with supervised learning using stochastic gradient descent. We take the derivative of the loss regarding the parameters with standard back-propagation. We learn 50-dimensional English and Chinese word embeddings using word2vec with default setting. The vocabulary size of English and Chinese word embeddings are 32K and 30K, respectively. For each relation type, we use relation untied parameters, randomly initialize the values of $W_e, W_r, W_{eg}, W_{rg}, W_R^1, W_R^2, u'$, b and b_g from a uniform distribution $U(-0.01/L, 0.01/L)$ where L is the input length of a neural layer, set the hidden length as 30, set the learning rate as 0.03 and tune the training round on development set.

3 Experiment

We apply the proposed method for English-Chinese KB translation to evaluate its effectiveness. We describe the data statistics, experimental settings and empirical results.

3.1 Experiment Settings

For the task of English-Chinese KB translation, since there is no publicly available benchmark dataset, we manually annotate a dataset by ourselves. We use Freebase as the English source, which is widely used in the field of KB population and KB completion. There are several relation types in Freebase, we only select "Profession" and "Cause of Death" in this work because they are representative relation types with large number of instances. We leave other relation types as future work. We use Bing translator to get the translation candidates, and employ two experts to annotate the best result among candidate list. The disagreements during annotation are fixed by detailed discussion.We randomly split the dataset as training, development and testing sets. The statistical information of the datasets are given in Table 1.

Relation Type	#Train	# Dev	# Test
Profession	3,000	500	500
Cause of death	2,000	500	500

Table 1: The statistics for Freebase including two different relations.

We conduct experiments in a supervised learning framework. For each relation type, we train the model on training set, tune parameters on dev set and evaluate on test set. We use $P@1$ (Manning and Schütze, 1999) as the evaluation metric, which indicates whether the first ranked translation results is the correct answer.

- Surface Matching. Given an English triple (e_1, r, e_2), we first get the translation candidates of e_1 and e_2. After that, we select the top-ranked entity in each set, and merge them as the best translate result.

Model	L1 Norm			L2 Norm		
	Profession	C-death	Avg	Profession	C-death	Avg
Surface Matching	52.4	51.6	52.0	52.4	51.6	52.0
Hints Similarity	58.2	56.4	57.3	58.2	56.4	57.3
Entity Model	70.4	64.6	67.5	72.2	67.4	69.9
Relational (Tensor) Model	71.8	65.2	68.5	72.4	68.6	70.5
Relational (Tensor*) Model	72.4	67.2	69.8	72.8	69.8	71.3
Average (Entity + Relational)	72.8	67.6	70.2	73.0	70.2	71.6
Linear (Entity + Relational)	73.2	68.0	70.6	73.6	70.6	72.1
Full Model	75.0	70.4	72.7	75.6	71.8	73.7

Table 2: Comparison of accuracy of the different models for English-Chinese KB translation. We run experiments in L1 Norm and L2 Norm. Evaluation metric is P@1.

- Hints Similarity. After obtaining the list of candidates with Cartesian product, we measure the similarity between entities in a candidate triple with web search. We concatenate two entities as a query and put them in a Chinese search engine. We count the co-occurrence frequency in snippets, and select the top ranked candidate as the answer.

Our model has several variations, which are detailed as below.

- Entity Model. We represent a triple by only using entity representation, without leveraging relational representation (Bordes et al., 2012) .

- Relational Model. We represent a triple by only using relational representation, without using entity representation. In Relational (Tensor) Model, the neural calculator only uses multiplicative composition function as described in Section 2.2.2. In Relational (Tensor*) Model, the neural calculator includes an additional linear layer, which is exploited in Socher et al. (2013b) and calculated as below.

$$v_t = [e_1^T W_R^{[1:k]} e_2 + V_R \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + b_R] \tag{7}$$

- Average (Entity + Relational). In this setting, we represent a triple by averaging its entity vector and relational vector, without using gated neural network.

$$v_t = \frac{v_e + v_r}{2} \tag{8}$$

- Linear (Entity + Relational). In this setting, we represent a triple by concatenating its entity vector and relational vector, and composing them with standard linear layer. This is a special case of the gated neural network, where α is always set to zero.

3.2 Results and Analysis

Table 2 shows the empirical results of the baseline methods and our method on two relations. We can find that the performances of these methods are consistent on two relations. Among all these algorithms, *Surface Matching* is the worst performer. The reason lies in that it does not capture the interaction between two entities in a triple, which is very important for KB translation. *Hints Similarity* outperforms *Surface Matching* by taking into account the relatedness of entities by web search results. However, its improvement over *Surface Matching* is not significant enough because it ignores the relation type. That is to say, the semantic similarities between two entities in a triple remain the same for different relation types (e.g. Profession, Cause of Death). This is problematic as the relation plays an important role in discovering entity similarity. Let us take "Barack Obama （奥巴马）" and "Honolulu（火奴鲁鲁）" as an example. Their similarity in terms of "born in" relation is high, but the similarity regarding "working place" should be extremely lower.

Table 2 shows that neural network models outperform *Surface Matching* and *Hints Similarity* method, which shows the powerfulness of neural network based representation learning methods. This is because that neural network methods map English triples and Chinese triples into a unified semantic vector space, which shares some characteristics with the human who is assigned to do this task.

Among all these neural network methods, *Entity Model* is the worst method because it captures the semantics of entities separately while ignoring the semantic interaction between them. On the other hand, *Relational (Tensor) Model* only use the relational information of a triple. We can find that it shows slight improvements over *Entity Model*, which indicates the importance of relation of triple for KB translation.

Relational (Tensor) Model* is an enhanced *Relational (Tensor) Model*, and can also be viewed as a tensor composition function added by a standard linear composition function (Socher et al., 2013b). We can find that Relational (Tensor*) Model yields better performances than previous two neural models. The full model yields the best performances among all baseline methods by simultaneously leveraging entity representation, relational representation and their interaction in an adaptive method.

Average (Entity + Relational) is a straight-forward method to compose entity- and relational- representations. From Table 2, we can find that *Average* does not yield obvious improvements over previous neural models. The main reason is that average function fails to model the interaction between entity vector and relational vector, which is important to effectively capture the complex linguistic phenomena in KB triple. In *Linear (Entity + Relational)*, we concatenate v_e and v_r, and calculate their semantic composition with a simple linear layer (Socher et al., 2011). It shows some improvements over *Average (Entity + Relational)*, which demonstrates the importance of semantic composition algorithm for obtaining semantic representation of a triple.

3.3 The Effect of Gated Neural Network

We explore the effectiveness of the gated neural model for English-Chinese KB translation on our datasets. We set the gated α as a fixed value from 0.1 to 1.0, increased by 0.1. This corresponds to a special case of gated neural network with a fixed weighted ratio between v_r and $\tilde{v}$. The model with α = 1.0 means that the representation of a triple only comes from relational representation, without using any entity representation as mentioned in Section 2.2.1.

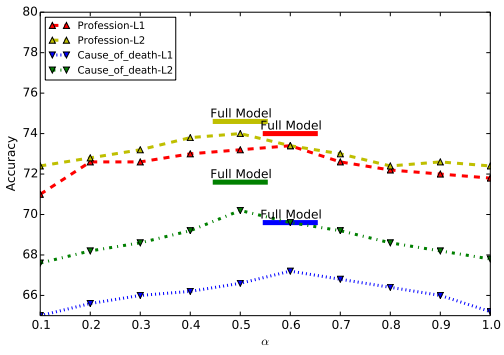


Figure 3: Experimental results with different $\alpha 1$ on the datasets.

We run experiments on two relation types with L1 and L2 norms, respectively. The results are illustrated in Figure 3. We can see that the performance of our full model is obviously better than the model with fixed trade-off weights. This is partly because that it is hard to measure the importances of entity representation and relational representation with a fixed weight for the complex phenomena in KB. The results also reveal the importance of an adaptive weighting strategy for semantic composition.

4 Related Work

We briefly describe existing studies on representation learning for natural language processing, learning continuous triple representation and gated recurrent neural network.

It is well accepted that feature representation is extremely important for natural language processing tasks. The main reason is that the effectiveness of a machine learner is highly dependent on the choice of data representation (Bengio et al., 2013). In past few decades, many studies leverage human ingenuity and prior knowledge to design hand-crafted features. Despite the effectiveness of feature engineering in some tasks, it is time consuming and typically fails to extract the discriminative information from the data. Recently, neural networks show their strengths in learning continuous representations of word/phrase (Mikolov et al., 2013), sentence (Socher et al., 2013c), document (Le and Mikolov, 2014), KB triple (Bordes et al., 2013) from data without any feature engineering. This study belongs to the family of neural network based representation learning for natural language processing tasks.

There are several neural network approaches proposed to model relational data, especially in the multi-relational case, where different kinds of relations are used to connect various data entities. Previous works focus on knowledge link prediction and triplet classification. Bordes et al. (2011) provide a structured embedding model where the regression loss was replaced by a ranking loss for learning embeddings of entities. Bordes et al. (2012) introduced a semantic matching energy function to map different instances in the same semantic vector space. Bordes et al. (2013) exploited a canonical model, which modeled relations by regarding the task as a translations operation on the low-dimensional embeddings of entities. Wang et al. (2014) made an extension on the translation model of TransE (Bordes et al., 2013) by projecting KB triple in relation-specific hyperplane. In this way, they can preserve the mapping properties of relation to some extent. Lin et al. (2015) considered that an entity may have multiple aspects, and different relations focused on different aspects of entities. They introduced TransR by representing entities and relations in distinct semantic vector space. Another related approach is introduced by Socher et al. (2013a), which used a neural tensor network to learn relational compositionality. Our relational representation method is similar to Socher et al. (2013a), which is on the basis of multiplicative vector-based semantic composition (Mitchell and Lapata, 2010).

The use of gated neural network in this paper shares some characteristics with the emerging gated recurrent neural network (Cho et al., 2014; Chung et al., 2015) and Long Short-Term Memory (Hochreiter and Schmidhuber, 1997; Tai et al., 2015). Standard recurrent neural network uses a set of shared parameters to represent a sequence of variable length to a vector representation. It suffers from the problem of vanishing gradient, which means that the influence of a given input on the hidden layer either decays or blows up exponentially. LSTM and gated recurrent neural network address this problem by adding several neural gates (e.g. input and forget gates) to adaptively memorize new content and forget history content. The gated neural network used in this work aims at integrating entity representation and relational representation in triple vector in an adaptive way.

5 Conclusion

We introduce a neural network approach for Knowledge Base (KB) translation from English (source) to Chinese (target) in this paper. We represent a triple in KB with an adaptive composition model, which produces triple representation by capturing entity- and relational- level information. The parameters of neural networks are effectively estimated with a ranking-type hinge loss function. We compare against several baseline methods on two KB translation datasets. Experimental results show that, our method performs better than baseline methods. In addition, we show that the newly introduced adaptive composition model improves standard composition method such as neural tensor network in terms of translation accuracy.

6 Acknowledgments

The authors give great thanks to Lifu Huang (RPI) and Shen Liu (HIT) for the fruitful discussions and related web crawlers work. We also would like to thank four anonymous reviewers for their valuable comments and suggestions. This work was supported by the National High Technology Development 863 Program of China (No. 2015AA015407), National Natural Science Foundation of China (No. 61632011 and No. 61370164).

References

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

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Analysis and Machine Intelligence*.

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *AAAI*.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence and Statistics*, pages 127–135.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Gosse Bouma, Sergio Duarte, and Zahurul Islam. 2009. Cross-lingual alignment and completion of wikipedia templates. In *Proceedings of the Third International Workshop on Cross Lingual Information Access: Addressing the Information Need of Multilingual Societies*, pages 21–29. ACL.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. *arXiv preprint arXiv:1502.02367*.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Bing Qin, Heng Ji, and Ting Liu. 2016. A language-independent neural network for event detection. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 66.

Sergio Ferrández, Antonio Toral, Óscar Ferrández, Antonio Ferrández, and Rafael Muñoz. 2009. Exploiting wikipedia and eurowordnet to solve cross-lingual question answering. *Information Sciences*, 179(20):3473–3488.

Jens Graupmann, Ralf Schenkel, and Gerhard Weikum. 2005. The spheresearch engine for unified ranked retrieval of heterogeneous xml and web documents. In *Proceedings of the 31st international conference on Very large data bases*, pages 529–540. VLDB Endowment.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, pages 297–304.

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

Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. 2006. *Information retrieval in folksonomies: Search and ranking*. Springer.

Rodolphe Jenatton, Nicolas L. Roux, Antoine Bordes, and Guillaume R Obozinski. 2012. A latent factor model for highly multi-relational data. In *Advances in Neural Information Processing Systems 25*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *ICML*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

Christopher D Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *NIPS*.

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

Chen Shi, Shujie Liu, Shuo Ren, Shi Feng, Mu Li, Ming Zhou, Xu Sun, and Houfeng Wang. 2016. Knowledge-based semantic embedding for machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Richard Socher, J. Pennington, E.H. Huang, A.Y. Ng, and C.D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP 2011*.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013a. Parsing with compositional vector grammars. In *ACLAnnual Meeting of the Association for Computational Linguisticsm*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Y Ng. 2013b. Reasoning with neural tensor networks for knowledge base completion. *NIPS*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013c. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP 2013*, pages 1631–1642.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015a. Training very deep networks. In *NIPS 2015*, pages 2368–2376.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015b. Highway networks. *arXiv preprint arXiv:1505.00387*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398.

Keyphrase Annotation with Graph Co-Ranking

Adrien Bougouin and **Florian Boudin** and **Béatrice Daille**
Université de Nantes, LINA, France
{adrien.bougouin,florian.boudin,beatrice.daille}@univ-nantes.fr

Abstract

Keyphrase annotation is the task of identifying textual units that represent the main content of a document. Keyphrase annotation is either carried out by extracting the most important phrases from a document, keyphrase extraction, or by assigning entries from a controlled domain-specific vocabulary, keyphrase assignment. Assignment methods are generally more reliable. They provide better-formed keyphrases, as well as keyphrases that do not occur in the document. But they are often silent on the contrary of extraction methods that do not depend on manually built resources. This paper proposes a new method to perform both keyphrase extraction and keyphrase assignment in an integrated and mutual reinforcing manner. Experiments have been carried out on datasets covering different domains of humanities and social sciences. They show statistically significant improvements compared to both keyphrase extraction and keyphrase assignment state-of-the art methods.

1 Introduction

Keyphrases are words and phrases that give a synoptic picture of what is important within a document. They are useful in many tasks such as document indexing (Gutwin et al., 1999), text categorization (Hulth and Megyesi, 2006) or summarization (Litvak and Last, 2008). However, most documents do not provide keyphrases, and the daily flow of new documents makes the manual keyphrase annotation impractical. As a consequence, automatic keyphrase annotation has received special attention in the NLP community and many methods have been proposed (Hasan and Ng, 2014).

The task of automatic keyphrase annotation consists in identifying the main concepts, or topics, addressed in a document. Such task is crucial to access relevant scientific documents that could be useful for researchers. Keyphrase annotation methods fall into two broad categories: keyphrase extraction and keyphrase assignment methods. Keyphrase extraction methods extract the most important words or phrases occurring in a document, while assignment methods provide controlled keyphrases from a domain-specific terminology (controlled vocabulary).

The automatic keyphrase annotation task is often reduced to the sole keyphrase extraction task. Unlike assignment methods, extraction methods do not require domain specific controlled vocabularies that are costly to create and to maintain. Furthermore, they are able to identify new concepts that have not been yet recorded in the thesaurus or ontologies. However, extraction methods often output ill-formed or inappropriate keyphrases (Medelyan and Witten, 2008), and they produce only keyphrases that actually occur in the document.

Observations made on manually assigned keyphrases from scientific papers of specialized domains show that professional human indexers both extract keyphrases from the content of the document and assign keyphrases based on their knowledge of the domain (Liu et al., 2011). Here, we propose an approach that mimics this behaviour and jointly extracts and assigns keyphrases. We use two graph representations, one for the document and one for the specialized domain. Then, we apply a co-ranking algorithm to perform both keyphrase extraction and assignment in a mutually reinforcing manner. We perform

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2945–2955, Osaka, Japan, December 11-17 2016.

experiments on bibliographic records in three domains belonging to humanities and social sciences: linguistics, information science and archaeology. Along with this approach come two contributions. First, we present a simple yet efficient assignment extension of a state-of-the-art graph-based keyphrase extraction method, TopicRank (Bougouin et al., 2013). Second, we circumvent the need for a controlled vocabulary by leveraging reference keyphrases from training data and further take advantage of their relationship within the training data.

2 Related Work

2.1 Keyphrase extraction

Keyphrase extraction is the most common approach to tackle the automatic keyphrase annotation task. Previous work includes many approaches (Hasan and Ng, 2014), from statistical ranking (Salton et al., 1975) to binary classification (Witten et al., 1999), through graph-based ranking (Mihalcea and Tarau, 2004) of keyphrase candidates. As our approach uses graph-based ranking, we focus on the latter. For a detailed overview of keyphrase extraction methods, refer to (Hasan and Ng, 2010; Hasan and Ng, 2014).

Since the seminal work of Mihalcea and Tarau (2004), graph-based ranking approaches to keyphrase extraction are becoming increasingly popular. The original idea behind these approaches is to build a graph from the document and rank its nodes according to their importance using centrality measures.

In TextRank (Mihalcea and Tarau, 2004), the input document is represented as a co-occurrence graph in which nodes are words. Two words are connected by an edge if they co-occur in a fixed-sized window of words. A random walk algorithm is used to iteratively rank the words, then extract the keyphrases by concatenating the most important words.

The random walk algorithm simulates the "voting concept", or recommendation: a node is important if it is connected to many other nodes, and if many of those are important. Thus, let $G = (V, E)$ be an undirected graph with a set of vertices V and a set of edges E, and let $E(v_i)$ be the set of nodes connected to the node v_i. The score $S(v_i)$ of a vertex v_i is initialized to 1 and computed iteratively until convergence using the following equation:

$$S(v_i) = (1 - \lambda) + \lambda \sum_{v_j \in E(v_i)} \frac{S(v_j)}{|E(v_j)|} \quad (1)$$

where λ is a damping factor that has been set to 0.85 by Brin and Page (1998) for a trade-off between ranking accuracy and fast convergence.

Following up the work of Mihalcea and Tarau (2004), Wan and Xiao (2008) added edge weights (co-occurrence numbers) to the random walk and further improved the graph with co-occurrence information borrowed from similar documents. To extract keyphrases from a document, they first look for five similar documents, then use them to add new edges between words within the graph and reinforce the weight of existing edges. Liu et al. (2010) biased multiple graphs with topic probabilities drawn from LDA (Latent Dirichlet Allocation) (Blei et al., 2003), to rank the words regarding each graph and to merge the rankings together. This method performs as many rankings as the number of topics and gives higher importance scores to high-ranking words for as many topics as possible. By doing so, Liu et al. (2010) increase the topic coverage provided by the extracted keyphrases.

Most recently, Zhang et al. (2013) and Bougouin et al. (2013) explored further the value of topics for keyphrase extraction. Zhang et al. (2013) used graph co-ranking to improve the method of Liu et al. (2010) by introducing LDA topics right inside the graph. Bougouin et al. (2013) proposed to represent topics as clusters of similar keyphrase candidates within the document (i.e. words and phrases from the document), to rank these topics instead of the words and to extract the most representative candidate as keyphrase for each important topic. As our work extends that of Bougouin et al. (2013), we present a detailed description of their method in Section 3.1.

2.2 Keyphrase assignment

Keyphrase assignment provides keyphrases for every document of a specific domain using a controlled vocabulary. Dissimilar to keyphrase extraction, keyphrase assignment also aims to provide keyphrases that do not occur within the document. This task is more difficult than keyphrase extraction and has, therefore, seldom been employed for automatic keyphrase annotation. The state-of-the art method for keyphrase assignment is KEA++ (Medelyan and Witten, 2006).

KEA++ uses a domain-specific thesaurus to assign keyphrases to a document. First, keyphrase candidates are selected among the n-grams of the document. N-grams that do not match a thesaurus entry are either removed or substituted by a synonym that matches a thesaurus entry. This candidate selection approach induces a limitation of keyphrase assignment, refered to as keyphrase indexing by Medelyan and Witten (2006), because it only assigns keyphrases if they occur within the document. Second, KEA++ exploits the semantic relationships between keyphrase candidates within the thesaurus as the main feature of a Naive Bayes classifier. Compared to similar methods without domain specific resources, KEA++ achieves better performance. However, such resources are not readily available for most domains, and if so, they could be quickly out of date. The application scenario of KEA++ are thus restricted.

Our proposition is to model with graphs both keyphrase extraction and assignment and to take benefit of this unified modelling to perform accurate keyphrase annotation.

3 Co-ranking for Keyphrase Annotation

This section presents TopicCoRank[1], our keyphrase annotation method built on the existing method TopicRank (Bougouin et al., 2013) to which we add keyphrase assignment. We first detail TopicRank, then present our contributions.

3.1 TopicRank

TopicRank is a graph-based keyphrase extraction method that relies on the following five steps:

1. **Keyphrase candidate selection.** Following previous work (Hasan and Ng, 2010; Wan and Xiao, 2008), keyphrase candidates are selected from the sequences of adjacent nouns and adjectives that occur within the document (`/(N|A)+/`).

2. **Topical clustering.** Similar keyphrase candidates c are clustered into topics based on the words they share. Bougouin et al. (2013) use a Hierarchical Agglomerative Clustering (HAC) with a stem overlap similarity (see equation 2) and an average linkage. At the beginning, each keyphrase candidate is a single cluster, then candidates sharing an average of $1/4$ stemmed words with the candidates of another cluster are iteratively added to the latter.

$$\text{sim}(c_i, c_j) = \frac{|\text{stems}(c_i) \cap \text{stems}(c_j)|}{|\text{stems}(c_i) \cup \text{stems}(c_j)|} \tag{2}$$

   where $\text{stems}(c_i)$ is the set of stemmed words of the keyphrase candidate c_i.

3. **Graph construction.** A complete graph is built, in which nodes are topics and edges are weighted according to the strength of the semantic relation between the connected topics. The closer are the pairs of candidates $\langle c_i, c_j \rangle$ of two topics t_i and t_j within the document, the stronger is their semantic relation $w_{i,j}$:

$$w_{i,j} = \sum_{c_i \in t_i} \sum_{c_j \in t_j} \text{dist}(c_i, c_j) \tag{3}$$

$$\text{dist}(c_i, c_j) = \sum_{p_i \in \text{pos}(c_i)} \sum_{p_j \in \text{pos}(c_j)} \frac{1}{|p_i - p_j|} \tag{4}$$

   where $\text{pos}(c_i)$ represents all of the offset positions of the first word of the keyphrase candidate c_i.

[1]TopicCoRank is open source and publicly available at `https://github.com/adrien-bougouin/KeyBench/tree/coling_2016/`

4. **Topic ranking.** Topics t are ranked using the importance score $S(t_i)$ of the TextRank formula, as modified by Wan and Xiao (2008) to leverage edge weights:

$$S(t_i) = (1 - \lambda) + \lambda \sum_{t_j \in E(t_i)} \frac{w_{ij} S(t_j)}{\sum_{t_k \in E(t_j)} w_{jk}} \quad (5)$$

5. **Keyphrase selection.** One keyphrase candidate is selected from each of the N most important topics: the first occurring keyphrase candidate.

Our work extends TopicRank to assign domain-specific keyphrases that do not necessarily occur within the document. First, we add a second graph representing the domain and unify it to the topic graph. Second, we define a co-ranking scheme that leverages the new graph. Finally, we redefine the keyphrase selection step for both extracting and assigning keyphrases.

3.2 Unified graph construction

TopicCoRank operates over a unified graph that connects two graphs representing the document topics, the controlled keyphrases and the relations between them (see Fig. 1). The controlled keyphrases are the keyphrases that were manually assigned to training documents. Considering the manually assigned keyphrases as the controlled vocabulary circumvents the need for a manually produced controlled vocabulary and also allows us to further take advantage of the semantic relatonship between the domain-specific (controlled) keyphrases. Because controlled keyphrases are presumably non-redundant, we do not topically cluster them as we do for keyphrase candidates.

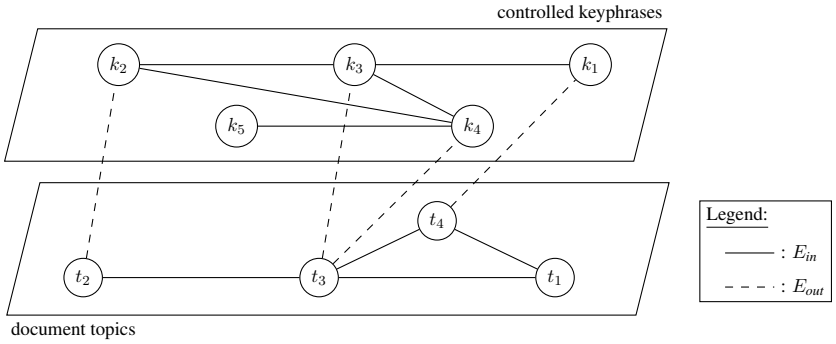


Figure 1: Example of a unified graph constructed by TopicCoRank and its two kinds of edges

Let $G = (V = T \cup K, E = E_{in} \cup E_{out})$ denote the unified graph. Topics $T = \{t_1, t_2, ..., t_n\}$ and controlled keyphrases $K = \{k_1, k_2, ..., k_m\}$ are vertices V connected to their fellows by edges $E_{in} \subseteq T \times T \cup K \times K$ and connected to the other vertices by edges $E_{out} \subseteq K \times T$ (see Fig. 1).

To unify the two graphs, we consider the controlled keyphrases as a category map and connect the document to its potential categories. We create an unweighted edge $\langle k_i, t_j \rangle \in E_{out}$ to connect a controlled keyphrase k_i and a topic t_j if the controlled keyphrase is a member of the topic, i.e. a keyphrase candidate of the topic[2]. We create an edge $\langle t_i, t_j \rangle \in E_{in}$ or $\langle k_i, k_j \rangle \in E_{in}$ between two topics t_i and t_j or two controlled keyphrases k_i and k_j when they co-occur within a sentence of the document or as keyphrases of a training document, respectively. Edges $\langle t_i, t_j \rangle \in E_{in}$ are weighted by the number of times ($w_{i,j}$) topics t_i and t_j occur in the same sentence within the document. Edges $\langle k_i, k_j \rangle \in E_{in}$ are weighted by the number of times ($w_{i,j}$) keyphrases k_i and k_j are associated to the same document among the training documents. Doing so, the weighting scheme of edges E_{in} is equivalent for both topics and controlled keyphrases. This equivalence is essential to ensure that not only controlled keyphrases occurring in the document can be assigned by properly co-ranking topics and controlled keyphrases.

[2]To accept inflexions, such as plural inflexions, we follow Bougouin et al. (2013) and perform the comparison with stems.

3.3 Graph-based co-ranking

TopicCoRank gives an importance score $S(t_i)$ or $S(k_i)$ to every topic or controlled keyphrase using graph co-ranking (see equations 6 and 7). Our graph co-ranking simulates the voting concept based on inner and outer recommendations.

The inner recommendation is similar to the recommendation computed in previous work (Bougouin et al., 2013; Mihalcea and Tarau, 2004; Wan and Xiao, 2008). The inner recommendation R_{in} comes from nodes of the same graph (see equation 8). A topic or a controlled keyphrase is important if it is strongly connected to other topics or controlled keyphrases, respectively.

The outer recommendation influences the ranking of topics by controlled keyphrases and of controlled keyphrases by topics. The outer recommendation R_{out} comes from nodes of the other graph (see equation 9). A topic or a controlled keyphrase gain more importance if it is connected to important controlled keyphrases or an important topic, respectively.

$$S(t_i) = (1 - \lambda_t)\ R_{out}(t_i) + \lambda_t\ R_{in}(t_i) \tag{6}$$

$$S(k_i) = (1 - \lambda_k)\ R_{out}(k_i) + \lambda_k\ R_{in}(k_i) \tag{7}$$

$$R_{in}(v_i) = \sum_{v_j \in E_{\text{in}}(v_i)} \frac{w_{ij} S(v_j)}{\sum_{v_k \in E_{\text{in}}(v_j)} w_{jk}} \tag{8}$$

$$R_{out}(v_i) = \sum_{v_j \in E_{\text{out}}(v_i)} \frac{S(v_j)}{|E_{out}(v_j)|} \tag{9}$$

where v_i is a node representing a keyphrase or a topic. λ_t and λ_k are parameters that control the influence of the inner recommendation over the outer recommendation ($0 \leqslant \lambda_t \leqslant 1$ and $0 \leqslant \lambda_k \leqslant 1$) for the topics and the controlled keyphrases, respectively.

3.4 Keyphrase annotation

Keyphrases are extracted and assigned from the N-best ranked topics and controlled keyphrases, regardless of their nature.

We extract topic keyphrases using the former TopicRank strategy. Only one keyphrase is extracted per topic: the keyphrase candidate that first occurs within the document.

We assign controlled keyphrases only if they are directly or transitively connected to a topic of the document. If the ranking of a controlled keyphrase has not been affected by a topic of the document nor by controlled keyphrases connected to topics, then its importance score is not related to the content of the document and it should not be assigned.

At this step, two variants of TopicCoRank performing either extraction or assignment can be proposed, namely TopicCoRank$_{extr}$ and TopicCoRank$_{assign}$. If keyphrases are only extracted from the topics, we obtain TopicCoRank$_{extr}$. If keyphrases are only assigned from the controlled keyphrases, we obtain TopicCoRank$_{assign}$.

4 Experimental Setup

4.1 Datasets

We conduct our experiments on data from the DEFT-2016 benchmark datasets (Daille et al., 2016)[3] in three domains: linguistics, information Science and archaeology. Table 1 shows the factual information

[3]Data has been provided by the TermITH project for both DEFT-2016 and this work. Parallely, the subset division has been modified for the purpose of DEFT-2016. Therefore, we use the same data as DEFT-2016, but the subset division is different. The subset division we used for our experiences can be found here: `https://github.com/adrien-bougouin/KeyBench/tree/coling_2016/datasets/`

about the datasets. Each dataset is a collection of 706 up to 718 French bibliographic records collected from the database of the French Institute for Scientific and Technical Information[4] (Inist). The bibliographic records contain a title of one scientific paper, its abstract and its keyphrases that were annotated by professional indexers (one per bibliographic record). Indexers were given the instruction to assign reference keyphrases from a controlled vocabulary and to extract new concepts or very specific keyphrases from the titles and the abstracts. Each dataset is divided into three sets: a test set, used for evaluation; a training set (denoted as train), used to represent the domain; and a development set (denoted as dev), used for parameter tuning.

Corpus	**Linguistics**			**Information Science**			**Archaeology**		
	train ⊃ dev		test	train ⊃ dev		test	train ⊃ dev		test
Documents	515	100	200	506	100	200	518	100	200
Tokens/Document	161	151	147	105	152	157	221	201	214
Keyphrases	8.6	8.8	8.9	7.8	10.0	10.2	16.9	16.4	15.6
Missing Keyphrases (%)	60.6	63.2	62.8	67.9	63.1	66.9	37.0	48.4	37.4

Table 1: Dataset statistics. "Missing" represents the percentage of keyphrases that cannot be retrieved within the documents.

The amount of missing keyphrases, i.e. keyphrases that cannot be extracted from the documents, shows the importance of keyphrase assignment in the context of scientific domains. More than half of the keyphrases of linguistics and information science domains can only be assigned, which confirms that these two datasets are difficult to process with keyword extraction approaches alone.

4.2 Document preprocessing

We apply the following preprocessing steps to each document: sentence segmentation, word tokenization and Part-of-Speech (POS) tagging. Sentence segmentation is performed with the PunktSentenceTokenizer provided by the Python Natural Language ToolKit (NLTK) (Bird et al., 2009), word tokenization using the Bonsai word tokenizer[5] and POS tagging with MElt (Denis and Sagot, 2009).

4.3 Baselines

To show the effectiveness of our approach, we compare TopicCoRank and its variants (TopicCoRank$_{extr}$ and TopicCoRank$_{assign}$) with TopicRank and KEA++. For KEA++, we use the thesauri maintained by Inist[6] to index the bibliographic records of Linguistics, Information Science and Archaeology.

4.4 TopicCoRank setting

The λ_t and λ_k parameters of TopicCoRank were tuned on the development sets, and set to 0.1 and 0.5 respectively. This empirical setup means that the importance of topics is much more influenced by controlled keyphrases than other topics, and that the importance of controlled keyphrases is equally influenced by controlled keyphrases and topics. In other words, the domain has a positive influence on the joint task of keyphrase extraction and assignment.

5 Experimental Results

This section presents and analyses the results of our experiments. For each document of each dataset, we compare the keyphrases outputed by each method to the reference keyphrases of the document. From the comparisons, we compute the macro-averaged precision (P), recall (R) and f1-score (F) per dataset and per method.

[4] http://www.inist.fr

[5] The Bonsai word tokenizer is a tool provided with the Bonsai PCFG-LA parser: http://alpage.inria.fr/statgram/frdep/fr_stat_dep_parsing.html

[6] Thesauri are available from: http://deft2016.univ-nantes.fr/download/traindev/

5.1 Macro-averages results

Table 2 presents the macro-averaged precision, recall and f1-score in percentage when 10 keyphrases are extracted/assigned for each dataset by TopicRank, KEA++, TopicCoRank$_{extr}$, TopicCoRank$_{assign}$ and TopicCoRank. First, we observe that the assignment baseline KEA++ mostly achieves the lowest performance, which is surprising compared to the performance reported by Medelyan and Witten (2006). The first reason for this observation is that KEA++ is restricted to thesauri entries while most keyphrases are missing within our documents. The second reason is that KEA++ relies on rich thesauri that contain an important amount of semantic relations between the entries, while our (real application) thesauri have a modest amount of semantic relations between the entries.

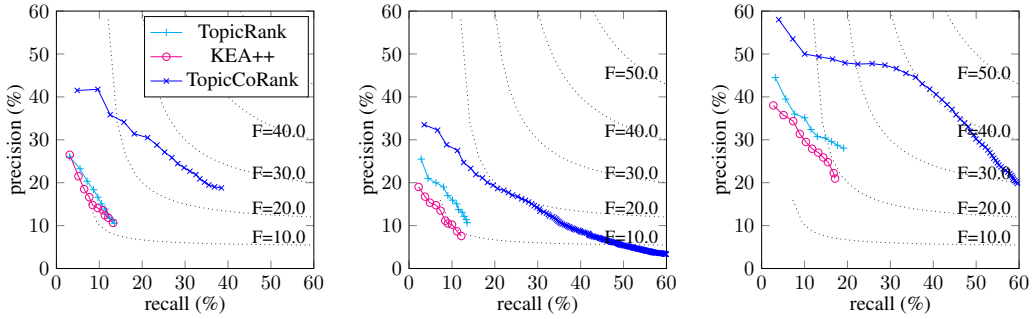
Overall, using graph co-ranking significantly outperforms TopicRank and KEA++. Comparing TopicRank to TopicCoRank$_{extr}$ shows the positive influence of the domain (controlled keyphrases) on the ranking of the topics. TopicCoRank$_{assign}$ outperforms every method, including TopicCoRank$_{extr}$ and TopicCoRank. Controlled keyphrases are efficiently ranked and the predominance of missing keyphrases in the dataset leads to a better performance of TopicCoRank$_{assign}$ over TopicCoRank.

Method	Linguistics			Information Science			Archaeology		
	P	R	F	P	R	F	P	R	F
TopicRank	11.82	13.1	11.9	12.1	12.8	12.1	27.5	19.7	21.8
KEA++	11.6	13.0	12.1	9.5	10.2	9.6	23.5	16.2	18.8
TopicCoRank$_{extr}$	15.9	18.2	16.7†	15.9	16.2	15.6†	39.6	26.4	31.0†
TopicCoRank$_{assign}$	**25.8**	**29.6**	**27.2**†	**19.9**	**20.0**	**19.5**†	**49.6**	**33.3**	**39.0**†
TopicCoRank	24.5	28.3	25.9†	19.4	19.6	19.0†	46.6	31.4	36.7†

Table 2: Results of TopicCoRank and the baselines at 10 keyphrases for each dataset. Precision (P), Recall (R) and F-score (F) are reported in percentages. † indicates a significant F-score improvement over TopicRank and KEA++ at 0.001 level using Student's t-test.

5.2 Precision/recall curves

Additionally, we follow Hasan and Ng (2010) and analyse the precision-recall curves of TopicRank, KEA++ and TopicCoRank. To generate the curves, we vary the number of evaluated keyphrases (cut-off) from 1 to the total number of extracted/assigned keyphrases and compute the precision and recall for each cut-off. Such representation gives a good appreciation of the advantage of a method compared to others, especially if the other methods achieve performances in the *Area Under the Curve* (AUC).



(a) Linguistics (b) Information Science (c) Archaeology

Figure 2: Precision-recall curves of TopicRank, KEA++ and TopicCoRank for each dataset

Figure 2 shows the precision/recall curves of TopicRank, KEA++ and TopicCoRank on each dataset. The final recall for the methods does not reach 100% because the candidate selection method does not provide keyphrases that do not occur within the document, as well as candidates that do not fit the POS tag pattern `/(N|A)+/`. Also, because TopicRank and TopicCoRank topically cluster keyphrase candidates

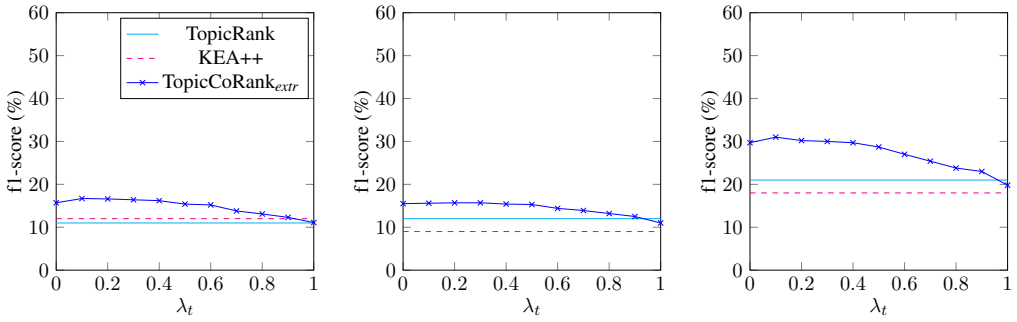
and output only one candidate per topic, their final recall is lowered every time a wrong keyphrase is chosen over a correct one from the topic.

We observe that the curve for TopicCoRank is systematically above the others, thus showing improvements in the area under the curve and not just in point estimate such as f1-score. Also, the final recall of TopicCoRank is much higher than the final recall of TopicRank and KEA++.

5.3 Extraction vs. assignment

As TopicCoRank is the first method for simultaneously extracting and assigning keyphrases, we perform an additional experiment that shows to which extent extraction and assignment contribute to the final results. To do so, we show the behavior of the extraction and the assignment depending on the influence of the inner recommendation on the ranking for each (test) document of each dataset.

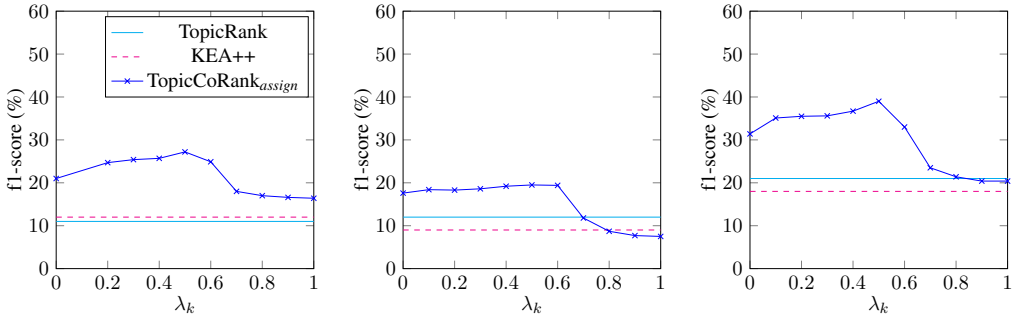
Fig. 3 shows the behavior of TopicCoRank$_{extr}$ when λ_t varies from 0 to 1. When $\lambda_t = 0$, only the domain influences the ranking of the topics. Slightly equivalent to KEA++, TopicCoRank$_{extr}$ with $\lambda_t = 0$ mainly extracts keyphrases from topics connected to controlled keyphrases. When $\lambda_t = 1$, the domain does not influence the ranking and the performance of TopicCoRank$_{extr}$ is in the range of TopicRank's performance. Overall, the performance curve of TopicCoRank$_{extr}$ decreases while λ_t increases. Thus, the experiment demonstrates that the domain has a positive influence on the keyphrase extraction.



(a) Linguistics (b) Information Science (c) Archaeology

Figure 3: Behavior of TopicCoRank$_{extr}$ depending on λ_t ($\lambda_k = 0.5$)

Fig. 4 shows the behavior of TopicCoRank$_{assign}$ when λ_k varies from 0 to 1. When $\lambda_k = 0$, only the document influences the ranking of the controlled keyphrases. As for TopicCoRank$_{extr}$ when $\lambda_t = 0$, TopicCoRank$_{assign}$ is slightly similar to KEA++ when $\lambda_k = 0$. When $\lambda_k = 1$, TopicCoRank$_{assign}$ always outputs the same keyphrases: the ones that are the most important in the domain. The first half of the curve increases, showing that the relations between the controlled keyphrases have a positive influence on the ranking of the controlled keyphrases. Conversely, the second half of the curve decreases. Thus, the sole domain is not sufficient for keyphrase annotation.



(a) Linguistics (b) Information Science (c) Archaeology

Figure 4: Behavior of TopicCoRank$_{assign}$ depending on λ_k ($\lambda_t = 0.1$)

Toucher : le tango des sens. Problèmes de sémantique lexicale **(The French verb 'toucher': the tango of senses. A problem of lexical)**

A partir d'une hypothèse sur la sémantique de l'unité lexicale 'toucher' formulée en termes de forme schématique, cette étude vise à rendre compte de la variation sémantique manifestée par les emplois de ce verbe dans la construction transitive directe 'C0 toucher C1'. Notre étude cherche donc à articuler variation sémantique et invariance fonctionnelle. Cet article concerne essentiellement le mode de variation co-textuelle : en conséquence, elle ne constitue qu'une première étape dans la compréhension de la construction des valeurs référentielles que permet 'toucher'. Une étude minutieuse de nombreux exemples nous a permis de dégager des constantes impératives sous la forme des 4 notions suivantes : sous-détermination sémantique, contact, anormalité, et contingence. Nous avons tenté de montrer comment ces notions interprétatives sont directement dérivables de la forme schématique proposée.

Keyphrases : *Français* (French); *modélisation* (modelling); *analyse distributionnelle* (distributional analysis); *interprétation sémantique* (semantic interpretation); *variation sémantique* (semantic variation); *transitif* (transitive); *verbe* (verb); *syntaxe* (syntax) and *sémantique lexicale* (lexical semantics).

Figure 5: Example of a bibliographic record in Linguistics (`http://cat.inist.fr/?aModele=afficheN&cpsidt=16471543`)

5.4 Qualitative example

To show the benefit of TopicCoRank, we compare it to TopicRank on one of our bibliographic records in Linguistics (see Figure 5). Over the nine reference keyphrases, TopicRank successfully identifies two of the reference keyphrases: "lexical semantics" and "semantic variation". TopicCoRank successfully identifies seven of them: "lexical semantics", "verb", "semantic variation", "French", "syntax", "semantic interpretation" and "distributional analysis".

TopicCoRank mostly outperforms TopicRank because it finds keyphrases that do not occur within the document: "French", "syntax", "semantic interpretation", and "distributional analysis". Some keyphrases, such as "French", are frequently assigned because they are part of most of the bibliographic records of our dataset[7] (48.9% of the Linguistics records contain "French" as a keyphrase); Other keyphrases, such as "semantic interpretation", are assigned thanks to their strong connection with controlled keyphrases occurring in the abstract (e.g. "lexical semantics").

Interestingly, the performance of TopicCoRank is not only better thanks to the assignment. For instance, we observe keyphrases, such as "verb", that emerge from topics connected to other topics that distribute importance from controlled keyphrases (e.g. "semantic variation").

6 Conclusion

In this paper, we have proposed a co-ranking approach to performing keyphrase extraction and keyphrase assignment jointly. Our method, TopicCoRank, builds two graphs: one with the document topics and one with controlled keyphrases (training keyphrases). We designed a strategy to unify the two graphs and rank by importance topics and controlled keyphrases using a co-ranking vote. We performed experiments on three datasets of different domains. Results showed that our approach benefits from both controlled keyphrases and document topics, improving both keyphrase extraction and keyphrase assignment baselines. TopicCoRank can be used to annotate keyphrases in scientific domains in a close way of professional indexers.

Acknowledgments

This work was supported by the French National Research Agency (TermITH project – ANR-12-CORD-0029) and by the TALIAS project (grant of CNRS PEPS INS2I 2016, `https://boudinfl.github.io/talias/`).

References

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

[7] Yet, TopicCoRank does not assign "French" to every bibliographic records.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction. In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 543–551, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Sergey Brin and Lawrence Page. 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30(1):107–117.

Béatrice Daille, Sabine Barreaux, Florian Boudin, Adrien Bougouin, Damien Cram, and Amir Hazem. 2016. Indexation d'articles scientifiques présentation et résultats du défi fouille de textes deft 2016. In *Actes de 12e Défi Fouille de Texte (DEFT)*, pages 1–12, Paris, France, July. Association pour le Traitement Automatique des Langues.

Pascal Denis and Benoît Sagot. 2009. Coupling an Annotated Corpus and a Morphosyntactic Lexicon for State-of-the-Art POS Tagging with Less Human Effort. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation (PACLIC)*, pages 110–119, Hong Kong, December. City University of Hong Kong.

Carl Gutwin, Gordon Paynter, Ian Witten, Craig Nevill Manning, and Eibe Frank. 1999. Improving Browsing in Digital Libraries with Keyphrase Indexes. *Decision Support Systems*, 27(1):81–104.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in Unsupervised Keyphrase Extraction: Making Sense of the State-of-the-Art. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters (COLING)*, pages 365–373, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic Keyphrase Extraction: A Survey of the State of the Art. In *Proceedings of the Association for Computational Linguistics (ACL)*, Baltimore, Maryland, June. Association for Computational Linguistics.

Anette Hulth and Beáta B. Megyesi. 2006. A study on automatically extracted keywords in text categorization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 537–544, Sydney, Australia, July. Association for Computational Linguistics.

Marina Litvak and Mark Last. 2008. Graph-Based Keyword Extraction for Single-Document Summarization. In *Proceedings of the Workshop on Multi-Source Multilingual Information Extraction and Summarization*, pages 17–24, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic Keyphrase Extraction Via Topic Decomposition. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing(EMNLP)*, pages 366–376, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhiyuan Liu, Xinxiong Chen, Yabin Zheng, and Maosong Sun. 2011. Automatic Keyphrase Extraction by Bridging Vocabulary Gap. In *Proceedings of the 15th Conference on Computational Natural Language Learning*, pages 135–144, Stroudsburg, PA, USA. Association for Computational Linguistics.

Olena Medelyan and Ian H Witten. 2006. Thesaurus Based Automatic Keyphrase Indexing. In *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, pages 296–297. ACM.

Olena Medelyan and Ian H. Witten. 2008. Domain-Independent Automatic Keyphrase Indexing with Small Training Sets. *Journal of the American Society for Information Science and Technology*, 59(7):1026–1040, may.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order Into Texts. In Dekang Lin and Dekai Wu, editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 404–411, Barcelona, Spain, July. Association for Computational Linguistics.

Gerard Salton, Andrew Wong, and Chungshu Yang. 1975. A Vector Space Model for Automatic Indexing. *Communication ACM*, 18(11):613–620, November.

Xiaojun Wan and Jianguo Xiao. 2008. Single Document Keyphrase Extraction Using Neighborhood Knowledge. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, pages 855–860. AAAI Press.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill Manning. 1999. KEA: Practical Automatic Keyphrase Extraction. In *Proceedings of the 4th ACM Conference on Digital Libraries*, pages 254–255, New York, NY, USA. ACM.

Fan Zhang, Lian'en Huang, and Bo Peng. 2013. WordTopic-MultiRank: A New Method for Automatic Keyphrase Extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 10–18, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

What's in an Explanation? Characterizing Knowledge and Inference Requirements for Elementary Science Exams

Peter Jansen[1*], **Niranjan Balasubramanian**[2], **Mihai Surdeanu**[3], **and Peter Clark**[4]
[1]School of Information, University of Arizona, Tucson, AZ
[2]Dept. of Computer Science, Stony Brook University, Stony Brook, NY
[3]Dept. of Computer Science, University of Arizona, Tucson, AZ
[4]Allen Institute for Artificial Intelligence, Seattle, WA
[*]Corresponding author: pajansen@email.arizona.edu

Abstract

QA systems have been making steady advances in the challenging elementary science exam domain. In this work, we develop an explanation-based analysis of knowledge and inference requirements, which supports a fine-grained characterization of the challenges. In particular, we model the requirements based on appropriate sources of evidence to be used for the QA task. We create requirements by first identifying suitable sentences in a knowledge base that support the correct answer, then use these to build explanations, filling in any necessary missing information. These explanations are used to create a fine-grained categorization of the requirements. Using these requirements, we compare a retrieval and an inference solver on 212 questions. The analysis validates the gains of the inference solver, demonstrating that it answers more questions requiring complex inference, while also providing insights into the relative strengths of the solvers and knowledge sources. We release the annotated questions and explanations as a resource with broad utility for science exam QA, including determining knowledge base construction targets, as well as supporting information aggregation in automated inference.

1 Introduction

Elementary science exams have recently become a common test of question answering (QA) models. Clark and Etzioni (2016) argue that these exams are an excellent benchmark for natural language processing (NLP) systems in many respects, both testing students for many different kinds of knowledge and inference abilities at varying levels of difficulty, while also allowing for a direct comparison of machine to human performance in the science domain on a standardized evaluation. Many different QA approaches have been developed and evaluated on these and similar exams, with methods using a range of representations from unstructured (BOW) lexical semantic models (Fried et al., 2015), structured relation-based representations (Clark et al., 2016; Khot et al., 2015), more complex first-order formalisms (Khot et al., 2015), and other inference methods (Khashabi et al., 2016). Together in concert, these methods can achieve substantial improvements in overall performance, with a 71% accuracy (i.e. passing performance) on one test set (Khashabi et al., 2016).

In this work, we focus on developing a deeper understanding of this problem domain by implementing a fine-grained characterization of the knowledge and inference requirements for science exam QA, driven by generating and annotating gold explanations that justify the correct answer. We believe that this can provide many tangible benefits. First, we can obtain a fine-grained assessment of the abilities of different QA systems to identify areas of competency, and those that need improvement. Second, the detailed knowledge requirements can serve as a specification for knowledge extraction. Third, it can support QA methods that can use problem solving strategies and knowledge tailored to the specific requirements of a given question. Finally, it can support design of QA systems that can provide explanations for why they choose an answer. In this last respect, multiple-choice elementary science questions currently lack a direct way to quantitatively assess systems on this aspect.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2956–2965, Osaka, Japan, December 11-17 2016.

Specifying broadly applicable knowledge requirements and explanations poses two main challenges. First, questions can be answered in many ways, and depending on the knowledge source used the type of knowledge ascribed to the question can differ. We follow a pragmatic approach, building on prior work in knowledge categorization, and use knowledge types that correspond to commonly used semantic structures relating to the automatic construction of knowledge bases (KB). Clark et al. (2013) compiled an initial analysis of the questions in these datasets, and identified 7 broad categories of knowledge and inference requirements. However, this analysis forced a single knowledge type for each question, for example *causality*, and from our detailed analysis we find that many types of knowledge are necessary to arrive at the correct answer, e.g., *causality, actions, and purposes*.

A second challenge relates to grounding the requirements and the explanations in appropriate resources such that they can facilitate automated analysis and provide compact, reusable, and linked knowledge for inference. To this end, we use grade appropriate texts, and first identify relevant sentences or nuggets of information that can serve as explanations or supports for the current answers. We then fill in sentences that provide missing links connecting knowledge and terminology in the sentences, while taking care to ensure as much reuse as possible.

We apply this methodology to obtain requirements on a set of 212 questions from an open standardized elementary science exam dataset, and present an analysis of these requirements. This work makes the following contributions:

- We construct a detailed characterization of the knowledge and inference requirements of elementary science exams, highlighting the prevalence of complex inference questions, which require inference methods that combine many facts across multiple types of knowledge.
- We provide an empirical analysis of the performance of different QA methods on questions with specific knowledge and inference requirements, demonstrating that while existing QA systems considerably outperform information retrieval (IR) methods on difficult questions, many of the more complex forms of inference remain to be addressed.
- We provide a knowledge resource in the form of gold explanations for hundreds of science exam questions, as well as annotation describing question-centered and explanation-centered knowledge and inference requirements. We believe this resource will be broadly useful for characterizing performance on current and future models, as well as developing automated methods supporting knowledge type identification, inference, and explanation construction.

2 Related Work

Analyzing knowledge and inference requirements is a first necessary step in designing QA systems. For factoid QA tasks, these requirements are often stated in terms of broad question categories (e.g., What, When, How) and finer-grained types for expected answers (e.g., cities, person, organization). Factoid QA systems use classifiers to identify the types of question and expected answers, which are subsequently used to select specific problem solving routines, and to filter answer candidates (Harabagiu et al., 2000; Li and Roth, 2006; Roberts and Hickl, 2008). For non-factoid QA tasks, requirements are often stated in terms of elements in knowledge representation ontologies. For instance, Chaudhri et al. (2014) study requirements for a QA task defined over AP Biology texts using relations and categories from the CLIB ontology (Barker et al., 2001). Some benchmarks, such as bAbI (Weston et al., 2016), are created to test specific reasoning abilities and come with a grouping of questions into the corresponding categories (e.g., negation reasoning, causal reasoning).

Our work aims to provide similar requirements for the elementary science QA benchmark (Clark and Etzioni, 2016). Prior analyses on this benchmark includes Clark et al. (2013), who identified seven broad kinds of knowledge and inference in three categories: *retrieval questions*, making use of taxonomic, definitional, or property knowledge; *inference questions*, testing a knowledge of causality, processes, or identifing examples of situations; and *domain-specific models*. Crouse and Forbus (2016) further identified questions that involve qualitative reasoning (13% of total), and provide a sub-categorization of these. Here we build upon these prior works and provide both a more fine-grained characterization of the knowledge types required to answer these questions, along with manually curated answer explanations.

This allows us to compare the relative strengths and weaknesses of different QA systems from knowledge and inference requirements identified using both bottom-up (from explanations) and top-down (from questions) approaches.

More broadly and with respect to explanations, there is a recent trend towards emphasizing interpretable models for machine learning (e.g. Ribeiro et al. (2016)) that are able to produce human-readable explanations for their reasoning, both to improve human trust in automated inference, as well as to verify that a given model is accurately capturing the aspects of complex reasoning required for a given task. We view this work as complementary, here characterizing the knowledge and inference requirements that an automated reasoning method for science exams must meet to assemble compelling human-readable explanations as part of the inference process.

3 Knowledge and Inference Analysis

Estimating knowledge and inference requirements is challenging for many reasons. Chief among these is that a question can be answered in many different ways, using different types of knowledge and reasoning depending on the sources of evidence used. At one extreme, with a large knowledge base (KB), many questions can be answered by simply retrieving a fact from the KB that readily provides the correct answer. At the other extreme, with a modest KB, multiple pieces of information have to be aggregated together using some inference method to arrive at the correct answer. A further difficulty in multiple choice exams is that a QA system may select the correct answer, but for the wrong reasons stemming from difficulties in retrieval, inference, or from simply using a backoff strategy (e.g. guessing)[1]. Question answering systems in the science and medical domains should also target providing human-readable explanations for why the selected answer is correct. We examine knowledge requirements for this explainable question answering task, which suggests that, at the very least, requirements must be grounded in explanations drawn from a reasonable collection of target sources of evidence.

Towards this goal, we develop an explanation-centered approach using appropriate grade-level resources, constructing gold natural language explanations that detail why a given answer is correct, and deriving a fine-grained distribution of common inference relations from these explanations. In this section, we first provide a question-centered analysis expanded to a larger set of questions compared to prior work, and demonstrate the challenges with this approach. We then present a fine-grained analysis using the explanation-centered approach on the same set of questions.

Questions: For the following analyses, we make use of the 432 training questions in the AI2 Elementary Science Questions set[2], collected from standardized 3^{rd} to 5^{th} grade science exams in 14 US states.

3.1 Question-centered Analysis

Figure 1 shows the distribution of knowledge and inference requirements when extending the question-centered analysis of Clark et al. (2013) to the larger AI2 elementary questions set. We find two differences when compared to their original analysis on 50 4^{th} grade questions from the New York Regents Science Exam: First, the distribution on this larger question set exhibits a much higher proportion of complex inference (77%) compared to retrieval methods. Second, even though we annotated one knowledge category per question according to the original procedure, we find that many of the complex inference questions naturally require integrating several different kinds of knowledge to arrive at the answer, with more than a third of the questions requiring at least two knowledge types.

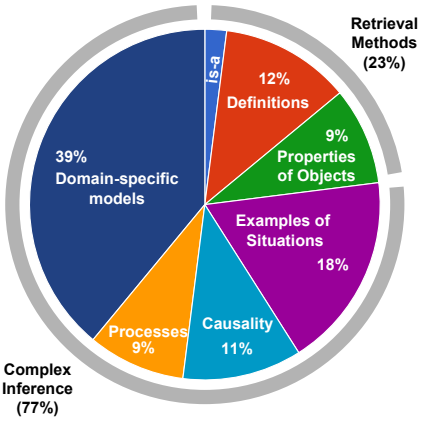


Figure 1: Knowledge types required to correctly answer a given question in the elementary science exam dataset.

[1]Jansen, Sharp, Surdeanu, and Clark (submitted) showed in their error analysis that, for elementary science questions, both retrieval and inference methods produce completely incorrect explanations approximately 20% of the time. A retrieval model produced complete explanations for 45% of questions, while an inference model incorporating intersentence aggregation produced complete explanations for 60% of questions.

[2]The original question set is available at: `http://allenai.org/data.html`

Question	Which of these organisms has cells with cell walls?
Answer Choices	(A) bluebird **(B) A pine tree** (C) A ladybug (D) A fox squirrel
Explanation	A pine tree is a kind of plant. A cell wall is a part of a plant cell.
Question	What form of energy causes an ice cube to melt?
Answer Choices	(A) mechanical (B) magnetic (C) sound **(D) heat**
Explanation	An ice cube is a solid. Changing from a solid to a liquid is called melting. Melting happens when solids are heated. Heated means added heat. Heat is a kind of energy.
Question	Which of the following events involves a consumer and producer in a food chain?
Answer Choices	(A) A cat eats a mouse. **(B) A deer eats a leaf.** (C) A hawk eats a mouse. (D) A snake eats a rat.
Explanation	A leaf is a kind of plant. A deer is a kind of animal. In a food chain, an animal is a consumer. In a food chain, green plants are producers.

Table 1: Explanations for three shorter example questions, including one simpler question about the property of an object (having cell walls), an explicitly causal question (melting), and one question about the role of two entities in a process or model (the food chain). Dashed underlines indicate bridge sentences.

3.2 Explanation-centered Analysis

3.2.1 Gold Explanations

For each question, we create gold explanations that describe the inference needed to arrive at the correct answer. Our goal is to derive an explanation corpus that is grounded in grade-appropriate resources. Accordingly, we use two elementary study guides, a science dictionary for elementary students, and the Simple English Wiktionary as relevant corpora. For each question, we retrieve relevant sentences from these corpora and use them directly, or use small variations when necessary. If relevant sentences were not located, then these were constructed using simple, straightforward, and grade-level appropriate language. Approximately 18% of questions required specialized domain knowledge (e.g. spatial, mathematical, or other abstract forms) that did not easily lend itself to simple verbal description, which we removed from consideration. This resulted in a total of 363 gold explanations.[3]

In addition to using grade-appropriate language, the following considerations were taken in developing the explanation corpus, with the aim to provide broad utility for a variety of tasks from automated knowledge type identification to information aggregation models of inference:

- *Single topic:* To help facilitate automated analysis and reuse, explanations were broken into multiple sentences, with each sentence focusing on a single aspect of the explanation.
- *Reuse:* To assist in identifying overlaps in knowledge between questions, the same explanation sentences were reused as much as possible, where applicable.
- *Sentence Linking:* To support automated inference, the terminology used in different explanation sentences is explicitly linked through "bridge sentences" that include both terms. For example, if one sentence mentions *melting*, and another mentions *heated*, here we include an explicit sentence that links the two, such as *"melting happens when solids are heated"*. Where appropriate, we also include other latent knowledge that may not be explicitly required to answer a question, but would likely be available to a human and link related questions. For example, for a process question about a specific stage of the *life cycle*, we also include a brief overview of where this stage fits in the process as a whole (e.g. *egg to baby to child to adult*). In this way many of the explanations appear overly verbose to a human, but contain enough information to make the inference explicit, link highly related topics, and evaluate the knowledge requirements for automated methods.

Example explanations are shown in Table 1. The 363 gold explanations contain a total of 1272 sentences, or an average of 4 sentences per explanation. With respect to reuse, 943 unique sentences appear across these explanations, with 180 appearing in more than one explanation, and the remaining 763 occurring in only a single explanation.[4]

[3]The gold explanations developed in this work are also available at: `http://allenai.org/data.html`

[4]Frequently-recurring sentences highlight common themes in questions: Sentences such as *"Evaporation is when a liquid changes to a gas", "Sunlight means solar energy", and "Metals conduct electricity"* are 5 of the 42 sentences found in the explanations of 4 or more questions.

Knowledge Type	Prop.	Structure and Examples
Retrieval Types		
Taxonomic	83%	*HYPONYM* is a kind of *HYPERNYM* a <*HYPO: plant*> is a kind of <*HYPER: living thing*>
Definition	64%	*ARG1* means *ARG2 (can be definitions or synonyms)* <*ARG1: cooling*> means <*ARG2: decreasing heat*> *(definition)*
Properties	41%	*PROPERTY* is a property of *ARG1* <*ARG1: iron*> is <*PROP: magnetic*>
PartOf	22%	*MERONYM* is a part of *HOLONYM*. a <*HOLO: bicycle*> has <*MERO: two pedals*>.
Contains	17%	*ARG1* contains *ARG2*. <*ARG1: soil*> contains <*ARG2: nutrients*> that plants absorb through their roots
ExampleOf	9%	*ARG1* is an example of *ARG2* an example of a <*ARG1: seasonal change*> is <*ARG2: growing thick fur*>
MadeOf	8%	*ARG1* is made of *ARG2*. a <*ARG1: rock*> is a hard substance composed of <*ARG2: minerals*>
Inference Supporting Types		
Actions	73%	*SUBJECT ACTION OBJECT* <*SUBJ: bees*> <*ACTION: eat*> <*OBJ: pollen*>
UsedFor	33%	*WHO* uses *WHAT*, and *WHY*. <*WHO: squirrels*> <*WHAT: store food in the autumn*> <*WHY: to eat over the winter*>
Source	23%	*WHO* generates/is a source of *WHAT*, and *HOW* . natural gas is can be burned in power stations to *make electricity* (note sourceof+generate)
IsWhen	22%	*ARG1* is when *ARG2* happens. *(often used for defining events/processes)* <*ARG1: mechanical weathering*> is when <*ARG2: rocks are broken down by mechanical ...*>
VehicleFor	17%	*WHAT* happens by/through some means or *VEHICLE*. ... when <*WHAT: pollen is carried from flower to flower*> <*VEHICLE: by pollinating animals*>
Requires	12%	*WHO* requires *WHAT*, and *WHY*. <*WHO: animals*> need to <*WHAT: eat food*> <*WHY: to get nutrients required for survival*>
Negation	12%	*ARG1* is not *ARG2*. aluminum is not <*NOT: magnetic*>
Duration	10%	*ARG1* has some *DURATION* many birds <*ARG1: migrate to warmer places*> <*DUR: for the winter*>
Complex Inference Types		
Changes	45%	*WHO/LABEL* changes *WHAT*, *FROM* something *INTO* something else. <*LABEL: boiling*> means changing from a <*FROM: solid*> to a <*INTO: liquid*>
Causes	21%	*ARG1* causes *ARG2*. <*ARG1: friction*> causes <*ARG2: the temperature of an object to increase*>
Transfer	21%	*WHAT* gets transfered from a *SOURCE* to *DESTINATION*, and *HOW* this happens. ... breaks down food into <*WHAT: nutrients*> that can be <*HOW: absorbed*> by <*DEST: the body*>
IfThen	14%	*IF* a condition occurs, *THEN* a result happens. if <*IF: something is on fire*>, <*THEN: it burns*>
Relationship	12%	As *EVENT1* happens, *EVENT2* will also happen. <*EVENT1: A decrease in the amount of water*> will cause <*EVENT2: a decrease in plant populations*>
Process	8%	A group of relations, e.g. A *PROCESS STAGE* takes some *ACTION* causing a *RESULT*. as an <*STAGE: adult bird*>, <*ACTION: it will reproduce*>, <*RESULT:starting the life cycle...*>

Table 2: Fine-grained knowledge types, and the proportion of explanations that include at least one instance of a given type. Types are *n-ary* relations, containing between two and five arguments each. Note that a given example sentence often includes more than one relation, as in the case of "cooling means decreasing heat", which includes both a *Definition* relation (i.e. means), and a *Change* relation (i.e. decreasing heat).

3.2.2 Fine-grained Knowledge Types

To characterize the knowledge present in these gold explanations, we annotated the explanation sentences with a fine-grained set of knowledge types which reuses many of the types from Clark et al. (2013) and includes additional types derived from frequently observed semantic structures in the explanation sentences. Each explanation sentence can contain more than one type (e.g. *"boiling means increasing temperature"* contains both a Definition type (*boiling means ...*) and a Change type (*increasing temperature*)). All types were manually annotated using a graphical annotation tool[5]. Due to the time involved in this process, we annotated 212 questions, or approximately 50% of the original set of questions.

Table 2 shows the new fine-grained set of knowledge types, their relative frequencies, and the associated semantic structures. About 21% of the annotated questions had between 1 and 5 instances of types

[5]This simple graphical annotation tool is included with the data distribution.

in their explanations, while 31% had between 6 and 10 instances. The remainder of questions with more than 10 relations across their explanations were largely complex questions that included latent or other background knowledge in their explanations.

The fine-grained types can also be grouped into three broad sets: *Retrieval Types* include binary relations commonly found in taxonomies, dictionaries, and property databases. *Inference Supporting Types* tend to ground the knowledge in the complex inference relations. This includes describing the vehicle that enables something to happen, it's purpose, it's needs, and specific actions that it can take. *Complex Inference Types* describe changes situated in particular contexts, such as causality (e.g. *X causes Y*), transfers (e.g. *X transfers from Y to Z*), and process knowledge (e.g. *Stage A follows Stage B*). Here, while our *Retrieval Types* are binary relations, both the *Complex Inference* and *Inference Supporting Types* can be viewed as *n-ary* relations or light semantic frames, often with two to five "slots" to fill.

4 QA Analysis

Here we conduct an empirical analysis of the performance of two types of QA solvers using the question-centered and explanation-centered views of knowledge and inference types.

4.1 Knowledge Bases

We evaluate performance on two knowledge bases, one free text, the other semi-structured:

Study Guides: A collection of free text from six resources: study guides for two elementary science exams, a teacher's manual, a set of flashcards, and two dictionary resources: a science dictionary for kids, and the open-domain Simple English Wiktionary[6]. A total of 3,832 science-domain sentences and 17,473 open-domain definition sentences were included.

Aristo TableStore: An open collection[7] of approximately 100 semi-formal tables (approximately 10k rows, 30k cells) containing knowledge tailored to elementary science exams, constructed using a mixture of manual and automatic methods (Dalvi et al., 2016). The table knowledge spans across knowledge types, from properties and taxonomic knowledge to causality, processes, and domain models. Each table encodes an aspect of the science domain (e.g., animal adaptations, measuring instruments, energy conversions, etc.), where variations are typically enumerated (e.g. "a <*grill*> converts <*chemical energy*> to <*heat energy*>", "a <*flashlight*> converts <*electrical energy*> into <*light energy*>", etc.).

4.2 Solvers

We characterize QA approaches from two families: a baseline that uses "learning to rank" (L2R) with information retrieval (IR) features, and more recent inference models.

Retrieval Model:

We use an L2R model which finds answers by scoring passage level evidence for each answer choice from the unstructured textual knowledge sources. Our implementation is based on the candidate ranking (CR) model described in Jansen et al. (2014). Short passages are scored based on how similar they are to the words in the question and the corresponding answer choice. The similarity scores are computed using cosine similarity of *tf.idf* representations of the question and passages, and used in a L2R framework to produce the final ranking of the answer choices. We created two versions of the solver: one that uses the study guide collection, and the other with a textual representation of the Aristo TableStore. Apache Lucene[8] is used to index and retrieve passages.

Inference Models:

For inference, we use two models that operate over a structured knowledge base of tables (TableStore). TableILP (Khashabi et al., 2016) is a model that finds answers by building a graph of chained facts, i.e., rows in the knowledge tables, to arrive at the answer. Starting from the question, the model selects rows from a table, and then iteratively uses the selected rows to find rows in other tables, as linkable facts,

[6] http://simple.wiktionary.org
[7] http://allenai.org/data.html
[8] http://lucene.apache.org

AKBC'13 Knowledge Type	N	Proportion Correct L2R (StudyGuides)	L2R (TableStore)	ILP (TableStore)	STITCH (Tablestore)
Retrieval Methods					
Taxonomic	4	75% (0%)	75%	100% (+25%)	100% (+25%)
Definition	27	56% (-7%)	63%	59% (-4%)	63% (0%)
Properties	19	21% (-11%)	32%	53% (+21%)	53% (+21%)
Complex Inference					
Examples	40	35% (-13%)	48%	70% (+22%)	58% (+10%)
Causality	30	30% (-10%)	40%	60% (+20%)	53% (+13%)
Processes	26	52% (+4%)	48%	36% (-12%)	64% (+16%)
Domain-specific Models	66	38% (+6%)	32%	43% (+11%)	53% (+21%)
Overall	212	39% (-4%)	43%	54% (+11%)	56% (+13%)

Table 3: Proportion of questions answered correctly broken down by AKBC'13 knowledge types. Values in parentheses reflect absolute differences with the L2R solver that uses the TableStore knowledge base.

until it arrives at facts that contain or overlap with the answer choices. Rows are selected based on lexical overlap. This graph building problem is modeled using Integer Linear Program (ILP) to find paths that maximize QA performance. STITCH is an alternative algorithm for reasoning over the same tables. It achieves similar overall performance using different heuristics for matching a question to table rows. For both inference models, we made use of the stock models, and did not incorporate any further training. As described below, we make use of a different question corpus and an expanded knowledge base compared to Khashabi et al., evaluating on approximately twice as many questions as were originally reported, including many questions at a higher grade level, and including questions from 13 other state exams in addition to the original New York Regents questions. Similarly, we make use of an expanded knowledge base that is approximately twice the size of that used in Khashabi et al. (2016). As such, our overall inference model performance is slightly lower than they originally reported.

Questions: We compare performance on the 212 elementary science questions from Section 3.2.2 that included a gold explanation annotated with the knowledge and inference types.[9]

4.3 Question-centered Evaluation

We first characterize performance of the two solvers using the seven broad question-centered categories of Clark et al. (2013), with performance shown in Table 3. Overall, the L2R models have lower performance than the inference models. This is in line with our explanation-based analysis of the requirements, which showed that there are more complex inference questions than there are simple retrieval ones. The results also show that the gains in the inference solvers are not completely due to tailored knowledge. Using the highly tailored knowledge base as a retrieval corpus shows a small benefit (+4%), whereas using the knowledge via appropriate inference substantially increases performance (+13%).

In terms of performance on questions with particular knowledge and inference requirements, we find that bulk of the performance benefit for the inference solvers comes from addressing more complex inference questions, rather than simply answering more of the (subjectively easier) retrieval questions. Performance on *Example Identification* and *Causality* questions using the L2R model increases 10-13% when switching from the study guide knowledge base to the Tablestore, and further increases by 10-22% when the inference solvers are used in conjunction with the Tablestore, demonstrating that some complex questions separately benefit from highly tailored knowledge and the capacity to aggregate multiple pieces of that knowledge to form a solution. Conversely, the more challenging *Process* and *Domain Model* categories are not directly benefited by the tailored Tablestore knowledge resource, but show moderate benefits when this knowledge is combined with the inference solvers to form more complex solutions.

On the balance, this high-level analysis shows that inference methods designed to aggregate multiple pieces of information from a knowledge base specifically benefit questions requiring complex inference, more than the contribution of tailoring a similarly-sized retrieval-centered knowledge base alone.

[9]Note that because this set excludes the 18% of questions that did not easily lend themselves to textual explanation, and that 70% of these excluded questions were categorized as requiring model-based reasoning, this evaluation set can be viewed as somewhat easier and containing fewer extremely difficult questions than the broader corpus.

Knowledge Type	N	L2R (Corpus)	Knowledge Advantage	L2R (TableStore)	Inference Advantage	ILP (TableStore)	STITCH (TableStore)
Retrieval Types							
Taxonomic	176	39% (-7%)	→	46%	→	56% (+10%)	55% (+9%)
Definition	135	39% (-2%)	X	41%	→	56% (+15%)	55% (+14%)
Properties	86	38% (+2%)	X	36%	→	49% (+13%)	56% (+20%)
PartOf	47	43% (+11%)	←	32%	→	45% (+13%)	68% (+36%)
Contains	35	29% (-11%)	→	40%	X	43% (+3%)	40% (+0%)
ExampleOf	19	47% (+5%)	X	42%	→	63% (+21%)	63% (+21%)
MadeOf	16	50% (-13%)	→	63%	X	56% (-7%)	63% (+0%)
Inference Supporting Types							
Action	154	40% (-4%)	X	44%	→	54% (+10%)	57% (+13%)
UsedFor	70	44% (0%)	X	44%	→	59% (+15%)	70% (+26%)
SourceOf/Generate	49	43% (+2%)	X	41%	→	53% (+12%)	65% (+24%)
IsWhen/IsCalled	46	28% (-22%)	→	50%	→	70% (+20%)	54% (+4%)
Vehicle	35	40% (-3%)	X	43%	→	54% (+11%)	54% (+14%)
Requires	26	39% (+12%)	←	27%	→	54% (+27%)	50% (+23%)
Negation	26	15% (-7%)	X	22%	→	44% (+22%)	52% (+30%)
Duration	21	57% (0%)	X	57%	→	67% (+10%)	48% (-9%)
Complex Inference Types							
Change	96	34% (-8%)	→	42%	→	53% (+11%)	51% (+9%)
Cause	45	38% (0%)	X	38%	→	56% (+18%)	53% (+15%)
Transfer	45	44% (-9%)	→	53%	→	64% (+11%)	62% (+9%)
Relationship	25	28% (0%)	X	28%	→	44% (+16%)	36% (+8%)
IfThen	29	41% (+6%)	X	35%	→	41% (+6%)	45% (+10%)
Process (Content/Roles)	25	44% (-17%)	→	61%	X	61% (0%)	57% (-4%)
Process (Structural)	12	25% (-50%)	→	75%	←	58% (-17%)	50% (-25%)
Average Performance		39% (-4%)	X	43%	→	54% (+11%)	56% (+13%)

Table 4: Performance on questions whose gold explanations contain *at least one* instance of a given type.Values in parentheses reflect absolute differences with the score of the L2R solver that uses the TableStore knowledge base. Arrows represent where performance on a given relation shows a benefit from either knowledge base, or switching from a retrieval to an inference solver, where an "X" signifies no benefit.

4.4 Explanation-centered Evaluation

We conduct an explanation-centered evaluation to understand the comparative finer-grained competencies of the solvers. Table 4 compares performance relative to whether the *gold explanation* for a given question contains *at least one* instance of the specific type. If a question contains a specific type according to the annotation, then we assert that type of knowledge or inference is required to answer (and produce an explanation for) that science exam question. We note three main observations.

First, the inference solver outperforms L2R solvers across the board, with strong improvements when there are retrieval or inference-supporting types, and smaller improvements for explanations with complex inference types, except for the causal types (+18% gain in P@1). Conversely, despite gains with inference solvers, questions of some complex inference types, such as *If/Then* conditional sequences, or *Coupled Directional Relationships* (i.e. *as X increases, Y decreases*), have low overall absolute performance, pointing to areas for future improvement.

Second, there is a variance in performance across the broader groups when switching over to Tablestore for the L2R solver. For example, *Taxonomic, Containment, and MadeOf* see benefits, whereas *Definition, Properties, and ExampleOf* do not. *PartOf*, and *Requirement* types work better with study guides rather than Tablestore knowledge, suggesting the entirety of the study guide knowledge is not subsumed by the tablestore. Similar variance exist for the complex inference types, as well.

Third, the broad types of the question-based analysis can be inadequate in some cases. The broad *Process* category in Table 3 showed some general improvement with inference methods, but the fine-grained analysis shows the opposite. This is likely because the broad *Process* category is an umbrella for several different types of questions. Some query only a very specific stage of a process (like a *producer's role in the food chain*), and are amenable to being answered by single sentences found using retrieval methods. Others require integrating structural knowledge across many stages of a process (such as *from egg to adult in the life cycle*), and appear to require much more complex inference to explainably answer.

5 Conclusions

In this work we developed an explanation-centered fine-grained characterization of elementary science exams, helping improve our understanding of this problem domain. Rather than existing in easily decoupled categories, these exams show a rich distribution of knowledge and inference requirements, with a majority requiring complex inference. The analyses validate the gains with some inference-based solvers by showing that they specifically address questions requiring complex inference. While a modern inference solver shows steady improvements in complex inference broadly, performance for a number of specific types of complex inference is still quite low, and provides targets for future work.

We release the annotated questions and explanations as a knowledge resource that can be broadly useful for science exam QA. As question variety, difficulty, and domain-specificity increase, any *single* solver is unlikely to work well across the board. This motivates development of solver ensembles and question-specific solver selection, which need the capacity to automatically recognize a given question's knowledge and inference requirements. We believe this resource may have a range of other uses, from providing a specification of knowledge base construction targets, to informing methods of information aggregation in automated inference.

References

Ken Barker, Bruce Porter, and Peter Clark. 2001. A library of generic concepts for composing knowledge bases. In *Proceedings of First International Conference on Knowledge Capture*, pages 14–21.

Vinay K. Chaudhri, Daniel Elenius, Andrew Goldenkranz, Allison Gong, Maryann E. Martone, William Webb, and Neil Yorke-Smith. 2014. Comparative analysis of knowledge representation and reasoning requirements across a range of life sciences textbooks. volume 5:51.

Peter Clark and Oren Etzioni. 2016. My computer is an honor student but how intelligent is it? standardized tests as a measure of ai. *AI Magazine*, 37(1):5–12.

Peter Clark, Philip Harrison, and Niranjan Balasubramanian. 2013. A study of the knowledge base requirements for passing an elementary science test. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, AKBC'13, pages 37–42.

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter D. Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2580–2586.

Maxwell Crouse and Kenneth D. Forbus. 2016. Elementary school science as a cognitive system domain: How much qualitative reasoning is required? In *Proceedings of Fourth Annual Conference on Advances in Cognitive Systems*.

Bhavana Dalvi, Sumithra Bhakthavatsalam, Chris Clark, Peter Clark, Oren Etzioni, Anthony Fader, and Dirk Groeneveld. 2016. IKE - an interactive tool for knowledge extraction. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction, AKBC@NAACL-HLT 2016, San Diego, CA, USA, June 17, 2016*, pages 12–17.

Daniel Fried, Peter Jansen, Gustave Hahn-Powell, Mihai Surdeanu, and Peter Clark. 2015. Higher-order lexical semantic models for non-factoid answer reranking. *Transactions of the Association for Computational Linguistics*, 3:197–210.

Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2000. Falcon: Boosting knowledge for answer engines. In *Proceedings of the Text REtrieval Conference (TREC)*, Gaithersburg, MD, USA.

Peter Jansen, Mihai Surdeanu, and Peter Clark. 2014. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. 2016. Question answering via integer programming over semi-structured knowledge. In *Proceedings of the International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 1145–1152.

Tushar Khot, Niranjan Balasubramanian, Eric Gribkoff, Ashish Sabharwal, Peter Clark, and Oren Etzioni. 2015. Exploring markov logic networks for question answering. In *EMNLP*.

Xin Li and Dan Roth. 2006. Learning question classifiers: The role of semantic information. *Natural Language Engineering*, pages 229–249.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA. ACM.

Kirk Roberts and Andrew Hickl. 2008. Scaling answer type detection to large hierarchies. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

Jason Weston, Antoine Bordes, Sumit Chopra, Sasha Rush, Bart van Merrienboer, Armand Joulin, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *Proceedings of the Fourth International Conference on Learning Representations*, volume abs/1502.05698.

"All I know about politics is what I read in Twitter": Weakly Supervised Models for Extracting Politicians' Stances From Twitter

Kristen Johnson and **Dan Goldwasser**
Department of Computer Science
Purdue University, West Lafayette, IN 47907
{john1187, dgoldwas}@purdue.edu

Abstract

During the 2016 United States presidential election, politicians have increasingly used Twitter to express their beliefs, stances on current political issues, and reactions concerning national and international events. Given the limited length of tweets and the scrutiny politicians face for what they choose or neglect to say, they must craft and time their tweets carefully. The content and delivery of these tweets is therefore highly indicative of a politician's stances. We present a weakly supervised method for extracting how issues are framed and temporal activity patterns on Twitter for popular politicians and issues of the 2016 election. These behavioral components are combined into a global model which collectively infers the most likely stance and agreement patterns among politicians, with respective accuracies of 86.44% and 84.6% on average.

1 Introduction

The trending decline in popularity of traditional media outlets and continued rise of social media usage emerged in the 2008 U.S. presidential election campaign and has continued to the present 2016 campaign. Social media platforms, such as the microblogging outlet Twitter, allow politicians to directly access the public, express their beliefs, and react to current events. Unlike its traditional media predecessors, Twitter requires politicians to compress their ideas, political stances, and reactions to 140 character long tweets. Consequently, politicians must cleverly choose how to frame controversial issues, as well as how and when to react to each other (Mejova et al., 2013; Tumasjan et al., 2010). Due to this limit, we argue that the stance of a tweet is not independent of the social context in which it was generated. Thus, for accurate predictions these social behaviors must also be modeled.

Converse to previous works which predict stance per individual tweet (SemEval, 2016), we instead present a novel approach better suited to model the dynamic political arena of Twitter, which uses the *overall* Twitter behavior per politician to predict a *politician's* stance on an issue. We explore two aspects of the problem, *stance* prediction over a wide array of issues, as well as stance *agreement and disagreement* patterns between politicians over these issues. While the two aspects are related, we argue they capture different information, as identifying agreement patterns reveals alliances and rivalries between candidates, across and within their party. In an extreme case, even the lack of Twitter activity on certain issues can be indicative of a stance.

For example, consider the three tweets on the issue of gun control shown in Figure 1. To identify the stance taken by each politician, our model combines both content and behavioral features, accumulated from all of a politician's tweets on that issue. First, the tweet's relevance to an issue can be identified using *issue* indicators (highlighted in green). Second, the similarity between the stances taken by two of the politicians (agreement) can be identified by observing differences in how the issue is *framed* (shown in yellow), a tool often used by politicians to create bias toward a stance and contextualize the discussion (Tsur et al., 2015; Card et al., 2015). Tweets (1) and (3) frame the issue as a matter of safety, while tweet (2) frames it as related to personal freedom, thus revealing the agreement and disagreement patterns between the politicians. Third, we can consider the timing of these tweets, i.e. whether these tweets are posted continually or just around events concerning gun violence. Finally, we can also use sentiment indicators (e.g., the negative sentiment of tweet (1)). Notice that each feature individually might not contain sufficient information for correct classification, but combining all aspects,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2966–2977, Osaka, Japan, December 11-17 2016.

(1) **Hillary Clinton (@HillaryClinton):** *We need to keep guns out of the hands of domestic abusers and convicted stalkers.*

(2) **Donald Trump (@realDonaldTrump):** *Politicians are trying to chip away at the 2nd Amendment. I won't let them take away our guns!*

(3) **Bernie Sanders (@SenSanders):** *We need sensible gun-control legislation which prevents guns from being used by people who should not have them.*

Figure 1: Tweets Discussing the Issue of Gun Control. Issue indicators (e.g. guns and gun-control) are highlighted in green and different frame indicators (e.g., domestic abusers or 2nd Amendment) are highlighted in yellow.

by propagating stance bias (e.g. from sentiment) to politicians who hold similar or opposing views (as determined from frame analysis), leads to a more reliable prediction.

Given the dynamic nature of political discourse on Twitter, we design our approach to use minimal supervision and naturally adapt to new issues. We build several weakly supervised local learners, whose only supervision is a small seed set of issue and frame indicators which characterize the stance of tweets (based on lexical heuristics (O'Connor et al., 2010) and framing dimensions (Card et al., 2015)), and Twitter activity statistics which capture temporally similar patterns between politicians. Our final model represents agreement and stance bias by combining these weak models into a weakly supervised joint model through Probabilistic Soft Logic (PSL), a recent probabilistic modeling framework (Bach et al., 2013). The information gained from the weakly supervised local learners is the only supervision used by PSL; the rest of the prediction is completely unsupervised. PSL combines these aspects declaratively by specifying high level rules over a relational representation of the politicians' activities (exemplified in Figure 2), which is further compiled into a graphical model called a hinge-loss Markov random field (Bach et al., 2013), and used to make predictions about stance and agreement between politicians.

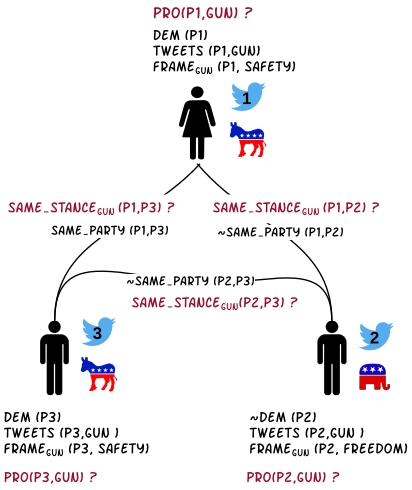


Figure 2: Relational Representation Example of Twitter Activity. P1, P2, and P3 represent 3 different politicians. Prediction target predicates (PRO and SAMESTANCE) are shown in red. Indicators of Twitter content and behavior include: DEM, TWEETS, FRAMEGUN, SAMEPARTY. GUN refers to the issue of gun control; SAFETY and FREEDOM refer to different frames for the issue.

We analyze the Twitter activity of 32 prominent U.S. politicians, including those who were U.S. 2016 presidential candidates, on 16 different issues. Our experiments demonstrate the effectiveness of our global modeling approach, which outperforms both the weak learners that provide the initial supervision and a supervised text based baseline. Our results show that understanding political discourse on Twitter requires modeling not only the word content of tweets but the social behavior associated with those tweets as well.

2 Related Work

To the best of our knowledge this is the first work predicting *politicians' stances using Twitter data, based on content, frames, and temporal activity*. Several works (Sridhar et al., 2015; Hasan and Ng, 2014; Abu-Jbara et al., 2013; Walker et al., 2012; Abbott et al., 2011; Somasundaran and Wiebe, 2010; Somasundaran and Wiebe, 2009) have studied mining opinions and predicting stances in online debate forum data by exploiting argument and threaded conversation structures, both of which are not always present in short Twitter data[1]. Social interaction and group structure has also been explored (Sridhar et al., 2015; Abu-Jbara et al., 2013; West et al., 2014). Works focusing on inferring signed social networks (West et al., 2014), stance classification (Sridhar et al., 2015), social group modeling (Huang et al., 2012), and PSL collective classification (Bach et al., 2015) are closest to our work, but these typically operate in supervised settings. Conversely, we use PSL *without direct supervision*, to assign *soft* values (0 to 1 inclusive) to output variables, rather than Markov Logic Networks, which assign *hard* (0 or 1) values to model variables and incur heavier inference time computational cost.

In recent years there has been a growing interest in analyzing political discourse in both traditional

[1]In our data set, there are few "@" mentions or retweet examples forming a conversation, thus we do not have access to argument or conversation structures for analysis.

and social media. Several previous works have explored topic framing (Tsur et al., 2015; Card et al., 2015; Baumer et al., 2015) of public statements, congressional speeches, and news articles. Other works focus on identifying and measuring political ideologies (Iyyer et al., 2014; Bamman and Smith, 2015; Sim et al., 2013; Lewenberg et al., 2016) and policies (Gerrish and Blei, 2012; Nguyen et al., 2015; Grimmer, 2010). To the best of our knowledge, this work is also the *first attempt to analyze issue framing in Twitter data*. To do so we use the frame guidelines developed by Boydstun et al. (2014). Issue framing is related to both analyzing biased language (Greene and Resnik, 2009; Recasens et al., 2013) and subjectivity (Wiebe et al., 2004).

Concerning Twitter specifically, analysis of users and political tweets has attracted considerable attention. Unsupervised and weakly supervised models of Twitter data for several various tasks have been suggested, such as user profile extraction (Li et al., 2014b), life event extraction (Li et al., 2014a), and conversation modeling (Ritter et al., 2010). Further, Eisenstein (2013) discusses methods for dealing with the unique language used in microblogging platforms.

Recently, SemEval Task 6 (SemEval, 2016) aimed to detect the stance of *individual tweets*. In contrast to this task, as well as most related work on stance prediction (e.g., those mentioned above), we *do not assume that each tweet expresses a stance*. Instead, we investigate how a politician's overall Twitter behavior, as represented by combined content and temporal indicators, is indicative of a stance (e.g., also capturing when politicians *fail to tweet about a topic*). Predicting political affiliation and other characteristics of Twitter users has been explored (Volkova et al., 2015; Volkova et al., 2014; Conover et al., 2011). Other works have focused on sentiment analysis (Pla and Hurtado, 2014; Bakliwal et al., 2013), predicting ideology (Djemili et al., 2014), analyzing types of tweets and Twitter network effects around political events (Maireder and Ausserhofer, 2013), automatic polls based on Twitter sentiment and political forecasting using Twitter (Bermingham and Smeaton, 2011; O'Connor et al., 2010; Tumasjan et al., 2010), as well as applications of distant supervision (Marchetti-Bowick and Chambers, 2012).

3 Data and Problem Setting

3.1 Data Collection

We collected tweets for 32 politicians, the 16 Republicans (all 2016 presidential candidates) and 16 Democrats (5 of which were candidates) listed in Table 1. Our initial goal was to compare politicians participating in the 2016 U.S. presidential election. To increase representation of Democrats, we collected tweets of Democrats who hold leadership roles within their party, because more well known politicians tend to focus their tweets on national rather than local (district/state) events. For all 32 politicians we have a total of 99,161 tweets: 39,353 Democrat and 59,808 Republican[2].

Based on tweet availability and politician coverage[3], we chose 16 issues (shown in Table 2) derived from the 58 questions used by `ISideWith.com` to match a user to politicians based on their responses as our stance prediction goals. These issues range over common policies including domestic and foreign policy, economy, education, environment, health care, immigration, and social issues.

Republicans	Jeb Bush, Ben Carson, Chris Christie, Ted Cruz, Carly Fiorina, Lindsey Graham, Mike Huckabee, Bobby Jindal, John Kasich, George Pataki, Rand Paul, Rick Perry, Marco Rubio, Rick Santorum, Donald Trump, Scott Walker
Democrats	Candidates: Lincoln Chafee, Hillary Clinton, Martin O'Malley, Bernie Sanders, Jim Webb
	Non-candidates: Joe Biden, Kirsten Gillibrand, John Kerry, Ben Lujan, Ed Markey, Nancy Pelosi, Harry Reid, Chuck Schumer, Jon Tester, Mark Warner, Elizabeth Warren

Table 1: Politicians Tracked in This Study. All Republicans were 2016 presidential candidates. Democrats are divided by whether or not they ran as a candidate.

[2]Our Twitter data set, keywords, and PSL scripts are available at: `purduenlp.cs.purdue.edu/projects/politicaltwitter`.

[3]For each of these 16 issues, at least 15 (with an average of 26) of the 32 politicians have tweeted on that issue; for the remaining issues, we found fewer than half (or none) of the politicians tweeted about that issue.

ISSUE	QUESTION
ABORTION	*Do you support abortion?*
ACA	*Do you support the Patient Protection and Affordable Care Act (Obamacare)?*
CONFEDERATE	*Should the federal government allow states to fly the confederate flag?*
DRUGS	*Do you support the legalization of Marijuana?*
ENVIRONMENT	*Should the federal government continue to give tax credits and subsidies to the wind power industry?*
GUNS	*Do you support increased gun control?*
IMMIGRATION	*Do you support stronger measures to increase our border security?*
IRAN	*Should the U.S. conduct targeted airstrikes on Irans nuclear weapons facilities?*
ISIS	*Should the U.S. formally declare war on ISIS?*
MARRIAGE	*Do you support the legalization of same sex marriage?*
NSA	*Do you support the Patriot Act?*
PAY	*Should employers be required to pay men and women, who perform the same work, the same salary?*
RELIGION	*Should a business, based on religious beliefs, be able to deny service to a customer?*
SOCIAL SECURITY	*Should the government raise the retirement age for Social Security?*
STUDENT	*Would you support increasing taxes on the rich in order to reduce interest rates for student loans?*
TPP	*Do you support the Trans-Pacific Partnership?*

Table 2: Sixteen Political Issues Used in This Study. Issues and their corresponding Yes/No questions were taken from `ISideWith.com`.

3.2 Data Pre-Processing

Using all tweets, we compiled a set of frequent keywords (an average of 7) for each issue. This set is small to avoid overselection, i.e., avoiding tweets about praying for a friend's *health* but keeping tweets discussing *health care*. Via Python scripts, these keywords are used to retain tweets related to the 16 issues shown in Table 2, while eliminating all irrelevant tweets (e.g., those about personal issues, campaigning, duplicates, and non-English tweets).

`ISideWith.com` uses a range of yes/no answers to their questions and provides proof (through quotes or voting records) of a politician's stance on that issue, *if available*. When unavailable, the site assigns an answer based on party lines or often provides no answer. Also, less popular politicians are not featured on the site. For these cases, we manually annotated the stance using online searches of newspapers or voting records. These stances are only used for evaluation of our predictions. Our weakly supervised approach requires *no* prior knowledge of the politician's stance, allowing it to generalize to situations such as these, where stance information is unavailable.

3.3 Prediction Goals

The collected stances represent the ground truth of whether a politician is for or against an issue. Based on these we define two target predicates using PSL notation (see Section 5.1) to capture the desired output as soft truth assignments to these predicates. The first predicate, PRO(P1, ISSUE), captures a positive stance by politician P1, on an ISSUE. A negative stance would be captured by its negation: ¬PRO(P1, ISSUE). The second target predicate, SAMESTANCE$_I$(P1, P2), classifies if two politicians share a stance for a given issue, i.e., if both are for or against an issue, where I represents 1 of the 16 issues of interest. Although the two predicates are clearly inter-dependent, we chose to model them as separate predicates since they can depend on different Twitter behavioral and content cues. Given the short and context-free style of Twitter we can often find indicators of politicians holding similar stances, *without* clear specification for which stance they actually hold.

4 Local Models of Twitter Activity

The only supervision required by our method consists of the keywords describing issues and frames, Twitter behavior patterns, and party affiliation, all of which is easily attainable and adaptable for new domains (e.g., different keywords to capture issues of another country). The weakly supervised local models described in this section capture similarities between tweet content and temporal activity patterns of users' timelines, as well as stance bias, and are used to provide the initial bias when learning the parameters of the otherwise unsupervised global PSL model.

4.1 Issue of Tweets

To capture which issues politicians are tweeting about, we used a keyword based heuristic, similar to the approach described in O'Connor et al. (2010), where each issue is associated with a small set of pre-selected keywords (as described previously). The keywords appearing in a given tweet may be mutually

exclusive (e.g., *fracking* for Environment will not appear in tweets discussing other issues); however, some may fall under multiple issues at once (e.g., *religion* may indicate the tweet refers to ISIS, Religion, or Marriage). Tweets are classified as relating to a certain issue based on the majority of matching keywords, with rare cases of ties manually resolved. The output of this classifier is all of the issue-related tweets of a politician, which are used as input for the PSL predicate TWEETS(P1, ISSUE), a binary predicate which indicates if that politician has tweeted about the issue or not.

4.2 Sentiment of Tweets

The sentiment of a tweet can indicate a politician's stance on a certain issue. OpinionFinder 2.0 (Wilson et al., 2005) is used to label each politician's issue-related tweets as positive, negative, or neutral. We observed, however, that for all politicians, a majority of tweets will be labeled as neutral. This may be caused by the difficulty of labeling sentiment for Twitter data. When this results with a politician having no positive or negative tweets, they are assigned their party's majority sentiment for that issue. The majority sentiment of a party is calculated by running all politicians' tweets through OpinionFinder and using whichever sentiment (positive or negative) is assigned the most per party. This output is used as input to the PSL predicates TWEETPOS(P1, ISSUE) and TWEETNEG(P1, ISSUE).

4.3 Content Agreement and Disagreement Patterns

We expect politicians that have a similar stance on an issue to use similar words in their tweets. To determine how well tweet content similarity captures agreement between politicians, we computed the pair-wise cosine similarity between all of the politicians' words used in tweets per issue. However, the use of similar words per issue resulted in most politicians being grouped together, even across different parties. To overcome this, we calculated the *frequency* of similar words within tweets (per issue). For each issue, all of a politician's words from tweets are aggregated and the frequency of each word is compared to all other politicians' word frequencies. Politicians, P1 and P2, are considered to have a similar LOCALSAMESTANCE$_I$(P1, P2) if their frequency counts per shared word of an issue are within the same range. For this study, we used a window of 10 (i.e., if the frequency count of a word is 30, then a count of 20 to 40 would be considered similar) to ensure that politicians who briefly mention an issue are not considered equivalent to those who discuss it more frequently.

4.4 Temporal Activity Patterns

We observed from reading Twitter feeds that most politicians tweet about an event the day it happens. However, for general issues, politicians will comment as frequently as desired to express their support or lack thereof for that particular issue. For example, Rand Paul tweeted daily in opposition of the NSA during his filibuster of the Patriot Act renewal. Conversely, Hillary Clinton has no tweets concerning the NSA or Patriot Act. To capture agreement patterns between politicians, we align their timelines based on days where they have tweeted about an issue. When two or more politicians tweet about the same issue on the same day, they are considered to have similar temporal activity, which may indicate stance agreement. This information is used as input to the predicate SAMETEMPORALACTIVITY$_I$(P1, P2).

4.5 Political Framing

Framing is a political strategy that describes the concept of how politicians word their statements in order to control the way the public views and discusses current issues. To investigate the intuition that the way politicians contextualize their tweets is strongly indicative of their stance on an issue, we compiled a list of unique keywords for each political framing dimension as described in Boydstun et al. (2014) and Card et al. (2015). We again use the keyword matching approach described in Section 4.1 to classify all tweets with a political frame. As noted in Card et al. (2015), some tweets may fall into multiple frames. After all tweets are classified, we sum over the total number of each frame type and use the frames with the maximum and second largest counts as that politician's frames for that issue. The top two frames are used because for most politicians a majority of their issue-related tweets will fall into two frames. In the event of a tie we assign the frame that appears most frequently within that politician's party. These frames are used as input to the PSL predicate FRAME(P1, ISSUE).

4.6 Temporal Framing Patterns

While we expect politicians within a party to use similar frames per issue (as captured by the PSL predicate FRAME), it is also possible that politicians will use certain frames around events. For example, when a mass shooting occurs, we observe that Democrats will tweet about enacting gun legislation and typically frame these tweets as a matter of a needed preemptive action for public safety (which falls under the *Health and Safety* frame). In reaction to this, Republicans will tweet about protecting American citizens' rights to gun ownership, which falls under the *Constitutionality* frame. Therefore, we expect similarities and differences in framing around events to indicate similarities and differences in stances and agreement patterns. To capture this idea, we combine the approaches of Sections 4.4 and 4.5: we align the politicians' timelines per issue and compare the frames used to discuss the issue-related events. When two or more politicians use the same frame for an issue on the same day, we consider them to have similar temporal framing patterns. This is used as input to the PSL predicate SAMETEMPORALFRAME$_I$(P1, P2).

5 Global Models of Twitter Activity

Information obtained from the local models alone is not strong enough to quantify stance or agreement for politicians, as shown by our baseline measurements in Section 6. Therefore, we use PSL to build global connections among the output of the local models (which acts as weak supervision), resulting in global PSL models which successively build upon the previous model in order to obtain the highest accuracy possible. In addition to the PSL predicates representing the target output (PRO and SAMESTANCE$_I$) [4] and local models (as defined in Section 4), we also use directly observed information: party affiliation, denoted DEM(P1) for Democrat and ¬DEM(P1) for Republican, and SAMEPARTY(P1, P2) to denote if two politicians belong to the same political party.

5.1 Global Modeling using PSL

PSL is a recent declarative language for specifying weighted first-order logic rules. A PSL model is specified using a set of weighted logical formulas, which are compiled into a special class of graphical model, called a hinge-loss MRF, defining a probability distribution over the possible continuous value assignments to the model's random variables and allowing the model to scale easily (Bach et al., 2015). The defined probability density function has the form:

$$P(\mathbf{Y} \mid \mathbf{X}) = \frac{1}{Z} \exp\left(-\sum_{r=1}^{M} \lambda_r \phi_r(\mathbf{Y}, \mathbf{X})\right)$$

where λ is the weight vector, Z is a normalization constant, and

$$\phi_r(\mathbf{Y}, \mathbf{X}) = (\max\{l_r(\mathbf{Y}, \mathbf{X}), 0\})^{\rho_r}$$

is the hinge-loss potential corresponding to the instantiation of a rule, specified by a linear function l_r, and an optional exponent $\rho_r \in 1, 2$. The weights of the rules are learned using maximum-likelihood estimation, which in our weakly supervised setting was estimated using the Expectation-Maximization algorithm. For more details we refer the reader to Bach et al. (2015).

Specified PSL rules have the form:

$$\lambda_1 : P_1(x) \wedge P_2(x, y) \rightarrow P_3(y), \qquad \lambda_2 : P_1(x) \wedge P_4(x, y) \rightarrow \neg P_3(y)$$

where P_1, P_2, P_3, P_4 are predicates, and $x,\ y$ are variables. Each rule is associated with a weight λ, which indicates its importance in the model. Given concrete constants *a, b* respectively instantiating the variables $x,\ y$, the mapping of the model's atoms to soft [0,1] assignments will be determined by the weights assigned to each one of the rules. For example, if $\lambda_1 > \lambda_2$, the model will prefer $P_3(b)$ to its negation. This contrasts with "classical" or other probabilistic logical models in which rules are strictly true or false. In our domain, the constant symbols correspond to politicians and predicates to: party affiliation, Twitter activity, and similarities between politicians based on temporal Twitter behaviors.

[4] In a supervised setting, jointly modeling the 2 target predicates can improve performance. Experiments using this approach yielded improvement in performance *and* a more complex model containing more parameters, resulting in slower inference.

5.2 Baseline: Using Local Classifiers Directly

To show that the local models do not provide enough information individually to make an accurate prediction, we implement a local baseline (LB) PSL model which does not take advantage of the global modeling framework. It instead learns weights over rules (shown in Table 3), which directly map the output of the local noisy classifiers described in Section 4 to PSL target predicates.

PSL Rules: LOCAL BASELINE MODEL (LB)
LOCALSAMESTANCE$_I$(P1, P2) →SAMESTANCE$_I$(P1, P2)
¬LOCALSAMESTANCE$_I$(P1, P2) →¬SAMESTANCE$_I$(P1, P2)
TWEETS(P1,ISSUE) ∧TWEETPOS(P1,ISSUE) →PRO(P1, ISSUE)
TWEETS(P1,ISSUE) ∧TWEETNEG(P1,ISSUE) →¬PRO(P1, ISSUE)

Table 3: Subset of PSL Rules Used in the Local Baseline Model.

5.3 Model 1: Agreement with Party Lines

The observation that politicians tend to vote with their political party on most issues is the basis of our initial assumptions in Model 1. The PSL rules listed in Table 4 are designed to capture this party based agreement. For some issues we initially assume Democrats (DEM) are for an issue, while Republicans (¬DEM) are against that issue, (e.g., ¬DEM(P1) →¬PRO(P1, ISSUE)), or vice versa. In the latter case, the rules of the model would change accordingly, e.g. the second rule would become ¬DEM(P1) →PRO(P1, ISSUE), and likewise for all other rules. Similarly, if two politicians are in the same party, we expect them to have the SAMESTANCE, or agree, on an issue. Though this is a strong initial assumption, the model can incorporate other indicators to overcome this bias when necessary. For all PSL rules, the reverse also holds, e.g., if two politicians are not in the same party, we expect them to have different stances.

PSL Rules: MODEL 1 (M1)
SAMEPARTY(P1, P2) →SAMESTANCE$_I$(P1, P2)
DEM(P1) →PRO(P1, ISSUE)
¬DEM(P1) →¬PRO(P1, ISSUE)
SAMEPARTY(P1, P2) ∧DEM(P1) →PRO(P2, ISSUE)
SAMEPARTY(P1, P2) ∧¬DEM(P1) →¬PRO(P2, ISSUE)
SAMEPARTY(P1,P2) ∧PRO(P1, ISSUE) ∧DEM(P1) →PRO(P2, ISSUE)
SAMEPARTY(P1, P2) ∧¬PRO(P1, ISSUE) ∧ ¬DEM(P1) →¬PRO(P2, ISSUE)

Table 4: Subset of PSL Rules Used in Model 1.

5.4 Model 2: Politicians' Twitter Activity

Model 2 builds upon the initial party line bias of Model 1. In addition to political party based information, we also include representations of the politician's Twitter activity, as shown in Table 5. This includes whether or not a politician tweets about an issue (TWEETS) as well as the sentiment of the tweets as determined in Section 4.2. The predicate TWEETPOS models if a politician tweets positively on the issue, whereas TWEETNEG models negative sentiment. Two sentiment predicates are used instead of the negation of TWEETPOS, which would cause all politicians for which there are no tweets, and hence no sentiment, on that issue to also be considered.

PSL Rules: MODEL 2 (M2)
TWEETS(P1, ISSUE) ∧DEM(P1) →PRO(P1, ISSUE)
TWEETS(P1, ISSUE) ∧¬DEM(P1) →¬PRO(P1, ISSUE)
TWEETS(P1, ISSUE) ∧TWEETS(P2, ISSUE) ∧SAMEPARTY(P1, P2) →SAMESTANCE$_I$(P1, P2)
TWEETS(P1, ISSUE) ∧TWEETS(P2, ISSUE) ∧DEM(P1) →PRO(P2, ISSUE)
TWEETS(P1, ISSUE) ∧TWEETS(P2, ISSUE) ∧¬DEM(P1) →¬PRO(P2,ISSUE)
TWEETPOS(P1, ISSUE) ∧TWEETPOS(P2, ISSUE) →SAMESTANCE$_I$(P1, P2)
TWEETPOS(P1, ISSUE) ∧TWEETNEG(P2, ISSUE) →¬SAMESTANCE$_I$(P1, P2)
TWEETPOS(P1, ISSUE) ∧TWEETPOS(P2, ISSUE) ∧DEM(P1) →PRO(P2, ISSUE)
TWEETNEG(P1,ISSUE) ∧TWEETNEG(P2,ISSUE) ∧¬DEM(P1) →¬PRO(P2,ISSUE)
TWEETPOS(P1, ISSUE) ∧TWEETPOS(P2, ISSUE) ∧SAMEPARTY(P1, P2) →SAMESTANCE$_I$(P1, P2)
TWEETPOS(P1,ISSUE) ∧TWEETNEG(P2,ISSUE) ∧¬SAMEPARTY(P1,P2) →¬SAMESTANCE$_I$(P1,P2)

Table 5: Subset of PSL Rules Used in Model 2.

5.5 Model 3: Politicians' Agreement Patterns

Table 6 presents a subset of the rules used in Model 3 to incorporate higher level Twitter information into the model. The incorporation of this information allows Model 3 to overcome Model 2 inconsistencies between stance and sentiment (e.g., when someone is attacking their opposition). Our intuition is that politicians who have similar tweets would also have similar stances on issues, which we represent with the predicate LOCALSAMESTANCE$_I$. SAMETEMPORALACTIVITY represents the idea that if politicians tweet on an issue around the same time range then they also share a stance for that issue. FRAME indicates the frame used by that politician for different issues. Finally, SAMETEMPORALFRAME$_I$ conveys that two politicians use the same frames for an issue at the same time. More details on these predicates are in Sections 4.3, 4.4, 4.5, and 4.6 respectively.

PSL Rules: MODEL 3 (M3)
LOCALSAMESTANCE$_I$(P1, P2) $\wedge$PRO(P1, ISSUE) $\rightarrow$PRO(P2, ISSUE)
SAMETEMPORALACTIVITY$_I$(P1, P2) $\wedge$SAMEPARTY(P1, P2) $\rightarrow$ SAMESTANCE$_I$(P1, P2)
SAMETEMPORALACTIVITY$_I$(P1, P2) $\wedge$FRAME(P1,ISSUE) $\wedge$FRAME(P2, ISSUE) $\rightarrow$SAMESTANCE$_I$(P1, P2)
FRAME(P1, ISSUE) $\wedge$FRAME(P2, ISSUE) $\rightarrow$SAMESTANCE$_I$(P1, P2)
FRAME(P1, ISSUE) $\wedge$FRAME(P2, ISSUE) $\wedge$SAMEPARTY(P1,P2) $\rightarrow$SAMESTANCE$_I$(P1, P2)
FRAME(P1, ISSUE) $\wedge$DEM(P1) $\rightarrow$PRO(P1, ISSUE)
FRAME(P1, ISSUE) $\wedge\neg$DEM(P1) $\rightarrow\neg$PRO(P1, ISSUE)
SAMETEMPORALFRAME$_I$(P1, P2) $\wedge$SAMEPARTY(P1, P2) $\rightarrow$SAMESTANCE$_I$(P1, P2)
SAMETEMPORALFRAME$_I$(P1, P2) $\wedge$PRO(P1, ISSUE) $\rightarrow$PRO(P2, ISSUE)

Table 6: Subset of PSL Rules Used in Model 3.

6 Experiments

6.1 Experimental Settings

Supervised Baseline: Previous works exploring stance classification typically predict stance based on an *individual item of text* (e.g., forum post or single tweet) in a *supervised* setting, making it difficult to directly compare to our approach. To facilitate comparison, we implemented a tweet-based supervised baseline, per issue. We labeled each tweet with the politician's stance (either for or against) on that tweet's issue. We trained an SVM on 80% of the politicians' tweets and tested on the remaining 20%, using 5-fold cross-validation. Because we aim to predict each politician's stance and *not* the stance of each tweet, we aggregated the SVM predictions by politician, i.e., the SVM predicts a stance for each tweet and the majority prediction among a politician's tweets is used as his or her stance. For agreement prediction, we compared this stance prediction across politicians to determine if the predicted stances agreed and whether or not this agreement was correct.

PSL Models: As described in Section 4, the data generated from the local models is used as weak supervision to initialize the PSL models described in Section 5. The Local Baseline model (LB) is initialized with only the information from the weak local models. We initialize Model 1 (M1), as described in Section 5.3, using knowledge of the politician's party affiliation. Model 2 (M2) builds upon (M1) by incorporating the results of the issue and sentiment analysis local models, as described in Sections 4.1 and 4.2 respectively. Model 3 (M3) combines all previous models with higher level knowledge of Twitter activity: tweet agreement (Section 4.3), temporal activity (Section 4.4), frames (Section 4.5), and temporal framing patterns (Section 4.6). We implement our PSL models to have an initial bias that candidates do not share a stance and are against an issue. Stances collected in Section 3.2 are used as the ground truth for evaluation of the predictions of the PSL models only, not for any form of supervision.

6.2 Experimental Results

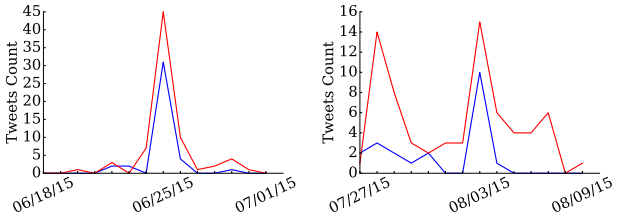
Results Per Issue: Table 7 presents the results of using the supervised baseline and our three proposed PSL models. While the supervised baseline results (SVM) are not directly comparable to our weakly supervised model, since the supervised model uses a different data split and approach, it does show that direct supervision does not lead to immediate prediction improvement and can result in weaker prediction scores. LB refers to using only the weak local models for prediction with no additional information about party affiliation. We observe that for prediction of stance (PRO) LB performs better than random chance in 11 of 16 issues; for prediction of agreement (SAMESTANCE$_I$), LB performs slightly lower overall, with only 9 of 16 issues predicted above chance. Using M1, we improve stance prediction accuracy for

Issue	Stance (Results of Pro Prediction)					Agreement (SameStance Prediction)				
	SVM	LB	M 1	M 2	M 3	SVM	LB	M 1	M 2	M 3
Abortion	61.25	81.25	96.88	96.88	96.88	44.34	49.31	93.75	93.75	95.36
ACA	87.5	96.88	100	100	100	79.7	51.61	100	100	100
Confederate	16.56	34.38	78.12	84.38	87.5	0	51.31	69.6	77.7	80.18
Drugs	48.13	87.5	78.12	88.88	96.88	44.34	50.42	63.6	84.07	84.07
Environment	69.06	53.12	78.12	78.13	81.08	65.86	45.16	65.59	68.75	71.37
Guns	84.38	93.75	93.75	93.75	93.75	57.33	48.59	68.54	99.5	99.59
Immigration	73.44	37.5	81.25	81.25	86.36	51.82	53.62	68.55	69.06	69.56
Iran	74.56	84.38	65.62	65.63	84.38	69.25	35.57	79.73	100	100
ISIS	80.0	40.32	76.28	93.75	94.44	74.19	59.68	76.28	76.28	90.04
Marriage	33.44	62.5	90.62	90.62	90.9	12.5	50.57	87.12	87.13	87.43
NSA	21.25	37.5	53.12	53.12	61.54	2.61	34.15	49.2	56.66	60.08
Pay	34.38	84.38	84.38	89.47	90.62	29.59	64.30	72.92	74.31	80.31
Religion	42.81	75	68.75	81.25	81.25	56.89	47.62	76.24	76.46	79.44
Social Security	35.31	28.12	78.12	78.13	78.13	0.91	53.76	73.25	90.03	90.88
Student	0	93.75	96.88	96.88	96.88	0	51.61	100	100	100
TPP	0	62.5	62.5	62.5	62.5	0	45.43	48.39	54.64	65.32

Table 7: Stance and Agreement Accuracy by Issue. The SVM columns show the results of the tweet-based, supervised baseline. LB columns show the results when using only the weak local models. M1 columns are the results based on party line agreement, M2 columns are the results when adding Twitter activity to M1, and M3 columns are the results when adding higher level Twitter behaviors to M1 and M2.

	Global		Rep		Dem	
	St	Ag	St	Ag	St	Ag
LB	68.36	52.49	66.80	49.10	69.92	44.86
M1	81.25	76.34	75.39	75.16	87.11	85.44
M2	85.16	87.30	81.25	84.26	89.06	91.37
M3	89.84	87.76	87.11	85.35	92.58	91.49

Table 8: Overall Accuracy for Stance (St) and Agreement (Ag) Prediction. Global represents the accuracy over all politicians, while Rep and Dem refer to Republicans or Democrats only.



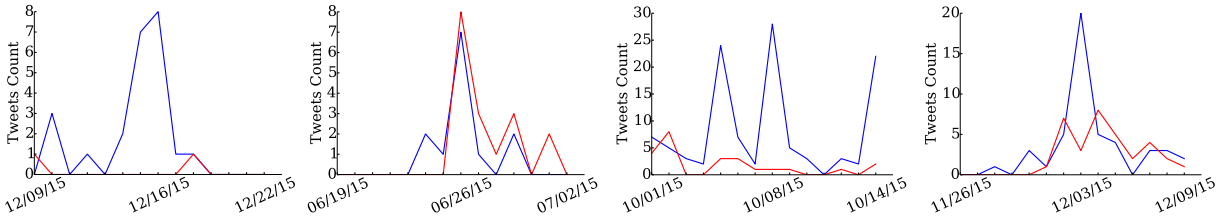
a ACA b Abortion

Figure 3: Temporal Twitter Activity by Party. The red and blue lines represent the temporal overlaps, or lack thereof, of Republicans and Democrats (respectively) in Twitter activity 1 week before and after a major event.

10 of the issues and agreement accuracy for all issues. M2 further improves the stance and agreement predictions for an additional 8 and 12 issues, respectively. M3, the combination of the previous models with Twitter behavioral features, increases the stance prediction accuracy of M2 for 9 issues and the agreement accuracy for 12 issues.

The final agreement predictions of M3 are notably improved over the initial LB for all issues, indicating that similarities and differences in Twitter behaviors help capture agreement and disagreement patterns among politicians. The final stance predictions of M3 are improved over all issues except Guns, Iran, and TPP. For Guns, the stance prediction remains the same throughout all models, meaning party information does not boost the initial predictions determined from Twitter based behaviors. For Iran, the addition of M1 and M2 lower the accuracy, but the temporal features from M3 are able to restore it to the original prediction. For TPP, this trend is likely due to the fact that all models incorporate party information and the issue of TPP is the most heavily divided within and across parties, with 8 Republicans and 4 Democrats in support of TPP and 8 Republicans and 12 Democrats opposed. Even in cases where M1 and/or M2 remained steady or lowered the initial baseline result (e.g. stance for Religion and Pay), the final prediction by M3 is still higher than that of the baseline.

Overall Results: Table 8 presents our overall results for stance and agreement prediction in terms of accuracy. The Global score is the overall average for all politicians, while Rep and Dem consider only Republicans or Democrats, respectively. Each model increases the accuracy of the previous model's prediction, showing that additional Twitter behavioral features can help overcome the strong party line biases captured by M1.



(a) Environment (b) Marriage (c) Guns (d) Guns

Figure 4: Temporal Twitter Activity by Party for Three Issues.

6.3 Effects of Framing and Temporal Activity Patterns

As shown in Table 7, performance for *some* issues does not improve in M3. Upon investigation, we found that for all issues, except Abortion which improves in agreement, one or both of the top frames for the party are shared across party lines. For example, for ACA both Republicans and Democrats have the *Economic* and *Health and Safety* frames as their top two frames. For TPP, both parties share the *Economic* frame. In addition to similar framing overlap, the Twitter timeline for ACA also exhibits overlap, as shown in Figure 3(a). This figure highlights one week before and after the Supreme Court ruling to uphold the ACA. The peak of Twitter activity is the day of the ruling, 6/25/2015.

Conversely, Abortion, which shares no frames between parties (Democrats frame Abortion with *Constitutionality* and *Health and Safety* frames; Republicans use *Economic* and *Capacity and Resources* frames), exhibits a timeline with greater fluctuation. The peak of Figure 3(b) is 8/3/2015, which is the day that the budget was passed to include funding for Planned Parenthood. Despite sharing a peak, both parties have different patterns over this time frame, allowing M3 to extract enough information to increase agreement prediction accuracy by 1.61%.

Figure 4(a) shows an example of one event for the Environment issue: when the mayor of Flint, Michigan declared a state of emergency over lead in the city's water supply. Due to different temporal patterns and frames for such events, the Environment accuracy improves across all models, as shown in Table 7. Similarly, Figure 4(b) shows the week before and after the Supreme Court ruled to uphold the legality of same-sex marriage. The two central peaks are shared by both parties, but each party also has one peak before (Democrats) or after (Republicans) the event. Additionally, both parties share the *Constitutionality* frame as their top frame, but the second most used frame is *Morality* for Republicans and *Fairness and Equality* for Democrats. These slight differences allow the M3 model to improve over the M2 prediction. Finally, Figure 4(c) shows the week before and after Democratic Senators pushed for gun control legislation after the Umpqua Community College shooting and Figure 4(d) shows tweets around the San Bernadino shooting. For these events, both parties exhibit different timeline patterns and frames. Consequently, these behavioral features dominate the stance prediction and allow agreement accuracy to reach 99.59%.

7 Conclusion

In this work we present a framework for modeling the dynamic nature of political discourse on Twitter. Though we focus on a small set of politicians and issues, our approach can be modified to handle additional politicians or issues, as well as those of other countries, by incorporating the proper domain knowledge (e.g., replacing party with voting history, using new keywords for different issues in other countries, or changing events such as Supreme Court rulings to Parliament votes), which we leave as future work. Contrary to previous works, which typically focus on a single aspect of this complex microblogging behavior, we build a holistic model connecting party line biases, temporal behaviors, and issue framing into a single predictive model which identifies fine-grained stances and agreement patterns. Despite having no direct supervision and using only intuitive local classifiers to bootstrap our global model, our approach results in a strong predictive model which helps shed light on political discourse within and across party lines.

Acknowledgments

We thank the anonymous reviewers for their thoughtful comments and suggestions.

References

Rob Abbott, Marilyn Walker, Pranav Anand, Jean E. Fox Tree, Robeson Bowmani, and Joseph King. 2011. How can you say such things?!?: Recognizing disagreement in informal political argument. In *Proc. of the Workshop on Language in Social Media*.

Amjad Abu-Jbara, Ben King, Mona Diab, and Dragomir Radev. 2013. Identifying opinion subgroups in arabic online discussions. In *Proc. of ACL*.

Stephen H. Bach, Bert Huang, Ben London, and Lise Getoor. 2013. Hinge-loss Markov random fields: Convex inference for structured prediction. In *Proc. of UAI*.

Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2015. Hinge-loss markov random fields and probabilistic soft logic. *arXiv preprint arXiv:1505.04406*.

Akshat Bakliwal, Jennifer Foster, Jennifer van der Puil, Ron O'Brien, Lamia Tounsi, and Mark Hughes. 2013. Sentiment analysis of political tweets: Towards an accurate classifier. In *Proc. of ACL*.

David Bamman and Noah A Smith. 2015. Open extraction of fine-grained political statements. In *Proc. of EMNLP*.

Eric Baumer, Elisha Elovic, Ying Qin, Francesca Polletta, and Geri Gay. 2015. Testing and comparing computational approaches for identifying the language of framing in political news. In *In Proc. of ACL*.

Adam Bermingham and Alan F Smeaton. 2011. On using twitter to monitor political sentiment and predict election results.

Amber Boydstun, Dallas Card, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2014. Tracking the development of media frames within and across policy issues.

Dallas Card, Amber E. Boydstun, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2015. The media frames corpus: Annotations of frames across issues. In *Proc. of ACL*.

Michael D Conover, Bruno Gonçalves, Jacob Ratkiewicz, Alessandro Flammini, and Filippo Menczer. 2011. Predicting the political alignment of twitter users. In *Proc. of Privacy, Security, Risk and Trust (PASSAT) and SocialCom*.

Sarah Djemili, Julien Longhi, Claudia Marinica, Dimitris Kotzinos, and Georges-Elia Sarfati. 2014. What does twitter have to say about ideology? In *NLP 4 CMC: Natural Language Processing for Computer-Mediated Communication*.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proc. of NAACL*.

Sean Gerrish and David M Blei. 2012. How they vote: Issue-adjusted models of legislative behavior. In *Advances in Neural Information Processing Systems*, pages 2753–2761.

Stephan Greene and Philip Resnik. 2009. More than words: Syntactic packaging and implicit sentiment. In *Proc. of NAACL*.

Justin Grimmer. 2010. A bayesian hierarchical topic model for political texts: Measuring expressed agendas in senate press releases. In *Political Analysis*.

Kazi Saidul Hasan and Vincent Ng. 2014. Why are you taking this stance? identifying and classifying reasons in ideological debates. In *Proc. of EMNLP*.

Bert Huang, Stephen H. Bach, Eric Norris, Jay Pujara, and Lise Getoor. 2012. Social group modeling with probabilistic soft logic. In *NIPS Workshops*.

Mohit Iyyer, Peter Enns, Jordan L Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proc. of ACL*.

Yoad Lewenberg, Yoram Bachrach, Lucas Bordeaux, and Pushmeet Kohli. 2016. Political dimensionality estimation using a probabilistic graphical model. In *Proc. of UAI*.

Jiwei Li, Alan Ritter, Claire Cardie, and Eduard H Hovy. 2014a. Major life event extraction from twitter based on congratulations/condolences speech acts. In *Proc. of EMNLP*.

Jiwei Li, Alan Ritter, and Eduard H Hovy. 2014b. Weakly supervised user profile extraction from twitter. In *Proc. of ACL*.

Axel Maireder and Julian Ausserhofer. 2013. National politics on twitter: Structures and topics of a networked public sphere. In *Information, Communication, and Society*.

Micol Marchetti-Bowick and Nathanael Chambers. 2012. Learning for microblogs with distant supervision: Political forecasting with twitter. In *Proc. of EACL*.

Yelena Mejova, Padmini Srinivasan, and Bob Boynton. 2013. Gop primary season on twitter: popular political sentiment in social media. In *WSDM*.

Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, and Kristina Miler. 2015. Tea party in the house: A hierarchical ideal point topic model and its application to republican legislators in the 112th congress. In *Proc. of ACL*.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proc. of ICWSM*.

Ferran Pla and Lluís F Hurtado. 2014. Political tendency identification in twitter using sentiment analysis techniques. In *Proc. of COLING*.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proc. of ACL*.

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

SemEval. 2016. Task 6. `http://alt.qcri.org/semeval2016/task6/`.

Yanchuan Sim, Brice DL Acree, Justin H Gross, and Noah A Smith. 2013. Measuring ideological proportions in political speeches. In *Proc. of EMNLP*.

Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing stances in online debates. In *Proc. of ACL*.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proc. of NAACL Workshops*.

Dhanya Sridhar, James Foulds, Bert Huang, Lise Getoor, and Marilyn Walker. 2015. Joint models of disagreement and stance in online debate. In *Proc. of ACL*.

Oren Tsur, Dan Calacci, and David Lazer. 2015. A frame of mind: Using statistical models for detection of framing and agenda setting campaigns. In *Proc. of ACL*.

Andranik Tumasjan, Timm Oliver Sprenger, Philipp G Sandner, and Isabell M Welpe. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment.

Svitlana Volkova, Glen Coppersmith, and Benjamin Van Durme. 2014. Inferring user political preferences from streaming communications. In *Proc. of ACL*.

Svitlana Volkova, Yoram Bachrach, Michael Armstrong, and Vijay Sharma. 2015. Inferring latent user properties from texts published in social media. In *Proc. of AAAI*.

Marilyn A. Walker, Pranav Anand, Robert Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Proc. of NAACL*.

Robert West, Hristo S Paskov, Jure Leskovec, and Christopher Potts. 2014. Exploiting social network structure for person-to-person sentiment analysis. *TACL*.

Janyce Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Computational linguistics*.

Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Opinionfinder: A system for subjectivity analysis. In *In Proc. of EMNLP*.

Leveraging Multiple Domains for Sentiment Classification

Fan Yang **Arjun Mukherjee** **Yifan Zhang**
Department of Computer Science, University of Houston, TX, USA
fyang11@uh.edu arjun4787@gmail.com aeryen@gmail.com

Abstract

Sentiment classification becomes more and more important with the rapid growth of user-generated content. However, sentiment classification task usually comes with two challenges: first, sentiment classification is highly domain-dependent and training sentiment classifier for every domain is inefficient and often impractical; second, since the quantity of labeled data is important for assessing the quality of classifier, it is hard to evaluate classifiers when labeled data is limited for certain domains. To address the challenges mentioned above, we focus on learning high-level features that are able to generalize across domains, so a global classifier can benefit with a simple combination of documents from multiple domains. In this paper, the proposed model incorporates both labeled and unlabeled data from multiple domains and learns new feature representations. Our model doesn't require labels from every domain, which means the learned feature representation can be generalized for sentiment domain adaptation. In addition, the learned feature representation can be used as classifier since our model defines the meaning of feature value and arranges high-level features in a prefixed order, so it is not necessary to train another classifier on top of the new features. Empirical evaluations demonstrate our model outperforms baselines and yields competitive results to other state-of-the-art works on the benchmark dataset.

1 Introduction

With the rapid growth of user-generated content, such as product reviews and microblogs, sentiment analysis and opinion mining have become more and more important as they address the problem of analyzing user's opinions, emotions, sentiments and attitudes. The applications of sentiment analysis have been found in almost every business and social domain (Liu, 2012; Bollen et al., 2011; Ku et al., 2006). Document-level sentiment classification predicts sentiment polarities for a given document or review. The large number of reviews not only help customers make better decisions but also make it possible yet challenge for product manufacturers to keep track opinions of the products (Hu and Liu, 2004).

While machine learning techniques provide interesting methods for analyzing sentiments (Turney, 2002; Go et al., 2009; Pang et al., 2002), challenges also arise certain limitations for the development of sentiment classification. For example, sentiment expression is highly domain-dependent (Pang and Lee, 2008), but training sentiment classifier for every domain is inefficient and often impractical. Simply combining data from different domains may not contribute to a generalized classifier for every domain, as users express the same sentiment in different domains using different words, or even express different sentiment using same words. For example, a user would prefer computer or car to "run fast" but not wish to use battery "die fast" or to see watch "move fast". A book or a movie can attract people by "unpredictable" endings but "unpredictable" economic trends scare away investors. In addition, certain domains don't have enough labeled data for building the classifier, which makes it indispensable to transfer knowledge between different domains.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2978–2988, Osaka, Japan, December 11-17 2016.

A common observation is, even though sentiment expression is domain-dependent and various words are isolated by domain categories, there are always domain-independent words expressing general sentiment polarities. In traditional sentiment domain adaptation that focuses on one domain to another domain (Daumé III et al., 2010; Ben-David et al., 2007; Ben-David et al., 2010), such words are usually defined as pivot features (Blitzer et al., 2006; Blitzer et al., 2007; Pan et al., 2010). Existing works have focused on generating new feature representations for pivot features (Glorot et al., 2011; Chen et al., 2012; Yang and Eisenstein, 2015; Bollegala et al., 2015) so that classifiers trained on new features can generalize well across domains.

In this paper, we follow the motivation of using new feature representations (Bengio et al., 2012) to bridge domain divergence and transfer knowledge among domains. The idea of proposed work is to learn a high-level feature space where three constraints are enforced: the model can incorporate multiple domains with both labeled and unlabeled data; the high-level feature space maximizes the margin between sentiment polarities; the high-level feature can represent original features well so that two feature space can be transformed to each other through a shared parametric matrix. Some of the key characters of the proposed model are:

1. Given multiple domains, our model can leverage sentiment similarity between instances across different domains regardless of the dissimilarity between domains. This is achieved by maximizing the distance between sentiments and minimizing the distance between domains in the high-level feature space.

2. Compared with one domain(source) to another domain(target) schema, our model collaborates all possible domains with both labeled and unlabeled data, which is a more generic framework and caters for better transfer across domains.

3. Our model directly maximizes the margin of sentiment polarities in the learned feature space. This is achieved by exploiting non-linear transformation with sigmoid function and aligning instances to pseudo-sentiment centroids.

4. Unlike traditional representation learning method which involves two stages: learning representation and building classifier, the new feature space learned by our model can be taken as classifier by itself. This is achieved through setting the order of learned high-level features and defining the meaning of feature values. As a result, it is not necessary to train another classifier on top of the new features.

5. We extend autoencoder (Vincent et al., 2010) by incorporating sentiment polarities. Unlike existing semi-supervised autoencoder (Liu et al., 2015; Socher et al., 2011) that needs another layer for labels, our model introduces pseudo-sentiment centroids, which can be prefixed and selected without fine-tuning.

2 Related Work

In this section, we review related works on sentiment analysis and transfer learning.

2.1 In-Domain Sentiment Analysis

For sentiment analysis of user generated content, traditional works have focused on textual content and dictionaries based approaches (Taboada et al., 2011; Hu and Liu, 2004; Pang and Lee, 2008). Pang et al. (2002) built sentiment classifier to predict sentiment polarities of movie reviews. Hu et al. (2013) exploited contextual emotional signals for effective sentiment analysis in an unsupervised manner. Tumasjan et al. (2010) evaluated and analyzed Twitter messages with the political sentiment to predict the popularity of parties. Bollen et al. (2011) explored how Twitter mood patterns can identify economic events. Other trends of sentiment analysis are based on visual content (Siersdorfer et al., 2010; Borth et al., 2013) and multi-modalities (Socher et al., 2011). All these works consider training and testing data are within the same domain or following similar distributions.

Even though our work only focuses on textual content, we omit explicit word to word analysis but project word features into high-level feature space, where visual content or multi-modalities can also be transformed. In addition, we examine the limitation of domain-dependent sentiment expression and investigate efforts on building a generalized representation for all domains.

2.2 Knowledge Transfer and Leveraging Multiple Domains

Knowledge-based sentiment analysis have been explored in (Mukherjee and Liu, 2012; Liu, 2014; Chen et al., 2013a; Chen et al., 2013b; Chen et al., 2013c). However, they have focused on aspect term extraction as opposed to sentiment polarity extraction which is the focus of this work. Another thread is to transfer sentiment knowledge across domains. It is usually defined as domain adaptation (Daume III and Marcu, 2006; Ben-David et al., 2010; Ben-David et al., 2007). It utilizes the knowledge learned from one domain, referred as the source domain, to solve tasks in another domain, referred as the target domain. Studies have focused on re-weighting features that cross domains (Jiang and Zhai, 2007; Xia et al., 2013), using feature embeddings to convert word feature to vector feature (Yang and Eisenstein, 2015; Bollegala et al., 2015) or generating new feature representations that align domain-specified features onto a generalized feature space which can bridge domain divergence (Blitzer et al., 2006; Cheng and Pan, 2014). Blitzer et al. (2007) proposed Structural Correspondence Learning by selecting pivot feature and creating correlations between the pivot and non-pivot features. Pan et al. (2010) introduced a bipartite graph based approach to connect domain-independent features and domain-specific features. Xiao et al. (2013) proposed supervised word clustering, which assumed that a document was composed of latent (topical) clusters and used expectation-maximization algorithm to find those clusters to transform documents from bag-of-words representation to clusters representation.

Besides transferring knowledge from one domain to another domain, researchers have also explored the area of leveraging multiple domains (Mansour et al., 2009; Duan et al., 2009; Daumé III et al., 2010). In addition, Gong et al. (2012) introduced a kernel metric identifying the optimal adaptability among different domains. Li and Zong (2008) leveraged domains by combining sentiment classifiers of different domains to make the final prediction. Wu and Huang (2015) collaborated multiple domains by exploring textual content relations and sentiment word relations via labeled data. Glorot et al. (2011) utilized unlabeled data through unsupervised deep learning approach with rectifier. Chen et al. (2012) extended linear autoencoder by learning with marginalized corrupted features. Liu et al. (2015) incorporated domain and sentiment supervision for sentiment classification cross domains.

While the above works have made important progress, there are some major differences from this proposed work. First, instead of considering the transfer from "source" to "target", our model leverages multiple domains and at the time is capable of learning from both labeled and unlabeled data across multiple domains. This is closer to the reality because the amount of labels are various among domains and we want the model to leverage all possible knowledge. Second, our model exploits non-linear transformation with the sigmoid function. The sigmoid function shrinks feature value within $(0, 1)$, which enable us to directly maximize the margin of sentiment polarities by supervised instances alignment. Moreover, we fix the order of high-level features and define the meaning of feature values, so classification can be acquired by the learned representation. This is more efficient in terms of performance as it does not require retraining another classifier as other methods do.

3 The Proposed Model

The general idea of the proposed work is to learn a high-level feature space for multi-domain sentiment classification with three constraints. First, the collaboration constraint, which allows the model to collaborate multiple domains with both labeled and unlabeled data. Second, the max-min constraint, that employs high-level feature space to maximize the margin between sentiment polarities of instances across different domains and minimize the distance between domain clusters. Third, the transformation constraint where the high-level feature space and original feature space are transformed to each other through a shared weight matrix, which reduces overfitting.

3.1 Notations

Given a binary sentiment classification problem with positive and negative labels, we have access to P domains. There are total N documents, M of which are labeled. So for each domain j, documents and labels are denoted as $\{\mathbf{X}^j \in \mathbb{R}^{N_j \times D}, \mathbf{y}^j \in \mathbb{R}^{M_j \times 1}\}$, where D is the dimensions of the original feature space. We assume the original feature space is shared across different domains. $\mathbf{x}_i^j \in \mathbb{R}^{1 \times D}$ is the i^{th} document in domain j and represented as a Boolean vector of bag-of-words. If the i^{th} document is labeled, then $y_i^j \in \{+1, -1\}$. The shared weight matrix is denoted as $\mathbf{W}$. The bias vectors for transformation between original space and high-level feature space are denoted as $\mathbf{b}_1, \mathbf{b}_2$, separately.

3.2 The Model with Three Constraints

3.2.1 The Collaboration Constraint

The collaboration constraint enforces the model to collaborate multiple domains with all possible data. We adopt one layer denoising autoencoder (Vincent et al., 2010) to encourage this constraint and treat all data as unlabeled at this point. A denoising autoencoder learns high-level feature space $h(\mathbf{x})$. It corrupts input $\mathbf{x}$ and feeds the corrupted version $\bar{\mathbf{x}}$ into the encoding layer. The decoder undoes the corruption by generating results back to uncorrupted $\mathbf{x}$. The parameters of denoising autoencoder are learned by minimizing the reconstruction loss $\mathcal{L}_r\big(\mathbf{x}, g(\bar{\mathbf{x}})\big)$, where

$$g(\bar{\mathbf{x}}) = s_2(\mathbf{w}_2 h(\bar{\mathbf{x}}) + \mathbf{b}_2) \quad \text{and} \quad h(\bar{\mathbf{x}}) = s_1(\mathbf{w}_1 \bar{\mathbf{x}} + \mathbf{b}_1) \tag{1}$$

We utilize masking noise for the corruption and implement component wise logistic sigmoid as the non-linear function for both $s_1(\mathbf{x})$ and $s_2(\mathbf{x})$. It is important to have the value of each high-level feature of $h(\mathbf{x})$ between $\big(0, 1\big)$, as it paves the way for the following steps and makes the learned representation advisable for different classifiers. Note that the focus of this work is sentiment classification and we only utilize a single layer autoencoder, so the stack version of our implementation and the issue of handling the vanishing gradient are not discussed.

3.2.2 The Max-Min Constraint

The max-min constraint utilizes labeled data and supports the new representation $h(\mathbf{x})$ to maximize the margin of sentiment polarities. Sentiment classification is highly domain-dependent, which implies domain clusters are easier separated than sentiment clusters in the original feature space. While in $h(\mathbf{x})$, instances are aligned to prefixed sentiment pseudo-centroids if they have same sentiment polarities. Since the value of each high-level feature is between $\big(0, 1\big)$, we prefix $[1, \cdots, 1, 0, \cdots, 0]^T$ for positive pseudo-centroid $\mathbf{c}^+$ and $[0, \cdots, 0, 1, \cdots, 1]^T$ for negative pseudo-centroid $\mathbf{c}^-$, so the pseudo-centroids are maximized by cosine distance. This prefix also allows the learned representation to be used as classifier. The dimension of pseudo-centroid $|\mathbf{C}|$ is same as the dimension of learned feature space $|h(\mathbf{x})|$. After the alignment, the true sentiment centroids of labeled data would also be maximized. The alignment is achieved by minimizing the alignment loss $\mathcal{L}_a\big(\mathbf{C}, h(\mathbf{x})\big)$, where

$$\mathbf{C} = \begin{cases} \mathbf{c}^+, \text{if the label of } \mathbf{x} \text{ is positive} \\ \mathbf{c}^-, \text{if the label of } \mathbf{x} \text{ is negative} \end{cases} \tag{2}$$

When maximizing sentiment polarities between instances, the distance between domains is also minimized as positive instances across domains are moving towards $\mathbf{c}^+$ and negative instances are moving towards $\mathbf{c}^-$. This alignment can suppress domain-specific features because domains would be hardly partitioned.

3.2.3 The Transformation Constraint

We share the weight matrix between encoding layer and decoding layer in equation 1, so the equations are updated to:

$$g(\bar{\mathbf{x}}) = sigmoid(\mathbf{W}^T h(\bar{\mathbf{x}}) + \mathbf{b}_2) \quad \text{and} \quad h(\bar{\mathbf{x}}) = sigmoid(\mathbf{W}\bar{\mathbf{x}} + \mathbf{b}_1) \tag{3}$$

By sharing the weight matrix, the transformation would be more robust and provide a better feature representation as it reduces overfitting. The shared weights can also be interpreted as a trade-off between suppressing and preserving domain-specific features. We cannot fully eliminate those features, as domain-specific features give the ability to reconstruct to the original feature space.

3.3 Loss Function and Optimization

We use sum of Bernoulli Cross Entropy for reconstruction loss $\mathcal{L}_r\big(\mathbf{x}, g(\bar{\mathbf{x}})\big)$ and alignment loss $\mathcal{L}_a\big(\mathbf{C}, h(\mathbf{x})\big)$. Combined with equation 2 and 3, the final loss function $\mathcal{L}$ is

$$\begin{aligned}\mathcal{L} =& \mathcal{L}_r\big(\mathbf{x}, g(\bar{\mathbf{x}})\big) + \alpha\mathcal{L}_s\big(\mathbf{C}, h(\mathbf{x})\big) \\ =& -\frac{1}{N}\sum_{i}^{N}\sum_{k}^{|\mathbf{x}|}\mathbf{x}_{i,k}\log g(\bar{\mathbf{x}}_i)_k + \big(1-\mathbf{x}_{i,k}\big)\log\big(1-g(\bar{\mathbf{x}}_i)_k\big) \\ & -\alpha\frac{1}{M}\sum_{j}^{M}\sum_{l}^{|\mathbf{C}|}\mathbf{C}_l\log h(\mathbf{x}_j)_l + \big(1-\mathbf{C}_l\big)\log\big(1-h(\mathbf{x}_j)_l\big)\end{aligned} \tag{4}$$

We use $\alpha = 1$ in the model[1]. The equation 4 is non-convex but can be optimized with gradient descend. The partial derivatives of $\mathcal{L}$ with respect to $\mathbf{b}_1, \mathbf{b}_2$ and $\mathbf{W}$ is

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}} = \frac{1}{N}\sum_{i}^{N}(h(\bar{\mathbf{x}}_i) + h(\bar{\mathbf{x}}_i)(1-h(\bar{\mathbf{x}}_i))^T\mathbf{W}\bar{\mathbf{x}}_i\big) \times \big(g(\bar{\mathbf{x}}_i) - \mathbf{x}_i\big)^T + \frac{\alpha}{M}\sum_{j}^{M}(h(\mathbf{x}_j) - \mathbf{C})\mathbf{x}_j^T \tag{5}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{b}_1} = \frac{1}{N}\sum_{i}^{N}h(\bar{\mathbf{x}}_i)(1-h(\bar{\mathbf{x}}_i))^T\mathbf{W}\big(g(\bar{\mathbf{x}}_i) - \mathbf{x}_i\big) + \frac{\alpha}{M}\sum_{j}^{M}(h(\mathbf{x}_j) - \mathbf{C}) \tag{6}$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{b}_2} = \frac{1}{N}\sum_{i}^{N}\big(g(\bar{\mathbf{x}}_i) - \mathbf{x}_i\big) \tag{7}$$

The output of our model is the learned feature representation $h(\mathbf{x})$. This representation can be used as features for another classifier or can be used as a classifier by itself.

3.4 Using the Representation as Classifier

To use $h(\mathbf{x})$ as a classifier, we calculate the distance from data points to the prefixed pseudo-centroids $\mathbf{C}$. Data points closer to $\mathbf{c}^+$ should have first half of $h(\mathbf{x})$ closer to 1 and the second half closer to 0, while data points closer to $\mathbf{c}^-$ should have first half of $h(\mathbf{x})$ closer to 0 and second half closer to 1. Therefore, the predicted label of a data point is:

$$sgn\big(\sum_{i=0}^{|\mathbf{C}|/2-1}h(\mathbf{x})_i - \sum_{j=0}^{|\mathbf{C}|/2-1}h(\mathbf{x})_{j+\frac{|\mathbf{C}|}{2}}\big) \tag{8}$$

The classifier can also be interpreted as using the decision of multiple logistic regression models. For the first half of $h(\mathbf{x})$, we define values greater than 0.5 represent positive and smaller than 0.5 represent negative, while for the second half, we define values greater than 0.5 represent negative and smaller than 0.5 represent positive.

4 Experimental Evaluations

We first report results of multi-domain sentiment classification, comparing different methods with in-domain classifier and multi-domain classifier. Then, we extend our model to domain adaptation problem with multiple source domains and one target domain. Finally, we evaluate the model with different metrics for a better understanding. All SVM classifiers are implemented through linear LibSVM (Chang and Lin, 2011) without tuning other parameters.

[1]We have tested the impact of different values of α from 0.2 to 10 and found the value of α does not significantly affect the performance of our model. A larger α generally gives a slightly better result.

4.1 Dataset

We use the Amazon product reviews (Blitzer et al., 2007) as our experimental dataset. The dataset has been widely used in multi-domain sentiment classification and domain adaptation for sentiment classification. There are 22 domains and more than 300,000 reviews in this dataset. We conduct experiments on reviews of 4 domains: **B**ooks, **D**vd, **E**lectronics and **K**itchen. Each of selected domain has 1000 positive and 1000 negative labeled reviews and roughly 5000 unlabeled reviews. The top 5000 of frequent 1-gram and 2-gram features are selected, as low frequent features are usually related to domains. The experiments are conducted based on 5-folder cross-validation by randomly splitting the labeled data into 5 partitions with equal size and we report the average result[2].

4.2 Performance Evaluation

We report results of multi-domain sentiment classification. The compared methods are listed below:

SVM-ID, SVM-MD: SVM classifier used for in-domain(ID) sentiment classification, and multi-domain(MD) sentiment classification with document-level combination.

ClassifierFusion (Li and Zong, 2008): Multi-domain sentiment classification with classifier-level combination. For each domain, a classifier is trained and used for all domains. The final prediction is the combination of the predictions of each individual classifier.

T-SVM (Sindhwani and Keerthi, 2006): Transductive SVM with document-level combination. All unlabeled data are used during learning. This method explicitly shows the performance of introducing unlabeled data to an SVM classifier, so it can be interpreted as SVM-MD with both labeled and unlabeled data.

SDA (Glorot et al., 2011): Unsupervised denoising autoencoder for representation learning. The feature space for the final SVM classifier is the concatenation of the original feature space and the learned representation feature space.

SDA-DSS (Liu et al., 2015): Representation learning with domain and sentiment supervision. The original implementation only incorporated sentiment labels of one domain, so we extend the model to incorporate labeled data of 4 domains. Same as SDA, the feature space for the SVM classifier is the concatenation of the original feature space and the learned representation feature space.

Proposed-R: Using representation learned from the proposed model. The feature space of the final SVM classifier is only the learned representation feature space. The hyper-parameter are explored as follow and selected by cross-validation: a masking noise probability in $\{0,0.5,0.6,0.7,0.8\}$ for corrupted $\bar{\mathbf{x}}$; dimension of learned feature space $h(\mathbf{x})$ in $\{100,250,500\}$; L_2 regularization penalty on shared weight matrix $\mathbf{W}$ in $\{0, 10^{-4}, 10^{-3}, 10^{-1}, 1\}$; learning rate for gradient descent in $\{0.01, 0.03, 0.1, 0.3\}$. The implementation is through Theano (Bastien et al., 2012).

Proposed-C: The learned representation is used as classifier with Equation 8. Because we only focus on the learned representation without tuning the parameter of SVM, the result of Proposed-C is only presented for a reference, not to demonstrate it is better than other classifiers.

All models, except SVM-ID, utilize the training set of 4 domains together and then make prediction on the test data of each domain. All models, except SVM-ID, SVM-MD, and ClassifierFusion, are implemented through transductive inference to better leverage the unlabeled data. However, it could be easily extended to inductive inference as all models return the feature transformation matrix.

	SVM-ID (%)		SVM-MD (%)		ClassifierFusion (%)		T-SVM (%)		SDA (%)		SDA-DSS (%)		Proposed-R (%)		Proposed-C (%)	
	Ac	F1	Ac	F1	Ac	F1	Ac	F1	Ac	F1	Ac	F1	Ac	F1	Ac	F1
B	77.81	78.25	78.03	78.26	79.16	79.28	77.50	76.67	80.49	80.70	79.60	79.84	**81.57**	**81.84**	84.14	84.56
D	77.61	78.05	79.14	79.13	81.37	81.45	79.67	79.53	80.73	80.93	80.23	80.25	**83.41**	**83.69**	84.91	85.44
E	82.54	82.68	83.52	83.55	84.77	84.03	86.17	86.05	84.65	84.64	84.21	84.16	**87.01**	**87.00**	88.17	88.18
K	84.78	84.56	84.39	84.26	86.42	86.05	86.17	85.33	86.18	85.97	85.53	85.16	**88.07**	**88.07**	89.30	89.40
Ave.	80.68	80.89	81.27	81.33	82.96	82.70	82.38	81.90	83.01	83.06	82.39	82.35	**85.02**	**85.15**	86.63	86.90

Table 1: Performance on Accuracy and F1 for Multi-Domain Sentiment Classification Tasks

[2]Micro average F1 on positive class is reported in this paper

According to Table 1, we find that our method consistently outperforms all other competitors showing that leveraging multiple domains can provide better results for sentiment classification if dissimilarities between domains are taken care of. It also shows that arbitrarily combining data together with bag-of-word representation cannot guarantee better results compared to in-domain sentiment classification. Compared to ClassifierFusion, the results validate that unlabeled instances and transductive inference can improve sentiment classification. Compared to T-SVM, it can be concluded that learning a new representation would benefit a generalized sentiment classifier across domains. Compared to SDA and SDA-DSS, the improvements can be explained as credit of transforming with sigmoid function, maximizing margin of sentiment polarities in learned feature space and suppressing domain-specific features during representation learning.

4.3 Unsupervised Knowledge Transfer

When incorporating multi-domains with both labeled and unlabeled data, a possible scenario is certain domains have very limited or even zero labels. In the literature of domain adaptation, unsupervised knowledge transfer or unsupervised domain adaptation usually refers to the situation where certain domains have no labeled instance at all, so the unlabeled domain, usually denoted as target domain, has to borrow labels from other domains denoted as source domain (Daume III and Marcu, 2006). In this experiment, we check the ability of our model for unsupervised domain adaptation. We consider 3 source domains and 1 target domain, and remove labels of each target domain separately. Table 2 presents the results of training on 3 domains and testing on the other. More specifically, SVM-ID remains the same as in Table 1 and the result is interpreted as an upper-bound for unsupervised knowledge transfer, while all other models are adjusted to access the training set of 3 source domains and test on the test set of the target domain.

	SVM-ID (%)		SVM-MD (%)		ClassifierFusion (%)		T-SVM (%)		SDA (%)		SDA-DSS (%)		Proposed-R (%)		Proposed-C (%)	
	Ac	F1	Ac	F1	Ac	F1	Ac	F1	Ac	F1	Ac	F1	Ac	F1	Ac	F1
B	77.81	78.25	74.22	74.42	75.00	75.08	75.33	75.37	76.54	76.75	76.86	76.33	**78.07**	**78.15**	81.00	81.41
D	77.61	78.05	76.41	77.76	77.00	76.53	78.00	77.86	79.64	**79.86**	**78.78**	79.83	78.11	79.03	83.35	83.74
E	82.54	82.68	80.90	80.27	82.16	81.86	82.17	**82.48**	81.95	81.42	**82.40**	81.73	82.29	81.30	85.09	84.20
K	84.78	84.56	81.90	82.14	83.83	83.90	83.67	83.11	84.04	84.00	83.92	84.23	**84.37**	**84.79**	87.25	87.31
Ave.	80.68	80.89	78.36	78.65	79.50	79.34	79.79	79.71	80.54	80.26	80.49	80.53	**80.71**	**80.82**	84.17	84.16

Table 2: Performance on Accuracy and F1 for Unsupervised Domain Adaptation with Multiple Sources

Comparing with transfer from one source to one target (Glorot et al., 2011; Liu et al., 2015), we observer from Table 2 that arbitrarily combining data together decreases the best transfer performances of SDA and SDA-DSS, which suggests domain-specific features are hurting unsupervised transfer. Moreover, our model yields limited improvements this time. One reason could be SDA and SDA-DSS separate domain-dependent and domain-independent features and keep all features in the learned representation, while our model suppresses domain-dependent feature. However, in general, "domain-dependent" is a relative definition. A word "story" could be a domain-independent feature for Books and DVD but also could be a domain-dependent feature for Books and Kitchen. Therefore, suppressing domain-dependent features for multiple domains which works better in the previous task could be the reason that limits our model on this unsupervised domain adaptation task.

4.4 Performance on Additional Metrics

We assess our model with additional metrics for multi-domain sentiment classification: sensitivity of labeled instances, proxy-A-distance, performance on different classifiers and suppressed features.

Sensitivity of Labeled Instances In this experiment, we explore how the proportion of labels affects the model as the model collaborates both labeled and unlabeled data from multiple domains. We limit the accessible labels in $\{0, 100, 300, 500, 1000, 1500\}$ for each domain to learn the representation and repeat the experiment of multi-domain sentiment classification with Proposed-R. With 0 labels, the model would be similar to a fully unsupervised SDA (Glorot et al., 2011) implementation, except with sigmoid activation and a lower feature dimension. According to Figure 1, limiting labels decreases the

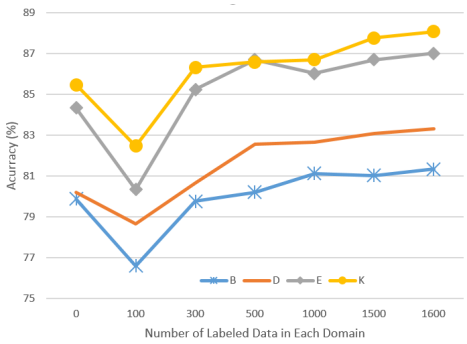


Figure 1: Sensitivity of Labeled Instances

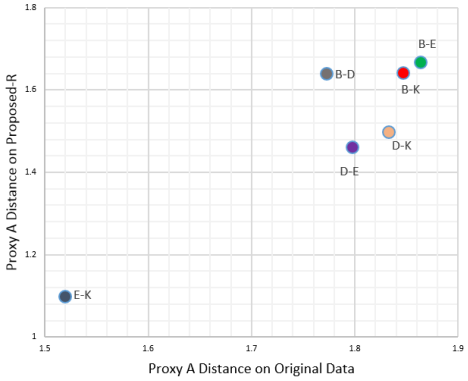


Figure 2: Proxy-A-Distance

performance, but this issue is generally solved as the proportion of labels is further increased and we see that at about 1000 labels performance starts to stabilize.

Proxy-A-Distance (PAD) We use PAD as an indirect metric to measure the ability to minimize dissimilarities between domains. The intuition is that removing domain-dependent features would weaken the discrimination among domains. The PAD metric (Ben-David et al., 2007) is defined as $2(1 - 2\epsilon)$, where ϵ is the generalization error and obtained by measuring how distinguishable are the two domains. In other words, we use the learned representation to accomplish domain recognition task. We randomly choose 1000 instances for training and 1000 for testing from each domain. Then we set up the recognition task as a binary classification problem with 6 combinations, for example, recognition between B and D. After applied our model, the PAD value of every recognition pair has decreased, which indicates the new feature representation learned from our model suppresses domain-dependent features.

According to Figure 2, recognizing Kitchen and Electronics are more difficult than Books and DVD. One reason is the reviews in Kitchen and Electronics are expressed using more domain-independent words. This observation also explains the sentiment classification results in Table 1 that classifying sentiment in Kitchen and Electronics generally have a better performance across different methods.

Performance of Different Classifiers We argue that a good representation should be able to benefit classification task without explicitly choosing or fine-tuning classifiers. Therefore, We repeat the experiment of multi-domain sentiment classification by comparing our Proposed-R(P-R) and SDA with another three state-of-art classifiers: K-Neighbors Classifier(KNC), Gaussian Naive Bayes(GNB) and RandomForest Classifier(RFC). According to Table 3, our model can still yield good results.

	KNC(%)		GNB(%)		RFC(%)	
	P-R	SDA	P-R	SDA	P-R	SDA
B	83.96	64.17	84.30	74.68	82.35	68.01
D	85.03	63.58	85.22	75.28	83.69	68.32
E	87.50	70.04	88.33	80.89	86.39	71.78
K	89.12	67.86	89.54	81.19	87.87	76.37

Table 3: Accuracy of Different Classifiers

Reconstructed	Original	Suppressed
not, poor, great, dont, was, after, bad, no, excellent, well, easy, best	was, my, so, num, you, all, one, if, very, great, good, just, not	since, way, all about, other, your time, them, when now, so, this_book

Table 4: Top Frequent Reconstruct Features, Original Features and Suppressed Features

Suppressed Features Our model has reconstructed 3785 features on average, compared to the original 5000, that means 1215 features are suppressed during the learning. We report some of the top frequent features in 3 ways: reconstructed by our model, in original space and suppressed by our model. From Table 4, the reconstructed feature are carrying more sentiment meaning than the original features, and the suppressed features involve domain-specific feature, such as "this_book", and non-sentiment features, such as "all", "so" and "your". This is what we expect by maximizing distance between sentiments in the learned representation feature space and sharing the transformation matrix between decoding and encoding layer.

5 Conclusion

This work proposed to leverage multiple domains with both labeled and unlabeled instances. The model learns high-level feature space with 3 constraints and achieves improved performance on multi-domain sentiment classification as attested by results on the benchmark dataset.

As our future work, we plan to explore multi-modality (e.g., mapping visual, video and sentiment content on the same space from multiple domains), and develop a recursive system where labeling work can be performed recursively with high confidence.

6 Acknowledgement

The authors would like to thank the anonymous reviewers for their comments. This work was supported in part by NSF 1527364.

References

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. 2012. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*.

Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. 2007. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175.

Yoshua Bengio, Aaron C Courville, and Pascal Vincent. 2012. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538*, 1.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics.

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447.

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. *arXiv preprint arXiv:1505.07184*.

Johan Bollen, Huina Mao, and Alberto Pepe. 2011. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *ICWSM*, 11:450–453.

Damian Borth, Tao Chen, Rongrong Ji, and Shih-Fu Chang. 2013. Sentibank: large-scale ontology and classifiers for detecting sentiment and emotions in visual content. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 459–460. ACM.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013a. Discovering coherent topics using general knowledge. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 209–218. ACM.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013b. Exploiting domain knowledge in aspect extraction. In *EMNLP*, pages 1655–1667.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013c. Leveraging multi-domain prior knowledge in topic models. In *IJCAI*.

Li Cheng and Sinno Jialin Pan. 2014. Semi-supervised domain adaptation on manifolds. *IEEE transactions on neural networks and learning systems*, 25(12):2240–2249.

Hal Daume III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

Hal Daumé III, Abhishek Kumar, and Avishek Saha. 2010. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59. Association for Computational Linguistics.

Lixin Duan, Ivor W Tsang, Dong Xu, and Tat-Seng Chua. 2009. Domain adaptation from multiple sources via auxiliary classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 289–296. ACM.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1:12.

Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. 2012. Geodesic flow kernel for unsupervised domain adaptation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2066–2073. IEEE.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. 2013. Unsupervised sentiment analysis with emotional signals. In *Proceedings of the 22nd international conference on World Wide Web*, pages 607–618. ACM.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *ACL*, volume 7, pages 264–271.

Lun-Wei Ku, Yu-Ting Liang, Hsin-Hsi Chen, et al. 2006. Opinion extraction, summarization and tracking in news and blog corpora. In *AAAI spring symposium: Computational approaches to analyzing weblogs*, volume 100107.

Shoushan Li and Chengqing Zong. 2008. Multi-domain adaptation for sentiment classification: Using multiple classifier combining methods. In *Natural Language Processing and Knowledge Engineering, 2008. NLP-KE'08. International Conference on*, pages 1–8. IEEE.

Biao Liu, Minlie Huang, Jiashen Sun, and Xuan Zhu. 2015. Incorporating domain and sentiment supervision in representation learning for domain adaptation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1277–1283. AAAI Press.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Zhiyuan Chen Arjun Mukherjee Bing Liu. 2014. Aspect extraction with automated prior knowledge learning.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2009. Domain adaptation with multiple sources. In *Advances in neural information processing systems*, pages 1041–1048.

Arjun Mukherjee and Bing Liu. 2012. Aspect extraction through semi-supervised modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 339–348. Association for Computational Linguistics.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760. ACM.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Stefan Siersdorfer, Enrico Minack, Fan Deng, and Jonathon Hare. 2010. Analyzing and predicting sentiment of images on the social web. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 715–718. ACM.

Vikas Sindhwani and S Sathiya Keerthi. 2006. Large scale semi-supervised linear svms. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 477–484. ACM.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Andranik Tumasjan, Timm Sprenger, Philipp Sandner, and Isabell Welpe. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408.

Fangzhao Wu and Yongfeng Huang. 2015. Collaborative multi-domain sentiment classification. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 459–468. IEEE.

Rui Xia, Xuelei Hu, Jianfeng Lu, Jian Yang, Chengqing Zong, et al. 2013. Instance selection and instance weighting for cross-domain sentiment classification via pu learning. In *IJCAI*.

Min Xiao, Feipeng Zhao, and Yuhong Guo. 2013. Learning latent word representations for domain adaptation using supervised word clustering. In *EMNLP*, pages 152–162.

Yi Yang and Jacob Eisenstein. 2015. Unsupervised multi-domain adaptation with feature embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 672–682, Denver, Colorado, May–June. Association for Computational Linguistics.

Political News Sentiment Analysis for Under-resourced Languages

Patrik F. Bakken, Terje A. Bratlie, Cristina Marco, Jon Atle Gulla
Norwegian University of Science and Technology
Trondheim, Norway
bakken.patrik@gmail.com, terje.a.bratlie@gmail.com,
{cristina.marco,jag}@ntnu.no

Abstract

This paper presents classification results for the analysis of sentiment in political news articles. The domain of political news is particularly challenging, as journalists are presumably objective, whilst at the same time opinions can be subtly expressed. To deal with this challenge, in this work we conduct a two-step classification model, distinguishing first subjective and second positive and negative sentiment texts. More specifically, we propose a shallow machine learning approach where only minimal features are needed to train the classifier, including sentiment-bearing Co-Occurring Terms (COTs) and negation words. This approach yields close to state-of-the-art results. Contrary to results in other domains, the use of negations as features does not have a positive impact in the evaluation results. This method is particularly suited for languages that suffer from a lack of resources, such as sentiment lexicons or parsers, and for those systems that need to function in real-time.

1 Introduction

In the rapidly changing World Wide Web, getting informed opinions about facts is becoming increasingly challenging for users. In online news, the same event can be presented from very different perspectives depending on the source. Journalism is supposed to be objective, yet opinions are also expressed in newswire text (Belyaeva and van Der Goot, 2008; Blaz et al., 2009). In this type of texts, opinions are generally expressed in a subtle way, by using language resources other than sentiment vocabulary, such as irony, sarcasm, metaphors, etc., as it is illustrated in the following example, where the use of an ironic comparison *som rundingsbøyer* 'as if they were human buoys' suggests a negative opinion:

> - Den nye forskriften betyr at i vannene der friluftsfolket fortsatt kunne få være i fred, i en liten bortgjemt vik, der er det nå åpent for skuterfolket å bruke turfolket som rundingsbøyer, sier Bjørn Hansen, på vegne av Naturvernforbundet i Finnmark. *(- Given by the new regulation, places near the waters where people still could have piece and quiet, such as small secluded coves, are now open for people on jet skis to make use of the former as if they were human buoys, says Bjrn Hansen, on behalf of the Nature Conservatory of Finnmark.)*[1]

In this context it is difficult for readers to have informed opinions about the events happening in the world. Thus there is a growing need for resources that can help readers filter out news information, so that they can have an informed opinion about the current events. These resources would be particularly useful in the domain of political news, where readers want to be informed about political parties, politicians, and policies. If successfully employing sentiment analysis in the political news domain, possible biases and opinions will be revealed, and readers will get a more complete and transparent news scenario to gain knowledge from, leading to more informed political opinions.

However, the unstructured nature of news articles together with the subtle ways of expressing opinions in these texts make this task particularly challenging for machines. To deal with these challenges, in this paper we propose an unsupervised machine learning approach that takes sentiment-bearing Co-Occurring

[1] https://www.nrk.no/finnmark/_-de-blir-a-bruke-turfolk-som-rundingsboyer-1.12519921

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2989–2996, Osaka, Japan, December 11-17 2016.

Terms (COTs) and negation words as features. We argue that this approach is suited for languages other than English where computational linguistics resources for sentiment analysis are scarce, and also for those systems that perform in real-time.

The contents of this paper are as follows. After introducing the state of the art in 2, in 3 we present the method used to deal with sentiment analysis in political news. Lastly, the results are evaluated in 4 and the paper closes with a discussion and conclusion in 5 and 6.

2 Related work

In sentiment analysis, classifiers have been trained to automatically detect the polarity and subjectivity of texts. The former takes into account that not all incoming text is opinionated, and that a system might have to distinguish between subjective and objective texts.

For example, (Yu and Hatzivassiloglou, 2003) focus on separating subjective texts from those that portray factual information. The latter assumes text to be opinionated, and thus classifies the text as falling in one out of two sentiment categories − in general, positive or negative (Pang and Lee, 2008). (Turney, 2002; Pang et al., 2002; Hu and Liu, 2004; Kim and Hovy, 2004, among others) focus on distinguishing an author's positive or negative opinion towards a certain topic or object. To perform these tasks, both supervised and unsupervised approaches have been used (Feldman, 2013).

A combination of these two classification tasks can be seen in (Wilson et al., 2005). Here, a two-step binary classification takes place, which firstly filters out neutral expressions, and secondly, classifies the polarity of the selected set of expressions. As (Mihalcea et al., 2007) suggest, improvements in the more challenging task of subjectivity detection might have a positive impact on polarity classification. By firstly increasing precision and recall in subjectivity detection, the performance of the second task in a live system will also be improved. In this paper we present work in this line, in that we perform a two-step classification task, where first subjective and second positive and negative texts are detected within those classified as subjective.

In both tasks, negation is often considered one of the most important training features. (Wilson et al., 2005) argue that a phrase's sentiment will be better understood if the contextual and prior polarity of the phrase is taken into account. In contrast, other research, such as (Kim and Hovy, 2004; Hu and Liu, 2004; Grefenstette et al., 2004), focuses on local negation - negation terms occurring within sentiment words. Negation is a very complex linguistic issue, with different semantic effects in the sentence depending on the scope of the negative term, see (Lasnik, 1972; Partee, 1992; Ladusaw, 1992, among many others). In order to determine the scope of the negative phrase, most computational approaches make use of a syntactic parser. However, for languages suffering from a lack of such resources, such as Norwegian, this strategy would be too expensive. Due to the lack of resources for the Norwegian languages, in the present paper we present a very simple method where only the presence of negative terms, such as English *not*, are considered.

Initial efforts for sentiment analysis in the newswire domain have been conducted by (Belyaeva and van Der Goot, 2008) and (Blaz et al., 2009). Similar work to that of ours has been completed for Turkish in the political news domain by (Kaya et al., 2012). In this work four supervised machine learning algorithms are evaluated, namely Naïve Bayes, Maximum Entropy, SVM and N-Gram character based Language Model, in which features are unigrams or single words pertaining to relevant morphosyntactic classes. Their approach yields a maximum accuracy of 76.78%. Our approach is different from the latter in that we focus on different machine learning algorithms, such as J48, known to be computationally faster. Besides, we employ a two-step binary classifier and analyze all paragraphs in a collection of newswire text, whereas they look at the overall sentiment of selected newspapers columns.

3 Method

Figure 1 shows a high-level overview of the two-step sentiment analysis approach presented in this paper. Firstly, the input dataset consists of paragraphs annotated with three different classes: positive, negative,

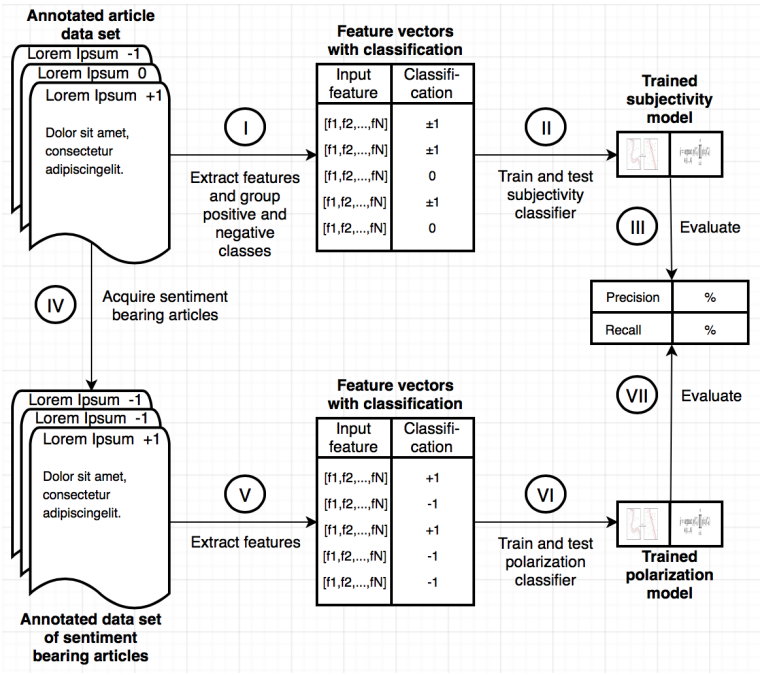


Figure 1: High-level system overview with two-step binary classification during testing and training.

or neutral. Features are then extracted from this dataset and used for training the subjectivity model. The precision of the resulting model is then evaluated using 10-fold crossvalidation. In the second phase, only the sentiment-bearing paragraphs from the dataset are used, and then the same procedure is followed for training and evaluation. We will now explain in further detail the different parts of our system.

3.1 Data and annotation

A total number of 3961 paragraphs within the political category was selected from the Norwegian online news sources *NRK*[2] and *VG*[3]. This dataset includes news articles over a span of four months during the summer of 2015, right before the municipal elections in October that year. Paragraphs were chosen instead of documents, sentences or phrases, because we noticed that in political news articles the presence of a sentiment target is more likely to be found at the paragraph level.

This dataset was annotated by the first two authors of this paper, obtaining an agreement of $\sim$76% (κ = 0.62), which is well within the range of other studies of sentiment analysis in similar domains (Njølstad et al., In press). The 3016 remaining paragraphs were used for training and testing using 10-fold crossvalidation. The paragraphs are labeled with one of the following three categories: positive, negative and neutral. The subjective class consists of both positive and negative paragraphs.

The criteria used for the authors to annotate the paragraphs were adapted from (Balahur and Steinberger, 2009), and are summarized in the following points: (i) world knowledge can be used if it is not clearly biased towards an entity; (ii) factual information should not be annotated as subjective; (iii) if polarity shifting occurs, the text should be annotated according to the one bearing the strongest sentiment and the most important entity; (iv) in case of uncertainty, the text should be annotated as objective.

3.2 Features

As summarized in Table 1, the features used to train the classifier are positive, negative and neutral Co-Occurring Terms (COTs), and negation words.

COTs are words with importance to each other that co-occur in a sentence (Pang and Lee, 2008; Matsuo and Ishizuka, 2004). In languages with a lack of computational resources to perform domain-

[2] https://www.nrk.no/
[3] http://www.vg.no/

Features	Description	Type
PositiveCots	Number positive COTs in a paragraph	Discrete
NeutralCots	Number of neutral COTs in a paragraph	Discrete
NegativeCots	Number of negative COTs in a paragraph	Discrete
Negations	Number of negation words in a paragraph	Discrete

Table 1: Input features for machine learning classifier.

specific sentiment analysis, the use of COTs as a sentiment lexicon has been shown to work effectively. Such domain-specific lexicon of sentiment-bearing COTs can be acquired with a reasonable amount of manual labour (Njølstad et al., In press).

The COTs used in the present work are two-word co-occurring terms, with a maximum of four words between, and belonging to one of the following morphosyntactic categories: adjectives, nouns, verbs and adverbs. The *term frequency - inverse document frequency* ranking function was used to limit the lexicon size to 4000 COTs. This ranking is important as we only want to include COTs that are relevant to our domain of investigation. The Oslo-Bergen-Tagger[4] for the Norwegian language was used to tokenize and part-of-speech tag paragraphs. Table 2 shows an excerpt from the lexicon obtained. Details on how this lexicon was acquired are outside the scope of this paper, but are described in detail in (Njølstad et al., In press).

Term 1	Term 2	Translation	Sentiment
ta	*ordet*	'take the word'	0
er	*allerede*	'is already'	0
stor	*glede*	'big happiness'	1
er	*misfornyd*	'is unsatisfied'	-1
skape	*arbeidsplasser*	'achieve job positions'	1
kan	*svekke*	'can weaken'	-1
skal	*i stedet*	'will instead'	0

Table 2: Excerpt from sentiment lexicon.

The second set of features used in this paper are negations. Negations are words that change the polarity of words, phrases or sentences and have been shown to work effectively to detect sentiment (Wilson et al., 2005). Table 3 shows the Norwegian negative words considered as features in this paper. As the dataset includes paragraphs written in both *bokmål* and *nynorsk* Norwegian, negation words from both varieties are included in this table.

Norwegian *bokmål*	Norwegian *nynorsk*	English
ikke	*ikkje*	'not'
ei	*ei*	'not'
nei	*nei*	'no'
aldri	*aldri*	'never'
neppe	*neppe*	'hardly'
ingen, inga, intet	*ingen, inga, inkje*	'none, any'

Table 3: Negation words in the Norwegian language.

We hypothesize that there will be more negations in negative sentiment-bearing paragraphs, which can positively contribute to classify a paragraph's sentiment. In order to observe this before obtaining the features, we analyzed the average number of negation words in each paragraph. As can be seen in

[4]http://www.tekstlab.uio.no/obt-ny/

Table 4, on average each paragraph annotated as negative has 0.53 negations, compared to only 0.23 per neutral paragraph. Besides, the subjective (positive+negative) class has 0.43 negations per paragraph, whereas the neutral class has only 0.23. These numbers suggest that negations can also be relevant to detect subjective paragraphs.

Annotated class	**Paragraphs**	**Negations per paragraph**
Positive	698	0.29
Neutral	1426	0.23
Negative	892	0.53
Positive+Negative	1590	0.43
All	3016	0.33

Table 4: Average number of negations per paragraphs.

4 Evaluation

The main goal of this paper was to experiment with simple feature combinations in a two-step binary classification process, in order to achieve state-of-the-art results within the domain of Norwegian political news. Tables 5 and 6 summarize the evaluation of this work. Three different classifiers were used: J48, Random Forest (RF), and Naïve Bayes (NB). These classifiers were selected because of their computational speed, as opposed to the more computational heavy algorithms such as SVM (Zhao and Zhang, 2008). In addition, current research has shown that J48 and RF yield higher precision results in the financial news domain (Njølstad et al., In press). All three classifiers in our system are set up using the WEKA framework.[5]

Table 5 presents the results of subjectivity classification. Highlighted in bold is the precision of the feature combination with the best result – 67.1%. This feature combination includes all three types of COTs. However, it does not include negations at all. We hypothesized that the difference in negations per paragraph between the neutral class and the subjective class could have a positive impact on the precision results for this step. Looking at these results, where there is a higher precision without the use of negations for both NB and J48, our hypothesis does not hold after all. RF is the only machine learning model which benefits from negations, though this model yielded unsatisfying results in general.

	PosCots	NeutCots	NegCots	Negations	Precision
NB	✓	✓	✓		**67.1**
	✓	✓	✓	✓	66.6
				✓	59.9
RF	✓	✓	✓		62.5
	✓	✓	✓	✓	63.8
				✓	59.8
J48	✓	✓	✓		67.2
	✓	✓	✓	✓	67.0
				✓	61.8

Table 5: Subjectivity classification precision results with various feature combinations.

As can be seen from Table 6, there are more feature combinations in the polarity classification step. The goal of this step is to differentiate between positive and negative paragraphs, and thus we could experiment with combinations that did not include neutral COTs. As can be seen from the highlighted precision score, this is in fact what yields the best result of 73.2%, which is in line with the current

[5] http://www.cs.waikato.ac.nz/~ml/index.html

state-of-the-art. Before objective paragraphs were removed, the system only yields 59.7% precision, which shows that a significant improvement is obtained by using a two-step classification system. It is interesting to observe that negations do not have a positive impact on these results either. Throwing out COTs yields very low precision scores in all cases.

	PosCots	NeutCots	NegCots	Negations	Precision
NB	✓	✓	✓		72.8
	✓	✓	✓	✓	69.5
	✓		✓		**73.2**
	✓		✓	✓	71.4
				✓	57.4
RF	✓	✓	✓		64.2
	✓	✓	✓	✓	63.5
	✓		✓		66.1
	✓		✓	✓	64.2
				✓	55.6
J48	✓	✓	✓		72.2
	✓	✓	✓	✓	71.1
	✓		✓		72.3
	✓		✓	✓	71.6
				✓	—

Table 6: Polarity classification precision results with various feature combinations.

For both steps, NB is the machine learning algorithm that performs best. However, J48 outperforms the other two when including negations. It is important to note that there is no score for J48 with only negations included. This is because this model was not able to build a decision tree based on this feature. Lastly, RF is the worst performer in all cases.

5 Discussion

The results in tables 5 and 6 only include the overall precision scores obtained during testing. However, in each classification step there are separate precision and recall scores for each class involved in the classification. These results can shed some light on which parts of the system perform better than others. Tables 7 and 8 summarize those results.

Class	**Precision**	**Recall**
Neutral	57.1%	**85%**
Subjective	**76%**	42.7%
Overall	67.1%	61.1%

Table 7: Precision and recall for subjectivity classification.

As can be seen in Table 7, the subjective class achieves a precision of 76%, which means that in a live system where the input of polarity classification are only subjective texts, 76% of these will be correctly classified. On the other hand, there will be many false negative cases, as can be seen from the low recall value. In contrast, the recall yields 85% in the objective class, while the precision is lower, 57.1%. This means that our engine does not have enough information to decide when a paragraph bears a sentiment. Instead, the features represented in this classification step are not suitable in order to detect sentiment, and thus it misclassifies too many subjective paragraphs as objective. As discussed in 3.2, the difference in the average of negations per paragraph seemed to indicate that this feature was suitable for subjectivity

classification. However, this is not the case. We believe that the complexity of the scope negation can be the reason for this. In order to include negations and observe a positive impact in the classifier, the syntactic structure of the sentence should probably be considered. This could, in turn, make the system considerably lower and make this solution unpractical from a real-system point of view.

Similar results are presented in Table 8. The positive class shows poor results in terms of recall, though in terms of precision ranks higher than the overall, achieving satisfactory results. For the negative class, precision is good enough, 70% whereas recall yields a much higher value of 88.2%.

Class	Precision	Recall
Positive	**77.4%**	51.6%
Negative	70%	**88.2%**
Overall	73.2%	72.1%

Table 8: Precision and recall for polarity classification.

6 Conclusion and future work

In this paper we have presented a two-step classification system to detect sentiment in the political news domain for an under-resourced language such as Norwegian. COTs and simple negation counts were included as features in this system. By performing a 10-fold crossvalidation, we obtain close to state-of-art results in this task, over 70%, which is an optimistic value given the inherent complexity of this domain. Interestingly, negation words do not contribute to either classification task. We believe that this is because of the semantics of the scope of negation, that goes beyond singular words. We intend to investigate further on how to deal with negation in real-time systems in future work. All experiments have been conducted in the Norwegian political newswire domain at the paragraph level. In future work, we want to continue the work in this domain. More specifically, we plan to experiment with sentiment targets, such as events or named entities, within text units to detect the sentiment around them.

References

Alexandra Balahur and Ralf Steinberger. 2009. Rethinking sentiment analysis in the news: from theory to practice and back. *Proceeding of WOMSA*, 9.

Evgenia Belyaeva and Erik van Der Goot. 2008. News bias of online headlines across languages. *The study of conflict between Russia and Georgia*.

Fortuna Blaz, Carolina Galleguillos, and Nello Cristianini. 2009. Detecting the bias in media with statistical learning methods text mining: Theory and applications.

Ronen Feldman. 2013. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89.

Gregory Grefenstette, Yan Qu, James G Shanahan, and David A. Evans. 2004. Coupling niche browsers and affect analysis for an opinion mining application. In *Coupling approaches, coupling media and coupling languages for information retrieval*, pages 186–194. Le Centre de Hautes Etudes Internationales d'Informatique Documentaire.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Mesut Kaya, Guven Fidan, and Ismail H. Toroslu. 2012. Sentiment analysis of turkish political news. In *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 174–180. IEEE Computer Society.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1367. Association for Computational Linguistics.

William A. Ladusaw. 1992. Expressing negation. In Chris Barker and David Dowty, editors, *Proceedings of the second conference on semantics and linguistic theory*, pages 237–59, Columbus. The Ohio State University.

Howard Lasnik. 1972. *Analyses of Negation in English*. Ph.D. thesis, Massachusetts Institute of Technology.

Yutaka Matsuo and Mitsuru Ishizuka. 2004. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(01):157–169.

Rada Mihalcea, Carmen Banea, and Janyce M Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections.

Pål-Christian Salvesen Njølstad, Lars Smøras Høysæter, and Jon Atle Gulla. In press. Optimizing supervised sentiment lexicon acquisition: Selecting co-occurring terms to annotate for sentiment analysis of financial news. *Language Resources and Evaluation*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Barbara H. Partee. 1992. On the "scope of negation" and polarity sensitivity. In Eva Hajicova, editor, *Functional Description of Language: Proceedings of the Conference*, Faculty of Mathematics and Physics, Charles University, Prague, 24-27 November.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 129–136. Association for Computational Linguistics.

Yongheng Zhao and Yanxia Zhang. 2008. Comparison of decision tree methods for finding active objects. *Advances in Space Research*, 41(12):1955–1959.

Fast Inference for Interactive Models of Text

Jeffrey Lund, Paul Felt, Kevin Seppi, Eric K. Ringger
Department of Computer Science
Brigham Young University
{jefflund, paul_felt, kseppi}@byu.edu, ringger@cs.byu.edu

Abstract

Probabilistic models are a useful means for analyzing large text corpora. Integrating such models with human interaction enables many new use cases. However, adding human interaction to probabilistic models requires inference algorithms which are both fast and accurate. We explore the use of Iterated Conditional Modes as a fast alternative to Gibbs sampling or variational EM. We demonstrate superior performance both in run time and model quality on three different models of text including a DP Mixture of Multinomials for web search result clustering, the Interactive Topic Model, and MOMRESP, a multinomial crowdsourcing model.

1 Introduction

One of the most popular and useful approaches for analysis of large bodies of text documents is probabilistic models. For example, topic models such as Latent Dirichlet Allocation (LDA) can automatically learn topics from a set of documents, giving users a glimpse into the common themes of the data (Blei et al., 2003). Other models such as the Mixture of Multinomials can be used to perform document clustering allowing users to automatically organize text data (Meila and Heckerman, 2001; Walker and Ringger, 2008).

We are interested in use cases for probabilistic models of text which include human interaction. For example, the Interactive Topic Model (ITM) is a topic model that extends LDA to allow the user to inject model constraints in the form of word groupings while the topics are being learned (Hu et al., 2011). By including the user in the training process rather than simply learning the topics offline, the user can fine-tune the resulting topic model to better suit individual user needs and to accommodate a user's domain knowledge. However, if the training algorithm is too slow, the delay between receiving user feedback and presenting the updated model will harm the interaction due to increased cognitive load. Consequently, we require an inference algorithm which is both fast enough to facilitate interaction, and maintains (or improves upon) the accuracy of existing inference techniques.

For models like LDA, we typically perform training by calculating *maximum a posteriori* estimates of the latent topic variables and parameters given observed document data, with the idea that the setting of topic variables and parameters which maximizes the posterior distribution will best explain the observed data. Although various exact methods exist, such as belief propagation (Pearl, 1988) and the junction tree algorithm (Koller and Friedman, 2009), the complexity of exact posterior inference on such models is NP-HARD in general, so we resort to various approximations in order to optimize the posterior distribution (Sontag and Roy, 2009; Cooper, 1990). Some popular algorithms for approximate posterior inference include Gibbs sampling and mean field variational inference.

Each of these approximate inference algorithms has some drawbacks. For example, while variational inference is often very fast, it makes simplifying assumptions about the posterior distribution which can seriously degrade the quality of solutions for certain models, such as Mixture of Multinomials (Walker, 2012). However, for other models such as LDA we can achieve good estimates very quickly (Asuncion et al., 2009). Gibbs sampling provably generates samples from the posterior distribution and unlike

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2997–3006, Osaka, Japan, December 11-17 2016.

variational inference, it is theoretically able to explore the entire support of the posterior manifold. Unfortunately, any reasonable restriction on the run time of the sampler means that we will only be able to explore a localized area of the support. Consequently, for most uses of probabilistic models of text, practitioners run a sampler for a period of time in the hope of finding an area of high probability, and then use the final sample as an approximation for the mode. While for some models, such as Mixture of Multinomials, this technique gives very good results (Walker and Ringger, 2008; Rigouste et al., 2007; Zhong and Ghosh, 2005), the lack of a convergence criteria can make the technique too slow for applications which require user interaction. For example, (2004) found that LDA requires hundreds of iterations of sampling before the log-likelihood of the model stabilizes in distribution. In practice, many of the probabilistic models of text we are interested in require similar numbers of sampling iterations.

As an alternative to techniques which introduce strong assumptions for posterior inference (e.g., variational inference) or lack clear and timely convergence criteria (e.g., Gibbs sampling), we will examine the use of Iterated Conditional Modes or ICM (Besag, 1986; Wellner et al., 2004), This algorithm is able to quickly achieve locally optimal *maximum a posteriori* estimates.

In section 2, we will briefly describe the ICM algorithm and compare it with other existing techniques. Then in section 3 we will empirically examine the performance of ICM in the context of three very different probabilistic models of text which can be used interactively. We first show that ICM performs well in the context of a non-parametric model by experimenting with a Dirichlet Process Mixture of Multinomials applied to the problem of web search result clustering. We then turn our attention to the Interactive Topic Model (Hu et al., 2011) to show that ICM improves performance over the previously published Gibbs sampler. Finally, we use ICM in the context of MOMRESP, a probabilistic model designed to infer true document class labels from noisy crowdsourced judgments (Felt et al., 2014).

2 Iterated Conditional Modes

Suppose we are given a probabilistic model of text with observed data x and unobserved variables θ. For the purpose of this discussion, θ may represent any number of unobserved parameters and latent variables. These parameters and variables can be either continuous or discrete. Like Gibbs sampling, Iterated Conditional Modes (ICM) relies on the fact that while computing a posterior distribution of the form $p(\theta|x)$ may be intractable, computing the complete conditional for a single variable θ_i while holding fixed both x and the rest of the parameters $\theta_{\neg i}$ is feasible in models with local conjugacy. By using the tractable complete conditional distribution $p(\theta_i|\theta_{\neg i}, x)$ we are able to locally maximize the posterior without the need to approximate the posterior with samples.

The general procedure for ICM is very similar to Gibbs sampling. We cycle through each unobserved variable θ_i in the model and update current value of the variable to be the mode of its complete condition distribution. The ICM update is repeated until convergence when the value of each θ_i is already the mode of its complete conditional distribution.

To see that Iterated Conditional Modes will find a local maxima of the posterior distribution, we will demonstrate that the ICM updates monotonically increase the current estimate of the posterior probability $p(\theta|x)$. Since the data x is fixed, the posterior is proportional to the joint distribution over all of the variables and data:

$$p(\theta|x) = \frac{p(\theta, x)}{p(x)} \propto p(\theta, x) \tag{1}$$

Using the chain rule, for some i we can then factor the joint distribution as:

$$p(\theta, x) = p(\theta_i|\theta_{\neg i}, x) \cdot p(\theta_{\neg i}|x) \cdot p(x) \tag{2}$$

Since x is fixed, we can also write

$$p(\theta, x) \propto p(\theta_i|\theta_{\neg i}, x) \cdot p(\theta_{\neg i}|x) \tag{3}$$

While updating the parameter θ_i, $\theta_{\neg i}$ is held fixed, which means that the second term $p(\theta_{\neg i}|x)$ is constant and the factored joint distribution is proportional to the complete conditional, thus:

$$p(\theta|x) \propto p(\theta_i|\theta_{\neg i}, x) \tag{4}$$

Since these two expressions are proportional, setting θ_i to the mode of its complete conditional will only increase the value of the posterior probability. Thus our update rule for the variable θ_i is given as:

$$\hat{\theta}_i = \underset{k}{\operatorname{argmax}}\ p(\theta_i = k|\theta_{\neg i}, x) \tag{5}$$

Since this update equation monotonically increases the estimate of the posterior probability, and the value of the posterior probability is bounded above by 1, we can use the monotone convergence theorem to conclude that this algorithm will converge to a local maximum in the posterior distribution. Furthermore, the ICM algorithm is able to do so without approximating the whole posterior distribution.

Iterated Conditional Modes is related to Expectation Conditional Maximization (Meng and Rubin, 1993) in that it employs conditionals to find local maxima in a distribution. However, unlike Expectation Conditional Maximization which maximizes a likelihood, ICM is not a variant of EM, as it takes into account prior distributions when computing the complete conditionals. In fact, we could describe ICM as a particular limit of Gibbs sampling in the same way that K-means can be viewed as the deterministic limit of the EM algorithm (Bishop, 2006).

Note however that the efficiency of the ICM algorithm depends entirely on the ability to quickly compute the mode of the conditional distributions. If the complete conditional distribution is a density, this may involve continuous optimization. However, there is a wide class of models for computing the mode of the conditional distribution is easy. Typically this is done through the use of conjugate priors which make the conditional distributions tractable. For example, for many probabilistic models of text, computing the mode of the conditional distribution is often easy due to the frequent use of the Dirichlet-Multinomial conjugate pair.

To be clear, Iterated Conditional Modes is a coordinate ascent algorithm, and as such, it can only locally optimize the posterior — there is no guarantee of finding a global maximum. Thus if the posterior manifold contains many sub-par local maxima, then ICM will be sensitive to initialization and may perform poorly. Random restarts may mitigate the problem. Alternatively, an initialization strategy which consistently starts the inference procedure near a good solution will yield better *maximum a posteriori* estimates. The best initialization strategy depends on the model to be optimized. Thus each experiment described below includes its own initialization strategy.

3 Experiments

We now demonstrate that Iterated Conditional Modes converges quickly enough to allow for interactive use cases involving various probabilistic models of text, while yielding high quality estimates. In the hopes of demonstrating the general applicability of the technique, we do so on three different models and tasks. The first model is a DP Mixture of Multinomials applied to the task of web search result clustering. We choose this model to show that ICM can work in the context of non-parametric models. The second task is the Interactive Topic Model or ITM. We choose this model to suggest that ICM may be viable for a wide variety of interactive topic modeling applications. Finally, we will apply ICM to the MOMRESP model, which is a probabilistic model for producing annotated corpora for NLP and machine learning research.

3.1 Web Search Result Clustering

As many as 16% of queries issued to search engines contain ambiguous search terms (Song et al., 2007). After issuing a search query with this kind of ambiguity, users may become confused by seemingly unrelated search results, or they may be slowed by the need to narrow the scope of the query. For example, suppose a user issues the query "tiger." The user may be surprised to see results about the large feline, the golfer Tiger Woods and the German tanks used in the 1940s, when only one of those meanings of "tiger" was intended. Web search result clustering helps users deal with query ambiguity by automatically discovering clusters among the search results and presenting the results as clusters (Carpineto et al., 2009b). With a web search result clustering system, the user can select the cluster in which they are actually interested, and immediately filter out irrelevant results. An example of such a system is the Carrot[2]

search framework, which is available online both as a web service and as a downloadable application.*

Client-side web search result clustering systems do not maintain their own search index or data but instead rely on search results (specifically the snippets) returned from an external search engine chosen by the user. This allows users to utilize web search result clustering systems on a wide variety of search engines, both public and private. Since web search result clustering systems are meant to work with arbitrary search results, the computation typically takes place client-side. An important consequence is that web search result clustering systems should be able cluster extremely small amounts of data: rather than tens of thousands of full documents encountered in typical document clustering tasks, a web search result clustering systems uses around 100 documents, each of which consist of no more than a sentence or two. Furthermore, web search result clustering systems must be run quickly enough to facilitate web search. For simple interactions like issuing a web search query, the interaction take less than one second (Cook and Thomas, 2005). This stands in contrast to typical document clustering settings which can be run offline possibly using parallel computation resources rather than online using a single commodity machine. Due to these run time constraints, it has been argued that traditional document clustering techniques may not work out of the box (Carpineto et al., 2009b). Consequently, various specialized algorithms for the problem of web search result clustering have been published.

The best reported solution to the problem of web search result clustering employs maximal spanning trees to perform word sense induction (Di Marco and Navigli, 2011). The algorithm, referred to as MST, uses the Google Web1T n-gram data set (Brants and Franz, 2006) to create a co-occurrence graph on the words in the snippet results and then calculates maximal spanning trees to remove edges from the graph. This process repeats until the desired number of word clusters is formed. Unfortunately, the requirement of large amounts of n-gram data is not amenable to client-side computation. Even just maintaining an up-to-date n-gram data set (so that the system can handle queries related to fast-changing subjects such as recent popular culture) is also necessary but requires web-scale data-gathering resources. Consequently, MST is not a client-side solution.

There are, however, a number of approaches which are amenable to client-side web search result clustering. One such system is Lingo, which was developed for use in the Carrot^2 search framework (Osiński et al., 2004). Another is KEYSRC, which extracts key phrases from snippet data and then uses hierarchical agglomerative clustering on those phrases (Bernardini et al., 2009).

Despite the fact that model-based approaches tend to yield higher quality results in the general problem of document clustering (Zhong and Ghosh, 2005), no study has applied model-based clustering to the specialized problem of web search result clustering. We rectify the lacuna by applying Iterated Conditional Modes to a Dirichlet Process Mixture of Multinomials model (hereafter referred to as DP-MOM), and we compare our results to those of the previously studied web search result clustering solutions.

Our model-based approach employs a Dirichlet Process (DP) mixture model, a well studied Bayesian non-parametric model (Antoniak and others, 1974; Neal, 2000). This type of model has been used to perform document clustering, albeit with modifications to include feature selection in the model (Yu et al., 2010). Given the scarcity of data in this application, we cannot realistically perform feature selection (although the snippet generation itself might be viewed as feature selection).

DP mixture models have a known relationship with the Chinese Restaurant Process (CRP) in that if we integrate over the random mixing measure in the DP mixture, the resulting model will have a CRP prior over the mixture components (Blackwell and MacQueen, 1973). Taking advantage of this relationship, the DP-MOM model can be written with the following form:

$$\phi_k|\beta \sim Dirichlet(\beta), k = 1, ..., K$$

$$z_d|\alpha \sim CRP(\alpha), d = 1, ..., M$$

$$w_d|z_d, \phi \sim Multinomial(N_d, \phi_{z_d}), d = 1, ..., M$$

where ϕ_k is the word distribution for topic k with a symmetric Dirichlet prior of β, z_d is the cluster assignment of document d, α is CRP concentration, and w_d gives the observed token counts for document d. M is the number of documents, and N_d is the number of tokens in the dth document.

*http://search.carrot2.org

Algorithm	AMBIENT	MORESQUE	All
ICM DP-MOM	**.768**	**.570**	**.625**
Gibbs DP-MOM	.758	.544	.604
KEYSRC	.665	.558	.588
Lingo	.628	.527	.555
MST	.815	.867	.852

Table 1: Clustering quality results, as measured by the Rand index. We include the MST results for reference though they do not constitute client-side results. Bold indicates the best client-side result, and underline indicates the absolute best result.

Following the advice of (2000), we derive a collapsed Gibbs sampler by integrating over ϕ. The complete conditional probability for the cluster assignment z_d, given the other assignments $z_{\neg d}$ and data is

$$p(z_d = j | z_{\neg d}, w) \propto \begin{cases} c_j \prod_{v \in V} \frac{\Gamma(\beta + n_{jv} + w_{dv})}{\Gamma(|V|\beta + n_{j\cdot} + w_{d\cdot})} \frac{\Gamma(|V|\beta + n_{j\cdot})}{\Gamma(\beta + n_{jv})}, & \text{if } c_j > 0 \\ \alpha \prod_{v \in V} \frac{\Gamma(\beta + w_{dv})}{\Gamma(|V|\beta + w_{d\cdot})} \frac{\Gamma(|V|\beta)}{\Gamma(\beta)}, & \text{otherwise} \end{cases} \tag{6}$$

where V is the set of words in the data, c_j is the count of documents assigned to cluster j, n_{jv} is the number of times the word type v is present in a document assigned to cluster j, and w_{dv} is the number of times word v is found in document d. Dots in the subscripts of these counters indicate marginalization over the missing index. For the sake of space, we omit the derivation, but it is similar to the derivation for the finite Mixture of Multinomials given by (2008).

We can also derive a mean field variational inference algorithm for DP-MOM. Such an algorithm, while fast, yields extremely poor *maximum a posteriori* estimates for DP-MOM (Walker, 2012). At least for this model, the independences introduced by the mean field assumption are too strong. Consequently, we compare ICM to a baseline Gibbs sampler instead of variational inference.

The final detail needed for implementing DP-MOM is an initialization strategy. A key advantage of using a non-parametric model is that the model can learn the number of clusters from data, thereby allowing our model to perform well with varying amounts of query term ambiguity. We can either initialize with a large number of clusters and let the model shrink to fit the data or to start with a small number of clusters and grow to fit the data. Our experiments indicate that starting with a single cluster performed the best, so we utilized this initialization strategy for our results.

In order to validate that our model-based approach performs well, we follow the same methodology as Di Marco and Navigli (2011) when evaluating the MST algorithm. We experiment with two different datasets: AMBIENT (Carpineto et al., 2009a) and MORESQUE (Di Marco and Navigli, 2011). Each dataset is a set of search queries issued to the YAHOO! search engine, along with the top 100 search result snippets which have all been manually labeled with topics. The primary difference between the two datasets is that the earlier AMBIENT dataset consists of single word queries, while the MORESQUE dataset extends AMBIENT to queries of length 2–4[†]. Taking both datasets together, we have a total of 158 ambiguous queries, each with between 3 and 15 topics.

Still following Di Marco and Navigli (2011), we evaluate clustering performance on these ambiguous query datasets with two metrics. The first is the Rand Index (Rand, 1971), a measure of similarity between two clusterings over the same set of elements. The Rand Index can be viewed a kind of accuracy, since it gives the percentage of pairing decisions which were correctly made with respect to a base clustering.

Table 1 summarizes the results of the various web search clustering algorithms measured by Rand Index. We see that the MST algorithm performs the best, but we remind the reader that this algorithm far exceeds the computation resource requirements for client-side web search result clustering so it only serves as a baseline. Among previously studied algorithms which respect resource constraints, our model-based approach outperforms existing techniques by a wide margin. Interestingly, Iterated Conditional Modes outperforms Gibbs sampling, indicating that for this model and this task, the extra exploration within

[†]Note that the data we use to cluster is the resulting snippets, *not* the queries themselves.

Algorithm	K=3	K=5	K=10	K=15	K=20
ICM DP-MOM	**.517**	**.650**	**.811**	**.885**	**.927**
Gibbs DP-MOM	.508	.635	.795	.873	.917
YAHOO!	.492	.600	.729	.785	.827
KEYSRC	.443	.558	.720	.791	.832
MST	.547	.656	.792	.867	.907

Table 2: Diversification results on all queries, as measured by S-recall@K. The ordering provided by the YAHOO! search engine is included as a baseline. Bold indicate the best client-side result, and underline indicates the best absolute result.

a region of high probability from sampling is not as important as the ability to jump to a mode in that region of high probability. Furthermore, our approach is extremely fast. Using a single core of an AMD Phenom II X6 1090T processor, the median time spent using ICM to perform clustering on results for a single query was 1.18 milliseconds. Our experiments using Gibbs sampling on the other hand took 6.78 milliseconds to complete.

Our second measure evaluates the diversification produced by a clustering algorithm. As outlined by Di Macro and Navigli (2011), we can use the clustering labels to re-rank the search results such that the top search results are more diverse. We measure the diversification with S-recall@K, which measures the percentage of ground-truth labeled topics present in the top K search results after re-ranking. We use the ordering returned by YAHOO! as a baseline for S-recall@K.

Table 2 shows the results of the various web search result clustering algorithms with respect to S-recall@K. In both cases, our model-based approach outperforms the baseline ranking. Both KEYSRC and Lingo actually did worse than the YAHOO! baseline. For $K \leq 5$, MST performs the best. However, for all other values of K, our model-based approach performs the best. This is likely due to the fact that our non-parametric model is able to increase the number of clusters in the presence of highly ambiguous queries, whereas the MST algorithm uses a pre-specified number of clusters.

3.2 Interactive Topic Model

We now turn our attention to the Interactive Topic Model or ITM (Hu et al., 2011). This model extends LDA by replacing the per-topic categorical distributions over words with a tree-structured Dirichlet-forest distribution. The user interactively injects constraints into the model by placing token types into Dirichlet trees. Depending on the prior for the Dirichlet trees, the constraint can either be a "must link" (positive correlation) or a "cannot link" (negative correlation) type of constraint (Andrzejewski et al., 2009). Due to issues with transitivity in "cannot link" constraints, we follow Hu et al. (2011) and focus on "must link" constraints by setting the Dirichlet trees parameter to be extremely high. A user is able to employ constraints to tell the model to give a particular set of word types similar probability within each individual topic. For the sake of brevity, we omit details about the ITM including the specific distributions for the model and the complete conditionals used to drive the collapsed Gibbs sampler for the model since they are thoroughly explained by Hu et al. (2011). We also note that variational inference is inappropriate for this model, as it only achieves good performance when used in conjunction with hyper-parameter optimization. However, such optimization tends to undo the constraints, rendering the model useless (Hu et al., 2011). Consequently, we use the published Gibbs sampler as our baseline inference algorithm.

The ITM model is trained as follows: first we train a base model with no constraints (equivalent to learning a vanilla LDA model). The user is then presented with the outcome of an analysis using the model, possibly by showing them the traditional topic lists wherein a topic is represented by the most probable words in the topic. The user then injects word constraints into the model according to the individual needs of the user or specific domain knowledge. Using the document-level ablation strategy recommended by Hu et al. (2011) the topic assignments of any document which contains a newly constrained word are revoked. In order to enforce model consistency, the rest of the topic assignments remain unchanged. Finally, inference is rerun with the new constraints and the updated model is presented to the user. This interactive process is repeated until the user is satisfied with the final state of the model.

We now investigate which inference algorithm performs the interactive model updates best. Cook and

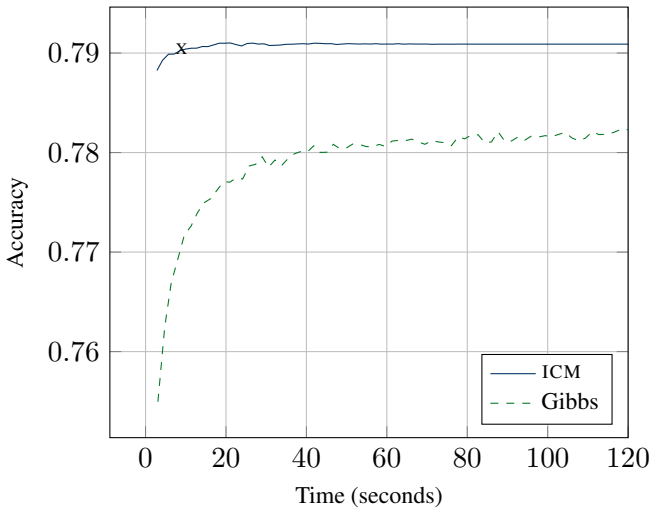


Figure 1: Accuracy versus time for both Iterated Conditional Modes and Gibbs sampling with the ITM. The X-mark indicates the median point of convergence for ICM.

Thomas (2005) show that for a complex user-initiated activity such as requesting a model update, we require a response time which is ten seconds or less or the human-computer interaction may suffer. Unfortunately, if we perform the recommended 30 iterations of sampling, even at one second per iteration, this can be taxing on the user. We turn to ICM as a faster alternative.

In order to validate the performance of Iterated Conditional Modes on the ITM, we employ an experimental setup similar to that of (2011) using the well-known 20 Newsgroups corpus, which consists of roughly 20,000 documents divided into 20 newsgroups. We simulate a user's constraints by selecting words using information gain with respect to the newsgroup labels. After training a base model with 100 iterations of Gibbs sampling for burn-in, we inject the simulated constraints into the model. Finally, we run inference using either ICM or Gibbs sampling from this point. We evaluate the model quality with a classification task in which we train a classifier to predict the source newsgroup of an unlabeled document using topic-word features. To do so, the corpus is split into a training and test set, and each word along with its assigned topic is used as a feature for the classifier. We report the classification accuracy from a support vector machine trained on the topic-word pairs from the documents in the training set. As with Hu et al. (2011), we do not hope to achieve state-of-the-art classification results for this dataset, but we do hope that the classification trends will demonstrate which inference algorithm better drives the model towards the original (withheld) human labels once the simulated constraints have been added.

As shown in Figure 1 Iterated Conditional Modes outperforms Gibbs sampling. This indicates that there is more value in reaching a local maximum than there is in the exploration that comes from sampling. More importantly, ICM has the potential to run much faster than Gibbs sampling: rather than running a Gibbs sampler for the recommended 30 iterations, the median number of iterations required for ICM to converge was 9, which allows us to present the updated model to the user within the ten second time frame recommended by Cook and Thomas (2005).

3.3 MOMRESP

Microtask markets such as Amazon's Mechanical Turk (mturk.com) allow corpora to be labeled at extremely low cost, a practice known as crowdsourcing. However, the recent emergence of crowdsourcing as the preferred method for labeling document corpora has introduced an important research problem: how to mitigate the inaccuracy of crowdsourced judgments. A common solution is to obtain multiple redundant judgments, or annotations, and aggregate them using a baseline strategy such as *majority vote*.

When annotations are both plentiful and highly accurate, majority vote works well. However, crowdsourced annotations are seldom highly accurate. State-of-the-art solutions are model-based and use standard inference algorithms. For example, the MOMRESP model presented by Felt et al. (2014) describes

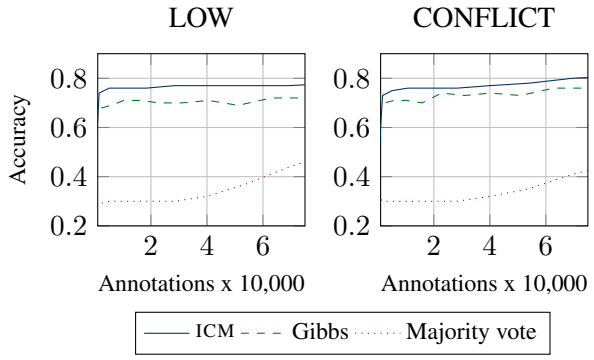


Figure 2: Accuracy of inferred labels versus the number of annotations given to the model. At the last plotted point each document has on average nearly 4 annotations. Gibbs and ICM use MOMRESP. A majority vote baseline is also shown for reference.

a joint model over document features and annotations. We include here a sketch of the model, deferring details to the referenced paper. Documents and annotations are both modeled as count vectors with multinomial distributions conditioned on the true but unobserved class label. Parameters include both per-class word distributions and class confusion matrices for each annotator. When annotations are scarce or of low-quality, the MOMRESP model trained with Gibbs sampling significantly outperforms majority vote in terms of inferred label accuracy. Labels inferred by Iterated Conditional Modes are even more accurate.

In order to validate this claim, we run MOMRESP with both Gibbs and ICM on synthetic annotations produced for the 20 newsgroups dataset. We draw synthetic annotators from the LOW and CONFLICT annotator pools described by (2014). Each pool consists of 5 annotators. In both pools, annotators give correct judgments with probabilities .5, .4, .3, .2, .1, respectively. In the LOW pool, annotator errors are distributed uniformly across incorrect classes. In the simulated CONFLICT pool, errors are systematic: a confusion matrix is created for each annotator whose diagonal is set to the annotator's accuracy and whose off-diagonal row entries are sampled from a symmetric Dirichlet distribution with parameter 0.1, to encourage sparsity, and then scaled so that each row sums to 1. CONFLICT errors are produced by corrupting true labels according to this confusion matrix. Documents are annotated in random order without replacement, and after all documents have one annotation, the process is repeated. Simulated annotation continues until we have reached the desired number of annotations. We then initialize MOMRESP using majority vote to set initial class label values and perform posterior inference using both Gibbs and ICM. We compare the model-inferred class labels with the gold standard class labels for each document in order to compute model accuracy.

Figure 2 plots the inferred label accuracy of Gibbs and ICM as well as majority vote for reference. Regardless of the number of annotations, ICM yields better accuracy than Gibbs sampling for both the LOW and the CONFLICT cases. While not shown, this trend hold even cases where majority vote outperforms MOMRESP.

In addition to inferring more accurate document labels than Gibbs, ICM has a run time which is orders of magnitude faster than that of Gibbs sampling. The median time of convergence was 6.72 seconds. This falls well within the run time recommended by Cook and Thomas (2005) for complex user-initiated tasks such as rerunning inference on MOMRESP given additional annotations. Consequently, if MOMRESP were to be adapted for an active learning task, then ICM would provide not only accurate posterior inference, but run times which are amenable to active learning.

4 Conclusion

Iterated Conditional Modes is a coordinate ascent algorithm that yields locally optimal *maximum a posteriori* estimates for models with tractable complete conditionals. We have shown that ICM identifies *maximum a posteriori* solutions that are superior to those found by Gibbs sampling for three applica-

tions: web search result clustering, topic modeling, and crowdsourcing problems. In addition, Iterated Conditional Modes has termination criterion which is easily identified, while it can be difficult to determine when a Gibbs sampler has reached the stationary distribution. Because of the convergence of ICM, we were able to significantly speed up the three applications compared to Gibbs sampling, enabling better human interactivity. These experiments motivate further exploration of this inference technique, particularly in interactive use cases of models in which both run time and model quality are crucial.

Acknowledgements

This work was supported by the NSF Grant IIS-1409739. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

References

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 25–32. ACM.

Charles E Antoniak et al. 1974. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174.

Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. 2009. On smoothing and inference for topic models. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 27–34. AUAI Press.

Andrea Bernardini, Claudio Carpineto, and Massimiliano D'Amico. 2009. Full-subtopic retrieval with keyphrase-based search results clustering. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*, volume 1, pages 206–213. IET.

Julian Besag. 1986. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 259–302.

Christopher M Bishop, 2006. *Pattern recognition and machine learning*, volume 1, pages 443–444. Springer New York.

David Blackwell and James B MacQueen. 1973. Ferguson distributions via pólya urn schemes. *The Annals of Statistics*, pages 353–355.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent dirichlet allocation. *JMLR*, 3:993–1022.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1.

Claudio Carpineto, Stefano Mizzaro, Giovanni Romano, and Matteo Snidero. 2009a. Mobile information retrieval with search results clustering: Prototypes and evaluations. *Journal of the American Society for Information Science and Technology*, 60(5):877–895.

Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. 2009b. A survey of web clustering engines. *ACM Computing Surveys (CSUR)*, 41(3):17.

Kristin A. Cook and James J. Thomas. 2005. Illuminating the path: The research and development agenda for visual analytics. Technical report, Pacific Northwest National Laboratory (PNNL), Richland, WA (US).

Gregory F Cooper. 1990. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, 42(2):393–405.

Antonio Di Marco and Roberto Navigli. 2011. Clustering web search results with maximum spanning trees. In *AI* IA 2011: Artificial Intelligence Around Man and Beyond*, pages 201–212. Springer.

Paul Felt, Robbie Haertel, Eric Ringger, and Kevin Seppi. 2014. Momresp: A bayesian model for multi-annotator document labeling. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, May. European Language Resources Association (ELRA).

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235.

Yuening Hu, Jordan L Boyd-Graber, and Brianna Satinoff. 2011. Interactive topic modeling. In *ACL*, pages 248–257.

Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

Marina Meila and David Heckerman. 2001. An experimental comparison of model-based clustering methods. *Machine learning*, 42(1-2):1–2.

Xiao-Li Meng and Donald B Rubin. 1993. Maximum likelihood estimation via the ecm algorithm: A general framework. *Biometrika*, 80(2):267–278.

Radford M Neal. 2000. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265.

Stanisław Osiński, Jerzy Stefanowski, and Dawid Weiss. 2004. Lingo: Search results clustering algorithm based on singular value decomposition. In *Intelligent information processing and web mining*, pages 359–368. Springer.

Judea Pearl. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850.

Loïs Rigouste, Olivier Cappé, and François Yvon. 2007. Inference and evaluation of the multinomial mixture model for text clustering. *Information processing & management*, 43(5):1260–1280.

Ruihua Song, Zhenxiao Luo, Ji-Rong Wen, Yong Yu, and Hsiao-Wuen Hon. 2007. Identifying ambiguous queries in web search. In *Proceedings of the 16th international conference on World Wide Web*, pages 1169–1170. ACM.

David Sontag and Daniel M Roy. 2009. Complexity of inference in topic models. In *NIPS Workshop on Applications for Topic Models: Text and Beyond*.

Daniel David Walker and Eric K. Ringger. 2008. Model-based document clustering with a collapsed gibbs sampler. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 704–712. ACM.

Daniel David Walker. 2012. *Bayesian text analytics for document collections*. Ph.D. thesis, Brigham Young University.

Ben Wellner, Andrew McCallum, Fuchun Peng, and Michael Hay. 2004. An integrated, conditional model of information extraction and coreference with application to citation matching. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 593–601. AUAI Press.

Guan Yu, Ruizhang Huang, and Zhaojun Wang. 2010. Document clustering via dirichlet process mixture model with feature selection. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 763–772. ACM.

S. Zhong and J. Ghosh. 2005. Generative model-based document clustering: a comparative study. *Knowledge and Information Systems*, 8(3):374–384.

Combining Heterogeneous User Generated Data to Sense Well-being

Adam Tsakalidis
University of Warwick
Coventry, UK
a.tsakalidis@warwick.ac.uk

Maria Liakata
University of Warwick
Coventry, UK
m.liakata@warwick.ac.uk

Theo Damoulas
University of Warwick
Coventry, UK
t.damoulas@warwick.ac.uk

Brigitte Jellinek
Salzburg University of Applied Sciences
FH Salzburg, Puch Urstein Österreich
brigitte.jellinek@fh-salzburg.ac.at

Weisi Guo
University of Warwick
Coventry, UK
weisi.guo@warwick.ac.uk

Alexandra I. Cristea
University of Warwick
Coventry, UK
a.i.cristea@warwick.ac.uk

Abstract

In this paper we address a new problem of predicting affect and well-being scales in a real-world setting of heterogeneous, longitudinal and non-synchronous textual as well as non-linguistic data that can be harvested from on-line media and mobile phones. We describe the method for collecting the heterogeneous longitudinal data, how features are extracted to address missing information and differences in temporal alignment, and how the latter are combined to yield promising predictions of affect and well-being on the basis of widely used psychological scales. We achieve a coefficient of determination (R^2) of $0.71 - 0.76$ and a ρ of $0.68 - 0.87$ which is higher than the state-of-the art in equivalent multi-modal tasks for affect.

1 Introduction

The World Health Organisation describes mental health as "the foundation for well-being and effective functioning for an individual and for a community" and highlights the importance of selecting suitable indicators of mental health (Herrman et al., 2005). One can distinguish between macro-level indicators, which are meant to provide a picture of generic well-being across a large population, usually at national scale, and individual indicators of mental health. Most of the macro measures typically use statistics from census, administrative and economic sources to measure the social and economic macro-environment as important determinants of mental health (e.g. Human Development Index, Gender Development Index, Human Poverty Indices (OECD, 2013). With the advent of widely available social media data, there have also been efforts to automatically obtain macro indicators of well-being and happiness, primarily through the analysis of geolocated Twitter posts (Dodds et al., 2011; Lansdall-Welfare et al., 2012; Lampos et al., 2013). These pieces of work seek to identify occurrence patterns for words with pre-defined affect scores at different levels of temporal granularity. Such approaches, with more sophisticated components for emotion recognition in social media content, can be alternatives to public surveys for mood and happiness indicators.

At the other end of the spectrum we have individual indicators of mental health. These include measures of positive mental health, such as coherence & meaning in life, self-esteem etc. as well as indicators of mental distress, such as negativity, anxiety, depression (Herrman et al., 2005). These measures can be used by experts or individuals for diagnostic and management purposes, but also in aggregation, for large scale surveys. However, the reliance on self-reporting required to obtain these measures is time consuming and expensive and can only produce sparse data on small populations. Moreover, self-reporting is likely to introduce bias into results. Recent work (Rachuri et al., 2010; Lathia et al., 2012; Canzian and Musolesi, 2015; Pejovic et al., 2015) shows the potential of experience sampling using mobile devices

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3007–3018, Osaka, Japan, December 11-17 2016.

for behavioural studies and clinical care, especially relating to mental health. A variety of longitudinal sensor data from a smart phone as well as location information, obtained passively from the user's phone, can be calibrated against the user's responses to behaviour or emotion related questions. The latter are usually harvested through regular prompts for input provided by a smart phone application.

Here we combine heterogeneous and asynchronous textual as well as non-linguistic data to train predictors of well-being scores that will circumvent the need for user input. Our contributions include:

- **A novel and unique dataset of heterogeneous sources** consisting of textual data from social media posts (Twitter, Facebook), SMS messages ($> 100,000$), 2436 mood forms as well as asynchronous mobile phone use data including location, Wi-Fi connection, mobile phone use and sensor data (42 GB).
- **Methodology for handling heterogeneous, incomplete and asynchronous data for longitudinal predictions**. We consider a number of baselines and appropriate normalisations as well as an approach based on multi-kernel learning, which aims to maximise the joint predictive power of each data source, and show very promising results.
- **Calibration of well-being predictors based on well established affect and well-being scales**, namely the Warwick-Edinburgh Mental Well-Being Scale (WEMWBS) (Tennant et al., 2007) and the Positive and Negative Affect Scale (PANAS) (Watson and Clark, 1988; Crawford and Henry, 2004).

While studies on macro-indicators have exploited simple textual features, we are not aware of another study which has worked on such an heterogeneous dataset for the automatic prediction of individual well-being scores, basing predictions on well established psychometric scales. Indeed to the best of our knowledge this is the first study to tackle predictions from heterogeneous, asynchrounous, longitudinal user generated content.

2 Related Work

Mobile-based studies on well-being and mental health: Researchers have used mobile phones to assess student moods and stress by correlating data from phone sensors, daily probes on student states and termly behavioural surveys (Wang et al., 2015). They have identified a strong correlation between automatic sensing data and a broad set of well-being scales. Their work focuses on the calibration of sensor data against self-reported mood without any indication of how these can be combined for prediction purposes. In a related study (Wang and Harari, 2015) employ mobile phone use data and survey data from students to predict their GPA score at the end of term. The temporal granularity here is rather coarse, while no textual data is considered and the predictive model does not consider raw data, but rather pre-built classifiers which feed into a regression model. Work by Canzian and Musolesi (2015) shows how mobility patterns based on GPS strongly correlate with depression, but other data sources, such are text, are not exploited. Jacques et al. (2015) applied a multi-task, multi-kernel approach for predicting students' wellbeing using survey, mobility, smartphone and physiology data over a one-month period; despite the ability of the prediction model to provide interpretable results by using one kernel per modality, the textual modality was not used while one of the most predictive modalities (survey data) demanded manual effort from the subjects, which is in contrast to our objective. Other studies focusing on stress detection (Bogomolov et al., 2014) and happiness recognition (Bogomolov et al., 2013) have also ignored the textual modality or require user input (e.g. personality traits) to be used by their model.
Work on multi-modal affect aims to combine synchronous audio, visual and linguistic cues to predict affect dimensions and faces the challenge of source heterogeneity, which is tackled by two main approaches: in early fusion models, the features from the different modalities are combined into a single vector, which is fed to a learning algorithm. Such approaches have been employed in various tasks including sentiment (Wang et al., 2014) and emotion analysis (Poria et al., 2015; Wimmer et al., 2008) and benefit from the ability of the learning model to capture the semantic relations between different modalities; however, the resulting features are treated in the same way by the learning model (Akbari

et al., 2015). On the contrary, in late fusion approaches different models are trained per modality and their outputs are combined at a later stage, usually by employing weighted sum (Dobrišek et al., 2013; Poria et al., 2016). Gupta et al (2014) train separate classifiers on audio and video for a particular time frame and then fuse the results together to create new meta-features, while a product rule combination method is introduced for the emotion recognition task by (Dobrišek et al., 2013). Late fusion approaches suffer from inability to capture across-modality dependencies and thus are in contrast to our objective of combining heterogeneous data.

3 A Dataset of Heterogeneous Textual and Mobile phone data

Dataset Design: In designing our dataset we wanted to collect real-world user-generated content that could provide information about the spatio-temporal influence on users' mental well-being. For this purpose, our goal was to combine longitudinal textual sources, such as messages and social media posts, with behavioural data, as manifested by mobility patterns and mobile phone usage. To control for the effect of variable age and stage of life we recruited student participants from the same university in a large cosmopolitan city (New York); unlike (Wang et al., 2015) the study was not confined to a campus environment. A cohort of 29 students gave us access to their Twitter and Facebook posts, SMS and Facebook messages as well as their mobile phone use data, together with location information and mobile phone sensors, over a period of 4 months each. Data collection was passive, with the exception of on-line submission of psychological tests for well-being (WEMWBS)(Tennant et al., 2007) and affect (PANAS)(Watson and Clark, 1988; Crawford and Henry, 2004), which students were asked to complete once a day, in the evening. WEMWBS was chosen as a robust, widely used measure of well-being, suitable for the general population and employed by the NHS. Since WEMWBS focusses on positive attributes, we also used PANAS to capture negative emotions. Unlike other work, we did not require any other manual effort from students such as the completion of on-line questionnaires mapping them to personality traits or prompts for self-reported emotional status.

Data Collection: The data was primarily collected from the Twitter API and two applications (Apps) that were installed for the purpose of the study, on the participants' mobile phones. The first App is DeviceAnalyzer (Wagner et al., 2013), which collects a wide range of time-stamped data, including location and phone usage (e.g. number and duration of calls). SMS data was collected through the NUS SMS collection App[1], which was configured to retrieve a batch of SMS messages authorised by a participant, as a weekly email. Users were asked to complete psychological scales (mood forms) by logging into a secure webserver, set up for the study. We collected a total of 2436 mood forms, each corresponding to completed PANAS and WEMWBS scales. Facebook data was downloaded by our participants twice during their time on the study and was uploaded to the secure webserver, where the participants could choose the data they wished to share and make available to us. We thus collected 111,270 textual posts and 42GB of DA data spanning the period February 2015-December 2015. Note that participants' time on the study was staggered, with each participant contributing data for 4 months.

Dataset Description: The data is heterogeneous by nature and design and asynchronous, with variable temporal granularity, reflecting a real-world scenario and presenting numerous challenges. The most challenges are presented by the DeviceAnalyzer (DA) data, due to their sheer volume and natural redundancy. For example, aggregates are required to represent most DA features (e.g. number of calls, time spent in a location etc.) but choosing the best aggregate and its respective temporal granularity is not straightforward. Moreover, timestamps are presented in epochs, so they had to be converted to absolute values, to be in alignment with those of textual data. We experimented with different methods for aggregation; for the purpose of this study, the decision was made to aggregate DA features at the hour level, by taking mean or cumulative values for the feature within an hourly interval. We selected a subset (153) of the DA features that can be potentially indicative of user behaviour, as opposed to being related to purely technical aspects of the phone. The former, among others, include: volume of images and SMS messages, physical sensor readings (physical environment and movement), location in terms of longi-

[1] http://wing.comp.nus.edu.sg:8080/SMSCorpus/contribution.jsp

tude and latitude as well as wireless network and data transfer (digital environment), battery level, ringer and other phone settings (user choices). Data is collected anonymously and linked together through user identifiers.

Location and Wi-Fi connection data: A further challenge was presented by how to make use of location and Wi-Fi connection data to allow: (a) compatibility with numeric aggregates (b) direct comparisons between different users, who inevitably spend time at different locations with different Wi-Fi connections, with no direct semantic mappings. Our solution to the above was to rank locations and Wi-Fi connections, respectively, according to the time spent in each of them, by each user. Thus we collected the top 10 locations and Wi-Fi connections per user. See also section 4.2.

Sensor data: There are 15 different sensors of which only accelerometer and light sensor data are provided by 22 of the 29 participants. Each of the two sensors corresponds to 10 different values, including resolution and range of values at a particular time-point.

Textual data: The fields associated with each textual instance are the speaker, the raw text, the absolute time stamp, the data source (e.g. Facebook) and the type of text (e.g. message).

Mood forms: Obtaining scores for the mood forms is straightforward and based on the scoring instructions associated with each of the two psychological scales.

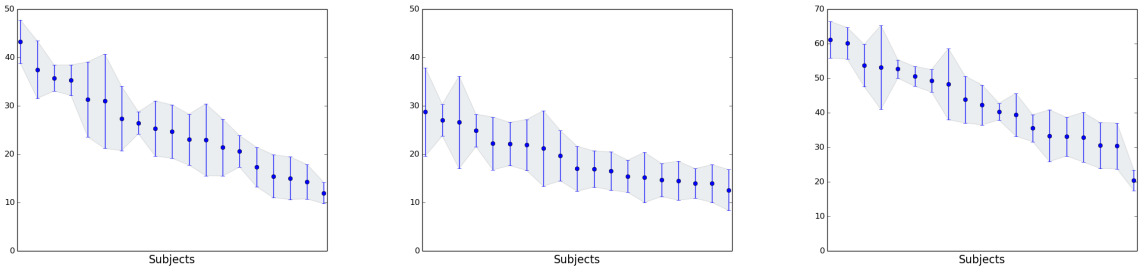
4 Methodology

4.1 Data matrix creation and Features

Our goal here is to combine features from both (i) the DeviceAnalyzer (DA) data and (ii) the textual sources (TEXT), in order to train a model that can automatically predict mood scores originating from the three daily mood forms. The latter correspond to the determination of positive affect ("positive") and negative affect ("negative"), calculated on the basis of the PANAS psychological scale and well-being ("wellbeing"), calculated on the basis of the WEMWBS psychological scale. Those three scores for positive, negative and well-being constitute our target values. Past research has shown a strong correlation between well-being and positive (r=.71) and a moderate (negative) correlation between well-being and negative affect (r=-.54) (Tennant et al., 2007). For the purpose of this work we keep the three targets distinct from each other, to aid the interpretability of results.

We had 29 participants on the study who agreed to give us access to both their DA and TEXT data and complete daily mood forms. During the study, two participants switched to iPhones, so they could no longer run DA on their mobile phones. For others, there was missing DA data, where missing data are defined as cases where one or more sources of DA data have no values for longer than a 6 hour period before the completion of a mood form, which was assumed as being most relevant for its completion. TEXT data on the other hand are never considered missing, as the lack of a post is considered to be a choice and a useful indicator of user behaviour. For the purposes of the current paper, we focused thus on the 19 users for whom we had both DA and TEXT data and no missing data in the 6 hour period prior to the completion of a mood form. This means that from an original set of 2436 mood forms, each corresponding to three mood score values, several textual posts (Twitter, Facebook, SMS) and several GB of DA data, we make use of 1438 mood forms and the corresponding features and target values.Thus, for this study, we used 40,786 textual posts written in English and the corresponding DA data ($\sim$10GB). Mood scores consist in scores for well-being, positive and negative affect. Figure 1 shows the mean values and the standard deviations for the three mood form scores based on the subjects that were used in our study. The average per-subject score is 25.2, 19.2 and 42.6 for the positive, negative and well-being target respectively. Interestingly, we observe that the average per-subject standard deviation is 5.0, 4.9 and 5.7 for the three targets, pointing to the subjects' affect and well-being fluctuations during the studied period, which makes our task more challenging and shows that simply identifying a subject based on his/her id is not sufficient for predicting his/her mood.

Our textual and DA data points have very different temporal granularity, with hundreds of DA data points in between textual posts. As mood forms are completed every 24 hours (some users being more diligent than others), we decided to extract features within the 24 hour window of a mood form. The



(a) Positive (b) Negative (c) Well-being

Figure 1: Average and standard deviations of the mood form scores obtained by the 19 subjects.

underlying assumption is that those features generated by a user during the past day are most likely to have influenced her mood, resulting in the observed mood scores. Thus, given a mood form completed by a certain user at time t, we focused on her past 24 hours before t, in order to extract our features from and aggregate these features in different time windows within the 24 hour period, and, more specifically, into 5 different windows (1, 6, 12, 18 and 24 hours before the completion of a mood form), to allow for an extra level of granularity to the effect of proximity to the mood form timestamp. This process was performed for the DA features that are described in the following section and not for the TEXT ones, which are only considered at the 24 hour window. This is due to the sparsity of some feature representations of the latter. In future work, we plan to make better use of the temporal granularity of the TEXT features and their interaction with the DA data. In the following, we describe a number of baselines, utilising subsets of the features (4.2) and different algorithms, tested under different settings (4.3) to establish the most effective approach to combining heterogeneous data for prediction.

4.2 Baseline definition

Baseline DA Features

Previous work in a controlled user study (Wang and Harari, 2015) looked at exploiting features from students' mobile phone usage within a semester, to predict student academic performance at the end of the semester. While we consider target objectives at much finer grained temporal intervals, we adopt a baseline from mobile phone data (DA) to approximate the ones considered in the StudentLife study (Wang et al., 2015). The latter relies on pre-built classifiers (i.e., accelerometer data (Lu et al., 2010)) to make use of sensor data, such as accelerometer, while we use aggregates of raw data. In our work, we have built classifiers that take into account all data variables, and as such offer more degrees of freedom, to better understand the underlying causes of emotions than studies that consist of disparate pre-built classifiers. Our DA baseline consists of:

- **Calls**: The total number and duration of the calls that a subject has made and received.
- **Locations**: The percentage of time that a subject has spent in her i^{th} preferred location.
- **Wi-Fi:**: The percentage of time that a subject has spent while connected to her i^{th} preferred Wi-Fi.
- **Other**: the percentage of time that a user's mobile: (i) headphones have been "on" ("off"); (ii) screen brightness has been set to "manual" ("auto"); (iii) airplane mode has been "active" ("inactive"); (iv) ringer mode has been set to "vibrate", "silent" and "normal"; (v) headset has been "on" ("off"); and (vi) has been disconnected, plugged in a USB port and plugged in AC.

For locations and Wi-Fi connections we generated features for $i = \{1, ..., 10\}$, the ten preferred locations and Wi-Fi connections respectively, and an eleventh feature, signaling respectively the total time spent in locations and Wi-Fi access points, other than the top ten. Figure 2 shows the projection of the locations visited by the subjects within the city of New York. All DA features were extracted from

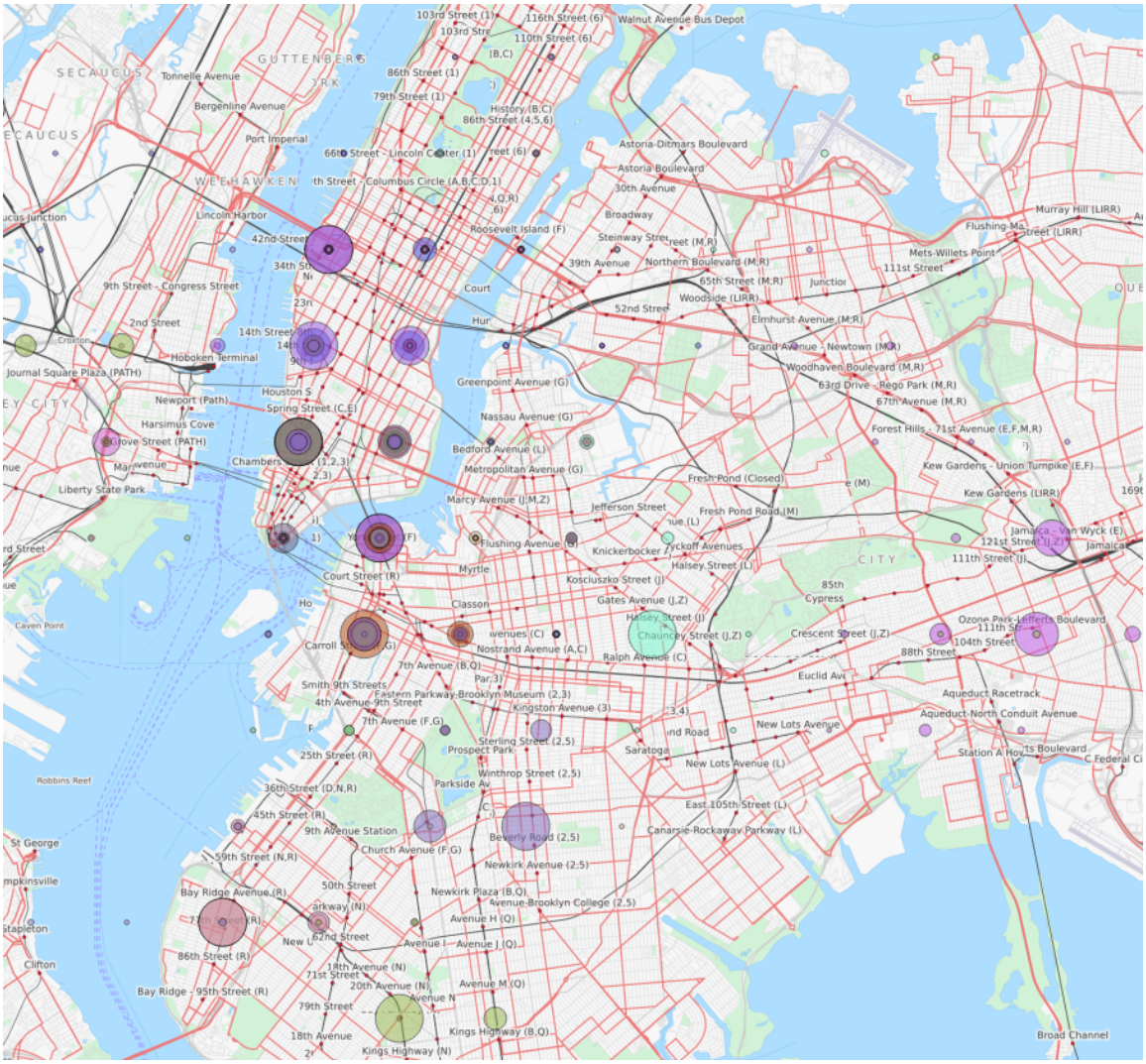


Figure 2: Geo-visual projection of the subjects' visited locations. Each colour indicates a unique user and the size of their spot indicates the number of unique GPS samples at that location.

five different time windows, before the completion of a mood form (1, 6, 12, 18 and 24 hours), leading to 200 DA features per instance. In the case of missing data for some feature (e.g. missing locations due to disconnections), we filled-in the gaps, by replacing the missing values of a feature with the past 6-hour mean of the same feature for that specific user. For example, if we have no indication of the time spent in particular locations 1 hour prior to the completion of a mood form, we use the 6-hour mean of each location feature from the 6-hour window leading up to the timestamp of the mood form for the user in question. If after this process an instance would still have some missing feature values, we would drop the instance out of our analysis. This resulted in reducing our dataset from 2436 instances (mood forms completed by 29 users) to 1, 438 complete ones, corresponding to 19 different users. Note that while we have sensor data from the phones (accelerometer and light sensor), and accelerometer data were quite predictive in the StudentLife study, we have not used them for the purposes of the current study, due to a large number of missing values exceeding a 6-hour window.

Baseline TEXT Features

All the texts (SMS and social media posts/messages) sent by a specific user over the past 24 hours before the completion of a mood form were concatenated in one 24-hour window. Focusing only on the English texts[2], the following commonly applied practices were performed: lowercasing, tokenisation (Gimpel et

[2]Language detection was performed using `https://pypi.python.org/pypi/langid`

al., 2011), replacement of usernames and URLs with placeholders, "usrmnt" and "urlink", respectively. We extracted the following textual features as potentially relevant to the mental state of users:

- **Ngrams**: We extracted *tfidf* representations of uni- and bi-grams, setting the max (min) document frequency to 99% (1%) and excluding all English stopwords, for noise reduction purposes.
- **Word embeddings**: We used the word embeddings created by (Tang et al., 2014), which have been used successfully before for the task of sentiment analysis, related to our problem. The unigrams of every text were matched against those vectors and seven functions were applied on every dimension of the resulting matrix (mean, median, min, max, stdev, first and third quartile).
- **Lexicons**: We employed several lexicons that have been effectively used in sentiment- or emotion-related works. Those were the Opinion Lexicon (Hu and Liu, 2004), NRC Hashtag, NRC Hashtag Emotion (Mohammad, 2012), Unigram and Bigram NRC Hashtag Sentiment and Sentiment 140 lexicons (Zhu et al., 2014), MaxDiff Twitter Sentiment Lexicon (Svetlana Kiritchenko and Mohammad, 2014), MSOL (Mohammad et al., 2009) and AFINN (Nielsen, 2011). For lexicons providing binary values (pos/neg), we counted the number of ngrams matching each of the positive and negative classes; for those lexicons with score values, we used the simple counts and the total summation of the corresponding scores from each ngram in the text matched against the lexicons.
- **Topics**: In order to better categorise the content that a subject has shared and to accommodate the sparse representations of the ngrams, we used the word clusters created by Preoţiuc-Pietro et al. (2015), which were based on word2vec representations of the most common keywords appearing on Twitter over a 2-month period. We measured the cosine similarity of the unigrams of every textual instance with each one of the 200 word clusters.
- **Other**: We extracted the following features related to the social activity level of a user: the number of SMS messages, Facebook posts, Facebook messages, Facebook images, twitter posts, twitter messages, and the total number of tokens and textual items (messages or posts) in the instance.

4.3 Experiments and Models

We applied five regression models, in order to predict each of the three target mood scores separately. All models were tested using 5-fold cross validation using the two sets of features (DA, TEXT) individually and in combination (ALL). Before feeding our features to the regression models, various transformations and normalisation techniques were tested. Those include:

- **The root transformation of the target labels**, often used in regression models to inflate the difference between lower values and stabilise the difference between higher scores[3].
- **Combinations of: (a) normalisation** (linear transformation of feature values to the $[-1, 1]$ range, based on the maximum/minimum value of the feature), **(b) standardisation** (zero mean, unit variance) or **(c) no transformation**.

Those transformations were performed on (i) a per-user basis (so that the feature values of different users become more comparable) and (ii) an overall basis (as a final transformation of all features from different users before applying our models). Notice that in the case of the per-user transformations, the model suffers from the cold-start problem, as it expects to have some past knowledge about the user, in order to predict her mood.

The algorithms that were tested under this setup were Linear Regression, LASSO, Random Forest for regression (RF), Support Vector Regression (SVR) and a multi-kernel SVR approach. The first four algorithms were chosen as widely accepted standards for regression problems, as well as for their diversity (two linear models, one with and one without feature selection, an ensemble of trees, a kernel-based method). Multi-kernel learning (MKL) was proposed in order to allow for a more advanced handling of the different data sources, by jointly learning different kernels, each optimised to a particular data source. For LASSO, different experiments with respect to the alpha parameter were tested (10^{-2}, ..., 10^{2}); for

[3]We also tried log-transformation but performance was lower.

RF we set the number of estimators to 200, after experimentation; for SVR we have used the Gaussian Kernel with varying kernel width and C values (all combinations of $\{10^{-2}, ..., 10^{2}\}$ for both)[4].

One drawback of SVR is the difficulty to interpret predictions and feature importance. Similarly performing algorithms, such as RF, can provide some indication of feature importance in the model learnt, but, when dealing with heterogeneous data sources, data source contribution is a lot less straightforward. For these reasons, we applied an MKL approach (Sonnenburg et al., 2006), which jointly learns an optimal combination of source-specific kernels. Formally, for a training set comprised of instances I and features S partitioned in subgroups $s \in S$, we apply a base kernel k per feature subgroup with some weight w, as follows: $f(\mathbf{x}) = \sum_{i \in I} \alpha_i \sum_{s \in S} w_s k_s(\mathbf{x}, \mathbf{x_i}) + b$ where the parameters α_i, the bias term b and the kernel weights are estimated by solving the optimisation problem:

$$\begin{aligned} \min \quad & \gamma - \sum_{i \in I} \alpha_i \\ \text{w.r.t.} \quad & \gamma \in R, \alpha \in R_+^{|I|} \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C \; \forall i, \;\; \sum_{i \in I} \alpha_i y_i = 0 \\ & \frac{1}{2} \sum_{i \in I} \sum_{j \in I} \alpha_i \alpha_j y_i y_j k_s(\mathbf{x}_i, \mathbf{x}_j) \leq \gamma \;\; \forall s \end{aligned}$$

We have opted for the L_2 norm to regularise the kernel weights. In order to compare our MKL approach with SVR, we selected one Gaussian kernel per feature set (9 kernels: 4 DA and 5 TEXT, for each of the feature sources defined in 4.2) and tuned the width of every kernel and the C parameter performing the same grid search as with SVR. This implies that we have used the same width for all nine kernels in every run. Further kernel selection and parameter optimisation techniques could be used, but those are out of the scope of the current work.

5 Evaluation and Results

We have used two standard measures for evaluating our models – the root mean squared error (RMSE, ϵ) and the coefficient of determination (R^2) . Those were selected in order to compare both the errors between the different approaches as well as the proportion of the variance that is predictable by them.

Table 1 presents the results obtained from our models. We provide separate results of the models for the two cases with respect to the per-user transformation of the features. Only the best transformation combinations are presented per model and the results obtained by Linear Regression are omitted, due to its poor performance. The feature transformation that was used is provided as an index.

In terms of comparing the three tasks (predicting each of the targets), our models can successfully capture much of the target variance in their predictions with respect to the well-being target. The lowest errors are observed with respect to the negative target (the comparison with the well-being case in terms of the error is not straight-forward, due to the larger scale that is used in WEMWBS). However, R^2 for this task is considerably lower, pointing to the low variance in each model's prediction.

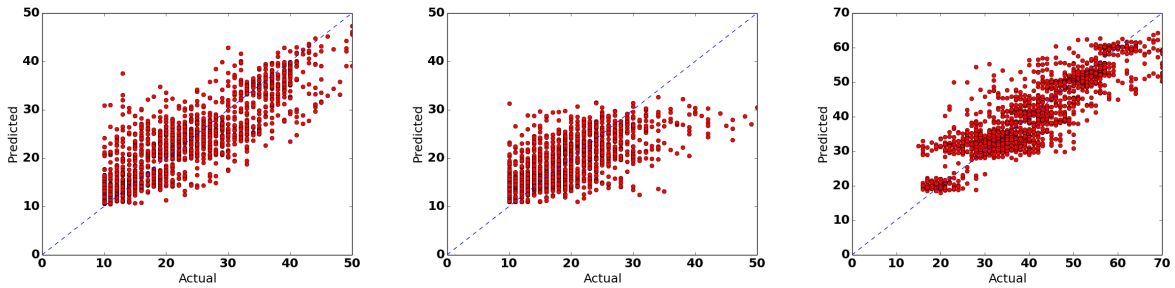
The task of user normalisation does not appear to have any significant effect when applied on the DA features for any task, implying that our models trained on DA features are user-independent and can generalise well. However, this is not the case for the TEXT features: for all algorithms and all tasks, the performance drops significantly when no user normalisation is applied. This is an interesting finding, pointing to future work on text-based user modelling, as it provides some evidence that population-wide analyses on mood prediction tasks that do not take it into account can be ineffective.

The comparison between the different algorithms illustrates that RF is the best in most experiments with respect to all target scores, achieving an R^2 of .76 in the best case (predicting the well-being score based on ALL features with user normalisation). To allow comparison with multi-modal affect our ρ

[4]Python sklearn library (`http://scikit-learn.org/stable/`) was used for the first three models and the Python interface for the Shogun library (`http://www.shogun-toolbox.org`) was used for SVR and MKL.

		Positive				Negative				Well-being			
		+User Norm		–User Norm		+User Norm		–User Norm		+User Norm		–User Norm	
		R^2	ϵ	R^2	ϵ	R^2	ϵ	R^2	ϵ	R^2	ϵ	R^2	ϵ
DA	LASSO	$_{n,n}$.31	8.24	$_{s}$.35	7.99	$_{n,n}$.11	6.71	$_{s}$.22	6.25	$_{n,n}$.30	10.51	$_{s}$.35	10.15
	RF	$_{s,s}$.69	5.55	$_{s}$.64	5.95	$_{s,s}$.43	5.38	$_{-}$.40	5.49	$_{s,s}$.75	6.33	$_{s}$.67	7.18
	SVR	$_{n,n}$.58	6.38	$_{n}$.60	6.27	$_{n,n}$.35	5.74	$_{n}$.36	5.69	$_{n,n}$.62	7.80	$_{n}$.62	7.77
	MKL	$_{n,n}$.61	6.15	$_{-}$.59	6.36	$_{n,n}$.38	5.60	$_{n}$.33	5.82	$_{n,s}$.65	7.43	$_{n}$.62	7.80
TEXT	LASSO	$_{n,n}$.53	6.80	$_{n}$.06	9.59	$_{n,n}$.23	6.23	$_{n}$.02	7.02	$_{n,n}$.55	8.46	$_{n}$.10	11.96
	RF	$_{-,s}$.70	5.42	$_{n}$.13	9.22	$_{-,s}$.45	5.26	$_{n}$.07	6.85	$_{s,s}$.74	6.36	$_{s}$.21	11.19
	SVR	$_{n,n}$.60	6.27	$_{n}$.11	9.31	$_{n,n}$.32	5.87	$_{n}$.06	6.88	$_{n,n}$.62	7.72	$_{n}$.19	11.30
	MKL	$_{n,n}$.62	6.08	$_{n}$.14	9.16	$_{n,n}$.36	5.69	$_{-}$.06	6.89	$_{n,n}$.65	7.43	$_{n}$.22	11.12
ALL	LASSO	$_{n,n}$.49	7.07	$_{n}$.31	8.20	$_{n,n}$.18	6.41	$_{n}$.20	6.33	$_{n,n}$.54	8.52	$_{n}$.38	9.92
	RF	$_{-,s}$.71	5.31	$_{n}$.63	6.00	$_{-,s}$.46	5.20	$_{s}$.40	5.51	$_{n,s}$.76	6.23	$_{-}$.68	7.12
	SVR	$_{n,n}$.60	6.27	$_{n}$.55	6.62	$_{n,n}$.34	5.76	$_{n}$.31	5.88	$_{n,n}$.62	7.75	$_{n}$.58	8.17
	MKL	$_{n,n}$.65	5.84	$_{n}$.61	6.14	$_{n,n}$.41	5.45	$_{n}$.36	5.67	$_{n,n}$.68	7.12	$_{n}$.64	7.58

Table 1: R^2 root mean squared error (ϵ) of the different models based on the three feature sets (DA, TEXT, ALL) and with respect to the three different ground truth scores (positive, negative, well-being). Values for both setups with respect to the user normalisation (with and without) are presented. The index used in the R^2 column indicates the (i) final and (ii) per-user normalisation of the best-performing setup (n for normalisation, s for standardisation, $-$ for none). Only the final normalisation method (i) is indicated in experiments performed without per-user normalisation.



(a) Positive (b) Negative (c) Well-being

Figure 3: Actual VS Predicted charts for the best performing algorithm (RF) on the three targets.

scores for RF for {positive, negative, well-being} are {.84, .68, .87} respectively, which is higher than for equivalent multi-modal tasks (Gupta et al., 2014). The charts in Figure 3 illustrate the corresponding predictions graphically. While our MKL does not outperform the RF, it achieves higher accuracy compared to SVR, showing that heterogeneous sources or feature sets can be effectively modelled via multiple kernels with a different weight, depending on their relative impact on the task. Importantly, this improvement comes without any kernel selection or dense parameter optimisation, which can be explored in future work. Also, comparing the results between MKL and RF in the cases without user normalisations shows that MKL is more robust to the cold-start problem for all three targets. This is important, as expecting to have past knowledge from any user is more challenging and resource greedy. A major advantage of MKL compared to SVR is the interpretation of the feature weights. By comparing the different kernel weights, we can see the contribution of each feature set separately. The bar charts in Figure 4 show the weights of each kernel (feature set), as determined by MKL, normalised to sum up to 1. For comparison purposes, we also present the corresponding weights from RF that were extracted by measuring every feature's importance across the trees and manually mapping those to the MKL's feature sets. For both the positive and the well-being targets, there are three TEXT feature sets that are preferred by both models (ngrams, word embeddings and topics), albeit with different weights. On the one hand, this points to a possibly weak feature engineering with respect to the DA data. On the

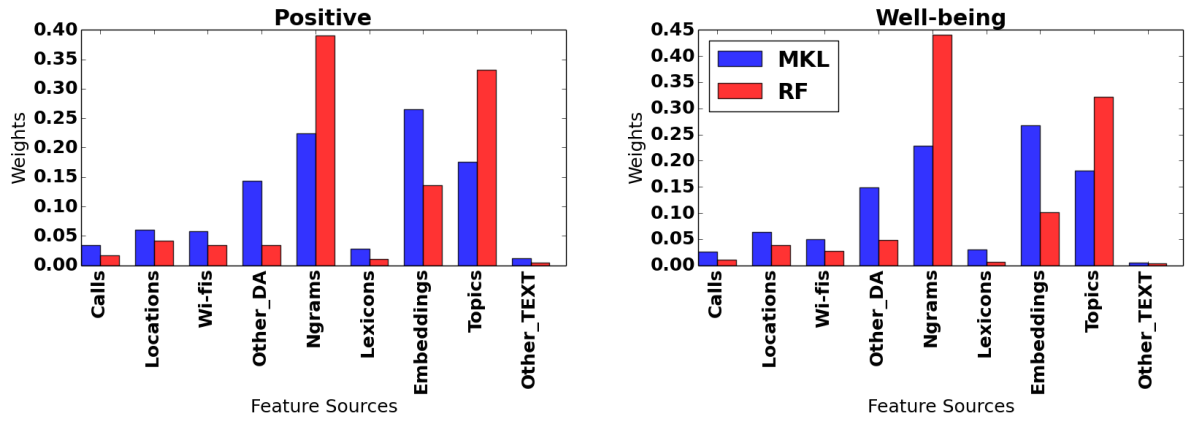


Figure 4: Feature set weights in RF (red) and MKL (blue) for the positive and the well-being targets, training on all features in the per-user normalisation approach.

other hand, it also explains the difference in accuracy between the two models, which can be highly reduced with further tuning of the MKL kernels and their parameters. Importantly though, the comparison between the feature weights of the two algorithms also explains the relatively small boost in accuracy that RF achieves when the DA features are incorporated in the TEXT-based RF (see Table 1), since the algorithm is relying much more on the TEXT features (12.7%, 18.0% and 12.7% of the feature weights come from DA sources with respect to positive, negative and well-being, compared to 29.5%, 28.9% and 28.7% for MKL). This means that MKL has much more potential in making use of heterogeneous sources compared to RF and further tuning of our MKL approach can provide an even more balanced kernel weighting for robustness purposes, while also increasing performance.

6 Conclusion and Future Work

We have presented a new real-world dataset consisting of heterogeneous, longitudinal and asynchronous textual and mobile phone use data. We have investigated different approaches for combining this heterogeneous data for daily predictions of mood scores and have proposed some strong baselines as well as a multi-kernel learning approach that learns a combination of source-specific representations, giving very promising results. We achieve a coefficient of determination (R^2) of $0.71 - 0.76$ and a ρ of $0.68 - 0.87$ which is higher than the state-of-the art in equivalent multi-modal tasks for affect.

In the future we aim to address the following: (a) optimise our MKL component and make better use of temporal granularity, by means of convolution and spectral mixture kernels (Lukasik and Cohn, 2016; Wilson et al., 2014) (b) look into the semantics of different locations and (c) address the challenge of missing data in one type of source, which is currently resulting in the loss of a large number of instances. Finally, we plan to apply our micro-level text-based model presented in this paper to the macro-level. Most approaches on predicting population-wide well-being indices based on online media have primarily relied on lists of pre-defined keywords, without using a per-user ground truth and ignoring the user modelling aspect. By applying our model on such a data stream, we will investigate whether we can build effective macro-level indicators for monitoring the well-being of a large population.

Acknowledgements

This work was supported by an IBM Faculty Award and a Warwick Research Development Fund Award to Dr Liakata and by the Engineering and Physical Sciences Research Council (grant EP/L016400/1) through the University of Warwick's Centre for Doctoral Training in Urban Science and Progress. The Centre for Urban Science and Progress at NYU has supported the project through funds for participant recruitment, server infrastructure and a graduate bursary (GRA). Finally we would like to thank our NYU student participants.

References

Mohammad Akbari, Liqiang Nie, and Tat-Seng Chua. 2015. amm: Towards adaptive ranking of multi-modal documents. *International Journal of Multimedia Information Retrieval*, 4(4):233–245.

Andrey Bogomolov, Bruno Lepri, and Fabio Pianesi. 2013. Happiness recognition from mobile phone data. In *Social Computing (SocialCom), 2013 International Conference on*, pages 790–795. IEEE.

Andrey Bogomolov, Bruno Lepri, Michela Ferron, Fabio Pianesi, and Alex Sandy Pentland. 2014. Daily stress recognition from mobile phone data, weather conditions and individual traits. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 477–486. ACM.

L. Canzian and M. Musolesi. 2015. Trajectories of Depression: Unobtrusive Monitoring of Depressive States by means of Smartphone Mobility Traces Analysis. In *ACM Ubicomp*.

J. Crawford and J. Henry. 2004. The Positive and Negative Affect Schedule (PANAS): Construct validity, measurement properties and normative data in a large non-clinical sample. *British Journal of Clinical Psychology*, 43.

Simon Dobrišek, Rok Gajšek, France Mihelič, Nikola Pavešić, and Vitomir Štruc. 2013. Towards efficient multi-modal emotion recognition. *Int J Adv Robotic Sy*, 10(53).

Peter Sheridan Dodds, Kameron Decker Harris, Isabel M. Kloumann, Catherine A. Bliss, and Christopher M. Danforth. 2011. Temporal patterns of happiness and information in a global social network: Hedonometrics and twitter. *PLoS ONE*, 6(12):e26752, 12.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics.

Rahul Gupta, Nikolaos Malandrakis, Bo Xiao, Tanaya Guha, Maarten Van Segbroeck, Matthew Black, Alexandros Potamianos, and Shrikanth Narayanan. 2014. Multimodal prediction of affective dimensions and depression in human-computer interactions. In *Proceedings of the 4th International Workshop on Audio/Visual Emotion Challenge*, pages 33–40. ACM.

Helen Herrman, Shekhar Saxena, Rob Moodie, et al. 2005. *Promoting mental health: concepts, emerging evidence, practice: a report of the World Health Organization, Department of Mental Health and Substance Abuse in collaboration with the Victorian Health Promotion Foundation and the University of Melbourne.* World Health Organization.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Natasha Jaques, Sara Taylor, Akane Sano, and Rosalind Picard. 2015. Multi-task, multi-kernel learning for estimating individual wellbeing. In *Proc. NIPS Workshop on Multimodal Machine Learning, Montreal, Quebec*.

V. Lampos, T. Welfare, R. Araya, and N. Christianini. 2013. Analysing Mood Patterns in the United Kingdom through Twitter Content. *arXiv: 1304.5507v*.

Thomas Lansdall-Welfare, Vasileios Lampos, and Nello Cristianini. 2012. Nowcasting the mood of the nation. *Significance*, 9(4):26–28.

N. Lathia, D. Quercia, and J. Crowcroft. 2012. The Hidden Image of the City: Sensing Community Well-Being from Urban Mobility. *Pervasive Computing*, 7319.

Hong Lu, Jun Yang, Zhigang Liu, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. 2010. The jigsaw continuous sensing engine for mobile phone applications. In *ACM Conference on Embedded Networked Sensor Systems (SenSys 2010)*.

Michal Lukasik and Trevor Cohn. 2016. Convolution kernels for discriminative learning from streaming text. In *Thirtieth AAAI Conference on Artificial Intelligence*. Citeseer.

Saif Mohammad, Cody Dunne, and Bonnie Dorr. 2009. Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 599–608. Association for Computational Linguistics.

Saif Mohammad. 2012. #emotional tweets. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 246–255, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

Finn Årup Nielsen. 2011. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*.

OECD. 2013. Hows life? 2013: Measuring well-being.

Veljko Pejovic, Neal Lathia, Cecilia Mascolo, and Mirco Musolesi. 2015. Mobile-based experience sampling for behaviour research. *arXiv preprint arXiv:1508.03725*.

Soujanya Poria, Erik Cambria, Amir Hussain, and Guang-Bin Huang. 2015. Towards an intelligent framework for multimodal affective data analysis. *Neural Networks*, 63:104–116.

Soujanya Poria, Erik Cambria, Newton Howard, Guang-Bin Huang, and Amir Hussain. 2016. Fusing audio, visual and textual clues for sentiment analysis from multimodal content. *Neurocomputing*, 174:50–59.

Daniel Preoţiuc-Pietro, Svitlana Volkova, Vasileios Lampos, Yoram Bachrach, and Nikolaos Aletras. 2015. Studying user income through language, behaviour and affect in social media. *PloS one*, 10(9):e0138717.

K. Rachuri, M. Musolesi, C. Mascolo, P. Renfrow, C. Longworth, and A. Aucinas. 2010. EmotionSense: A Mobile Phones based Adaptive Platform for Experimental Social Psychology Research. In *ACM Ubicomp*.

Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. 2006. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7(Jul):1531–1565.

Xiaodan Zhu Svetlana Kiritchenko and Saif M. Mohammad. 2014. Sentiment analysis of short informal texts. 50:723–762.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL (1)*, pages 1555–1565.

R. Tennant, L. Hiller, R. Fishwick, S. Platt, S. Joseph, S. Weich, J. Parkinson, J. Secker, and S. Stewart-Brown. 2007. The Warwick-Edinburgh Mental Well-being Scale (WEMWBS):development and UK validation. *Health and Quality of Life Outcomes*, 5.

D. Wagner, A. Rice, and A. Beresford. 2013. Device Analyzer: Understanding smartphone usage. In *International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*.

R. Wang and G. Harari. 2015. SmartGPA: How Smartphones Can Assess and Predict Academic Performance of College Students. In *ACM Ubicomp*.

Min Wang, Donglin Cao, Lingxiao Li, Shaozi Li, and Rongrong Ji. 2014. Microblog sentiment analysis based on cross-media bag-of-words model. In *Proceedings of international conference on internet multimedia computing and service*, page 76. ACM.

R. Wang, F. Chen, Z. Chen, T. Li, G. Harari, S. Tignor, X. Zhou, D. Ben-Zeev, and A. Campbell. 2015. StudentLife: Assessing Mental Health, Academic Performance and Behavioral Trends o College Students using Smartphones. In *ACM Ubicomp*.

D. Watson and L. Clark. 1988. Development and Validation of Brief Measures of Positive and Negative Affect: The PANAS Scales. *Journal of Personality and Social Psychology*, 54.

Andrew Wilson, Elad Gilboa, John P Cunningham, and Arye Nehorai. 2014. Fast kernel learning for multidimensional pattern extrapolation. In *Advances in Neural Information Processing Systems*, pages 3626–3634.

Matthias Wimmer, Björn Schuller, Dejan Arsic, Gerhard Rigoll, and Bernd Radig. 2008. Low-level fusion of audio, video feature for multi-modal emotion recognition. In *VISAPP (2)*, pages 145–151.

Xiaodan Zhu, Svetlana Kiritchenko, and Saif M Mohammad. 2014. Nrc-canada-2014: Recent improvements in the sentiment analysis of tweets. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 443–447. Citeseer.

Hashtag Recommendation with Topical Attention-Based LSTM

Yang Li†, Ting Liu†, Jing Jiang ‡, Liang Zhang †
†Center for Social Computing and Information Retrieval, Harbin Institute of Technology, China
‡School of Information Systems, Singapore Management University, Singapore
{yli,tliu,lzhang}@ir.hit.edu.cn
jingjiang@smu.edu.sg

Abstract

Microblogging services allow users to create hashtags to categorize their posts. In recent years, the task of recommending hashtags for microblogs has been given increasing attention. However, most of existing methods depend on hand-crafted features. Motivated by the successful use of long short-term memory (LSTM) for many natural language processing tasks, in this paper, we adopt LSTM to learn the representation of a microblog post. Observing that hashtags indicate the primary topics of microblog posts, we propose a novel attention-based LSTM model which incorporates topic modeling into the LSTM architecture through an attention mechanism. We evaluate our model using a large real-world dataset. Experimental results show that our model significantly outperforms various competitive baseline methods. Furthermore, the incorporation of topical attention mechanism gives more than 7.4% improvement in F1 score compared with standard LSTM method.

1 Introduction

Over the past few years, microblogging has experienced tremendous success and become very important as both a social network and a news media. There is a significant amount of information generated every day. To facilitate the navigation in the deluge of information, microblogging services allow users to insert hashtags starting with the "#" symbol (e.g., #followfriday) into their posts to indicate the context or the core idea. In this way, hashtags help bring together relevant microblogs on a particular topic or event and enhance information diffusion in microblog services. It has been proven that hashtags are important for many applications in microblogs (Efron, 2010; Bandyopadhyay et al., 2012; Davidov et al., 2010; Wang et al., 2011; Li et al., 2015). However, not all microblog posts have hashtags created by their authors. Reported in a recent study, only about 11% of tweets were annotated with one or more hashtags (Hong et al., 2012). Hence, the task of recommending hashtags for microblogs has become an important research topic and attracted much attention in recent years.

Existing approaches to hashtag recommendation range from classification and collaborative filtering to probabilistic models such as naive Bayes and topic models. Most of these methods depend on sparse lexical features including bag-of-word (BoW) models and exquisitely designed patterns. However, feature engineering is labor-intensive and the *sparse* and *discrete* features cannot effectively encode semantic and syntactic information of words. On the other hand, neural models recently have shown great potential for learning effective representations and delivered state-of-the-art performance on various natural language processing tasks (Cho et al., 2014; Tang et al., 2015; Rush et al., 2015). Among these methods, the long short-term memory (LSTM), a variant of recurrent neural network (RNN), is widely adopted due to its capability of capturing long-term dependencies in learning sequential representations (Hochreiter and Schmidhuber, 1997; Gers et al., 2000; Palangi et al., 2016).

In this work, we model the hashtag recommendation task as a multi-class classification problem. A typical approach is to adopt LSTM to learn the representation of a microblog post and then perform text classification based on this representation. However, a potential issue with this approach is that all the

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3019–3029, Osaka, Japan, December 11-17 2016.

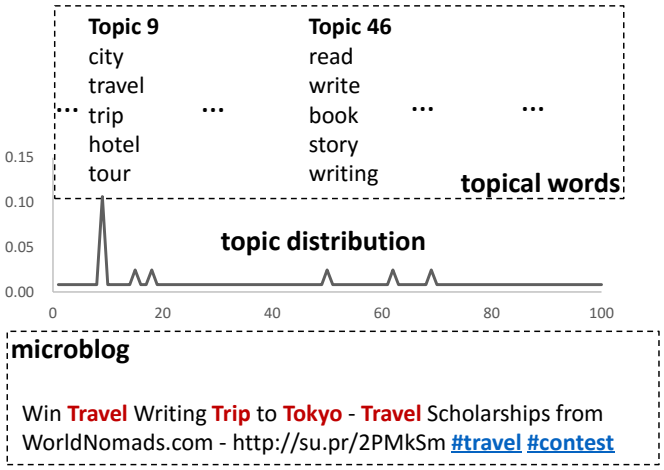


Figure 1: Illustration of hashtags of a microblog post and its topic distribution. The example post has a high probability in a travel topic (Topic 9). Words that are related to the topic (marked in red bold) can be selected through the topic distribution of the post. Using these words, we can predict its hashtag #travel.

necessary information of the input post has to be compressed into a fixed-length vector. This may make it difficult to cope with long sentences (Bahdanau et al., 2015). One possible solution is to perform an average pooling operation over the hidden vectors of LSTM (Boureau et al., 2011), but not all words in a microblog post contribute equally for hashtag recommendation. Inspired by the success of attention mechanism in computer vision and natural language processing (Mnih et al., 2014; Bahdanau et al., 2015; Luong et al., 2015), we investigate the use of attention mechanism to automatically capture the most relevant words in a microblog to the recommendation task. Furthermore, it has been observed that most hashtags indicate the topics of a microblog (Ding et al., 2012; Godin et al., 2013), as illustrated in Figure 1. To this end, we propose a novel attention-based LSTM model which incorporates LDA topics of microblogs into the LSTM architecture through an attention mechanism. By modeling the interactions between the words and the global topics, our model can learn effective representations of microblogs for hashtag recommendation. Experimental results on a large real microblogging dataset show that our model significantly outperforms various competitive baseline methods. Furthermore, the incorporation of topical attention mechanism gives more than 7.4% improvement in F1 score compared with standard LSTM method.

The main contributions of this paper can be summarized as follows:

- We thoroughly investigate several neural attention-based models for hashtag recommendation.
- We propose a novel attention-based LSTM model that incorporates topics of microblog posts into the LSTM architecture through an attention mechanism. Experiments on data from a real microblogging service show that our model achieves significantly better performance than various state-of-the-art methods.

2 Background

Before going to the details of our method, we provide some background on two topics relevant to our work: the attention mechanism and Latent Dirichlet Allocation (LDA).

2.1 Attention Mechanism

Attention-based models have demonstrated success in a wide range of NLP tasks including sentence summarization (Rush et al., 2015), reading comprehension (Hermann et al., 2015) and text entailment (Rocktäschel et al., 2016; Wang and Jiang, 2016). The basic idea of the attention mechanism is that it assigns a weight to each position in a lower-level of the neural network when computing an upper-level

representation (Bahdanau et al., 2015; Luong et al., 2015). Bahdanau et al. (2015) made the first attempt to use an attention-based neural machine translation (NMT) approach to jointly translate and align words. The model is based on the basic encoder-decoder model (Cho et al., 2014). Differently, it encodes the input sentence into a sequence of vectors and chooses a subset of these vectors adaptively through the attention mechanism while generating the translation.

In this work, we adopt the attention mechanism to scan input microblog posts and select key words to hashtag recommendtaion. Motivated by Bahdanau et al. (2015), we first investigate a vanilla attention-based LSTM model, which is referred to as VAB-LSTM in Section 4.2.1. In VAB-LSTM, we use the last hidden vector from the LSTM that processes a post as the global representation of that post and incorporate attentions to measure the interactions between each word and the global representation. Then we further compare it with our proposed topical attention-based LSTM model.

2.2 Latent Dirichlet Allocation (LDA)

Topic models have been a powerful technique for finding useful structures in a collection of documents. Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is a well-developed and widely-used topic model for inferring the semantic meaning of documents through a set of representative words (topics). It models a document as a mixture of latent topics. In LDA, each document of the corpus is assumed to have a distribution over K topics, where the discrete topic distributions are drawn from a symmetric Dirichlet distribution. The high probability words in each distribution gives us a way of understanding the contents of the corpus at a very high level.

Given a collection of microblog posts, LDA is able to learn a sparse topic representation for each post. The topic representation is viewed as a kind of global semantic information of a post, which we can utilize to learn the interactions between each words and the whole microblog post.

3 The Approach

In this section, we will present our proposed model for hashtag recommendation. We formulate the hashtag recommendation task as a multi-class classification problem. It has been observed that hashtags indicate the primary topics of microblog posts (Ding et al., 2012; Godin et al., 2013). To incorporate the topics of microblogs, we take into account the attention mechanism and develop a novel Topical Attention-Based LSTM model, or TAB-LSTM for short. The basic idea of TAB-LSTM is to combine local hidden representations with global topic vectors through an attention mechanism. We believe that in this way our model can capture the importance of different local words according to the global topics of a microblog post.

Our overall model is illustrated in Figure 2. The model mainly consists of three parts, namely, LSTM based sequence encoder, topic modeling, and topical attention. In the rest of this section, we will present each of these three parts in detail. A basis of all three parts is that each word is represented as a low dimensional, continuous and real-valued vector, also known as word embedding (Bengio et al., 2003; Mikolov et al., 2013). All the word vectors are stacked in a word embedding matrix $L_w \in \mathbb{R}^{dim \times |V|}$, where dim is the dimension of word vector and $|V|$ is vocabulary size. We pre-train the values of word vectors from text corpus with embedding learning algorithms to make better use of semantic and grammatical associations of words (Mikolov et al., 2013). Given an input microblog s, we take the embeddings $\mathbf{x}_t \in \mathbb{R}^{dim}$ for each word in the microblog to obtain the first layer. Hence, a microblog post of length N is represented with $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$.

3.1 LSTM based Sequence Encoder

LSTM is special form of recurrent neural networks (RNNs), widely used to model sequence data. LSTM uses input gate, forget gate and output gate vectors at each position to control the passing of information along the sequence and thus improves the modeling of long-range dependencies (Hochreiter and Schmidhuber, 1997).

Given a microblog $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$, LSTM processes it sequentially. For each position $\mathbf{x}_t$, given the previous output $\mathbf{h}_{t-1}$ and cell state $\mathbf{c}_{t-1}$, an LSTM cell use the input gate $\mathbf{i}_t$, the forget gate $\mathbf{f}_t$ and

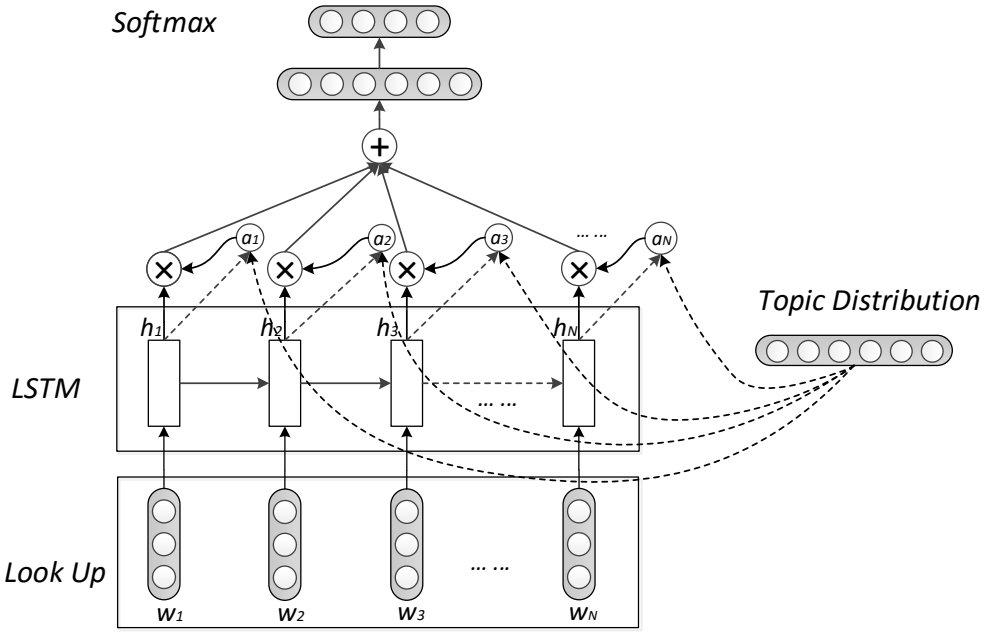


Figure 2: The graphical illustration of the proposed topical attention-based LSTM model (TAB-LSTM).

the output gate $\mathbf{o}_t$ together to generate the next output $\mathbf{h}_t$ and cell state $\mathbf{c}_t$. The transition equations of LSTM are defined as follows:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}^i\mathbf{x}_t + \mathbf{U}^i\mathbf{h}_{t-1} + \mathbf{b}^i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}^f\mathbf{x}_t + \mathbf{U}^f\mathbf{h}_{t-1} + \mathbf{b}^f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}^o\mathbf{x}_t + \mathbf{U}^o\mathbf{h}_{t-1} + \mathbf{b}^o) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}^c\mathbf{x}_t + \mathbf{U}^c\mathbf{h}_{t-1} + \mathbf{b}^c) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{1}$$

where $\odot$ stands for element-wise multiplication, σ is the sigmoid function, all $\mathbf{W} \in \mathbb{R}^{d\times l}$ and $\mathbf{U} \in \mathbb{R}^{d\times d}$ are weight matrices, all $\mathbf{b} \in \mathbb{R}^d$ are bias vectors.

The output of LSTM layer is a sequence of hidden vectors $[\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N]$. Each annotation $\mathbf{h}_t$ contains information about the whole input microblog with a strong focus on the parts surrounding the t-th word of the input microblog.

3.2 Topic Modeling

In TAB-LSTM, we propose an topical attention to introduce a series of attention-weighted combinations of these hidden vectors using the external topical distribution. We use LDA to learn the topic structures of microblogs and the model is trained offline. Specifically, given a set of microblog posts $\mathcal{S}$, where each post $s \in \mathcal{S}$ contains N_s words $\{w_{s,1}, w_{s,2}, ..., w_{s,N_s}\}$, LDA makes the following assumptions. There exist K topics, each associated with a multinomial word distribution φ_k. Each post has a topic distribution θ_s in the K-dimensional topic space. Each word in a microblog post has a hidden topic label drawn from the post's topic distribution. Formally, the generative process of LDA is described as follows:

- For each topic $k = 1, ..., K$, draw $\varphi_k \sim Dir(\beta)$
- For each microblog post $s \in \mathcal{S}$, draw $\theta_s \sim Dir(\alpha)$
  - For each word $w_{s,n}$, draw $z_{s,n} \sim Multi(\theta_s)$ and $w_{s,n} \sim Multi(\phi_{z_{s,n}})$

where α and β are parameters of the Dirichlet priors, θ_s is the topic distribution we will incorporate in our model.

3.3 Topical Attention

Taking all hidden states $[\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N]$ and the external topic vector $\theta_\mathbf{s} \in \mathbb{R}^{K\times 1}$, the topical attention layer outputs a continuous context vector $vec \in \mathbb{R}^{d\times 1}$ for each microblog post s. The output vector is computed as a weighted sum of each hidden state $\mathbf{h}_j$:

$$vec = \sum_{j=1}^{N} a_j \mathbf{h}_j \tag{2}$$

where d is the hidden dimension of LSTM, $a_j \in [0, 1]$ is the attention weight of $\mathbf{h}_j$ and $\sum_j a_j = 1$.

Next, we will introduce how we obtain $[a_1, a_2, ..., a_N]$ in detail. Specifically, for each $\mathbf{h}_j$, we use the following equation to compute scores on how well the inputs around position j and the topic distribution θ_s match:

$$g_j = v_a^\top \tanh(\mathbf{W^a}\theta_\mathbf{s} + \mathbf{U^a}\mathbf{h}_j) \tag{3}$$

where K is the number of topics, $\mathbf{W^a} \in \mathbb{R}^{d\times K}$, $\mathbf{U^a} \in \mathbb{R}^{d\times d}$ and $v_a \in \mathbb{R}^{d\times 1}$ are the weight matrices. After obtaining $[g_1, g_2, ..., g_N]$, we feed them to a $softmax$ function to calculate the final weight scores $[a_1, a_2, ..., a_N]$.

Finally, we use the output from the topical attention layer as the embedding of the microblog from our deep neural network. We feed the output vector vec to a linear layer whose output length is the number of hashtags. Then a softmax layer is added to output the probability distributions of all candidate hashtags. The softmax function is calculated as follows, where M is the number of hashtag categories:

$$softmax(m_i) = \frac{exp(m_i)}{\sum_{i'=1}^{M} exp(m_{i'})} \tag{4}$$

3.4 Model Training

We model hashtag recommendation as a multi-class classification task. We train our model in a supervised manner by minimizing the cross-entropy error of the hashtag classification. The loss function is given below:

$$\mathcal{J} = -\sum_{s\in S} \sum_{t\in tags(s)} \log p(t|s) \tag{5}$$

where S stands for all training instances, $tags(s)$ is the hashtag collection for microblog s.

4 Experiments

We apply the proposed method to the task of hashtag recommendation to evaluate the performance. In this section, we first describe our dataset and experimental settings, then the results and analysis.

4.1 Dataset

Our dataset is constructed from a large Twitter dataset which spans the second half of 2009 (Yang and Leskovec, 2011). We collect a dataset with 185,391,742 tweets from October to December 2009. Among them, there are 16,744,189 tweets including hashtags annotated by users. We randomly select 500,000 tweets as training set, 50,000 tweets as development and test set respectively. The statistics of our dataset is shown in Table 1.

# Tweets	# Hashtags	Vocabulary Size	Nt(avg)
600,000	27,720	337,245	1.308

Table 1: Statistics of the dataset, Nt(avg) is the average number of hashtags in the dataset.

4.2 Experimental Settings

4.2.1 Baseline Methods

For comparison, we consider the following baseline methods:

• **LDA**: We use the LDA based method proposed by Krestel et al. (2009) to recommend hashtags.

• **SVM**: We build a multi-class SVM classification model (Hearst et al., 1998) with LibSVM. The feature we use are word embedding features with 300 dimension. We believe that comparing to Bag-of-words, word embedding features can capture deep semantic information of the microblog posts. SVM parameters are chosen by grid search on the development set.

• **TTM**: The topical translation model is proposed by Ding et al. (2013) for hashtag extraction. We implement their method for evaluating it on the corpus constructed in this work.

• **LSTM**: We regard the last hidden vector from LSTM as the microblog representation. Then we feed it to a linear layer whose output length is the number of hashtags. Finally, a softmax layer is added to output the probability distributions of all candidate hashtags.

We also compare two degenerate versions of our model TAB-LSTM as follows.

• **AVG-LSTM**: We perform an average pooling operation on the hidden vectors at each position of LSTM that processes a post, and use the result as the representation of that post.

• **VAB-LSTM**: In this model, we use the last hidden vector from the LSTM that processes a post as the global representation of that post and incorporate attentions to measure the interactions between each word and the global representation. This method is similar to our model except that we replace the topic distribution θ_s with the last hidden vector h_N in Equation (3).

4.2.2 Experimental Setup

We perform hashtag recommendation as follows. Suppose given an unlabeled dataset, we first train our model on training data, and save the model which has the best performance on the validate dataset. For the microblog of the unlabeled data, we will encode the microblog post through our proposed model. We train four types of neural models including LSTM, AVG-LSTM, VAB-LSTM and our proposed model TAB-LSTM. For each of the above models, the sentences of length is up to 40 words. We set the dimension of all the hidden states of the LSTMs to be 500. We use a minibatch stochastic gradient descent (SGD) algorithm together with the Adam method to train each model (Kingma and Ba, 2014). The hyperparameters $\beta 1$ is set to 0.9 and $\beta 2$ set to 0.999 for optimization. The learning rate is set to be 0.001. The batch size is set to be 100. For TAB-LSTM, we tested with different numbers of LDA topic size K and found $K = 100$ is an optimal setting.

For both our models and the baseline methods, we use the validation data to tune the hyperparameters, we report the results of the test data in the same setting of hyperparameters. Furthermore, the word embeddings used in all methods are pre-trained from the original twitter data released by (Yang and Leskovec, 2011) with the word2vec toolkit (Mikolov et al., 2013).

We use hashtags annotated by users as the golden set. To evaluate the performance, we use precision (P), recall (R), and F1-score (F) as the evaluation metrics. The same settings are adopted by previous work (Ding et al., 2012; Ding et al., 2013; Gong et al., 2015).

4.3 Comparison to Other Methods

In Table 2, we compare the results of our method and the state-of-the-art discriminative and generative methods on the dataset. TAB-LSTM denotes our proposed model. We have the following observations. (1) First of all, SVM performs much better than LDA, showing that the embedding features capture more semantic information than bag-of-words (BoW). (2) Both TAB-LSTM and the degenerate models significantly outperform the baseline methods LDA, SVM and TTM. The results demonstrate that the neural network can achieve better performance on this task. (3) If we compare TAB-LSTM with LSTM and AVG-LSTM, we can see that TAB-LSTM improves the F1-score by nearly 7.4% in the same setting of parameters, showing that incorporating attention mechanism is useful. (4) Using topical attention, TAB-LSTM outperforms VAB-LSTM, which shows the effectiveness of topic information for this task.

Methods	Precision	Recall	F1-score
LDA	0.098	0.078	0.087
SVM	0.238	0.203	0.219
TTM	0.324	0.280	0.300
LSTM	0.470	0.404	0.434
AVG-LSTM	0.472	0.405	0.436
VAB-LSTM	0.489	0.419	0.452
TAB-LSTM	**0.503**	**0.435**	**0.467**

Table 2: Evaluation results of different methods for hashtag recommendation. The dimension of word embeddings is set to be 300 for all methods. All improvements obtained by TAB-LSTM over other methods are statistically significant within a 0.99 confidence interval using the t-test.

Considering that many microblog posts have more than one hashtags, we also evaluate the top k results of different methods. Figure 3 shows the precision, recall, and F1 curves of LDA, SVM, TTM, LSTM and TAB-LSTM on the test data. Each point of a curve represents the extraction of a different number of hashtags, ranging from 1 to 5. From Figure 3, we can see although the precision and F1-score of TAB-LSTM decreases when the number of hashtags is larger, the performance of TAB-LSTM still outperforms the other methods. In addition, the relative improvement on extracting only one hashtag is higher than that on more than one hashtags, showing that it is more difficult to recommend hashtags for a microblog post with more than one hashtags.

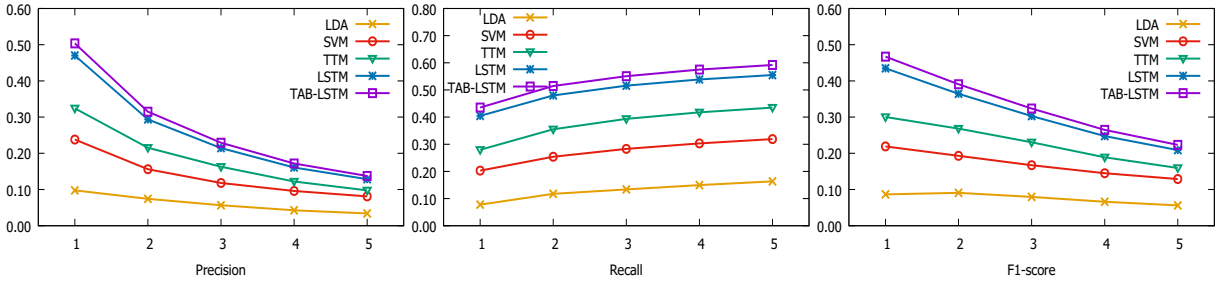


Figure 3: Precision, Recall and F1 with recommended hashtags range from 1 to 5.

4.4 Parameter Sensitive Analysis

We further investigate the effect of hyperparameters to the performance. First, we vary the values of topic size K while fixing the other parameters. We've tried various settings with $K = 50, 100, 150, 200$ when training the topic models with LDA. Results in Table 3 show that the best results are achieved when K is larger than 100.

Methods	Precision	Recall	F1-score
Attn50	0.492	0.422	0.454
Attn100	0.503	0.435	0.467
Attn150	0.501	0.432	0.464
Attn200	0.499	0.431	0.463

Table 3: Precision, Recall and F1 of TAB-LSTM with different number of topics when the dimension of word vectors is set to be 300.

It is well accepted that a good word embedding is crucial to composing a powerful text representation at a higher level. Next, we would like to study the effects of different word embeddings. Table 4 shows

the precision, recall and F1-score when we vary the dimension of word embeddings. We find a larger dimension of word embedding is more effective for this task.

Methods	Precision	Recall	F1-score
Emb50	0.470	0.403	0.434
Emb100	0.487	0.419	0.450
Emb200	0.495	0.425	0.457
Emb300	0.503	0.435	0.467

Table 4: Precision, Recall and F1 of TAB-LSTM with different dimension of word embeddings when the number of topics is 100.

4.5 Qualitative Analysis

We also perform qualitative analysis of our results. In Figure 4, we compare the attention heat maps learned by TAB-LSTM and VAB-LSTM of two example microblog posts. In the first example, hashtag #H1N1 is correctly recommended by TAB-LSTM because the word *H1N1* is selected by the topic of this post, while in the case of VAB-LSTM, H1N1 is not selected. In the second example, both hashtags are correctly predicted by TAB-LSTM, while VAB-LSTM missed the word "ff", which is short for #followfriday.

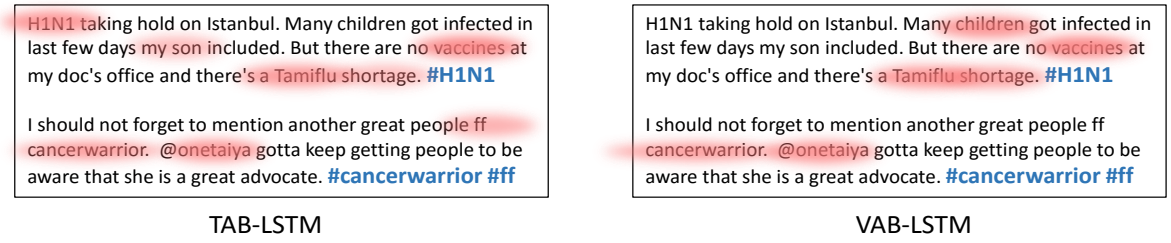


Figure 4: Attention heat maps for two example microblog posts.

5 Related Work

There has been a variety of work proposed for hashtag recommendation in the past few years. Zangerle et al. (2011) exploit the similarity between tweets. For a given tweet, they first retrieve its similar tweets and then rank the hashtags by their usage on the most similar tweets. Sedhai and Sun (2014) formulate hashtag recommendation task as a learning to rank problem. They represent each candidate hashtag as a feature vector and use pairwise learning to rank method to find the top ranked hashtags from the candidate set. Mazzia and Juett (2009) apply a Naive Bayes model to estimate the maximum a posteriori probability of each hashtag class given the words of the tweet. Furthermore, Godin et al. (2013) propose to incorporate topic models to learn the underlying topic assignment of language classified tweets, and suggest hashtags to a tweet based on the topic distribution. Under the assumption "hashtags and tweets are parallel description of a resource" that proposed by Liu et al. (2011), Ding et al. try to integrate latent topical information into translation model. The model uses topic-specific word trigger to bridge the vocabulary gap between the words in tweets and hashtags (Ding et al., 2012; Ding et al., 2013).

Most of the works mentioned above are based on textual information. There have also been some attempts that combine text with other types of data. Kywe et al. propose a collaborative filtering model to incorporate user preferences in hashtag recommendation (Kywe et al., 2012). Besides that, Zhang et al. (2014) and Ma et al. (2014) try to incorporate temporal information. Gong et al. (2015) propose to model type of hashtag as a hidden variable into their DPMM (Dirichlet Process Mixture Models) based method.

More recently, Gong and Zhang (2016) propose an attention-based convolutional neural network, which incorporates a local attention channel and global channel for hashtag recommendation. However, to the best of our knowledge, there is no work yet on employing both topic models and deep neural networks for this task.

6 Conclusion

In this paper, we investigated a novel topical attention-based LSTM model for the task of hashtag recommendation. We adopted the architecture of LSTM to avoid hand-crafted features. Our model incorporates topic modeling into the LSTM architecture through an attention mechanism and takes over the advantages of the both. Through evaluations run on a large dataset from Twitter, we have demonstrated that the proposed method outperforms competitive baseline methods effectively.

The present work does not consider the use of other types of data in microblogs for hashtag recommendation. In the future, other types of data such as user information and temporal information can be incorporated into the model. We will also consider using alternative topic models which are particularly designed for short microblog texts such as Twitter-LDA (Zhao et al., 2011).

7 Acknowledgements

We thank the anonymous reviewers for their constructive comments, and gratefully acknowledge the support of the National Basic Research Program (973 Program) of China via Grant 2014CB340503, the National Natural Science Foundation of China (NSFC) via Grant 61472107 and 71532004. Corresponding author: Ting Liu, E-mail: tliu@ir.hit.edu.cn.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*.

Ayan Bandyopadhyay, Kripabandhu Ghosh, Prasenjit Majumder, and Mandar Mitra. 2012. Query expansion for microblog retrieval. *International Journal of Web Science*, 1(4):368–380.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *journal of machine learning research*, 3(Feb):1137–1155.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Y-Lan Boureau, Nicolas Le Roux, Francis Bach, Jean Ponce, and Yann LeCun. 2011. Ask the locals: multi-way local pooling for image recognition. In *2011 International Conference on Computer Vision*, pages 2651–2658. IEEE.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd international conference on computational linguistics: posters*, pages 241–249. Association for Computational Linguistics.

Zhuoye Ding, Qi Zhang, and Xuanjing Huang. 2012. Automatic hashtag recommendation for microblogs using topic-specific translation model. In *Proceedings of COLING 2012: Posters*, pages 265–274, Mumbai, India, December. The COLING 2012 Organizing Committee.

Zhuoye Ding, Xipeng Qiu, Qi Zhang, and Xuanjing Huang. 2013. Learning topical translation model for microblog hashtag suggestion. In *IJCAI*. Citeseer.

Miles Efron. 2010. Hashtag retrieval in a microblogging environment. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 787–788. ACM.

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

Fréderic Godin, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. 2013. Using topic models for twitter hashtag recommendation. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 593–596. International World Wide Web Conferences Steering Committee.

Yuyun Gong and Qi Zhang. 2016. Hashtag recommendation using attention-based convolutional neural network. In *IJCAI*.

Yeyun Gong, Qi Zhang, and Xuanjing Huang. 2015. Hashtag recommendation using dirichlet process mixture models incorporating types of hashtags. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 401–410, Lisbon, Portugal, September. Association for Computational Linguistics.

Marti A. Hearst, Susan T Dumais, Edgar Osman, John Platt, and Bernhard Scholkopf. 1998. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

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

Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J. Smola, and Kostas Tsioutsiouliklis. 2012. Discovering geographical topics in the twitter stream. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 769–778, New York, NY, USA. ACM.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. 2009. Latent dirichlet allocation for tag recommendation. In *Proceedings of the third ACM conference on Recommender systems*, pages 61–68. ACM.

Su Mon Kywe, Tuan-Anh Hoang, Ee-Peng Lim, and Feida Zhu. 2012. On recommending hashtags in twitter networks. In *Social Informatics*, pages 337–350. Springer.

Yang Li, Jing Jiang, Ting Liu, and Xiaofei Sun. 2015. Personalized microtopic recommendation with rich information. In *Social Media Processing: 4th National Conference, SMP 2015, Guangzhou, China*, pages 1–14. Springer.

Zhiyuan Liu, Xinxiong Chen, and Maosong Sun. 2011. A simple word trigger method for social tag suggestion. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1577–1588, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. pages 1412–1421, September.

Zongyang Ma, Aixin Sun, Quan Yuan, and Gao Cong. 2014. Tagging your tweets: A probabilistic modeling of hashtag annotation in twitter. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 999–1008. ACM.

Allie Mazzia and James Juett. 2009. Suggesting hashtags on twitter. *EECS 545m, Machine Learning, Computer Science and Engineering, University of Michigan*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(4):694–707.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. *ICLR*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. pages 379–389, September.

Surendra Sedhai and Aixin Sun. 2014. Hashtag recommendation for hyperlinked tweets. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 831–834. ACM.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1442–1451, San Diego, California, June. Association for Computational Linguistics.

Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. 2011. Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1031–1040. ACM.

Jaewon Yang and Jure Leskovec. 2011. Patterns of temporal variation in online media. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 177–186. ACM.

Eva Zangerle, Wolfgang Gassler, and Gunther Specht. 2011. Recommending#-tags in twitter. In *Proceedings of the Workshop on Semantic Adaptive Social Web (SASWeb 2011). CEUR Workshop Proceedings*, volume 730, pages 67–78.

Qi Zhang, Yeyun Gong, Xuyang Sun, and Xuanjing Huang. 2014. Time-aware personalized hashtag recommendation on social media. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 203–212, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. 2011. Comparing twitter and traditional media using topic models. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval*, ECIR'11, pages 338–349, Berlin, Heidelberg. Springer-Verlag.

Better call Saul: Flexible Programming for Learning and Inference in NLP

Parisa Kordjamshidi[*†] **Daniel Khashabi**[‡] **Christos Christodoulopoulos**[‡]
Bhargav Mangipudi[‡] **Sameer Singh**[§] **Dan Roth**[‡]
[†] Tulane University [‡] University of Illinois at Urbana-Champaign [§] University of California, Irvine
[†] pkordjam@tulane.edu [§] sameer@uci.edu
[‡] {khashab2,christod,mangipu2,danr}@illinois.edu

Abstract

We present a novel way for designing complex joint inference and learning models using *Saul* (Kordjamshidi et al., 2015), a recently-introduced declarative learning-based programming language (DeLBP). We enrich *Saul* with components that are necessary for a broad range of learning based Natural Language Processing tasks at various levels of granularity. We illustrate these advances using three different, well-known NLP problems, and show how these generic learning and inference modules can directly exploit *Saul*'s graph-based data representation. These properties allow the programmer to easily switch between different model formulations and configurations, and consider various kinds of dependencies and correlations among variables of interest with minimal programming effort. We argue that *Saul* provides an extremely useful paradigm both for the design of advanced NLP systems and for supporting advanced research in NLP.

1 Introduction

Most of the problems in natural language processing domain can be viewed as a mapping from an input structure to an output structure that represents lexical, syntactical or semantic aspects of the text. For example, Part-of-Speech (POS) tagging provides a syntactic representation, Semantic Role Labeling (SRL) is a (shallow) semantic representation, and all variations of information extraction such as Entity-Relation (ER) extraction provide a lightweight semantic representation of unstructured textual data. Even though the text data looks initially unstructured, solving such problems requires one to consider various kinds of relationships between linguistic components at multiple levels of granularity.

However, designing machine learning models that deal with structured representations is a challenging problem. Using such representations is challenging in that designing a new learning model or tackling a new task requires a significant amount of task-specific and model-specific programming effort for learning and inference. Additionally, global background knowledge in these models is usually hard-coded, and changing or augmenting it is extremely time-consuming. There are several formal frameworks (Tsochantaridis et al., 2004; Chang et al., 2013) for training structured output models but these provide no generic solution for doing inference on arbitrary structures. Likewise, probabilistic programming languages (Pfeffer (2009), McCallum et al. (2009) *inter alia*) try to provide generic probabilistic solutions to arbitrary inference problems but using them in the context of training arbitrary structured output prediction models for real world problems is a challenge; in addition, encoding structured features and high-level background knowledge necessary for these problems is not a component of those frameworks.

In this paper we build on the abstractions made available in *Saul* programming language (Kordjamshidi et al., 2015), in order to create a unified and flexible machine learning programming framework using the generic Constrained-Conditional-Model (CCM) paradigm (Chang et al., 2012).

Specifically, we build upon *Saul*'s graph-based data representation and enrich it with primitive structures and *sensors* for the NLP domain that facilitate operating at arbitrary levels of 'globality' for learning and inference. For example, a programmer can easily decide if he or she needs to operate at document,

[*] Most of the work was performed at the University of Illinois.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3030–3040, Osaka, Japan, December 11-17 2016.

sentence, or phrase level. In other words, without any programming effort the user can specify at which level of granularity the context should be considered, and how 'global' should learning and inference be. This ability also better supports the declarative specification of the structured features and is useful in a wide range of tasks that involve mapping the language to its syntactic or semantic structure. Using this new framework, we show how one can design structured output prediction models in an easy and flexible way for several well-known and challenging NLP tasks and achieve comparable results to the existing state-of-the-art models.

The paper is structured as follows: Section 2, describes the general formulation used for structured output prediction and its possible configurations. Section 3 explains how the structured input and output spaces are represented as a graph, the form of the objective and the way arbitrary dependencies can be represented in *Saul*. In Section 4, various model configurations are presented for our case study tasks, alongside experimental results. Section 5 provides a brief overview of related work. Section 6 concludes.

2 Structured Output Prediction

Punyakanok et al. (2005) describe three fundamentally different and high level solutions towards designing structured output prediction models,

(a) **Learning Only (LO):** Local classifiers are trained to predict each output component independently;

(b) **Learning plus inference (L+I):** Training is performed locally as in the LO model, but global constraints/dependencies among components are imposed during prediction (Chang et al., 2012). In the context of training probabilistic graphical models this is referred to as *piecewise training* (Sutton and McCallum, 2009);

(c) **Inference based training (IBT):** Here, during the training phase, predictions are made globally so that constraints and dependencies among the output variables are incorporated into the training process (Collins, 2004; Taskar et al., 2002; Tsochantaridis et al., 2004).

When training structured output models there is a spectrum of *configurations* (model compositions) between the two extremes – only local training as in the LO and L+I schemes and the full global training as in the IBT scheme (Samdani and Roth, 2012). The key here is choosing the best decomposition of the variables/structures which is largely an empirical question; having an expressive and flexible machinery for modeling the data and for learning from it is thus useful and eases in designing, assessing, decomposing and improving the models (Kordjamshidi and Moens, 2013). With this as motivation, we aim here to enrich *Saul* with components that facilitate these analyses in the NLP domain (see Section 4).

We first introduce the notation and the formal framework for designing global (IBT) models in *Saul*. In supervised *structured* learning we are given a set of examples i.e. pairs of input and output, $E = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) \in \mathcal{X} \times \mathcal{Y} : i = 1 \dots N\}$, where both inputs ($\mathcal{X}$) and outputs ($\mathcal{Y}$) can be complex structures; the goal is learning the mapping, $g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ (Bakir et al., 2007). Making predictions in this formulation requires an inference procedure over g, that finds the best $\hat{\boldsymbol{y}}$ for a given $\boldsymbol{x}$. Thus the prediction function h is, $h(\boldsymbol{x}; \boldsymbol{w}) = \arg\max_{\hat{\boldsymbol{y}} \in \mathcal{Y}} g(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w})$. In the generalized linear models (Tsochantaridis et al., 2005), the function g is assumed to be a linear model of the input and output features $f(\boldsymbol{x}, \boldsymbol{y})$, i.e. $g(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \langle \boldsymbol{w}, f(\boldsymbol{x}, \boldsymbol{y}) \rangle$, where $\boldsymbol{w}$ denotes the parameters of the model (weight vector). A commonly used discriminative training method is to minimize the following convex upper bound of the loss function over the training data (Tsochantaridis et al., 2004):

$$L = \sum_{i=1}^{N} \max_{\hat{\boldsymbol{y}} \in \mathcal{Y}} \Big[g(\boldsymbol{x}^i, \hat{\boldsymbol{y}}; \boldsymbol{w}) - g(\boldsymbol{x}^i, \boldsymbol{y}^i; \boldsymbol{w}) + \Delta(\boldsymbol{y}^i, \hat{\boldsymbol{y}}) \Big]$$

The inner maximization is called loss-augmented inference and quantifies the most violated output per training example. This is a crucial inference problem to be solved during training of such models. Here we assume that the distance function Δ is decomposed in the same way as the feature function. During training and at prediction time, there is a need to solve the same inference problem to find the best $\hat{\boldsymbol{y}}$

given the objective $g(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w})$. This will be the focus of the paper: to show how we can write high level specifications of g and model it in a generic, efficient and flexible fashion.

3 Graph Representation

Saul is a powerful programming paradigm which uses graphs to explicitly declare the structure of the data that serves as the model of the domain. This graph is called *data-model*[1] and is comprised of nodes, edges and properties that describe nodes. The *data-model* is a global structure that facilitates designing a wide range of learning and inference *configurations*, based on arbitrary model decompositions.

Inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y}$ are sub-graphs of the *data-model* and each learning model can pick specific correlations and substructures from it. In other words, $\boldsymbol{x}$ is a set of nodes $\{x_1, \ldots, x_K\}$ and each node has a *type* p. Each $x_k \in \boldsymbol{x}$ is described by a set of properties; this set of properties will be converted to a feature vector ϕ_p. For instance, an input type can be a word (*atomic node*) or a pair of words (*composed node*), and each type is described by its own properties (e.g. a single word by its part of speech, the pair by the distance of the two words). The output $\boldsymbol{y}$ is represented by a set of *labels* $\boldsymbol{l} = \{l_1, \ldots, l_P\}$ each of which is a *property* of a node. The labels can have semantic relationships. We conceptually (not technically) distinguish between two types of labels: the *single labels* and *linked labels* that refer to an independent concept and to a configuration of a number of related single labels respectively. *Linked label*s can represent different types of semantic relationships between single labels.

For convenience, to show which labels are connected by a *linked label*, we represent the *linked labels* by a concatenation of the labels' names that are linked together and construct a bigger semantic part of the whole output. For example an SRL predicate-argument relation (see Figure 1 in Section 4) can be denoted by *pred-arg* meaning that it is *composed-of* the two single labels, *pred* (predicate), and *arg* (argument). The labels are defined with a graph query that extracts a property from the *data-model*. The $l_p(x_k)$ or shorter l_{pk} denotes an indicator function indicating whether component x_k has the label l_p. Kordjamshidi et al. (2015) introduced the term *relational constrained factor graph* to represent each possible learning and inference configuration (Taskar et al., 2002; Taskar et al., 2004; Bunescu and Mooney, 2007; Martins et al., 2011).

The structure of the learning/inference i.e. the relational constraint factor graph is specified with a set of *templates* which can be constraint templates or feature templates, $\mathcal{C} = \{\mathcal{C}_1, .., \mathcal{C}_P\}$. Each template $\mathcal{C}_p \in \mathcal{C}$ is specified by three main components: 1) *A subset of joint features*, denoted by $f_p(\boldsymbol{x}_k, l_p)$, where $\boldsymbol{x}_k$ is an input component that is a node in the *data-model* graph, and l_p is a single/linked label (a property in the *data-model*). In the case of constraint templates, l_p is a Boolean label denoting the satisfiability of the constraint. 2) *A candidate generator*, that generates candidate components upon which the specified subset of joint features is applicable, the set of candidates for each template is denoted by C_{l_p}. For constraint templates the candidate generator is the propositionalization of the constraint's first-order logical expression. 3) *A block of weights* $\boldsymbol{w}_p$, which is a part of the main weight vector $\boldsymbol{w}$ of the model and is associated to the local joint feature function of $\mathcal{C}_p$. In general, $\boldsymbol{w}_p$ can also be defined for the constraint templates. The main objective g is written in terms of the instantiations of the (feature) templates and their related blocks of weights $\boldsymbol{w}_p$ in $\boldsymbol{w} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_P]$,

$$g(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \sum_{l_p \in \boldsymbol{l}} \sum_{x_k \in C_{l_p}} \langle \boldsymbol{w}_p, f_p(x_k, l_p) \rangle = \sum_{l_p \in \boldsymbol{l}} \sum_{x_k \in C_{l_p}} \langle \boldsymbol{w}_p, \phi_p(x_k) \rangle \, l_{pk} = \sum_{l_p \in \boldsymbol{l}} \left\langle \boldsymbol{w}_p, \sum_{x_k \in C_{l_p}} (\phi_p(x_k) l_{pk}) \right\rangle, \quad (1)$$

where the local joint feature vector $f_p(\boldsymbol{x}_k, l_p)$, is an instantiation of the template $\mathcal{C}_{l_p}$ for candidate $\boldsymbol{x}_k$. This feature vector is computed by scalar multiplication of the input feature vector of $\boldsymbol{x}_k$ (i.e. $\phi_p(\boldsymbol{x}_k)$), and the output label l_{pk}.

Given this objective, we can view the inference task as a *combinatorial constrained optimization* given the polynomial g which is written in terms of labels, subject to the constraints that describe the relationships between the labels (either single or linked labels). For example, the *is-a* relationships can be defined as the following constraint, $(l(\boldsymbol{x}_c)\ is\ 1) \Rightarrow (l'(\boldsymbol{x}_c)\ is\ 1)$, where l and l' are two distinct

[1] Kordjamshidi et al. (2015) used the term *model-graph*.

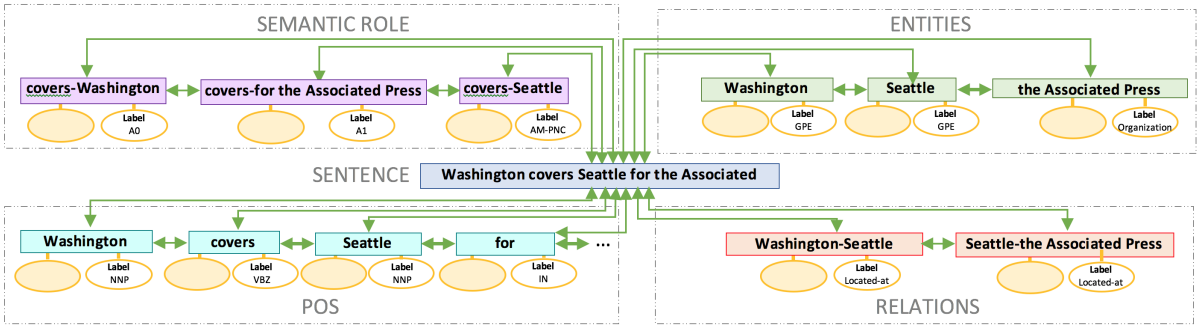


Figure 1: An instantiation of the *data-model* for the NLP domain. The colored ovals are some observed properties, while the white ones show the unknown labels. For the POS and Entity Recognition tasks, the boxes represent candidates for *single labels*; for the SRL and Relation Extraction tasks, they represent candidates for *linked labels*.

labels that are applicable on the node with the same type of $\boldsymbol{x}_c$. These constraints are added as a part of *Saul*'s objective, so we have the following objective form, which is in fact a constrained conditional model (Chang et al., 2012), $g = \langle \boldsymbol{w}, f(\boldsymbol{x}, \boldsymbol{y}) \rangle - \langle \rho, c(\boldsymbol{x}, \boldsymbol{y}) \rangle$, where c is the constraint function and ρ is the vector of penalties for violating each constraint. This representation corresponds to an integer linear program, and thus can be used to encode any MAP problem. Specifically, the g function is written as the sum of local joint feature functions which are the counterparts of the probabilistic factors:

$$g(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \sum_{l_p \in \boldsymbol{l}} \sum_{\boldsymbol{x}_k \in \{\tau\}} \langle \boldsymbol{w}_p, f_p(\boldsymbol{x}_k, l_{pk}) \rangle + \sum_{m=1}^{|C|} \rho_m c_m(\boldsymbol{x}, \boldsymbol{y}), \tag{2}$$

where C is a set of global constraints that can hold among various types of nodes. g can represent a general scoring function rather than the one corresponding to the likelihood of an assignment. The constraints are used during training for loss-augmented inference as well as during prediction.

4 Calling *Saul*: Case Studies

For **programming global models** in *Saul* the programmer needs to declare a) the *data-model* which is a global structure of the data and b) the templates for learning an inference decompositions. The templates are declared intuitively in two forms of classifiers using `Learnable` construct and first order constraints using `ConstrainedClassifier` construct. With these components have been specified, the programmer can easily choose which templates to use for learning (training) and inference (prediction). In this way the global objective is generated automatically for different training and testing paradigms in the spectrum of local to global models.

One advantage of programming in *Saul* is that one can define a generic *data-model* for various tasks in each application domain. In this paper, we enrich *Saul* with an NLP *data-model* based on EDISON, a recently-introduced NLP library which contains raw data readers, data structures and feature extractors (Sammons et al., 2016) and use it as a collection of *Sensors* to easily generate the *data-model* from the raw data. In *Saul*, a *Sensor* is a 'black-box' function that can generate nodes, edges and properties in the graph. An example of a sensor for generating nodes and edges is a sentence tokenizer which receives a sentence and generates its tokens. Here, we will provide some examples of *data-model* declaration language but more details are available on-line[2].

In the rest of the paper, we walk through the tasks of Semantic Role Labeling (SRL), Part-of-Speech (POS) tagging and Entity-Relation (ER) extraction and show how we can design a variety of local to global models by presenting the related code[3].

[2]https://github.com/IllinoisCogComp/saul/blob/master/saul-core/doc/DATAMODELING.md
[3]https://github.com/IllinoisCogComp/saul/tree/master/saul-examples/src/main/scala/edu/illinois/cs/cogcomp/saulexamples/nlp

```
val sentences = node[TextAnnotation]
val predicates = node[Constituent]
val arguments = node[Constituent]
val pairs =  node[Relations]
val pos-tag = property(arguments)
val word-form = property(arguments)
val relationsToArguments = edge(relations, arguments)
relationsToArguments.addSensor(relToArgument _)
```

Figure 2: An Example of *data-model* declarations for nodes, edges, properties and using sensors. The `sentences` nodes are of type `TextAnnotation` class, which is a part of *Saul*'s underlying NLP library; many predefined sensors can be applied on it to generate various nodes of type `Constituent` and `Relations`, properties of those nodes and establish edges between them.

4.1 Semantic Role Labeling

SRL (Carreras and Màrquez, 2004) is a shallow semantic analysis framework, whereby a sentence is analysed into multiple propositions; each one consisting of a predicate and one or more core arguments, labeled with protosemantic roles (agents [Arg0], patient/theme [Arg1], beneficiary [Arg2], etc.), and zero or more optional arguments, labeled according to their semantic function (temporal, locative, manner, etc.). See Figure 1 for an example annotation.

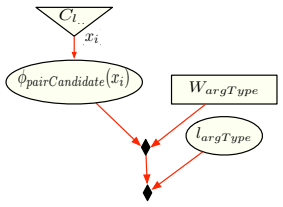
4.1.1 Input-Output Spaces

Each sentence is a node in the *data-model*, comprised of *constituents* (derived from a tokenizer *Sensor*). These constituents are atomic components of $\boldsymbol{x}$ (see Figure 1) and are identified as $\boldsymbol{x} = \{x_1, \ldots, x_4\}$, where x_i is the identifier of the ith constituent in the sentence. Each constituent is described by a number of properties (word-form, pos-tag, ...) and the corresponding feature vector representation of these properties is denoted by $\phi_{constituent}(x_i)$. There are also composed components – pairs of constituents; their descriptive vectors are referred to as $\phi_{pair}(x_i, x_j)$. The feature vector of a composed component such as a pair, $\phi_{pair}(x_1, x_2)$ is usually described by the local features of x_1, x_2 and the relational features between them, such as the order of their position, etc.

The main labels set for the SRL model is $\boldsymbol{l} = \{l_{isPred}, l_{isArg}, l_{argType}\}$ which indicate whether a constituent is a predicate, whether it is an argument and the argument role respectively. l_{isArg} and $l_{argType}$ are *linked labels*, meaning that they are defined with respect to another constituent (the predicate). Depending on the type of correlations we aim to capture, we can introduce new *linked labels* in the model. These labels are not necessarily the target of the predictions but they help to capture the dependencies among labels. For example, to capture the long distance dependencies between two different arguments of same predicate we can introduce a *linked label* linking the label of two pairs and impose consistency constraints between this new *linked label* and the label of each pair. Figure 2 shows some declarations of the *data-model*'s graph representing input and output components of the learning models. The graph can be queried using our invented graph traversal language and the queries are directly used as structured machine learning features later.

4.1.2 Classifiers and Constraints

As mentioned in Section 3, the structure of a learning model is specified by templates which are defined as classifiers (feature templates) and global constraints (constraint templates) in *Saul*. SRL has four main feature templates: 1) *Predicate* template: connects an input constituent to a single label l_{isPred}. The input features of this template are generated based on the properties of the constituents $\phi_{constituent}$. The candidate generator of this template is a filter that takes all constituents whose pos-tag is *VP*; 2) *Argument* template: connects a pair of constituents to a *linked label* l_{isArg}. The candidate generator of this template is a set of rules that are suggested by Xue and Palmer (2004); 3) *ArgumentType* template: connects a pair of constituents to the linked label $l_{argType}$. Same Xue-Palmer heuristics are used; 4) *ArgumentTypeCorrelations* template: connects two pairs of pairs of constituent (i.e. relations between relations) to their join linked label. The candidates are the pairs of Xue-Palmer candidates.




```
object ArgTypeLearner extends Learnable(pairs){
    def label = argumentLabelGold
    def feature = using(containsMOD, containsNEG,
    clauseFeatures, chunkPathPattern, chunkEmbedding,
    chunkLength, constituentLength, argPOSWindow,
    argWordWindow, headwordRelation, syntacticFrameRelation
    , pathRelation, phraseTypeRelation, predPosTag,
    predLemmaR, linearPosition)
}
```

Figure 3: Left: shows the components of the *ArgumentType* feature template. $l_{argType}$ is one *linked label* as a part of the objective in Equation 1, along with the corresponding block of weights and the `pair` candidates (diamonds show dot products). Right: shows the code for the template. `label` and `feature` are respectively one property and a list of properties of `pair` nodes declared in the *data-model*, these serve as the output and input parts of this template. This template can be used as a local classifier or as a part of the objective of a global model, depending on the indicated learning paradigm by the programmer.

```
val legalArgumentsConstraint = constraint(sentences) { x =>
    val constraints = for {
      predicate <- sentences(x) ~> sentenceToPredicates
      candidateRelations = (predicates(y) ~> -relationsToPredicates)
      argLegalList = legalArguments(y)
      relation <- candidateRelations
    } yield classiferLabelIsLegal(argumentTypeLearner, relation, argLegalList)
        or (argumentTypeLearner on relation is "none")
}

def classiferLabelIsLegal(classifier, relation, legalLabels) = {
    legalLabels._exists { l => (classifier on relation is l) }
}
```

Figure 4: Given a predicate, some argument types are illegal according to PropBank Frames (e.g. the verb 'cover' with sense 03 can take only Arg0 or Arg1), which means that they should be excluded from the inference. The `legalArguments(y)` returns the predefined list of legal arguments for a predicate y. In line 3, graph traversal queries (using the $\sim>$ operator) are applied to use an edge and go from a *sentence* node to all contained predicate nodes in the sentence and then apply the constraint to all of those predicates. Each constraint imposes the `argumentTypeLearner` to assign a legal argument type to each candidate argument or does not count it as an argument at all, i.e., to assign `none` value to the argument type.

The feature templates are instances of `Learnable` in *Saul* and in fact they are treated as local classifiers. The script of Figure 3 shows the *ArgumentType* template. The *Constraints* are specified by means of first-order logical expressions. We use the constraints specified in Punyakanok et al. (2008) in our models. The script in Figure 4, shows an example expressing the *legal argument* constraints for a sentence.

4.1.3 Model Configurations

Programming for learning and inference configurations in *Saul* is simply composing the basic building blocks of the language, that is, feature and constraint templates in different ways.

Local models. Training local models is as easy as calling the train function over each specified feature template separately (e.g. `ArgTypeLearner.train()`). The test on these models also is simply done by calling test for each template (e.g. `ArgTypeLearner.test()`). In addition, `TestClassifeirs(/*a list of classifiers*/)` and `TrainClassifiers(/*a list of classifiers*/)` can be used to train/test a number of classifiers independently by passing a list of classifier's names to these functions. The training algorithm can be specified when declaring the `Learnable`; here we have used averaged perceptrons in the experiments which is the default model.

Model	Precision	Recall	F1
ArgTypeLearner^{G}(GOLDPREDS)	85.35	85.35	85.35
ArgTypeLearner^{G}(GOLDPREDS) + C	85.35	85.36	85.35
ArgTypeLearner^{Xue}(GOLDPREDS)	82.32	80.97	81.64
ArgTypeLearner^{Xue}(GOLDPREDS) + C	82.90	80.70	81.79
ArgTypeLearner^{Xue}(PREDPREDS)	82.47	80.79	81.62
ArgTypeLearner^{Xue}(PREDPREDS) + C	83.62	80.54	82.05
ArgIdentifier^{Xue} | ArgTypeLearner^{Xue}(PREDPREDS)	82.55	81.59	82.07
ArgIdentifier^{G}(PREDPREDS)	95.51	94.19	94.85

Table 1: Evaluation of SRL various labels and configurations. The superscripts over the different Learners refer to the whether gold argument boundaries (G) or the Xue-Palmer heuristics (Xue) were used to generate argument candidates as input. GOLD/PREDPREDS refers to whether the Learner used gold or predicted predicates. 'C' refers to the use of constraints during prediction and |denotes the pipeline architecture.

Pipeline. Previous research on SRL (Punyakanok et al., 2008) shows that a good working model is the one that first decides on argument identification and then takes those arguments and decides about their roles. This configuration is made with a very minor change in the templates of the local models. Instead of using Xue-Palmer candidates, we can use the identified arguments by a isArgument classifier as input candidates for the ArgTypeLearner model. The rest of the model is the same.

L+I model. This is simply a locally trained classifier that uses a number of constraints on prediction time. We define a constrained argument predictor based on a previously trained local *Learnable* as follows:

```
object ArgTypeConstraintClassifier extends ConstrainedClassifier(ArgTypeLearner)
{
    def subjectTo = srlConstraints
}
```

where the srlConstraints is a constraint template. Having this definition we only need to call the ArgTypeConstraintClassifier constraint predictor during the test time as ArgTypeConstraintClassifier(x) which decides for the label of x in a global context.

IBT model. The linguistic background knowledge about SRL that is described in Section 4.1.2 provides the possibility of designing a variety of global models. The constraints that limit the argument arrangement around a specific predicate help to make sentence level decisions for each predicate during training phase and/or prediction phase. To train the global models we simply call the joint train function and provide the list of all declared constraint classifiers as parameters.

The results of some versions of these models are shown in Table 1. The experimental settings, the data and the train/test splits are according to (Punyakanok et al., 2008) and the results are comparable. As the results show the models that use constraints are the best performing ones. For SRL the global background knowledge on the arguments in IBT setting did not improve the results.

4.2 Part-Of-Speech Tagging

This is perhaps the most often used application in ML for NLP. We use the setting proposed by Roth and Zelenko (1998) as the basis for our experiments. The graph of an example sentence is shown in Figure 1. We model the problem as a single-node graph representing constituents in sentences. We make use of context window features and hence our graph has edges between each token and its context window. This enables us to define contextual features by traversing the relevant edges to access tokens in the context. The following code uses the gold POS-tag label (POSLabel) of the two tokens before the current token during training and POS-tag classifier's prediction (POSTaggerKnown) of the two tokens before the current token during the test.

```
val labelTwoBefore = property(tokens) { x: Constituent =>
  // Use edges to jump to the previous constituent
  val cons = (tokens(x) ~> constituentBefore ~> constituentBefore).head
  if (POSTaggerKnown.isTraining)
    POSLabel(cons)
  else POSTaggerKnown(cons)
}
```

4.2.1 Model configurations

Here, we point to a few interesting settings for this problem and report the results we obtained by *Saul* in Table 2.

Count-based baseline. The simplest scenario is to create a simple count-based baseline: for each constituent choose the most popular label. This is trivial to program in *Saul*.

Independent classifiers. We train independent classifiers for known and unknown words. Though both classifiers use similar sets of features, the unknown classifier is trained only on tokens that were seen fewer than 5 times in the training data. Here the 'Learnable' is defined as exampled in Section 4.1.2.

Classifier combination. Given the known and unknown classifiers, one easy extension is to combine them, depending whether the input instance is seen during the training phase or not. To code this, the defined Learnables for the two classifiers are simply reused in an 'if' construct.

Sequence tagging. One can extend the previous configurations by training higher-order classifiers, i.e. classifiers trained on pair/tuple of neighboring constituents (similar to HMM or chain-CRF). At the prediction time one needs to choose the best structure by doing constrained inference on the predictions of the local classifiers. The following snippet shows how one can simply write a consistency constraint, given a pairwise classifier `POSTaggerPairwise` which scores two consecutive constituents.

```
def sentenceLabelsMatch = constraint(sentences) {
    t: TextAnnotation =>
    val constituents = t.getView(ViewNames.TOKENS).getConstituents
    // Go through a sliding window of tokens
    constituents.sliding(3)._forall { cons: List[Constituent] =>
      POSTaggerPairwise on (cons(0), cons(1)).second === POSTaggerPairwise on (
    cons(1), cons(2)).first }
}
```

4.3 Entity-Relation extraction

This task is for labeling entities and recognizing semantic relations among them. It requires making several local decisions (identifying named entities in the sentence) to support the relation identification. The models we represent here are inspired some well-known previous work (Zhou et al., 2005; Chan and Roth, 2010). The nodes in our models consists of `Sentences`, `Mentions` and `Relations`.

4.3.1 Features and Constraints

For the entity extraction classifier, we define various lexical features for each mention – head word, POStags, words and POStags in a context window. Also, we incorporate some features based on gazetteers for organization, vehicle, weapons, geographic locations, proper names and collective nouns. The relation extraction classifier uses lexical, collocation and dependency-based features from the baseline implementation in Chan and Roth (2010). We also use features from the brown word clusters (Brown et al., 1992). The features for each word are based on a path from the root in its Brown clustering representation. These features are easily available in our NLP *data-model*. We also use a decayed down-sampling of negative examples between training iterations.

Setting	Accuracy
Count-based baseline	91.80%
Unknown Classifier	77.09%
Known Classifier	94.92 %
Combined Known-Unknown	96.69%

Table 2: The performance of the POStagger, tested on sections 22–24 of the WSJ portion of the Penn Treebank (Marcus et al., 1993).

	Scenario	Precision	Recall	F1
E	Mention Coarse-Label	77.14	70.62	73.73
	Mention Fine-Label	73.49	65.46	69.24
R	Basic	54.09	43.89	50.48
	+ Sampling	52.48	56.78	54.54
	+ Sampling + Brown	54.43	54.23	54.33
	+ Sampling + Brown + HCons	55.82	53.42	54.59

Table 3: 5-fold CV performance of the fine-grained entity (E) and relation (R) extraction on Newswire and Broadcast News section of ACE-2005.

Relation hierarchy constraint. Since the coarse and fine labels follow a strict hierarchy, we leverage this information to boost the prediction of the fine-grained classifier by constraining its prediction upon the (more reliable) coarse-grained relation classifier.

4.3.2 Model Configuration

Entity type classifier. For the entity type task, we train two independent classifiers - one for coarse-label and the second for the fine-grained entity type. We generate the candidates for entities by taking all nouns and possessive pronouns, base noun phrases, selective chunks from the shallow parse and named entities annotated by the NE tagger of Ratinov and Roth (2009).

Relation type classifier. For the relation types, we train two independent classifiers - coarse-grained relation type label and fine-grained relation type label. We use features from our unified *data-model* which are `properties` defined on the `relations` node in the data-model graph. We also incorporate the Relation Hierarchy constraint during inference so that the predictions of both classifiers are coherent. We report some of our results in Table 3.

5 Related Work

This work has been done in the context of *Saul*, a recently developed declarative learning based programming language. DeLBP is a new paradigm (Roth, 2005; Rizzolo, 2011; Kordjamshidi et al., 2015) which is related to probabilistic programming languages (PPL) (Pfeffer, 2009; McCallum et al., 2009) *(inter alia)*, sharing the goal of facilitating the design of learning and inference models. However, compared to PPL, it is aimed at non-expert users of machine learning, and it is a more generic framework that is not limited to probabilistic models. It focuses on learning over complex structures where there are global correlations between variables, and where first order background knowledge about the data and domain could be easily considered during learning and inference. The desideratum of this framework is the conceptual representation of the domain, data and knowledge, in a way that is suitable for non-experts in machine learning and, it considers the aspect of relational feature extraction; this is different also from the goals of Searn (Hal et al., 2009) and Wolf (Riedel et al., 2014). DeLBP focuses on data-driven learning and reasoning for problem solving and handling collections of data from heterogeneous resources, unlike Dyna (Eisner, 2008) which is a generic declarative problem solving paradigm based on dynamic programming. This paper exhibits the capabilities and flexibility of Saul for solving problems in the NLP domain. Specifically, it shows how a unified predefined NLP *data-model* can help performing various tasks at various granularity levels.

6 Conclusion

We presented three examples of NLP applications as defined in the declarative learning-based programming language *Saul*. The main advantage of our approach compared to traditional, task-specific, systems is that *Saul* allows one to define all the components of the models declaratively, from feature extraction to learning and inference with arbitrary structures. This allows designers and researchers a way to explore different way to decompose, learn and do inference and easily gain insights into the impact of these on the

task. We enriched *Saul* with an extensive NLP *data-model* that enables users to perform various tasks at different levels of granularity and eventually to perform multiple tasks jointly. This work will help pave the way for more learning-based programming applications which will allow both practitioners and researchers in the field to develop quick solutions to advanced NLP tasks and to focus on exploring the tasks while staying at a sufficient level of abstraction from the component's implementation.

Acknowledgments

The authors would like to thank all the students who have helped in the implementation of this project, as well as the anonymous reviewers for helpful comments. This research is supported by NIH grant U54-GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov), and Allen Institute for Artificial Intelligence (`allenai.org`). This material is based on research sponsored by DARPA under agreement number FA8750-13-2-0008. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government nor NIH.

References

G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan. 2007. *Predicting Structured Data (Neural Information Processing)*. The MIT Press.

P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, pages 467–479.

R. Bunescu and R. J. Mooney. 2007. Statistical relational learning for natural language information extraction. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*, pages 535–552. MIT Press.

X. Carreras and L. Màrquez. 2004. Introduction to the CoNLL-2004 shared tasks: Semantic role labeling. In *Proceedings of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 89–97. Boston, MA, USA.

Y. Chan and D. Roth. 2010. Exploiting background knowledge for relation extraction. In *Proc. of the International Conference on Computational Linguistics (COLING)*, Beijing, China.

M. Chang, L. Ratinov, and D. Roth. 2012. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431, 6.

K.-W. Chang, V. Srikumar, and D. Roth. 2013. Multi-core structural svm training. In *Proc. of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*.

M. Collins. 2004. Parameter estimation for statistical parsing models: theory and practice of distribution-free methods. In *New Developments in Parsing Technology*, pages 19–55. Kluwer.

J. Eisner. 2008. Dyna: A *non*-probabilistic programming language for probabilistic AI. Extended abstract for talk at the NIPS*2008 Workshop on Probabilistic Programming.

D. Hal, J. Langford, and D. Marcu. 2009. Search-based structured prediction. *Machine Learning*, pages 297–325, June.

P. Kordjamshidi and M-F. Moens. 2013. Designing constructive machine learning models based on generalized linear learning techniques. In *NIPS Workshop on Constructive Machine Learning*.

P. Kordjamshidi, D. Roth, and H. Wu. 2015. Saul: Towards declarative learning based programming. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*.

M. P. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, June.

A. FT. Martins, M. AT. Figeuiredo, P. MQ. Aguiar, N. A. Smith, and E. P. Xing. 2011. An augmented Lagrangian approach to constrained MAP inference. In *International Conference on Machine Learning (ICML)*.

A. McCallum, K. Schultz, and S. Singh. 2009. FACTORIE: Probabilistic Programming via Imperatively Defined Factor Graphs. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*.

A. Pfeffer. 2009. Figaro: An object-oriented probabilistic programming language. Technical report, Charles River Analytics.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2005. Learning and inference over constrained output. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1124–1129.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*.

L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*.

S. Riedel, S. Singh, V. Srikumar, T. Rocktäschel, L. Visengeriyeva, and J. Noessner. 2014. WOLFE: strength reduction and approximate programming for probabilistic programming. *Statistical Relational Artificial Intelligence*.

N. Rizzolo. 2011. *Learning Based Programming*. Ph.D. thesis, University of Illinois, Urbana-Champaign. http://cogcomp.cs.illinois.edu/papers/Rizzolo11.pdf.

D. Roth and D. Zelenko. 1998. Part of speech tagging using a network of linear separators. In *The 17th International Conference on Computational Linguistics (COLING-ACL)*, pages 1136–1142.

D. Roth. 2005. Learning based programming. *Innovations in Machine Learning: Theory and Applications*.

R. Samdani and D. Roth. 2012. Efficient decomposed learning for structured prediction. In *Proc. of the International Conference on Machine Learning (ICML)*.

M. Sammons, C. Christodoulopoulos, P. Kordjamshidi, D. Khashabi, V. Srikumar, P. Vijayakumar, M. Bokhari, X. Wu, and D. Roth. 2016. Edison: Feature Extraction for NLP, Simplified. In *LREC*.

C. Sutton and A. McCallum. 2009. Piecewise training for structured prediction. *Machine Learning*, pages 165–194.

B. Taskar, P. Abbeel, and D. Koller. 2002. Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, UAI, pages 485–492. Morgan Kaufmann Publishers Inc.

B. Taskar, M. Fai Wong, P. Abbeel, and D. Koller. 2004. Link prediction in relational data. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the International Conference on Machine Learning (ICML)*.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research (JMLR)*, pages 1453–1484.

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 88–94, Barcelona, Spain.

G. Zhou, J. Su, J. Zhang, and M. Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 427–434. Association for Computational Linguistics.

Crowdsourcing Complex Language Resources: Playing to Annotate Dependency Syntax

Bruno Guillaume
Inria Nancy Grand-Est/LORIA
bruno.guillaume@inria.fr

Karën Fort
Université Paris-Sorbonne / STIH
karen.fort@paris-sorbonne.fr

Nicolas Lefebvre
Inria Nancy Grand-Est/LORIA
nicolas.lefebvre@inria.fr

Abstract

This article presents the results we obtained on a complex annotation task (that of dependency syntax) using a specifically designed Game with a Purpose, ZombiLingo.[1] We show that with suitable mechanisms (decomposition of the task, training of the players and regular control of the annotation quality during the game), it is possible to obtain annotations whose quality is significantly higher than that obtainable with a parser, provided that enough players participate. The source code of the game and the resulting annotated corpora (for French) are freely available.

1 Introduction

For better or worse,[2] the overwhelming domination of machine-learning systems in natural language processing (NLP) over the past 25 years and the increasing use of evaluation campaigns and shared tasks have put human-annotated corpora at the heart of the field. Manual annotation is now the place where linguistics hides.

The availability of manually annotated corpora of high quality (or, at least, reliability) is therefore key to the development of the field in any given language. However, the creation of such resources is notoriously costly, especially when complex annotations, e.g. for dependency syntax, are at issue. For example, the cost of the Prague Dependency Treebank was estimated at $600,000 in (Böhmová et al., 2001).

Over the years, many solutions have been investigated in the attempt to lower manual annotation costs. One obvious avenue is to use an appropriate annotation tool, as shown for example in (Dandapat et al., 2009). As a complementary aid, NLP systems can be used to reduce the annotation burden – either beforehand, for example with tag dictionaries (Carmen et al., 2010) and more generally with pre-annotation (Skjærholt, 2013), or more iteratively during the annotation process, for instance through active learning (Baldridge and Osborne, 2004). These solutions have proved efficient and have indeed helped reduce the annotation cost, but the creation of a large annotated corpus in the traditional manner remains very expensive.

Another way to address the issue is simply to limit the amount paid to the human annotators. This is the case with microworking crowdsourcing, especially through the use of platforms like Amazon Mechanical Turk, via which the workers are (micro)paid to perform simplified tasks (Snow et al., 2008). Apart from the ethical issues raised by these platforms (detailed in (Fort et al., 2011)), microworking platforms do not support true training of the workers: tests can be set up to select them, but they have to perform hidden work to train themselves (as shown in (Gupta et al., 2014)). Consequently, tasks must be simplified to be manageable by workers: for example, a task related to recognition of textual entailment across several sentences (which might actually reflect entailment, neutrality, or contradiction)

[1] See: http://zombilingo.org/.

[2] See (Church, 2011) for an in-depth reflection on the subject.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3041–3052, Osaka, Japan, December 11-17 2016.

could be simplified by presenting only one pair of sentences and a binary response to a single question, such as "Would most people say that if the first sentence is true, then the second sentence must be true?" (Bowman et al., 2015). To paraphrase (Dandapat et al., 2009), it seems that there is no easy escape from the high cost of complex linguistic annotation.

We present here an on-line game that enables the production of quality annotations of complex phenomena (here, dependency syntax), tested on French. The produced corpus is constantly growing and is designed to be (i) completely free and available, and (ii) of sufficient quality. We first examine the existing treebanks for French and their limitations and detail some previous experiments with Games with a Purpose (GWAPs). Then we present the game we developed and the evaluation we performed on the produced annotations. We finally conclude by discussing the potentials and limits of GWAPs for the creation of complex language resources.

2 Previous Work

2.1 Treebanking for French

As surprising as it may seem, French has long been relatively low-resourced, primarily due to legal issues: lexicons and annotated corpora existed but could not be used or redistributed freely. This is in particular the case for the French Treebank (FTB or *corpus arboré de Paris 7*) (Abeillé et al., 2003), which is available for research purposes only and cannot be freely redistributed.[3] Several versions are reported in the literature, with the size varying from 12,351 sentences and 350,947 words (for the FTB-UC) to 18,535 sentences and 557,149 words[4] (for the FTB-SPRML).

To try and circumvent this restriction, Candito and Seddah created the Sequoia corpus (Candito and Seddah, 2012), which is freely available[5] under a LGPL-LR license, but is limited to 67,038 tokens. The same authors developed an additional question bank with 23,236 tokens (Seddah and Candito, 2016). Both corpora use the same annotation guide and set of relations as the FTB.

A Universal Dependency corpus (McDonald et al., 2013) was created for French and is freely available under a CC BY-NC-SA license. In version 1.3, released in May 2016, it contains 401,960 tokens, but it "has not been manually corrected systematically".[6] Moreover, the annotation format for Universal Dependencies suffers from certain drawbacks, for example, it does not distinguish between arguments and modifiers for nominal complements of verbs.

Other treebanks exist for French, but they either concern spoken language, as with Rhapsodie (Lacheret et al., 2014) or the oral Treebank (Abeillé and Crabbé, 2013), or specific language types, like the Social Media Treebank (Seddah et al., 2012).

This situation (in which references exist, but a large, fully available, manually annotated corpus is lacking) makes dependency syntax for French an appropriate candidate for testing a new paradigm for complex linguistic resource development: the use of on-line Games with a Purpose.

2.2 Playing to Create Language Resources

Games with a Purpose are games in which participants, knowingly or not, create data by playing. They are not serious games as such, as their main purpose is not to train people, but to produce data (such as annotations, lexicon entries, image labels, etc). GWAPs for NLP are broadly surveyed in (Lafourcade et al., 2015), and a detailed analysis of their performance in comparison with other means of language resource production is provided in (Chamberlain et al., 2013). Thus, we will focus here on the complexity of the resource to be produced, rather than on giving an exhaustive list of games.

Most GWAPs rely on the players' knowledge of the world and innate capabilities to enable creation of data, in our case language resources. This reliance was seen for example in the very well-known ESP Game (von Ahn and Dabbish, 2004), in which participants played by labeling images. Another early GWAP is JeuxDeMots[7] (Lafourcade, 2007), in which players created a lexical network of more than

[3] See `http://www.llf.cnrs.fr/fr/Gens/Abeille/French-Treebank-fr.php`.

[4] These numbers correspond to the version found on: `http://gforge.inria.fr/projects/fdtb-v1`.

[5] See here: `http://deep-sequoia.inria.fr/`.

[6] See `http://universaldependencies.org/fr/overview/introduction.html`.

[7] The game is still running online: `http://www.jeuxdemots.org`.

47 million relations between more than 900,000 entries (terms and named entities), simply by entering terms and named entities associated in a more or less specified way with another entry. The video games presented in (Jurgens and Navigli, 2014) also seem[8] to target the users' intuition and knowledge of the world to perform word sense disambiguation and a mapping from WordNet senses to images. A more limited gamified interface is used in the Language Quiz[9], which asks the participants to perform sentence or tweet sentiment analysis, without any training.[10]

Other GWAPs benefit from the players' school knowledge. One such game is Phrase Detectives[11] (Poesio et al., 2013), in which participants are asked to identify the antecedent of a noun phrase in a text. The task is more complex than labeling images or associating ideas, so the game includes a mandatory short training phase, allowing the players to become familiar with it. Almost 8,000 players participated in the game, which enabled the annotation of anaphora relations in a 162,000 word corpus. The agreement between players and experts was evaluated overall at 84%.

The wordrobe website[12] presents a family of eight gamified tasks[13] whose results help to improve the Groningen Meaning Bank (Venhuizen et al., 2013). The proposed tasks all relate to semantic disambiguation (noun *vs* verb, co-reference identification, named entity annotation, etc) and while some are relatively easy – like Play Twins (noun *vs* verb) or Play Names (named entity annotation) – most require some more advanced (at least school-level) knowledge. However, the interface does not provide a training phase and the only help available is a short guide to the task.

In a very different domain (biochemistry), the creators of FoldIt[14] demonstrated that people with no specific knowledge of the subject could be trained to perform complex protein folding tasks with impressive results: players helped find the solution to a long-standing problem concerning protein crystal structure (Khatib et al., 2011). To achieve these results, players are trained within "introductory levels", solving puzzles of varying complexity "[...] introducing the player to new problem related concepts as though they are the rules of a game." (Cooper et al., 2010).

This approach seemed particularly suited to our task (dependency annotation with nearly twenty relations to annotate), so we decided to build upon it, asking the player to annotate one relation at a time, with a specific training process for each. We report here the results we obtained, which show that a GWAP can be used for a complex linguistic annotation task that is not directly linked to world knowledge or to human intuition. To our knowledge, no other such experiment has been carried out to date. The model of dependency syntax that we use is well-known in linguistics and NLP but not in the French educational system, so we can assume that people without a linguistic or NLP background have no knowledge of it.

It should be noted that an un-gamified crowdsourcing-based method has been proposed in (Hana and Hladká, 2012; Hladká et al., 2014) for the dependency syntax annotation of Czech. (In the Czech Republic, the presentation of syntax in school is very close to dependency syntax.) However, the authors report in (Hana and Hladká, 2012) that the accuracy of the annotations they obtained is significantly lower than that of their parser. Moreover, they evaluate the results on a small set of 100 sentences selected or created by the authors. In their more recent publication, they report only the tree editing distance between the produced annotation and the reference annotation; hence, we are unable to compare their results to those presented here.

3 Designing a Game for a Complex Task

We describe below the mechanisms used in the game to take into account the complexity of the task and the game's avoidance of direct reliance on intuition. We have chosen to rely on the Sequoia annotation guidelines (which largely follow the FTB guidelines). This is a natural choice if we want to use avail-

[8] The games do not seem to be running as of mid-July 2016, so we could not test them.

[9] See: `http://quiz.ucomp.eu`

[10] Two of the authors tested it in September 2016 and were at that time the only participants.

[11] See: `https://anawiki.essex.ac.uk/phrasedetectives/`.

[12] See: `http://wordrobe.housing.rug.nl`.

[13] The game features are reduced to a leader board and a betting option, we are therefore reluctant to call it a game, although there is obviously a continuum here.

[14] See: `http://fold.it`.

able resources for training players and for evaluation. Moreover, existing parsers for French have been developed with the same schema.

3.1 Decomposing the Complexity of the Task

Since the analysis of a whole sentence is complex,[15] we decided to decompose it: the full annotation of a sentence is split into atomic tasks, where each task is linked to one type of dependency relation. Thus, the player is trained on one type of relation at a time, instead of twenty, and must focus on only one type of dependency relation across several sentences. S/he therefore has fewer elements of information to remember at a time. Moreover, each relation was awarded a level which reflects its difficulty (from the point of view of the game developers) and corresponds to a level in the game. The player can therefore choose amongst more and more relations as s/he progresses in the game. We decided not to include certain relations in the current version of the game. This concerns **punct**, which is not consistently annotated in reference, reldep, which is an underspecified relation used in different types of contexts, and relations which are not processed by Talismane. Note that the results given in Section 4 is based on all the relations except **punct**.

For each relation, explanations and examples are provided with the different contexts where it may appear.[16]

In addition to explanations, it is also important to give players examples of real utterances taken from a corpus. Reference examples are taken from the Sequoia corpus and are used in the two game mechanisms, TRAINING and CONTROL, described below.

We used only a part of the corpus in order to keep sentences aside for the evaluation of the produced annotations (see Section 4). In Table 1, we show how the reference corpus is split for different uses.

$\text{REF}_{Train\&Control}$	REF_{Eval}	*Unused*
50%	25%	25%
1,549 sentences	776 sentences	774 sentences

Table 1: Uses of the 3,099 sentences of the Sequoia reference corpus.

3.2 Playing the Game

The organization of ZombiLingo is illustrated in Figure 1, which presents how the TRAINING and play phases are articulated with both the sub-corpora described in Table 1 and the raw corpora to be annotated (usually extracted from Wikipedia). The reference corpus is used in three different phases, TRAINING, CONTROL and EVAL, as explained below.

3.2.1 TRAINING phase

Following the decomposition of the task, for each dependency relation, a specific TRAINING phase is required before entering the game. During this phase, sentences from the $\text{REF}_{Train\&Control}$ corpus are presented to the player and feedback is given in case of error.

Figure 2 illustrates the feedback given to the players in the TRAINING phase: the player was asked to find the second conjunct of a coordination *et* (and) and s/he wrongly answered *Europe*. A skull and crossbones flashes and a message revealing the right answer *parvient* (reaches) are displayed.

An advantage of offering a separate TRAINING for each relation is that the player does not have to wait long before starting the game. Once s/he is connected to the game, only a few minutes are required before starting actual play and production of annotations.

3.2.2 Play phase

In the general mode, the player chooses a relation from those available and a sequence of ten questions is proposed. The questions vary as follows:

[15] Insight concerning the complexity of syntactic annotation for the Penn Treebank is provided in (Marcus et al., 1993), where the learning curve for syntax was estimated to be twice that for part-of-speech (two months *vs* one).

[16] This is much like in an annotation guide but with simpler vocabulary and fewer details.

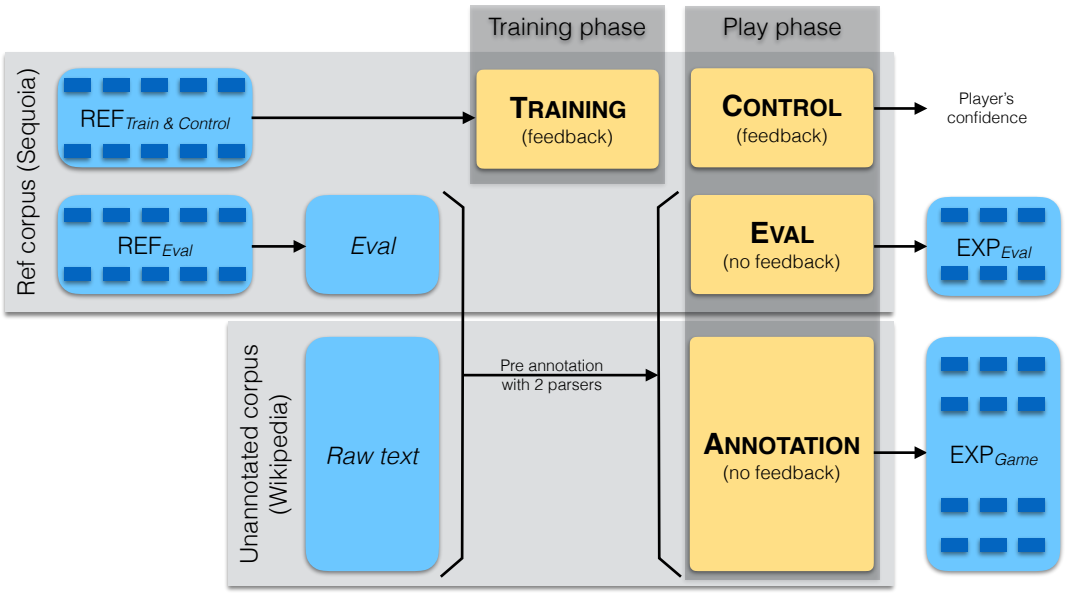


Figure 1: Organization of the different mechanisms and corpora

- For relations where the dependent element tends to be unique for a given governor (e.g. a verb has one subject, a noun has one determiner, and so on), the governor is given and the players must find the dependent;
- For relations involving several dependent elements (e.g. a verb has several prepositional phrases modifying it, a noun may be modified by several adjectives, and so on), the dependent element is given and the players must find the governor.

The player's answer is registered in the database and points are awarded if the same answer was previously given by a parser or by another player.[17] The number of points (in the game, they are represented as brains) that can be won on a certain relation depends on a global measure of the sentence complexity (measured as the maximum number of nested dependency relations) and on the level of the relation in the game.

3.2.3 CONTROL mechanism

TRAINING is not sufficient to ensure that players annotate data consistently: if they did the TRAINING on a certain relation a long time ago or if they play a lot, their ability to annotate this relation may decrease. To deal with this problem, we added a CONTROL mechanism during the game. The player is sometimes asked to annotate a relation in a sentence taken from the $\text{REF}_{Train\&Control}$ corpus. If the player fails to find the right answer, feedback including the solution is displayed (as in Figure 2). After a given number of failures on the same relation, the player has to redo the corresponding TRAINING. Using this CONTROL mechanism, we can also estimate our degree of confidence in a specific player on a specific relation (see Section 4) and take this into account to weigh his or her answer.

3.3 Behind the Curtain

3.3.1 Preprocessing Data

In the first version of the game, when new sentences were added, a parser was used to pre-annotate the sentences. Then, the players were asked to confirm or correct the parser's predictions. The main drawback of this arrangement is that the player would have a large number of very easy annotations to decide on and would usually agree with the parser.

[17] Unfortunately, if a player is the first to give a right answer, which is not given by the pre-annotation, s/he receives no points. A mechanism to give points off-line in this situation is planned but not yet implemented.

Figure 2: The main interface of the game during the TRAINING phase.

Modifying this arrangement, we now use two parsers to pre-annotate the corpora and ask the participants to play items for which the parsers give different annotations. The two parsers used are Talismane (Urieli, 2013) and FRDEP-PARSE (Guillaume and Perrier, 2015).

Talismane is a statistical parser trained on the training part of the FTB. It was evaluated on the dev and test part of the FTB with LAS scores ranging from 86.8% to 88.5%. As for FRDEP-PARSE, it is a parser which combines statistical methods for POS-tagging (using MElt (Denis and Sagot, 2012)) and symbolic methods for dependency parsing (with graph rewriting). FRDEP-PARSE was evaluated on Sequoia with a LAS score of 76.04% and a precision of 85.96%. (The system returns partial dependency syntax structures.)

It is important that the two parsers are based on different paradigms (statistical and symbolic): we can hope that the two tools are complementary and will produce different types of errors.

The next section will provide the two parsers' detailed results for the REF_{Eval} corpus.

3.3.2 Exporting Data

The players' annotations are stored in a database and each annotation receives a score. When a player is asked a question, we consider the set of possible answers in the database and adjust the score as follows:

- If the player's answer belongs to the set, the score of the answer is increased and the scores of its competing annotations (the rest of the set) are decreased.
- If the player gives an answer not in the set, a new annotation is created in the database with a default score and the scores of the answers in the set are decreased.

The positive or negative score adjustments are weighted by the level of the player, we thus award higher confidence to heavy players (who have usually reached higher levels) than to beginners. When a corpus is exported, for each token (lexical unit), we consider all the annotations in the database for which it is a dependent element and select the one with the highest score. Thus, each token receives exactly one governor with one relation and we can ensure that the exported corpus contains well-formed dependency trees.

4 Quantitative and Qualitative Evaluations

4.1 Participation and Production

If a crowdsourcing approach to annotation is to succeed, enough participants must be induced to participate and create data. As of 2016, July 10, there were 647 players registered for our game. They have

collectively produced 107,719 annotations.

As a reminder, Table 2 shows a quantitative comparison of the existing corpora for French. What this table does not show is that the corpus produced through the game is still growing, and will grow as long as we support and advertise the application.

	Sequoia 7.0	UD-French 1.3	FTB-UC	FTB-SPMRL	ZombiLingo
Sentences	3,099	16,448	12,351	18,535	5,221
Tokens	67,038	401,960	350,947	557,149	128,046
Tokens/sentence	21.6	24.4	28.4	30.1	24.5

Table 2: ZombiLingo corpus size, as compared to other existing French corpora annotated with dependency syntax.

On the ZombiLingo corpus the average number of players' annotations by tokens (called the *density*) is 0.84 (107,719 / 128,046). Figure 3 shows the density of annotations per relation. The coverage is clearly not homogeneous: the density on the relation **aff** (affix) is greater than 6, whereas for some relations it remains below 1.

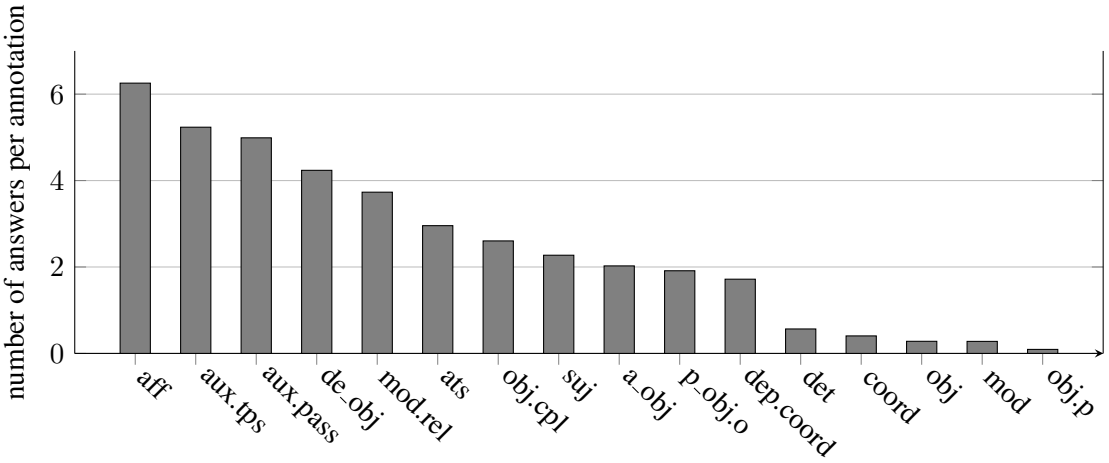


Figure 3: Density of answers for each relation.

This means that we need to attract at least some players to play a wider variety of relations types or to direct them towards some under-played relations, such as **mod** (modifier) or **coord** (coordination).

4.2 Qualitative Evaluation Methodology

In the first version of the game, we used the CONTROL mechanism to evaluate the quality of annotations produced by participants. We observed that 85.4% of CONTROL items were correctly annotated.

While this evaluation showed that it is possible to train players for a difficult task, it remains a partial evaluation, with severe drawbacks. It measures the performance of the participants on reference sentences but it gives no information about the annotations produced on pre-annotated sentences: if such a sentence contains a dependency relation not predicted by the parser, the player will never play this relation in this sentence. We call this omission *silence* in the produced data. On the other hand, if a relation is predicted by the parser but cannot be found in the sentence, the player will be asked to give an opinion on a question for which there is no sensible answer: we call such nonsensical situation *noise* in the data.

During the game, the user is supposed to click on a specific image (crossed bones) if a question has no answer but, although we train the participants on this alternative, we know that it is underused. For instance, when a player is tasked with finding the object of a verb, s/he tends to search for a word which looks like an object without realizing that this verb may actually have no object at all.

In order to obtain a precise evaluation of the produced annotations, we entered a part of our reference

corpus (named REF_{Eval}) into the game (see Figure 1), as if it were a raw corpus. We used it in the following way:

- the raw text corresponding to the REF_{Eval} corpus is put into the general pipeline (i.e., it is parsed with the two parsers, the differences in the annotation being proposed to the players; this is the EVAL mechanism in the Figure 1);
- we report the scores obtained by each parser (i.e., recall/precision/F-measure) and the score of the corpus EXP_{Eval}, as exported by the game.

4.3 Results

The REF_{Eval}corpus contains 12,660 relations which can be played. On this subset of relations, the two parsers give the same output in 10,134 of the cases. Their analysis is correct in 96.84% of the cases (i.e. in 9,814 cases out of 10,134) and it is wrong in only 320 cases, which represents 3.16% of the output (i.e. 320 cases out of 10,134). However, even in cases where the two parsers provide two different wrong answers, the players will be asked to give their opinion on these cases and, hopefully, will produce a correct final annotation. In the end, the number of wrong annotations that will never be proposed to the players represents only 2.53% (i.e. 320 cases out of 12,660) of the relations which can be played.

In Table 3, we report the scores in three settings: the Talismane parser, the FRDEP-PARSE parser, and data exported by the game (EXP_{Eval}). In all of these setting, we may have partial dependency structures, so we report recall, precision and F-measure. We consider all relations (except punctuation which is not consistently annotated in the Sequoia corpus).

	Talismane	FRDEP-PARSE	Game
LAS / Recall	0.745	0.743	0.674
Precision	0.759	0.852	0.932
F-measure	0.752	0.794	0.782

Table 3: Evaluation of the data produced by the game on the REF_{Eval} corpus (except punctuation).

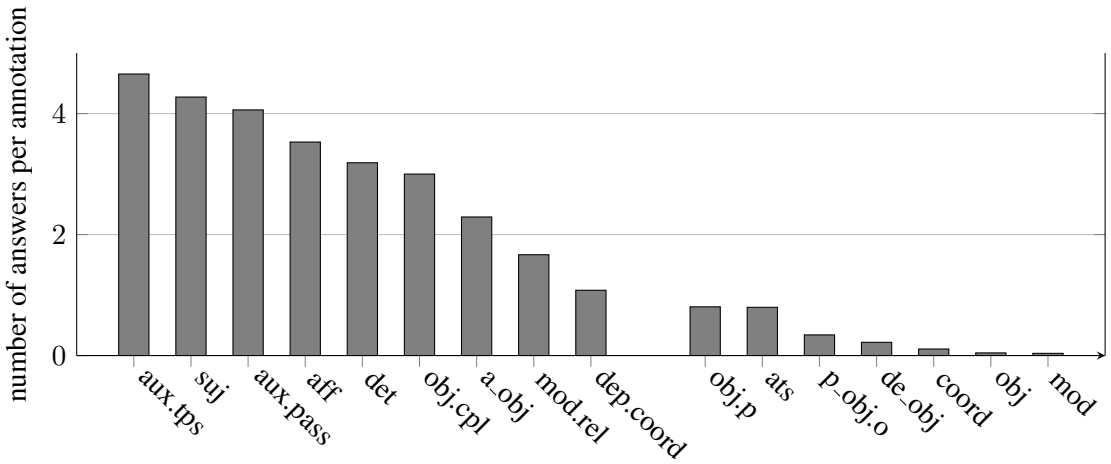


Figure 4: Density of answers for each relation on the REF_{Eval} corpus (a whitespace separates relations where density is greater than 1 from the others).

Note that the score of the annotations produced by the game is lower than that of the second parser. This difference comes about because some relations were not played by enough players, as already observed in Subsection 4.1. To explore in more detail how the players influence the corpus score, we must look at these values relation by relation.

First, 3,575 game items correspond to the REF_{Eval} corpus and only 2,644 game actions were performed by the player on these items (so the average density is 0.74). Moreover, as already observed in

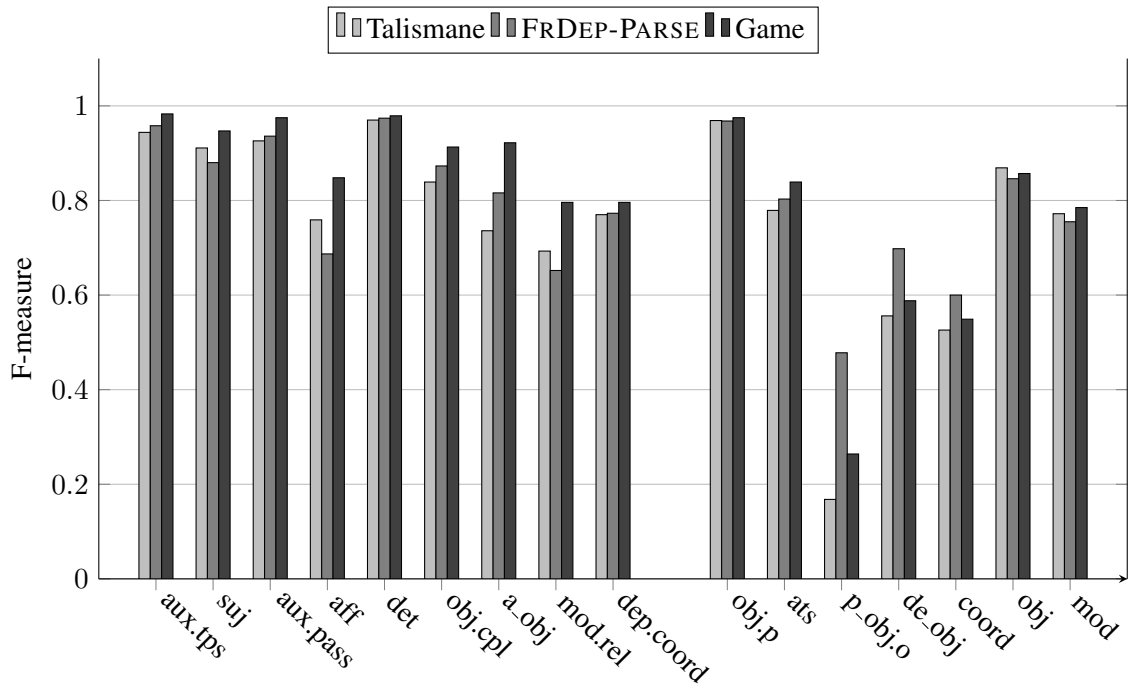


Figure 5: F-measures for the two parsers and the game, on each relation. (As in Figure 4, relations are split into two groups, with density greater than 1 on the left and density lower than 1 on the right.)

Figure 3, the distribution of the game actions is not at all homogeneous. If we discard relations for which the number of occurrences in the REF_{Eval} corpus is too low[18] to obtain significant results, the average density of game actions on game items ranges from 0.04 on **mod** (modifier) to 4.65 on **aux.tps** (tense auxiliary).

In Figure 4, we report the density of annotations on the corpus REF_{Eval} and Figure 5 gives, for the same relations in the same order, the values of the F-measure in the three settings (with Talismane in light gray, FRDEP-PARSE in a darker gray and finally the game export in dark gray).

For the relations where the density is higher than 1 (seen in the left hand part of the figures), we observe that the F-measure computed on the corpus from the game is always higher than that of the two parsers. From these experiments, we can conclude that we manage to obtain annotations with a quality significantly higher than that obtainable with a parser, provided that enough players participate. It is worth noting that amongst the relations which are densely played, some are considered as complex, such as **dep.coord** (the player has to find the head of the second conjunct of a coordination). Figure 2 shows an example of such a difficult case: the player has to read and to understand the whole sentence to give the right answer (it is a long distance dependency).

The next challenge is to induce players to annotate a wider variety of relations so as to increase the quality of the whole corpus. Relations with a very low density are either relations with a high number of occurrences (the REF_{Eval}corpus contains 1,874 **mod** relations) or relations which are less intuitive for the players. We will take this last factor into account and improve the documentation on these relations.

5 Conclusion

We have presented ZombiLingo, a game designed for a complex linguistic annotation task, namely dependency syntax. A first prototype of the game was released in July 2014 and an engineer started working on a production version in October 2015, completing the first version by the end of that year. As of July 2016, the game had enabled the production of more than 100,000 annotations for French, with a precision of 0.93. These results are very promising, especially for low-resourced languages. We still need to

[18]We have discarded relations with less than 25 occurrences in REF_{Eval}, namely **aux.caus** (causative auxiliary, 4 occurrences), **arg** (specific relation linking two parts of a range, 2 occurrences) and **dis** (dislocation, 1 occurrence).

fill some annotation gaps (e.g. for relations insufficiently played) and have developed for this purpose a new duel mode, which will allow senior players to compete with each other in the annotation of whole sentences.

The most difficult factor when using GWAPs is to find ways of attracting and keeping participants (Poesio et al., 2013), which requires a continuing communication effort. We have experienced "waves" of players following specific events we participated in or challenges we organized. We also attracted new players when advertising the game on social networks, but this effort must be regularly maintained and renewed.

The game source code is freely available on GitHub[19] under an CeCILL open-source license[20]. The code is designed to be easily adaptable to any human language: all messages are isolated from the code, which is Unicode compliant. We plan to adapt it to English and to a less-resourced language in the coming months.

The resource created for French is directly and freely available under a CC BY-NC-SA license from the game website[21] and is updated every night.

Acknowledgements

The development of ZombiLingo is funded by Inria, through an ADT grant. The French Ministry of culture has also helped us to communicate and publish results concerning ZombiLingo through two grants.

Above all, we want to thank all of the ZombiLingo players, without whom the resource would simply not exist! Special thanks to players JYA, Chouchou and Xohwohxo, who gave us valuable feedback on the game. We also wish to thank Djamé Seddah, for his help in building a solid state of the art and Guy Perrier, who, as Professor Frankenperrier, helped us to write the guidelines and to check errors in the reference. Finally, we want to thank the reviewers, who gave us interesting suggestions for improving this article.

References

Anne Abeillé and Benoît Crabbé. 2013. Vers un treebank du français parlé. In *Proceedings of TALN 2013 - 20ème conférence du Traitement Automatique du Langage Naturel*, Sables d'Olonne, France, June.

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for French. In Anne Abeillé, editor, *Treebanks*, pages 165 –187. Kluwer, Dordrecht.

Jason Baldridge and Miles Osborne. 2004. Active learning and the total cost of annotation. In *Proceedings of Empirical Methods in Natural Language Processing*, volume 15, pages 9–16.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2001. The prague dependency treebank: Three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*. Kluwer Academic Publishers.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Marie Candito and Djamé Seddah. 2012. Le corpus Sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical. In *Proceedings of Traitement Automatique des Langues Naturelles (TALN)*, Grenoble, France, June.

Marc Carmen, Paul Felt, Robbie Haertel, Deryle Lonsdale, Peter McClanahan, Owen Merkling, Eric Ringger, and Kevin Seppi. 2010. Tag dictionaries accelerate manual annotation. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Valletta, Malta, May.

Jon Chamberlain, Karën Fort, Udo Kruschwitz, Mathieu Lafourcade, and Massimo Poesio. 2013. Using games to create language resources: Successes and limitations of the approach. In Iryna Gurevych and Jungi Kim,

[19] `https://github.com/zombilingo`

[20] `http://www.cecill.info/licences/Licence_CeCILL_V2.1-en.html`

[21] See the bottom of the following page: `http://zombilingo.org/informations`.

editors, *The People's Web Meets NLP*, Theory and Applications of Natural Language Processing, pages 3–44. Springer Berlin Heidelberg.

Kenneth Church. 2011. A pendulum swung too far. *Linguistic Issues in Language Technology - LiLT*, 6.

Seth Cooper, Adrien Treuille, Janos Barbero, Andrew Leaver-Fay, Kathleen Tuite, Firas Khatib, Alex Cho Snyder, Michael Beenen, David Salesin, David Baker, and Zoran Popović. 2010. The challenge of designing scientific discovery games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, pages 40–47, New York, NY, USA. ACM.

Sandipan Dandapat, Priyanka Biswas, Monojit Choudhury, and Kalika Bali. 2009. Complex linguistic annotation - no easy way out! a case from bangla and hindi POS labeling tasks. In *Proceedings of the third ACL Linguistic Annotation Workshop*, Singapore.

Pascal Denis and Benoît Sagot. 2012. Coupling an annotated corpus and a lexicon for state-of-the-art pos tagging. *Language Resources and Evaluation*, 46(4):721–736.

Karën Fort, Gilles Adda, and Kevin Bretonnel Cohen. 2011. Amazon Mechanical Turk: Gold mine or coal mine? *Computational Linguistics (editorial)*, 37(2):413–420.

Bruno Guillaume and Guy Perrier. 2015. Dependency Parsing with Graph Rewriting. In *Proceedings of IWPT 2015, 14th International Conference on Parsing Technologies*, pages 30–39, Bilbao, Spain.

Neha Gupta, David Martin, Benjamin V. Hanrahan, and Jacki O'Neill. 2014. Turk-life in india. In *Proceedings of the 18th International Conference on Supporting Group Work*, GROUP '14, pages 1–11, New York, NY, USA. ACM.

Jirka Hana and Barbora Hladká. 2012. Getting more data - schoolkids as annotators. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 4049–4054, Istanbul, Turkey, May.

Barbora Hladká, Jirka Hana, and Ivana Luksová. 2014. Crowdsourcing in language classes can help natural language processing. In *Proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2014, November 2-4, 2014, Pittsburgh, Pennsylvania, USA*.

David Jurgens and Roberto Navigli. 2014. It's All Fun and Games until Someone Annotates: Video Games with a Purpose for Linguistic Annotation. *Transactions of the Association for Computational Linguistics (TACL)*, 2:449–464.

Firas Khatib, Frank DiMaio, Seth Cooper, Maciej Kazmierczyk, Miroslaw Gilski, Szymon Krzywda, Helena Zabranska, Iva Pichova, James Thompson, Zoran Popović, et al. 2011. Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature structural & molecular biology*, 18(10):1175–1177.

Anne Lacheret, Sylvain Kahane, Julie Beliao, Anne Dister, Kim Gerdes, Jean-Philippe Goldman, Nicolas Obin, Paola Pietrandrea, and Atanas Tchobanov. 2014. Rhapsodie: a prosodic-syntactic treebank for spoken french. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, May.

Mathieu Lafourcade, Nathalie Le Brun, and Alain Joubert. 2015. *Games with a Purpose (GWAPS)*. Wiley-ISTE, July.

Mathieu Lafourcade. 2007. Making people play for lexical acquisition. In *Proceedings of the 7th Symposium on Natural Language Processing (SNLP 2007)*, Pattaya, Thailand, December.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English : The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu, and Castelló Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of ACL 13*, Sofia, Bulgaria, August.

Massimo Poesio, Jon Chamberlain, Udo Kruschwitz, Livio Robaldo, and Luca Ducceschi. 2013. Phrase detectives: Utilizing collective intelligence for internet-scale language resource creation. *ACM Trans. Interact. Intell. Syst.*, 3(1):3:1–3:44, April.

Djamé Seddah and Marie Candito. 2016. Hard time parsing questions: Building a questionbank for french. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portoroz, Slovenia, May.

Djamé Seddah, Benoît Sagot, Marie Candito, Virginie Mouilleron, and Vanessa Combet. 2012. The French Social Media Bank: a Treebank of Noisy User Generated Content. In *Proceedings of COLING 2012 - 24th International Conference on Computational Linguistics*, Mumbai, India, December.

Arne Skjærholt. 2013. Influence of preprocessing on dependency syntax annotation: speed and agreement. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 28–32, Sofia, Bulgaria, August.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP 2008*, pages 254–263.

Assaf Urieli. 2013. *Robust French syntax analysis: reconciling statistical methods and linguistic knowledge in the Talismane toolkit*. Ph.D. thesis, Université de Toulouse II le Mirail, France.

Noortje Joost Venhuizen, Valerio Basile, Kilian Evang, and Johan Bos. 2013. Gamification for word sense labeling. In Katrin Erk and Alexander Koller, editors, *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Short Papers*, pages 397–403, Potsdam, Germany, March.

Luis von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, pages 319–326, New York, NY, USA. ACM.

Borrow a Little from your Rich Cousin: Using Embeddings and Polarities of English Words for Multilingual Sentiment Classification

Prerana Singhal and **Pushpak Bhattacharyya**
Dept. of Computer Science and Engineering
IIT Bombay, Maharashtra, India
{singhal.prerana,pushpakbh}@gmail.com

Abstract

In this paper, we provide a solution to multilingual sentiment classification using deep learning. Given input text in a language, we use word translation into English and then the embeddings of these English words to train a classifier. This projection into the English space plus word embeddings gives a simple and uniform framework for multilingual sentiment analysis. A novel idea is augmentation of the training data with polar words, appearing in these sentences, along with their polarities. This approach leads to a performance gain of 7-10% over traditional classifiers on many languages, irrespective of text genre, despite the scarcity of resources in most languages.

1 Introduction

Sentiment Analysis deals with extraction of opinion polarity from texts. Extensive work has been done in this field over the past years, mainly for English. Subsequently, rich English resources like SentiWordNet (Esuli and Sebastiani, 2006) and pre-trained word embeddings (Mikolov et al., 2013a) are available publicly. However, lack of rich resources and annotated corpora in other languages like Dutch, Russian or Hindi makes it difficult to analyze texts with as good accuracy as that in English.

Deep learning models have achieved astonishing results in several fields like Speech Recognition and Computer vision, and have shown promising results when used for several NLP tasks (like Convolutional Neural Networks for Sentence Classification (Kim, 2014), LSTMs for tweet classification (Wang et al., 2015) and Recursive Deep Models for Sentiment Analysis (Socher et al., 2013)). A key feature of deep learning models, which seems to attract NLP researchers, is their lack of demand for manual feature engineering, unlike other classical machine learning algorithms (SVM, *etc*.) (Pang et al., 2002).

Multilingual Sentiment Analysis has been a challenging, yet an important area of research since a long time, mainly involving other NLP tools like Sentence-level Machine translation (Joshi et al., 2010; Wan, 2009), Machine Translation with SentiWordNet scores (Denecke, 2008), semantic orientation calculator (Brooke et al., 2009), and Wordnets (Balamurali et al., 2012) to serve the purpose. Machine learning methods have been used and evaluated on different sets of languages (Boiy and Moens, 2009; Seki et al., 2010; Seki et al., 2008), which involve many constraints and manual functionalities. In addition, strategies to build a multilingual corpus (Schulz et al., 2010) have been devised which, quite frankly, are not possible for large number of languages, owing to the diversity involved.

In this paper, we present a simple approach to multilingual sentiment classification which: (a) poses minimal restrictions on the language or the text genre to be used, (b) has minimal demands for preprocessing tools, and (c) shows no aversion to the small size of datasets or inadequacy of resources in any language. **Our approach uses deep learning models for sentiment classification in a given language by *borrowing* word-embeddings and word polarities from the *rich cousin* English.**

The main idea is to combine existing lexical resources and neural network classification techniques and provide an effective solution to the problem of multilingual sentiment classification. Although the

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3053–3062, Osaka, Japan, December 11-17 2016.

individual components (polar words (Pang et al., 2002) and word embeddings (Wang et al., 2015)) have been used by others before, **the novelty of this paper lies in the method of integrating *known components* and *available resources*, for *languages with insufficient resources*.**

2 Motivation

Multilingual Sentiment Analysis faces several challenges including (a) linguistic variations and (b) lack of sufficient resources for supervised classification. We need a simple classification system which can perform efficiently on a dataset, independent of the language or the text genre. Quite often, the size of the annotated corpus available for a particular language and text genre is generally not sufficient enough, in fact, far too small, to be helpful on its own. Hence, we propose to **use deep learning models and leverage publicly available word embeddings and polar words of *English* for sentiment classification** to overcome the limitations in a multilingual framework.

2.1 Why Deep Learning?

Classical machine learning algorithms like SVM require (a) *pre-processing* and (b) *manual feature engineering*, (Pang et al., 2002) both of which vary across languages as well as text genres. Hence, building a classification system for a new language is cumbersome and may not be as effective for all languages. We need a model, which can train effectively on any dataset, irrespective of its language or text characteristics, with minimal or no manual adjustments across datasets. Hence, deep learning models like Convolutional neural networks (Collobert et al., 2011) assume importance, since they are well-known to have no such text dependent constraints.

2.2 Why English word-embeddings?

Deep learning NLP models require word representations (containing context information) as input. One way to do so is to randomly initialize the word vectors and trust the sentiment classification model itself to learn the word representations, besides the network parameters. However, this requires a large annotated corpus, which is difficult to obtain in most languages. The other way is to train a suitable deep learning model (Collobert et al., 2011; Mikolov et al., 2013a) on a raw corpus in that language and then use the obtained embeddings of these *in-language words* as input to the sentiment classification model. However, learning context-rich word embeddings in any language requires large datasets, generally of the order of billions of words, thereby eating up a lot of time as well as system resources.

Hence, the pre-trained word embeddings of *English* prove to be useful, which have already been obtained by training suitable models on billions of data. Their context-rich information can be utilized to compensate for the small size of the available corpora in a language. An interesting property of context-rich word embeddings is that they capture linguistic regularities (Mikolov et al., 2013b), including contextual similarities. Hence, similar words or synonyms will have closer word vectors (measured by cosine distance). Thus, for the purpose of sentiment polarity classification (with no intensity segregation), all synonyms will contribute to the same polarity in a similar manner. So, one can pick any one of the synonyms (say, one of *amazing, splendid, spectacular, etc*.) and it will not affect the underlying sentiment (say positive or negative or neutral) of the text.

We propose to *obtain a mapping between the in-language words and the English word-embeddings through word-to-word translation*. For this, any publicly available (decent) bilingual dictionary or translation tool (like Google translate) can be used. The in-language words are translated to English individually, *solely for the purpose of obtaining a mapping to the pre-trained English word vectors*; hence, the orientation of the input texts is not disturbed.

2.3 Why English polar words?

A small change in the choice of words, or the order in which the words occur in a piece of text, may change the whole opinion underlying the text. For instance, the sentence "*This is not good*" conveys negative sense, as opposed to the positive sense conveyed by "*This is good*", with a single word *not* reversing

the polarity. Also, the sentiment polarity of the sentence "*He does not seem bad, he is good*" is clearly opposite to that of "*He does not seem good, he is bad*", in spite of exactly the same words being used in both cases. So the classification system needs to *see all these patterns* for making correct predictions on *unseen* data.

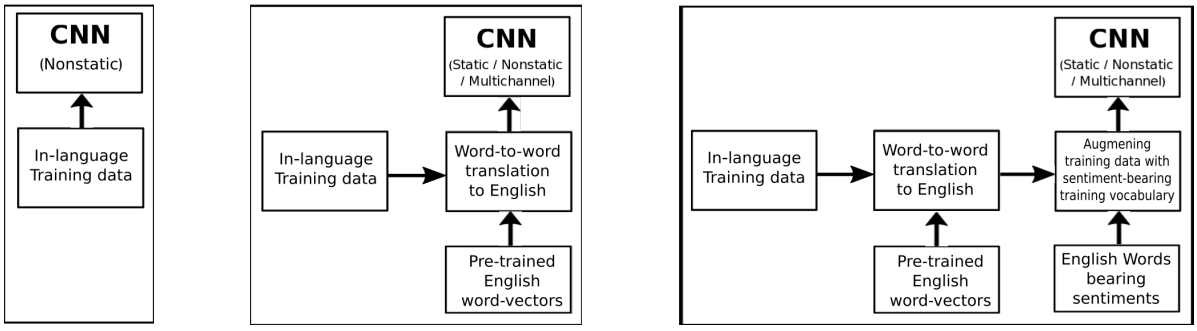
The point is, if enough instances are not provided to the classification system for training, it is likely to learn wrong patterns. For example, if the system has seen "*This may be nice but I do not like it.*" as the only training example with the word *nice*, it tends to associate the word *nice* with negative sentiment and is likely to wrongly predict an unseen sentence, say "*She seems nice to me.*", as negative. Generally, a large annotated corpus is able to overcome this anomaly, as the network gets enough data to learn the correct linguistic patterns. However, when the labeled dataset is small in size, as is often the case in most languages, this problem adversely affects the classification performance.

To overcome this hurdle, one way is to make use of some list of frequently used sentiment-polar words. The idea is to familiarize the network with the fact that the text "*This may be nice but I do not like it*" reflects negative opinion but "*nice*" portrays positive sentiment. This improves the chances of correct prediction, as now the network is able to learn that the word *nice* is generally used for positive opinion but when used in some particular context (say, with discourse particle *but* or negation element *not*), it displays negative sentiment.

Several rich English resources like SentiWordNet (Esuli and Sebastiani, 2006) or list of positive/negative English vocabulary (Hu and Liu, 2004) are publicly available. We propose to make use of this information for supervised sentiment classification in a multilingual scenario, *by augmenting sentiment bearing words with their polarities to the annotated training corpora*.

3 Proposed Method

We use a deep learning model like Convolutional neural network as our classification system. We use randomly initialized word-vectors and no other resources in our baseline model. For example, a simple CNN-Rand (Non-static) model (Kim, 2014) can be used to work as our baseline. The proposed method for sentiment classification, as depicted by the block diagram (Figure 1), can be divided into two stages (Figures 1b and 1c) to be applied on top of this base-line (Figure 1a).



(a) Baseline (b) 1st Stage (c) 2nd Stage

Figure 1: Block Diagrams of the Deep Learning Classification Methods

3.1 First Stage: *Mapping in-language texts to English Word Embeddings*

Given text in one language, we translate each word into English using bilingual dictionary or translation system like Google translate. Then, we initialize the word vectors using the corresponding English word embeddings, obtained from the existing list of pre-trained English word vectors. In case the in-language word fails to get translated, it is *transliterated* into English and its word-vector is randomly initialized during the training phase.

To illustrate our approach, let us consider dummy training data in Hindi:

1. Original Hindi text :: हम इस धारणा के प्रभाव को महसूस करते हैं, लेकिन उत्तेजित और प्रेरित नहीं होते।
   Transliteration :: *Hum is dharnaa ke prabhaav ko mehsoos karte hain, lekin uttejit aur prerit nahi hote.*
   Meaning in English :: ***We feel the effects of this notion, but do not get excited and motivated.***
   Label :: Negative
2. Original Hindi text :: फिल्म का संगीत कमजोर है।
   Transliteration :: *Film ka sangeet kamzor hai.*
   Meaning in English :: ***The film's music is weak.***
   Label :: Negative
3. Original Hindi text :: स्क्रिप्ट की बंदिशों के बावजूद अभिनेताओं ने लाजवाब काम किया है।
   Transliteration :: *Script ki bandishon ke baavajud abhinetaon ne laajawaab kaam kiya hai.*
   Meaning in English :: ***Despite the restrictions of the script, the actors did an amazing job.***
   Label :: Positive

Now, we translate (or transliterate) the training data on **word-level** to English and **map them to the corresponding English word-embeddings**. This results in the following as our training data:
*(**<w>** implies embedding of the English word **w** to which the in-language word has been translated)*

1. **Instance ::** ***<we> <this> <assumption> <of> <effect> <to> <are> <,> <but> <excited> <and> <inspired> <no> <there> <.>*** (Label :: Negative)
2. **Instance ::** ***<film> <of> <music> <weak> <is> <.>*** (Label :: Negative)
3. **Instance ::** ***<script> <of> <restrictions> <of> <despite> <actors> <has> <excellent> <work> <the> <is> <.>*** (Label :: Positive)

Once the word-vectors have been initialized, the network can be trained on the training data (Figure 1b), which now contains concatenated word vectors as training instances with their corresponding sentiment-labels. The word-to-word translation is done mainly *to obtain a mapping* between the in-language words and the English word embeddings. *The word order in original text sequence is not disturbed* and hence, there is neither any chaos nor any problems in handling phenomenon like negation. As compared to the baseline, this technique is expected to work better, because, unlike in the former case, it now has access to the extensive context information offered by the pre-trained English word embeddings, which can be exploited to improve the training procedure on *scarce* datasets.

3.2 Second Stage: *Augmenting training data with English Polar Words*

After obtaining the word vector mappings, we choose some authentic English resource like SentiWordNet or list of common positive-negative words (Hu and Liu, 2004), to form a list of frequently used polar sentiment-bearing lexicons. Then, once the in-language text is mapped to English on word level, the training data vocabulary is matched against this list and the intersection of the two is appended to the training data with their polarities as their labels. The idea is *to augment training data with polar words that have occurred in the training texts.*

In our example, the training data will now consist of following instances:
*(**<w>** implies embedding of the English word **w** to which the in-language word has been translated)*

1. **Instance ::** ***<we> <this> <assumption> <of> <effect> <to> <are> <,> <but> <excited> <and> <inspired> <no> <there> <.>*** (Negative)
2. **Instance ::** ***<film> <of> <music> <weak> <is> <.>*** (Negative)
3. **Instance ::** ***<script> <of> <restrictions> <of> <despite> <actors> <has> <excellent> <work> <the> <is> <.>*** (Positive)
4. **Instance ::** ***<excited>*** (Positive)
5. **Instance ::** ***<inspired>*** (Positive)

6. **Instance :: *<weak>*** (Negative)
7. **Instance :: *<restrictions>*** (Negative)
8. **Instance :: *<excellent>*** (Positive)

Now the word vectors are initialized as before and the network is trained as usual on *this extended training data* (Figure 1c). This technique is expected to further enhance the performance of the classification system, as it tries to fill gaps in the sentiment-related information of the small training datasets.

4 Datasets and Experimental Setup

We perform experiments on datasets in two Indian languages[1], and Russian and six European languages[2]. Neutral labels (if any) have been removed from these datasets for the purpose of *binary sentiment* (positive-negative) classification.

The datasets are essentially movie reviews in Hindi (Joshi et al., 2010), tourism reviews in Hindi and Marathi (Balamurali et al., 2012), and tweets from different contexts in Dutch, French, Spanish, Italian, German, Portuguese and Russian languages (Araujo et al., 2016). These datasets reflect the **diversity in terms of language family** (Indian languages are not as close to English as European languages), **text characteristics and length of the texts** (reviews are long and grammatically sane while tweets are short irregular piece of texts) as well as the **relatively small size of the labeled corpora**. Relevant information about the datasets have been summarized in Table 1.

Dataset Language	Average Text Length	Vocabulary Size	Dataset Size		
			#pos	#neg	Total
Reviews in Indian Languages					
Hindi (Movie)	27	1600	127	125	252
Hindi (Tourism)	128	3601	98	100	198
Marathi (Tourism)	89	3766	75	75	150
Tweets in Russian and European Languages					
Russian	15	8021	1145	1188	2333
Dutch	21	1258	88	63	151
French	17	1800	159	160	319
German	13	1254	143	95	238
Portuguese	16	2472	297	213	510
Spanish	21	5092	683	350	1033
Italian	19	8429	820	1422	2242

Table 1: Summary Statistics for the Binary Labeled Datasets in different languages

4.1 Neural Network Architecture

For the purpose of our experiments on all datasets, we use convolutional neural network (CNN), which consists of a convolutional layer with filter windows of sizes 3, 4 and 5, a feature map of size 50 for each of the filters and sigmoid as the activation function, followed by max-pooling and an output layer with softmax as activation function. For the network, we choose negative log likelihood as the error function, learning rate to be 0.95, dropout rate to be 0.4 and number of training epochs to be 30. We train the model through stochastic gradient descent with the Adadelta update rule (Zeiler, 2012).

The hyper-parameters of the CNN model have been chosen via a grid search on the *hindi-movie-review* dataset. Three different variants of this CNN model have been tried on the datasets, as shown by Yoon Kim (2014). which are:

- **CNN Static (C-S) ::** Here, pre-trained word-vectors are used and these undergo no change during training, *i.e.*, only the parameters of the network are learned through back-propagation and not the word embeddings.

[1]available at `http://www.cfilt.iitb.ac.in/Sentiment_Analysis_Resources.html`
[2]available at `http://homepages.dcc.ufmg.br/~fabricio/sentiment-languages-dataset/`

- **CNN Non-Static (C-N) ::** Here, either pre-trained word-vectors are used and/or word vectors are randomly initialized, and during training, apart from the parameters of the network, the word-vectors are also tuned through back-propagation in order to learn word-embeddings for the specific task of sentiment classification.
- **CNN Multi-Channel (C-M) ::** Here, two sets of pre-trained word-vectors are used with the convolutional filters applied separately on each, and during training, apart from the parameters of the network, one set of the word-vectors are tuned through back-propagation (non-static channel) while the other set undergoes no change (static channel).

4.2 Resources and Tools used

For our experiments, we use Google translate[3] for word-to-word translation from a given language to English. The English word embeddings we use are pre-trained vectors trained on a part of Google News dataset (about 100 billion words) [4]. These are 300-dimensional vectors for approximately 3 million words and phrases. We use the list of English words expressing sentiment polarities[5] compiled by Bing Liu and Minqing Hu (2004), which consists of approximately 6800 positive and negative words. We use these resources for our experiments solely because of their richness and easy availability; however, other alternatives can also be used.

	Traditional Classifier	**Deep Learning Models (Convolutional Neural Networks)**						
		Baseline	Our Approach (Stage I)			Our Approach (Stages I & II)		
Dataset Language	**SVM (unigrams)**	**C-N Lang**	**C-S E-wv**	**C-N E-wv**	**C-M E-wv**	**C-S E-wv-pw**	**C-N E-wv-pw**	**C-M E-wv-pw**
Reviews in Indian Languages								
Hindi (Movie)	71.6	73.4	71.5	76.6	74.7	76.2	**80.2**	78.4
Hindi (Tourism)	80.3	80.3	88.1	84.4	85.1	**88.9**	87.1	87.3
Marathi (Tourism)	95.7	93.3	92.4	93.4	88.8	95.8	95.7	**96.1**
Tweets in Russian and European Languages								
Russian	59.7	63.0	69.5	73.2	74.2	71.4	74.2	**74.9**
Dutch	67.0	63.6	75.9	72.7	68.8	77.5	77.0	**78.1**
French	69.7	71.3	76.1	75.2	77.2	79.5	80.4	**81.8**
German	63.3	67.7	77.2	78.0	77.9	**81.1**	80.9	79.5
Portuguese	66.4	67.7	78.3	73.9	75.3	**79.9**	77.9	79.2
Spanish	72.2	75.6	82.9	83.2	83.0	84.8	83.8	**85.2**
Italian	63.7	64.5	74.1	73.1	74.3	**75.2**	74.9	75.1

Table 2: Classification Performance results (Average F-scores) of the models in different languages

4.3 Experimental Configurations

We perform two-fold validation of five repeats on the datasets, where the configurations of training and test documents are randomly chosen for each repeat. The data is not subjected to any tuning prior to training/testing, apart from changing all (English) words to lowercase and inserting space between letters and punctuations. This configuration is maintained across all datasets and model variants, in order to maintain uniformity while comparing results.

To compare the performance of the deep learning models, we also apply SVM (Pang et al., 2002) classifiers (with unigram as features) on the datasets. We perform experiments for the following models:

- Classical Machine Learning (Baseline) Classifier Models:
  1. **SVM (unigram)::** SVM (with unigram as features) on the original in-language training data.

[3] https://translate.google.co.in/

[4] https://code.google.com/archive/p/word2vec/

[5] http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar

- In-language Deep Learning (CNN) Models:
  1. **C-N Lang::** CNN model in non-static mode with the *original in-language training data*, and randomly initialized word-vectors.
- Proposed Deep Learning (CNN) Models (only Stage-I Approach :: *mapping to English Word Vectors (E-wv)*):
  1. **C-S E-wv::** CNN model in static mode with the training data mapped to pre-trained English word embeddings through word-level translation.
  2. **C-N E-wv::** CNN model in non-static mode with the training data mapped to pre-trained English word embeddings through word-level translation.
  3. **C-M E-wv::** CNN model in multi-channel mode with the training data mapped to pre-trained English word embeddings through word-level translation.
- Proposed Deep Learning (CNN) Models (Stage-I and Stage-II Approach :: *mapping to English Word Vectors and augmenting training data with Polar Words (E-wv-pw)*):
  1. **C-S E-wv-pw::** CNN model in static mode with the training data mapped to pre-trained English word embeddings through word-level translation and augmented with English polar words.
  2. **C-N E-wv-pw::** CNN model in non-static mode with the training data mapped to pre-trained English word embeddings through word-level translation and augmented with English polar words.
  3. **C-M E-wv-pw::** CNN model in multi-channel mode with the training data mapped to pre-trained English word embeddings through word-level translation and augmented with English polar words.

5 Results

In Table 2, we report F-score values (mean of positive and negative class F-scores) of all our models on the different datasets.

Although an increase in the performance of the deep learning models, aided by the English word-embeddings and word-polarities, was expected, **the magnitude of the gains encountered is overwhelming, in spite of the incredibly small size of the annotated corpora in many cases.** In fact, the improvement in performance is almost consistent across all language datasets in spite of the varied nature of the texts (movie reviews, tourism reviews and tweets in different languages).

The best results, **more than 80%** in many cases, are reported in the last three columns of Table 2, which are the variants of the CNN models, utilizing the two stages of our proposed approach. These are not only better than the CNN baseline (C-N Lang) but also **show astonishing performance gains over the SVM model across all datasets, as high as 10% in some cases**.

6 Observations and Discussions

It is evident that across all languages and datasets, our proposed approach of using deep learning models (one of the variants of CNN in our case), leveraging the pre-trained word-embeddings and the polar words of English, consistently performs better than the classical model (SVM) as well as the CNN baseline.

The most significant property exhibited by our proposed system is effective handling of (a) unknown words and (b) scarcity of datasets; two challenges frequently faced by the task of sentiment classification in most languages.

6.1 Unknown Words

Unknown words pose a great challenge to almost all NLP tasks including Sentiment Analysis. They are infamous for bringing the performance of the NLP systems considerably down, as these can play no role to predict the label of unseen data. Table 3 shows the average number of unknown words encountered during

the testing phase with and without being mapped to English word embeddings. Clearly, *when relying solely on the in-language training data, the number of unknown words encountered is more than 50% of the test vocabulary size* in most cases. This is bound to decrease the performance of the classification system. However, *leveraging the context information of the pre-trained English word embedding space helps to 'know' a large fraction of the unknown words*. Consequently, the number of unknown words reduce considerably after applying our proposed method and Table 3 clearly backs up this argument. Furthermore, this 'knowledge' is useful for the task at hand, as clearly reflected by the performance gains by our system (Table 2).

Datasets	**Average Vocabulary Size in test data**	Average number of Unknown Words encountered in test data	
		Without using English words	**Using English words**
Reviews in Indian Languages			
Hindi (Movie)	1162	658	95
Hindi (Tourism)	3030	1710	284
Marathi (Tourism)	3240	2283	506
Tweets in Russian and European Languages			
Russian	6757	5149	1262
Dutch	807	589	115
French	1203	877	164
German	855	612	104
Portuguese	1706	1197	300
Spanish	3654	2510	826
Italian	6289	4416	1442

Table 3: Statistics of Unknown words with and without being mapped to English word embeddings

6.2 Scarcity of data

One basic requirement of a supervised classification model is a substantial amount of data for training the network, which often becomes a major challenge for multilingual sentiment analysis. In-language classification results in far more amount of unseen instances and/or words than that which can be handled, because of which the baseline models do not perform very well on the scarce datasets. This shortcoming is somewhat overcome by our proposed method of *projecting the in-language text to the English word space and augmenting the training data with polar words*. Table 2 substantiates our claim: we see more than 80% accuracy for most of the datasets, in spite of their small size. These numbers are statistically significant too, despite the small size of the datasets, not only because the results show consistent improvements over several datasets which is *not a coincidence*, but also, large annotated corpora are *practically* not available for most languages.

7 Conclusion

In this work, we established that, even with a simple deep learning classification model, and easy usage of publicly available word embeddings and polar words of English, appreciable results can be obtained for *multilingual sentiment classification*. **The main idea of our proposed approach is to map the in-language words with English word embeddings and augment the training dataset with polar words**. Although the individual components like polar words and word embeddings have been utilized before, the proposed approach, as a whole, is substantially different from any of the previous works. The *novelty* of this paper lies in the method of *borrowing known components from the 'rich' language English, and integrating them using deep learning models, for the task of multilingual sentiment classification.*

The experimental results show more than 80% performance F-score values *with as high as 10% performance gain* over the classical models in many cases. These observations substantiate the viability of our proposed approach in handling the key issues of multilingual sentiment classification, namely, diversity of texts and scarcity of datasets across languages.

We currently targeted binary sentiment classification for different languages in this paper. We show that **exploiting the embeddings and sentiment polarities of English words (without relying on any complex tools), and effectively applying deep learning models, is a viable approach to sentiment classification in a multilingual setup**. This work can further be extended to multi-class sentiment classification and possibly aspect classification in different languages. Also, further experimentations can be conducted with other publicly available resources, datasets and tools as well as other deep learning neural network configurations for multilingual sentiment analysis.

References

Matheus Araujo, Julio Reis, Adriano Pereira, and Fabricio Benevenuto. 2016. An evaluation of machine translation for multilingual sentence-level sentiment analysis. In *Proceedings of the 31st ACM symposium on Applied computing*.

AR Balamurali, Aditya Joshi, and Pushpak Bhattacharyya. 2012. Cross-lingual sentiment analysis for indian languages using linked wordnets. In *Proceedings of COLING 2012*, pages 73–82.

Erik Boiy and Marie-Francine Moens. 2009. A machine learning approach to sentiment analysis in multilingual web texts. *Information retrieval*, 12(5):526–558.

Julian Brooke, Milan Tofiloski, and Maite Taboada. 2009. Cross-linguistic sentiment analysis: From english to spanish. In *RANLP*, pages 50–54.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Kerstin Denecke. 2008. Using sentiwordnet for multilingual sentiment analysis. In *Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on*, pages 507–512.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC*, volume 6, pages 417–422.

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

Maria Holmqvist, Sara Stymne, Jody Foo, and Lars Ahrenberg. 2009. Improving alignment for smt by reordering and augmenting the training corpus. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 120–124.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Aditya Joshi, AR Balamurali, and Pushpak Bhattacharyya. 2010. A fall-back strategy for sentiment analysis in hindi: a case study. In *Proceedings of the 8th ICON*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of ACL*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on EMNLP*, page 1746–1751.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on EMNLP-Volume 10*, pages 79–86.

Julia Maria Schulz, Christa Womser-Hacker, and Thomas Mandl. 2010. Multilingual corpus development for opinion mining. In *LREC*.

Yohei Seki, David Kirk Evans, Lun-Wei Ku, Le Sun, Hsin-Hsi Chen, Noriko Kando, and Chin-Yew Lin. 2008. Overview of multilingual opinion analysis task at ntcir-7. In *NTCIR*.

Yohei Seki, Lun-Wei Ku, Le Sun, Hsin-His Chen, and Noriko Kando. 2010. Overview of multilingual opinion analysis task at ntcir-8. In *Proc. of the 7th NTCIR Workshop*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, volume 1631, pages 1631–1642.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 235–243.

Xin Wang, Yuanchao Liu, Chengjie Sun, Baoxun Wang, and Xiaolong Wang. 2015. Predicting polarities of tweets by composing word embeddings with long short-term memory. In *Proceedings of the 53rd Annual Meeting of ACL and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1343–1353.

Bin Wei and Christopher Pal. 2010. Cross lingual adaptation: an experiment on sentiment classifications. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 258–262.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *CoRR*.

A Character-Aware Encoder for Neural Machine Translation

Zhen Yang, Wei Chen*, Feng Wang, Bo Xu
Institute of Automation, Chinese Academy of Sciences
No.95 Zhongguancun East Road
{yangzhen2014, wei.chen.media, feng.wang, xubo}@ia.ac.cn

Abstract

This article proposes a novel character-aware neural machine translation (NMT) model that views the input sequences as sequences of characters rather than words. On the use of row convolution (Amodei et al., 2015), the encoder of the proposed model composes word-level information from the input sequences of characters automatically. Since our model doesn't rely on the boundaries between each word (as the whitespace boundaries in English), it is also applied to languages without explicit word segmentations (like Chinese). Experimental results on Chinese-English translation tasks show that the proposed character-aware NMT model can achieve comparable translation performance with the traditional word based NMT models. Despite the target side is still word based, the proposed model is able to generate much less unknown words.

1 Introduction

Neural machine translation conducts end-to-end translation with a source encoder and a target decoder, producing promising results (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014). With the emerging of the attention-based encoder-decoder model, NMT has achieved comparable even better translation performance with the traditional statistical machine translation (SMT) (Bahdanau et al., 2014; Ranzato et al., 2015; Shen et al., 2015; Tu et al., 2016). The success of NMT lies in its strong ability of composing the global context information. However, as a newly approach, the NMT model has some flaws and limitations that may jeopardize its translation performance (Luong et al., 2014; Sennrich et al., 2015; He et al., 2016). One of the most glaring limitations is that the NMT model is weak in handling the rare and out-of-vocabulary (OOV) words, since the NMT system usually uses the top-N (30000-50000) frequent words in the training corpus and regards other words as unseen words (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014; Cohn et al., 2016). Two different kinds of approaches have been proposed to handle the OOV problem in NMT: ***vocabulary-specific*** approaches and ***unit-specific*** approaches.

The ***vocabulary-specific*** approaches seek to cover more words by using a larger vocabulary (Mnih and Kavukcuoglu, 2013; Cho et al., 2015) or using an identity translation dictionary in a post-processing step (Luong et al., 2014). Intuitively, these approaches can alleviate the OOV problems to a certain extent only if the vocabulary can be expanded large enough. However, these approaches are incapable of solving the OOV problems completely since the vocabulary is always limited.

As opposed to ***vocabulary-specific*** approaches, the ***unit-specific*** approaches try to use more fine-grained processing units than words, like sub-word units (Sennrich et al., 2015) or even character-level units (Ling et al., 2015b; Chung et al., 2016). Regarding the character as the basic processing unit is a new trend in the field of NLP and the character-level models have been widely used in NLP tasks (Ling et al., 2015a; Zhang et al., 2015; Golub and He, 2016). Developing character-level NMT models is attractive for multiple reasons. Firstly, it opens the possibility for models to generate unseen source words, since each word can be composed from different characters. Secondly, the vocabulary size of the

*Wei Chen is the corresponding author of this paper

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3063–3070, Osaka, Japan, December 11-17 2016.

model can be reduced dramatically as only the characters need to be modeled explicitly. This enables the character-level NMT model to solve many scalability issues, both in terms of computational speed and memory requirements. Finally, as each character occurs frequently in the training corpus, all of the character embeddings are able to get full trained. Hence they represent their corresponding characters very well. However, in the word-based NMT, the word embeddings for rare words, which hardly occur in the training corpus, are absent of enough training. Based on the state-of-the-art attention based encoder-decoder framework, some character-level NMT models have been proposed recently. (Chung et al., 2016) focus on representing the target side as a character sequence with a bi-scale recurrent neural network. In (Ling et al., 2015b), a character-based word representation model is proposed in the source side. (Luong and Manning, 2016) proposes a hybrid architecture for NMT that translates mostly at the word level and consults the character components for rare words when necessary. Most of the works mentioned above apply the bidirectional RNN to compose the word representation from its characters. Hence, its necessary to know the boundary between two words beforehand. These models are applicable for languages in which the words are segmented with explicit boundaries, such as English, French and etc.

In this work, we propose a novel character-aware NMT model that learns to encode at the character level. On the use of row convolution, the proposed model can be applied to the language which has no explicit word segmentations, like Chinese. We still represent the target side as a sequence of words. This paper has two main contributions:

- We propose a simple and novel NMT model which views the input as sequences of characters. We firstly rule out the word segmentation processing step for languages without explicit word segmentations.
- We introduce several different row convolution methods and investigate their effectiveness in NMT. Row convolution is a newly-emerged technique and has shown its great effectiveness in speech recognition (Amodei et al., 2015).

Experimental results show that contrarily to previous belief, the proposed character-aware NMT model can generate results on par with the word-based NMT models. The rest of this paper is organized as follows. Section 2 describes related works. In Section 3, we propose our character-aware NMT model. Experiments and results are described in Section 4. We conclude in Section 5.

2 Related works

In this section, we describe the basis of this work: the attention-based encoder-decoder NMT model and the row convolution method.

2.1 Attention-based encoder-decoder

This subsection briefly describes the attention-based NMT (RNNsearch) (Bahdanau et al., 2014), on which the character-aware NMT model is built. The RNN search model simultaneously conducts dynamic alignment and generation of the target sentence and it produces the translation sentence by generating one target word at every time step. Given an input sequence $\boldsymbol{x} = (x_1, \ldots, x_{T_x})$ and previous translated words $(y_1, \ldots, y_{i-1})$, the probability of next word y_i is:

$$p(y_i|y_1, \ldots, y_{i-1}, \boldsymbol{x}) = g(y_{i-1}, s_i, c_i) \tag{1}$$

where s_i is an decoder hidden state for time step i, which is computed as:

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \tag{2}$$

Here f and g are nonlinear transform functions, which can be implemented as long short term memory network(LSTM) or gated recurrent unit (GRU), and c_i is a distinct context vector at time step i, which is

calculated as a weighted sum of the input annotations h_j:

$$c_i = \sum_{j=1}^{T_x} a_{i,j} h_j \tag{3}$$

where h_j is the annotation of x_j from a bidirectional RNN. The weight a_{ij} for h_j is calculated as:

$$a_{i,j} = \frac{exp(e_{ij})}{\sum_{t=1}^{T_x} exp(e_{i,t})} \tag{4}$$

where

$$e_{i,j} = v_a tanh(W s_{i-1} + U h_j) \tag{5}$$

The mechanism of attention in RNN search makes the decoder focus on relevant words in the source sentence when generating the target word. The graphical illustration of the RNN search model is depicted in Fig.1(a).

2.2 Row convolution

(Amodei et al., 2015) firstly proposes the row convolution, which is used to look forward for a small portion of future information at the current time-step. Suppose at time-step t, the input h_t is a d-dimensional continuous vector and a future context of τ steps is considered. The model gets a feature matrix $h_{t:t+\tau}$ of size $d \times (\tau + 1)$. A parameter matrix of the same size as $h_{t:t+\tau}$ is defined as W. The activation r_t at the time-step t is computed as:

$$r_{t,i} = \sum_{j=1}^{\tau+1} W_{i,j} h_{t+j-1,i}, for 1 \leq i \leq d \tag{6}$$

Since the convolution-like operation in Eq.6 is row oriented for both W and $h_{t:t+\tau}$, it is called row convolution.

3 The character-aware NMT model

In this section, we describe the proposed character-aware NMT model in detail. Fig.1(b) is the graphical illustration of the proposed model. The basic architecture is a character-level encoder, which composes the input character embedding and its context embedding into the corresponding word-level representation.

3.1 Model overview

In a slightly generalized sense, the proposed character-aware NMT model is still an encoder-decoder. The encoder transformed the source sequence into the vector representation, which is then read by decoder to generate the output sequence.

Encoder The encoder in the proposed model is utilized in the character level. Specifically, the model is designed to compose the word-level information from the input sequence of characters. Compared to the RNN-encoder in (Bahdanau et al., 2014), there are two important differences

- **Context Computing** The proposed model builds a row convolution layer to compute the context vector for the current input character. The context vector preserves the context information which guides how to compose the word-level information.
- **Character composing** A character composing layer is used to compose the word-level information from the current input character and its corresponding context vector. The word-level information is then fed as input to the bidirectional RNN.

Decoder An RNN that reads the hidden variables of the encoder and predicts the target sequence. It is almost the same with the canonical RNN-decoder in (Bahdanau et al., 2014).

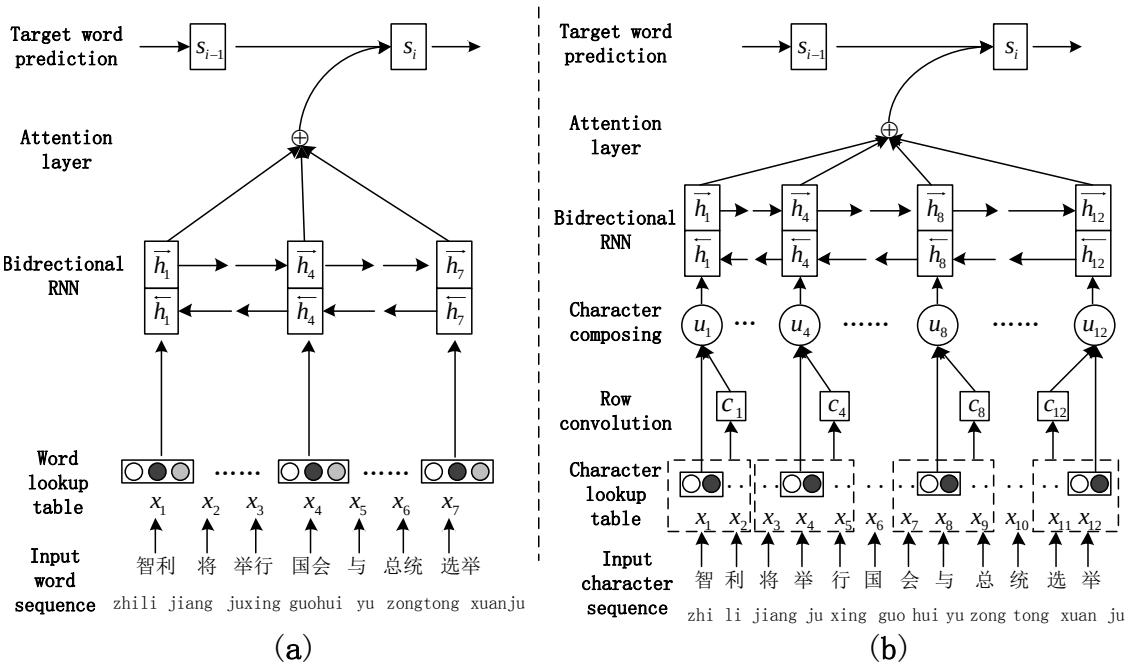


Figure 1: The graphical illustration of the proposed model. (a) is the traditional attention-based encoder-decoder model proposed by (Bahdanau et al., 2014). (b) is the proposed character-aware NMT model in this paper. Both of the two models try to translate the Chinese sentence "zhi li jiang ju xing guo hui yu zong tong xuan ju". The traditional attention-based model needs to segment the input sentence into Chinese words first. However, our character-aware model encodes the characters of the input sentence directly.

3.2 Bidirectional and concatenated row convolution for context

Given an input sequence of characters, the model projects each character into a continuous d-dimensional character vectors x_i using a character lookup table, which is similar to the word lookup table in the word-based NMT. Then, it builds a bidirectional and concatenated row convolution layer to compute the context vector c_i for x_i. Different from the traditional row convolution proposed in (Amodei et al., 2015) which only looks forward for the future context, the proposed bidirectional row convolution also looks behind to the history context. The intuition behind this layer is that, in additional to the future context, a small portion of history context is also needed to compose a full word-level representation for the character in current context. Suppose the input character x_i at time-step i, and the window size of the bidirectional row convolution is set as τ, we get a context matrix $x_{i-\tau,i+\tau}$ of size $d \times (2\tau + 1)$. We define a convolution matrix W of the same size as $x_{i-\tau,i+\tau}$. The activation c_i for the layer at time-step i is computed as:

$$c_i = [v_{i-\tau}; v_{i-\tau+1}; \ldots; v_{i+\tau}] \tag{7}$$

where $v_{i-\tau+t}$ is computed as:

$$v_{i-\tau+t} = w_t \times x_{i-\tau+t} (1 \leq t \leq (2\tau + 1)) \tag{8}$$

Since c_i is concatenated from $v^{i-\tau:i+\tau}$, the row convolution proposed in this paper is referred to as concatenated row convolution. For comparison and clarity, we call the row convolution in (Amodei et al., 2015) as summed row convolution.

3.3 Character composing

To fully utilize the character embedding x_i and its context vector c_i, we propose two different structures to compose the word-level representation u_i.

Forward character composing The forward character composing simply uses a linear transformation to combine the character embedding and its context embedding. For each character x_i, the corresponding word-level representation u_i is computed as:

$$u_i = M_1 x_i + M_2 c_i + b \tag{9}$$

where the weight matrix $M_1 \in R^{d \times d}$, the transformation matrix $M_2 \in R^{2\tau+1}$ convert the context c_i into the same dimension with x_i, the bias vector $b \in R^d$. Hence, the word-level representation u_i is kept the same dimension. In this model, the forward character composing layer is expected to learn the word-formation from the character and its neighboring characters automatically.

Recurrent character composing The recurrent character composing considers the interference from the former word-level representation u_{i-1} when computes u_i. Since the character x_{i-1} shares part of the neighboring characters with x_i, the context c_{i-1} and c_i hold some information in common. To reflect this interaction, or articulation, the u_i is calculated as:

$$u_i = M_1 x_i + M_2 c_i + M_3 u_{i-1} + b \tag{10}$$

Where the matrix $M_3 \in R^{d \times d}$ reflects the interference from u_{i-1}. The intuition behind this recurrent connection is that if the shared neighboring characters have provided much information for u_{i-1}, they should show less effects on u_i.

4 Experiments and results

We evaluate the proposed character-aware NMT model on the Chinese to English translation task. The open-source NMT system, GroundHog [*] (Bahdanau et al., 2014) is used as the baseline system.

4.1 Dataset

For the Chinese to English translation task, the training data consists of 2.3M pairs of sentences. As the traditional RNN search model relies on vector representations for words, we build a fixed vocabulary for each language respectively by choosing 45k of the most frequent words for the source language and 41k of the most frequent words for the target language. Words not included in the vocabulary are replaced with "UNK". For the proposed character-aware NMT model, we build a fixed character vocabulary with the size of 7009 for Chinese, which covers all of the Chinese characters in the training data. The vocabulary for English is the same with the traditional RNN search model. We use the BLEU metric to evaluate the translation quality and test the translation performance on IWSLT04, IWSLT05, IWSLT07, IWSLT08, MT08 and MT12.

4.2 Training detail

In the character-aware NMT model, the character embedding of the source side and the word embedding in the target side are all regarded as part of the model's parameters, and initialized by Gaussian distribution or uniform distribution, same as other parameters of the model. The window size of the row convolution τ is a hyper-parameter which can be set by the user ahead of time. In our implementation, we test the influence of τ by setting it as two, three and four respectively. We use parallel corpus to train RNN search model on a cluster with 8 Tesla K40 GPUs and it takes about 3 days to train the model for a total of 6 epochs. We use the same corpus to train the character-aware NMT model on the same cluster and the training time is longer than RNN search model: 4 days are needed to train the character-aware model for 6 epochs.

4.3 Impacts of the window size

The window size τ is a hyper-parameter which can be set by the user beforehand and it shows great impacts on the translation performance of our proposed model. Table 1 shows the translation performance of the character-aware NMT model when the window size is set as two, three and four respectively.

[*] https://github.com/lisa-groundhog/GroundHog

From table 1, it's easily to be found that the character-aware NMT model achieves the best performance when the window size is set as two. When the window size comes to four, the model can't be trained to converge. We explain this as that when the window size set as four, the neighboring word-level representations share too much information so that there is no distinction between the inputs to the bidirectional RNN. Hence the model may get confused and uneasily to be trained to converge.

Window size	IWSLT04	IWSLT05	IWSLT07	IWSLT08
2	50.01	52.13	33.10	43.15
3	48.62	51.30	31.15	41.26
4	—	—	—	—

Table 1: The impacts of the window size on the character-aware NMT model which has the recurrent connections.

4.4 Results on Chinese-English translation

Table 2 shows the BLEU score on Chinese-English test sets. The word embedding in traditional RNN search model and the character embedding in the proposed character-aware NMT model are all initialized to 512 dimensions by Gaussian distribution. The window size τ is set as two. In table 2, the RNNsearch-Word is the traditional RNN search model which serves as a baseline. To show the ability of our proposed character-aware NMT model, we also test the performance of the model RNNsearch-Char. The only difference between the RNNsearch-Char and the RNNsearch-Word is that the former regards the input sentence as a sequence of characters and the latter regards it as a sequence of words. The Character-aware-forward is the proposed character-aware NMT model which composes the word-level representation with the forward character composing layer and the Character-aware-recurrent utilize the recurrent character composing. By comparing the RNNsearch-word and RNNsearch-Char, we can find that the traditional RNN search model is incapable of handling the case where the input is a sequence of characters. Compared to the RNNsearch-Char, the proposed Character-aware-forward model leads to improvement up to 1.4 BLEU points although its performance is still worse than the baseline of RNNsearch-Word. The Character-aware-recurrent model leads to more significant improvement than the RNNsearch-Char and achieves comparable results with RNNsearch-Word.

Model	IWSLT04	IWSLT05	IWSLT07	IWSLT08	MT08	MT12
RNNsearch-Word	50.14	51.99	33.12	43.02	20.66	20.20
RNNsearch-Char	45.18	49.33	31.29	40.72	17.64	18.39
Character-aware-forward	47.56	50.74	32.48	42.17	19.25	19.65
Character-aware-recurrent	50.01	52.13	33.10	43.15	20.58	20.31

Table 2: The results on the Chinese to English translation tasks.

4.5 Comparison between the summed and concatenated row convolution

To show the effectiveness of our proposed concatenated row convolution, we compare the translation performance between the bidirectional summed row convolution and the bidirectional concatenated row convolution. Both of the two models have recurrent connections in the row convolution layer and the window size are both set as two. From table 3, we can find that the concatenated row convolution outperforms the summed row convolution at every test set. We conjecture that the summed row convolution have lost the information of the context vectors' relative position, which may be vital for composing word-level representation from characters.

Model	IWSLT04	IWSLT05	IWSLT07	IWSLT08
Concatenated row convolution	50.01	52.13	33.10	43.15
Summed row convolution	49.14	50.56	32.48	41.83

Table 3: The comparison between the summed and concatenated row convolution.

RNNsearch_Word	Character-aware-recurrent
Source: 我 的 名字 叫 ***铃木直子*** 。 Translation: My name is <unk>.	Source: 我 的 名 子 叫 ***铃 木 直 子*** 。 Translation: My name is Naoko Suzuki.
Source: 你 知道 ***清水寺*** 在 哪儿 吗？ Translation: Do you know where the <unk> is ?	Source: 你 知 道 ***清 水 寺*** 在 哪 儿 吗？ Translation: Do you know where the water-temple is ?
Source: 你好， ***艾米 哈里斯*** 夫人。 Translation: Hello, MS Amy.	Source: 你 好， ***艾 米 哈 里 斯*** 夫 人 。 Translation: Hello, MS Amy Harris.

Table 4: The translation performance on name entity.

4.6 Performance on name entity and unknown words

To our surprise, the character-aware NMT model is able to translate the name entity very well, as shown in table 4. According to table 4, we can see that the proposed model achieves better translation performance on name entity than the traditional word-based RNN search model. This is partly because that the name entity usually occurs rarely in the training corpus and is often mapped to an unknown word by the RNN search model. However, in character-aware NMT model, the name entity is split into a sequence of characters and each character can be found in the vocabulary. Despite that we still use a word-based decoder in the proposed character-aware NMT model, the number of unknown words in output sentences has decreased dramatically.

5 Conclusions and future work

In this work, we present a novel and simple character-aware NMT model which encodes the input sentence at the character-level. In addition to be applied to the language that has a clear boundary between words, the proposed model is also applied to the language without explicit word segmentation. Hence, no relying on the word boundaries is the most obvious advantage of our model. We firstly introduce the row convolution into NMT and test the effectiveness of several different row convolution methods. Experimental results show that the proposed character-aware NMT model can achieve comparable results with the traditional word-based RNNsearch model. Despite the target side of the proposed model is still word based, the number of unknown words in the output sentences get decreased dramatically. Moreover, the character-aware NMT model shows its superiority on the translation of name entities.

One limitation of our model is that the decoder is still word based. However, this has allowed us a more fine-grained analysis. But in the future, a setting where the target side is also represented as a character sequence must be investigated.

Acknowledgements

This work is supported by National Program on Key Basic Research Project of China(973 Program)(Grant No.2013CB329302). We would like to thank Xu Shuang for her preparing data used in this work. Additionally, we also want to thank Chen Zhineng, Li Jie, Geng Wang, Wang Wenfu and Zhao Yuanyuan for their invaluable discussions on this work.

References

Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, et al. 2015. Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv preprint arXiv:1512.02595*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Sébastien Jean Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. *arXiv preprint arXiv:1601.01085*.

David Golub and Xiaodong He. 2016. Character-level question answering with attention. *arXiv preprint arXiv:1604.00727*.

Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. 2016. Improved neural machine translation with smt features. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. *EMNLP*, pages 1700–1709.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015a. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015b. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *arXiv preprint arXiv:1604.00788*.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pages 2265–2273.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433*.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, pages 3104–3112.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Coverage-based neural machine translation. *arXiv preprint arXiv:1601.04811*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

Convolution-Enhanced Bilingual Recursive Neural Network for Bilingual Semantic Modeling

Jinsong Su[1], **Biao Zhang**[1], **Deyi Xiong**[2]*, **Ruochen Li**[1], **Jianmin Yin**[3]
Xiamen University, Xiamen, China 361005[1]
Soochow University, Suzhou, China 215006[2]
Weifang Beida Jade Bird Huaguang Information Technology Co., Ltd, Weifang, China 261205[3]
jssu@xmu.edu.cn, zb@stu.xmu.edu.cn
dyxiong@suda.edu.cn, lrc_n@stu.xmu.edu.cn, jimyin@vip.sina.com

Abstract

Estimating similarities at different levels of linguistic units, such as words, sub-phrases and phrases, is helpful for measuring semantic similarity of an entire bilingual phrase. In this paper, we propose a convolution-enhanced bilingual recursive neural network (ConvBRNN), which not only exploits word alignments to guide the generation of phrase structures but also integrates multiple-level information of the generated phrase structures into bilingual semantic modeling. In order to accurately learn the semantic hierarchy of a bilingual phrase, we develop a recursive neural network to constrain the learned bilingual phrase structures to be consistent with word alignments. Upon the generated source and target phrase structures, we stack a convolutional neural network to integrate vector representations of linguistic units on the structures into bilingual phrase embeddings. After that, we fully incorporate information of different linguistic units into a bilinear semantic similarity model. We introduce two max-margin losses to train the ConvBRNN model: one for the phrase structure inference and the other for the semantic similarity model. Experiments on NIST Chinese-English translation tasks demonstrate the high quality of the generated bilingual phrase structures with respect to word alignments and the effectiveness of learned semantic similarities on machine translation.

1 Introduction

Recently, adapting deep neural networks to statistical machine translation (SMT) is of growing interest due to their superior capacity against conventional lexical models in feature learning and representation (Yang et al., 2013; Liu et al., 2013; Li et al., 2013; Devlin et al., 2014; Liu et al., 2014; Setiawan et al., 2015). As phrases are the basic translation units in many SMT systems, one line of research among these studies is to learn the semantic similarity of bilingual phrases for translation selection in SMT (Zhang et al., 2014a; Gao et al., 2014; Cho et al., 2014; Su et al., 2015; Hu et al., 2015).

Typically, these bilingual semantic similarity models learn source and target phrase representations with some bilingual constraints (Gao et al., 2014; Hu et al., 2015; Zhang et al., 2014a). In spite of their success, they often suffer from two problems. Firstly, it is difficult for them to recover the semantic hierarchy (binary tree structure) of a bilingual phrase. In this respect, Su et al. (2015) improve tree construction by incorporating word alignments into their objective function. Unfortunately, they still employ the recursive autoencoder (RAE) as the underlying model to build tree structures of phrases according to the minimum reconstruction error. As a result, word alignments are not fully exploited for phrase structure generation. Secondly, the previous bilingual semantic similarity models are incapable of leveraging representations at different levels of linguistic units, such as words, sub-phrases and phrases. They usually represent a phrase (a sequence of words) with a single, fixed vector. However, as demonstrated in attention-based neural machine translation (Bahdanau et al., 2014), one vector is not semantically sufficient to encode a sequence of words preserving representations at different levels of linguistic units may be beneficial.

*Corresponding author.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3071–3081, Osaka, Japan, December 11-17 2016.

To solve these problems, we propose a convolution enhanced bilingual recursive neural network (ConvBRNN), which exploits word alignments to guide the generation of phrase structures and then integrates embeddings of different linguistic units on the phrase structures into bilingual semantic modeling. Specifically, we develop a new recursive neural network, in which the composition criterion for tree construction is the degree of consistency to word alignments rather than the reconstruction error. Furthermore, we propose a variant of the tree-based convolutional neural network (Mou et al., 2015) to fully access all embeddings on the phrase structures, which can be used to produce better phrase representations (see Section 3.2). All these make ConvBRNN more suitable for the subsequent bilingual semantic modeling, where a bilinear model is introduced to interact and compare the source and target phrase representations in terms of the degree of semantic equivalence. To train our model, we introduce two max-margin losses: one for the bilingual semantic structure inference and the other for the semantic similarity model, both of which are derivable.

We conduct experiments on large-scale corpus to examine the effectiveness of ConvBRNN on bilingual phrase structure learning and semantic similarity estimation. Experiment results on NIST MT06 and MT08 datasets show that our system achieves significant improvements over baseline methods. We further analyze the generated bilingual phrase structures and semantic scores, both of which indicate that ConvBRNN indeed learns information from word alignments that is beneficial for bilingual semantic representations.

Our major contributions lie in the following three aspects:

- We develop a new recursive neural network with an alignment-based semantic composition metric to generate word-alignment-consistent bilingual phrase structures.
- we develop a variant of tree-based convolutional neural model, which utilizes all embeddings on a phrase structure rather than the embedding of the entire phrase to model bilingual semantics.
- We carry out a series of experiments and demonstrate that our model is superior to baselines in terms of both the learned phrase structures and semantic similarities.

2 Related Work

A straightforward approach to learning bilingual phrase representations is to adapt monolingual phrase models with bilingual supervisions. For example, Li et al. (2013) encode reordering orientations into RAE-generated embeddings. To utilize the semantic equivalence constraint between source and target phrases, Gao et al. (2014) use a feedforward neural network to model phrase embeddings and try to maximize their semantic similarity, while Zhang et al. (2014a) introduce a bilingually-constrained RAE. Furthermore, Hu et al. (2015) incorporate context information to disambiguate translation selection. Very recently, neural machine translation trains a unified encoder-decoder (Sutskever et al., 2014; Bahdanau et al., 2014) neural network for translation, where an encoder maps the input sentence into a fixed-length vector, and a decoder generates a translation from the encoded vector.

Unlike the work mentioned above, our model mainly explore word alignments to guide the generation of bilingual phrase structures. The most relevant work to ours is the model proposed by Su et al. (2015), where they treat word alignments as a constraint to the RAE model. However, as discussed in Section 1, the composition criterion in RAE (i.e. reconstruction errors) does not allow us to fully benefit from word alignments. Therefore, we introduce a new composition criterion based on word alignment consistency. The proposed recursive neural network works in a way similar to that in (Socher et al., 2011b) except for our specific bilingual supervision. Zhang et al. (2014b) also propose a recursive neural network. However, their model mainly focuses on the composition in machine translation process (namely, *swap* or *monotone*), which is different from ours.

Additionally, our model also adapts convolutional neural network (Kalchbrenner et al., 2014; Kim, 2014) to extract semantic information encoded in phrase structures. Our model is related to the tree-based convolution (Mou et al., 2015). The differences are 1) that we treat the whole tree structure as the window for convolution; and 2) that the underlying phrase structure for a sentence is generated automatically in our model, instead of taking from a given constituency or dependency tree. Besides, the exploration of the semantic embeddings at different levels of granularity is firstly investigated in

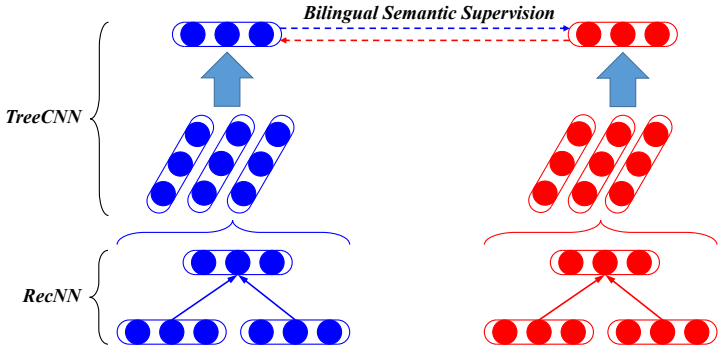


Figure 1: An illustration of the convolution-enhanced bilingual recursive neural network.

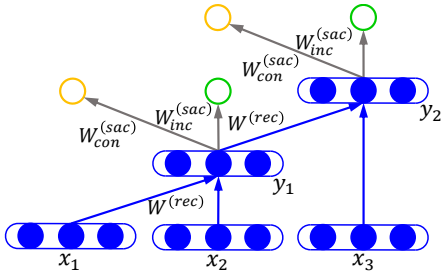


Figure 2: An illustration of the proposed RecNN. We use a yellow/green circle to represent the preference score of a node to be an SAC/non-SAC node.

(Socher et al., 2011a), where they compute an interaction matrix from which discriminative features are dynamically extracted for paraphrase identification. He et al. (2015) and Yin et al. (2015b) further extend this idea to convolutional neural network. Although our method is partially inspired by them, we implement it in a completely different manner.

3 Convolution-Enhanced Bilingual Recursive Neural Network

This section elaborates the proposed ConvBRNN model, of which network structure is shown in Figure 1. We begin with the generation of phrase structures via a recursive neural network. We then elaborate how to perform convolution upon the generated phrase structures. After that, we describe our bilingual semantic similarity model. Finally, we provide a detailed illustration on the training of ConvBRNN.

3.1 Recursive Neural Network for Generating Phrase Structures

To generate phrase structures, the conventional RAE usually composes neighboring nodes based on their reconstruction errors, which we argue are insufficient to model bilingual semantics. In SMT, one important auxiliary for a bilingual phrase is its word alignments, which contain some useful guidance signals for the bilingual structure construction, as discussed in Section 1. To make better use of these signals, we introduce the following recursive neural network (*RecNN*).

As shown in Figure 2, the input to our RecNN is a list of ordered d-dimensional vectors x=(x_1, x_2, x_3), each of which can be retrieved from a word embedding matrix $\mathbb{L} \in \mathbb{R}^{d \times |V|}$ via its corresponding word index. Here $|V|$ is the size of the vocabulary. Given two neighboring children c_1 and c_2, we compose them into a parent node n (For example, in Figure 2, if we set c_1=x_1 and c_2=x_2, then n=y_1) and produce its semantic vector p_n through a non-linear transformation:

$$p_n = f(W^{(rec)}[c_1; c_2] + b^{(rec)}) \tag{1}$$

where $[c_1; c_2] \in \mathbb{R}^{2d}$ is the concatenation of c_1 and c_2, $W^{(rec)} \in \mathbb{R}^{d \times 2d}$ and $b^{(rec)} \in \mathbb{R}^d$ is the parameter matrix and bias term respectively, and $f(\cdot)$ is an element-wise activation function such as $\tanh(\cdot)$, which is used throughout our experiments. As discussed in Section 1, the previous RAE-style models (Zhang

et al., 2014a; Su et al., 2015) adopt reconstruction error to measure how well p_n represents its children c_1 and c_2, which, however, is not a good solution to directly fully exploit word alignments for bilingual semantic modeling. To fully exploit different levels of bilingual semantic constraints within phrase pairs, we design a new semantic composition metric based on word alignments. As word alignments are shared across the source and target language, they are suitable to act as a desirable bridge for modeling bilingual semantics.

To achieve this goal, we first use the *structural alignment consistency* (SAC) (Su et al., 2015) that is the basis of our model to classify resultant nodes of semantic compositions into two categories. Specifically, if the node n covers a sub-phrase, and there exists a sub-phrase in the other language such that these two sub-phrases are consistent with word alignments (Och and Ney, 2003), we say n satisfies the structural alignment consistency, and it is referred to as an SAC node, otherwise, it is a non-SAC node.

Then, we introduce two functions $Score_{con}(n)$ and $Score_{inc}(n)$ to measure the preference strength of node n to be an SAC or a non-SAC node, respectively

$$Score_{con}(n) = W_{con}^{(sac)}\, p_n, \quad Score_{inc}(n) = W_{inc}^{(sac)}\, p_n \tag{2}$$

where $W_{con}^{(sac)} \in \mathbb{R}^{1\times d}$ and $W_{inc}^{(sac)} \in \mathbb{R}^{1\times d}$ are parameter matrices. Furthermore, we calculate the final semantic composition score of node n as follows

$$Score_{sc}(n) = \frac{exp(Score_{con}(n))}{exp(Score_{inc}(n))} \tag{3}$$

Obviously, the larger $Score_{con}(n)$ is than $Score_{inc}(n)$, the larger $Score_{sc}(n)$ should be.

We traverse each possible semantic composition of neighboring children and calculate its semantic composition score, and finally select the composition with the largest score. This combination process on neighboring children repeats at each node until the structure and embedding of the entire bilingual phrase are generated. To obtain the optimal binary tree and phrase representation for x, we minimize the following objective function formulated as follows:

$$\begin{aligned} E_{align}(x) = & \sum_{n\in T_{con}(x)} max\{0, 1 - Score_{con}(n) + Score_{inc}(n)\} \\ & + \sum_{n\in T_{inc}(x)} max\{0, 1 - Score_{inc}(n) + Score_{con}(n)\} \end{aligned} \tag{4}$$

where $T_{con}(x)$ and $T_{inc}(x)$ denote the SAC and non-SAC node sets in the binary tree of x, respectively. It should be noted especially that we use different max-margin loss functions for different types of nodes. On the one hand, we simultaneously maximize the $Score_{con}(*)$ and minimize the $Score_{inc}(*)$ of SAC nodes. On the other hand, we take an opposite approach to deal with non-SAC nodes. In this way, the node type (SAC/non-SAC) with word alignment information performs as a guidance signal to encourage the generation of word-alignment-consistent phrase structures.

3.2 Convolutional Neural Network for Learning Phrase Representations

Given a generated phrase structure, a straightforward way to obtain phrase representation is to extract the embedding of the root node of the phrase structure, as implemented in the conventional RAE. However, a major limitation of this method is the neglect of lower-level linguistic units, e.g. words and sub-phrases. To alleviate this problem, we stack a variant of tree-based convolutional neural network (*TreeCNN*) to incorporate all the embeddings inside the phrase structure.

Upon the generated structure $T(x)$ of an input phrase x, we first perform postorder traversal to extract embeddings of all nodes, and then concatenate them column-wisely into a matrix $M \in \mathbb{R}^{d\times|n|}$, where $|n|$ is the number of nodes in $T(x)$. Note that the node number varies with different phrases. In this way, the representations at different levels are interlaced along the rows of M, which facilitates the upcoming window-based convolution. To construct our TreeCNN, we take the matrix M as the input layer. Figure

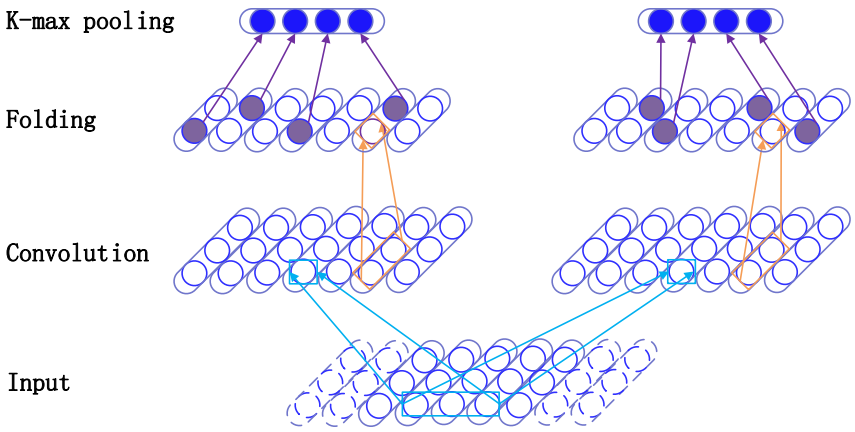


Figure 3: Illustration of the proposed TreeCNN with window size 3, pooling size 2 and filter number 2. We use a light blue, orange and purple color to indicate the convolution, folding and pooling operation, respectively. The nodes with dashed circles represent zero-padded embeddings for wide convolution.

3 shows the architecture of our TreeCNN, which consists of three different layers: *convolution*, *folding* and *k-max pooling*.

Convolution Layer This layer iteratively convolves an h-sized sliding window on M, and uses a filter F to summarize the information inside the window. Since the length of phrases in the translation model is usually not long, we pad the matrix M with h-1 zero embeddings on both sides and adopt the wide convolution (Kalchbrenner et al., 2014) (see the dashed circles in Figure 3). To discover semantic information at a finer granularity, we further construct *per-dimension filters* $F^{[r]}$ ($1 \leq r \leq d$) (He et al., 2015) to convolve the embeddings in the r-th row of M.

Formally, applying the per-dimension filter $F^{[r]}$ on M produces an output vector $C^{[r]} \in \mathbb{R}^{|n|+h-1}$ where the i-th entry ($1 \leq i \leq |n| + h - 1$) is computed as follows:

$$C_i^{[r]} = (W_{F^{[r]}})^T M_{i:i+h-1}^{[r]} \tag{5}$$

where $W_{F^{[r]}} \in \mathbb{R}^h$ is the parameter vector of $F^{[r]}$. This procedure is illustrated in Figure 3 with a light blue color. By applying all per-dimension filters to traverse all windows of matrix M, we can obtain a feature map $C \in \mathbb{R}^{d \times (|n|+h-1)}$. It encodes complex dependencies across different levels of linguistic units and contains linguistic properties implied in each dimension, which, nevertheless, makes different dimensions independent of each other. Next we will introduce a folding layer to exploit these dimensions simultaneously.

Folding Layer This layer bridges the gap across different dimensions through averaging each nonoverlapping neighboring rows in the convoluted feature map C. Specifically, for each row index r ($1 \leq r \leq \lfloor \frac{d}{2} \rfloor$), the output can be computed as follows (shown in the orange color in Figure 3):

$$A^{[r]} = (C^{[2r-1]} + C^{[2r]})/2 \tag{6}$$

where $A^{[r]} \in \mathbb{R}^{|n|+h-1}$ is the r-th row of $A \in \mathbb{R}^{\lceil \frac{d}{2} \rceil \times (|n|+h-1)}$. Different from previous work, we allow dimension size d to be odd. In this case, we simply append the last row of C onto A.

After the above operation, each element of A captures complex dependencies across both rows and columns of M. To mingle these dependencies, we further perform a non-linear transformation following Yin et al. (2015a):

$$U = f(A + b_{[:]}) \tag{7}$$

where $b \in \mathbb{R}^{\lceil \frac{d}{2} \rceil}$ is the bias term that is shared across different columns, and the subscript $[:]$ indicates a column-wise *broadcasting* operation. It should be noted that the column dimension of U (i.e. $|n|+h-1$) differs for different phrases. This raises a key problem: how can we transform the variable-length matrix

U into a fixed-length vector. In order to deal with this problem, we further stack a K-max Pooling layer (Kalchbrenner et al., 2014).

K-max Pooling Layer This layer extracts the top-k values over each row of U so as to: 1) preserve rich semantic information of a sentence; and 2) eliminate the variance in the column dimension of U (shown in the purple color in Figure 3). In doing so, we obtain a phrase vector representation with the dimension size $\lceil \frac{d}{2} \rceil \cdot k$. Notice that k in this layer is predefined. Although we can use the dynamic version of k-max pooling to stack more convolution, folding and pooling layers (Kalchbrenner et al., 2014), we do not take this strategy due to the trade-off between performance and cost. Theoretically, more layers should capture much deeper semantic information. We leave this for our future research.

So far we have described how we apply the wide convolution, folding layer and k-max pooling layer onto an input phrase matrix to obtain a fixed-length phrase representation. Inspired by studies on convolutional networks for object recognition, we introduce L filters to produce multiple feature maps, which are used to capture semantics of input phrases. Finally, we concatenate the vector representations derived from L filters to obtain the final phrase representation $p \in \mathbb{R}^{\lceil \frac{d}{2} \rceil \cdot k \cdot L}$.

3.3 Bilingual Semantic Supervision

Through the above procedures, we obtain the semantic representations of bilingual phrase (f, e), denoted by p_f and p_e. To measure the semantic similarity of f and e, we introduce two transformation matrixes $W_f^{(sem)} \in \mathbb{R}^{d_{sem} \times (\lceil \frac{d_s}{2} \rceil \cdot k \cdot L)}$ and $W_e^{(sem)} \in \mathbb{R}^{d_{sem} \times (\lceil \frac{d_t}{2} \rceil \cdot k \cdot L)}$ to project their semantic representations p_f and p_e into a common semantic space:

$$p'_f = f(W_f^{(sem)} p_f + b^{(sem)}), \;\; p'_e = f(W_e^{(sem)} p_e + b^{(sem)}) \tag{8}$$

where p'_f and p'_e are transformed representations of f and e, d_s/d_t is the dimension size of phrase representation in the source/target semantic space, d_{sem} is the that of the common semantic space. Although we distinguish the transformation matrices for the source and target language, we share the same bias term $b^{(sem)}$ for both languages. The advantage of this is that our model will learn to encode bilingual semantics into these transformation matrices, rather than biases.

Then, we further stack a bilinear model over the transformed representations to compute the semantic similarity score $Sim(f, e)$:

$$Sim(f, e) = {p'_f}^T W_{bi}^{(sem)} p'_e \tag{9}$$

where $W_{bi}^{(sem)} \in \mathbb{R}^{d_{sem} \times d_{sem}}$ is a squared matrix of parameters to be learned. Intuitively, each element in $W_{bi}^{(sem)}$ represents an interaction between p'_f and p'_e, which is used to capture the semantic correspondence within f and e.

To make the semantic scores of translation equivalents as large as possible while scores of non-translation pairs as small as possible, we introduce the following max-margin loss for (f, e):

$$\begin{aligned} E_{sem}(f, e) = & \; max\{0, 1 - Sim(f, e) + Sim(f, e^-)\} \\ & + max\{0, 1 - Sim(f, e) + Sim(f^-, e)\} \end{aligned} \tag{10}$$

where f^-/e^- is a bad translation that replaces the words in f/e with randomly chosen source/target language words.

3.4 Model Training

As described above, there are two types of errors involved for the phrase pair (f, e): (1) structural alignment error $E_{align}(f, e)$ that estimates how well the generated structures of f and e comply with word alignments, and (2) semantic error $E_{sem}(f, e)$ that measures how well the learned phrase embeddings of f and e are semantically equivalent.

Given a training corpus $D = \{(f, e)\}$, the final objective of ConvBRNN is formulated as follows:

$$J_{ConvBRNN}(\theta) = \frac{1}{|D|} \sum_{(f,e) \in D} \{\alpha E_{align}(f, e) + (1 - \alpha) E_{sem}(f, e)\} + R(\theta) \tag{11}$$

where $E_{align}(f,e)$ is the sum of $E_{align}(f)$ and $E_{align}(e)$, the hyper-parameter α is used to balance the effects of $E_{align}(f,e)$ and $E_{sem}(f,e)$, and $R(\theta)$ is a regularization term.

Parameters θ are divided into four sets[1]: (1) θ_L: the word embedding matrix (Section 3.1); (2) θ_{RT}: the structure parameters of RecNN (Section 3.1) and TreeCNN (Section 3.2); (3) θ_{wa}: the parameters for structural alignment consistency (Section 3.1); (4) θ_{sem}: the parameters for semantic similarity (Section 3.3). Following previous work (Zhang et al., 2014a; Su et al., 2015), we assign each parameter set a unique weight for regularization:

$$R(\theta) = \frac{\lambda_L}{2}\|\theta_L\|^2 + \frac{\lambda_{RT}}{2}\|\theta_{RT}\|^2 + \frac{\lambda_{wa}}{2}\|\theta_{wa}\|^2 + \frac{\lambda_{sem}}{2}\|\theta_{sem}\|^2 \quad (12)$$

We apply L-BFGS to tune parameters based on gradients over the joint error, as implemented in (Socher et al., 2011c). Word vector embeddings θ_L are initialized with the toolkit Word2Vec[2] on a large scale unlabeled data. Other parameters are randomly initialized according to a normal distribution (μ = 0,σ = 0.01). With the trained model parameters, we can easily obtain the dense semantic vectors for bilingual phrases. During translation, we incorporate the derived phrasal similarity feature into the standard log-linear framework (Och and Ney, 2002) of SMT for translation selection.

4 Experiment

We conducted experiments on NIST Chinese-English translation task to validate the effectiveness of ConvBRNN.

System Overview Our baseline decoder is a state-of-the-art phrase-based translation system equipped with a maximum entropy based reordering model, which adopts three bracketing transduction grammar rules (Wu, 1997; Xiong et al., 2006). We compared the proposed model with two models: (1) the bilingual correspondence model (*BCorrRAE*) proposed by Su et al. (2015); (2) the proposed model without the convolutional neural network (*ConvBRNN-CNN*), which simply treats the embedding of root node of the phrase structure as the semantic representation of the whole phrase, instead of the convoluted one. Other components of ConvBRNN-CNN are the same as those in the ConvBRNN model.

All translation systems used the log-linear framework. The adopted sub-models include: (1) rule translation probabilities in two directions, (2) lexical weights in two directions, (3) targets-side word number, (4) phrase number, (5) language model score, (6) the score of maximal entropy based reordering model, (7) the semantic similarities of phrase pairs. We performed minimum error rate training to tune the optimal feature weights on the development set (Och and Ney, 2003).

Experiment Setup Our training corpus contains 1.0M sentence pairs (25.2M Chinese words and 29M English words) that are from the FBIS corpus and Handsards part of LDC2004T07 corpus. We ran GIZA++[3] on the training data in two directions and applied the "*grow-diag-final-and*" heuristic rule to obtain word alignments. We trained a 5-gram language model on the Xinhua portion of the GIGAWORD corpus using *SRILM* Toolkit[4] with modified Kneser-Ney Smoothing. We chose the 2005 NIST MT evaluation test data as the development set, and the 2006, 2008 NIST MT evaluation test data as the test sets. We used case-insensitive BLEU-4 metric (Papineni et al., 2002) to evaluate translation quality, and conducted paired bootstrap sampling (Koehn, 2004) for significance test.

Network Training To train ConvBRNN, we applied forced decoding (Wuebker et al., 2010) on the training corpus to extract high-quality bilingual phrases for model training. We tuned the optimal hyper-parameters via *random search* method (Bergstra and Bengio, 2012) to minimize the joint error on a small portion of our training data. Finally, we set $d_s = d_t = d_{sem} = 50$, $h = 5$, $L = 10$, $k = 3$, $\alpha = 0.116$, $\lambda_L = 2.14e^{-7}$, $\lambda_{RT} = 2.43e^{-5}$, $\lambda_{wa} = 7.33e^{-5}$ and $\lambda_{sem} = 4.03e^{-6}$, the L-BFGS iteration number N_{iter}=100. To train BCorrRAE, we used the same training data and method for hyper-parameter optimization.

[1]Note that the source and target languages have different four sets of parameters.

[2]https://code.google.com/p/word2vec/

[3]http://www.statmt.org/moses/giza/GIZA++.html

[4]http://www.speech.sri.com/projects/srilm/download.html

Method	MT06	MT08	AVG
Baseline	29.66	21.52	25.59
BCorrRAE	30.94	23.33	27.14
ConvBRNN-CNN	31.16^{+}	23.39^{+}	27.28
ConvBRNN	$\mathbf{31.48}^{+*}$	$\mathbf{23.89}^{+*}$	**27.69**

Table 1: Experiment results on the MT 06/08 test sets, where we highlight the best result in bold. **AVG** = average BLEU scores on test sets; "+": significantly better than *Baseline* ($p < 0.01$); "$*$": significantly better than *BCorrRAE* ($p < 0.05$);

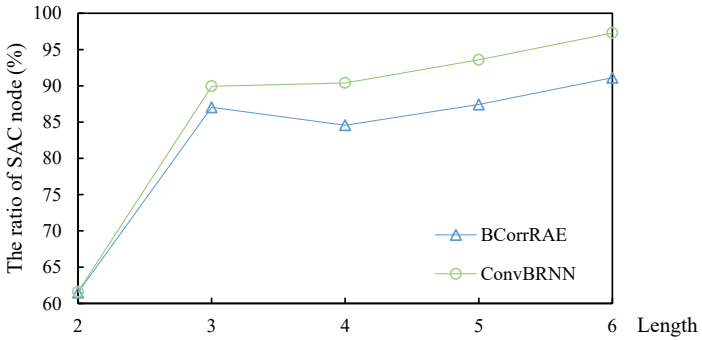


Figure 4: The ratio of *SAC* node specific to the length of covered source phrase. We limit the maximal length of source and target phrase to be 7, and the results when length is 1 and 7 are not shown because they are the same for both models.

4.1 Translation Results

The first experiment checks whether the learned bilingual semantic similarity is able to improve the translation quality. Table 1 summarizes the detailed results. We can observe that our ConvBRNN model significantly improves translation quality in terms of BLEU score on all test sets. Overall, ConvBRNN obtains a gain of up to *2.1* BLEU points on average over the Baseline. Particularly, on the MT08 data set, the improvements over the Baseline can be up to *2.37* BLEU points.

The integration of bilingual correspondence helps BCorrRAE gain *1.55* BLEU points on average over the Baseline. With the recursive neural network for phrase structure generation, ConvBRNN-CNN performs slightly better than BCorrRAE. By integrating the tree-based convolution network for phrase representation learning, our ConvBRNN achieves further improvements over BCorrRAE, which is significant at $p<0.05$. For this result, the reasons may be the following two points: 1) bilingual phrase structures generated by ConvBRNN are more close to the actual semantic structures of phrases; 2) the ConvBRNN model encodes different levels of linguistic units inside phrase structures into final phrase representations. These two points are not adequately considered in BCorrRAE.

4.2 Result Analyses

In order to know how the ConvBRNN model improves the performance of the SMT system, we study the bilingual phrases of our model from the following two respects:

First, we investigate the ability of our model in generating word-alignment-consistent bilingual phrase structures. For this, we extracted phrase pairs from our translation model filtered by NIST test sets and computed the percentage of SAC nodes (Section 3.1) specific to the length of covered source phrase. Following the previous work (Su et al., 2015), we define this percentage as the ratio of the number of SAC nodes to that of all nodes.

Figure 4 reports the ratio values. The ConvBRNN model consistently outperforms the BCorrRAE model. Additionally, as the length grows, the ratio gap between two models becomes larger, with a gain of up to absolute 6%. This indicates that word alignments are more efficiently exploited by our

Source Phrase	BCorrRAE	ConvBRNN
wǒ rènwéi zhè zhǒng	((((*i think*) *this*) *is*) *a*) (((*i think*) *that*) *was*) ((*i regard*) *this*)	((*i think*) ((*this type*) *of*)) ((*i think*) ((*this kind*) *of*)) ((*i find*) ((*that kind*) *of*))
biǎoshì qiángliè bùmǎn	(*strong* ((*opposition against*) *the*)) ((((*expressed strong*) *opposition*) *to*) *the*) (*voice* (*my* (*strong* (*opposition against*))))	((*strongly* (*dissatisfied with*)) *the*) (((*voice* (*my* (*strong opposition*))) *against*) *the*) ((*express* (*strong dissatisfaction*)) *at*)
jiānjué zh̄ichí zhèngfǔ	((*resolutely* (*support the*)) *government*) ((*firm* (*supporter* (*of our*))) *government*) ((*staunchly* (*support* (*the Chinese*))) *government*)	((*staunchly support*) ((*the Chinese*) *government*)) ((*resolutely support*) (*the government*)) ((*firm supporter*) (*of* (*our government*)))

Table 2: Semantically similar target phrases in the training set for example source phrases. The brackets indicate the learned binary tree structure.

ConvBRNN model to generate word-alignment-consistent bilingual phrase structures.

Second, we study whether ConvBRNN can extract meaningful information for semantic similarity from the learned phrase structures. We show some source phrases in Table 2 with their most semantically similar translations learned by BCorrRAE and ConvBRNN in the training corpus. We find that both models are able to distinguish semantic equivalents from non-translation pairs. However, in contrast to BCorrRAE, ConvBRNN prefers diverse expressions. For example, "zhǒng" can be translated into "*type*" or "*kind*", and "bùmǎn" also has two candidate translations "*opposition*" and "*dissatisfaction*". Therefore, during translation, the decoder has many candidate translations for the same source phrase, which we argue is one of the reasons for our success.

We also provide phrase structures in Table 2. We observe that the semantic compositions in BCorrRAE are relatively meaningless because they often do not respect the linguistic phenomena. For example, BCorrRAE prefers branching structures in the same composition direction, such as "(*voice* (*my* (*strong* (*opposition against*))))". Besides, BCorrRAE is more likely to produce undesirable nodes covering high-frequency sub-phrases. For instance, the target phrase "*firm supporter of our government*" has different structures learned by BCorrRAE and ConvBRNN: "((*firm* (*supporter* (*of our*))) *governmen-t*))" (BCorrRAE) and "((*firm supporter*) (*of* (*our government*)))" (ConvBRNN). Obviously, the phrase structure learned by ConvBRNN is more syntactically meaningful. This again demonstrates the advantage of ConvBRAE over BCorrRAE in exploiting word alignments for learning better bilingual phrase structures.

5 Conclusion and Future Work

In this paper, we have presented a convolution-enhanced bilingual recursive neural network to learn bilingual semantic similarity. We first introduce a recursive neural network which directly exploits word alignments to generate word-alignment-consistent bilingual phrase structures. Based on these structures, we further employ a variant of tree-based convolutional neural network to produce bilingual phrase embeddings by summarizing embeddings at different levels of lingual units. Experiment results and analyses on machine translation demonstrate the effectiveness of our model.

In the future, we would like to explore more different selection functions in Eq. (3) for our model due to its importance for the generation of bilingual phrase structures. Besides, as discussed in Section 3.2, we will further enhance the proposed model by trying more effective components, such as dynamic version of k-max pooling, multi-layer convolutions.

Acknowledgements

The authors were supported by National Natural Science Foundation of China (Grant Nos. 61303082, 61403269 and 61672440), Natural Science Foundation of Fujian Province (Grant No. 2016J05161), Natural Science Foundation of Jiangsu Province (Grant No. BK20140355) and CCF Opening Project of Chinese Information Processing (Grant No. CCF2015-01-01). We also thank the anonymous reviewers for their insightful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, pages 281–305.

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

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proc. of ACL*, pages 1370–1380.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *Proc. of ACL*, pages 699–709.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proc. of EMNLP*, pages 1576–1586.

Baotian Hu, Zhaopeng Tu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2015. Context-dependent translation selection using convolutional neural network. In *Proc. of ACL*, pages 536–541.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proc. of ACL*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. of EMNLP*, pages 1746–1751.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP*, pages 388–395.

Peng Li, Yang Liu, and Maosong Sun. 2013. Recursive autoencoders for ITG-based translation. In *Proc. of EMNLP*, pages 567–577.

Lemao Liu, Taro Watanabe, Eiichiro Sumita, and Tiejun Zhao. 2013. Additive neural networks for statistical machine translation. In *Proc. of ACL*, pages 791–801.

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. 2014. A recursive recurrent neural network for statistical machine translation. In *Proc. of ACL*, pages 1491–1500.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proc. of EMNLP*, pages 2315–2325.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of ACL*, pages 295–302.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, pages 19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*, pages 311–318.

Hendra Setiawan, Zhongqiang Huang, Jacob Devlin, Thomas Lamar, Rabih Zbib, Richard Schwartz, and John Makhoul. 2015. Statistical machine translation features with multitask tensor networks. In *Proc. of ACL-IJCNLP*, pages 31–41.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y. Ng. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proc. of NIPS*, pages 801–809.

Richard Socher, Cliff C. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011b. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proc. of ICML*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011c. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proc. of EMNLP*, pages 151–161.

Jinsong Su, Deyi Xiong, Biao Zhang, Yang Liu, Junfeng Yao, and Min Zhang. 2015. Bilingual correspondence recursive autoencoder for statistical machine translation. In *Proc. of EMNLP*, pages 1248–1258.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics, Volume 23, Number 3, September 1997*.

Joern Wuebker, Arne Mauser, and Hermann Ney. 2010. Training phrase translation models with leaving-one-out. In *Proc. of ACL*, pages 475–484.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proc. of ACL*, pages 521–528.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word alignment modeling with context dependent deep neural network. In *Proc. of ACL*, pages 166–175.

Wenpeng Yin and Hinrich Schütze. 2015a. Convolutional neural network for paraphrase identification. In *Proc. of NAACL-HLT*, pages 901–911.

Wenpeng Yin and Hinrich Schütze. 2015b. Multigrancnn: An architecture for general matching of text chunks on multiple levels of granularity. In *Proc. of ACL-IJCNLP*, pages 63–73.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014a. Bilingually-constrained phrase embeddings for machine translation. In *Proc. of ACL*, pages 111–121.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014b. Mind the gap: Machine translation by minimizing the semantic gap in embedding space. In *Proc. of AAAI*, pages 1657–1664.

Improving Attention Modeling with Implicit Distortion and Fertility for Machine Translation

Shi Feng [1] † **Shujie Liu** [2] **Nan Yang** [2] **Mu Li** [2] **Ming Zhou** [2] **Kenny Q. Zhu** [3]
[1] University of Maryland, College Park
[2] Microsoft Research, Beijing, China
[3] Shanghai Jiao Tong University, Shanghai, China
shifeng@cs.umd.edu
shujliu,nanya,muli,mingzhou@microsoft.com
kzhu@cs.sjtu.edu.cn

Abstract

In neural machine translation, the attention mechanism facilitates the translation process by producing a soft alignment between the source sentence and the target sentence. However, without dedicated distortion and fertility models seen in traditional SMT systems, the learned alignment may not be accurate, which can lead to low translation quality. In this paper, we propose two novel models to improve attention-based neural machine translation. We propose a recurrent attention mechanism as an implicit distortion model, and a fertility conditioned decoder as an implicit fertility model. We conduct experiments on large-scale Chinese–English translation tasks. The results show that our models significantly improve both the alignment and translation quality compared to the original attention mechanism and several other variations.

1 Introduction

Sequence-to-sequence neural machine translation (NMT) has shown promising results lately (Sutskever et al., 2014; Cho et al., 2014b). An NMT model typically consists of an encoding neural network which transforms the source sentence into some vector representation, and a decoding neural network which generates the target sentence from the vector representation. This is called the encoder-decoder model. In order to handle variable length inputs, recurrent neural networks (RNN) are usually used as the encoder and the decoder. The encoder RNN will read the words in the source sentence one by one and generate a sequence of corresponding hidden states; the decoder will then by conditioned on the encoder states to output each word in the target sentence. In (Cho et al., 2014b), only the last encoder state is used for target sentence generation, so the single hidden state vector must preserve all the necessary information in the source sentence for the decoding process , which is very difficult when the source sentence is long.

To leverage the whole sequence of encoder states and retrieve information from the source sentence in a more flexible way, the attention mechanism (Bahdanau et al., 2014) was introduced into the encoder-decoder model. In an attention-based encoder-decoder model, matching scores between the source and target words are calculated based on their corresponding encoder and decoder states. These scores are then normalized and used as weights for the source words given each target word. This can be seen as a soft alignment and the attention mechanism here plays similar role to that of a traditional alignment model.

In alignment models used in traditional machine translation models such as IBM Models (Brown et al., 1993), distortion and fertility are modeled explicitly. By comparison, in the attention mechanism, alignment is computed by matching the previous decoder hidden state with all the encoder hidden states, without modeling distortion and fertility. Since the translation of target words is guided by the attention

† Work done while author was an undergraduate student of Shanghai Jiao Tong University and intern at Microsoft Research.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3082–3092, Osaka, Japan, December 11-17 2016.

mechanism, the translation accuracy of an attention-based NMT model is largely dependent on the accuracy of the alignment, and a large portion of errors seen in the translation result can be associated with the lack of distortion and fertility models. Without a distortion model, the generated alignment sometimes contains incorrect word reordering and as a result the meaning of the sentence could be twisted. Due to the lack of a fertility model, the number of times that each word in the source sentence be aligned to is not restricted, and as a result we sometimes observe that part of the sentence is translated repeatedly, or part of the sentence is missing in the translation.

In the following sections, we first review the attention-based encoder-decoder model, and then give a detailed analysis of these problems using example alignment matrices generated by the standard model. In Section 4 we introduce the two proposed extensions to the attention-based encoder decoder. We first introduce a recurrent attention mechanism with extra recurrent paths as an implicit distortion model to solve the reordering problem. To address the lack of fertility model, we use a fertility vector which memorizes the words that have been translated and design a decoder that is conditioned on this vector. In Section 6 we will show the results of our experiments on large-scale Chinese–English translation tasks and demonstrate that our proposed methods can significantly improve the translation performance.

2 Attention-based Encoder-Decoder

Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Cho et al., 2014b) are often used as RNNs in attention-based encoder-decoder models. In this section, we will briefly introduce GRU, followed by a short review of how attention is modeled between encoder and decoder states as described in (Bahdanau et al., 2014).

2.1 Gated Recurrent Unit

At time i, a recurrent function RNN computes its hidden state $\boldsymbol{h}_i$ based on the input $\boldsymbol{x}_i$ and previous hidden state $\boldsymbol{h}_{i-1}$:

$$\boldsymbol{h}_i = \text{RNN}(\boldsymbol{h}_{i-1}, \boldsymbol{x}_i)$$

A GRU uses reset gate and update gate to help model long-term dependencies:

$$\begin{aligned}
\boldsymbol{r}_i &= \sigma(\boldsymbol{W}^r \boldsymbol{x}_i + \boldsymbol{U}^r \boldsymbol{h}_{i-1}) \\
\boldsymbol{z}_i &= \sigma(\boldsymbol{W}^z \boldsymbol{x}_i + \boldsymbol{U}^z \boldsymbol{h}_{i-1}) \\
\boldsymbol{h}'_i &= \tanh(\boldsymbol{U}(\boldsymbol{r}_i \odot \boldsymbol{h}_{i-1}) + \boldsymbol{W}\boldsymbol{x}_i) \\
\boldsymbol{h}_i &= (1 - \boldsymbol{z}_i) \odot \boldsymbol{h}'_i + \boldsymbol{z}_i \odot \boldsymbol{h}_{i-1}
\end{aligned}$$

where $\boldsymbol{x}_i$ is the input, and $\boldsymbol{h}_{i-1}$ is the previous hidden state. $\boldsymbol{r}_i$ and $\boldsymbol{z}_i$ are reset and update gates respectively. $\odot$ denotes element-wise product.

2.2 RNNSearch

RNNSEARCH referes to the attention-based encoder-decoder model proposed by (Bahdanau et al., 2014). It consists of two RNNs: an encoder RNN that maps the source sentence to a sequence of hidden states, and a decoder RNN that generates the target sentence based on the encoder states with attention mechanism.

Encoder The encoder used in RNNSEARCH is a bi-directional GRU. It consists of two independent RNNs, one reading the source sentence from left to right, another from right to left, generating two hidden states at each position. The two hidden states produced by forward and backward RNNs are concatenated to generate the sequence of encoder states $\boldsymbol{s}_1^J$, where J is the length of source sentence.

Decoder Unlike the decoder of (Cho et al., 2014b; Sutskever et al., 2014), which takes only the last encoder state as the context vector, the decoder with attention mechanism uses encoder states from all time-stamps as context. Decoder with attention mechanism is illustrated in Figure 5.

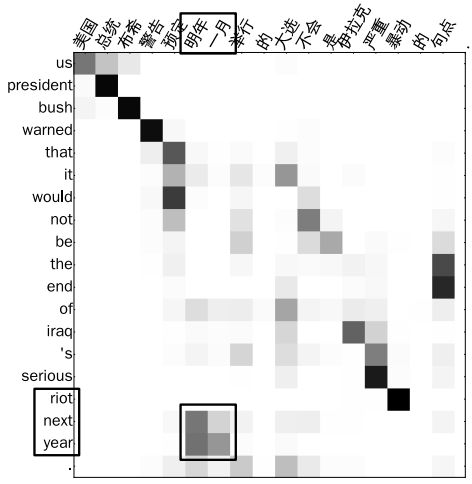


Figure 1. Incorrect reordering by the attention mechanism. The correct translation is "US president Bush warned that the election to be held on January 30th next year would not be an end to serious violence in Iraq."

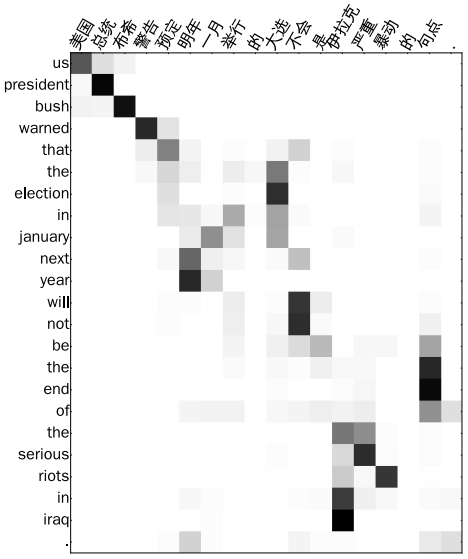


Figure 2. Our proposed model RECATT produced the correct reordering of the source words, and based on that generated a better translation.

At position i in the target sentence, the attention model computes a matching score e_{ij} with match function α, for the previous decoder state $\boldsymbol{h}_{i-1}$ and each encoder state $\boldsymbol{s}_j$.

$$e_{ij} = \boldsymbol{v}^\top \tanh\left(\alpha(\boldsymbol{h}_{i-1}, \boldsymbol{s}_j)\right)$$
$$w_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})}$$

We wrap this computation of weights as ALIGN:

$$\boldsymbol{w}_i = \text{ALIGN}(\boldsymbol{h}_{i-1}, \boldsymbol{s}_1^J)$$

There are various match functions, as analyzed in (Luong et al., 2015). In our paper we use the sum match function $\alpha(\boldsymbol{h}_{i-1}, \boldsymbol{s}_j) = \boldsymbol{W}^\alpha \boldsymbol{h}_{i-1} + \boldsymbol{U}^\alpha \boldsymbol{s}_j$. The weighted average of the encoder states $\boldsymbol{s}_1^J$ is calculated as the context $\boldsymbol{c}_i = \sum_j w_{ij} \boldsymbol{s}_j$. It is added to the input of each gate in the decoder, together with previous state $\boldsymbol{h}_{i-1}$ and previous target word embedding $\boldsymbol{y}_{i-1}$:

$$\boldsymbol{h}_i = \text{RNN}(\boldsymbol{h}_{i-1}, \boldsymbol{y}_{i-1}, \boldsymbol{c}_i)$$

3 Problems of the Attention Mechanism

Although attention modeling works well in finding translation correspondence between source and target words, there are still some issues that can be systematically identified, which fall into three categories: incorrect reordering, missing translation and repeated translation.

Incorrect Reordering Reordering is often required for the translation between two languages with different grammars. When the source words are translated in the wrong order, the meaning of the sentence can be twisted.In the example shown in Figure 1, the phrase "明年 一月" (meaning "January next year") in the source is attended to after the translation of "暴动" (meaning "riot"), resulting in a translation that twisted the meaning of the source sentence.

Missing Translation In Figure 3, we can see that only the first half of the source sentence is translated, because the last half sentence is never chosen for attention.

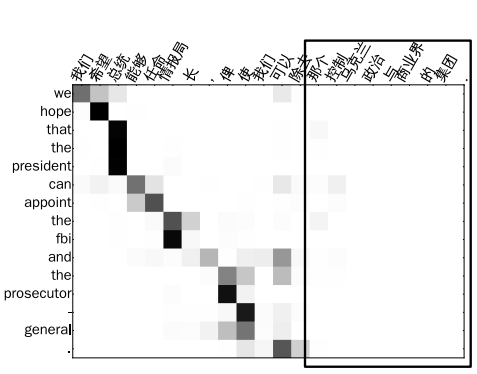


Figure 3. Missing translation example. The correct translation is "We hope that the president could appoint the chief of the intelligence bureau so we can eliminate groups that control Ukrainian's politics and business."

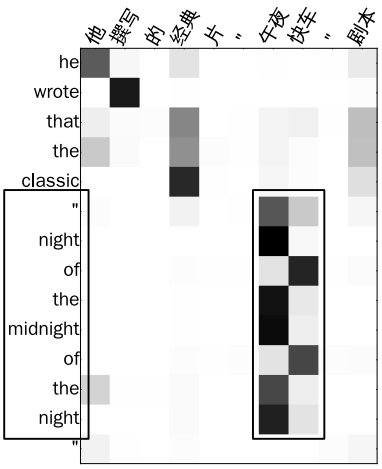


Figure 4. Repeated translation example. The correct translation is "Powell will attend the annual meeting of the organization of europe."

Repeated Translation In the example shown in Figure 4, part of the source sentence, "欧安 组织" ("the organization of europe"), is repeatedly translated into "the organization of europe the organization of europe". This is because the attention mechanism focused on this phrase twice.

4 Our Methods

In traditional SMT methods, the distortion model controls the order of target word generation. It can thus prevent the meaning of source sentences to be twisted due to wrong reordering. We propose to address the incorrect reordering problem using a implicit distortion model which leverages information about previous alignments.

In traditional SMT methods, the fertility model controls how many target words should be generated from a source word. It can thus prevent a source word to be repeatedly translated, which corresponds to the repeated translation problem, or not translated, which corresponds to the missing translation problem. We propose to address the missing and repeated translation problems in NMT by by using a fertility model which memorizes which words have been translated and which have not.

In the following sections, we introduce our extended attention-based encoder-decoder models. For implicit distortion model, we propose a recurrent attention mechanism, RECATT; for implicit fertility model, we propose a fertility-conditioned decoder FERTDEC.

4.1 RECATT

The structure of RECATT is illustrated in Figure 6. At position i in the target sentence, the attention model outputs a weight vector for the encoder states and a weighted-average context. To inform the attention model about the previous alignments, we pass the previous context vector $\boldsymbol{c}_{i-1}$ to it. The decoder with RECATT follows:

$$\boldsymbol{w}_i = \text{ALIGN}(\boldsymbol{h}_{i-1}, \boldsymbol{c}_{i-1}, \boldsymbol{s}_1^J)$$

$$\boldsymbol{c}_i = \sum_{j=1}^{J} w_{ij} \boldsymbol{s}_j$$

$$\boldsymbol{h}_i = \text{RNN}(\boldsymbol{h}_{i-1}, \boldsymbol{y}_{i-1}, \boldsymbol{c}_i)$$

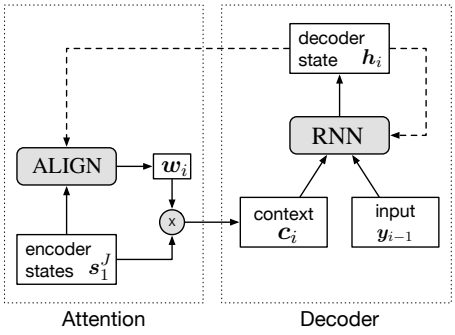


Figure 5. Decoder with attention mechanism. The dashed lines denote passing the previous state to the current attention model and current state.

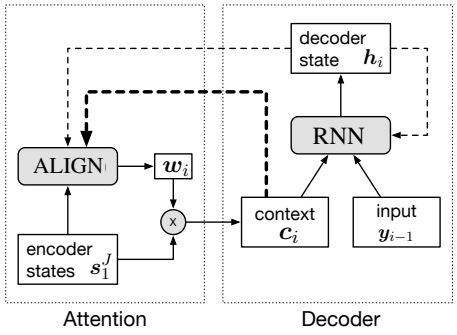


Figure 6. RECATT, decoder with recurrent attention mechanism. The thick dashed line denotes passing the previous attention-generated context to the attention model.

The new ALIGN function with modified match function α is computed as:

$$\alpha(\boldsymbol{h}_{i-1}, \boldsymbol{c}_{i-1}, \boldsymbol{s}_j) = \boldsymbol{W}^\alpha \boldsymbol{h}_{i-1} + \boldsymbol{U}^\alpha \boldsymbol{c}_{i-1} + \boldsymbol{V}^\alpha \boldsymbol{s}_j$$
$$e_{ij} = \boldsymbol{v}^\top \tanh \alpha(\boldsymbol{h}_{i-1}, \boldsymbol{c}_{i-1}, \boldsymbol{s}_j)$$
$$w_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})}$$

By using the previous context vector, the new attention model can avoid focusing on the same position repeatedly, or jumping from the previous attended position incorrectly. We note that RNNSEARCH is a special case of RECATT where the previous context $\boldsymbol{c}_{i-1}$ is ignored in the match function.

One important design choice of RECATT is to use the previous context vector instead of the previous weight vector. Using the context vector makes the attention model aware of the content of source words, instead of the weight vector, which contains only the position information. Furthermore, the length of the source sentence is variable, so is the length of the weight vector. To use it in the match function, transformation to a fixed-length vector is needed. Possible methods including taking a fixed-size window or passing it through a convolution, both result in a local and partial recurrent information. When we need a long-distance jump from the previous attended position, especially out of the window, partial information might not suffice. Using the context vector, as in RECATT, is not restricted in this scenario. The attention model can always have full information about the previous alignments even when a long-distance jump happens, which makes the implicit distortion model much more flexible.

4.2 FERTDEC

To address the missing and repeated translation problems, we introduce fertility-conditioned decoder FERTDEC. FERTDEC uses a coverage vector [1] to represent the information of the source sentence that has not been translated. Initialized by the sum of source word embeddings $\sum_{j=1}^{J} \boldsymbol{x}_j$ and updated along the translation dynamically, our trainable coverage vector is different from the predefined condition vector used in (Wen et al., 2015). In order to leverage the coverage vector in decoding, we change the

[1]The coverage vector in our work plays a similar role with the one used in beam search decoder (Koehn, 2004).There are two major two differences between them: 1. our coverage vector is used as a soft constraint instead of a hard constraint. 2. we tract untranslated words instead of translated words.

decoding recurrent unit as follows:

$$
\begin{aligned}
\boldsymbol{d}_i &= \boldsymbol{e}_{i-1} \odot \boldsymbol{d}_{i-1} \\
\boldsymbol{r}_i &= \sigma(\boldsymbol{W}^r \boldsymbol{y}_{i-1} + \boldsymbol{U}^r \boldsymbol{h}_{i-1} + \boldsymbol{V}^r \boldsymbol{d}_i) \\
\boldsymbol{z}_i &= \sigma(\boldsymbol{W}^z \boldsymbol{y}_{i-1} + \boldsymbol{U}^z \boldsymbol{h}_{i-1} + \boldsymbol{V}^z \boldsymbol{d}_i) \\
\boldsymbol{e}_i &= \sigma(\boldsymbol{W}^e \boldsymbol{y}_{i-1} + \boldsymbol{U}^e \boldsymbol{h}_{i-1} + \boldsymbol{V}^e \boldsymbol{d}_i) \\
\boldsymbol{h}'_i &= \tanh(\boldsymbol{U}(\boldsymbol{r}_i \odot \boldsymbol{h}_{i-1}) + \boldsymbol{W}\boldsymbol{y}_{i-1} + \boldsymbol{V}\boldsymbol{d}_i) \\
\boldsymbol{h}_i &= (1 - \boldsymbol{z}_i) \odot \boldsymbol{h}'_i + \boldsymbol{z}_i \odot \boldsymbol{h}_{i-1} + \tanh(\boldsymbol{V}^h \boldsymbol{d}_i)
\end{aligned}
$$

where $\boldsymbol{d}_i$ is the coverage vector, $\boldsymbol{e}_i$ is the new added *extract gate*, which is used to update $\boldsymbol{d}_i$ based on the words that has been translated.

$\boldsymbol{d}_i$ is designed to track the untranslated words during decoding, so it is not expected to change drastically between consecutive time-stamps. Also, it should converge to zero at the end of the sentence. Therefore in the training stage, we update the loss function as follows:

$$
\sum_{i=1}^{T} -\log p(y_i) + \frac{1}{T} \sum_{i=1}^{T} ||\boldsymbol{d}_i - \boldsymbol{d}_{i-1}||_2 + ||\boldsymbol{d}_i||_2
$$

where the first term is the negative log-likelihood used in the encoder–decoder model. The new introduced second and third terms are **step-decay** and **left-over** costs. **Step-decay** cost prevents the extract gate from extracting too much information at each time-step. It is different than that of (Wen et al., 2015) [2]. While **left-over** cost ensures all the source words are translated after generating the whole target sentence.

5 Related Work

There are variations of the attention mechanism with recurrent paths similar to those in our recurrent attention mechanism. In this section, we put them in a general framework and compare them with ours.

INPUTFEED Input-feeding method (Luong et al., 2015) also has a recurrent path - the previous attention-generated context is passed to the decoder together with current one:

$$
\begin{aligned}
\boldsymbol{w}_i &= \text{ALIGN}(\boldsymbol{h}_{i-1}, \boldsymbol{s}_1^J) \\
\boldsymbol{c}_i &= \sum_{j=1}^{J} w_{ij} \boldsymbol{s}_j \\
\boldsymbol{h}_i &= \text{RNN}(\boldsymbol{h}_{i-1}, \boldsymbol{y}_{i-1}, \boldsymbol{c}_i, \boldsymbol{c}_{i-1})
\end{aligned}
$$

Using the previous context helps the decoder generate better target words, but it doesn't help the attention model select source words more accurately or generate better alignment. This makes INPUTFEED very different from our RECATT.

MARKOV In Markov condition model (Cohn et al., 2016), ξ takes a fixed-width window of the previous weight vector $\boldsymbol{w}_{i-1}$ and passes it to the attention model:

$$
\begin{aligned}
\xi(\boldsymbol{w}_{i-1}, j) &= [w_{i-1,j-k}, .., w_{i-1,j}, .., w_{i-1,j+k}]^\top \\
\boldsymbol{w}_i &= \text{ALIGN}(\boldsymbol{h}_{i-1}, \boldsymbol{s}_1^J, \xi(\boldsymbol{w}_{i-1})) \\
\boldsymbol{c}_i &= \sum_{j=1}^{J} w_{ij} \boldsymbol{s}_j
\end{aligned}
$$

This can be seen as a location-based counterpart of RECATT. As discussed in Section 4.1, this method is less flexible - it can only use partial recurrent information and is not content-aware.

[2] These two cost functions achieve similar result on our task, but our has no hyperparameter.

LOCFER In local fertility model (Cohn et al., 2016), ξ uses all previous weight vectors $\boldsymbol{w}_{<i}$ and computes the sum of previous attention weights.

$$\xi(\boldsymbol{w}_{<i}, j) = \left[\sum_{i'<i} w_{i',j-k}, .., \sum_{i'<i} w_{i',j+k}\right]^\top$$
$$\boldsymbol{w}_i = \text{ALIGN}(\boldsymbol{h}_{i-1}, \boldsymbol{s}_1^J, \xi(\boldsymbol{w}_{<i}))$$
$$\boldsymbol{c}_i = \sum_{j=1}^{J} w_{ij} \boldsymbol{s}_j$$

The intuition is to consider the fertility up to the current position, and use it to guide new alignment. This is done by a location-based recurrent path similar to that of MARKOV. LOCFER can prevent focusing nearby words already translated, and it is a blend of distortion and fertility model.

6 Experiments

6.1 Settings

Datasets We use NIST Chinese–English training set excluding Hong Kong Law and Hong Kong Hansard as the training set (500,000 sentence pairs after exclusion). The test set is Nist2005 (1082 sentence pairs). The validation set is Nist2003 (913 sentence pairs).

Following (Bahdanau et al., 2014), we use a vocabulary size of 30,000 for both source and target language, covering 97.4% and 98.9% of the words. Out-of-vocabulary words are replaced with a special token $\langle$UNK$\rangle$.

UNK Replacement With word alignment result on the training set generated by GIZA++ (Och and Ney, 2003), we build a translation table. We choose the most frequently aligned target word as the translation for each source word. UNK replacement is performed after the translation is completed, based on the alignment matrix generated by the attention model. If a target word is UNK, we replace it with the translation (from the translation table) of its aligned source word, the one with the highest attention weight.

Model & Baseline Two baseline systems are used in our experiment. The first one is HPSMT, our in-house implementation of hierarchical phrase-based SMT (Chiang, 2007) with standard features. For a fair comparison, the 4-gram language model is trained only with the target sentences of the training set. The second one is RNNSEARCH [3] (Cho et al., 2014b), the original attention-based encoder-decoder. Other compared models are our implementations of: INPUTFEED (Luong et al., 2015), MARKOV and LOCFER (Cohn et al., 2016) as discussed in Section 5.

Training Details For all the NMT models, the hidden GRU states are 1000-dimensional, source and target word embeddings are 620-dimensional. Dropout rate is 0.5. The settings of other hyperparameters follow (Bahdanau et al., 2014). Each model is trained with AdaGrad (Duchi et al., 2011) on a K40m GPU for approximately 4 days, finishing about $400,000$ updates, equivalent to 64 epochs.

6.2 Experiment Results

6.2.1 End-to-end Translation Quality

BLEU scores on the test set are shown in Table 1. The two proposed methods RECATT and FERTDEC both out-performed the original model RNNSEARCH. Note that RECATT gained the most improvement from UNK replacement, 5.04 BLEU points. The effectiveness of our UNK replacement depends largely on the quality of the alignment, so the gain can be seen as a measurement of alignment quality. This is an evidence that RECATT improved attention-generated alignment and as a result improved translation quality. The last line shows the results obtained by the combination of RECATT and FERTDEC, which further out-performed both models.

[3]The implementation of RNNSEARCH is from `https://github.com/mila-udem/blocks-examples`

	Before	After	Diff
HPSMT	/	32.25	/
RNNSEARCH	26.65	31.02	4.37
INPUTFEED	25.44	29.02	3.58
LOCFER	27.05	31.68	4.63
MARKOV	27.54	32.21	4.67
RECATT	28.10	33.14	5.04
FERTDEC	27.51	32.44	4.93
RECATT + FERTDEC	28.87	33.76	4.89

Table 1. BLEU scores w/o UNK replacement and the improvement from UNK replacement.

	SAER	AER
RNNSEARCH	54.75	44.13
RECATT	52.88	42.51
FERTDEC	52.70	42.37
RECATT + FERTDEC	52.40	42.11

Table 2. AER & SAER scores, lower is better.

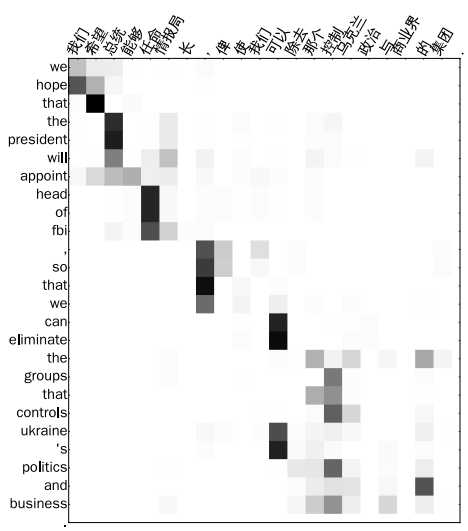


Figure 7. FERTDEC resolved the problem of missing translation problem that is shown in Figure 3.

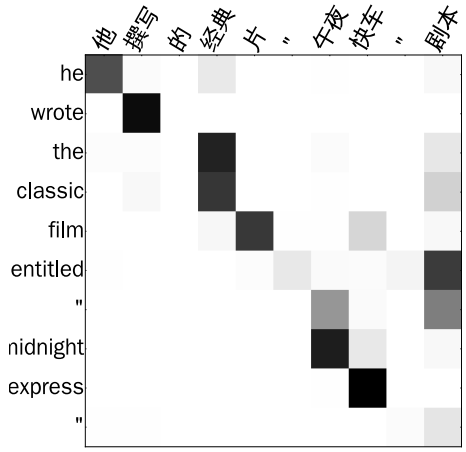


Figure 8. FERTDEC resolved the problem of repeated translation shown in Figure 4.

6.2.2 Alignment Quality

To analyze the effect of our extentions to the attention mechanism in detail, we evaluate the quality of attention-generated alignment by computing the AER (Och and Ney, 2003) and smoothed-AER (Tu et al., 2016) scores on a manually aligned Chinese–English alignment dataset (Haghighi et al., 2009), which contains 491 sentence pairs. We force the model to generate the correct target sentence and evaluate the attention-generated alignment matrix. From the results shown in Table 2, we can see that all three proposed methods achieved better alignment quality, compared with the original attention method.

6.3 Qualitative Analysis

In this section we qualitatively evaluate how our models addressed the problems analyzed in Section 3. All examples shown are from the test set.

Incorrect Reordering In Figure 2 we can see, RECATT generated the correct alignment on the example sentence shown in Figure 1: "will not" is correctly aligned to "不会" (means "will not") and "next year" is correctly translated after "the election to be held" instead of "riot in iraq". The meaning of the source sentence is correctly preserved in the translation.

Missing Translation As shown in Figure 7, FERTDEC resolved the missing translation problem of RNNSEARCH on the same sentence shown in Figure 3. All the information from the source sentence is captured by the translation.

Repeated Translation In Figure 8 we can see that, FERTDEC resolved the the repetition problem of RNNSEARCH shown in Figure 4. "东方 快车" (means "midnight express") is repeatedly focused on and translated into "night of the midnight of the night". As shown on the right, FERTDEC produces both the correct alignment and the correct translation "midnight express".

7 Conclusions and Future Work

In this paper we demonstrated how distortion and fertility models can improve the quality of alignment learned by attention mechanism in encoder-decoder models. We proposed recurrent attention mechanism RECATT as implicit distortion models, and FERTDEC as an implicit fertility model. We conducted various experiments and verified that our proposed methods can improve translation quality by generating better alignment. Compare to the original attention-based encoder-decoder, our best result achieved an improvement of over 2 BLEU points on large-scale Chinese–English translation task.

Our RECATT model is a simple yet effective extension to the attention mechanism, and potentially we can design more complicated mechanisms to model the distortion even better. The key observation is, in RECATT, only the previous context vector is used to provide information about previous alignments, and in effect only the alignment of the previous target word is considered. To extend this short-term information to a long-term one so that the model is aware of all previous alignments, we designed a attention unit that contains a recurrent neural network to encode all previous context vectors. The hidden state vector of this RNN should contain all the information about previous alignments. However in our experiment, this model and several variants did not perform as well as RECATT. But we still think that trying to provide more information about previous alignments, as a natural extension to this work, has the potential of improving both the alignment and translation accuracy.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2015. End-to-end attention-based large vocabulary speech recognition. *arXiv preprint arXiv:1508.04395*.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. 2012. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, volume 4, page 3. Austin, TX.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. 2013. Audio chord recognition with recurrent neural networks. In *ISMIR*, pages 335–340.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. 2015. Abc-cnn: An attention based convolutional neural network for visual question answering. *arXiv preprint arXiv:1511.05960*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. End-to-end continuous speech recognition using attention-based recurrent nn: First results. *arXiv preprint arXiv:1412.1602*.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. *arXiv preprint arXiv:1601.01085*.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1370–1380. Association for Computational Linguistics.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. *arXiv preprint arXiv:1302.4389*.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised itg models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 923–931. Association for Computational Linguistics.

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

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

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, volume 3, page 413.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Machine translation: From real users to research*, pages 115–124. Springer.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Kevin J Shih, Saurabh Singh, and Derek Hoiem. 2015. Where to look: Focus regions for visual question answering. *arXiv preprint arXiv:1511.07394*.

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

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Coverage-based neural machine translation. *arXiv preprint arXiv:1601.04811*.

Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. 2015. Blocks and fuel: Frameworks for deep learning. *arXiv preprint arXiv:1506.00619*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2014. Show and tell: A neural image caption generator. *arXiv preprint arXiv:1411.4555*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Huijuan Xu and Kate Saenko. 2015. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. *arXiv preprint arXiv:1511.05234*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

Neural Machine Translation with Supervised Attention

Lemao Liu, Masao Utiyama, Andrew Finch and Eiichiro Sumita
National Institute of Information and Communications Technology (NICT)
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan
{lmliu,first.last}@nict.go.jp

Abstract

The attention mechanism is appealing for neural machine translation, since it is able to dynamically encode a source sentence by generating a alignment between a target word and source words. Unfortunately, it has been proved to be worse than conventional alignment models in alignment accuracy. In this paper, we analyze and explain this issue from the point view of reordering, and propose a supervised attention which is learned with guidance from conventional alignment models. Experiments on two Chinese-to-English translation tasks show that the supervised attention mechanism yields better alignments leading to substantial gains over the standard attention based NMT.

1 Introduction

Neural Machine Translation (NMT) has achieved great successes on machine translation tasks recently (Bahdanau et al., 2015; Sutskever et al., 2015). Generally, it relies on a recurrent neural network under the Encode-Decode framework: it firstly encodes a source sentence into context vectors and then generates its translation token-by-token, selecting from the target vocabulary. Among different variants of NMT, attention based NMT, which is the focus of this paper,[1] is attracting increasing interests in the community (Bahdanau et al., 2015; Luong et al., 2015). One of its advantages is that it is able to dynamically make use of the encoded context through an attention mechanism thereby allowing the use of fewer hidden layers while still maintaining high levels of translation performance.

An attention mechanism is designed to predict the alignment of a target word with respect to source words (Bahdanau et al., 2015). In order to facilitate incremental decoding, it tries to make this alignment prediction without the information about the target word itself, and thus this attention can be considered to be a form of a reordering model (see §2 for more details). In contrast, conventional alignment models are able to use the target word to infer its alignments (Och and Ney, 2000; Dyer et al., 2013; Liu and Sun, 2015), and as a result there is a substantial gap in quality between the alignments derived by this attention based NMT and conventional alignment models (54 VS 30 in terms of AER for Chinese-to-English as reported in (Cheng et al., 2016)). This discrepancy might be an indication that the potential of attention-based NMT is limited. In addition, the attention in NMT is learned in an unsupervised manner without explicit prior knowledge about alignment.[2] However, in conventional statistical machine translation (SMT), it is standard practice to learn reordering models in a supervised manner with the guidance from conventional alignment models (Xiong et al., 2006; Koehn et al., 2007; Bisazza and Federico, 2016).

Inspired by the supervised reordering in conventional SMT, in this paper, we propose a *Supervised Attention* based NMT (SA-NMT) model. Specifically, similar to conventional SMT, we first run off-the-shelf aligners (GIZA++ (Och and Ney, 2000) or fast_align (Dyer et al., 2013) etc.) to obtain the alignment of the bilingual training corpus in advance. Then, treating this alignment result as the supervision of attention, we jointly learn attention and translation, both in supervised manners. Since the

[1] Throughout this paper, without the special statement, NMT means attention-based NMT.

[2] We do agree that NMT is a supervised model with respect to translation rather than reordering.

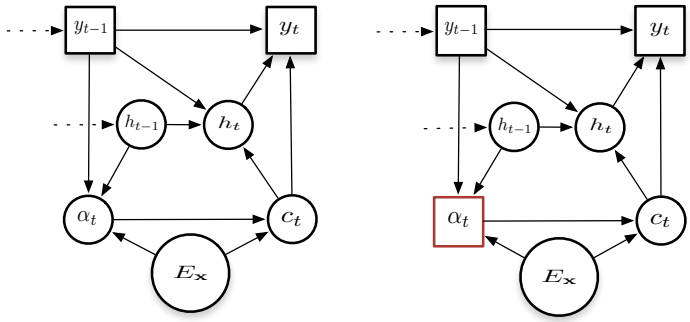
Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3093–3102, Osaka, Japan, December 11-17 2016.

conventional aligners delivers higher quality alignment, it is expected that the alignment in the supervised attention NMT will be improved leading to better end-to-end translation performance. One advantage of the proposed SA-NMT is that it implements the supervision of attention as a regularization in the joint training objective (§3.2). Furthermore, since the attention variables lies in the middle of the entire network architecture rather than the top (as the translation variables (see Figure 1(b)), it serves to mitigate the vanishing gradient problem during the back-propagation, by adding supervision into the intermediate layers in the network (Szegedy et al., 2015).

This paper makes the following contributions:

- It revisits the attention model from the point view of reordering (§2), and propose a supervised attention for NMT that is supervised by statistical alignment models (§3). The proposed approach is simple and easy to be implemented, and it is generally applicable to any attention-based NMT models, although in this case it is implemented on top of the model in (Bahdanau et al., 2015).
- On two Chinese-to-English translation tasks, it empirically shows that the proposed approach gives rise to improved performance (§4): on a large scale task, it outperforms three baselines including a state-of-the-art Moses, and leads to improvements of up to 2.5 BLEU points over the strongest one in this paper; on a low resource task, it even obtains about 5 BLEU points over the attention based NMT system on which is it based.

2 Revisiting Neural Machine Translation



(a) NMT (b) SA-NMT

Figure 1: One slice of the computational graphs for both (a) NMT and (b) SA-NMT. Circles denote the hidden variables; while squares denote the observable variables, which receive supervision during training. The difference (marked in red) in (b) regarding to (a) is treating α_t as an observable variable instead of a hidden variable.

Suppose $\mathbf{x} = \langle x_1, x_2, \cdots, x_m \rangle$ denotes a source sentence, $\mathbf{y} = \langle y_1, y_2, \cdots, y_n \rangle$ a target sentence. In addition, let $x_{<t} = \langle x_1, x_2, \cdots, x_{t-1} \rangle$ denote a prefix of $\mathbf{x}$. Neural Machine Translation (NMT) directly maps a source sentence into a target under an encode-decode framework. In the encoding stage, it uses two bidirectional recurrent neural networks to encode $\mathbf{x}$ into a sequence of vectors $E_{\mathbf{x}} = \langle E_{x_1}, E_{x_2}, \cdots, E_{x_m} \rangle$, with E_{x_i} representing the concatenation of two vectors for i_{th} source word from two directional RNNs. In the decoding stage, it generates the target translation from the conditional probability over the pair of sequences $\mathbf{x}$ and $\mathbf{y}$ via a recurrent neural network parametrized by θ as follows:

$$p(\mathbf{y} \mid \mathbf{x}; \theta) = \prod_{t=1}^{n} p(y_t \mid y_{<t}, E_{\mathbf{x}}) = \prod_{t=1}^{n} \text{softmax}\big(g(y_{t-1}, h_t, c_t)\big)[y_t] \quad (1)$$

where h_t and c_t respectively denote an RNN hidden state (i.e. a vector) and a context vector at timestep t; g is a transformation function mapping into a vector with dimension of the target vocabulary size; and $[i]$ denotes the i_{th} component of a vector.[3] Furthermore, $h_t = f(h_{t-1}, y_{t-1}, c_t)$ is defined by an activation

[3]In that sense, y_t in Eq.(1) also denotes the index of this word in its vocabulary.

function, i.e. a Gated Recurrent Unit (Chung et al., 2014); and the context vector c_t is a dynamical source representation at timestep t, and calculated as the weighted sum of source encodings $E_{\mathbf{x}}$, i.e. $c_t = \alpha_t^\top E_{\mathbf{x}}$. Here the weight α_t implements an attention mechanism, and $\alpha_{t,i}$ is the alignment probability of y_t being aligned to x_i. α_t is derived through a feedforward neural network a as follows:

$$\alpha_t = a(y_{t-1}, h_{t-1}, E_{\mathbf{x}}) \tag{2}$$

where a consists of two layers, the top one being a softmax layer. We skip the detailed definitions of a together with $E_{\mathbf{x}}$, f and g, and refer the readers to (Bahdanau et al., 2015) instead.[4] Figure 1(a) shows one slice of computational graph for NMT definition at time step t.

To train NMT, the following negative log-likelyhood is minimized:

$$-\sum_i \log p(\mathbf{y}^i \mid \mathbf{x}^i; \theta) \tag{3}$$

where $\langle \mathbf{x}^i, \mathbf{y}^i \rangle$ is a bilingual sentence pair from a given training corpus, $p(\mathbf{y}^i \mid \mathbf{x}^i; \theta)$ is as defined in Eq.(1). Note that even though the training is conducted in a supervised manner with respect to translation, i.e., $\mathbf{y}$ are observable in Figure 1(a), the attention is learned in a unsupervised manner, since α is hidden.

In Eq.(2), α_t is defined only on y_{t-1}, h_{t-1} and $E_{\mathbf{x}}$ but not on the target word y_t, as y_t is unknown at the current timestep $t-1$ during the testing. Therefore, at timestep $t-1$, NMT firstly tries to calculate α_t, through which NMT figures out those source words will be translated next, even though the next target word y_t is unavailable. From this point of view, the attention mechanism plays a role in reordering and thus can be considered as a reordering model. Unlike this attention model, conventional alignment models define the alignment α directly over $\mathbf{x}$ and $\mathbf{y}$ as follows:

$$p(\alpha \mid \mathbf{x}, \mathbf{y}) = \frac{\exp(F(\mathbf{x}, \mathbf{y}, \alpha))}{\sum_{\alpha'} \exp(F(\mathbf{x}, \mathbf{y}, \alpha'))}$$

where F denotes a feature function over a pair of sentences $\mathbf{x}$ and $\mathbf{y}$ together with their word alignment α, and it is either a log-probability $\log p(\mathbf{y}, \alpha \mid \mathbf{x})$ for a generative model like IBM models (Brown et al., 1993) or a well-designed feature function for discriminative models (Liu and Sun, 2015). In order to infer α_t, alignment models can readily use the entire $\mathbf{y}$, of course including y_t as well, thereby they can model the alignment between $\mathbf{x}$ and $\mathbf{y}$ more sufficiently. As a result, the attention based NMT might not deliver satisfying alignments, as reported in (Cheng et al., 2016), compared to conventional alignment models. This may be a sign that the potential of attention-based NMT is limited in end-to-end translation.

3 Supervised Attention

In this section, we introduce supervised attention to improve the alignment, which may lead to better translation performance for NMT. [5] Our basic idea is simple: similar to conventional SMT, it firstly uses a conventional aligner to obtain the alignment on the training corpus; then it employs these alignment results as supervision to train the NMT. During testing, decoding proceeds in exactly the same manner as standard NMT, since there is no alignment supervision available for unseen test sentences.

3.1 Preprocessing Alignment Supervision

As described in §2, the attention model outputs a soft alignment α, such that α_t is a normalized probability distribution. In contrast, most aligners are typically oriented to grammar induction for conventional SMT, and they usually output 'hard' alignments, such as (Och and Ney, 2000). They only indicate whether a target word is aligned to a source word or not, and this might not correspond to a distribution for each target word. For example, one target word may align to multiple source words, or no source words at all.

[4] In the original paper, α_t is not explicitly dependent on the y_{t-1} in Eq.(2), but this dependency was explicitly retained in our direct baseline NMT2.

[5] Although the alignment is loosely related to the downstream translation (Liu and Sun, 2015), substantial improvements in alignment usually leads to the improvements in translation as observed in our experiments.

Therefore, we apply the following heuristics to preprocess the hard alignment: if a target word does not align to any source words, we inherit its affiliation from the closest aligned word with preference given to the right, following (Devlin et al., 2014); if a target word is aligned to multiple source words, we assume it aligns to each one evenly. In addition, in the implementation of NMT, there are two special tokens 'eol' added to both source and target sentences. We assume they are aligned to each other. In this way, we can obtain the final supervision of attention, denoted as $\hat{\alpha}$.

3.2 Jointly Supervising Translation and Attention

We propose a soft constraint method to jointly supervise the translation and attention as follows:

$$-\sum_i \log p(\mathbf{y}^i \mid \mathbf{x}^i; \theta) + \lambda \times \Delta(\alpha^i, \hat{\alpha}^i; \theta) \tag{4}$$

where α^i is as defined in Eq. (1), Δ is a loss function that penalizes the disagreement between α^i and $\hat{\alpha}^i$, and $\lambda > 0$ is a hyper-parameter that balances the preference between likelihood and disagreement. In this way, we treat the attention variable α as an observable variable as shown in Figure 1(b), and this is different from the standard NMT as shown in Figure 1(a) in essence. Note that this training objective resembles to that in multi-task learning (Evgeniou and Pontil, 2004). Our supervised attention method has two further advantages: firstly, it is able to alleviate overfitting by means of the λ; and secondly it is easier to address the vanishing gradient problem by adding supervision into the intermediate layers of the entire network (Szegedy et al., 2015), because the supervision of α is more close to $E_{\mathbf{x}}$ than $\mathbf{y}$ as in Figure 1(b).

In order to quantify the disagreement between α^i and $\hat{\alpha}^i$, three different methods are investigated in our experiments:

- *Mean Squared Error* (MSE)

$$\Delta(\alpha^i, \hat{\alpha}^i; \theta) = \sum_m \sum_n \frac{1}{2}\big(\alpha(\theta)^i_{m,n} - \hat{\alpha}^i_{m,n}\big)^2$$

  MSE is widely used as a loss for regression tasks (Lehmann and Casella, 1998), and it directly encourages $\alpha(\theta)^i_{m,n}$ to be equal to $\hat{\alpha}^i_{m,n}$.

- *Multiplication* (MUL)

$$\Delta(\alpha^i, \hat{\alpha}^i; \theta) = -\log\big(\sum_m \sum_n \alpha(\theta)^i_{m,n} \times \hat{\alpha}^i_{m,n}\big)$$

  MUL is particularly designed for agreement in word alignment and it has been shown to be effective (Liang et al., 2006; Cheng et al., 2016). Note that different from those in (Cheng et al., 2016), $\hat{\alpha}$ is not a parametrized variable but a constant in this paper.

- *Cross Entropy* (CE)

$$\Delta(\alpha^i, \hat{\alpha}^i; \theta) = -\sum_m \sum_n \hat{\alpha}^i_{m,n} \times \log \alpha(\theta)^i_{m,n}$$

  Since for each t, $\alpha(\theta)_t$ is a distribution, it is natural to use CE as the metric to evaluate the disagreement (Rubinstein and Kroese, 2004).

4 Experiments

We conducted experiments on two Chinese-to-English translation tasks: one is the NIST task oriented to NEWS domain, which is a large scale task and suitable to NMT; and the other is the speech translation oriented to travel domain, which is a low resource task and thus is very challenging for NMT. We used the case-insensitive BLEU4 to evaluate translation quality and adopted the multi-bleu.perl as its implementation.

Alignment Losses	BLEU
Mean Squared Error (MSE)	39.4
Multiplication (MUL)	39.6
Cross Entropy (CE)	40.0

Table 1: Performance of SA-NMT on development set for different loss functions to supervise the attention in terms of BLEU.

Alignment Methods	BLEU
fast_align	39.6
GIZA++	40.0

Table 2: Comparision of aligners between fast_align and GIZA++ for SA-NMT in terms of BLEU on the development set.

4.1 The Large Scale Translation Task

4.1.1 Preparation

We used the data from the NIST2008 Open Machine Translation Campaign. The training data consisted of 1.8M sentence pairs, the development set was nist02 (878 sentences), and the test sets are were nist05 (1082 sentences), nist06 (1664 sentences) and nist08 (1357 sentences).

We compared the proposed approach with three strong baselines:

- Moses: a phrase-based machine translation system (Koehn et al., 2007);
- NMT1: an attention based NMT (Bahdanau et al., 2015) system at https://github.com/lisa-groundhog/GroundHog;
- NMT2: another implementation of (Bahdanau et al., 2015) at https://github.com/nyu-dl/dl4mt-tutorial.

We developed the proposed approach based on NMT2, and denoted it as **SA-NMT**.

We followed the standard pipeline to run Moses. GIZA++ with grow-diag-final-and was used to build the translation model. We trained a 5-gram target language model on the Gigaword corpus, and used a lexicalized distortion model. All experiments were run with the default settings.

To train NMT1, NMT2 and SA-NMT, we employed the same settings for fair comparison. Specifically, except the stopping iteration which was selected using development data, we used the default settings set out in (Bahdanau et al., 2015) for all NMT-based systems: the dimension of word embedding was 620, the dimension of hidden units was 1000, the batch size was 80, the source and target side vocabulary sizes were 30000, the maximum sequence length was 50, [6] the beam size for decoding was 12, and the optimization was done by Adadelta with all hyper-parameters suggested by (Zeiler, 2012). Particularly for SA-NMT, we employed a conventional word aligner to obtain the word alignment on the training data before training SA-NMT. In this paper, we used two different aligners, which are fast_align and GIZA++. We tuned the hyper-parameter λ to be 0.3 on the development set, to balance the preference between the translation and alignment. Training was conducted on a single Tesla K40 GPU machine. Each update took about 3.0 seconds for both NMT2 and SA-NMT, and 2.4 seconds for NMT1. Roughly, it took about 10 days to NMT2 to finish 300000 updates.

4.1.2 Settings on External Alignments

We implemented three different losses to supervise the attention as described in §3.2. To explore their behaviors on the development set, we employed the GIZA++ to generate the alignment on the training set prior to the training SA-NMT. In Table 1, we can see that MUL is better than MSE. Furthermore, CE performs best among all losses, and thus we adopt it for the following experiments.

[6] This excludes all the sentences longer than 50 words in either source or target side only for NMT systems, but for Moses we use the entire training data.

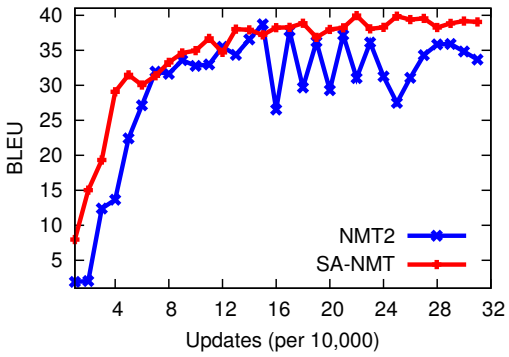


Figure 2: Learning curves of NMT2 and SA-NMT on the development set.

Systems	nist02	nist05	nist06	nist08
Moses	37.1	35.1	33.4	25.9
NMT1	37.8	34.1	34.7	27.4
NMT2	38.7	35.3	36.0	27.8
SA-NMT	40.0*	37.8*	37.6*	29.9*

Table 3: BLEU comparison for large scale translation task. The development set is nist02, and the test sets are nist05,nist06 and nist08. '*' denotes that SA-NMT is significantly better than Moses, NMT1 and NMT2 with $p < 0.01$. Note that Moses is trained with more bilingual sentences and an additional monolingual corpus.

In addition, we also run fast_align to generate alignments as the supervision for SA-NMT and the results were reported in Table 2. We can see that GIZA++ performs slightly better than fast_align and thus we fix the external aligner as GIZA++ in the following experiments.

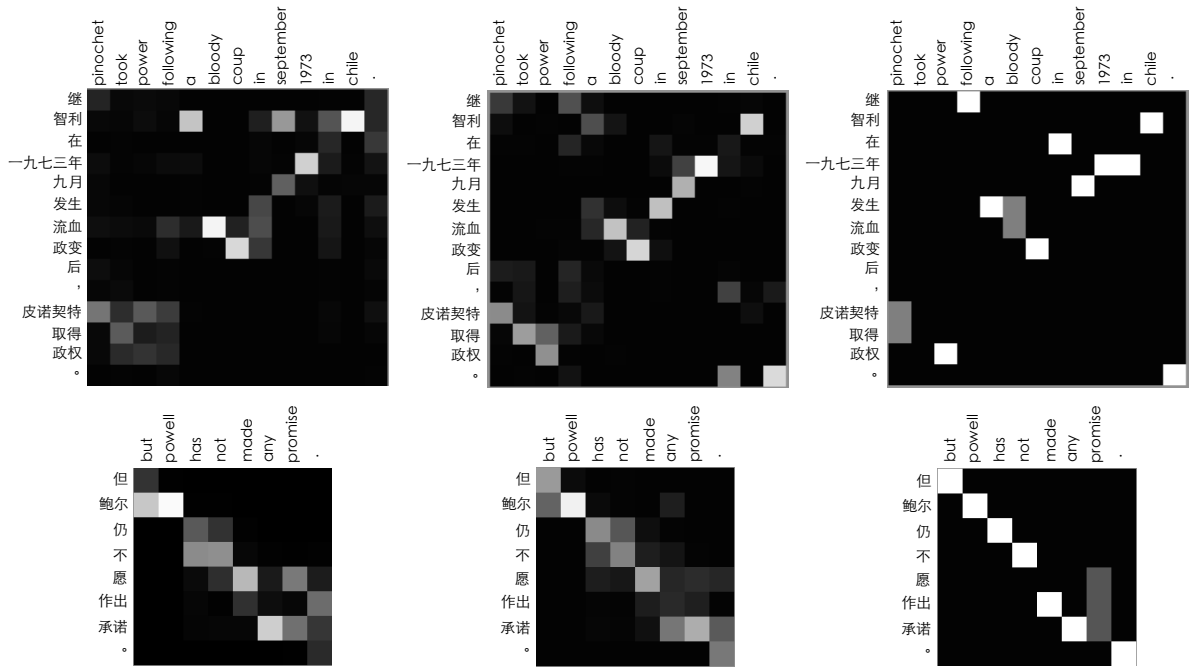
4.1.3 Results on Large Scale Translation Task

Figure 2 shows the learning curves of NMT2 and SA-NMT on the development set. We can see that NMT2 generally obtains higher BLEU as the increasing of updates before peaking at update of 150000, while it is unstable from then on. On the other hand, SA-NMT delivers much better BLEU for the beginning updates and performs more steadily along with the updates, although it takes more updates to reach the peaking point.

Table 3 reports the main end-to-end translation results for the large scale task. We find that both standard NMT generally outperforms Moses except NMT1 on nist05. The proposed SA-NMT achieves significant and consistent improvements over all three baseline systems, and it obtains the averaged gains of 2.2 BLEU points on test sets over its direct baseline NMT2. It is clear from these results that our supervised attention mechanism is highly effective in practice.

4.1.4 Results and Analysis on Alignment

As explained in §2, standard NMT can not use the target word information to predict its aligned source words, and thus might fail to predict the correct source words for some target words. For example, for the sentence in the training set in Figure 3 (a), NMT2 aligned 'following' to '皮诺契特 (gloss: pinochet)' rather than '继 (gloss: follow)', and worse still it aligned the word '.' to '在 (gloss: in)' rather than '。' even though this word is relatively easy to align correctly. In contrast, with the help of information from the target word itself, GIZA++ successfully aligned both 'following' and '.' to the expected source words (see Figure3(c)). With the alignment results from GIZA++ as supervision, we can see that our SA-NMT can imitate GIZA++ and thus align both words correctly. More importantly, for sentences in the unseen test set, like GIZA++, SA-NMT confidently aligned 'but' and '.' to their correct source words respectively as in Figure3(b), where NMT2 failed. It seems that SA-NMT can learn its alignment behavior from GIZA++, and subsequently apply the alignment abilities it has learned to unseen test sentences.



(a) NMT2 (b) SA-NMT (c) GIZA++

Figure 3: Example (soft) alignments of (a) NMT2 (i.e., standard NMT with unsupervised attention), (b) SA-NMT (i.e. NMT with supervised attention), and (c) GIZA++ on two Chinese-English sentence pairs. The soft alignments of (c) is converted from hard alignment as in §3.1. The first row shows the alignments of the sentence pair from the training set while the second row shows the alignments from test sets.

Methods	AER
GIZA++	30.6*
NMT2	50.6
SA-NMT	43.3*

Table 4: Results on word alignment task for the large scale data. The evaluation metric is Alignment Error Rate (AER). '*' denotes that the corresponding result is significanly better than NMT2 with $p < 0.01$.

Table 4 shows the overall alignment results on word alignment task in terms of the metric, alignment error rate. We used the manually-aligned dataset as in (Liu and Sun, 2015) as the test set. Following (Luong and Manning, 2015), we force-decode both the bilingual sentences including source and reference sentences to obtain the alignment matrices, and then for each target word we extract one-to-one alignments by picking up the source word with the highest alignment confidence as the hard alignment. From Table 4, we can see clearly that standard NMT (NMT2) is far behind GIZA++ in alignment quality. This shows that it is possible and promising to supervise the attention with GIZA++. With the help from GIZA++, our supervised attention based NMT (SA-NMT) significantly reduces the AER, compared with the unsupervised counterpart (NMT2). This shows that the proposed approach is able to realize our intuition: the alignment is improved, leading to better translation performance.

Note that there is still a gap between SA-NMT and GIZA++ as indicated in Table 4. Since SA-NMT was trained for machine translation instead of word alignment, it is possible to reduce its AER if we aim to the word alignment task only. For example, we can enlarge λ in Eq.(4) to bias the training objective towards word alignment task, or we can change the architecture slightly to add the target information crucial for alignment as in (Yang et al., 2013; Tamura et al., 2014).

Systems	CSTAR03	IWSLT04
Moses	44.1	45.1
NMT1	33.4	33.0
NMT2	36.5	35.9
SA-NMT	39.8*	40.7*

Table 5: BLEU comparison for low-resource translation task. CSTAR03 is the development set while IWSLT04 is the test set. '*' denotes that SA-NMT is significantly better than both NMT1 and NMT2 with $p < 0.01$.

4.2 Results on the Low Resource Translation Task

For the low resource translation task, we used the BTEC corpus as the training data, which consists of 30k sentence pairs with 0.27M Chinese words and 0.33M English words. As development and test sets, we used the CSTAR03 and IWSLT04 held out sets, respectively. We trained a 4-gram language model on the target side of training corpus for running Moses. For training all NMT systems, we employed the same settings as those in the large scale task, except that vocabulary size is 6000, batch size is 16, and the hyper-parameter $\lambda = 1$ for SA-NMT.

Table 5 reports the final results. Firstly, we can see that both standard neural machine translation systems NMT1 and NMT2 are much worse than Moses with a substantial gap. This result is not difficult to understand: neural network systems typically require sufficient data to boost their performance, and thus low resource translation tasks are very challenging for them. Secondly, the proposed SA-NMT gains much over NMT2 similar to the case in the large scale task, and the gap towards Moses is narrowed substantially.

While our SA-NMT does not advance the state-of-the-art Moses as in large scale translation, this is a strong result if we consider that previous works on low resource translation tasks: Arthur et al. (2016) gained over Moses on the Japanese-to-English BTEC corpus, but they resorted to a corpus consisting of 464k sentence pairs; Luong and Manning (2015) revealed the comparable performance to Moses on English-to-Vietnamese with 133k sentences pairs, which is more than 4 times of our corprus size. Our method is possible to advance Moses by using reranking as in (Neubig et al., 2015; Cohn et al., 2016), but it is beyond the scope of this paper and instead we remain it as future work.

5 Related Work

Many recent works have led to notable improvements in the attention mechanism for neural machine translation. Tu et al. (2016) introduced an explicit coverage vector into the attention mechanism to address the over-translation and under-translation inherent in NMT. Feng et al. (2016) proposed an additional recurrent structure for attention to capture long-term dependencies. Cheng et al. (2016) proposed an agreement-based bidirectional NMT model for symmetrizing alignment. Cohn et al. (2016) incorporated multiple structural alignment biases into attention learning for better alignment. All of them improved the attention models that were learned in an unsupervised manner. While we do not modify the attention model itself, we learn it in a supervised manner, therefore our approach is orthogonal to theirs.

It has always been standard practice to learn reordering models from alignments for conventional SMT either at the phrase level or word level. At the phrase level, Koehn et al. (2007) proposed a lexicalized MSD model for phrasal reordering; Xiong et al. (2006) proposed a feature-rich model to learn phrase reordering for BTG; and Li et al. (2014) proposed a neural network method to learn a BTG reordering model. At the word level, Bisazza and Federico (2016) surveyed many word reordering models learned from alignment models for SMT, and there are some neural network based reordering models, such as (Zhang et al., 2016). Our work is inspired by these works in spirit, and it can be considered to be a recurrent neural network based word-level reordering model. The main difference is that in our approach the reordering model and translation model are trained jointly rather than separately as theirs.

Supervising the attention variables for attention-based neural networks is pioneered by Liu et al.

(2016). On image caption task, Liu et al. (2016) supervise the attention with external guidances in either a strong or a weak supervision manner. Their method requires the training data to be associated with direct annotation or indirect annotation. In parallel to our work, particularly on machine translation, Mi et al. (2016) and Chen et al. (2016) guide the attention for NMT from conventional word alignment models as teachers without any annotation on machine translation task. The differences of our work lie in that: we consider the attention as a form of a reordering model, which is thereby straightforward to be learned from conventional word alignment models; and we also provide a theoretical explanation why the attention leads to the worse alignment accuracy than the conventional word alignment models, standing upon the point view of reordering.

6 Conclusion

It has been shown that attention mechanism in NMT is worse than conventional word alignment models in its alignment accuracy. This paper firstly provides an explanation for this by viewing the attention mechanism from the point view of reordering. Then it proposes a supervised attention for NMT with guidance from external conventional alignment models, inspired by the supervised reordering models in conventional SMT. Experiments on two Chinese-to-English translation tasks show that the proposed approach achieves better alignment results leading to significant gains relative to standard attention based NMT.

Acknowledgements

We would like to thank Xugang Lu for invaluable discussions and three anonymous reviewers for many valuable comments and helpful suggestions on this work.

References

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. *CoRR*, abs/1606.02006.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Arianna Bisazza and Marcello Federico. 2016. A survey of word reordering in statistical machine translation: Computational models and language phenomena. *Computational Linguistics*, 42.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311.

Wenhu Chen, Evgeny Matusov, Shahram Khadivi, and Jan-Thorsten Peter. 2016. Guided alignment training for topic-aware neural machine translation. In *Proceedings of AMTA*.

Yong Cheng, Shiqi Shen, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Agreement-based joint training for bidirectional attention-based neural machine translation. In *Proceedings of IJCAI*.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

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

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of ACL*.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *In Proc. NAACL*.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04.

Shi Feng, Shujie Liu, Mu Li, and Ming Zhou. 2016. Implicit distortion and fertility models for attention-based encoder-decoder NMT model. *CoRR*, abs/1601.03317.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of ACL: Demonstrations*.

E.L. Lehmann and G. Casella. 1998. *Theory of Point Estimation*. Springer Verlag.

Peng Li, Yang Liu, Maosong Sun, Tatsuya Izuha, and Dakun Zhang. 2014. A neural reordering model for phrase-based translation. In *Proceedings of COLING*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of HLT-NAACL*.

Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features.

Chenxi Liu, Junhua Mao, Fei Sha, and Alan L. Yuille. 2016. Attention correctness in neural image captioning. *CoRR*, abs/1605.09553.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of IWSLT*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proceedings of EMNLP*.

Graham Neubig, Makoto Morishita, and Satoshi Nakamura. 2015. Neural reranking improves subjective quality of machine translation: NAIST at WAT2015. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL*, pages 440–447.

Reuven Y. Rubinstein and Dirk P. Kroese. 2004. *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-carlo Simulation (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2015. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2014. Recurrent neural networks for word alignment model. In *Proceedings of ACL*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of ACL*.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of ACL*.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word alignment modeling with context dependent deep neural network. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, August.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*.

Jingyi Zhang, Masao Utiyama, Eiichiro Sumita, Hai Zhao, Graham Neubig, and Satoshi Nakamura. 2016. Learning local word reorderings for hierarchical phrase-based statistical machine translation. *Machine Translation*.

Lightly Supervised Quality Estimation

Matthias Sperber[1,⋆], **Graham Neubig**[2], **Jan Niehues**[1], **Sebastian Stüker**[1], **Alex Waibel**[1]
[1]Karlsruhe Institute of Technology, Germany
[2]Carnegie Mellon University, USA
⋆matthias.sperber@kit.edu

Abstract

Evaluating the quality of output from language processing systems such as machine translation or speech recognition is an essential step in ensuring that they are sufficient for practical use. However, depending on the practical requirements, evaluation approaches can differ strongly. Often, reference-based evaluation measures (such as BLEU or WER) are appealing because they are cheap and allow rapid quantitative comparison. On the other hand, practitioners often focus on manual evaluation because they must deal with frequently changing domains and quality standards requested by customers, for which reference-based evaluation is insufficient or not possible due to missing in-domain reference data (Harris et al., 2016). In this paper, we attempt to bridge this gap by proposing a framework for lightly supervised quality estimation. We collect manually annotated scores for a small number of segments in a test corpus or document, and combine them with automatically predicted quality scores for the remaining segments to predict an overall quality estimate. An evaluation shows that our framework estimates quality more reliably than using fully automatic quality estimation approaches, while keeping annotation effort low by not requiring full references to be available for the particular domain.

1 Introduction

Quality evaluation is a key requirement for developing and employing language technology, enabling users and engineers to judge overall quality of the output, detect key problems, improve systems, and choose among competing systems. Although most users and engineers share these goals, the chosen evaluation approaches can differ strongly, with some people resorting to automatic, reference-based evaluation, while others rely on manual evaluation for their purposes. This is especially pronounced in the case of machine translation (MT), as pointed out by Harris et al. (2016). On one hand, much research effort has been devoted to devising reference-based methods such as BLEU (Papineni et al., 2002) that are well-correlated with human judgment. On the other hand, practitioners need to react to changing domains from customer to customer, and reflect multi-faceted quality requirements that are difficult to measure in a single, generic score, often leaving manual evaluation as the only choice.

In recent years, automatic quality estimation (QE) has emerged as a method that could potentially address the lack of flexibility of reference-based evaluation to deal with changing requirements, and the high effort of manual evaluation. Automatic QE uses machine learning techniques that are trained on a dataset of output-quality pairs in order to predict the quality of some new system output. It can be used to predict arbitrary quality metrics, provided that suitably labeled training data is available. However, in practice automatic QE has been found difficult and not reliable enough when applied to new domains or unknown systems, e.g. in MT (de Souza et al., 2015a) and automatic speech recognition (ASR) (Negri et al., 2014).

In this work, we explore a middle ground between automatic QE and manual evaluation, aiming to allow the QE system to adapt to the particular test data under consideration while keeping manual effort at an affordable level. We refer to this approach as *lightly supervised*[1] QE. Our general approach is

[1]In this paper, the terms supervised and unsupervised refer to whether or not manual annotation is necessary at *test* time.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3103–3113, Osaka, Japan, December 11-17 2016.

to divide the test data into smaller segments such as sentences or utterances, obtain automatic quality estimates for all these segments, and manually annotate only a small number of segments. We then compute aggregated quality estimates by averaging over segment-level scores, and explore two ways to aggregate manual and automatic scores.

The first aggregation approach computes the overall quality by averaging over *only the automatic* segment scores, using the manual annotations to adapt the automatic QE regressor. The method has the potential to improve predictions by reflecting not only the topical domain, but also the particular ASR or MT system, and even use-case specific quality standards.

The second aggregation approach computes quality by averaging over only the collected *manual segment scores*, using the automatic scores to choose what segments should be manually evaluated. In particular, we expect segments that are predicted to be close to the average across segments to be most indicative of overall quality. Instead of QE predictions, this aggregation approach can also directly exploit confidence scores from the decoder, eliminating the need for training a QE regressor.

We evaluate our approach for a variety of situations, focusing on the output of MT and ASR systems. We find that for MT, we can achieve a desired accuracy of quality estimation with much less effort compared to fully manual annotation. For instance, when annotating 100 words, in an in-domain setting we reduce the error by 22% and 19% relative over a fully automatic and a fully manual baseline. In an out-of-domain setting, relative improvements are 10% and 71%. For ASR, we obtain no improvements using the regression approach, but obtain promising results when using confidence scores instead: At 100 annotated words, the relative improvements are 59% and 54%.

2 Lightly Supervised Estimation Framework

This section outlines our lightly supervised estimation framework. The input to the framework is the hypothesized translation (transcription) of some MT (ASR) system given some particular input document (audio). We will refer to these as *hypothesis*, *system*, and *document* throughout the paper. Here, the document can be any form of test corpus that is representative of our targeted application, such as a document written in a particular style, data from one or several speakers, or a topical domain. Our goal is to estimate the average quality of the generated target sentences for a particular document with respect to some evaluation measure. The evaluation measure can be chosen arbitrarily, the only requirement being that it can be assigned on a per-sentence level. Possible examples are translation edit rate (TER) or sentence-level BLEU for MT, word error rate (WER) for ASR, or human ratings. We assume that the hypothesis is segmented in some way. The choice of segmentation is arbitrary, but sentence or utterance boundaries are a natural choice.

For purposes of this paper, we define document-level quality as the weighted average of the segment-level scores. Let ALL denote the index set of all segments in our document, y_i and w_i the true quality and weight of the i-th segment. The overall quality $Q^{(\text{true})}$ to be estimated is defined as:

$$Q^{(\text{true})} := \sum_{i \in \text{ALL}} w_i y_i \tag{1}$$

Note that this definition does not aim to handle document-level discourse phenomena such as coherence, cohesion, and and consistency, but estimates the average sentence-level quality for the document in order to evaluate the overall quality of the whole output hypothesis. In this paper, we weigh segments proportionally to their length, although future work may investigate more sophisticated notions of segment importance. Our definition of document quality is simplistic, but widely used in both MT and ASR communities, e.g. document- or corpus-level WER, TER, METEOR (Banerjee and Lavie, 2005), and human rankings are usually computed this way. BLEU is computed on the corpus level and thus not directly usable with our approach, but we can instead resort to computing average sentence-level BLEU variants such as BLEU+1 (Lin and Och, 2004) that essentially differ only in the smoothing details.

Our lightly supervised estimation framework determines document quality in several steps. The first step is to automatically estimate the quality for each segment in the hypothesis (§4). This can be achieved by training a regressor with the desired target measure, as discussed in numerous previous works. The

second step is then to manually annotate the quality score for a certain number of segments. In our evaluation (§5), we experiment with typical amounts of tens to hundreds of annotated words. The final step is to aggregate manual and automatic scores into a document-level estimate (§3).

3 Aggregation of Manual and Automatic Scores

We assume for a moment that we know how to collect automatic and manual segment-level scores, and discuss how they may be aggregated into a document-level estimate. Let $p_i, \forall i \in \text{ALL}$ denote the automatically predicted scores for all segments in the document, corresponding to the index set ALL. Let $y_i, \forall i \in \text{SEL}$ denote the manually annotated segment scores for a subset of all segments in a document, with indices $\text{SEL} \subseteq \text{ALL}$. w_i is the number of tokens in the system output for the i-th segment, and $Z^{(\mathcal{S})}$ is the total length of all segments indexed by $\mathcal{S}$. $Q^{(\text{true})}$ is the true document quality to be estimated.

3.1 Baselines

First, consider a *fully manual* baseline: Here, we randomly select segments from ALL to create SEL, and use the weighted average of manually annotated quality scores over these segments. Note that even though the segment-level scores y_i are reliable, the aggregate $Q^{(\text{man})}$ will be inaccurate if we keep the size of the annotated sample SEL small, as desired:

$$Q^{(\text{man})} = \frac{1}{Z^{(\text{SEL})}} \sum_{i \in \text{SEL}} w_i y_i \tag{2}$$

In contrast, the *fully automatic* baseline uses only the automatically predicted scores. Because automatic scores are available for all segments, the sample size is much bigger. On the other hand, the regressor used for prediction is often biased to predicting values p_i close to those indicated by the training data, which may differ considerably from the true values for y_i, especially for documents that are less similar to the training data. The formula is as follows:

$$Q^{(\text{auto})} = \frac{1}{Z^{(\text{ALL})}} \sum_{i \in \text{ALL}} w_i p_i \tag{3}$$

3.2 Bias vs. Variance: Why Aggregation is Challenging

Our goal is to improve over these baselines, and to motivate our methods we first frame these baselines in the context of the bias-variance tradeoff widely discussed in machine learning (Hastie et al., 2009). Here, the $\text{bias} = \mathbf{E}\big[Q^{(\text{est})} - Q^{(\text{true})}\big]^2$ describes the (squared) average error of the document-level estimate, with the expectation averaging over the randomness in the selection of annotated segments SEL, and other factors that one would consider random, such as particular training data for automatic QE, or randomness in the document that one might want to abstract from. The $\text{variance} = \mathbf{E}\big[\big(Q^{(\text{est})} - \mathbf{E}[Q^{(\text{est})}]\big)^2\big]$ describes by how much the estimates vary from one another, by again taking into account the randomness in the selection of segments, data, etc.

Based on these definitions, we observe that $Q^{(\text{man})}$ is subject to high variance, because the small sample size makes it sensitive to the randomness of the data. On the other hand, its bias is zero, since averaging over all randomness would cancel out estimation errors exactly. $Q^{(\text{auto})}$, on the other hand, has low variance because it always considers the whole document, but high bias for documents that are less similar to the data used to train the regressor. Figure 1 illustrates this interplay.

A straight-forward way to combine automatic and manual scores would be some form of interpolation:

$$Q^{(\text{interp})} = \alpha Q^{(\text{man})} + (1 - \alpha) Q^{(\text{auto})} \tag{4}$$

We experimented with several interpolation approaches in this spirit, but found that they do not work, because the high bias of the automatic estimate hurts accuracy more than could be compensated for by reduced variance of the interpolated estimate. The following subsections describe our refined aggregation strategies, which are more suited to solving this problem.

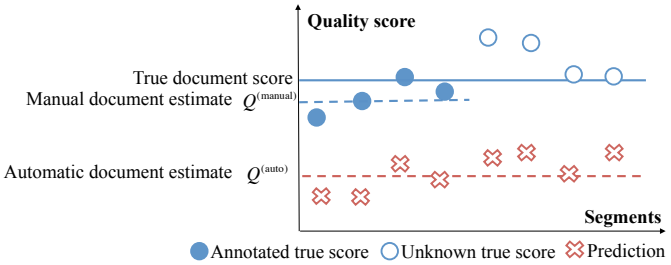


Figure 1: Schematic interplay between annotated and predicted estimates. The automatic aggregate has low variance because it includes all segments, but regressor-training data mismatch can lead to severely biased estimates. On the other hand, the manual annotations are unbiased per definition, but the potentially small number of samples causes high variance in the aggregate.

3.3 Proposed Strategy 1: Regressor Adaptation

In our first strategy, we use the annotated segments to adapt the regressor. This is done by casting the aggregation as a domain adaptation problem, in which we treat the original training data as background data, and the annotated segments of our document as in-domain data. Details for how the domain adaptation is performed are given in §4.1. As in the fully manual baseline, segments for annotation are selected at random. The document estimate is as in the fully automatic baseline, except that we now use the adapted regressor to produce the automatic scores $p_i^{(\text{adapt})}$. The hope is that the adaptation will reduce the regressor bias by shifting predictions closer to the document mean. The formula is as below:

$$Q^{(\text{adapt})} = \frac{1}{Z^{(\text{ALL})}} \sum_{i \in \text{ALL}} w_i p_i^{(\text{adapt})} \tag{5}$$

3.4 Proposed Strategy 2: Active Selection of Segments for Annotation

This approach attempts to select segments for manual annotation that are representative of the whole document. As a proxy for representativeness, we compute how close a segment's predicted quality is to the median prediction across the document. This choice is motivated by our definition of overall quality as the averaged segment-level quality. Specifically, we sort the segments by their automatic scores, and add the median[2] segment to SEL, the set of segments to annotate. Then, we add an equal number of segments to the left and to right according to this ordering, until the desired number of segments is reached. The document estimate is then calculated in the same manner as Equation 2.

The hope is that the active selection of annotated segments will reduce the sample variance. Note that unlike the first strategy, this one does not restrict the metric of the automatic scores to correspond to the target evaluation metric, because p_i does not appear directly in the sum. For instance, it would be possible to use segment-level BLEU scores or even confidence scores to compute SEL, despite our evaluation target being TER or human rating. The only requirement is that the predicted scores correlate with the target metric to some extent.

4 Automatic Segment Scores

4.1 Regression-based Approach

The first class of segment-level automatic scores we consider are QE scores obtained by a black box regressor that uses only surface features derived from the system input and output. It has the advantage of being agnostic to the interiors of the system that was used, and being able to predict any desired metric.

Features: We follow previous works for feature extraction. For MT, we use the 17 baseline features from the WMT QE shared task (Bojar et al., 2015), extracted using the QUEST toolkit (Specia et al., 2013). For ASR, we extracted signal-, hybrid-, and textual features as described in (Negri et al., 2014).

[2]We found the median criterion to outperform a mean criterion and other alternatives.

Regression Algorithm: We use extremely randomized trees (XTs) (Geurts et al., 2006), following Negri et al. (2014). XTs are a tree ensemble with a high amount of randomization, which has been reported to effectively reduce prediction variance, and can be trained quickly. Negri et al. (2014) report slight improvements for XTs over support vector regression (suggested by Bojar et al. (2015) as a baseline). In addition, our preliminary experiments indicate that XTs have more stable adaptation behavior.

Domain Adaptation: Our regressor adaptation aggregation strategy requires adapting the regressor using the collected manual labels. Our adaptation approach is to add binary indicator features for every document and system, including the document and system currently evaluated. Formally, we perform a projection $\Phi^{i,j} : \mathbb{R}^F \rightarrow \mathbb{R}^{F+D+S}$ from the original F-dimensional feature space into an $(F + D + S)$-dimensional augmented feature space. Here, F is the number of original features, D is the number of training documents plus one for the document under test, and S is the number of training systems plus one for the system under test. Further, i is the index of the particular training- or test-document, j is the index of the particular training- or test-system. The projection is defined as

$$\Phi^{ij}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{e}_i, \mathbf{e}_j \rangle \tag{6}$$

with $\mathbf{e}_i$ being the D-dimensional zero-vector with only the i-th position set to one, and $\mathbf{e}_j$ an analogously defined S-dimensional vector.

We hope that this approach enables decoupled learning of the overall quality of a document or system (via the indicator features), and learning to distinguish quality between segments (via the original features). In general, the indicator features permit the regressor more flexibility to deal with the in-domain training samples, whose number is much smaller than that of the background data. We found the indicators to be helpful also when no adaptation is performed, possibly because it allows a form of multitask learning over the different documents and systems. Therefore, we use indicator features for both aggregation strategies. Our indicator features can be seen as a simplification of Daumé III (2007)'s method: here, in a typical adaptation scenario with in-domain and out-of-domain data, all features are *replicated* for both domains. In our case we deal with a potentially large number of domains (for each document and system), and feature replication would greatly enlarge the feature space and risk overfitting. Hence, we deem the proposed indicator features more appropriate for our purposes.

Predicting Deviation from the Document-Mean: Our active selection strategy only makes use of relative differences between segment scores. Therefore, at training time, we replace the label y_{ji} for the i-th segment of the j-th document (system output) by $y_{ji} - Q_j^{(true)}$, where $Q_j^{(true)}$ is the true overall quality of the j-th document. In this way, the regressor only needs to predict by how far (and in what direction) the segment deviates from the document mean, and the training objective becomes more directly meaningful. The indicator features are particular helpful here as they allow the model to learn and factor out the overall document quality. Note that predicting deviations is not useful for the regressor adaptation strategy, because the automatic scores are no longer meaningful in absolute terms.

4.2 Confidence-based Approach

The second class of automatic segment-level scores are confidence scores obtained directly from the system. Confidence scores are better-informed and potentially better correlated with true segment quality. However, unlike the regressor outputs, confidence scores do not share the same unit as the target evaluation metric. It would be possible to add these as a feature to a regressor as above, but at least for the QE training data used in our experiments, confidence scores were not available for all systems.

Instead, we propose using confidence scores as-is, which frees us from having to train a regressor and collecting training data for it. This approach can only be used with the active selection aggregation strategy. In this paper, we use confidence scores only in the ASR setting, and compute segment confidences as average word-level posterior probabilities derived from lattice- or consensus decoding.

5 Experiments

We conduct experiments to answer the following research questions:

Name	Description	# systems	# documents	# segments
MT.in-domain	WMT 2015 submissions, English to German news task	20	81	43380
MT.out-of-domain	WMT 2014 submissions, English to German medical task (khreshmoi testset)	6	1	6000
ASR.in-domain	IWSLT 2013 submissions, English TED task	13	28	28496
ASR.out-of-domain	English KIT lectures, decoded with a standard in-house ASR system	1	6	1849
ASR.conf-task	Mixed ASR data for which confidence scores were available	4	21	2431

Table 1: Data used in our experiments.

- How much effort is needed to produce quality estimates of a certain accuracy?
- Can we reduce effort, compared to the baselines, using our two proposed strategies?
- How effective are the proposed features, such as indicator features, predicting deviations, and using black box regression or confidences?
- Are there difference between in-domain and out-of-domain settings, or between MT and ASR?

We restrict ourselves to reference-based metrics for simplicity, namely TER for MT, and WER for ASR. Our experiments are simulated in the sense that we have reference translations/transcriptions available, and simulate a human annotator who replicates these exactly. Our main evaluation measure is mean absolute error (MAE)[3] between the predicted and true document-level TER/WER.

We use 5 datasets as indicated in Table 1, with testing data varying over all datasets, but only the ones labeled "in-domain" used for regressor training. Moreover, for each evaluated document the QE regressor was retrained with training data excluding that which corresponded to the same system *or* document currently tested (for in-domain tests). This makes even our in-domain scenario more challenging than some of the previous works on automatic QE (Specia et al., 2015).

We use scikit-learn (Pedregosa et al., 2011) for regressor training. We assign weights to training samples proportional to their segment length, because longer segments are weighted more strongly in our aggregation strategies (§3) and are thus more important to be accurately predicted. We perform random search with 20 iterations to optimize hyper-parameters (namely, the max-depth and min-samples-split parameters of XTs) in terms of mean squared error.[4] Tuning is conducted separately for every test document, using 10-fold cross validation on the respective training data. For regressor adaptation, we experimented with different weights for the adaptation samples, but observed only minor gains and decided to weight all data equally for simplicity.

5.1 Evaluation of Automatic Scores

We first analyze the performance of the fully automatic scores in terms of MAE and Pearson linear correlation coefficients. We investigate both segment-level and document-level performance, the latter being identical to the fully automatic baseline (§3.1). For comparison, we evaluate a mean-predictor baseline that always predicts the training mean, regardless of the input features. This baseline has been found surprisingly strong previously (Negri et al., 2014; Specia et al., 2015), which we confirm in Table 2. On segment-level, gains over the mean-predictor baseline are clearly visible only for the ASR setting. As expected, the out-of-domain tasks appear much more difficult than the in-domain setting. Note that even though the mean baseline sometimes achieves lower MAE, the XT regressor maintains the advantages

[3]Graham (2015) argues that correlation is better for evaluating *sentence-level* QE, because MAE can be improved by transformation to match estimated global mean and variance. However, we find MAE more indicative for our purpose as it measures not only how well systems are compared against one another, but also how well overall quality is judged in absolute terms. Moreover, collecting global statistics for transformation seems problematic when flexibility for domain changes is required.

[4]Tuning directly for MAE yielded similar results.

	Segment level			Document level		
	↓MAE mean	↓MAE XT	↑Pearson XT	↓MAE mean	↓MAE XT	↑Pearson XT
MT.in-domain	**21.0**	21.2	0.13	7.3	**5.8**	0.16
MT.out-of-domain	**14.9**	15.8	0.04	3.4	3.6	0.12
ASR.in-domain	15.4	**14.0**	0.35	9.6	**9.0**	0.29
ASR.out-of-domain	58.2	**52.9**	0.10	42.8	**37.7**	0.23

Table 2: Black box regression accuracy. Lowest MAE is in bold font if statistically significant (p=0.05) according to the bootstrap resampling significance test (Koehn, 2004).

Pearson correlation	Segment-level (all)	Segment-level (within document)	Document-level
Black box regressor	0.23	0.09	0.44
Negative confidence	0.34	0.61	0.12

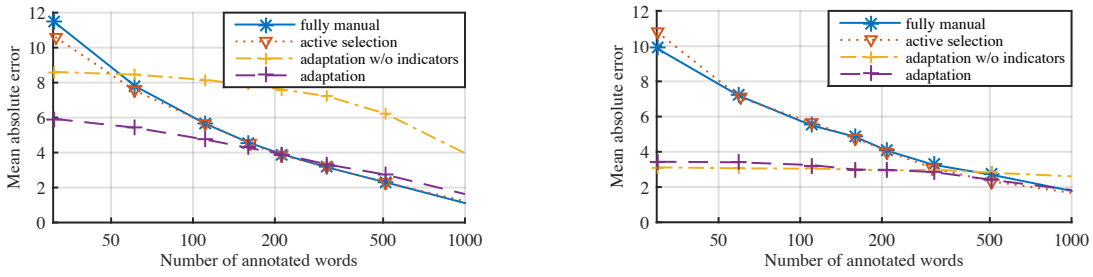
Table 3: Correlation for black box regressor vs. confidence scores with true WER labels on the ASR.conf-task dataset. Confidence scores are strong especially when evaluating the correlation on a per-document basis (averaging over documents), indicating that they may be more suitable for the active selection strategy than the black box approach.

of achieving positive segment correlation and supporting adaptation (§4.1). On document-level, the XT regressor outperforms the baseline in all but the MT.out-of-domain setting in terms of MAE, and correlation is stronger as well.

In Table 3 we also evaluate the performance of ASR confidence scores on the ASR.conf-task dataset. Since regressor outputs and confidences have different units, we compare them in terms of Pearson correlation. Compared to the black box regressor, it can be seen that confidence scores excel especially for the average within-document correlation. We would thus expect confidence scores to outperform black box regression for the active selection strategy.

5.2 MT Setting

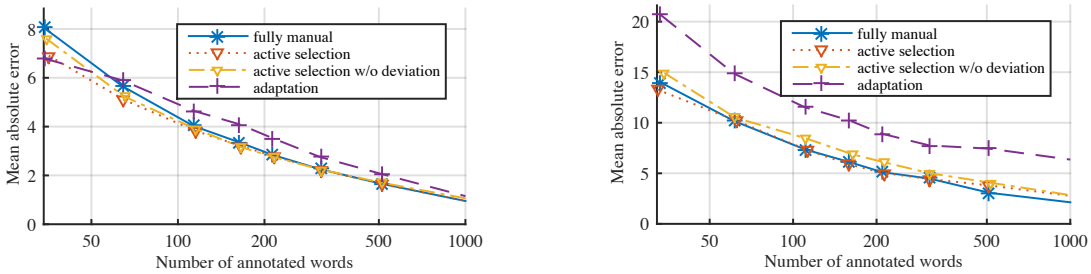
Figure 2 shows the results for the lightly supervised scenario for the MT datasets. Note that the fully automatic baseline corresponds to the leftmost point of the regressor adaptation curve. While the active selection strategy did not clearly outperform the fully manual baseline, regressor adaptation with enabled indicator features performed well. It can be seen that even a moderate amount of annotation improved the estimates, and the advantage over the fully manual baseline was especially strong for the out-of-domain setting. When removing the indicator features, regressor adaptation performs poorly in the in-domain setting. For the out-of-domain setting, performance changes only slightly. In both settings, it seems that the indicator features help to reach similar performance as the fully manual baseline for larger amounts of annotated data, suggesting that they help improve adaptation behavior.



(a) MT.in-domain

(b) MT.out-of-domain

Figure 2: Lightly supervised setting for MT.



(a) ASR.in-domain (b) ASR.out-of-domain

Figure 3: Lightly supervised setting for ASR.

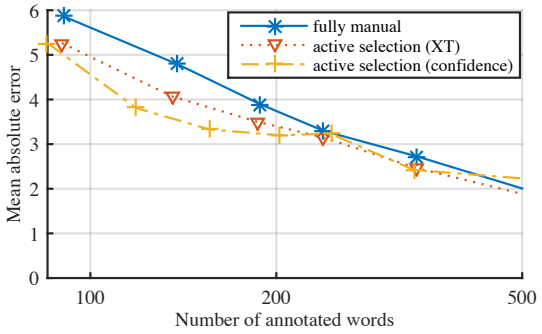


Figure 4: Active selection on the ASR.conf-task dataset, comparing the baseline and selection via XT regression scores and confidence scores.

5.3 ASR Setting

Figure 3 shows the results for the ASR datasets. The regressor adaptation strategy performs rather poorly. For active selection, we evaluate predicting deviations (§4.1), and observe small but consistent gains compared to the unaltered objective function. In result, we slightly outperform the fully manual baseline for the in-domain setting, and perform on par with the baseline for the out-of-domain setting.

However, the observed gains are probably too small to be considered worthwhile. We therefore also investigate replacing the XT regression scores by confidence scores provided directly by the ASR (§4.2). Here, XT regression was trained on data similar as in the test (in-domain setting), but again excluding all data from the particular document and system being tested. For the confidence scores, there is no distinction between in-domain and out-of-domain. Figure 4 reveals the more solid gains over both the fully manual baseline and active selection based on XT regression scores. We conclude that confidence scores are a promising way of reducing labeling effort in our slightly supervised quality estimation framework.

5.4 Discussion

As was seen in the above experiments, the proposed regressor adaptation strategy performed well for MT but not for ASR, whereas for the proposed active selection strategy it was the other way around. We also observe that for the MT datasets, there was relatively high between-segment variance compared to a relatively low between-document variance. This puts the fully manual baseline at a disadvantage, and may be the reason for the good regression-based results (fully automatic and regressor adaptation). In contrast, for the ASR datasets we observed relatively low between-segment variance and high between-document variance, which would put the annotation-based strategies at an advantage (fully manual and active selection). Scarton et al. (2015) made similar observations for MT, and we expect these findings to hold for other datasets. We also confirmed that results are similar when using BLEU+1 (Lin and Och, 2004) as the metric, instead of TER. Whether or not the situation changes when switching to very different metrics, such as non reference-based metrics, is left as a question for future work.

6 Relation to Prior Work

A good overview over the state-of-the-art in automatic QE for MT is given by Bojar et al. (2015) and Bojar et al. (2016), and for ASR by Ogawa et al. (2012) and Negri et al. (2014). Document-level QE was first explored by Soricut and Echihabi (2010) by exploiting document-level features, and was later improved by using sentence-level information (Soricut and Narsale, 2012; Specia et al., 2015; Bojar et al., 2015). Scarton and Specia (2014) and Scarton et al. (2015) argue that document-level quality metrics should consider discourse information that cannot be captured when processing segments individually, and explore features for QE that capture discourse information. Quantitative assessment of discourse information remains challenging (Bojar et al., 2016). Our aggregation approach supports only sentence-level information.

The out-of-domain case investigated in this work, i.e. predicting quality for a previously unknown system or task, has been found challenging in both ASR (Negri et al., 2014) and MT (de Souza et al., 2015a). The WMT 2015 shared task on QE considered only the scenario of training and testing on output produced by the same system (Bojar et al., 2015). The usual way to address domain mismatch when training data for all domains is available is via adaptation/multitask learning (Beck et al., 2014; de Souza et al., 2015b). Our indicator features can be seen as a form of multitask learning. A strategy for the case where no in-domain training data is available is to obtain such training data cheaply via active learning (Beck et al., 2013). This work is probably most similar in spirit to ours in that it attempts reliable quality estimation at low labeling costs.

A different line of research, crowd-sourced annotation, critically depends on quality control, as well. Approaches are usually based on comparing results between several workers (Passonneau and Carpenter, 2014), querying gold standard "testing" labels occasionally (Joglekar and Garcia-Molina, 2013), and/or automatically predicting quality (Roy et al., 2010; Gao et al., 2015). Our work can be seen as a generalization of the latter two, with the gold labels corresponding to our fully manual baseline, the automatic estimation corresponding to our fully automatic baseline, and the *workers* being our *systems*.

7 Conclusion

We proposed lightly supervised quality estimation at the document level, a framework that allows flexible quality estimation across changing domains and quality requirements, while requiring only moderate human annotation effort. We suggested two strategies for combining manual and automatic segment-level scores into a document-level estimate. The first strategy, regressor adaptation, was able to reduce annotation effort considerably for TER-based quality estimation in MT. The second strategy, active selection of segments for annotation, showed promising results for ASR quality estimation in terms of WER, when confidence scores are available.

As future work, we suggest exploring further evaluation metrics, in particular non reference-based metrics. Depending on the metrics, user interfaces that allow manual annotation at low effort should be designed. Also, the confidence-based active selection strategy may be worth investigating for MT.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful suggestions. The project leading to this application has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement n° 645452.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR : An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine translation and/or Summarization*, pages 65–72.

Daniel Beck, Lucia Specia, and Trevor Cohn. 2013. Reducing Annotation Effort for Quality Estimation via Active Learning. In *Association for Computational Linguistics Conference (ACL)*, pages 543–548, Sofia, Bulgaria.

Daniel Beck, Kashif Shah, and Lucia Specia. 2014. SHEF-Lite 2.0: Sparse Multi-task Gaussian Processes for Translation Quality Estimation. In *Association for Computational Linguistics Conference (ACL)*, pages 307–312, Baltimore, USA.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 Workshop on Statistical Machine Translation. In *Workshop on Statistical Machine Translation (WMT)*, pages 1–46, Lisbon, Portugal.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 Conference on Machine Translation. In *Conference on Machine Translation (WMT)*, pages 131–198.

Hal Daumé III. 2007. Frustratingly Easy Domain Adaptation. In *Association for Computational Linguistic (ACL)*, pages 256–263, Prague, Czech Republic.

José G. C. de Souza, Matteo Negri, Elisa Ricci, and Marco Turchi. 2015a. Online Multitask Learning for Machine Translation Quality Estimation. In *Association for Computational Linguistics (ACL)*, pages 219–228, Beijing, China.

José G. C. de Souza, Hamed Zamani, Matteo Negri, Marco Turchi, and Daniele Falavigna. 2015b. Multitask Learning for Adaptive Quality Estimation of Automatically Transcribed Utterances. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies Conference (NAACL-HLT)*, pages 714–724, Denver, USA.

Mingkun Gao, Wei Xu, and Chris Callison-Burch. 2015. Cost Optimization for Crowdsourcing Translation. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies Conference (NAACL-HLT)*, pages 705–713, Denver, USA.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine Learning*, 63(1):3–42.

Yvette Graham. 2015. Improving Evaluation of Machine Translation Quality Estimation. In *Association for Computational Linguistics (ACL)*, pages 1804–1813, Beijing, China.

Kim Harris, Aljoscha Burchardt, Georg Rehm, and Lucia Specia. 2016. Technology Landscape for Quality Evaluation: Combining the Needs of Research and Industry. In *LREC Workshop on Translation Evaluation*, pages 50–54, Portorož, Slovenia.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*. Springer, second edition.

Manas Joglekar and Hector Garcia-Molina. 2013. Evaluating the Crowd with Confidence. In *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 686–694, Chicago, USA.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395, Barcelona, Spain.

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: A method for evaluating automatic evaluation metrics for machine translation. In *International Conference on Computational Linguistics (COLING)*, pages 501–507, Geneva, Switzerland.

Matteo Negri, Marco Turchi, G. C. de Souza, and Daniele Falavigna. 2014. Quality Estimation for Automatic Speech Recognition. In *International Conference on Computational Linguistics (COLING)*, pages 1813–1823, Dublin, Ireland.

Atsunori Ogawa, Takaaki Hori, and Atsushi Nakamura. 2012. Error Type Classification and Word Accuracy Estimation using Alignment Features from Word Confusion Network. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4925–4928, Kyoto, Japan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Association for Computational Linguistic (ACL)*, pages 311–318, Philadephia, Pennsylvania.

Rebecca J. Passonneau and Bob Carpenter. 2014. The Benefits of a Model of Annotation. *Transactions of the Association for Computational Linguistics (TACL)*, 2:311–326.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Brandon C. Roy, Soroush Vosoughi, and Deb Roy. 2010. Automatic Estimation of Transcription Accuracy and Difficulty. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, Makuhari, Japan.

Carolina Scarton and Lucia Specia. 2014. Document-level translation quality estimation: exploring discourse and pseudo-references. In *European Association for Machine Translation (EAMT)*, pages 101 – 108.

Carolina Scarton, Marcos Zampieri, Mihaela Vela, Josef van Genabith, and Lucia Specia. 2015. Searching for Context: a Study on Document-Level Labels for Translation Quality Estimation. In *Conference of the European Association for Machine Translation (EAMT)*, pages 121–128, Antalya, Turkey.

Radu Soricut and Abdessamad Echihabi. 2010. TrustRank: Inducing trust in automatic translations via ranking. In *Association for Computational Linguistic (ACL)*, pages 612–621.

Radu Soricut and Sushant Narsale. 2012. Combining quality prediction and system selection for improved automatic translation output. In *Association for Computational Linguistic (ACL)*, pages 163–170.

Lucia Specia, Kashif Shah, Jose G. C. de Souza, and Trevor Cohn. 2013. QuEst - A Translation Quality Estimation Framework. In *Association for Computational Linguistics: System Demonstrations*, pages 79–84, Sofia, Bulgaria.

Lucia Specia, Gustavo H. Paetzold, and Carolina Scarton. 2015. Multi-level Translation Quality Prediction with QUEST++. In *Association of Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 115–120, Beijing, China.

Improving Translation Selection with Supersenses

Haiqing Tang[1], **Deyi Xiong**[1]*, **Oier Lopez de Lacalle**[2] **and Eneko Agirre**[2]
Soochow University, Suzhou, China[1]
University of the Basque Country, Donostia, Spain[2]
hqtang@stu.suda.edu.cn, dyxiong@suda.edu.cn
{oier.lopezdelacalle, e.agirre}@ehu.eus

Abstract

Selecting appropriate translations for source words with multiple meanings still remains a challenge for statistical machine translation (SMT). One reason for this is that most SMT systems are not good at detecting the proper sense for a polysemic word when it appears in different contexts. In this paper, we adopt a supersense tagging method to annotate source words with coarse-grained ontological concepts. In order to enable the system to choose an appropriate translation for a word or phrase according to the annotated supersense of the word or phrase, we propose two translation models with supersense knowledge: a maximum entropy based model and a supersense embedding model. The effectiveness of our proposed models is validated on a large-scale English-to-Spanish translation task. Results indicate that our method can significantly improve translation quality via correctly conveying the meaning of the source language to the target language.

1 Introduction

Phrase-based SMT has achieved better performance than word-based SMT. One of the reasons is that continuous phrases, rather than single words, are used as translation units so that useful context information can be captured for selecting appropriate translations. Even so, when translating sentences containing ambiguous words, which have multiple meanings, the state-of-the-art phrase-based SMT is still suffering from inaccurate lexical choice which makes translations unable to correctly convey the meaning of source sentences. Recent studies show that in order to improve translation quality, one must correctly identify the most likely senses of source-side ambiguous words when selecting target translation (Gao et al., 2013; Zou et al., 2013; Zhang et al., 2014).

One common approach to deal with ambiguity is to incorporate a word sense disambiguation (WSD) system into SMT system. At first, Carpuat and Wu (2005) attempt to use the senses of source ambiguous words predicted by a standard formulation of WSD directly in a word-based SMT but results are disappointing. They are skeptical of the assumption that WSD systems are useful for SMT. Instead of the standard WSD task, Vickrey et al. (2005) propose a novel formulation of WSD for SMT: directly predicting possible target translation candidates as senses for ambiguous source words. This reformulated WSD has been shown to help SMT by several subsequent studies, including later work by Carpuat and Wu (2007). Following this WSD reformulation for SMT, they integrate the WSD training, where sense definitions are drawn automatically from all phrasal translation candidates rather than from a predefined sense inventory into a phrase-based SMT.

In addition to WSD, topic model (Blei et al., 2003) is yet another technique used to detect most likely senses of source words. Various topic-specific lexicon translation models are proposed to improve translation quality. These models can be classified into two categories: word-level translation models (Zhao and Xing, 2006; Tam et al., 2007) and phrase-level models (Xiao et al., 2012). Especially, Xiong and Zhang (2014) propose a sense-based translation model that integrates hidden word senses into machine

*Corresponding author

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3114–3123, Osaka, Japan, December 11-17 2016.

translation to investigate whether hidden senses are useful for SMT. They resort to word sense induction (WSI) and build a broad-coverage sense tagger that relies on the nonparametric Bayesian model to obtain hidden senses for each source word in large-scale corpora. Different from what the previous reformulated WSD does, they first predict word senses which are automatically learned from data for ambiguous words and then make use of predicted word senses along with other context features to predict possible target translations for these words. They conclude that word senses automatically induced by WSI are very useful for SMT in dealing with inaccurate lexical choice.

However, all the models mentioned above are focusing on investigating how to exploit fine-grained word senses (e.g., predefined senses, target translation candidates, hidden senses) in SMT to improve translation quality. Then how about coarse-grained word senses such as supersenses that are WordNet (Fellbaum, 1998) semantic labels grouping semantically close synsets into a coarse-grained ontology? Are they useful for SMT? Since supersense tagging is the task of assigning high-level ontological classes to open-class words such as nouns, verbs, it is thus a coarse-grained word sense disambiguation task. To the best of our knowledge, we are the first to be dedicated to systematically investigating whether supersenses can be used in SMT to alleviate source word sense translation errors. Specifically, we try to model supersenses for SMT in the following two ways:

- A maximum entropy (MaxEnt) based model: Building multiple MaxEnt classifiers with one classifier per source word type, which incorporate supersenses as features.

- And a supersense embedding model: Projecting word supersenses to a multidimensional vector space with word2vec[1], and inducing source phrase supersense embeddings for phrasal translation.

These two supersenses-based translation models are integrated into a state-of-the-art SMT system and a series of experiments are conducted on English-to-Spanish translation based on large-scale training data. Results show that supersenses, high-level ontological concepts, are capable of improving translation quality and the supersense embedding model outperforms the MaxEnt classifiers-based model.

Our work is different from previous standard WSD, reformulated WSD and WSI with hidden senses for SMT. The first uses fine-grained senses predefined by WordNet. The second explores surrounding words to disambiguate word senses. And the third integrates topics inferred from pseudo documents as hidden senses. We employ coarse-grained predefined categories from WordNet to disambiguate ambiguous words for SMT.

The remainder of this paper is organized as follows. Section 2 introduces related studies exploiting word sense knowledge to improve lexical selection in SMT and various applications of supersenses and word embeddings. Section 3 elaborates how we obtain supersense tags for words in large-scale data. Section 4 describes our supersense-based translation models. Section 5 presents the way that we integrate supersense-based translation models into SMT. Section 6 discusses our experiments and results. In section 7 we summarize our findings and directions for future work.

2 Related Work

The problem of accurate lexical choice is an unsolved challenge for phrase-based SMT. Much work has been done to identify proper senses of source ambiguous words to aid system in choosing appropriate translations. Integrating WSD into an SMT system is typical of this work as described in Section 1 (Carpuat and Wu, 2005; Vickrey et al., 2005; Carpuat and Wu, 2007; Chan et al., 2007). Exploring topic model for SMT is another attempt. Gong et al. (2010) introduce document-level topics to help SMT generate target translations. They use a monolingual LDA model to assign a specific topic to the document to be translated. Similarly, each phrase pair is also assigned with one specific topic. A phrase pair will be filtered from phrase table if its topic mismatches the document topic. Xiao et al. (2012) propose a topic similarity model which incorporates the rule-topic distributions on both the source and target side into a hierarchical phrase-based system for rule selection.

[1] https://code.google.com/archive/p/word2vec/

noun	Tops	act	animal	artifact	attribute
	body	cognition	communication	event	feeling
	food	group	location	motive	object
	person	phenomenon	plant	possession	process
	quantity	relation	shape	state	substance
	time				
verb	body	change	cognition	communication	competition
	consumption	contact	creation	emotion	motion
	perception	possession	social	stative	weather

Table 1: Supersense labels for nouns and verbs in WordNet.

Supersenses are useful and have been used as high-level features in various tasks. Ciaramita and Altun (2006) define a tagset based on WordNet supersenses to perform broad-coverage word sense disambiguation and information extraction which they approach as a unified tagging problem. They achieve considerable improvements over the first sense baseline. Koo and Collins (2005) utilize supersense reranking that provides a partial disambiguation step in syntactic parse to build useful latent semantic features. Other tasks like preposition sense disambiguation (Ye and Baldwin, 2007), noun compound interpretation (Tratz and Hovy, 2010) can also employ supersenses to improve performance.

Word embeddings, also called distributed word representations, are used in many natural language processing areas such as information retrieval (Manning et al., 2008), search query expansions (Jones et al., 2006), or representing semantics of words (Reisinger and Mooney, 2010). As to SMT, Zou et al. (2013) propose a method to learn bilingual word embeddings for recognizing and quantifying semantic similarities across languages.

Our work is to integrate supersenses into a phrase-based SMT. We adopt two ways to train our supersense-based models: one is a MaxEnt classifier-based model which is closely related to Xiong and Zhang's (2014) work, the other is a model built on supersense embeddings that are different from word embeddings in that supersense embeddings can provide more high-level semantic information than word embeddings.

3 Supersense Tagging

Supersenses, first defined by Ciaramita and Johnson (2003), are coarse-grained semantic labels used by lexicographers to facilitate the development of WordNet. There are 45 supersense labels, 26 for nouns, 15 for verbs, 3 for adjectives and 1 for adverbs, used in WordNet to classify synsets into several domains based on syntactic category and semantic coherence. Normally, an ambiguous word belongs to several synsets. Since supersense labels are assigned to synsets, word sense ambiguity can be preserved to a certain degree at this level. In this paper, we focus on noun and verb supersenses. Table 1 shows the corresponding supersense labels in WordNet.

We use supersenses as our semantic classes to tag our training data for obtaining contextual information for the following advantages. First, this set of semantic labels is fairly general and therefore small. The reasonable size of the label set makes it possible to have only one model. In contrast, we have one model per word for fine-grained WSD. Second, the sensible semantic categories are easily recognizable and not too abstract. Since similar words tent to be merged together, these semantic categories seem promising to be used in MT. Third, while individual glossary can not embody too much about the narrow concept it is attached to, at the supersense level this information accumulates. In order to be more intuitive, we take the verb ***"help"*** as an example to show how fine-grained senses are grouped in supersenses, which is presented in Table 2.

In this paper, supersense tagging is carried out with a model based on Ciaramita and Altun's (2006) work. We deploy the implementation provided by Michael Heilman[2]. The model takes a sequence labeling approach to learn a model for supersense tagging. Specifically, we employ a sequential labeller

[2]http://www.ark.cs.cmu.edu/mheilman/questions/SupersenseTagger-10-01-12.tar.gz

Supersense	WN Senses	Gloss
social	1	give help or assistance; be of service
social	6	contribute to the furtherance of
body	2	improve the condition of
stative	3	be of use
stative	4	abstain from doing; always used with a negative
change	8	improve; change for the better
consumption	5	help to some food; help with food or drink
consumption	7	take or use

Table 2: An example of grouping different fine-grained senses of a word into supersenses. WN senses: senses from WordNet.

that is based on a Hidden Markov Model (HMM) trained in a discriminative way. That is, the model can be seen as a perceptron-trained HMM (Collins, 2002) that jointly models observation/label sequences. The model is trained on Semcor Corpus (Miller et al., 1993) following the experimental setting described in Ciaramita and Altun (2006). WordNet fine-grained senses are mapped to their corresponding supersense. In our case, only nouns and verbs are mapped, labeling as "NULL" the rest of the tokens (including adjectives and adverbs). In some cases "noun.Tops" refers to more specific supersenses, such as "food", "person", or "animal". In those cases we substitute the "noun.Tops" with more specific label (e.g "animal" as "noun.animal"). Although the tagger learns 41 semantic categories, we included (B) beginning and (I) continuation as supersense prefixes to learn more categories. Thus, actual label space to be learned increases to 83 (including "NULL").

Semcor is divided in three parts: "brown1" and "brown2", in which nouns, verbs, adjectives and adverbs are annotated. But the section "brownv", contains annotations only for verbs. To avoid many nouns being labeled with "NULL" we take the same procedure as Ciaramita and Altun (2006) to extract the text segment including a verb but not a noun.

Regarding features, our implementation replicates the features used in the original work, which include words and part-of-speech tags occurring in a context-window, word-shapes of the surrounding words, the first-sense of the word and the surrounding words, and the previous label. Please refer to the original work for a more detailed description of the model.

4 Supersense-based Translation Model

In this section, we present two methods to build the proposed supersense-based translation models.

4.1 A MaxEnt Classifier-based Model

Given a source word c with its contextual information including supersenses, we resort to a MaxEnt classifier to estimate the probability $p(e|C(c))$ of a target phrase e. The MaxEnt classifier is formulated as follows.

$$P(e|C(c)) = \frac{exp(\sum_i \theta_i h_i(e, C(c)))}{\sum_{e'} exp(\sum_i \theta_i h_i(e', C(c)))} \quad (1)$$

where h_i are binary features, θ_i are weights of these features.

We use two groups of features: lexicon features and supersense features, which are used to define $C(c)$ as follows:

$$C(c) = \{c_{-k}, s_{c_{-k}}, ..., c_{-1}, s_{c_{-1}}, c, s_c, c_1, s_{c_1}, ..., c_k, s_{c_k}\} \quad (2)$$

where c represents words and s for supersenses. In this way, not only centered word c and its supersenses s_c are included, but the preceding and succeeding k words with their corresponding supersenses are also involved. Particularly c_{-k} is the kth preceding word of c and $s_{c_{-k}}$ is the supersense of word c_{-k}. We extract all training events defined by $C(c)$ for each source word c and train multiple MaxEnt classifiers with one classifier per source word.

4.2 A Supersense Embedding Model

A traditional way to generate a phrase table is to use the training component of Moses[3] that allows you to automatically train translation models for any language pairs. Normally, each entry in the phrase table contains a source phrase, a target phrase, their word alignments, and five types of translation scores. From the training corpora where the source side is annotated with supersenses, we want to learn the distribution of the supersenses tagged for a source-side phrase. In order to achieve this goal, we have made some changes on the phrase extraction and scoring module of Moses training system to make them output source word supersenses. The number of extracted phrase pairs are expanded greatly since the same source phrase may correspond to several sequences of tagged supersenses.

With word2vec, we can train word sense embeddings on the supersense-tagged corpus. Assuming that a source phrase src containing n words $(w_1, w_2, ..., w_n)$ has k supersense sequences: $ps_1, ps_2, ..., ps_k$. Since each word supersense in the phrase constitutes the phrase supersense sequence, we can use Eq. (3) to represent ps_i, and Eq. (4) to represent the supersense sequence embedding.

$$ps_i = (w_1|ws_1\ w_2|ws_2\ ...\ w_n|ws_n) \tag{3}$$

$$\overrightarrow{ps_i} = \overrightarrow{w_1|ws_1} + \overrightarrow{w_2|ws_2} + ... + \overrightarrow{w_n|ws_n} \tag{4}$$

where ws represents supersense labels and | is a separator between the word and its supersense label.

Each supersense sequence has a unique sense embedding according to the calculating method above. Then how can we represent the supersense embeding of a source phrase in the phrase table? We adopt a dividing and merging method.

The dividing method A translation rule in the original phrase table is divided into several rules for the reason that the source phrase of the rule has several supersense sequences. Direct and indirect phrase translation probability calculated in Eq. (5) and (6) are reformulated and recalculated according to Eq. (7) and (8) respectively.

$$P(e|f) = \frac{Count(e, f)}{Count(f)} \tag{5}$$

$$P(f|e) = \frac{Count(f, e)}{Count(e)} \tag{6}$$

$$P(e|f, ps) = \frac{Count(e, f, ps)}{Count(f, ps)} \tag{7}$$

$$P(f, ps|e) = \frac{Count(f, ps, e)}{Count(e)} \tag{8}$$

where f, e, ps stands for source phrase, target phrase, source phrase supersense sequence respectively. In this way, a source phrase may have multiple supersense embeddings.

The merging method Supposing a source phrase src has k supersense sequences: $ps_1, ps_2, ..., ps_k$, we study the probability distribution of each supersense sequence according to the following formula.

$$P(ps_i|src) = \frac{Count(ps_i, src)}{Count(src)} \tag{9}$$

We assign each source phrase a unique sense embedding $\overrightarrow{s_{src}}$ using the following formula (10).

$$\overrightarrow{s_{src}} = \lambda_1\overrightarrow{ps_1} + \lambda_2\overrightarrow{ps_2} + ... + \lambda_k\overrightarrow{ps_k} \tag{10}$$

where λ_i represents the probability of the ith supersense sequence calculated according to Eq. (9).

[3] http://www.statmt.org/moses/

5 Decoding

We integrate the proposed supersense-based translation models described above into a log-linear translation framework of SMT as a new knowledge source to disambiguate source words. Both supersense-based translation models require that each sentence should be sense-tagged to annotate each word with supersenses before being translated. We adopt an integration strategy similar to that introduced by Xiong and Zhang (2014) to incorporate the MaxEnt classifier-based model into our SMT system. During decoding, once a new source word c is translated, we find its target phrase e according to word alignments that are kept in the phrase table. Then we compute the translation probability $p(e|C(c))$ via the equation (1) using the corresponding classifier. As to integrating the supersense embedding model, we load the pre-trained word sense embeddings to calculate the source phrase sense embeddings according to Eq. (4) when translating a sentence. We compute the similarity between a source phrase in a source sentence and the corresponding matched phrase from the phrase table using the following formula in order to select appropriate translation rules.

$$Sim(\overrightarrow{s_{ssrc}}, \overrightarrow{s_{tsrc}}) = \frac{\overrightarrow{s_{ssrc}} \bullet \overrightarrow{s_{tsrc}}}{||\overrightarrow{s_{ssrc}}|| \times ||\overrightarrow{s_{tsrc}}||} = \frac{\sum_i (a_i \times b_i)}{\sqrt{\sum_i {a_i}^2 \times \sum_i {b_i}^2}} \tag{11}$$

where s_{ssrc} is a phrase in a source sentence, s_{tsrc} is the same phrase in the phrase table, a_i and b_i are the value of ith dimension of their sense embeddings.

Given a source sentence $\{c_i\}_1^N$, We define the score of supersense embedding model as follows.

$$Score_{M_s} = \sum_{ssrc_i \in \Gamma} Sim(\overrightarrow{s_{ssrc_i}}, \overrightarrow{s_{tsrc_i}}) \tag{12}$$

where Γ is a set of source phrases which have translation rules in the phrase table.

6 Experiments

In this section, we conducted a series of experiments on English-to-Spanish translation using massive training data. With the trained supersense-based translation model, we would like to investigate the following two questions:

- Whether coarse-grained supersenses can improve translation quality.
- Whether supersense embeddings can play a role in lexical selection.

6.1 Setup

Our baseline is a state-of-the-art SMT system which adapts Bracketing Transduction Grammars (Wu, 1997) to phrasal translation and augment itself with a maximum entropy based reordering model (Xiong et al., 2006). Our training corpora are English-Spanish sentences from the Europarl parallel corpus (Koehn, 2005) consisting of 1.9M sentence pairs with 51M English words and 54M Spanish words. We ran GIZA++ on the training data in both directions and then applied the "grow-diag-final" refinement rule (Koehn et al., 2003) to obtain final word alignments. Our phrase table was generated according to the word-aligned data. As to the language model, we trained a separate 5-gram LM using the SRILM toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing (Chen and Goodman, 1996) on each subcorpus[4] and then interpolated them according to the corpus used for tuning.

We trained our MaxEnt classifiers with the off-the-shelf MaxEnt tool.[5] We performed 100 iterations of the L-BFGS algorithm implemented in the training toolkit on the collected training events from the sense-annotated data. We set the Gaussian prior to 1 to avoid overfitting.

[4]There are 12 subcorpora: commoncrawl, europarl, kde4, news2007, news2008, news2009, news2010, news2011, news2012, newscommentary, openoffice, un

[5]http://homepages.inf.ed.ac.uk/lzhang10/maxenttoolkit.html

System	batch2	batch2a	batch2q
Base	37.86	36.06	42.59
Max_ss	38.25**	36.33	43.10*
Max_hs	38.28**	36.47**	42.94

Table 3: Results of MaxEnt-based sense models with supersenses (Max_ss) vs. hidden senses (Max_hs) against the baseline. **/*: significantly better than the baseline at $p < 0.01$ and $p < 0.05$ respectively.

	System	batch2	batch2a	batch2q
	Base	37.86	36.06	42.59
Dividing	SSTM(100)	38.51**	36.55*	43.22**
	SSTM(200)	38.42**	36.32	43.23**
Merging	SSTM(100)	38.05	36.17	42.93
	SSTM(200)	38.64**	36.68**	43.17*

Table 4: Results of using the dividing and merging method to train supersense embedding-based translation model (SSTM) with vector dimensionality varying from 100 to 200. **/*: significantly better than the baseline at $p < 0.01$ and $p < 0.05$ respectively.

The method used to learn supersense embeddings, word2vec, in this paper was implemented based on continuous bag-of-words model (Mikolov et al., 2013). We only varied vector dimensionality from 100 to 200 and set the value of threshold for occurrence of words to 0.00001. Default values of other parameters such as the training algorithm and the size of the window were all taken.

The QTLeap corpus[6] was divided into two parts equally to be used as our development set *batch1* and test set *batch2*. The corpus was composed by 4000 question and answer pairs in the domain of computer and IT troubleshooting for both hardware and software. This material was collected using a support service via chat, this implies that the corpus is composed by naturally occurring utterances produced by users while interacting with a service. Only interactions composed by one question and the respective answer were included in the corpus. We also divided our test set *batch2* into two parts equally ***batch2a*** and ***batch2q*** respectively. In other words, we used three test sets to verify the effectiveness of our proposed models. We adopted the case-insensitive BLEU-4 (Papineni et al., 2002) as evaluation metric and ran MERT (Och, 2003) three times to alleviate the instability. We reported average BLEU scores over the three runs as final results.

6.2 Results

Our first group of experiments are designed to investigate whether supersenses can be modeled like hidden senses using a MaxEnt classifier. We use the same experiment settings as Xiong and Zhang (2014) did. Especially, we also find that 10-word window is the most suitable window for extracting semantic information according to experiments. Table 3 shows the experimental results for the two SMT systems equipped with multiple MaxEnt classifiers trained on supersenses and hidden senses respectively.

We can easily find that resorting to a MaxEnt classifier, supersenses can also be integrated into the SMT system and achieve an improvement over the baseline, which is comparable to that obtained by hidden senses.

Inspired by the idea that distribution word representations can convey contextual information, we conduct our second group of experiments to investigate whether distributed supersense representations can be used to improve SMT. We are also concerned about the potential impact of the sense embedding dimensionality on the performance of the supersense embedding model. Hence, in addition to the different methods to represent source phrase supersense embeddings, we consider the dimension as 100 and 200 when training word supersense embeddings. Experimental results are listed in Table 4. From the table, we can observe that

[6]http://metashare.metanet4u.eu/go2/qtleapcorpus

	System	batch2	batch2a	batch2q
	Base	37.86	36.06	42.59
Dividing	SSTM	38.51**	36.55*	43.22**
	HSTM	38.20*	36.25	42.85
Merging	SSTM	38.64**	36.68**	43.17*
	HSTM	38.15*	36.29	42.95

Table 5: Results of using the dividing and merging method to obtain supersense embedding-based translation model (SSTM) vs. hidden sense embedding-based translation model (HSTM). **/*: significantly better than the baseline at $p < 0.01$ and $p < 0.05$ respectively.

- No matter which method we use to calculate source phrase supersense embeddings, the supersense embedding-based translation model is able to achieve an average of 0.7 BLEU points over the baseline on three test sets.
- When using the dividing method to calculate source phrase sense embeddings, the sense embedding dimension has a slight impact on the translation quality in terms of BLEU.
- On the contrary, when using the merging method to obtain source phrase supersense embeddings, training supersense embeddings with 100-dimension performs worse than 200-dimension on the test set. The BLEU score drops by 0.6 points on average. This may be because the merging method uses multiple supersense sequences to compute the final supersense embeddings (see Eq. (10)). The fluctuations caused by the dimensionality of embeddings may be amplified by the summation in Eq. (10).

Our final group of experiments is to study 1) whether hidden senses can be integrated into the SMT system in the way similar to supersense embeddings and 2) which kind of word senses can perform better. When using the dividing method to obtain source phrase hidden sense embeddings, we set the dimension value to 100 in which case supersense embeddings perform a little better than 200-dimension according to Table 4. As for the merging method, we set the dimension value to 200 to make an equitable comparison with supersenses. Table 5 shows the results. We find that supersense embedding-based model performs better than hidden sense embedding-based model in all cases. The reason for this may be that hidden senses have already encoded distributional information in themselves.

7 Conclusion

We have exploited coarse-grained supersenses which are semantic labels defined by WordNet to conduct high-level word sense disambiguation for SMT. After each source word in the training data is tagged with a supersense, we take two strategies to train our supersense-based translation models. One is utilizing a maximum entropy classifier to predict the target translation for a source word given its surrounding words and their corresponding supersenses. The other is taking advantage of a word2vec tool to learn supersense embeddings on corpus annotated with supersenses and then calculating the semantic similarity between phrases in source sentence and matched phrases from phrase table. The supersense-based translation model is integrated into a phrase-based SMT system.

We have conducted a series of experiments to validate the effectiveness of the proposed supersense-based translation models. Final experimental results show us that

- The supersense-based translation model is capable of improving translation quality significantly in terms of BLEU.
- When using a MaxEnt classifier to predict target translation for a source word given its surrounding semantic information, both supersenses and hidden senses perform well.
- When using distributed sense representations to build sense-based translation models, supersense embeddings perform better than hidden sense embeddings.

In the future, we would like to investigate new neural models to learn embeddings for a single word supersense as well as a supersense sequence. We are also interested in incorporating the relations of supersenses (i.e., high-level ontological concepts) for machine translation.

Acknowledgements

The authors were supported by National Natural Science Foundation of China (Grant Nos. 61403269, 61432013, and 61525205) and Natural Science Foundation of Jiangsu Province (Grant No. BK20140355). In addition, this work was partially funded by MINECO (TUNER, TIN2015-65308-C5-1-R) and the European Commission (QTLeap, FP7-ICT-2013.4.1-610516). We also thank the anonymous reviewers for their insightful comments.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Marine Carpuat and Dekai Wu. 2005. Word sense disambiguation vs. statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 387–394. Association for Computational Linguistics.

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 61–72.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 33–40.

Stanley F Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602. Association for Computational Linguistics.

Massimiliano Ciaramita and Mark Johnson. 2003. Supersense tagging of unknown nouns in wordnet. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 168–175. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2013. Learning semantic representations for the phrase translation model. *Computer Science*.

Zhengxian Gong, Yu Zhang, and Guodong Zhou. 2010. Statistical machine translation based on lda. In *Universal Communication Symposium (IUCS), 2010 4th International*, pages 286–290. IEEE.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th International Conference on World Wide Web*, pages 387–396. ACM.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*, volume 5, pages 79–86.

Terry Koo and Michael Collins. 2005. Hidden-variable models for discriminative reranking. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 507–514. Association for Computational Linguistics.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*, volume 43. Cambridge university press Cambridge.

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

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *The Workshop on Human Language Technology*, pages 303–308.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *InterSpeech*, volume 2002, page 2002.

Yik Cheung Tam, Ian Lane, and Tanja Schultz. 2007. Bilingual lsa-based adaptation for statistical machine translation. *Machine Translation*, 21(4):187–207.

Stephen Tratz and Eduard Hovy. 2010. A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 678–687. Association for Computational Linguistics.

David Vickrey, Luke Biewald, Marc Teyssier, and Daphne Koller. 2005. Word-sense disambiguation for machine translation. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 771–778. Association for Computational Linguistics.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Xinyan Xiao, Deyi Xiong, Min Zhang, Qun Liu, and Shouxun Lin. 2012. A topic similarity model for hierarchical phrase-based translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 750–758. Association for Computational Linguistics.

Deyi Xiong and Min Zhang. 2014. A sense-based translation model for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 521–528. Association for Computational Linguistics.

Patrick Ye and Timothy Baldwin. 2007. Melb-yb: Preposition sense disambiguation using rich semantic features. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 241–244. Association for Computational Linguistics.

Min Zhang, Xinyan Xiao, Deyi Xiong, and Qun Liu. 2014. Topic-based dissimilarity and sensitivity models for translation rule selection. *Journal of Artificial Intelligence Research*, 50(1):1–30.

Bing Zhao and Eric P Xing. 2006. Bitam: Bilingual topic admixture models for word alignment. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, pages 969–976. Association for Computational Linguistics.

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398.

Is all that Glitters in Machine Translation Quality Estimation really Gold?

Yvette Graham[*] **Timothy Baldwin**[†] **Meghan Dowling**[*] **Maria Eskevich**[‡]
Teresa Lynn[*] **Lamia Tounsi**[*]

[*]ADAPT Centre
Dublin City University
firstname.surname@dcu.ie

[†]Computing and Info Systems
University of Melbourne
tb@ldwin.net

[‡]Centre for Language Studies
Radboud University
m.eskevich@let.ru.nl

Abstract

Human-targeted metrics provide a compromise between human evaluation of machine translation, where high inter-annotator agreement is difficult to achieve, and fully automatic metrics, such as BLEU or TER, that lack the validity of human assessment. Human-targeted translation edit rate (HTER) is by far the most widely employed human-targeted metric in machine translation, commonly employed, for example, as a gold standard in evaluation of quality estimation. Original experiments justifying the design of HTER, as opposed to other possible formulations, were limited to a small sample of translations and a single language pair, however, and this motivates our re-evaluation of a range of human-targeted metrics on a substantially larger scale. Results show significantly stronger correlation with human judgment for HBLEU over HTER for two of the nine language pairs we include and no significant difference between correlations achieved by HTER and HBLEU for the remaining language pairs. Finally, we evaluate a range of quality estimation systems employing HTER and direct assessment (DA) of translation adequacy as gold labels, resulting in a divergence in system rankings, and propose employment of DA for future quality estimation evaluations.

1 Introduction

Although human evaluation of translation quality in theory provides the most meaningful assessment of machine translation (MT), achieving high levels of agreement between human assessors has proven challenging. For example, the annual Workshop on Statistical Machine Translation (WMT) provides large-scale human evaluation of systems, and reports inter-annotator agreement ranging from 0.260 (WMT-13) to 0.405 (WMT-15), with intra-annotator agreement not faring much better, from 0.407 (WMT-12) to 0.595 (WMT-15) (Callison-Burch et al., 2012; Bojar et al., 2013; Bojar et al., 2015).[1] Low agreement levels in human annotation cause challenges for tasks that require evaluation of MT output on the segment-level. For example, evaluation of MT quality estimation (QE) requires the comparison of system predictions of translation quality with segment-level human assessment. Automatic metrics, such as BLEU or TER, although fully repeatable, unfortunately lack the validity of human assessment and are well-known to suffer from bias in favour of translations that happen to be superficially similar to reference translations.

Part-automatic human-targeted metrics, however, are commonly employed as a substitute for human assessment on the segment-level, as they appear to provide a happy medium between fully automatic metrics, that lack the validity of human assessment, and human assessment that lacks reliability/reproducibility of automatic metrics. Human-targeted reference translations, each manually created by minimally post-editing the individual MT output translation to be assessed, remove the bias usually introduced by comparison with a generic reference. Subsequently an automatic component is applied to quantify the error now present between the MT output and its human-targeted reference.

[1]Agreement levels provided are average Kappa coefficients for all language pairs included in the main translation shared task.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3124–3134, Osaka, Japan, December 11-17 2016.

Although human-targeted metrics are indisputably more valid than their generic-reference counterparts, the scores they produce are nonetheless still partly automatic. Given the vast number of possible methods of comparing a given MT output with its (human-targeted) reference translation, it is necessary to provide evidence that any given choice of human-targeted metric provides the best formulaic substitute for human assessment.

As with fully automatic metrics, human-targeted metrics are themselves evaluated by strength of correlation with human assessment. For example, when the most widely applied human-targeted metric, human-targeted translation edit rate (HTER) was first proposed, Snover et al. (2006) reported that "HTER is more highly correlated to human judgments than BLEU or HBLEU" (p. 230), and hence, since 2006, it is HTER, as opposed to HBLEU, that is employed in MT as a human assessment substitute.

2 Relevant Work

HTER is generally regarded as a valid substitute for human assessment. In MT QE, for example, HTER scores are used to evaluate systems in large-scale shared tasks (Bojar et al., 2015; Graham, 2015). If a metric such as HTER is to be relied upon as a substitute for human assessment, it is important that trust in the metric is well-placed to avoid inaccuracies in empirical evaluations. For instance, Bojar et al. (2013) and Graham (2015) make the assumption that HTER provides a valid representation of translation quality and subsequently employ HTER scores as a gold standard representation when evaluating QE systems, ultimately leading to rankings for competing systems. If HTER scores do not in fact provide a valid representation of translation quality, however, system rankings are likely to be incorrect. On review of experiments that originally led to trust in HTER as a substitute for human assessment, a possible disparity emerges between the degree of trust placed in HTER and the limitations of original experiments. These limitations include:

- A sample of translations produced by two distinct MT systems, one of known *low-performance* and one known *high-performance* system;
- A single language pair (Arabic to English);
- A maximum of two human annotators per translation;
- Employment of human-targeted reference translations created by human post-editors coached to specifically minimize HTER scores as opposed to other possible formulations.

The degree to which general conclusions can be drawn from experiments should reflect the extent to which experiments were carried out, and the broad conclusions drawn that HTER has a strong correlation with human assessment is not ideal considering experiments were limited to a small number of translations produced by two MT systems and for a single language pair. Furthermore, translations sampled from systems that operate at the two extremes of performance, a known high quality system and a poorly performing one, risks exaggeration of correlation with human assessment.

In addition, it is possible that issues in relation to levels of inter-annotator agreement, common in other human evaluations, were also present in the human evaluation used in HTER experiments, as Snover et al. (2006) report that "... the four-reference variants of TER and HTER correlate with human judgments as well as – or better than – a second human judgment does" (p. 223). The fact that HTER scores correlated with the quality assessments of one human assessor more than those of another, highlights the likelihood that agreement between human annotators was low, although no precise agreement levels are reported. If, for example, the correlation of scores produced by a (part-automatic) metric, such as HTER, and human assessment scores provided by a given human assessor, Annotator$_a$, is stronger than the correlation between Annotator$_a$ and a second human annotator, Annotator$_b$, it follows that this latter correlation, between human assessments of the two distinct annotators cannot be all that high. Choosing to trust the annotations of Annotator$_a$ simply on the basis that those annotations yield a more favorable correlation with HTER is not justified. The fact that part-automatic metric scores correlate better with one human annotator than another is not evidence of how well the metric is performing, merely that the human evaluation employed is somewhat unstable.

	into English					out of English		
	es-en	de-en	es-en	fr-en	ru-en	en-de	en-es	en-ru
All Crowd-sourced Assessments	2,700	7,100	60,400	4,400	2,700	10,300	10,000	8,400
Post Quality Control	1,800	2,300	33,200	1,700	1,600	5,800	7,600	5,400
Distinct Translations	70	70	500	70	70	140	140	140

Table 1: Initial human assessment of nine language pairs, sample taken from all WMT-13 translation task participating systems.

Finally, the comparison made between the correlation of HTER (with human assessment) and those of other possible formulations, such as HBLEU, was unfortunately biased in favour of HTER, as Snover et al. (2006) report "Annotators were coached on how to minimize the edit rate. The coaching given to the annotators in order to minimize HTER, consisted mostly of teaching them which edits were considered by TER. The annotators also consulted with each other during training to compare various techniques they found to minimize HTER scores" (p. 227). A clear bias was therefore introduced into the comparison of correlations achieved by HTER and other human-targeted metrics, such as HBLEU for example, by coaching post-editors specifically to reduce HTER scores. Although bias in favour of HTER was identified in Snover et al. (2006), reporting "It is possible that performance of HTER is biased, as the targeted references were designed to minimize TER scores rather than BLEU scores" (p. 228), this was not highlighted or reconsidered when reporting correlations and drawing conclusions about the superiority of HTER over other metrics.

In this paper, we repeat the original experiments to re-assess HTER as a human assessment substitute. In doing so, we vastly expand experimental settings relative to the limitations of the original, as follows:

- Data sets include translations sampled from the output of 131 MT systems, operating at all levels of performance;
- Correlations of HTER with human assessment are reported for nine language pairs;
- Evaluation is based on between 15 and 80 human assessments per translation – to overcome human annotator agreement issues;
- Human-targeted reference translations are created without any coaching.

3 Re-evaluating Human-Targeted Metrics

As in evaluation of fully automatic metrics, evaluation of human-targeted metrics is by correlation with human assessment. Like the original evaluation of HTER, our re-evaluation also operates on the segment-level, with the performance of different possible human-targeted metric formulations measured by correlation of metric scores with segment-level human assessment. To compute correlations for human-targeted metrics, two distinct human annotations of each translation are required.

Firstly, we require a human assessment rating of the quality of each translation in order to compute correlations of each human-targeted metric. Our evaluation additionally requires manually created post-editions of each original MT output translation. This human post-edit is employed as the human-targeted reference translation for computing human-targeted metric scores for translations. We firstly describe the methodology employed for human assessment, before providing post-editing details.

3.1 Human Assessment of Translation Adequacy

Overall in this work, human assessment was carried out for two distinct data sets, firstly a nine language pair data set to re-evaluate HTER and subsequently an English to Spanish data set (see Section 4.2). Below we describe human assessment of the nine language pair data set.

Human direct assessment (DA) of the adequacy of translations was collected by means of a 100-point continuous rating scale via crowd-sourcing. Large numbers of repeat human judgments for translations were collected on Amazon Mechanical Turk, by way of re-implementation and minor adaptation of

Graham et al. (2015).[2] Translations were sampled at random from all systems competing in WMT-13 translation task for Czech-to-English (CS-EN), German-to-English (DE-EN), Spanish-to-English (ES-EN), French-to-English (FR-EN), Russian-to-English (RU-EN), English-to-German (EN-DE), English-to-Spanish (EN-ES), English-to-French (EN-FR) and English-to-Russian (EN-RU). Human assessors rated the adequacy of translations in a monolingual human evaluation by comparing the meaning of the original generic human reference translation (rendered in gray) with a given MT output translation (rendered in black) by stating the degree to which they agree that:[3]

The text in black adequately expresses the meaning of the text in gray in English.

Quality control was applied by comparison of ratings provided by each Mechanical Turk worker for pairs of original and degraded translations hidden within HITs, where each pair contained an original MT output translation along with a degraded version of that translation. Ratings of workers who were not reliable (or at any point during the course of data collection became unreliable) in their ability to accurately distinguish between the adequacy of degraded and original system output translations were omitted from the final data set.

Between 15 and 80 (22.4 on average) human assessments of adequacy were collected per translation and combined into a mean score for a given translation, after standardization of scores based on the individual annotators mean and standard deviation rating. Table 1 shows numbers of human assessments collected contributing to mean adequacy scores for translations.

3.2 Human-Targeted Reference Translations

A human-targeted reference translation was created for every translation in the data set by a known-reliable volunteer post-editor native in the target language of the MT output in question. All human post-editors were unaware of the purpose of the post-editing work and experiments. Annotators were shown the reference and MT output with post-editing instructions as follows:[4]

**Making as few changes as possible **, correct the hypothesis segment to make it

(a) have the same meaning as the reference segment;

(b) grammatically correct.

3.3 HTER Re-evaluation Results

Table 2 shows correlations achieved by a range of human-targeted metric formulations with human assessment, including human-targeted versions of segment-level BLEU (Papineni et al., 2002; Koehn et al., 2007), TER (Snover et al., 2006), WER, PER and CDER (Leusch and Ney, 2008). Correlations with human assessment achieved by HTER are, for three of the nine language pairs we evaluate, below a Pearson correlation of 0.6. In addition, comparison of correlations achieved by HTER and HBLEU reveal a *higher* correlation achieved by HBLEU for five of the nine language pairs we evaluate.

As recommended by Graham et al. (2015), we test for significance of difference in dependent correlations using Williams test, as shown in Figures 1 and 2. For two language pairs, DE-EN and FR-EN, correlations achieved by HBLEU with human assessment are significantly stronger than those achieved by HTER, and for all other language pairs, the difference in correlation achieved by HBLEU and HTER is not statistically significant.

Table 2 also includes correlations achieved by all metrics with human assessment when the original generic reference is employed. As expected correlations of generic-reference metrics are in all cases substantially lower than that of their human-targeted counterparts.

3.4 Possibility of Reference Bias in Monolingual MT Assessment

As mentioned in Section 1, in automatic evaluation of MT reference bias is a known problem, as high quality MT output can be unfairly penalized simply due to a lack of superficial similarity with the generic

[2]Complete implementation is made available at `https://github.com/ygraham/direct-assessment`

[3]All instructions were translated into the appropriate target language by a known-reliable native speaker.

[4]Due to space limitations, complete annotation instructions are provided at `https://github.com/ygraham/direct-assessment/post-editing-guidelines`

	CS-EN	DE-EN	ES-EN	FR-EN	RU-EN	EN-DE	EN-ES	EN-FR	EN-RU
H-TER	0.607	0.779	0.669	0.674	0.589	0.543	0.732	0.654	0.550
H-BLEU	0.612	**0.845**	0.664	**0.740**	0.588	**0.582**	0.710	0.637	0.573
H-CDER	0.603	0.828	**0.677**	0.635	**0.632**	0.579	0.718	**0.655**	**0.579**
H-WER	0.560	0.796	0.660	0.668	0.593	0.556	**0.733**	0.609	0.557
H-PER	**0.670**	0.757	0.650	0.651	0.552	0.467	0.679	0.619	0.543
TER	0.230	0.480	0.389	0.508	0.041	0.247	0.315	0.271	0.421
BLEU	0.153	0.429	**0.433**	0.389	0.040	**0.475**	**0.411**	**0.372**	**0.547**
CDER	**0.247**	**0.530**	0.426	0.409	**0.187**	0.353	0.363	0.238	0.401
PER	0.192	0.479	0.351	**0.553**	0.013	0.174	0.271	0.213	0.419
WER	0.198	0.489	0.382	0.425	0.065	0.273	0.325	0.216	0.384

Table 2: Correlation of segment-level human-targeted metric scores with human assessment and correlations of raw metrics with human assessment for a random sample of translations from WMT-13 translation task system submissions

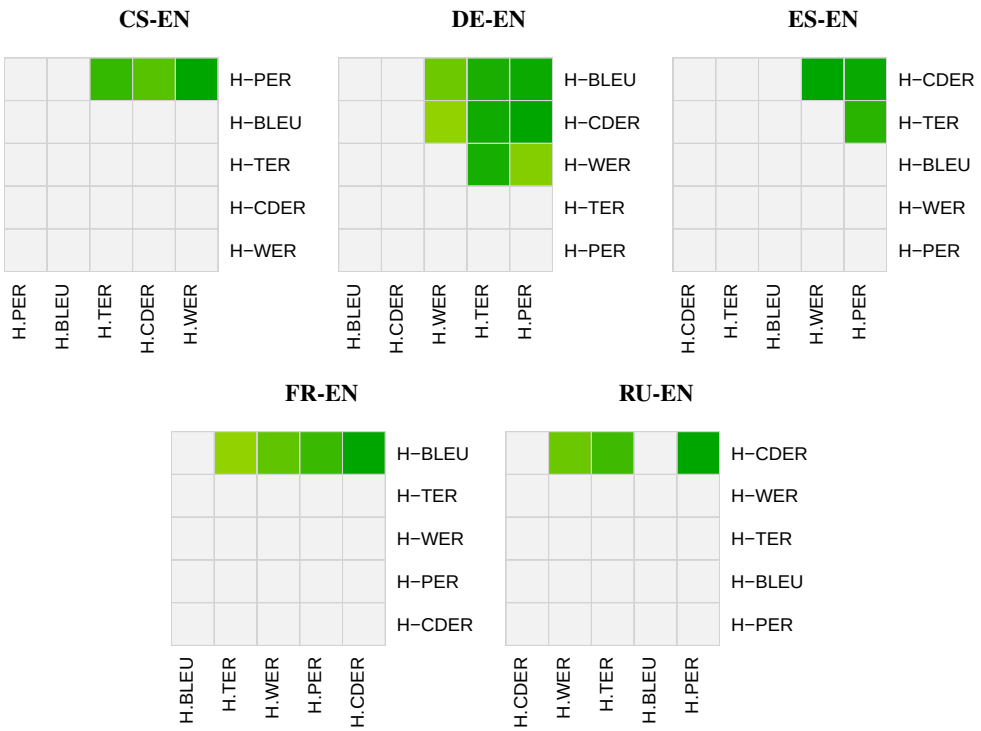


Figure 1: Significance test results (Williams test) of statistical significance of a difference in dependent correlations with human assessment for competing to-English human-targeted metrics; a green cell denotes a significant increase in correlation with human assessment for the metric in a given row over the metric in a given column at $p < 0.05$.

reference translation. Although certainly not to the same extreme, human assessment of MT that employs a reference translation could incur similar reference bias. Graham et al. (2013) provide discussion on how reference bias could exist for direct assessment of translation adequacy, and, as a solution, propose inclusion of a separate reference-free fluency assessment, to provide a component of the evaluation that cannot be biased in any way by a reference translation.

However, it is inevitably the case that resources available to conduct human evaluation are limited, and therefore a trade-off exists between adding fluency assessments of translations and numbers of translations we can feasibly include. For example, in our earlier human evaluation of HTER, it would have been possible to allocate resources to assessment of the fluency of translations, in addition to adequacy, but this would have come at the cost of reducing the size of the test set by approximately one half. Therefore,

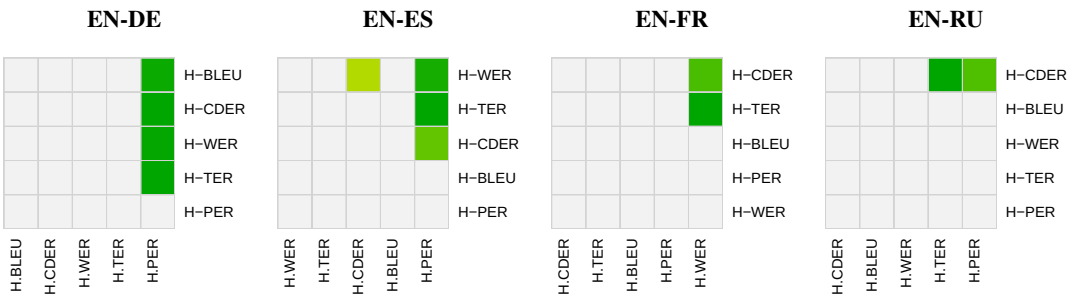


Figure 2: Significance test results (Williams test) of statistical significance of a difference in dependent correlations with human assessment for competing out-of-English human-targeted metrics; a green cell denotes a significant increase in correlation with human assessment for the metric in a given row over the metric in a given column at $p < 0.05$.

the decision to include fluency should be weighed against the degree to which reference bias is actually present in the adequacy evaluation. If there is strong reference bias, for example, it would be highly advisable to include both fluency and adequacy but for a smaller number of translations.

Fomicheva and Specia (2016) investigate bias in monolingual evaluation of MT and conclude reference bias to be a serious issue, with human annotators strongly biased by the reference translation provided. Their conclusion was based on estimation of confidence intervals for Kappa coefficients by means of an unconventional resampling technique, where samples used to estimate confidence intervals were smaller than the original sample size, drawn without replacement, and averaged; this diverges from standard methods of confidence interval estimation, such as bootstrap resampling. As a result, the reported confidence limits are unreliable, bringing their conclusions into question. We therefore consider it necessary to investigate the effect of reference bias with respect to the human evaluation we employ, to assess the likelihood of our own results with respect to the evaluation of HTER being contaminated by strong reference bias.[5]

Reference bias is the attribution of unfairly low scores to translations simply because they are not superficially similar to generic reference translations. In our evaluation of HTER, we employ DA adequacy scores by human assessment, where the MT output is compared to a generic reference translation. We will refer to DA scores collected in this way as $\text{DA}_{\text{gen-ref}}$. It is these gold standard $\text{DA}_{\text{gen-ref}}$ scores that potentially run the risk of introducing a strong reference bias into our evaluation of HTER.

At first, it might seem reasonable to attempt to measure this reference bias by comparison of $\text{DA}_{\text{gen-ref}}$ scores for translations with scores from an equivalent bilingual human evaluation, where the MT output is no longer compared to a reference translation, but is instead compared to the source language input and evaluated by a bilingual speaker. We refer to DA scores collected in this way as DA_{src}. However, in this case, DA_{src} scores could in fact include a different bias, one introduced by the fact that human assessors are now non-native in at least one of the required languages. To complicate things further, since non-native language skill levels will vary considerably from one human assessor to another, the degree of bias is likely to change considerably depending on particular language ability levels. Therefore, in addition to the potential of monolingual reference bias, we point out the real possibility in the case of DA_{src} of bias caused by reliance on bilingual human assessors not native in either the source or the target language. A comparison between DA_{src} and $\text{DA}_{\text{gen-ref}}$ scores for translations, therefore, would unfortunately not provide a reliable measurement of the monolingual reference bias, since such a comparison could be confounded by this additional bias.

Subsequently, we carry out a distinct comparison and motivate it as follows. Reference bias in $\text{DA}_{\text{gen-ref}}$ scores should cause unfairly low scores for only *some* translations, those that do not closely match generic reference translations but are in fact high quality translations. The difficulty in measur-

[5]This additional investigation was requested during peer review. Due to time limitations we only provide a preliminary investigation including a single language pair in this current publication. We hope to provide a more detailed study in the near future.

ing reference bias lies in separating the fair attribution of low scores to translations that do not closely match the reference and are *genuinely low quality* from those that also do not correspond closely to a given reference but nonetheless are in fact *high quality*. A comparison that would provide insight into the degree to which reference bias exists for $DA_{\text{gen-ref}}$ is one that compares $DA_{\text{gen-ref}}$ scores with those collected in a corresponding monolingual DA evaluation in which the surface similarity between MT output and reference translations is maximized for all translations in the test set. In this way, translations that have been unfairly penalized by $DA_{\text{gen-ref}}$ can be identified as having low $DA_{\text{gen-ref}}$ scores, when unfairly evaluated by comparison with a dissimilar generic reference, while at the same time achieving a high score when evaluated against a maximally similar reference translation. We therefore run DA with reference translations that have maximal surface similarity to MT output translations, by creating a human-targeted reference translation for each MT output.

3.4.1 DA with Source-Generated Human-Targeted References

In our earlier evaluation of HTER, human-targeted references were created by comparison with the generic reference. To guard against reference-bias being introduced in this current experiment, human-targeted references are now created without the use of the generic reference, instead by a bilingual examination of the original source language input and MT output only. We therefore employ a known-reliable bilingual native speaker of the source language with high fluency in the target language, as well as a known-reliable native speaker of the target language. In a first step, the bilingual post-edits the MT output, and in a second step, the post-edited MT output is checked for fluency in the target language by the monolingual in consultation with the bilingual to ensure human-targeted references remained faithful to the meaning of the source input.[6] We will refer to this additional set of DA scores as $DA_{\text{src-ht-ref}}$. In this way, translations that have been unfairly penalized by $DA_{\text{gen-ref}}$ can be identified as having low $DA_{\text{gen-ref}}$ scores, when unfairly evaluated by comparison with a dissimilar generic reference, while at the same time achieving a high $DA_{\text{src-ht-ref}}$ score when evaluated against a maximally similar reference translation.

3.4.2 Reference Bias in Monolingual MT Evaluation Experiment Results

Figure 3 (b) shows a scatter plot of $DA_{\text{src-ht-ref}}$ and $DA_{\text{gen-ref}}$ scores for our previous sample of RU-EN translations originally sampled from WMT-13. If a set of translations scored by $DA_{\text{gen-ref}}$ include reference bias, we would expect them to appear in the lower right quadrant of Figure 3 (b), as they will receive a low $DA_{\text{gen-ref}}$ score in combination with a high $DA_{\text{src-ht-ref}}$ score. As can be seen from the lower right portion of Figure 3 (b), only a small number of translations fall into the lower right quadrant, and the small number that do, all lie in very close proximity to neighboring upper-right and lower-left quadrants. Conversely, Figure 3 (a) shows the correspondence between $DA_{\text{src-ht-ref}}$ scores and the top-performing metric for that language pair, CDER,[7] where, in contrast, many translations, located in the lower right quadrant, receive an unfairly low CDER score while being scored highly by $DA_{\text{src-ht-ref}}$.

The lack of translations simultaneously receiving a low $DA_{\text{gen-ref}}$ score and high $DA_{\text{src-ht-ref}}$ score indicates that even though $DA_{\text{gen-ref}}$ employs a generic reference translation, scores are not strongly biased by the presence of this reference translation. This reinforces the validity of our human evaluation methodology, and should provide reassurance that in our direct assessment set-up, human assessors do in fact apply their human intelligence to the task, by reading a given translation and comparing its meaning to that of the generic reference, as instructed, as opposed to attributing scores to translations based on mere surface similarities between the MT output and generic reference translation.

4 Machine Translation Quality Estimation and Human-targeted Metrics

The now apparent lack of reliability of HTER as a valid substitute for human assessment raises doubts as to whether or not HTER scores should be relied upon as a gold standard representation for tasks such as QE, where the ultimate goal of systems is to predict translation quality. For example, previous evaluations of quality estimation systems have made the assumption that HTER provides a reliable human evaluation

[6] Post-editing guidelines are provided at `https://github.com/ygraham/direct-assessment/bilingual-postedit-guidelines`.

[7] Metric scores are standardized by the mean and standard deviation for ease of comparison.

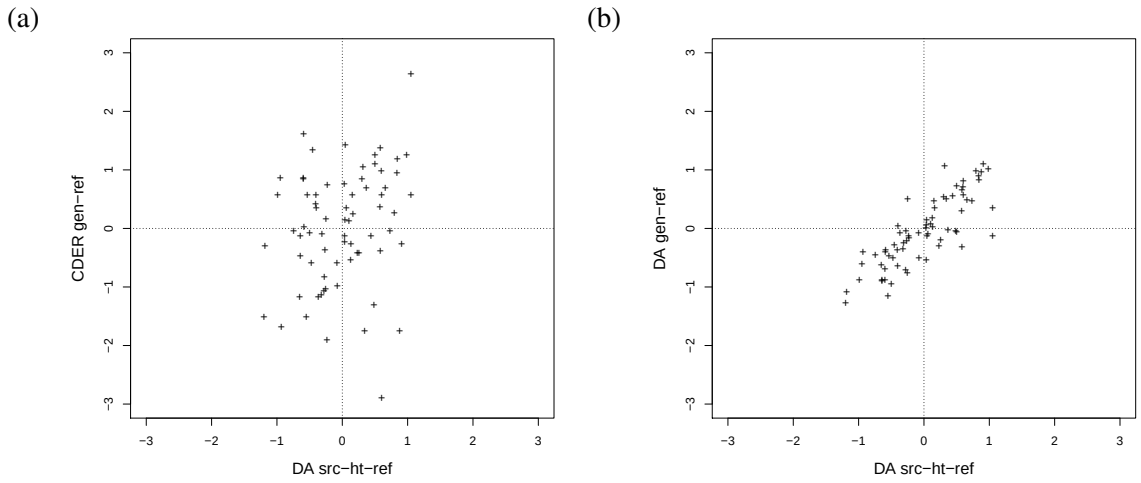


Figure 3: (a) Correspondence between DA scores collected by comparison with a source-generated human-targeted reference translation, DA src-ht-ref, where all translations benefit equally from close similarity to human-targeted references and automatic metric scores, CDER gen-ref, known to be biased by the generic reference and (b) the same instead compared to DA scores collected with comparison to generic reference translations, DA gen-ref. Lack of translations appearing in the lower right quadrant of (b) suggest DA human evaluation that employs a generic reference does not suffer from the strong reference bias known to be present in automatic metric scores (WMT-13 Russian to English).

substitute, but since we now know that HTER scores can be unrepresentative of translation quality, we rerun the previous evaluation provided by Bojar et al. (2013) and subsequently Graham (2015), instead employing DA human assessment as gold standard labels.

4.1 Human Annotation

Human-targeted reference translations were readily available in the WMT-13 shared task data set, while the human assessment method described in Section 3.1, was again applied to translations via crowdsourcing on Mechanical Turk, again applying strict quality control and computing mean scores for individual translations from a minimum of 15 human assessments. All human assessors who passed quality control by having significantly different scores for degraded versus genuine MT system output also showed no significant difference between mean rating scores for exact repeat translations.

Figure 4 (a) plots correspondence between HTER scores and human assessment, where the correlation of HTER with human assessment achieved is 0.678, almost at the same level as the strongest correlation in our nine-language pair evaluation. This data set therefore provides a best-case scenario for comparing differences in QE system rankings when HTER, as opposed to when human assessment, is employed as gold labels.

Before we compare QE system rankings, however, we carry out an analysis of a somewhat surprising relationship between HTER scores and human assessment for some translations shown in the scatter-plot in Figure 4 (a), where a number of translations appear to achieve a perfect HTER score of zero while at the same time cover a wide range of different human adequacy levels. Achieving a perfect HTER score means that the MT output was considered completely correct and required no post-editing whatsoever, so observing perfect HTER scores for translations that also receive low human assessment scores is a likely indication that something is amiss with either the post-editing carried out to create human-targeted references used to compute HTER scores or the human assessment.

Details of two example translations are provided in Figure 5. In Example 1, the source of error is the human post-editor (WMT-13 data set), since the MT output was in fact ungrammatical and despite this went uncorrected by the human post-editor, and this subsequently yielded an incorrect HTER score. Conversely, the error in Example 2 is caused by there being no direct translation of *straightforward* in Spanish and something unexpected subsequently taking place. Both the generic reference and the

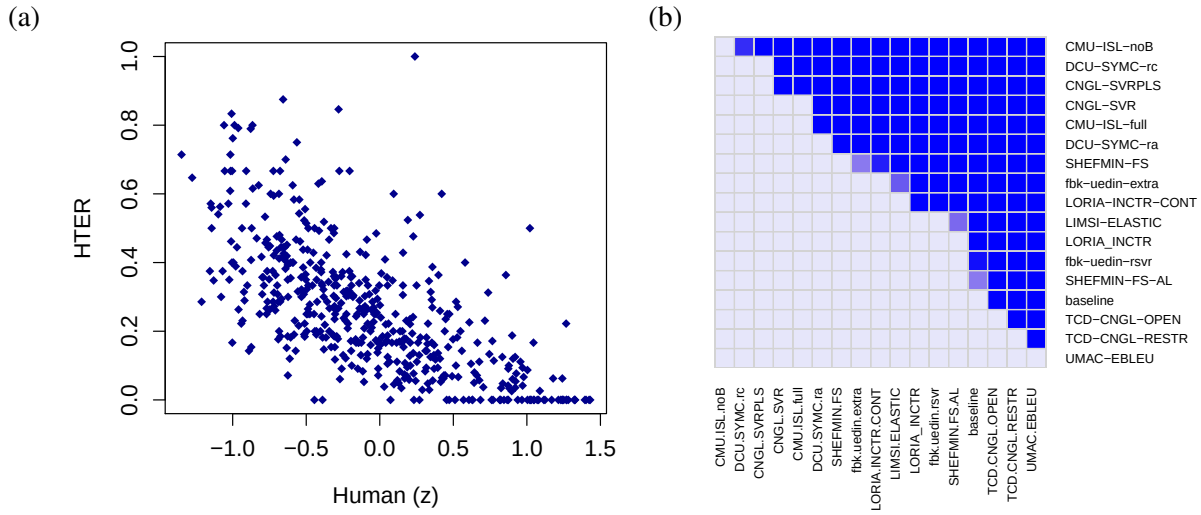


Figure 4: (a) Correspondence of HTER scores for translations in WMT-13 quality estimation Task 1.1 and human adequacy scores, and (b) significance test results (Williams test) of statistical significance of a difference in dependent correlations with human assessment for competing WMT-13 Task 1.1 quality estimation systems, a dark blue cell denotes a significant increase in correlation with human assessment for the QE system in a given row over the system in a given column at $p < 0.05$.

			HTER	Hum. z	Hum. raw
1.	source	Maybe we're more "Mexican" than I thought.			
	MT == HT ref.	*Tal vez estamos más "de México" de lo que se pensaba. *Perhaps we are more "from Mexico" than previously thought.*	0	−0.44	43%
	generic ref.	Quizás seamos más "mexicanos" de lo que pensaba. *Perhaps we are more "Mexican" than I thought.*			
2.	source	A straightforward man			
	MT == HT ref.	Un hombre sencillo *A simple man*	0	−0.39	43%
	generic ref.	Un hombre sincero *A sincere man*			

Figure 5: Example translations with a perfect HTER score (according to WMT-13 QE data set) and range of human assessment scores, where *denotes text ungrammatical in Spanish; generic ref. = the generic reference displayed to human assessors; MT = the MT output or human-targeted reference (HT ref.) employed by HTER (the latter being one and the same for translations receiving a perfect HTER score).

MT output could both be considered correct translations of the source while at the same time each of their meanings diverges somewhat from the other. Since the generic reference is employed for human assessment, the translation receives a low human adequacy score, despite it being a good translation of the original. Although human assessment is more valid than part-automatic metric scores, neither method is entirely impervious to other sources of error, therefore.

4.2 Quality Estimation Results

Table 3 shows correlations of QE system predictions with gold labels, where the rank order of systems diverges considerably when gold labels take the form of HTER scores as opposed to human assessment.

Significance test results for differences in correlation with human assessment for each pair of competing systems are provided in Figure 4 (b), where a new distinct outright winner of the task is identified that significantly outperforms all others, when evaluated with human assessment, as CMU-ISL-noB.

	Human Assessment		HTER			
QE System	r	Rank (r)	r	Rank (r)	MAE	Rank (MAE)
CMU-ISL-noB	0.571	1	0.516	5	0.138	7
DCU-SYMC-rc	0.557	2	0.595	1	0.135	5
CNGL-SVRPLS	0.553	3	0.560	4	0.133	3
CNGL-SVR	0.536	4	0.508	6	0.138	7
CMU-ISL-full	0.532	5	0.494	7	0.152	15
DCU-SYMC-ra	0.510	6	0.572	3	0.135	5
SHEFMIN-FS	0.489	7	0.575	2	0.124	1
fbk-uedin-extra	0.475	8	0.483	8	0.144	9
LORIA-INCTR-CONT	0.470	9	0.474	10	0.148	11
LIMSI-ELASTIC	0.459	10	0.475	9	0.133	3
LORIA_INCTR	0.452	11	0.461	13	0.148	11
fbk-uedin-rsvr	0.447	12	0.464	12	0.145	10
SHEFMIN-FS-AL	0.444	13	0.474	10	0.130	2
baseline	0.430	14	0.451	14	0.148	11
TCD-CNGL-OPEN	0.278	15	0.329	15	0.148	11
TCD-CNGL-RESTR	0.227	16	0.291	16	0.152	15
UMAC-EBLEU	0.017	17	0.113	17	0.170	17

Table 3: Pearson correlation of QE system predictions with HTER scores and correlation of system predictions with DA human adequacy scores for WMT-13 QE Task 1.1

The divergence in system rankings between evaluation of systems with HTER compared to DA indicates that even in the case that HTER correlates with human assessment to a relatively strong degree, 0.678, HTER is not a sufficient stand-in for human assessment. We subsequently recommend, where possible, employment of DA human assessment for evaluation of quality estimation systems, as it provides a valid human assessment without an automatic component, and has been shown to produce replicable sentence-level human assessment scores for translations (Graham et al., 2015).

5 Conclusion

Concerns were raised about the reliability of conclusions drawn in past experiments about HTER's superiority to other human-targeted metric formulations, and issues with respect to human post-editor coaching were highlighted as a source of bias, in addition to limited experiment settings. Subsequently, this motivated our re-evaluation of HTER extending to include a large number of systems across nine different language pairs. Results of our re-evaluation of HTER and several other possible metrics, including HBLEU, reveal significantly stronger correlations with human assessment for HBLEU compared to those achieved by HTER for two language pairs and no significant difference present in all other cases.

Further results showed correlations achieved by HTER may vary considerably across language pairs and in some cases are too low to provide a valid substitute for human assessment. As a test-case we replicated a previous quality estimation shared task and even when HTER scores correlate with human assessment at 0.678, they do not provide a valid human assessment substitute, with system rankings diverging considerably when DA human assessment is employed. We recommend DA human assessment for future evaluation of quality estimation.

Acknowledgments

We wish to thank Antonio Toral, Joachim Wagner and the anonymous reviewers for their feedback. This project has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement 645452 (QT21) and the ADAPT Centre for Digital Content Technology (`www.adaptcentre.ie`) at Dublin City University funded under the SFI Research Centres Programme (Grant 13/RC/2106) co-funded under the European Regional Development Fund.

References

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal, September. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 Workshop on Statistical Machine Translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada. Association for Computational Linguistics.

Marina Fomicheva and Lucia Specia. 2016. Reference bias in monolingual machine translation evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 77–82, Berlin, Germany. Association for Computational Linguistics.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2013. Continuous measurement scales in human evaluation of machine translation. In *Proceedings of the 7th Linguistic Annotation Workshop & Interoperability with Discourse*, pages 33–41, Sofia, Bulgaria. Association for Computational Linguistics.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2015. Accurate evaluation of segment-level machine translation metrics. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies*, pages 1183–1191, Denver, Colorado. Association for Computational Linguistics.

Yvette Graham. 2015. Improving evaluation of machine translation quality estimation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1804–1813, Beijing, China. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, Evan Herbst, and Hieu Hoang. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Gregor Leusch and Hermann Ney. 2008. BLEUSP, INVWER, CDER: Three improved MT evaluation measures. In *Proceedings of NIST Metrics for Machine Translation 2008 Evaluation*, Honolulu, HI.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA. Association for Computational Linguistics.

M. Snover, B. Dorr, R. Scwartz, J. Makhoul, and L. Micciula. 2006. A study of translation error rate with targeted human annotation. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translaiton in the Americas*, pages 223–231, Boston, MA.

Connecting Phrase based Statistical Machine Translation Adaptation

Rui Wang[1,2], **Hai Zhao**[1,2] *, **Bao-Liang Lu**[1,2], **Masao Utiyama**[3] * **and Eiichro Sumita**[3]
[1]Department of Computer Science and Engineering,
[2]Key Lab of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering,
Shanghai Jiao Tong University, Shanghai, China
[3]National Institute of Information and Communications Technology, Kyoto, Japan
wangrui.nlp@gmail.com, {zhaohai, blu}@cs.sjtu.edu.cn,
{mutiyama, eiichiro.sumita}@nict.go.jp

Abstract

Although more additional corpora are now available for Statistical Machine Translation (SMT), only the ones which belong to the same or similar domains of the original corpus can indeed enhance SMT performance directly. A series of SMT adaptation methods have been proposed to select these similar-domain data, and most of them focus on sentence selection. In comparison, phrase is a smaller and more fine grained unit for data selection, therefore we propose a straightforward and efficient connecting phrase based adaptation method, which is applied to both bilingual phrase pair and monolingual n-gram adaptation. The proposed method is evaluated on IWSLT/NIST data sets, and the results show that phrase based SMT performances are significantly improved (up to +1.6 in comparison with phrase based SMT baseline system and +0.9 in comparison with existing methods).

1 Introduction

Large corpora are important for Statistical Machine Translation (SMT) training. However only the relevant additional corpora, which are also called in-domain or related-domain corpora, can enhance the performance of SMT effectively. Otherwise the irrelevant additional corpora, which are also called out-of-domain corpora, may not benefit SMT (Koehn and Schroeder, 2007).

SMT adaptation means selecting useful part from mix-domain (mixture of in-domain and out-of-domain) data, for SMT performance enhancement. The core task in adaptation is about how to select the useful data. Existing works have considered selection strategies with various granularities, though most of them only focus on sentence-level selection (Axelrod et al., 2011; Banerjee et al., 2012; Duh et al., 2013; Hoang and Sima'an, 2014a; Hoang and Sima'an, 2014b). There is a potential problem for sentence level adaptation: different parts of a sentence may belong to different domains. That is, it is possible that a sentence is overall out-of-domain, although part of it can be in-domain. Therefore a few works consider more granular level for selection. They build lexicon, Translation Models (TMs), reordering models or Language Models (LMs) to select fragment or directly adapt the models (Bellegarda, 2004; Deng et al., 2008; Moore and Lewis, 2010; Foster et al., 2010; Mansour and Ney, 2013; Carpuat et al., 2013; Chen et al., 2013a; Chen et al., 2013b; Sennrich et al., 2013; Mathur et al., 2014; Shi et al., 2015). One typical example of these methods is to train two Neural Network (NN) models (one from in-domain and the other from out-of-domain) and penalize the sentences/phrases similar to out-of-domain corpora (Duh et al., 2013; Joty et al., 2015; Durrani et al., 2015). As we know, Phrase Based SMT (PBSMT) mainly contains two models: translation model and LM, whose components are bilingual phrase pairs and monolingual n-grams. Meanwhile, most of the above methods enhance SMT performance by adapting single specific model.

*Corresponding authors. H. Zhao and B. L. Lu were partially supported by Cai Yuanpei Program (CSC No. 201304490199 and 201304490171), National Natural Science Foundation of China (No. 61672343, 61170114 and 61272248), National Basic Research Program of China (No. 2013CB329401), Major Basic Research Program of Shanghai Science and Technology Committee (No. 15JC1400103), Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04), and Key Project of National Society Science Foundation of China (No. 15-ZDA041).

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3135–3145, Osaka, Japan, December 11-17 2016.

Instead of focusing on sentence selection or single model adaptation, we propose a phrase adaptation method, which is applied to both bilingual phrase pair and monolingual n-gram selection. It is based on a linguistic observation that the translation hypotheses of a phrase-based SMT system are concatenations of phrases from Phrase Table (PT), which has been applied to LM growing (Wang et al., 2014a; Wang et al., 2015). As a straightforward linear method, it is much efficient in comparison with NN based non-linear methods.

The remainder of this paper is organized as follows. Section 2 will introduce the connecting phrase based adaptation method. The size of adapted connecting phrase will be tuned in Section 3. Empirical results will be shown in Section 4. We will discuss the methods and conduct extension experiments in Section 5. The last section will conclude this paper.

2 Connecting Phrase based Adaptation

Suppose that two phrases '*would like to learn*' and '*Chinese as second language*' are in the in-domain PT. In decoding, these two phrases may be connected together as '*would like to learn Chinese as second language*". The phrases '*would like to learn Chinese*' or '*learn Chinese as second language*' may be outside in-domain PT/LM, but they may possibly be in out-of-domain PT/LM. Traditionally their translation probabilities are only calculated by the combination of probabilities from in-domain PT/LM. For the proposed methods, the translation probabilities of connecting phrases from out-of-domain corpus are estimated by real corpus directly. If we can add these connecting phrases with their translation probabilities, which may be useful in decoding, into in-domain bilingual (together with source part phrases) PT or monolingual LM, they may help improve SMT.

Note that connecting phrases are generated from in-domain PT, it is necessary to check if these in-domain connecting phrases actually occur in out-of-domain PT/LM. Connecting phrases can occur in decoding by combining two phrases from in-domain PT.

Let w_a^b be a phrase starting from the a-th word and ending with the b-th word, and $\gamma w_a^b \beta$ be a phrase including w_a^b as a part of it, where γ and β represent any word sequence or none. An i-gram phrase $w_1^k w_{k+1}^i$ $(1 \leq k \leq i-1)$ is a connecting phrase[1] (Wang et al., 2014a), if

1) w_1^k is right (rear) part of one phrase γw_1^k in the in-domain PT, and

2) w_{k+1}^i is left (front) part of one phrase $w_{k+1}^i \beta$ in the in-domain PT.

For example, let '*a b c d*' be a 4-gram phrase, it is a connecting phrase if at least one of the following conditions holds:

1) 'γ *a*' and '*b c d* β' are in phrase table, or

2) 'γ *a b*' and '*c d* β' are in phrase table, or

3) 'γ *a b c*' and '*d* β' are in phrase table.

For a phrase pair (F, E) in out-of-domain PT, there are four cases: a) Both F and E, b) either F or E, c) only F, d) only E are/is connecting phrase(s). We empirically evaluate the performance of these four cases and the results show that a) gains the highest BLEU, so it is adopted at last. For an n-gram LM, we only consider target side information.

3 Adapted Phrase Size Tuning

A lot of connecting phrases are generated in the above way. We propose two methods to rank these phrases and only the top ranked ones are added into in-domain PT/LM.

[1]We are aware that connecting phrases can be applied to three or more phrases. Experimental results show that using more than two connecting phrases cannot further improve the performance, so only two connecting phrases are applied.

3.1 Occurring Probability based Tuning

The potential Occurring Probability (OP) of a source phrase $P_{op}(F)$ and $P_{op}(E)$ are defined as,

$$P_{op}(F) = \sum_{k=1}^{p-1}(\sum_{\beta} P_s(\beta f_1^k) \times \sum_{\gamma} P_s(f_{k+1}^p \gamma)),$$

$$P_{op}(E) = \sum_{k=1}^{q-1}(\sum_{\beta} P_t(\beta e_1^k) \times \sum_{\gamma} P_t(e_{k+1}^q \gamma)),$$

respectively, where P_s (for source phrase f_1^p) or P_t (for target phrase e_1^q) is calculated using source or target monolingual LM trained from in-domain corpus.

The $P_{op}(F, E)$ of a connecting phrase pair (F, E) in SMT decoding is defined as $P_{op}(F) \times P_{op}(E)$. $P_{op}(F, E)$ is used to rank connecting phrase pairs. For target LM, only $P_{op}(E)$ is used to rank connecting n-gram (Wang et al., 2014a).

3.2 NN based Tuning

The basic hypothesis of NN based adaptation is: two NN models (translation model as NNTM or LM as NNLM), one from in-domain and one from out-of-domain are trained. Taking NNTM as example, for a phrase pair (F, E) relevant with in-domain ones, the translation probabilities $P_{in}(E|F)$ by $NNTM_{in}$ should be larger and $P_{out}(E|F)$ by $NNTM_{out}$ should be lower. This hypothesis is partially motivated by (Axelrod et al., 2011), which use bilingual cross-entropy difference to distinguish in-domain and out-of-domain data.

The translation probability of a phrase-pair is estimated as,

$$P(E|F) = P(e_1, ..., e_q|f_1, ..., f_p), \tag{1}$$

where f_s ($s \in [1, p]$) and e_t ($t \in [1, q]$) are source and target words, respectively. Originally,

$$P(e_1, ..., e_q|f_1, ..., f_p) = \prod_{k=1}^{q} P(e_k|e_1, ..., e_{k-1}, f_1, ...f_p). \tag{2}$$

The structure of NN based translation model is similar to Continuous Space Translation Model (CSTM) (Schwenk, 2012). For the purpose of adaptation, the dependence between target words is dropped[2] and the probabilities of different length target phrase are normalized. For an incomplete source phrase, i.e. with less than seven words, we set the projections of the missing words to zero. The normalized translation probability $Q(E|F)$ can be approximately computed by the following equation,

$$Q(E|F) \approx \sqrt[q]{\prod_{k=1}^{q} P(e_k|f_1, ...f_p)}. \tag{3}$$

Finally, the minus $D_{minus}(E|F)$ is used to rank connecting phrase pairs from mix-domain PT,

$$D_{minus}(E|F) = Q_{in}(E|F) - Q_{in}(E|F). \tag{4}$$

where $Q_{in}(E|F)$ and $Q_{in}(E|F)$ are corresponding probabilities from in-domain and out-of-domain NNTMs.

For monolingual n-gram tuning, two NNLMs (in and out) are trained, and

$$D_{minus}(E) = Q_{in}(E) - Q_{out}(E), \tag{5}$$

[2]We have also empirically compared the performance of using NN with target word dependence and the results are not that positive.

where $Q_{in}(E)$ and $Q_{out}(E)$ are corresponding probabilities from in-domain and out-of-domain NNLMs. $D_{minus}(E)$ is used for n-gram ranking.

Beside for connecting phrases size tuning, this NN[3] based method can also be applied to phrase adaptation directly, which is similar as other NN based adaptation methods, such as (Duh et al., 2013) for sentence selection and (Joty et al., 2015) for joint model adaptation. In addition, the translation probabilities of connecting phrases calculated by NN can also be used to enhance SMT, and the experimental results will be shown in Section 5.4.

3.3 Integration into SMT

The thresholds of P_{op} and D_{minus} are tuned using development data. Selected phrase pairs are added into the in-domain PT. Because they are not so useful as the in-domain ones, a penalty score is added. For in-domain phrase pairs, the penalty is set as 1; for the out-of-domain ones the penalty is set as e (= 2.71828...). Other phrase scores (lexical weights et. al.) are used as they are. This penalty setting is similar to (Bisazza et al., 2011). Penalty weights, together with all of existing score weights, will be further tuned by MERT (Och, 2003). The phrase pairs in re-ordering model are selected using the same way as PT. The selected monolingual n-grams are added to the original LM, and the probabilities are re-normalized by SRILM (Stolcke, 2002; Stolcke et al., 2011).

4 Experiments

4.1 Data sets

The proposed methods are evaluated on two data sets. 1) IWSLT 2014 French (FR) to English (EN) corpus[4] is used as in-domain data and dev2010 and test2010/2011 (Niehues and Waibel, 2012), are selected as development (dev) and test data, respectively. Out-of-domain corpora contain Common Crawl, Europarl v7, News Commentary v10 and United Nation (UN) FR-EN parallel corpora[5]. 2) NIST 2006 Chinese (CN) to English corpus[6] is used as in-domain corpus, which follows the setting of (Wang et al., 2014b) and mainly consists of news and blog texts. Chinese to English UN data set (LDC2013T06) and NTCIR-9 (Goto et al., 2011) patent data are used as out-of-domain bilingual (Bil) parallel corpora. The English patent data in NTCIR-8 (Fujii et al., 2010) is also used as additional out-of-domain monolingual (Mono) corpus. NIST Eval 2002-2005 and NIST Eval 2006 are used as dev and test data, respectively.

IWSLT FR-EN	Sentences	Tokens
in-domain	178.1K	3.5M
out-of-domain	17.8M	450.0M
dev	0.9K	20.1K
test2010	1.6K	31.9K
test2011	1.1K	21.4K
NIST CN-EN	**Sentences**	**Tokens**
in-domain	430.8K	12.6M
out-of-domain (Bil)	8.8M	249.4M
out-of-domain (Mono)	33.7M	1.0B
dev (average of four)	4.4K	145.8K
test (average of four)	1.6K	46.7K

Table 1: Statistics on data sets ('B' for billions).

[3]NN based methods have been applied to a series of NLP tasks, such as Chinese word segmentation and parsing (Cai and Zhao, 2016; Zhang et al., 2016).

[4]https://wit3.fbk.eu/mt.php?release=2014-01

[5]http://statmt.org/wmt15/translation-task.html

[6]http://www.itl.nist.gov/iad/mig/tests/mt/2006/

4.2 Common Setting

The basic settings of IWSLT-2014 FR to EN and NIST-06 CN to EN phrase based translation baseline systems are followed. 5-gram interpolated KN (Kneser and Ney, 1995) LMs are trained. Translation performances are measured by case-insensitive BLEU (Papineni et al., 2002) with significance test (Koehn, 2004) and METEOR (Lavie and Agarwal, 2007). MERT (Och, 2003) (BLEU based) is run three times for each system and the average BLEU/METEOR scores are recorded. 4-layer CSTM (Schwenk, 2012) are applied to NN translation models: phrase length limit is set as seven, shared projection layer of dimension 320 for each word (that is 2240 for seven words), projection layer of dimension 768, hidden layer of dimension 512. The dimensions of input/output layers for both in/out-of-domain CSTMs follows the size of vocabularies of source/target words from in-domain corpora. That is 72K/57K for IWSLT and 149/112K for NIST. Since out-of-domain corpora are huge, part of them are resampled (resample coefficient 0.01 for IWSLT and NIST).

Several related existing methods are selected as baselines[7]: Koehn and Schroeder (2007)'s method for using two (in and out-of-domain) TMs and LMs together, entropy based method for TM (Ling et al., 2012) and LM (Stolcke, 1998) adaptation (pruning), (Duh et al., 2013) for NNLM based sentence adaptation, (Sennrich, 2012) for TM weights combination, and (Bisazza et al., 2011) for TM fill-up. In Table Tables 2 and 3, 'in-domain', 'out-of-domain' and 'mix-domain' indicate training all models using corresponding corpora, 'in+NN' indicates applying NN based adaptation directly for all phrases, and 'in+connect' indicates adding all connecting phrases and n-grams to in-domain PT and LM, respectively. For tuning methods, 'in+connect+OP/NN' indicates tuning connecting phrase pairs and n-grams using Occurring Probability (OP) and NN, respectively. Only the best preforming systems (for both the baselines and proposed methods) on development data are chosen to be evaluated on test data.

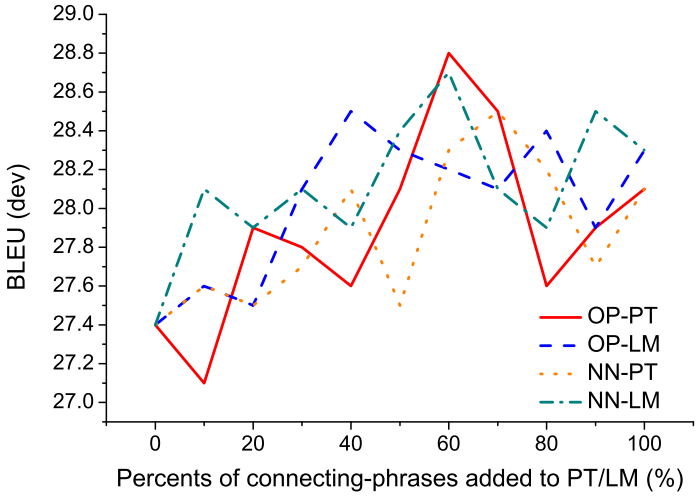


Figure 1: Connecting phrases size tuning on IWSLT.

4.3 Results and Analysis

For all ranked connecting phrase pairs and n-grams, we empirically add different sized (top) parts of them into PT/LM for size tuning. Figure 1 shows performances of the proposed tuning methods on IWSLT development data set. The results show that adding connecting phrases can enhance SMT performance in most of cases. Meanwhile, the tuned connecting phrases, which are parts of the whole, gain more BLEU improvement. They are considered as the most useful connecting phrases and evaluated on the test data sets.

[7]We are aware that there are various SMT adaptation works such as (Deng et al., 2008; Hoang and Sima'an, 2014a; Joty et al., 2015). However, there does not exist a commonly used evaluation corpus for this task, and either detailed implementations or experimental settings are absent for most published works.

Methods	PT Size	LM Size	BLEU test10	METEOR test10	BLEU test11	METEOR test11
in-domain	9.8M	7.9M	31.94	34.07	29.16	32.34
out-of-domain	759.0M	497.4M	27.34	32.22	23.80	30.48
mix-domain	765.4M	503.1M	30.07	33.19	26.42	31.06
Koehn's method	N/A	N/A	32.42	**34.32**	29.41	32.41
entropy method	247.8M	146.1M	32.54	34.12	29.23	32.17
Duh's method	765.4M	271.0M	**32.65**	34.31	29.18	32.57
Sennrich's method	765.4M	503.1M	32.41	34.32	**29.67**	**32.71**
Bisazza's method	765.4M	503.1M	32.24	34.28	29.35	32.53
in+NN	296.8M	156.2M	32.54	34.25	29.67	32.68
in+connect	184.5M	133.8M	33.26+	34.60	30.07	32.89
in+connect+OP	122.0M	53.5M	**33.53**++	**34.77**	30.25	32.91
in+connect+NN	141.3M	80.3M	32.91	34.56	**30.32**+	**33.17**

Table 2: IWSLT FR-EN Results. "++": BLEU significantly better than corresponding the best performed baseline (in **bold**) at level $\alpha = 0.01$, "+": $\alpha = 0.05$. Koehn's method uses two TMs and LMs, so their sizes are hard to tell.

Methods	PT Size	LM Size	BLEU	METEOR
in-domain	27.2M	23.9M	32.10	29.29
out-of-domain	365.8M	1.2B	27.85	22.48
mix-domain	370.9M	1.2B	31.37	28.80
Koehn's method	N/A	N/A	31.93	29.32
entropy method	165.3M	279.5M	32.29	29.17
Duh's method	160.5M	519.3M	**32.51**	29.36
Sennrich's method	370.9M	1.2B	32.36	**29.88**
Bisazza's method	370.9M	1.2B	32.15	29.72
in+NN	187.6M	394.1M	32.82+	30.23
in+connect	142.6M	298.1M	32.63	29.97
in+connect+OP	92.6M	208.7M	32.76	**30.63**
in+connect+NN	113.6M	142.1M	**33.23**++	30.54

Table 3: NIST-06 CN-EN Results.

Tables 2 and 3 shows that directly using 'out-of-domain' or 'mix-domain' data will cause SMT performances decrease in comparison with 'in-domain' data. Adding connecting phrases will enhance SMT performances and the proposed tuning method can further increase SMT performances significantly (up to +1.6 BLEU in IWSLT task and +1.1 in NIST task) and outperform the existing methods (up to +0.9 BLEU in IWSLT task and +0.7 in NIST task). The NN method performs better as a tuning method than as a direct adaptation method.

5 Discussions

5.1 Individual Model Analysis

Most of the existing methods focus on single model adaptation, however the proposed connecting phrase method can be applied to both TM and LM. So it seems a little unfair to compare the existing methods with our methods. To compare with them in a more fair way, we show the performance of individual model in Tables 4 and 5 for IWSLT tasks. Similar as the previous experiments, only the best performing system on development data of each method is evaluated on the test data.

Methods	LM Size	BLEU test10	BLEU test11
in-domain	7.9M	31.94	29.16
out-of-domain	497.4M	31.01	27.42
mix-domain	503.1M	32.23	28.42
Koehn's method	N/A	32.34	29.10
entropy method	146.1M	32.31	29.24
Duh's method	271.0M	32.65	29.18
in+NN	156.2M	32.66	29.38
in+connect	133.8M	32.78	29.32
in+connect+OP	53.5M	**32.95**	29.45
in+connect+NN	80.3M	32.56	**29.78**

Table 4: IWSLT FR-EN results on LM adaptation.

Methods	PT Size	BLEU test10	BLEU test11
in-domain	9.8M	31.94	29.16
out-of-domain	759.0M	28.62	24.56
mix-domain	765.4M	29.56	26.78
Koehn's method	N/A	31.97	29.21
entropy method	247.8M	32.43	28.73
Sennrich's method	765.4M	32.41	29.67
Bisazza's method	765.4M	32.24	29.35
in+NN	296.8M	32.31	29.63
in+connect	184.5M	32.87	29.48
in+connect+OP	122.0M	**33.05**	29.77
in+connect+NN	141.3M	32.73	**29.89**

Table 5: IWSLT FR-EN results on TM adaptation.

As shown in Tables 4 and 5, the proposed methods outperform existing methods in individual model performance (up to +0.3 BLEU in LM task and +0.6 BLEU in TM task for test10 and +0.5 BLEU in LM task and +0.2 BLEU in TM task for test11). Another observation is that adding out-of-domain data into

TM hurt SMT system more seriously than LM (-0.9 BLEU in LM task versus -3.4 BLEU in TM task for test10 and -1.7 BLEU in LM task versus -4.6 BLEU in TM task for test11).

5.2 Manual Example

A few adapted phrase examples of IWSLT FR-EN task are in Table 6. For NN based method (direct apply NN in adaptation), some phrases with similar meaning are adapted, such as *third world countries* and *developing countries*. For connecting phrase method, phrases which are combination of phrases are adapted, such as *the reason* and *why I like* form *the reason why I like*.

Methods	Source Phrases	Original Target Phrases	Adapted Phrases
NN	*les pays en voie de développement*	*1. developing countries* *2. the developing countries* *3. all developing countries*	*1. developing countries* *2. third world countries* *3. countries in the developing world*
Connect	*la raison pour laquelle je tiens*	*1. the reason I want* *2. why I like* *3. I therefore wish*	*1. the reason why I like* *2. the reason I want* *3. the reason I would like*

Table 6: Some examples of adapted phrases, which are ranked by translation probabilities.

5.3 Efficiency Comparison

Table 7 shows the adaptation time of each method[8] on IWSLT task. The proposed methods show significant advantage over others, and NN based methods are very time consuming.

Methods	Adaptation Time
entropy method	12 hours
Duh's method	7 days
Bisazza's method	6 hours
in+NN	10 days
in+connect	**2 hours**
in+connect+OP	3 hours
in+connect+NN	3 days

Table 7: Efficiency comparison (CPU time) on IWSLT.

5.4 Adding NN Probabilities

As mentioned in Section 3.2, NN can be used to predict the translation probabilities of bilingual phrase pairs and the occurring probabilities of monolingual n-grams. The minus D_{minus} between in-domain NN probabilities Q_{in} and out-of-domain NN probabilities Q_{out} are used to judge whether a phrase (pair) is similar to the in-domain ones. Meanwhile, these in-domain NN probabilities Q_{in} themselves are also useful information. In the previous sections, the adapted phrase pairs are added into original PT or LM with their own probabilities. In this subsection, Q_{in} of adapted and original phrases are also adopted in SMT decoding. That is, $Q_{in}(E|F)$ is added as a feature for adapted and original phrase pairs in PT and $Q_{in}(E)$ of adapted and original n-grams are interpolated with n-gram LM probabilities.

The results in Table 8 show that the NN feature can enhance SMT performance slightly. Although this is not our main contribution, it shows the NN method cannot only be applied to phrase pair and n-gram adaptation, but also to probability estimation.

[8] Koehn and Schroeder (2007) is only for model combination, so we do not compare with it.

Methods	PT Size	LM Size	BLEU without Q_{in}	BLEU with Q_{in}
in-domain	9.8M	7.9M	31.94	32.34
in+NN	296.8M	156.2M	32.54	32.48
in+connect	184.5M	133.8M	33.26	33.45
in+connect+OP	122.0M	53.5M	33.53	**33.67**
in+connect+NN	141.3M	80.3M	32.91	33.12

Table 8: IWSLT FR-EN Results.

6 Conclusion

In this paper, we propose a straightforward connecting phrase based SMT adaptation method. Two model size tuning methods, NN and occurring probability are proposed to discard less reliable connecting phrases. The empirical results in IWSLT French to English and NIST Chinese to English translation tasks show that the proposed methods can significantly outperform a number of the existing SMT adaptation methods in both performance and efficiency. We also show some empirical results to discuss where the SMT improvements come from by individual model and manual example analysis.

Acknowledgements

Thanks Zhisong Zhang for helpful discussions and the three anonymous reviewers for helpful comments.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, U.K.

Pratyush Banerjee, Sudip Kumar Naskar, Johann Roturier, Andy Way, and Josef van Genabith. 2012. Translation quality-based supplementary data selection by incremental update of translation models. In *Proceedings of 24th International Conference on Computational Linguistics*, pages 149–166, Mumbai, India.

Jerome R Bellegarda. 2004. Statistical language model adaptation: review and perspectives. *Speech Communication*, 42(1):93–108.

Arianna Bisazza, Nick Ruiz, Marcello Federico, and FBK-Fondazione Bruno Kessler. 2011. Fill-up versus interpolation methods for phrase-based smt adaptation. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 136–143, San Francisco.

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 409–420, Berlin, Germany.

Marine Carpuat, Hal Daume III, Katharine Henry, Ann Irvine, Jagadeesh Jagarlamudi, and Rachel Rudinger. 2013. Sensespotting: Never let your parallel data tie you to an old domain. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1435–1445, Sofia, Bulgaria.

Boxing Chen, George Foster, and Roland Kuhn. 2013a. Adaptation of reordering models for statistical machine translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 938–946, Atlanta, Georgia.

Boxing Chen, Roland Kuhn, and George Foster. 2013b. Vector space model for adaptation in statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1285–1293, Sofia, Bulgaria.

Yonggang Deng, Jia Xu, and Yuqing Gao. 2008. Phrase table training for precision and recall: What makes a good phrase and a good phrase pair? In *Proceedings of 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 81–88, Columbus, Ohio.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 678–683, Sofia, Bulgaria, August.

Nadir Durrani, Hassan Sajjad, S Joty, A Abdelali, and S Vogel. 2015. Using joint models for domain adaptation in statistical machine translation. *Proceedings of the Fifteenth Machine Translation Summit*.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 451–459, Cambridge, MA, October.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, and Takehito Utsuro. 2010. Overview of the patent translation task at the ntcir-8 workshop. In *In Proceedings of the 8th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-lingual Information Access*, pages 293–302, Tokyo, Japan.

Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita, and Benjamin K. Tsou. 2011. Overview of the patent machine translation task at the NTCIR-9 workshop. In *Proceedings of NTCIR-9 Workshop Meeting*, pages 559–578, Tokyo, Japan.

Cuong Hoang and Khalil Sima'an. 2014a. Latent domain phrase-based models for adaptation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 566–576, Doha, Qatar.

Cuong Hoang and Khalil Sima'an. 2014b. Latent domain translation models in mix-of-domains haystack. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1928–1939, Dublin, Ireland.

Shafiq Joty, Hassan Sajjad, Nadir Durrani, Kamla Al-Mannai, Ahmed Abdelali, and Stephan Vogel. 2015. How to avoid unwanted pregnancies: Domain adaptation using neural network models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1259–1270, Lisbon, Portugal.

R. Kneser and H. Ney. 1995. Improved backing-off for m-gram language modeling. *Proceedings of 1995 IEEE International Conference on Acoustics, Speech, and Signal Processing,*, 1:181–184.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227, Prague, Czech Republic.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: an automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231.

Wang Ling, João Graça, Isabel Trancoso, and Alan Black. 2012. Entropy-based pruning for phrase-based machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 962–971, Jeju Island, Korea.

Saab Mansour and Hermann Ney. 2013. Phrase training based adaptation for statistical machine translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 649–654, Atlanta, Georgia.

Prashant Mathur, Sriram Venkatapathy, and Nicola Cancedda. 2014. Fast domain adaptation of smt models without in-domain parallel data. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1114–1123, Dublin, Ireland.

Robert C Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 220–224, Uppsala, Sweden.

Jan Niehues and Alex Waibel. 2012. Continuous space language models using restricted boltzmann machines. In *Proceedings of the International Workshop for Spoken Language Translation*, IWSLT 2012, pages 311–318, Hong Kong, China.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In *Proceedings of 24th International Conference on Computational Linguistics: Posters*, pages 1071–1080, Mumbai, India.

Rico Sennrich, Holger Schwenk, and Walid Aransa. 2013. A multi-domain translation model framework for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 832–840, Sofia, Bulgaria.

Rico Sennrich. 2012. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 539–549, Avignon, France.

Yangyang Shi, Martha Larson, and Catholijn M. Jonker. 2015. Recurrent neural network language model adaptation with curriculum learning. *Computer Speech and Language*, 33(1):136 – 154.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. SRILM at sixteen: Update and outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, Waikoloa, HI.

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proceeding of DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274, Lansdowne, VA.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286, Seattle.

Rui Wang, Hai Zhao, Bao-Liang Lu, Masao Utiyama, and Eiichiro Sumita. 2014a. Neural network based bilingual language model growing for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 189–195, Doha, Qatar.

Xiaolin Wang, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2014b. Empirical study of unsupervised chinese word segmentation methods for smt on large-scale corpora. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 752–758, Baltimore, Maryland.

Rui Wang, Hai Zhao, Bao-Liang Lu, M. Utiyama, and E. Sumita. 2015. Bilingual continuous-space language model growing for statistical machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(7):1209–1220.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1382–1392, Berlin, Germany.

Fast Collocation-Based Bayesian HMM Word Alignment

Philip Schulz
ILLC
University of Amsterdam
P.Schulz@uva.nl

Wilker Aziz
ILLC
University of Amsterdam
W.Aziz@uva.nl

Abstract

We present a new Bayesian HMM word alignment model for statistical machine translation. The model is a mixture of an alignment model and a language model. The alignment component is a Bayesian extension of the standard HMM. The language model component is responsible for the generation of words needed for source fluency reasons from source language context. This allows for untranslatable source words to remain unaligned and at the same time avoids the introduction of artificial NULL words which introduces unusually long alignment jumps. Existing Bayesian word alignment models are unpractically slow because they consider each target position when resampling a given alignment link. The sampling complexity therefore grows linearly in the target sentence length. In order to make our model useful in practice, we devise an auxiliary variable Gibbs sampler that allows us to resample alignment links in constant time independently of the target sentence length. This leads to considerable speed improvements. Experimental results show that our model performs as well as existing word alignment toolkits in terms of resulting BLEU score.

1 Introduction

Word alignment is one of the basic problems in statistical machine translation (SMT). The IBM models were originally devised for translation by Brown et al. (1993). Later, when SMT started to employ entire phrases instead of single words (Koehn et al., 2003), the IBM models were repurposed as word alignment models. The alignments they produce guide the phrase extraction heuristics that are used in many modern SMT systems.

There are several extensions of the classical IBM models that try to weaken their independence assumptions. Notably, Vogel et al. (1996) introduced Markovian dependencies between individual alignment links. Those links were treated as independent events in IBM models 1, 2 and 3. The model of Vogel et al. (1996) can be viewed as a Hidden Markov Model (HMM) in which the hidden Markov Chain induces a probability distribution over latent alignment links. Besides weakening the independence assumptions of the simpler IBM models, the HMM alignment model has the additional benefit of being tractable. This means that expectations under the HMM aligner can be computed exactly using the forward-backward algorithm. These expectations are then used in the Baum-Welch algorithm (Baum et al., 1970) to compute parameter updates for the model. Crucially, the Baum-Welch algorithm is a special case of the EM algorithm and thus guaranteed to never decrease the model's likelihood at each parameter update (Dempster et al., 1977).

Tractability and convergence (at least to a local optimum) are clear advantages of the HMM aligner over IBM models 3 to 5 which are all intractable. In practice, hill-climbing heuristics are employed to approximate expectations in these more complex models. Unfortunately, all convergence guarantees of the learning algorithm are lost this way.

A major problem for the HMM aligner is the handling of NULL words.[1] The NULL word is a special

[1]The original work of Vogel et al. (1996) did not use NULL words and instead aligned *all* source words. Later work by Och and Ney (2003) and Liang et al. (2006) has shown that using NULL words to allow for unaligned words improves performance.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3146–3155, Osaka, Japan, December 11-17 2016.

lexical item that is hypothesised to stand at the beginning of every target (English) sentence. Since in word alignment the source (French) side is generated given the target side, the NULL word is used to generate source words that do not have lexical translations on the target side. Such untranslatable words are often idiosyncratic to the source language. Examples are prepositions such as *de* in French. The translation of the English *orange juice* is *jus **d'** orange*. In that case the (clitic) preposition *de* would be generated from the NULL word on the English side. For the HMM alignment model the NULL word is troublesome since it stands in the 0^{th} English position and thus induces unusually long jumps which have to be captured by the jump distribution of the HMM.

In this work we present a Bayesian HMM aligner that does not make use of artificial NULL words. Instead, untranslatable source words are generated from the words preceding them. This way, our proposed model only needs to account for lexically motivated alignment links. Moreover, since our model is a hierarchical Bayesian model we can bias it towards inducing sparse lexical distributions. This in turn leads to significantly better translation distributions (Mermer and Saraçlar, 2011). Moreover, we can even perform inference on the model's hyperparameters, freeing us of having to choose arbitrary prior distributions.

Existing Bayesian word aligners are often too slow to be useful in practice. We overcome this problem by designing an auxiliary variable Gibbs sampler that reduces sampling complexity by an order of magnitude. We also provide a formal proof that this sampler works correctly.

We provide several detailed experiments which show that our model performs on par with or better than standardly used alignment toolkits in terms of BLEU score.

Notation Throughout this paper we will denote random variables (RVs) by upper case Roman letters. If we want to express the probability of a specific outcome for the RV X we write $P(X = x)$. If we want to leave the value of X underspecified, we simply write $P(x)$. We abbreviate a sequence of RVs X_1 to X_n as X_1^n and a sequence of outcomes x_1 to x_n as x_1^n. We also distinguish notationally between probability mass functions (pmfs) and probability density functions (pdfs). We denote pmfs by $P(\cdot)$ and pdfs by $p(\cdot)$. Finally, we use v_F and v_E to denote the French (source) and English (target) vocabulary sizes.

2 The HMM Word Alignment Model

The HMM model of Vogel et al. (1996) defines a joint distribution over alignments and source words given a target sentence.[2] The source words are observed as part of the parallel corpus whereas the alignments are hidden. An alignment consists of as many alignment links as there are source words. Unlike the IBM models 1 and 2, which assume independence between alignment links, the HMM model assumes a first-order Markov dependency between them. This means that each alignment link depends on its predecessor.

We define random variables E ranging over the target vocabulary $\mathcal{E}$, variables F ranging over the source vocabulary $\mathcal{F}$, and variables A modelling alignment links. We use F_j for the variable associated with the source word occupying position j in the source sentence f_1^m. Consequently, A_j is the random variable for the alignment link associated with position j. This random variable ranges over word positions in the target sentence e_0^l, where 0 is a special position occupied by the hypothetical NULL word. Finally, we use e_{a_j} to denote the target word that position j is aligned to. We can then express the likelihood of the HMM model for a single sentence pair as shown in Equation 1.

$$P(f_1^m|e_0^l) = \prod_{j=1}^{m} \sum_{i=0}^{l} P(A_j = i|a_{j-1}) P(f_j|e_i) \tag{1}$$

Here, the Markovian dependency of the alignment links is captured by a distribution over *alignment jumps*. The distance between the current and the previous link can be thought of as a jump from one position to the next. The jump width is then given by $i - i'$ where i is the current value of A_j and i'

[2]The extension to a corpus of parallel sentences is trivial because independence between sentence pairs is assumed. To keep the notation simple we describe all models on the sentence level.

is the value of the previous alignment link A_{j-1}. The distribution over alignment jumps is then simply $P(A_j = i|A_{j-1} = i') = P(i - i')$. In practice, we set a_0 to a special start token, so as to provide a conditioning event for the first alignment decision.

A severe problem for the HMM alignment model is the handling of NULL words. If we did not treat alignments to the NULL position in a special way, those alignments would distort the alignment distribution because they induce unusually long jumps. To solve this problem, we introduce a special jump value for NULL alignments.[3] This has the effect that jumps to and from NULL are modelled explicitly and behave just like other jump values. On the other hand, if the preceding alignment is an alignment to NULL, the distribution over jumps becomes uniform. This is because when jumping from NULL a special jump value gets invoked that does not depend on the position that the next jump leads to. This behaviour obviously restricts the expressive power of the model somewhat but provides a clean handling of NULL alignments.

The parameters of the standard HMM of Vogel et al. (1996) are estimated through likelihood maximization. Here, we extend their model with prior distributions over parameters. This means that we are turning it into a Bayesian model. The parameters of the alignment HMM are categorical parameters of the following distributions:

- A distribution over the source lexicon for each target word. We call this the translation distribution and use Θ_e as a variable over the corresponding parameter vector.
- A distribution over alignment jumps. We call this the alignment distribution and use Θ_a as a variable over the corresponding parameter vector.

Since the model's distributions are categorical, we impose (symmetric) Dirichlet priors on their parameters. The Dirichlet is a standard choice in this case as it is conjugate to the categorical.[4] The variables in the Bayesian HMM aligner are thus generated as follows:

$$A_j|a_{j-1} \sim \theta_a \qquad \Theta_a \sim \mathrm{Dir}(\alpha)$$
$$F_j|e_{a_j} \sim \theta_{e_{a_j}} \qquad \Theta_e \sim \mathrm{Dir}(\beta)$$

We describe how to do inference in this model in Section 4.

3 An HMM Aligner with a Language Model Component

Our model is a mixture between a Bayesian HMM aligner and a Bayesian source language model. It differs from the Bayesian HMM aligner of Section 2 in that the language model component is the one responsible for generating source words from (source) context when those do not have a target translation. These are the words that would otherwise be aligned to NULL under the IBM models and our own Bayesian HMM (Section 2).

Formally, we extend the Bayesian HMM alignment model with binary choice variables Z which we are used to indicate collocations. There is one such variable for each source position, thus Z_j is the choice variable for position j. We use this variable as an indicator for whether the language model is used. Thus, if $Z_j = 1$, the source word f_j is generated from f_{j-1}. Otherwise, if $Z_j = 0$, f_j is generated from the target word it is aligned to (e_{a_j}).

Since our model is Bayesian, we put priors on all parameters. In particular, the translation parameters (θ_e), language model parameters (θ_f) and jump parameters (θ_a) are drawn from Dirichlet distributions with parameter vectors α, β and γ. The binary choice variables Z only have one parameter q_f which depends on the previous source word and follows a Beta distribution with parameters s and r. Since it is hard to guess reasonable values for the parameters s and r, we further assume that they are independently drawn from a Gamma distribution whose shape and rate parameters we set to 1. This extra level of hierarchy frees us from having to choose arbitrary values for s and r. Empirically, it also improves the performance of our model.

[3] Conceptually, we add a categorical event TONULL and another FROMNULL to the distribution over alignment jumps.

[4] Conjugacy in this case means that the posterior over parameters will again be a Dirichlet. This has the advantage that we can analytically integrate with respect to the posterior.

To better understand our model, we formulate a generative story:

- Draw values for s and r independently from $\mathrm{Gamma}(1, 1)$
- Generate q_f from $\mathrm{Beta}(s, r)$ for each source word f
- Draw values for θ_a, θ_e and θ_f from their respective Dirichlet priors
- For each source position j
  1. Generate an alignment link a_j conditional on a_{j-1}
  2. Choose a value for Z_j conditional on the previous source word f_{j-1}
     (a) If $Z_j = 0$, generate source word f_j from target word e_{a_j}
     (b) If $Z_j = 1$, generate source word f_j from source word f_{j-1}

In terms of variable generation we have to adjust the model description from Section 2. For reasons of clarity, we condition all variables only on those events that they depend on.

$$
\begin{aligned}
S &\sim \mathrm{Gamma}(1,1) & R &\sim \mathrm{Gamma}(1,1) \\
Z_j|f_{j-1} &\sim \mathrm{Bernoulli}(q_{f_{j-1}}) & Q_f &\sim \mathrm{Beta}(s,r) \\
A_j|a_{j-1} &\sim \mathrm{Cat}(\theta_a) & \Theta_a &\sim \mathrm{Dir}(\alpha) \\
F_j|a_j, Z_j = 0, e_{a_j} &\sim \mathrm{Cat}(\theta_{e_{a_j}}) & \Theta_e &\sim \mathrm{Dir}(\beta) \\
F_j|a_j, Z_j = 1, f_{j-1} &\sim \mathrm{Cat}(\theta_{f_{j-1}}) & \Theta_f &\sim \mathrm{Dir}(\gamma)
\end{aligned}
$$

Our model as described above defines a joint distribution over French words, collocation and alignment variables and parameters given an English sentence and the hyperparameters.

4 Inference

In this section we derive a Gibbs sampler for our collocation-based model (Section 3). The sampler for the Bayesian HMM with NULL words (Section 2) follows from that. We also describe how to sample the hyperparameters of the Beta prior on the collocation distributions.

4.1 Dirichlet Predictive Posterior

Let us first establish a general useful fact about the Dirichlet distribution.[5] Let us call the posterior Dirichlet parameter vector η.[6] Then the posterior predictive distribution for a categorical outcome x is given by the following integral:

$$P(x|\eta) = \int P(x|\theta)p(\theta|\eta)d\theta = \int \theta_x p(\theta|\eta)d\theta \; . \tag{2}$$

Here, we use θ_x to denote the categorical parameter for outcome x. Note that the last integral in Equation (2) is in fact equal to the expectation of θ_x under $\mathrm{Dir}(\eta)$. It is well known that for a k-dimensional Dirichlet distribution this expectation is $\mathbb{E}[\theta_x|\eta] = \frac{\eta_x}{\sum_{i=1}^{i=k} \eta_i}$. Thus, the predictive posteriors for the alignment and collocation variables after integrating over the categorical or Beta parameters take the form of this expectation.

4.2 Gibbs Sampling the Hidden Variables

Since we are only interested in the assignments of the alignment and collocation variables, we integrate over the model parameters θ_e, θ_a, θ_f and q_f. This gives us a collapsed Gibbs sampler.

In general, a Gibbs sampler resamples one variable at a time while conditioning on the current assignments of all other variables. In our case we are interested in resampling A_j and Z_j. This means that we

[5] This fact immediately carries over to the Beta distribution which can be viewed as a 2-dimensional Dirichlet distribution.

[6] Since the categorical is in the exponential family and the Dirichlet is conjugate to it, the posterior Dirichlet parameter η is simply the sum of the prior Dirichlet parameter (which we will call ϵ here) and the sufficient statistics of the categorical likelihood. In our concrete case, these sufficient statistics are simply the number of times an outcome has been observed in the data. Let us denote these counts by $c(x)$ for an outcome x. Thus, for this outcome x, we have $\eta_x = \epsilon_x + c(x)$.

will need to derive posterior predictive distributions for these variables. Let us first deal with the posterior predictive for the collocation variables where we assume the Beta parameters r and s as given. To ease notation we introduce the set $\mathcal{C} = \{r, s, \alpha, \beta, \gamma\}$ which contains all remaining (hyper-)parameters. We use $x_{\neg j}$ to denote the set of all outcomes x except the j^{th} one. We also introduce the function $c(\cdot)$ as a (conditional) count function for an outcome across the entire corpus, excluding the j^{th} such outcome if necessary. Proportionality in the following equations comes about because we eliminate normalization constants that do not depend on the value of the variable that we sample.

$$\begin{aligned} P(Z_j = 0 | z_{\neg j}, a_1^m, f_{\neg j}, e_1^l, \mathcal{C}) &\propto P(Z_j = 0 | f_{j-1}, s, r) \times P(f_j | a_j, e_{a_j}, \beta) \\ &= \frac{c(z = 0 | f_{j-1}) + r}{c(f_{j-1}) + r + s} \times \frac{c(f_j | e_{a_j}) + \beta}{c(e_{a_j}) + v_F \beta} \\ &\propto (c(z = 0 | f_{j-1}) + r) \times \frac{c(f_j | e_{a_j}) + \beta}{c(e_{a_j}) + v_F \beta} \end{aligned} \tag{3}$$

$$\begin{aligned} P(Z_j = 1 | z_{\neg j}, a_1^m, f_{\neg j}, e_1^l, \mathcal{C}) &\propto P(Z_j = 1 | f_{j-1}, s, r) \times P(f_j | f_{j-1}, \gamma) \\ &= \frac{c(z = 1 | f_{j-1}) + s}{c(f_{j-1}) + r + s} \times \frac{c(f_j | f_{j-1}) + \gamma}{c(z = 1 | f_{j-1}) + v_F \gamma} \\ &\propto (c(z = 1 | f_{j-1}) + s) \times \frac{c(f_j | f_{j-1}) + \gamma}{c(z = 1 | f_{j-1}) + v_F \gamma} \end{aligned} \tag{4}$$

When resampling the alignment variables, we need to distinguish between the two cases where the collocation variable is active or not. In case the collocation variable is switched off, inference for the alignment variable is similar to that in the Bayesian HMM aligner. The crucial difference, however, is that in the standard case, the values of the alignment variable range from 0, the NULL position, to the target sentence length l. In our model, however, the possible values for the alignment variable start at 1 as there is no NULL position.

$$\begin{aligned} P(a_j | Z_j = 0, a_{\neg j}, f_1^m, e_1^l, \mathcal{C}) &\propto P(a_j | a_{j-1}, \alpha) \times P(a_{j+1} | a_j, \alpha) \times P(f_j | e_{a_j}, \beta) \\ = \frac{c(a_j - a_{j-1}) + \alpha}{\sum_{i=1}^{l} (c(i - a_{j-1}) + \alpha)} &\times \frac{c(a_{j+1} - a_j) + \alpha}{\sum_{i=1}^{l} (c(a_{j+1} - i) + \alpha)} \times \frac{c(f_j | e_{a_j}) + \beta}{c(e_{a_j}) + v_F \beta} \\ &\propto (c(a_j - a_{j-1}) + \alpha) \times (c(a_{j+1} - a_j) + \alpha) \times \frac{c(f_j | e_{a_j}) + \beta}{c(e_{a_j}) + v_F \beta} \end{aligned} \tag{5}$$

Again, let us point out that the predictive posterior in (5) is the one we use for inference in the Bayesian HMM model described in Section 2 with difference that the alignment positions include 0 in that case.

If the collocation variable is switched on, i.e. if the j^{th} French word is generated from its predecessor, there is no lexical influence on the alignment posterior and the new link is simply sampled from the alignment distribution.

$$\begin{aligned} P(a_j | Z_j = 1, a_{\neg j}, f_{\neg j}, e_1^l, \mathcal{C}) &\propto P(a_j | a_{j-1}, \alpha) \times P(a_{j+1} | a_j, \alpha) \\ = \frac{c(a_j - a_{j-1}) + \alpha}{\sum_{i=1}^{l} (c(i - a_{j-1}) + \alpha)} &\times \frac{c(a_{j+1} - a_j) + \alpha}{\sum_{i=1}^{l} (c(a_{j+1} - i) + \alpha)} \\ &\propto (c(a_j - a_{j-1}) + \alpha) \times (c(a_{j+1} - a_j) + \alpha) \end{aligned} \tag{6}$$

At this point it is important to note that the naïve Gibbs sampler described here considers all target positions as candidate alignment points for a given source position j. This means that the time it takes to re-sample an alignment link grows linearly with the length of the target sentence. In practice this makes the sampler unpractically slow. We address this problem in the following section.

4.3 Sampling Alignments Efficiently

We are interested in sampling from $P(a_j|a_{\neg j}, \mathcal{C})$, where we conflate all other variables in $\mathcal{C}$ to avoid clutter. Note that we know $P(a_j|a_{\neg j}, \mathcal{C})$ up to a normalisation constant, thus sampling requires evaluating Equation 5 (or 6) for all $1 \leq i \leq l$. Naturally, this procedure runs in time proportional to $O(l)$. In this section we present an auxiliary variable sampler that brings this down to constant time.

The idea is to evaluate $P(a_j|a_{\neg j}, \mathcal{C})$ only for assignments in a subset of target positions.[7] This subset must include at least 2 candidates and it must be such that it contains the current assignment of the variable we are resampling. That is, if we let $\left(a_{\neg j}^{(t)}, a_j^{(t)}\right)$ denote the current state of the Markov chain, then we need to select $a_j^{(t)}$ as well as at least one random candidate from the remaining available positions $\{1, \ldots, l\} \setminus \{a_j^{(t)}\}$. We denote a selection of target positions by a vector $k \subseteq \{0,1\}^l$ such that $k_i = 1$ signifies that i is a reachable candidate. Then, once a selection is made, we sample the next state of the Markov chain, i.e. $\left(a_{\neg j}^{(t)}, a_j^{(t+1)}\right)$, from a distribution proportional to $P(A_j^{(t+1)} = i|a_{\neg j}^{(t)}, \mathcal{C}^{(t)}) \times k_i$. Note that, under this distribution, only selected positions have non-zero probability. This makes sampling the next state run in constant time independent of the target sentence length.

Formally, this is a case of sampling by data augmentation (Tanner and Wong, 1987). Let K be a random variable taking values in $\mathcal{K} \subset \{0,1\}^l$. We interpret an assignment of K as a random selection of at most l target positions. We define the joint distribution $P(a_j, k|a_{\neg j}, \mathcal{C}) = P(a_j|a_{\neg j}, \mathcal{C}) \times P(k|a_j, a_{\neg j})$, where we take the conditional $P(k|A_j = i, a_{\neg j}) = \frac{k_i}{\sum_{k' \in \mathcal{K}} k'_i}$ to distribute uniformly over all selections in $\mathcal{K}$ that contain i. This guarantees that the current state of the Markov chain is part of the selection, a condition necessary for irreducibility. Then, the conditional $P(A_j = i'|k, a_{\neg j}, \mathcal{C})$ follows directly:

$$P(A_j = i'|k, a_{\neg j}, \mathcal{C}) \propto P(A_j = i'|a_{\neg j}, \mathcal{C}) \times P(k|A_j = i', a_{\neg j}) \propto P(A_j = i'|a_{\neg j}, \mathcal{C}) \times k_{i'} \quad (7)$$

Claim If $\mathcal{K}$ includes at least all subsets of size 2 where one of the elements is the previous state of the Markov chain, then the transition kernel $\kappa(i'|i) = \sum_{k \in \mathcal{K}} P(i'|k) \times P(k|i)$ is strictly positive for every $i, i' \in \{1, \ldots, l\}$ and, therefore, the resulting Markov chain is Harris ergodic.

Proof.

$$\kappa(i'|i) = \sum_{k \in \mathcal{K}} P(i'|k) \times P(k|i) = \sum_{k \in \mathcal{K}} \frac{P(i') \times P(k|i')}{P(k)} \times P(k|i) = P(i') \sum_{k \in \mathcal{K}} \frac{\frac{k_{i'}}{\sum_{k' \in \mathcal{K}} k'_{i'}} \times \frac{k_i}{\sum_{k' \in \mathcal{K}} k'_i}}{P(k)} \quad (8)$$

In the last term of Equation 8, $P(i')$ is strictly positive because $i' \in \{1, \ldots, l\}$ by construction, and the sum is strictly positive when $\mathcal{K}$ includes at least one subset containing both i and i'. This is why we can start the candidate set with $\{a_j^{(t)}\}$ and enlarge it by sampling uniformly from $\{1, \ldots, l\} \setminus \{a_j^{(t)}\}$ without replacement. We need to do it at least once, but we can also repeat it a fixed number of times. □

The complete algorithm consists of 2 simulations, $K^{(t+1)} \sim P(\cdot|a_j^{(t)}, a_{\neg j}^{(t)})$ and $A_j^{(t+1)} \sim P(\cdot|k^{(t+1)}, a_{\neg j}^{(t)}, \mathcal{C}^{(t)})$, each feasible in isolation. With this improved sampler we can resample alignment links in constant time whereas the naïve sampler would require time linear in target sentence length. It is easy to see that to resample all alignment links in a source sentence with m words, the naïve sampler would take time $O(l \times m) \approx O(l^2)$, whereas our improved auxiliary variable sampler does the same in $O(l)$. This improvement makes our model competitive with maximum-likelihood models.

4.4 Sampling of the Beta Parameters

The parameters s and r of the Beta distribution which serves as a prior on the decision variables are random variables in our model. Since there is no easily computable conjugate distribution for these parameters, we can not integrate them out analytically. Instead, we choose to approximate the integral

[7] Recall that l is the target sentence length.

through repeated sampling of these variables. Since both these variables take on values in the positive reals, we impose a Gamma prior on them as described in Section 3.

There are several ways of sampling variables in non-conjugate Bayesian models. Here we use slice sampling (Neal, 2003) as it is fast and easy to implement. The idea of slice sampling is that we augment our sampling distribution with an auxiliary variable U such that the marginal distribution of S (or R) stays unchanged.[8] Simulation then follows by Gibbs sampling, whereby we sample U conditioned on S, and S conditioned on U in turn. It can be shown that, with conditionals as shown in (9), the transition kernel underlying this Gibbs sampling procedure is Harris ergodic (Neal, 2003). Thus, the procedure is not only correct but also very efficient as we only sample from uniform distributions.

$$p(u|s) = \frac{\mathbb{1}(u < p(s))}{p(s)} \qquad p(s|u) \propto \begin{cases} 1 & \text{if } p(s) \geq u \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The posterior that we slice sample from is simply proportional to the product of likelihood and prior of the Beta distribution: $p(s|f_1^m, z_1^m, r) \propto P(z_1^m|f_1^m, s, r) \times p(s)$.

Notice that the conditions in Equation (9) guarantee that at least the current point will be in the slice. We will therefore always be able to obtain a new sample. Intuitively, slice sampling works because the marginal distribution $p(s)$ stays unchanged.

4.5 Decoding

After we have taken a number of samples, we are ready to decode. We use a version of maximum marginal decoding (Johnson and Goldwater, 2009) in which we assign to each source position j the target position that was most often sampled as a value for A_j. If most of the time the collocation variable Z_j was active, however, we leave that source position unaligned.

5 Experiments and Results

All experiments were run using the Moses phrase-based system (Koehn et al., 2007) with lexicalized reordering. In order to speed up our experiments we used cube pruning with a pop limit of 1000 in both tuning and evaluation. Symmetrised alignments were obtained with the `grow-diag-final-and` heuristic.

Data We used the WMT 2014 news commentary data[9] to train our models and the corresponding dev (`newstest2013`) and test (`newstest2014`) sets for tuning and evaluation. We use all available monolingual data to train 5-gram language models with KenLM (Heafield, 2011).

Models We report results for the Bayesian HMM described in Section 2 (BHMM) and our collocation-based model described in Section 3 (BHMM-Z). To enable comparison with standard alignment toolkits, we also report results with Giza++ and fastAlign (Dyer et al., 2013). Finally, we make a comparison with the collocation-based IBM2 model of Schulz et al. (2016) (BIBM2-Z).

Hyperparameters The hyperparameters of our model were set to $\beta = \gamma = 0.0001$ to obtain sparse lexical distributions. For the jump prior we chose $\alpha = 1$, not giving preference to any particular distribution. The same choices apply for the BIBM2-Z. However, that model does not employ hyperparameter inference and we therefore set $s = 1, r = 0.01$. Finally, the BHMM uses the same parameters as the previous two models, except those associated with the collocation variable.

All samplers were run for 1000 iterations without burn-in and samples were taken after each 25^{th} iteration. The initial state was chosen to be the Viterbi decoding of IBM model 1. For our auxiliary variable sampler we select exactly 2 candidates, thus making inference as fast as possible. Giza++ and fastAlign were run under their standard parameter settings. For Giza++ this means that we ran EM for IBM model 1 and the HMM (5 iterations each) and for IBM models 3 and 4 (3 iterations each). Since

[8] In the following exposition we will describe the resampling of S and assume a fixed value r. Resampling R works analogously with a fixed value s.

[9] `http://statmt.org/wmt14/translation-task.html`

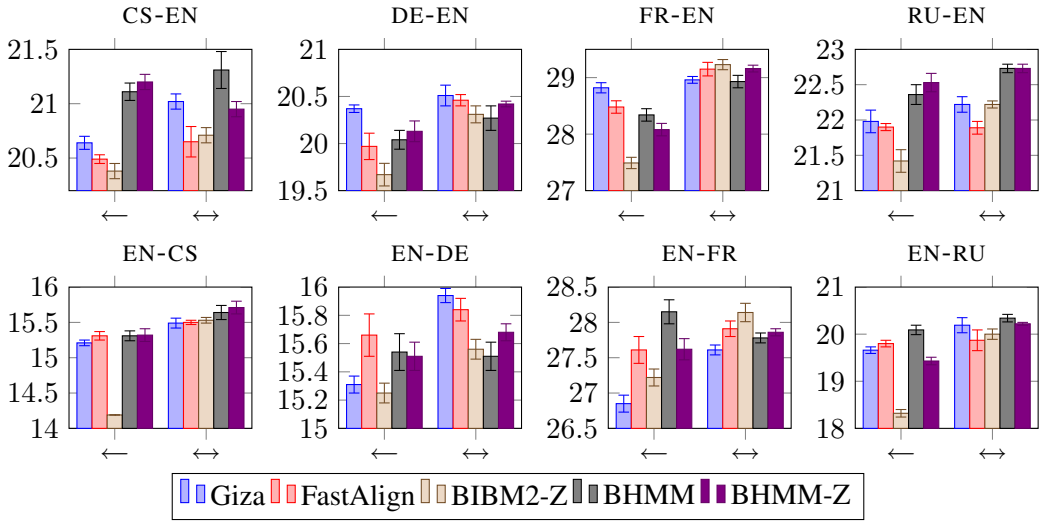


Figure 1: BLEU scores for each translation direction trained on (←) directional (condition on target and generate source) and (↔) symmetrised alignments (`grow-diag-final-and`). Observe that the plots are on different scales. This means that results cannot directly be compared across plots.

the maximum likelihood HMM is incorporated in the Giza++ pipeline, we do not report separate results for it.

Results Figure 1 shows translation quality results in terms of BLEU for different aligners, where BHMM is the Bayesian HMM (Section 2) and BHMM-Z is our novel collocation-based model (Section 3). Furthermore, BIBM2-Z is the model of Schulz et al. (2016). To account for optimiser instability (MERT in this case), we plot average and standard deviation across 5 independent runs. Note that our Bayesian models perform mostly on par with maximum-likelihood models. In the directional case, our Bayesian models lose to Giza++ only in DE-EN by a very slim margin. Except for EN-DE (symmetrised) our models are never worse than fastAlign. Moreover, our models perform notably well on Czech and Russian outperforming the maximum-likelihood models. Finally, except for EN-FR (directional), our collocation-based BHMM-Z improves upon the more basic BHMM. It also often improves upon the BIBM2-Z. That model is only better on the symmetrised English-French data where it outperforms all other models.

We also analysed the number of alignment links that our systems set. It is noteworthy that the Bayesian HMM with NULL words consistently sets much fewer links than all other systems. Taking BHMM as baseline other models set additionally (on average across languages and translation direction) 39.5% (our collocation-based model), 39.2% (Giza++), and 36.2% (fastAlign) more links. Setting more links constrains phrase extraction heuristics more and leads to smaller phrase tables. Empirically, the phrase tables of the collocation-based model are roughly three times smaller than those of the Bayesian HMM. Thus, the collocation-based system is at an advantage here.

Finally, in terms of speed, Giza++ takes on average (across languages and translation directions) 202 minutes on 2 CPUs, while BHMM and BHMM-Z take respectively 81 and 267 minutes in 1 CPU.[10] The slower performance of BHMM-Z in relation to BHMM is not per se due to the sampling of collocation variables, but rather due to hyperparameter inference (see Section 4.4).

6 Related Work

There are several extensions of the classical IBM models. Dyer et al. (2013) use a log-linear model for the distortion distribution. While they keep the independence assumptions for alignment links, they bias

[10]The run times for our models include the computation of the initial state of the sampler with IBM model 1. The run time of the sampler itself is thus slightly lower than the reported times. Also notice that due to its highly optimised posterior computations, fastAlign finishes in under 10 minutes on average.

their model to preferably align positions which are close to each other. Using standard results for series, they manage to make the posterior computations in their model extremely fast.

Another interesting extension of the HMM alignment is presented in Zhao and Gildea (2010) who added a fertility distribution to the HMM. This made posterior computations in their model intractable, however, they avoided the use of heuristics and instead approximated the posterior using MCMC.

The idea of using Bayesian inference together with a Gibbs sampler for word alignment was first presented for IBM model 1 by Mermer and Saraçlar (2011). They also gave a more detailed analysis of their method and extended it to IBM model 2 in Mermer et al. (2013). The model presented here is also similar in spirit to Schulz et al. (2016) who proposed to use a language model component together with an alignment model. However, they used IBM models 1 and 2 as alignment components.

Apart from the present work, the only other work on Bayesian word alignment that we know of that performed as well as or better than Giza++ was presented in Gal and Blunsom (2013). These authors reformulated IBM models 1-4 as hierarchical Pitman-Yor processes. While their models are highly expressive, the Gibbs sampler based on Chinese Restaurant processes that they used is very slow and thus their models are unfortunately not useful in practice.

7 Discussion and Future Work

We envision several useful extensions of our model for the future. Firstly, we plan to turn the language model distribution into a hierarchical distribution. We plan to use either a hierarchical Dirichlet process or Pitman-Yor process for this. The advantage of this technique is that information about untranslatable source words can be shared across preceding source words. Sampling from such a distribution using a Chinese Restaurant Process is potentially time-consuming. However, we are confident that we can maintain a good speed if we apply our auxiliary variable technique and employ efficient samplers as described by Blei and Frazier (2011).

Since our model aligns many words and thus restricts the amount of phrases that can be extracted, it may also be a good alignment model for hierarchical phrase-based SMT. We plan to apply our model to this scenario in the future.

The code used in our experiments is freely available at `https://github.com/philschulz/Aligner`.

Acknowledgements

We would like to thank Stella Frank for helpful discussions about hyperparameter inference. We also thank our reviewers for their excellent feedback. This work was supported by the Dutch Organization for Scientific Research (NWO) VICI Grant nr. 277-89-002.

References

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171.

David M. Blei and Peter I. Frazier. 2011. Distance dependent Chinese Restaurant Processes. *Journal of Machine Learning Research*, 12:2461–2488.

Peter F. Brown, Vincent J.Della Pietra, Stephen A. Della Pietra, and Robert. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia, June. Association for Computational Linguistics.

Yarin Gal and Phil Blunsom. 2013. A systematic bayesian treatment of the IBM alignment models. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 969–977.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 187–197, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mark Johnson and Sharon Goldwater. 2009. Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325, Boulder, Colorado, June. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 104–111, New York City, USA, June. Association for Computational Linguistics.

Coşkun Mermer and Murat Saraçlar. 2011. Bayesian word alignment for statistical machine translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, pages 182–187.

Coşkun Mermer, Murat Saraçlar, and Ruhi Sarikaya. 2013. Improving statistical machine translation using Bayesian word alignment and Gibbs sampling. *IEEE Transactions on Audio, Speech & Language Processing*, 21(5):1090–1101.

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

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Philip Schulz, Wilker Aziz, and Sima'an Khalil. 2016. Word alignment without NULL words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Martin A. Tanner and Wing Hung Wong. 1987. The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 82(398):528–540, June.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, pages 836–841, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shaojun Zhao and Daniel Gildea. 2010. A fast fertility hidden Markov model for word alignment using MCMC. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 596–605, Cambridge, MA, October. Association for Computational Linguistics.

Learning to translate from graded and negative relevance information

Laura Jehl
Computational Linguistics
Heidelberg University
69120 Heidelberg, Germany
jehl@cl.uni-heidelberg.de

Stefan Riezler
Computational Linguistics & IWR
Heidelberg University
69120 Heidelberg, Germany
riezler@cl.uni-heidelberg.de

Abstract

We present an approach for learning to translate by exploiting cross-lingual link structure in multilingual document collections. We propose a new learning objective based on structured ramp loss, which learns from graded relevance, explicitly including negative relevance information. Our results on English-German translation of Wikipedia entries show small, but significant, improvements of our method over an unadapted baseline, even when only a weak relevance signal is used. We also compare our method to monolingual language model adaptation and automatic pseudo-parallel data extraction and find small improvements even over these strong baselines.

1 Introduction

Typically, parameters of an SMT system are learned on a small parallel data set from the domain or genre of interest. However, while many multilingual data sets, especially in the realm of user-generated data, contain document-level links, sentence-parallel training data are not always available. A small number of sentences can be manually translated for in-domain parameter tuning, but this ignores most of the available multilingual resource. Monolingual language model adaptation via concatenation or interpolation is one viable solution which makes use of the target side part of a collection (see e.g. Koehn and Schroeder (2007) or Foster and Kuhn (2007)). Additionally, there are several approaches to automatic parallel data extraction from cross-lingual document-level links, such as Munteanu and Marcu (2005)'s work on news data, or more recent work on Wikipedia by Wołk and Marasek (2015), and on websites by Smith et al. (2013). We argue that these approaches work well if the cross-lingual links are a strong signal for parallelism, but fail if the signal linking documents across languages is weaker. We propose a method for tuning sparse lexicalized features on large amounts of multilingual data which contain some cross-lingual document-level relevance annotation. We do so by re-formulating the structured ramp loss objective proposed by Chiang (2012) and Gimpel and Smith (2012) to incorporate graded and negative cross-lingual relevance signals. Using translation of Wikipedia entries as a running example, we evaluate the efficacy of our method along with the traditional approaches on a manually created in-domain test set. We show that our method is able to produce small, but significant, gains, even if only a weak relevance signal is used.

Section 2 explains our learning objective and cost function. In Section 3 we describe the construction of our training and evaluation data, including pseudo-parallel data extraction. Section 4 contains details of our experimental setup and presents our experimental results. Section 5 concludes the paper.

2 Learning from graded relevance feedback

2.1 Learning objectives

We work within a scenario where we want to learn the parameters of an SMT system, but have no in-domain reference translations available. What we have, is a large collection of source and target language documents and a signal telling us that some target documents are *more relevant* to a source document

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3156–3166, Osaka, Japan, December 11-17 2016.

than others. For example, in the multilingual Wikipedia, cross-lingual documents can be connected in different ways. First, a link exists between two documents if they are connected by an interlanguage link (we call this a *mate* relation). This is a very strong relevance signal. Second, a more indirect link exists between a source language document and a target language document if the target language document is connected to the source language document's mate by a hyperlink (we call this a *link* relation). This is a weaker relevance signal. A cross-lingual mate is more relevant to an input document than a document that is only linked to by the mate. In turn, this linked document is more relevant to the input than a document that has no direct link to the mate. Any Wikipedia document is more relevant than a document from a different data set. We write relevance as $d_1 \succ_{\mathbf{f}} d_2$ ("d_1 is more relevant to $\mathbf{f}$ than d_2"). Another example of graded relevance information, which has been used in information retrieval, occurs in multilingual patent collections, where patent documents can be in a "family" relation if they contain publications of the same patent, or be related to a lesser degree, if a target document is cited by a source document's family patent. Of course, other notions of relevance are conceivable, e.g. by document similarity, and we plan to further investigate such notions in future work.

In order to incorporate graded relevance information, we modify the structured ramp loss objective by Gimpel and Smith (2012) to also include negative relevance information. Ramp loss based SMT tuning methods as presented by Gimpel and Smith (2012) and Chiang (2012) usually try to find parameters that separate a "good" hypothesis with respect to the reference from one that is "bad" with respect to the same reference. Goodness and badness are measured by an external cost function, or a cost function combined with the model prediction. Equation 1 shows one version of the structured ramp loss ("ramp loss 3"/equation 8 from Gimpel and Smith (2012)):

$$L_{Gimpel}(\mathbf{F};\theta) = \sum_{\mathbf{f}\in\mathbf{F}} \underbrace{-\max_{\mathbf{e}}(\text{score}(\mathbf{e};\theta) - \text{cost}(\mathbf{e}))}_{\text{hope derivation}} + \underbrace{\max_{\mathbf{e}}(\text{score}(\mathbf{e};\theta) + \text{cost}(\mathbf{e}))}_{\text{fear derivation}} \quad (1)$$

where $\mathbf{F} = \{\mathbf{f}_1, \mathbf{f}_2 \ldots \mathbf{f}_n\}$ is a finite set of input examples, θ refers to the model parameters and $\mathbf{e}$ is a translation hypothesis; $\text{score}(\mathbf{e};\theta)$ is the log-linear model score of the hypothesis, which is proportional to the dot product between the feature vector associated with the hypothesis and the weight vector; $\text{cost}(\mathbf{e})$ is a cost function, which measures the quality of the current hypothesis. Usually, this function is some per-sentence approximation of the BLEU score against one or more reference translations. Following Chiang (2012)'s terminology, this loss tries to maximize the distance between a *hope* derivation – which has high model score and low cost – from a *fear* derivation – which has high model score, but high cost.

We define two training objectives which are variations of this loss function, but which incorporate positive and negative relevance information. Our intuition is that instead of trying to separate *hope* and *fear* with respect to the same reference, we try to separate a hypothesis that has high model score and low cost with respect to a relevant document from one that has high model score and low cost with respect to a document that is irrelevant. Our first objective is given in Equation 2:

$$L_{ramp_1}(\mathbf{F};\theta) = \sum_{\mathbf{f}\in\mathbf{F}} \underbrace{-\max_{\mathbf{e}}(\text{score}(\mathbf{e},\mathbf{f};\theta) - \text{cost}(\mathbf{e},d^+_{\mathbf{f}}))}_{\text{hope derivation w.r.t. } d^+} + \underbrace{\max_{\mathbf{e}}(\text{score}(\mathbf{e},\mathbf{f};\theta) - \text{cost}(\mathbf{e},d^-_{\mathbf{f}}))}_{\text{hope derivation w.r.t. } d^-} \quad (2)$$

In this objective, $\text{cost}(\mathbf{e},d) \in [0,1]$ is the cost of a hypothesis $\mathbf{e}$ with respect to a document d. d^+ and d^- are documents, such that $d^+ \succ_{\mathbf{f}} d^-$. Unlike L_{Gimpel}, this loss tries to separate two different hope derivations. One potential weakness of L_{ramp_1} is that it treats d^+ and d^- completely independently. This could lead to very similar hypotheses being selected, if there exist hypotheses that have low cost in both d^+ and in d^-. To solve this issue, we propose a second modification of the loss.

In the second variant we define "good" and "bad" hypotheses as those which have the largest *difference* between the cost with respect to d^+ and d^-, i.e. hypotheses that best distinguish d^+ from d^-. This leads to the following objective given in Equation 3.

$$L_{ramp_2}(\mathbf{F};\theta) = \sum_{\mathbf{f}\in\mathbf{F}} \underbrace{-\max_{\mathbf{e}}(\text{score}(\mathbf{e},\mathbf{f};\theta) - (\text{cost\_diff}(\mathbf{e},d^+_{\mathbf{f}},d^-_{\mathbf{f}}))}_{\text{derivation with lowest cost}(d^+)\textit{ and }\text{highest cost}(d^-)} + \underbrace{\max_{\mathbf{e}}(\text{score}(\mathbf{e},\mathbf{f};\theta) - (\text{cost\_diff}(\mathbf{e},d^-_{\mathbf{f}},d^+_{\mathbf{f}}))}_{\text{derivation with lowest cost}(d^-)\textit{ and }\text{highest cost}(d^+)} \quad (3)$$

where cost_diff is defined as

$$\text{cost\_diff}(\mathbf{e},d_1,d_2) = \text{cost}(\mathbf{e},d_1) - \text{cost}(\mathbf{e},d_2) \quad (4)$$

Note that with the above definition of cost_diff, equation 3 can be reformulated as

$$L_{ramp_2}(\mathbf{F};\theta) = \sum_{\mathbf{f}\in\mathbf{F}} \underbrace{-\max_{\mathbf{e}}(\text{score}(\mathbf{e},\mathbf{f};\theta) - \text{cost\_diff}(\mathbf{e},d^+_{\mathbf{f}},d^-_{\mathbf{f}}))}_{\text{hope derivation}} + \underbrace{\max_{\mathbf{e}}(\text{score}(\mathbf{e},\mathbf{f};\theta) + \text{cost\_diff}(\mathbf{e},d^+_{\mathbf{f}},d^-_{\mathbf{f}}))}_{\text{fear derivation}} \quad (5)$$

which is identical to the original structured ramp loss (Equation 1), but still allows to include positive and negative relevance signals via the cost function. We apply a linear scaling operation to squash our new cost function to return values between 0 and 1.

2.2 Implementation and learning

Parallelized stochastic subgradient descent. Algorithm 1 shows our learning procedure. Optimization is done using stochastic subgradient descent (SSD) as proposed for ramp loss by Keshet and McAllester (2011). In order to be able to train on thousands of documents, we use the method described in Algorithm 4 of Simianer et al. (2012), which splits training data into shards (line 1 in Algorithm 1), trains one epoch on each shard (line 3 to 12), and then applies feature selection by ℓ_1/ℓ_2 regularization (line 13) before starting the next epoch.

Sampling. For each training example, we first sample a document pair (d^+, d^-) (line 6). The *sample()* procedure draws documents d^+ from a set of relevant documents D^+ and d^- from a set of "contrast documents", D^-, according to some cross-lingual relevance signal. In our experiments, we first use random sampling. We also try out a weighted sampling strategy, if the relevance signal is weaker. In this case, we want to sample a document more frequently from D^+, if it is more similar to the input document. We calculate cross-lingual document similarity by using document representations from bilingual word embeddings. The embeddings are learned from the aligned parallel training corpus using the Bilingual Skip-gram model of Luong et al. (2015).[1] Document representations are computed by averaging over all word representations in the document, weighted by the inverse document frequencies of the words. Cosine similarity is used to measure similarity between the current source document and the documents in D^+. We use weighted reservoir sampling (Efraimidis and Spirakis, 2006) to draw a document weighted by its similarity to the current source document. The contrast document d^- is drawn randomly from D^-, but is re-drawn if d^- is more similar to the input then d^+.

Search. In lines 7 and 8 we identify the "good" and "bad" hypotheses h^+ and h^- by running *search()*. Most tuning algorithms use k-best lists to approximate the search space over possible translation hypotheses. However, k-best lists cover only a very small portion of the possible hypothesis space and often contain very similar hypotheses. Since we may not have a strong enough signal to differentiate between those hypotheses, we also experiment with using the entire search space, which in hierarchical phrase-based translation can be represented by a packed hypothesis forest, or hypergraph. In this scenario, *search()* amounts to finding the Viterbi derivation after annotating the translation hypergraph

[1] github.com/lmthang/bivec

Algorithm 1 SSD

Require: input X, epochs T, initial weights w_0, cost function $cost$, document collection D^+, D^-, stepsize η, regularization strength C, number of shards S

1: $\{X_1 \dots X_S\} \leftarrow$ make_shards(S, X) ▷ Create shards for parallel training
2: **for** $t = 1$ to T **do**
3: **for** $s = 1$ to S parallel **do**
4: $w_{s,t-1}^{(0)} \leftarrow w_{t-1}$
5: **for** $i = 1$ to $|X_s|$ **do**
6: $(d^+, d^-) \leftarrow$ sample$(X_s^{(i)}, D^+, D^-)$ ▷ Sample relevant and irrelevant document
7: $(h^+, h^-) \leftarrow$ search$(X_s^{(i)}, w_{s,t-1}^{(i-1)}, cost, d^+, d^-)$ ▷ Find hope and fear
8: $w_{s,t-1}^{(i)} \leftarrow w_{s,t-1}^{(i-1)} + \eta(\phi(h^+) - \phi(h^-)) - \eta C(\frac{w_{s,t-1}^{(i-1)} - w_0}{|X|})$ ▷ Update weights
9: **end for**
10: $w_{s,t} \leftarrow w_{s,t-1}^{(|X|)}$
11: **end for**
12: $w_t \leftarrow$ select$(w_{1,t} \dots w_{S,t})$ ▷ Select features by ℓ_1/ℓ_2 regularization
13: **end for**

edges with the cost for each edge. This requires a cost function which decomposes over edges, as will be detailed in section 2.3. We run experiments both using a k-best lists and the full search space.

Finally, the weights are updated in line 9 by adding the negative subgradient multiplied by learning rate η and a regularization term which is obtained from adding $C\frac{1}{2|X|}\|(w - w_0)^2\|$ to the ramp loss objective.

2.3 Cost function

So far, we have not yet specified the cost function. Usually, $1 - psBLEU(\mathbf{e}, \mathbf{r})$ is used as a cost function, where $\mathbf{r}$ is a reference translation and $psBLEU$ is a per-sentence approximation of the $BLEU$ score. Since we do not have reference translations as feedback, we need to use a cost function that will evaluate the quality of a hypothesis with respect to a relevant document. Like BLEU we use average n-gram precision. Unlike BLEU, we cannot use reference length to control the length of the produced translation. Our solution is to use the source length, multiplied by the average source-target length ratio r which can be empirically determined on the training set.

For *k*-best training, where we can evaluate complete sentences, we use average *n*-gram precision:

$$\mathrm{nprec}(\mathbf{e}, \mathbf{f}, d) = \frac{1}{N}\sum_{n=1}^{N} \frac{\sum_{u_n} c_{u_n}(\mathbf{e}) \cdot \delta_{u_n}(d)}{\sum_{u_n} c_{u_n}(\mathbf{e})} \cdot min(1, \frac{r \cdot |\mathbf{e}|}{|\mathbf{f}|})$$

where N is the maximum *n*-gram size, u_n are *n*-grams present in $\mathbf{e}$, $c_{u_n}(\mathbf{e})$ counts the occurrences of u_n in $\mathbf{e}$ and $\delta_{u_n}(d)$ returns 1 if u_n is present in document d and 0 otherwise. The second term is the brevity penalty. As a cost function, this becomes $1 - \mathrm{nprec}$. BLEU uses the geometric mean to account for the exponentially decaying precision, as n increases. When calculating per-sentence BLEU, we might face the problem of zero-precision, as n increases. Since BLEU is measured over a corpus and not over a sentence, the case of zero-values was not taken into consideration. A common solution to this is count smoothing. We use the arithmetic instead of the geometric mean, since it avoids the problem of zeros, and will return the same ranking as the geometric mean.

When training on hypergraphs, we are facing the problem that *n*-gram precision is not edge-decomposable. For our hypergraph experiments, we tried the simplest possible approach, which is to compute nprec at edge level:

$$\mathrm{nprec}(\mathbf{e}, \mathbf{f}, d) = \sum_{\bar{e} \in \mathbf{e}} \mathrm{nprec}(\bar{e}, \bar{f}, d)$$

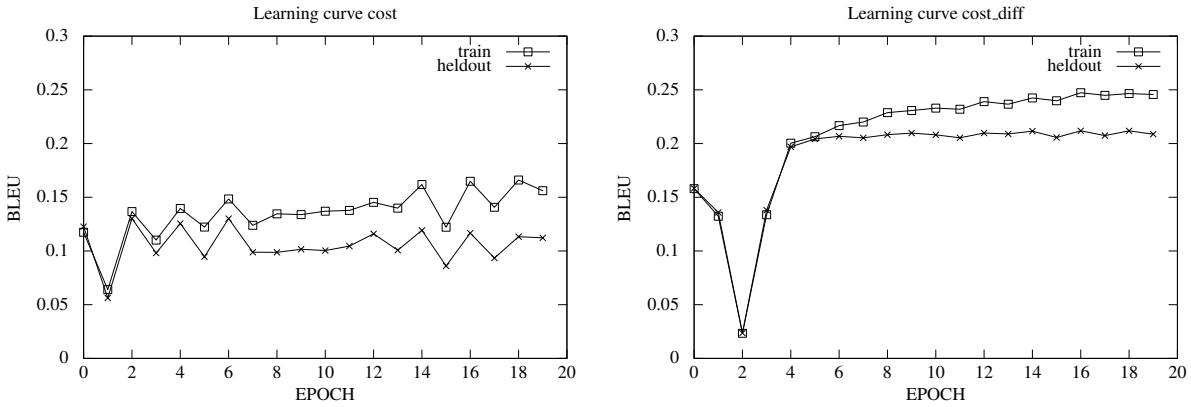


Figure 1: Learning curves on training and heldout data when training on references. The left side uses the loss from equation 2, the right side uses the loss from equation 3.

In order to test the proposed loss and cost functions, we look at how they perform in a case, where we are learning translation model weights from a perfect signal, i.e. reference translation. Instead of sampling d^+ from a set of relevant documents D^+ , we use the reference translation of input $\mathbf{f}_i$. For the contrast set d^-, we sample another sentence from the target side of the training data. We train on 500-best lists for 20 epochs. We conduct this experiment on the IWSLT evaluation data, using IWSLT tst2010 for training and tst2013 for evaluation. The model is trained on out-of-domain data in the same way as the model described in 4.1. Figure 1 shows learning curves for L_{ramp_1} which uses cost and L_{ramp_2} which uses $\mathrm{cost\_diff}$. We found $\mathrm{cost\_diff}$ to perform much better than cost on both train and heldout data. Why does cost do so much worse? Remember, that in this scenario we select two *hope*-derivations. What is more, we only select them from a small k-best list (500 translations). With the $\mathrm{cost\_diff}$ function we are required to choose hypotheses that *distinguish* most between d^+ and d^-. This will select a h^- that is far away from the reference (similar to a *fear* derivation). While with cost, there is no guarantee that h^- will differ from h^+.

3 Data preparation and extraction

3.1 Initial Wikipedia data set

Wikipedia is internally structured by cross-lingual links and inter-article links. We use the German-English WikiCLIR collection by Schamoni et al. (2014), along with their definition of cross-lingual relevance levels: A target language document has relevance level 3 if it is the cross-lingual mate of an input document. It is assigned relevance level 2, if there is a bidirectional link relation between the cross-lingual mate and the document. WikiCLIR contains a total of 225,294 mate relations with 1 average German mate per English document, and over 1.7 million bidirectional link relations, with on average 8.5 links per English document. We use the link information provided by WikiCLIR, but we work with the full Wikipedia documents rather than WikiCLIR's abbreviated queries.

3.2 Automatic sentence alignment

The cross-lingual mate relation in Wikipedia is a strong indicator for parallelism. However, Wikipedia entries in different languages are not necessarily translations of each other, but can be edited independently. In order to find parallel sentences, we use an automated extraction method. We do this for three purposes:

1. To identify nearly parallel document pairs for the construction of a clean in-domain evaluation set without having to rely on manual translation.

2. To examine whether bidirectional links provide a strong enough signal for extracting pseudo-parallel training data.

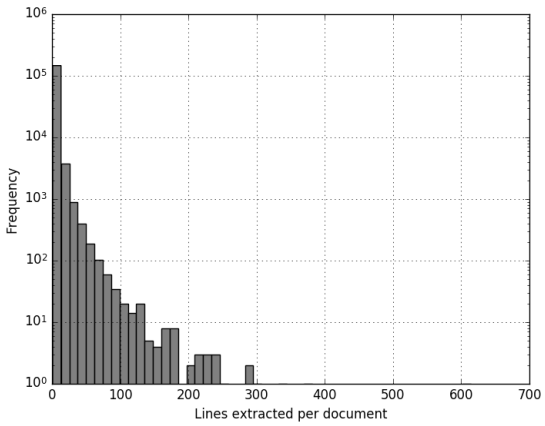


Figure 2: Number of documents (y-axis, on log-scale) from which n lines were extracted (x-axis).

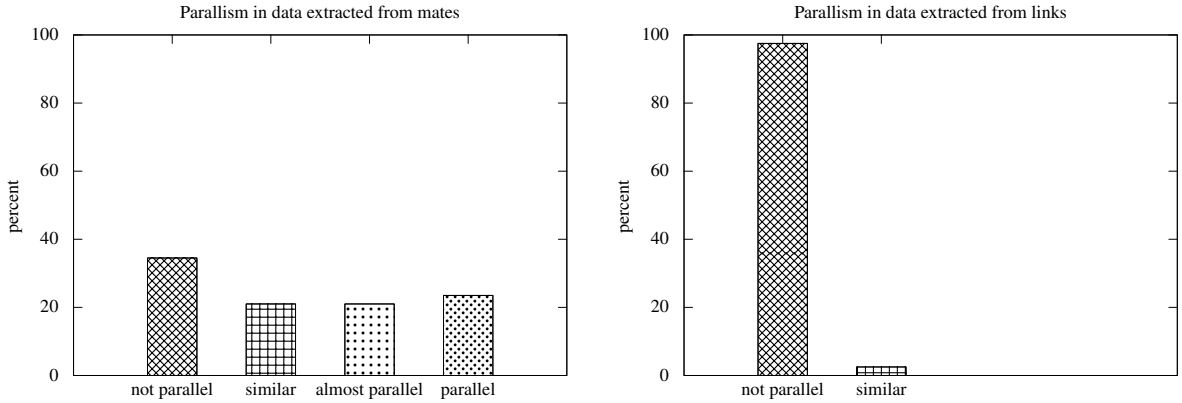


Figure 3: Sentence aligner precision for mates and links.

3. To compare our method to automatic parallel data extraction based on relevance annotation.

We use the modified `yalign` method described by Wołk and Marasek (2015) for pseudo-parallel data extraction.We adapt their software to handle the WikiCLIR format. `yalign` requires a bilingual dictionary with translation probabilities. Following Wołk and Marasek (2015), we use a lexical translation table created from the TED parallel training data[2] as our bilingual dictionary. We filter the dictionary for punctuation and numerals and discard all entries whose lexical translation probability is smaller than 0.3.

Figure 2 shows the frequency histogram of the number of extracted lines per document pair for document pairs with a mate relation. For most document pairs, only a single sentence pair was extracted. However, there were a few document pairs that yielded several hundred sentence pairs. In total, 533,516 sentence pairs were extracted.

Figure 3 shows our evaluation of `yalign`'s precision for the mate and link relations. We manually evaluated a sample of 200 automatically aligned sentence pairs. The sentence pairs were annotated using four categories: "fully parallel", "almost parallel" – this category contains sentence pairs that have parallel segments, with other segments missing from the aligned part, "similar" – for sentence pairs that have similar content or wording but differ factually –, and "non parallel". While 65.5% of sentence pairs from the mate relation were similar or parallel, the link relation yielded only 2.6% sentence pairs that were at least similar. We conclude that the bidirectional link relation is too weak to extract useful pseudo-parallel data.

[2] `https://wit3.fbk.eu/`

Set	Length	# parallel	Title
set1	323	285	Polish culture during World War II
	710	677	Black-figure pottery
	457	375	Ulm Hauptbahnhof
	587	375	Characters of Carnivàle
Total		1712	
set2	360	268	J-pop
	501	388	Schüttorf
	549	438	Military history of Australia during World War II
	676	432	Arab citizens of Israel
Total		1526	

Table 1: Wikipedia development and test documents.

3.3 Evaluation data construction

To construct our in-domain evaluation data, we sorted all automatically aligned documents by the number of aligned sentences up to a limit of 10,000 sentences. We then selected eight documents for manual alignment, discarding other document pairs which appeared to have been machine-translated, only contained few parallel sentences, or consisted of lists of proper names. During manual alignment, we also fixed sentence splitting errors and removed image captions and references. We split the documents into two groups of four, making sure to keep the sets diverse. Table 1 shows the two sets of extracted documents. They are topically diverse, similar in length, and contain a considerable percentage of parallel sentences.

4 Experiments

4.1 Out-of-domain translation system

Our baseline English-German translation system is trained on 2.1 million sentence pairs (61/59 million English/German tokens) from the Europarl v7[3] corpus (1.78 million sentence pairs), the News Commentary v10[4] corpus (200K sentence pairs) and the MultiUN v1[5] corpus (150K sentence pairs). Word alignments are computed using MGIZA++[6], alignments are symmetrized using the `grow-diag-final-end` heuristic. A 4-gram count-based language model is estimated from the target side of the training data using `lmplz` (Heafield et al., 2013). All experiments use the hierarchical phrase-based decoder `cdec` (Dyer et al., 2010). Hierarchical phrase rules are extracted using `cdec`'s implementation of the suffix array extractor by Lopez (2007) with default settings. Our baselines use 21 decoder features (7 translation model features, 2 language model features, 7 pass through features, 3 arity penalty features, word penalty and glue rule count features), which are implemented in `cdec`. Feature weights are optimized on the WMT Newstest 2014 data set (3003 sentence pairs) using the pairwise ranking optimizer `dtrain`[7]. We run `dtrain` for 15 epochs with the hyperparameters k-best size=100, loss-margin=1, and a learning rate of $1e^{-5}$. The final weights are averaged over all epochs. Performance of our baseline system (*baseline 1*) is given in the first row of Table 2.

4.2 Translation model and language model adaptation

For translation model (TM) adaptation we add the automatically extracted pseudo-parallel Wikipedia data (see Section 3) to our baseline training data and re-train the translation model. For language model (LM) adaptation, we sample 500,000 sentences from the German Wikipedia data, which we add to the out-of-domain language model data to re-build a combined 4-gram language model. Both language model and translation model adaptation boosted performance. Rows 1 and 3 in Table 3 show BLEU

[3] `www.statmt.org/europarl/`, see (Koehn, 2005)
[4] `www.statmt.org/wmt15/training-parallel-nc-v10.tgz`
[5] `www.euromatrixplus.net/multi-un/`, see (Eisele and Chen, 2010)
[6] www.cs.cmu.edu/ qing/giza/
[7] `https://github.com/pks/cdec-dtrain`

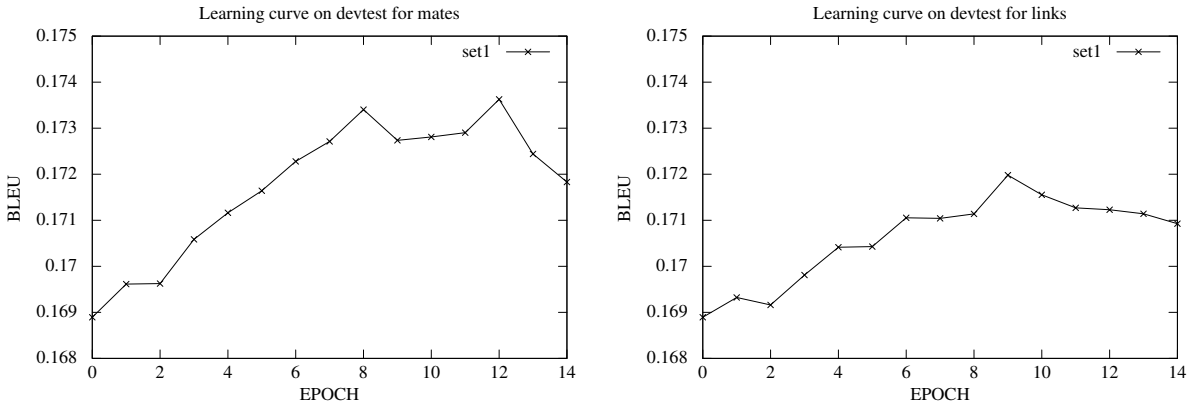


Figure 4: Performance on heldout *set1* for mates and links.

Experiment	%BLEU *set2*
baseline 1 (out-of-domain)	12.46
kbest-train (mates, cost_diff)	12.57 (+0.11)
hypergraph-train (mates, cost_diff)	13.05* (+0.59)
hypergraph-train (mates, cost)	12.81* (+0.34)
hypergraph-train (mates+links, cost_diff)	12.85* (+0.38)
hypergraph-train (links, cost_diff, random sampling)	12.67 (+0.21)
hypergraph-train (links, cost_diff, weighted sampling)	12.77* (+0.31)

Table 2: Results for training on Wikipedia with out-of-domain model. * indicates a significant difference to the baseline at a significance level of 0.05.

scores for an LM-adapted model (*baseline 2*) and a model with both LM and TM adaptation (*baseline 3*). The good performance of TM adaptation leads to the conclusion that if there is a strong signal for potential parallelism like in the Wikipedia data, automatic pseudo-parallel data extraction works well.

4.3 Learning from Wikipedia mates and links

We train our method on 10,000 input sentences sampled from the English WikiCLIR documents. Each input sentence is annotated with a document identifier in order to sample positive and negative examples. The relevant document collection D^+ includes all German documents which are linked to an English document by a mate or bidirectional link relation. For the contrast documents D^- we use the News Commentary corpus, split into documents. In a pre-processing step, we extract n-grams up to order 3 from each document, which we need to calculate n-gram precision. We also experimented with a larger training set of 200,000 input sentences but found no significant improvement. All experiments use the same 21 features as the baseline, keeping those weights fixed, but train sparse lexicalized features (rule identifiers, rule source and target bigrams and lexical alignment features described in Simianer et al. (2012)) in parallel on 10 shards, followed by an ℓ_1/ℓ_2 feature selection step which keeps at most 100,000 features. We use a constant learning rate of $\eta = 1e^{-4}$ and regularization strength $C = 1$. Experiments were run for up to 20 epochs and performance on the heldout *set1* was used as an early stopping criterion.

Table 2 reports BLEU scores on *set2*, when our model is trained on an unadapted baseline model (*baseline 1*). Significance tests were conducted by `multeval` (Clark et al., 2011). While training on k-best lists produced a small incremental gain, training on hypergraphs improved up to 0.6 BLEU over the baseline. Both cost and cost_diff produced an improvement over the baseline, however, cost_diff performed slightly better. As expected, using the strongest signal, the cross-lingual mate relation, worked best. When only the link relation was used, only the experiment with the weighted sampling strategy produced a significant improvement. Figure 4 shows learning curves over epochs on heldout *set1* for

Experiment	%BLEU *set2*
baseline 2 (LM adaptation)	13.62
hypergraph-train (mates, cost_diff)	13.93* (+0.31)
baseline 3 (LM and TM adaptation)	14.96
hypergraph-train (mates, cost_diff)	15.17* (+0.21)

Table 3: Results for training on Wikipedia with adapted model. * indicates a significant difference to the baseline at a significance level of 0.05.

training on mates and links (both with random sampling).

Table 3 shows results for training on mates with an adapted baseline model. In both experiments, there was a small, yet significant, improvement over the adapted model, showing that additional information can be learned from the relevance signal.

Examples. To give a better impression what is learned by our method, Table 4 contains some example translations from *baseline 1* and the best adapted model from Table 2. Spans in which our model performed better are marked in **boldface**. *Example 1* shows that our model fixed word order mistakes made by the baseline, such as "Apartheid zionistischen", which is fixed to "zionistischen Apartheid". The same is true for the proper name and attribution "Thomas Michael Hamerlik (CDU)" in *Example 2*. Both examples suggest that by training on Wikipedia documents, which include frequent parentheses, quotations and named entities, our model becomes better at handling these types of phrases. *Example 3* is interesting, because in this case the baseline produced an idiomatic, rather informal translation for "postponed indefinitely" ("auf den Sankt - Nimmerleins - Tag verschoben") which would be correct in a spoken language context but strange to use in a Wikipedia article, while our model produced the correct translation ("auf unbestimmte Zeit verschoben").

5 Conclusion and future work

In this paper we have presented a new objective for learning translation model parameters from graded and negative relevance signals. Using Wikipedia translation as an example, we were able to achieve significant improvements over an unadapted baseline. As expected, a stronger relevance signal produced larger gains, but we were able to produce small, but significant, improvements, even when learning only from indirect links. We compare our method to baselines that use monolingual data to adapt the language model or rely on strong parallelism signals to adapt the translation model. Our approach was able to yield a small gain even when combined with these strong baselines.

It is worth mentioning that our approach is not restricted to Wikipedia data, but could be applied to other large multilingual collections where cross-lingual relevance information can be extracted. For example, cross-lingual mates could be extracted for multilingual patent corpora through patent family relations (i.e. versions of the same patent submitted to different patent organizations). In addition, weaker links are given by the international patent classification system, or by citations between patents. Another application scenario could be social media data which use the same hashtags across languages. If no explicit signals are available, or if they are not strong enough, one could also use unsupervised document similarity metrics or cross-language information retrieval techniques to detect relevant documents in a target language collection and use these documents as positive examples. We plan to explore these directions in the future.

Since our general learning setup and objective is agnostic about the type of translation system we also plan to apply it to neural machine translation.

Acknowledgements

This research was supported in part by DFG grant RI-2221/1-2 "Weakly Supervised Learning of Cross-Lingual Systems".

Example 1	
Source	political demands include " **the return of all Palestinian refugees to their homes and lands** , [an] end [to] the Israeli occupation and **Zionist apartheid** and the establishment [of] a democratic secular state in Palestine as the ultimate solution to the Arab - Zionist conflict . "
Baseline 1	politische Forderungen : " **alle palästinensischen Flüchtlingen die Rückkehr an ihre Heimstätten und Land beenden** , [an] [. . .] der israelischen Besatzung und **Apartheid zionistischen** [der] sowie die Einrichtung einer demokratischen säkularen Staat in Palästina als die ultimative Lösung für das arabisch - zionistischen Konflikt . "
Hypergraph-train	politische Forderungen aufzunehmen " **die Rückkehr aller palästinensischen Flüchtlinge in ihre Heimat und zu ihren Ländereien** , [an] Ende [. . .] der israelischen Besatzung und **zionistischen Apartheid** und die Einrichtung [der] einen demokratischen säkularen Staat in Palästina als die ultimative Lösung des arabisch - zionistischen Konflikt . "
Reference	politische Forderungen von Abnaa el-Balad sind u. a. " ... **die Rückkehr aller palästinensischen Flüchtlinge in ihre Heimat und auf ihr Land** , [ein] Ende [der] israelischen Besatzung und **zionistischen Apartheid** und die Gründung eines demokratischen säkularen Staates in Palästina als endgültige Lösung des arabisch - zionistischen Konflikts .
Example 2	
Source	the current mayor is **Thomas Michael Hamerlik (CDU)** with two deputies :
Baseline 1	der derzeitige Bürgermeister **Michael Hamerlik Thomas** ist mit zwei Stellvertreter **(CDU)** :
Hypergraph-train	der derzeitige Bürgermeister ist **Thomas Michael Hamerlik (CDU)** mit zwei Abgeordneten :
Reference	Bürgermeister ist zurzeit **Thomas Michael Hamerlik (CDU)** mit zwei Stellvertretern :
Example 3	
Source	this plan was frustrated by the Japanese defeat in the Battle of the Coral Sea and was **postponed indefinitely** after the Battle of Midway .
Baseline 1	dieser Plan wurde von den Japanern frustriert Niederlage im Kampf der Coral See und nach der Schlacht von Midway **auf den Sankt - Nimmerleins - Tag verschoben** wurde .
Hypergraph-train	dieser Plan wurde frustriert durch die japanische Niederlage im Kampf der Coral Meer und nach der Schlacht von Midway **auf unbestimmte Zeit verschoben** wurde .
Reference	der japanische Plan erlitt mit der Niederlage in der Schlacht im Korallenmeer einen ersten Rückschlag und wurde nach der Niederlage in der Schlacht um Midway **auf unbestimmte Zeit verschoben** .

Table 4: Translation examples from the test set, comparing the unadapted baseline to adaptation with our method.

References

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research*, 13(Apr):1159–1187.

Jonathan Clark, Chris Dyer, Alon Lavie, and Noah Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Lingustics*, ACL '11, Portland, Oregon, USA.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, Uppsala, Sweden.

Pavlos S. Efraimidis and Paul G. Spirakis. 2006. Weighted random sampling with a reservoir. *Information Processing Letters*, 97(5):181 – 185.

Andreas Eisele and Yu Chen. 2010. Multiun: A multilingual corpus from united nation documents. In *LREC*.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for smt. In *Proceedings of the Second Workshop on Statistical Machine Translation*, WMT '07, Prague, Czech Republic.

Kevin Gimpel and Noah A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, Montreal, Canada.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, Sofia, Bulgaria.

Joseph Keshet and David A McAllester. 2011. Generalization bounds and consistency for latent structural probit and ramp loss. In *Advances in Neural Information Processing Systems*, pages 2205–2212.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, WMT '07, Prague, Czech Republic.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Prague, Czech Republic.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159, Denver, Colorado, June. Association for Computational Linguistics.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504.

Shigehiko Schamoni, Felix Hieber, Artem Sokolov, and Stefan Riezler. 2014. Learning translational and knowledge-based similarities from relevance rankings for cross-language retrieval. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, ACL '14, Baltimore, MD, USA.

Patrick Simianer, Stefan Riezler, and Chris Dyer. 2012. Joint feature selection in distributed stochastic learning for large-scale discriminative training in smt. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, ACL '12, Jeju Island, Korea.

Jason R. Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. 2013. Dirt cheap web-scale parallel text from the common crawl. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, Sofia, Bulgaria.

Krzysztof Wołk and Krzysztof Marasek. 2015. Unsupervised comparable corpora preparation and exploration for bi-lingual translation equivalents. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, IWSLT '15, Da Nang, Vietnam.

Universal Reordering via Linguistic Typology

Joachim Daiber **Miloš Stanojević** **Khalil Sima'an**
Institute for Logic, Language and Computation (ILLC)
University of Amsterdam
{initial.last}@uva.nl

Abstract

In this paper we explore the novel idea of building a single *universal* reordering model from English to a large number of target languages. To build this model we exploit typological features of word order for a large number of target languages together with source (English) syntactic features and we train this model on a *single combined* parallel corpus representing all (22) involved language pairs. We contribute experimental evidence for the usefulness of linguistically defined typological features for building such a model. When the universal reordering model is used for preordering followed by monotone translation (no reordering inside the decoder), our experiments show that this pipeline gives comparable or improved translation performance with a phrase-based baseline for a large number of language pairs (12 out of 22) from diverse language families.

1 Introduction

Various linguistic theories and typological studies suggest that languages often share a number of properties and that their differences fall into a small set of parameter settings (Chomsky, 1965; Greenberg, 1966; Comrie, 1981). While this intuition has influenced work on multilingual parsing (Zeman and Resnik, 2008; McDonald et al., 2011), it has found less practical use in other areas of natural language processing, such as the task of machine translation. In machine translation, significant word order differences between languages often constitute a challenge to translation systems. Word order differences are frequently given special treatment, such as in the case of preordering (Xia and McCord, 2004; Neubig et al., 2012; Stanojević and Sima'an, 2015, inter alia), which is a technique heavily used in practice as a means to improve both translation quality and efficiency. In preordering, word order is predicted based on manually created rules or based on statistical models estimated on word-aligned training data exploiting only source language features. This approach works well for some language pairs, however it usually demands a separate, dedicated preordering model for every source-target language pair, trained on a word-aligned corpus specific for the particular language pair.

But if the similarities and differences between languages can indeed be captured with a small set of features, as linguistic theory suggests, then it seems more expedient to try to benefit from the similarities between target languages in the training data, which is not possible when training a separate preordering model for every new target language. Ideally, the word-aligned data obtained for various target languages should be combined to train a single, *universal* reordering model with a single set of features. The questions addressed in this paper are (1) could a linguistically inspired universal reordering model show any promising experimental results and (2) how can such a universal reordering model be built?

For building an effective universal reordering model, we need access to a resource that describes the similarities and differences between (target) languages in a small set of properties. The World Atlas of Language Structures (Dryer and Haspelmath, 2013), WALS, is a major resource which currently specifies the abstract linguistic properties of 2,679 languages.[1] In this paper we explore the use of the

[1] http://wals.info/languoid

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3167–3176, Osaka, Japan, December 11-17 2016.

linguistically defined WALS features for a broad range of target languages and show that these features have merit for building a single, universal reordering model. The universal reordering model is based on a feed-forward neural network which is trained to predict the target word order given source syntactic structure and all available WALS parameter settings for each of the 22 target languages involved. By training the feed-forward neural network on the WALS-enriched data from a broad range of target languages, we enable the universal reordering model to both learn how much to trust the WALS parameters and to exploit possible interactions between them for different target languages. When the universal reordering model is followed by *monotone translation* (no reordering inside the decoder), our experiments show that this pipeline gives comparable or improved translation performance to a Moses baseline with standard distortion settings, for a large number of language pairs. This suggests that typological target language features could play a key role in building better, more general preordering models, which have, heretofore, been trained solely on source sentences and word alignments, but had no access to other target-side information.

We believe that the experiments presented in this paper have both theoretical and practical implications. Firstly, they show the utility and provide empirical support for the value of linguistic typology. Secondly, they enable building more compact preordering models that should generalize to a broad set of target languages and which potentially apply for the low resource setting where no or little parallel data is available for a specific target language.

2 Related Work

The most basic usage of linguistic knowledge in preordering is in restricting the search space of possible reorderings by using syntactic parse trees. Earlier work was done mostly on constituency trees (Khalilov and Sima'an, 2012; Xia and McCord, 2004) while more recent versions of preordering models mostly use dependency trees (Lerner and Petrov, 2013; Jehl et al., 2014). Preordering in syntax-based models (whether dependency or constituency) is done on the local level where for each constituent (or head word) the classifier decides how the children (or dependent words) should be reordered.

Employing classifiers to make local decisions on each tree node is one machine learning approach to solving this problem. An alternative to employing machine learning techniques is the use of linguistic knowledge that can in some cases give clear rules for the reordering of children in the tree. An early example of rule-based preordering is by Collins et al. (2005), who develop linguistically justified rules for preordering German into English word order. Similar in spirit but much simpler is the approach of Isozaki et al. (2010), who exploit the fact that Japanese word order is in large part the mirror image of English word order—the heads of constituents in English are in final position while in Japanese they are in initial position. Preordering English sentences into Japanese word order thus only involves two simple steps: (1) Finding the parse tree of the English sentence (the authors used HPSG derivations) and (2) moving the head of each constituent to the initial position. However, this approach does not seem to scale up easily because manually encoding reordering rules for all the world's language pairs would be a rather difficult and very slow process.

In contrast to manually encoding rules for language pairs, we could use similarities and differences between target languages encoded in existing *typological databases* of structural properties of the world's languages, e.g., the World Atlas of Language Structures, WALS (Dryer and Haspelmath, 2013). Therefore, the challenge taken up in the present work is how to exploit typological databases such as WALS to guide the learning algorithm into making the right decisions about word order. So if, for instance, a feature indicates that the target language follows VSO (verb-subject-object) word order, then the preordering algorithm should learn to transform the English parse tree from SVO into a VSO tree. Using typological features like these in a machine learning system for preordering constitutes a compromise between knowledge-based (rules) and data-driven (learning) approaches to preordering.

Researchers in linguistic typology have produced various initiatives to collect typological data in a centralized and structured format out of which WALS is the most comprehensive one. We briefly discuss WALS in Section 3. WALS has been used before in computational linguistics, e.g., by Östling (2015) who performed a typological study of word order based on a corpus of New Testament translations in 986

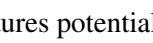
Feature	Name	Distribution of Values	Feature	Name	Distribution of Values
37A	Definite Articles		82A	Order of Subj. and Verb	
46A	Indefinite Pronouns		83A	Order of Object and Verb	
48A	Person Marking on Adpositions		84A	Order of Object, Oblique, and Verb	
52A	Comitatives and Instrumentals		85A	Order of Adposition and NP	
53A	Ordinal Numerals		86A	Order of Genitive and Noun	
54A	Distributive Numerals		87A	Order of Adjective and Noun	
55A	Numeral Classifiers		88A	Order of Demonstrative and Noun	
56A	Conj. and Universal Quantifiers		89A	Order of Numeral and Noun	
57A	Pos. of Pron. Poss. Affixes		90A	Order of Relative Clause and Noun	
61A	Adjectives without Nouns		91A	Order of Degree Word and Adj.	
66A	The Past Tense		92A	Position of Polar Quest. Particles	
67A	The Future Tense		93A	Position of Interr. Phrases	
68A	The Perfect		94A	Order of Adv. Subord. + Clause	
69A	Pos. of Tense-Aspect Affixes		95A	Rel. of Obj. + Verb and Adp. + NP	
81A	Order of Subj., Obj. and Verb		96A	Rel. of Obj. + Verb and Rel. Clause + Noun	
81B	Two Dominant SVO Orders		97A	Rel. of Obj. + Verb and Adj. + Noun	

Table 1: WALS features potentially relevant to determining word order.

languages. This study found that the word order typology created from such data and the information in WALS show a high level of agreement. Finally, Bisazza and Federico (2016) survey word reordering in machine translation and categorize languages based on their WALS features. In the present work, we make novel use of linguistic typological features from WALS for building a universal reordering model.

3 Linguistic Typology

The field of linguistic typology studies the similarities and distinguishing features between languages and aims to classify them accordingly. Among other areas, the World Atlas of Language Structures describes general properties of each language's word order. Overall, WALS contains 192 features, but not all features are relevant to determining word order. Many WALS features deal with phonology, morphology or lexical choice: Feature 129A, for example, describes whether the language's words for "hand" and "arm" are the same. Hence, for simplicity's sake we pre-select the subset of WALS features potentially relevant to determining word order and describe this subset in the following. Table 1 provides an overview of these features, along with an indication of the relative frequency distribution of each of their values over all languages in WALS.

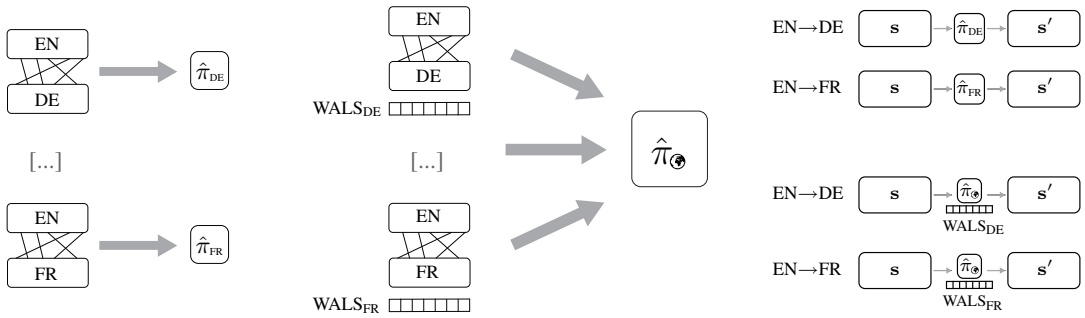
One of the most common ways to classify languages is according to the order of the subject, the object and the verb in a transitive clause. Accordingly, a number of WALS features describe the order of these elements. WALS Feature 81A classifies languages into 6 dominant clause-level word orders. For languages such as German or Dutch, which do not exhibit a single dominant clause-level order, Feature 81B describes 5 combinations of two acceptable word orders. Additionally, two features describe whether the verb precedes the subject (82A) and whether the verb precedes the object (83A). The position of adjuncts in relation to the object and the verb are described in Feature 84A and the internal structure of adpositional phrases is described in Feature 85A, which specifies whether the language uses pre-, post- or inpositions. Finally, the following properties describe the order of words in relation to nouns: Feature 86A specifies the position of genitives (e.g. *the girl's cat*), Feature 87A the position of adjectives (e.g. *yellow house*), Feature 89A the position of numerals (e.g. *10 houses*) and Feature 90A the position of relative clauses (e.g. *the book that I am reading*) in relation to the noun.

4 Universal Reordering Model

Our universal reordering model uses a preordering architecture similar to the (non-universal) preordering model of De Gispert et al. (2015), which in turn is based on the authors' earlier work on logistic regression and graph search for preordering (Jehl et al., 2014).

4.1 Basic Preordering Model

In this neural preordering model, a feed-forward neural network is trained to estimate the swap probabilities of nodes in the source-side dependency tree. The learning task is defined as follows: How likely is it that two nodes *a* and *b* are in the linear order (a, b) or (b, a) in the target language? Preordering then consists of finding the best sequence of swaps according to this model. While De Gispert et al. (2015)



(a) Basic preordering training. (b) Universal reordering model training. (c) Applying the models.

Figure 1: Training and application of basic preordering models and the universal reordering model.

use a depth-first branch-and-bound algorithm to find the best permutation, we use the k-best version of this algorithm and minimize the resulting preordering finite-state automaton to produce a lattice of word order choices (Daiber et al., 2016).

Model estimation Training examples are extracted from all possible pairs of children of the source dependency tree node, including the head itself. The crossing score of two nodes a and b (a precedes b in linear order) and their aligned target indexes A_a and A_b is defined as follows:

$$\mathrm{cs}(a, b) = |\{(i, j) \in A_a \times A_b : i > j\}|$$

A pair (a, b) is swapped if $\mathrm{cs}(b, a) < \mathrm{cs}(a, b)$, i.e. if swapping reduces the number of crossing alignment links. Training instances generated in this manner are then used to estimate the order probability $p(i, j)$ for two indexes i and j. The best possible permutation of each node's children (including the head) is determined via graph search. The score of a permutation π of length k consists of the order probabilities of all possible pairs:

$$\mathrm{score}(\pi) = \prod_{1 \le i < j \le k | \pi[i] > \pi[j]} p(i, j) \cdot \prod_{1 \le i < j \le k | \pi[i] < \pi[j]} 1 - p(i, j)$$

De Gispert et al. (2015) use a feed-forward neural network (Bengio et al., 2003) to predict the orientation of a and b based on 20 source features, such as the words, POS tags, dependency labels, etc.[2]

Permutation lattices To find the sequence of swaps leading to the best overall permutation according to the model, the score of a permutation is obtained by extending a partial permutation π' of length k' by one index i (Jehl et al., 2014). This score can be efficiently computed as:

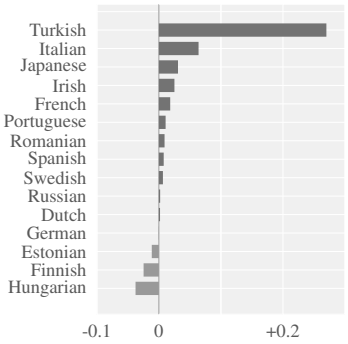
$$\mathrm{score}(\pi' \cdot \langle i \rangle) = \mathrm{score}(\pi') \cdot \prod_{j \in V | i > j} p(i, j) \cdot \prod_{j \in V | i < j} 1 - p(i, j)$$

Instead of extracting the single-best permutation, we use the *k*-best extension of branch-and-bound search (van der Poort et al., 1999). The resulting k-best permutations are then compressed into a minimal deterministic acceptor and unweighted determinization and minimization are performed using OpenFST (Allauzen et al., 2007).

4.2 Estimating a Universal Reordering Model

The universal reordering model differs from the basic neural preordering model in terms of features and training data collection. Differences in training data collection and application are illustrated in Figure 1.

[2]Full set of features: words, word classes, dependency labels, POS tags, coarse POS tags, word and class of the left-most and right-most child token, and the tokens' distance to their parent.



(a) Absolute 1-best Kendall τ improvements.

	Manual word alignments			Automatic word alignments		
Language	τ@10	τ@100	τ@1000	τ@10	τ@100	τ@1000
French	+01.95	+03.05	+03.05	+01.28	+02.12	+02.13
German	+04.03	+05.61	+05.61	+06.85	+08.27	+08.27
Italian	+06.39	+06.75	+06.75	+03.07	+03.32	+03.32
Portuguese	+05.87	+07.89	+07.89	+03.24	+03.55	+03.55
Spanish	+05.97	+06.57	+06.57	+04.28	+05.11	+05.11
Romanian	+02.49	+03.37	+03.37	+01.23	+02.09	+02.10
Swedish	+00.13	+00.42	+00.42	+00.71	+01.18	+01.18

(b) N-best permutation quality on manually and autom. aligned data.

Figure 2: Intrinsic quality of word order predictions (improvement over source word order).

In addition to the source features used in the standard neural preordering model (cf. Section 4.1), we add a feature indicating the source word order of the two tokens, as well as the type of end-of-sentence punctuation. We then add WALS features 37, 46, 48, 52–57, 61, 66–69 and 81–97. WALS features are represented by their ID and the value for the current target language (e.g. "WALS_87A=Adj-Noun" or "WALS_87A=Noun-Adj"). For the most basic word order features (81, 82 and 85–91), we additionally add a feature indicating if the order of the node pair agrees with the order specified by the WALS feature.[3]

While the training data for a standard preordering model consists of source sentences and their target-language order retrieved via word alignments, the training data for the universal reordering model is comprised of training examples from a large number of language pairs. Because of the diversity of this data, special care has to be taken to ensure a balanced dataset. We use an equal number of sentences from each language-specific training subcorpus. Additionally, we reduce class imbalance by further randomly shuffling the source tokens when creating training instances. This ensures a balanced distribution of classes in the training data. The distribution of the two classes is 84.5%/15.5% in the original and 50.1%/49.9% in the randomized dataset.

4.3 Intrinsic Evaluation

We use NPLM (Vaswani et al., 2013) to train a feed-forward neural network to predict the orientation of two nodes a and b based on the features described in Section 4.2. The network consists of 50 nodes on the input layer, 2 on the output layer, and 50 and 100 on the two hidden layers. We use a learning rate of 0.01, batch sizes of 1000/64 and perform 60 training epochs, ensuring convergence of the log-likelihood on a validation set.

Preordering data The training data for the universal reordering model consists of a combined corpus of 30k sentence pairs each from the Tatoeba corpus (Tiedemann, 2012) for French, German, Japanese, Portuguese, Russian, and Spanish as well as 100k sentence pairs each from the OpenSubtitles 2012 corpus (Tiedemann, 2012) for Spanish, Portuguese, Italian, Danish, Romanian, Swedish, French, Greek, Russian, Polish, Arabic, Hebrew, Hungarian, Czech, Finnish, Icelandic, Dutch, Slovak, Chinese, German and Turkish. Word alignments for all corpora were produced using MGIZA (Och and Ney, 2003) using *grow-diag-final-and* symmetrization and performing 6, 6, 3 and 3 iterations of IBM M1, HMM, IBM M3 and IBM M4 respectively. To evaluate the model, we also use sets of manually word-aligned sentence for the following language pairs: En–Ja (Neubig, 2011), En–De (Padó and Lapata, 2006), En–It (Farajian et al., 2014), En–Fr (Och and Ney, 2003), En–Es and En–Pt (Graça et al., 2008).

Quality of word order predictions Figure 2a shows the intrinsically measured quality of the predictions by the universal reordering model. We use Kendall τ (Kendall, 1938) to measure the correlation

[3]Example for WALS feature 87A=Adj-Noun: $f(a,b) = \begin{cases} \text{"W87A:ab"} & \text{if } a = \text{adj} \wedge b = \text{noun} \\ \text{"W87A:ba"} & \text{if } a = \text{noun} \wedge b = \text{adj} \end{cases}$

Language	# sent.	# tok.	BiHDE ↑	Language	# sent.	# tok.	BiHDE ↑	Language	# sent.	# tok.	BiHDE ↑
Spanish	800k	14.29	0.57	Greek	800k	14.36	0.65	Finnish	800k	14.36	0.69
Portuguese	800k	14.29	0.58	Russian	800k	15.08	0.65	Icelandic	800k	14.10	0.69
Italian	800k	14.68	0.61	Polish	800k	14.22	0.67	Dutch	800k	14.37	0.70
Danish	800k	14.50	0.62	Arabic	800k	14.84	0.68	Slovak	638k	15.08	0.70
Romanian	800k	14.24	0.64	Hebrew	800k	14.61	0.68	Chinese	636k	10.39	0.71
Swedish	800k	14.49	0.64	Hungarian	800k	14.33	0.68	German	800k	14.62	0.72
French	800k	14.25	0.65	Czech	800k	14.19	0.69	Turkish	800k	14.25	0.72

Table 2: Properties of training data from the 2012 OpenSubtitles corpus.

between the predicted word order and the *oracle* word order determined via the word alignments. Figure 2a plots absolute Kendall τ improvement over the original, i.e. unreordered, source sentence for the single best permutation for a number of language pairs. The three worst-performing target languages in Figure 2a, Estonian, Finnish and Hungarian, are all morphologically rich, indicating that additional considerations may be required to improve word order for languages of this type. Figure 2b shows the quality of n-best permutations of the universal reordering model for both manually and automatically word-aligned sentence pairs. This table allows two observations: Firstly, the evaluation of word order quality using automatic alignments shows good agreement with the evaluation using manually word-aligned sentences, thus highlighting that automatic alignments should suffice for this purpose in most cases. Secondly, we can observe that for all datasets presented in this table little is gained from increasing the number of extracted permutations beyond 100 predictions. We therefore apply a maximum number of 100 permutations per sentence in all experiments presented in the rest of this paper.

5 Translation Experiments

To evaluate the universal reordering model in a real-world task, we perform translation experiments on various language pairs. As a baseline system, we use a plain phrase-based machine translation system using a distortion-based reordering model with a distortion limit of 6. When applying the universal reordering model, we produce a lattice from each sentence's best 100 word order permutations. This lattice is then passed to the machine translation system and no additional reordering is allowed. During training, we choose the source sentence permutation closest to the gold word order determined via the word alignments (lattice silver training; Daiber et al., 2016). The word alignments for the preordered training corpus are then recreated from the original MGIZA alignments and the selected permutation.[4]

Translation experiments are performed with a phrase-based machine translation system, a version of Moses (Koehn et al., 2007) with extended lattice support.[5] We use the basic Moses features and perform 15 iterations of batch MIRA (Cherry and Foster, 2012). To control for optimizer instability, we perform 3 tuning runs for each system and report the mean BLEU score for these runs (Clark et al., 2011). As a baseline we use a translation system with distortion limit 6 and a distance-based reordering model. For each language pair, a 5-gram language model is estimated using *lmplz* (Heafield et al., 2013) on the target side of the parallel corpus.

5.1 Evaluating on a Broad Range of Languages

In order to test the ideas presented in this paper, we evaluate our model on a broad range of languages from various language families. While doing so, it is important to ensure that the results are not skewed by differences in the corpora used for training and testing each language pair. We therefore build translation systems from the same corpus and domain for every language pair. We use the 2012 OpenSubtitles corpus[6] (Tiedemann, 2012) to extract 800,000 parallel sentences for each language pair, ensuring that every sentence pair contains only a single source sentence and that every source sentence contains at least 10 tokens. For each language pair, 10,000 parallel sentences are retained for tuning and testing. We use English as the source language in all language pairs. Table 2 summarizes properties of the data

[4]To keep the experiments manageable, we opted not to re-align the preordered training corpus using MGIZA. Re-alignment often leads to improved translation results, therefore we are likely underestimating the potential preordered translation quality.

[5]Made available at `https://github.com/wilkeraziz/mosesdecoder`.

[6]`http://opus.lingfil.uu.se/OpenSubtitles2012.php`

	BLEU	Δ BLEU				BLEU	Δ BLEU		
Language	Baseline	No WALS	WALS ↓	Gold	Language	Baseline	No WALS	WALS ↓	Gold
Dutch	13.76	+0.11	+0.79	+3.44	Greek	7.22	−0.02	+0.01	+0.49
Italian	23.59	+0.04	+0.48	+1.83	Arabic	5.36	−0.10	−0.01	+0.36
Turkish	5.89	−0.36	+0.43	+0.80	Swedish	25.60	−0.14	−0.03	+2.04
Spanish	23.82	−0.27	+0.29	+1.98	Slovenian	10.56	−0.35	−0.10	+1.21
Portuguese	25.94	−0.48	+0.21	+1.64	Slovak	15.56	−0.09	−0.13	+1.98
Finnish	9.95	+0.13	+0.16	+0.51	Icelandic	14.97	−0.31	−0.14	+0.66
Hebrew	11.64	+0.30	+0.11	+2.24	Polish	17.68	−0.45	−0.16	+0.40
Romanian	16.11	+0.11	+0.11	+1.14	Russian	20.12	−0.47	−0.17	+0.92
Hungarian	8.26	−0.10	+0.10	+0.61	German	17.08	−0.21	−0.19	+3.31
Danish	26.36	−0.13	+0.08	+1.56	Czech	12.81	−0.47	−0.21	+0.70
Chinese	11.09	−0.32	+0.05	+0.44	French	19.92	−0.70	−0.23	+1.20

Table 3: Translation experiments with parallel subtitle corpora.

used in these experiments. Apart from the average sentence length and the number of training examples, we report Bilingual Head Direction Entropy, BiHDE (Daiber et al., 2016), which indicates the difficulty of predicting target word order given the source sentence and its syntactic analysis. The language pairs in Table 2 are sorted by their BiHDE score, meaning that target languages whose word order is more deterministic are listed first. For each language pair, we train four translation systems:

Baseline The baseline system is a standard phrase-based machine translation system with a distance-based reordering model, a distortion limit of 6, and a maximum phrase length of 7.

Gold The gold system provides an indication for the upperbound achievable translation quality using preordering. In this system, the tuning and test sets are word-aligned along with the training portion of the corpus and the word alignments are then used to determine the optimal source word order. While this system provides an indication for the theoretically achievable improvement, this improvement may not be achievable in practice since not all information required to determine the target word order may be available on the source side (e.g. morphologically rich languages can allow several interchangeable word order variations). Apart from the source word order, the gold system is equivalent to the Baseline system.

No WALS As a baseline for our preordering systems, we create a translation system that differs from our universal reordering model only in the lack of WALS information. The preordering model is trained using the standard set of features described in Section 4.1 with only a single additional feature: the name of the target language. As in the WALS system, this system is applied by generating a minimized lattice from the 100-best permutations of each sentence and restricting the decoder's search space to this lattice. This system therefore isolates two potential sources of improvement: (1) improvement due to restricting the search space by the source dependency tree and (2) improvement from the preordering model itself, independent of the typology information provided by WALS.

WALS The WALS system applies the universal reordering model introduced in Section 4.2. For each language pair, the preordering model is provided with the target language and all the WALS features available for this language. The MT system's search space is then restricted using the minimized lattice of the 100-best word order permutations for each sentence and no additional reordering within the MT decoder is allowed.

The results of the translation experiments using the OpenSubtitles corpora are presented in Table 3. BLEU scores for the No WALS, WALS and Gold systems are reported as absolute improvement over the Baseline system (Δ BLEU). Over the three tuning runs performed for each system, we observe minor variance in BLEU scores (mean standard deviations: Baseline 0.04, No WALS 0.05, WALS 0.05, Gold 0.07), thus we report the mean BLEU score for each system's three runs.

While performing monotone decoding (i.e., allowing no reordering on top of the input lattice), the universal reordering model (WALS) enables improvements or comparable performance for the majority

	Dataset				Baseline	WALS
Language	Domain	# sent.	# tok.	BiHDE	BLEU	Δ BLEU ↓
Turkish	News	0.20m	23.54	0.73	8.27	+0.34
Spanish	Parl. + News	1.73m	23.47	0.58	24.34	+0.18
Italian	Parl. + News	1.67m	24.49	0.61	24.83	+0.13
Portuguese	Parl. + News	1.73m	23.67	0.58	32.13	−0.08
Hungarian	Parl. + News	1.41m	17.11	0.70	7.63	−0.19

Table 4: Translation experiments with varying training data and domains.

of the language pairs we evaluated while the No WALS system performs worse for most language pairs. This suggests that the improvements are not due to the neural preordering model or the lattice-based translation alone, but that the WALS information is crucial in enabling these results.

5.2 Influence of Domain and Data Size

While the experiments using the subtitle corpora presented in the previous section allow a fair comparison of a large number of language pairs, they also exhibit certain restrictions: (1) all experiments are limited to a single domain, (2) the source sentences are fairly short, and (3) to ensure consistent corpus sizes, a limited number of 800k sentence pairs had to be used. Therefore, we perform an additional set of experiments with data from different domains, longer sentences and a larger number of sentence pairs.

To train the translation systems for these experiments, we use the following training data: For En–It, En–Es and En–Pt, we train systems on Europarl v7 (Koehn, 2005). En–Hu uses the WMT 2008 training data,[7] En–Tr the SETIMES2 corpus (Tiedemann, 2009). Tuning is performed on the first 1512 sentences of newssyscomb2009+newstest2009 (En–It), newstest2009 (En–Es), newsdev2016 (En–Tr), newstest2008 (En–Hu), and the first 3000 sentences of news commentary v11 (En–Pt). As test sets we use the rest of newssyscomb2009+newstest2009 (En–It), newstest2013 (En–Es), newstest2016 (En–Tr), newstest2009 (En–Hu), and the first 3000 sentences of news commentary v11 not used in the dev set (En–Pt). All datasets are filtered to contain sentences up to 50 words long, and tokenization and truecasing is performed using the Moses tokenzier and truecaser. Statistics about each dataset and the dataset's domains, as well as translation results for the baseline system and the universal reordering model are summarized in Table 4. The results indicate that despite the longer sentences and different domains, the universal reordering model performs similarly as in the experiments performed in Section 5.1.

Our intrinsic evaluation (Section 4.3) as well as the extrinsic evaluation on a translation task (Section 5) indicate that a universal reordering model is not only feasible but can also provide good results on a diverse set of language pairs. The performance difference between the No WALS baseline and the universal reordering model (cf. Table 3) further demonstrates that the typological data points provided by WALS are the crucial ingredient in enabling this model to work.

6 Conclusion

In this paper, we show that linguistics in the form of linguistic typology and modern methods in natural language processing in the form of neural networks are not rivaling approaches but can come together in a symbiotic manner. In the best case, combining both approaches can yield the best of both worlds: the generalization power of linguistic descriptions and the good empirical performance of statistical models. Concretely, we have shown in this paper that it is possible to use linguistic typology information as input to a preordering model, thus enabling us to build a single model with a single set of model parameters for a diverse range of languages. As an empirical result, our findings provide support for the adequacy of the language descriptions found in linguistic typology. Additionally, they open the way for more compact and universal models of word order that can be especially beneficial for machine translation between language pairs with little parallel data. And finally, our results suggest that target-language typological features could play a key role in building better preordering models.

[7] http://www.statmt.org/

Acknowledgements

We thank the three anonymous reviewers for their constructive comments and suggestions. This work received funding from EXPERT (EU FP7 Marie Curie ITN nr. 317471), NWO VICI grant nr. 277-89-002, DatAptor project STW grant nr. 12271 and QT21 project (H2020 nr. 645452).

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer. `http://www.openfst.org`.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

Arianna Bisazza and Marcello Federico. 2016. A survey of word reordering in statistical machine translation: Computational models and language phenomena. *Computational Linguistics*, 42(2):163–205, June.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Montréal, Canada, June.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. The MIT Press, Cambridge, MA.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 176–181, Portland, Oregon, USA, June.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 531–540, Ann Arbor, Michigan, June.

Bernard Comrie. 1981. *Language Universals and Linguistic Typology: Syntax and Morphology*. Blackwell, Oxford.

Joachim Daiber, Miloš Stanojević, Wilker Aziz, and Khalil Simaan. 2016. Examining the relationship between preordering and word order freedom in machine translation. In *Proceedings of the First Conference on Machine Translation (WMT16)*, Berlin, Germany, August. Association for Computational Linguistics.

Adrià De Gispert, Gonzalo Iglesias, and Bill Byrne. 2015. Fast and accurate preordering for SMT using neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1012–1017, Denver, Colorado, May–June.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

M. Amin Farajian, Nicola Bertoldi, and Marcello Federico. 2014. Online word alignment for online adaptive machine translation. In *Proceedings of the EACL 2014 Workshop on Humans and Computer-assisted Translation*, pages 84–92, Gothenburg, Sweden, April.

João Graça, Joana Paulo Pardal, and Luísa Coheur. 2008. Building a golden collection of parallel multi-language word alignments.

Joseph H. Greenberg. 1966. *Universals of Language*. The MIT Press, Cambridge, MA, 2nd edition.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria, August.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2010. Head finalization: A simple reordering rule for sov languages. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 244–251, Uppsala, Sweden, July.

Laura Jehl, Adrià de Gispert, Mark Hopkins, and Bill Byrne. 2014. Source-side preordering for translation using logistic regression and depth-first branch-and-bound search. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 239–248, Gothenburg, Sweden.

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, pages 81–93.

Maxim Khalilov and Khalil Sima'an. 2012. Statistical translation after source reordering: Oracles, context-aware models, and empirical analysis. *Natural Language Engineering*, 18(4):491–519.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X*, volume 5, pages 79–86.

Uri Lerner and Slav Petrov. 2013. Source-side classifier preordering for machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 62–72, Edinburgh, Scotland, UK., July.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. 2012. Inducing a discriminative parser to optimize machine translation reordering. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 843–853, Jeju Island, Korea, July.

Graham Neubig. 2011. The Kyoto free translation task. http://www.phontron.com/kftt.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Robert Östling. 2015. Word order typology through multilingual word alignment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 205–211, Beijing, China, July.

Sebastian Padó and Mirella Lapata. 2006. Optimal constituent alignment with edge covers for semantic projection. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1161–1168, Sydney, Australia, July.

Miloš Stanojević and Khalil Sima'an. 2015. Reordering grammar induction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 44–54, Lisbon, Portugal, September.

Jörg Tiedemann. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Advances in Natural Language Processing*, volume V, pages 237–248. John Benjamins, Amsterdam/Philadelphia, Borovets, Bulgaria.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 2214–2218, Istanbul, Turkey, May. European Language Resources Association (ELRA).

Edo S van der Poort, Marek Libura, Gerard Sierksma, and Jack A.A van der Veen. 1999. Solving the k-best traveling salesman problem. *Computers & Operations Research*, 26(4):409 – 425.

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

Fei Xia and Michael McCord. 2004. Improving a statistical mt system with automatically learned rewrite patterns. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *IJCNLP 2008 Workshop on NLP for Less Privileged Languages*, pages 35–42, Hyderabad, India. International Institute of Information Technology.

A Deep Fusion Model for Domain Adaptation in Phrase-based MT

Nadir Durrani
ndurrani@qf.org.qa

Hassan Sajjad
hsajjad@qf.org.qa

Shafiq Joty
sjoty@qf.org.qa

Ahmed Abdelali
aabdelali@qf.org.qa

Qatar Computing Research Institute – HBKU

Abstract

We present a novel fusion model for domain adaptation in Statistical Machine Translation. Our model is based on the joint source-target neural network (Devlin et al., 2014), and is learned by fusing in- and out-domain models. The adaptation is performed by backpropagating errors from the output layer to the word embedding layer of each model, subsequently adjusting parameters of the composite model towards the in-domain data. On the standard tasks of translating English-to-German and Arabic-to-English TED talks, we observed average improvements of +0.9 and +0.7 BLEU points, respectively over a competition grade phrase-based system. We also demonstrate improvements over existing adaptation methods.

1 Introduction

The quality of machine translation systems is sensitive to the domain of the training data. More data, but out-of-domain data, may not be best suited and could even be harmful for specific translation tasks such as translating TED talks (Cettolo et al., 2014), patents (Fujii et al., 2010) and educational content (Guzmán et al., 2013). This is because of the difference in stylistic variations, vocabulary choices and word sense ambiguities across genres. *Domain adaptation* aims at finding the optimal point, that maximizes on the useful information available in the out-domain data, in favor of the in-domain data, while preventing it from degrading the performance of the system. This is either done by selecting a subset from the out-domain data, which is closer to the in-domain data (Matsoukas et al., 2009; Moore and Lewis, 2010), or by re-weighting the probability distribution in favor of the in-domain data (Foster and Kuhn, 2007; Sennrich, 2012).

Recently, there has been a growing interest in deep neural networks (DNNs) and word embeddings with application to numerous NLP problems. The ability to generalize and to better capture non-local dependencies gives edge to the neural models over their traditional counter-part. Several researchers have also attempted to employ DNNs for domain adaptation in SMT. Duh et al. (2013) and Durrani et al. (2015) used DNNs for data selection. Joty et al. (2015) proposed a DNN-based adaptation model for SMT that regularizes the loss function with respect to the in-domain model.

In this paper, we propose a deep fusion approach to domain adaptation for SMT. We use the Neural Network Joint Model (NNJM) proposed by Devlin et al. (2014) as our base model. However, rather than training the model on a plain concatenation of in- and out-domain data or a weighted concatenation (Joty et al., 2015), we first train in- and out-domain NNJM models, and then learn a composite model by readjusting their parameters through backpropagating errors from the output layer to the word embedding layer of each model. The intuition behind learning the models separately, is to learn in-domain model parameters without contaminating them with the out-domain data. In a variant of our model, we restrict backpropagation to only the outermost hidden layer and adjust only the final layer combination weights. The composite model is used as a feature during decoding, where it can help assigning higher scores to the hypotheses that represent lexical choices and patterns preferred by the in-domain data.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,
pages 3177–3187, Osaka, Japan, December 11-17 2016.

We evaluated our model on a standard task of translating TED talks for English-to-German and Arabic-to-English language pairs. Compared to baseline NNJM models trained on a concatenation of in- and out-domain data, our model achieves an average improvement of up to 0.9 BLEU points. The most relevant to our work are the NDAM model of Joty et al. (2015) and the *fine-tuning* method of Luong and Manning (2015). The NDAM model uses data dependent regularization in the NNJM model to weight training instances, while training the model on the concatenated data. The fine-tuning method first trains the NNJM on the concatenated data, then runs a few additional epochs on the in-domain data to tune the model towards in-domain. We found our method to outperform both the NDAM and fine-tuning methods. We also carried experiments against phrase-table weighting methods such as instance weighting (Sennrich, 2012), and phrase-table fill-up combination (Bisazza et al., 2011) and found our approach to outperform these. Our approach is complementary and the gains obtained were found to be additive on top of phrase-table adaptation and MML-based data-selection (Axelrod et al., 2011).

The remainder of this paper is organized as follows. Section 2 briefly describes neural network joint model. Section 3 describes our fusion model for domain adaptation. Section 4 presents results and analysis. Section 5 gives an account on the related work and Section 6 concludes the paper.

2 Neural Network Joint Model

Neural models are quickly becoming the state-of-the-art in machine translation. The ability to generalize and better capture non-local dependencies gives them edge over traditional models. The two most prevalent approaches are to use NNs as a feature inside SMT decoder (Vaswani et al., 2013; Devlin et al., 2014), or as an end-to-end translation system (Luong et al., 2015; Bahdanau et al., 2015; Sennrich et al., 2016) designed as fully trainable model of which every component is tuned based on training corpora to maximize its translation performance. Our work falls in the former category and extends NNJM.

The NNJM model learns a feed-forward neural network from augmented streams of source and target sequences. For a bilingual sentence pair (S, T), NNJM defines a conditional probability distribution:

$$P(T|S) \approx \prod_{i=1}^{|T|} P(t_i|t_{i-1} \dots t_{i-n+1}, \mathbf{s_i}) \tag{1}$$

where, $\mathbf{s_i}$ is an m-word source window for a target word t_i based on the one-to-one alignment between T and S. This is essentially an $(m+n)$-gram bilingual language model originally proposed by Bengio et al. (2003). As shown in Figure 1, each input word in the context has a D dimensional vector representation in the shared embedding layer $E \in \mathbb{R}^{|V_i| \times D}$, where V_i is the input vocabulary; E is considered as a model parameter to be learned. The context of the sequence is represented by a concatenated vector $\mathbf{x_n} \in \mathbb{R}^{(m+n-1)D}$, which is then passed through non-linear hidden layers to learn high-level abstract representations. The output layer defines a `softmax` over the output vocabulary V_o:

$$P(y_n = k|\mathbf{x_n}, \theta) = \frac{\exp\ (\mathbf{w}_k^T \mathbf{z_n})}{\sum_{m=1}^{|V_o|} \exp\ (\mathbf{w}_m^T \mathbf{z_n})} \tag{2}$$

where $\mathbf{z}_n = \phi(\mathbf{x_n})$ defines the non-linear transformations of $\mathbf{x_n}$ through one or more hidden layers, and $\mathbf{w}_k$ are the weights from the outermost hidden layer to the output layer. By setting m and n to be sufficiently large, NNJM can capture long-range cross-lingual dependencies between words. The maximum (log) likelihood objective of the model can be written as:

$$J(\theta) = \sum_{n=1}^{N} \sum_{k=1}^{|V_o|} y_{nk}\ log\ P(y_n = k|\mathbf{x_n}, \theta) \tag{3}$$

where, $y_{nk} = I(y_n = k)$ is an indicator variable (i.e., y_{nk}=1 when y_n=k, otherwise 0).

A major efficiency bottleneck of NNJM is to surmount the computational cost involved in training the model and applying it for MT decoding. Devlin et al. (2014) proposed two tricks to speed up computation in decoding. The first one is to pre-compute the hidden layer computations and fetch them directly as

needed during decoding. The second technique is to train a *self-normalized* NNJM to avoid computation of the softmax normalization factor (i.e., the denominator in Equation 2) in decoding. However, self-normalization does not solve the computational cost of training the model. In the following, we describe a method that addresses this issue.

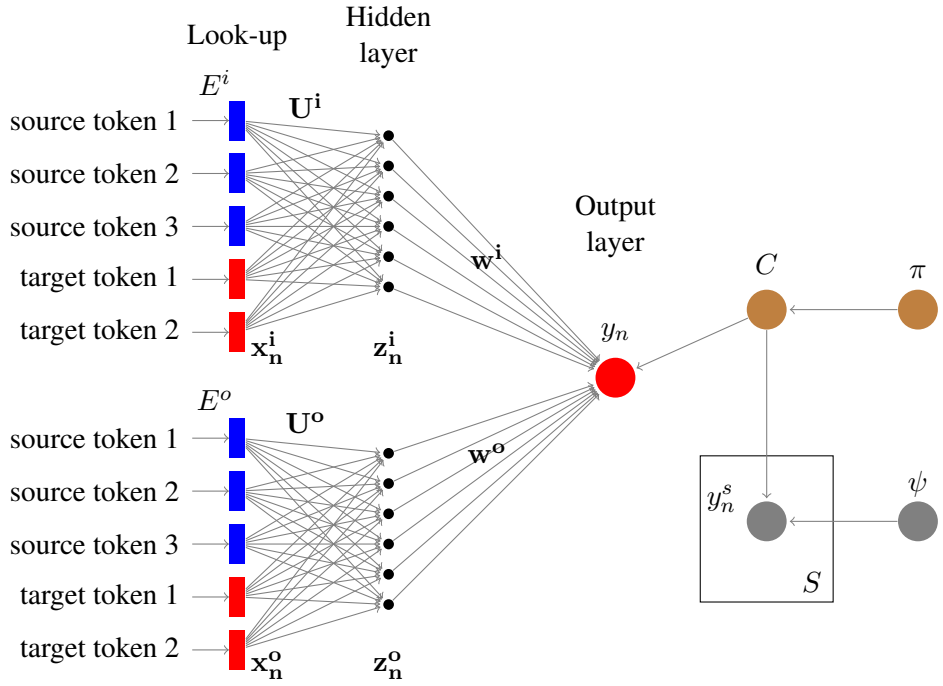


Figure 1: Fusion of two simplified neural network joint models with noise contrastive loss. For illustration, we use 3-gram target words (i.e., 2-words history) and a source context of size 3. The output y_n is shown as a single categorical variable (scalar) as opposed to the traditional one-hot vector representation.

Noise Contrastive Estimation: Like other deep neural models, optimization in NNJM is performed using first-order online methods, such as stochastic gradient ascent (SGA) with standard *backpropagation* algorithm. Unfortunately, training NNJM could be impractically slow using the standard maximum likelihood objective (Eq. 3), because for each training instance $(\mathbf{x_n}, y_n)$, the softmax output layer (Eq. 2) needs to compute a summation over all words in the output vocabulary.

One way to avoid this repetitive computation is to use Noise contrastive estimation or NCE (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012), which adds S noise samples (see Figure 1) for each training instance by sampling from a known noise distribution (e.g., `unigram`). NCE is then defined as such to discriminate a *true* instance from a *noisy* one. Let $C \in \{0, 1\}$ denote the class of an instance with $C = 1$ indicating *true* and $C = 0$ indicating *noise*. NCE maximizes the following conditional log likelihood:

$$J(\theta) = \sum_{n=1}^{N} \Big[log[P(C = 1|y_n, \mathbf{x}_n, \theta)] + \sum_{m=1}^{M} log[P(C = 0|y_n^m, \mathbf{x}_n, \psi)] \Big], \quad (4)$$

which can be simplified as $J(\theta) = \sum_{n=1}^{N} \sum_{k=1}^{|V_o|} \Big[\; y_{nk} \; log \; \sigma_{nk} + \sum_{s=1}^{S} y_{nk}^s \; log \; \psi_{nk} \; \Big]$, where $\sigma_{nk}{=}\exp(\mathbf{w}_k^T \mathbf{z_n})$ is the unnormalized score and $\psi_{nk}{=}P(y_n^s = k|\mathbf{x}_n, \psi)$ is the noise distribution; y_{nk} and y_{nk}^s are indicator variables indicating the true output and the noise sample, respectively. NCE reduces the number of computations needed at the output layer from $|V_o|$ to $S + 1$, where $S << |V_o|$.

3 Neural Fusion Models

The NNJM model trained from a simple concatenation of large and diverse multi-domain data with in-domain data could be sub-optimal, because the resulting probability distribution can diverge from the

target domain hurting the system performance. The goal in domain adaptation is to restrict this drift while still making the best use of the available data. Joty et al. (2015) recently proposed a weighted concatenation approach using data dependent regularizations in the loss function. While it helps, their approach does not fully prevent out-domain data from contaminating the model parameters.

3.1 Fusion Models

We take a different approach of learning in- and out-domain models separately, then fusing them together by readjusting their parameters towards in-domain data.

Fusion Model-I: Let θ^i and θ^o be the parameter sets of the trained in- and out-domain NNJMs, respectively. We combine the two models by redefining the `softmax` output layer (Eq. 2) as follows:

$$P(y_n = k|\mathbf{x_n^i}, \mathbf{x_n^o}, \theta^i, \theta^o) = \frac{\exp\ ([\mathbf{w}_k^i, \mathbf{w}_k^o]^T [\mathbf{z}_n^i, \mathbf{z}_n^o])}{\sum_{m=1}^{|V_o|} \exp\ ([\mathbf{w}_m^i, \mathbf{w}_m^o]^T [\mathbf{z}_n^i, \mathbf{z}_n^o])}$$

where $[\mathbf{w}_k^i, \mathbf{w}_k^o]$ is the concatenation of the output layer weights of in-domain and out-domain models; similarly, $[\mathbf{z}_n^i, \mathbf{z}_n^o]^T$ is the concatenation of the activations at the outermost hidden layer of the two models. Figure 1 demonstrates the fusion process with two simplified NNJMs: (*i*) the in-domain NNJM parameterized by $\theta^i = [E^i, U^i, W^i]$, and (*ii*) the out-domain NNJM parameterized by $\theta^o = [E^o, U^o, W^o]$.[1]

We train this model on the in-domain data using backpropagation on the NCE objective, where each participating model uses its own noise distribution. The errors (i.e., gradients) of the objective with respect to the final layer weight vectors $\mathbf{w}_j^d$ are:

$$\nabla_{\mathbf{w}_j^d} J(\theta) = \sum_{n=1}^{N} \Big[(y_{nj} - \sigma_{nj}) \mathbf{z}_n^d \Big] \tag{5}$$

where $d \in \{i, o\}$ in the superscript denotes the domain, $y_{nj} = I(y_n = j)$ is an indicator variable, and $\sigma_{nj} = \exp(\mathbf{w}_j^T \mathbf{z_n^d})$ is the unnormalized score in the output. We train the model by backpropagating these errors from the output layer of the neural network to the word embedding layer (i.e., look-up layer) of each model. Therefore, all the parameters of the participating models (i.e., E, U and W) are fine-tuned on the in-domain data. Such training yields adjusted models that are collectively optimized for the target in-domain data.

Fusion Model-II: A variation of the above model is to tune only the final layer combination weights $[\mathbf{w}_k^i, \mathbf{w}_k^o]$, which can be achieved by restricting the backpropagation only to the outermost hidden layer of the neural network models. This model is faster to train than the above model but could suffer from limited representation power.

3.2 Interpolation

Another approach we explored is to combine in- and out-domain NNJM models through linear interpolation. This approach has been extensively tried out in the literature to interpolate phrase-translation models (Foster and Kuhn, 2007). Several metrics such as tf/idf, LSA or perplexity have been employed for weighting. Here we interpolate multiple NNJM models instead. The mixture weights are computed by optimizing perplexity on the in-domain tuning set[2] using a standard EM-based algorithm as described below:

Model Weighting by EM: Let $\theta_d \in \{\theta_1, \ldots \theta_D\}$ represent an NNJM model trained on domain d, where D is the total number of domains. The probability of a sequence $\mathbf{x}_n$ can be written as a mixture of D probability densities, each coming from a different model:

[1] Although we define the fusion for two NNJM models, this can be easily generalized to multiple models.

[2] The tuning-set is required to be word-aligned and then converted into an augmented streams of source and target strings (for NNJM) to compute model-wise perplexities.

$$P(\mathbf{x}_n|\theta,\lambda) = \sum_{d=1}^{D} P(\mathbf{x}_n|z_n = d, \theta_d)\ \lambda_d$$

where $P(\mathbf{x}_n|z_n = d, \theta_d)$ represents the probability of $\mathbf{x}_n$ assigned by model θ_d, and the mixture weights λ_d satisfy $0 \leq \lambda_d \leq 1$ and $\sum_{d=1}^{D} \lambda_d = 1$. In our setting, $\theta = \{\theta_1, \ldots \theta_D\}$ is known, and we can use EM to learn the mixture weights. The expected complete data log likelihood is given by:

$$E[L(\lambda)] = \sum_{n=1}^{N}\sum_{d=1}^{D} r_{nd}\ log\left[P(\mathbf{x}_n|z_n = d, \theta_d)\lambda_d\right]$$

where $r_{nd} = P(z_n = d|\mathbf{x}_n, \theta_d, \lambda_d^{t-1})$ is the responsibility that domain d takes for data point n given the mixing weight in the previous step λ_d^{t-1}. In the E-step, we compute r_{nd} and we update λ in the M-step. More specifically:

E-step: Compute $r_{nd}^t = \frac{\lambda_d^{t-1} P(\mathbf{x}_n|z_n=d,\theta_d)}{\sum_{d'=1}^{D} \lambda_{d'}^{t-1} P(\mathbf{x}_n|z_n=d,\theta_{d'})}$

M-step: Update $\lambda_d^t = \frac{1}{N}\sum_{n=1}^{N} r_{nd}^t$

Once we have learned the relative weights of the models based on the in-domain tuning data, we can linearly interpolate the models as:

$$P_{nnjm}(T|S) \approx \prod_{i=1}^{|T|}\sum_{d} \lambda_d P(t_i|t_{i-1}\ldots t_{i-n+1}, \mathbf{s_i}, \theta_d)$$

An alternative way to combine the models is through log-linear interpolation by optimizing weights, directly on BLEU, along with other features inside of the SMT pipeline.

Note that both fusion and linear interpolation have the same number of parameters, which is the sum of the size of the base models. In fusion, we readjust all the parameters of the base models (or just the output layer weights in fusion-II), where in linear interpolation, we only learn their mixing weight.

4 Experiments

Data: We experimented with the data made available for the translation task of the International Workshop on Spoken Language Translation IWSLT (Cettolo et al., 2014). We used TED talks as our in-domain ($\approx$ 177K sentences) corpus. For Arabic-to-English, we used the multiUN ($\approx$ 3.7M sentences) (Eisele and Chen, 2010) as our out-domain corpora. For English-to-German, we used data made available ($\approx$ 4.4M sentences) for the 9^{th} Workshop on Machine Translation[3] as our out-domain data. Language models were trained on all the available monolingual data (English: $\approx$ 287.3M and German: $\approx$ 59.5M sentences). Machine translation systems were tuned on concatenation of the dev- and test2010 and evaluated on test2011-2013 datasets. We used *Farasa* (Abdelali et al., 2016) to tokenize Arabic and the default *Moses* tokenizer for English-and German. All data was truecased. See Table 1 for data sizes.

NN Training: The NNJM models were trained using the NPLM[4] toolkit (Vaswani et al., 2013) with the following settings: a target context of 5 words and an aligned source window of 9 words. We restricted source and target side vocabularies to the 20K and 40K most frequent words in the in-domain data.[5] The word vector size D and the hidden layer size were set to 150 and 750, respectively. Training was done using SGD with NCE using 100 noise samples and a mini-batch size of 1000. All models were trained for 15 epochs. Training NN models is expensive.[6] In the interest of time, we therefore reduced the NN

[3]http://www.statmt.org/wmt14/translation-task.html

[4]http://nlg.isi.edu/software/nplm/

[5]Less frequent words are mapped to `unk` class if they were found in the in-domain data or $\texttt{unk}_o$ otherwise. Please refer to (Joty et al., 2015) for how the vocabularies from in-domain and out-of-domain are mapped.

[6]Training model with the whole corpus requires roughly 12 days of wall-clock time (18 hours/epoch) to train NNJM models on our machines (on a Linux Ubuntu 12.04.5 LTS running on a 16 Core Intel Xeon E5-2650 2.00Ghz and 64Gb RAM). We ran a baseline experiment with all the data and did not find it better than the system trained on randomly selected subset.

English-German				Arabic-English			
Corpus	Sentences	Tok$_{EN}$	Tok$_{DE}$	Corpus	Sentences	Tok$_{AR}$	Tok$_{EN}$
iwslt	177K	3.5M	3.3M	iwslt	186K	2.7M	1.8M
news	200K	5.0M	5.1M	un	3.7M	12.4M	12.3M
ep	1.9M	51.0M	48.7M	-	-	-	-
cc	2.3M	57.5M	53.9M	-	-	-	-
Test Set	Sent.	Tok$_{EN}$	Tok$_{DE}$	Corpus	Sent.	Tok$_{AR}$	Tok$_{EN}$
tune	2437	51K	48K	tune	2456	48K	52K
test-11	1433	4K	23K	test-11	1199	21K	24K
test-12	1700	28K	26K	test-12	1702	30K	32K
test-13	993	18K	17K	test-13	1169	26K	28K

Table 1: Statistics of the English-German and Arabic-English training corpora in terms of Sentences and Tokens (represented in millions). ep = Europarl, cc = Common Crawl, un = United Nations

training to a subset of 1 million sentences containing all the in-domain data and a random selection of sentences from the out-domain data.

Machine Translation Settings: We trained a Moses system (Koehn et al., 2007), with the settings described in (Birch et al., 2014): a maximum sentence length of 80, Fast-Aligner for word-alignments (Dyer et al., 2013), an interpolated Kneser-Ney smoothed 5-gram language model (Heafield, 2011), lexicalized reordering model (Galley and Manning, 2008), a 5-gram operation sequence model (Durrani et al., 2013) and other defaults. We used k-best batch MIRA (Cherry and Foster, 2012) for tuning. Arabic OOVs were transliterated using unsupervised transliteration module (Durrani et al., 2014) in Moses.

Baselines: Baseline MT systems were trained by simply concatenating all the data. We included NNJM model trained on a plain concatenation of the data as a feature in our baseline system. In the adapted systems, we either replaced it with the NDAM models trained on weighted concatenation, or with our fusion models (NFM*), where models are trained independently and adjusted towards in-domain data or by interpolating them linearly (EM-weighting) or log-linearly. We also tried the approach of Luong and Manning (2015) by *Fine Tuning* baseline model towards in-domain data (i.e., by training the neural network model for additional epochs on in-domain data). We trained for 10 more epochs.

Phrase-table Adaptation: We also compared performance of our models against state-of-the-art model adaptation techniques available in Moses. Rather than training the MT systems on concatenated data, we train phrase-tables from in- and out-domain data separately and combine them through *Linear Phrase-table Interpolation*, *Instance Weighting* and *Fill-up* methods.[7]

Data Selection: Finally, we compared our system against MML-filtering (Axelrod et al., 2011), although this technique falls in the array of data-selection methods unlike our method which is a model weighting technique. We selected 0%, 2.5%, 5%, 10%, 20%, 40% and 100% out-domain data, and trained MT systems by concatenating the selected data with the in-domain data. The optimal thresholds were found to be 20% and only 5% in English-German and Arabic-English[8] respectively.

Comparing with Neural Model Weighting: The first row in Table 2 shows results for the baseline system, which includes NNJM trained on plain concatenation. The next set of rows show results for systems, where the NNJM model is adapted using weighted concatenation (NDAM), by interpolating (linearly or log-linearly) in- and out-domain models, by fine tuning the baseline model by running epochs on in-domain data only. This method gave significantly better results on the English-German task, but was not found effective in the Arabic-English direction. Next rows show results of our neural fusion

[7] Word alignment is still carried on the concatenated data.

[8] Sajjad et al. (2013) selected 3% of the UN data in their best competition grade system.

	English-to-German					Arabic-to-English				
System	tst11	tst12	tst13	Avg	Δ	tst11	tst12	tst13	Avg	Δ
Baseline (NNJM)	27.3	22.9	24.5	24.9		26.1	29.4	30.5	28.7	
NDAM	27.5	23.4	25.1	25.3	+0.4	26.1	29.6	30.9	28.9	+0.2
Linear	27.2	23.5	25.0	25.3	+0.4	26.7	30.2	30.3	29.1	+0.4
Log-Linear	27.0	23.8	25.2	25.3	+0.4	26.4	30.0	30.5	29.0	+0.3
Fine Tuning	27.7	23.9	25.3	25.6	+0.7	26.1	29.6	30.9	28.9	+0.2
NFM-I	27.8	24.1	25.6	25.8	+0.9	26.9	30.2	31.1	29.4	+0.7
NFM-II	27.5	23.9	25.4	25.6	+0.7	26.7	30.0	31.0	29.2	+0.5

Table 2: Comparing with Neural Model Weighting Methods – NDAM (Joty et al. 2015), Linear (Durrani et al. 2015), Fine Tuning (Luong and Manning 2015), NFM* = Neural Fusion Models (this work)

	English-to-German					Arabic-to-English				
System	tst11	tst12	tst13	Avg	Δ	tst11	tst12	tst13	Avg	Δ
Baseline (NNJM)	27.3	22.9	24.5	24.9		26.1	29.4	30.5	28.7	
PT Interpolation	27.5	23.2	24.8	25.2	+ 0.3	26.4	29.9	30.3	28.9	+0.2
Instance Wt.	27.3	23.4	25.1	25.3	+ 0.4	26.9	30.3	30.3	29.2	+0.5
Fill Up	27.3	23.4	24.6	25.1	+ 0.2	26.4	29.7	30.4	28.9	+0.2
NFM-I	27.8	24.1	25.6	25.8	+0.9	26.9	30.2	31.1	29.4	+0.7
NFM-I + Instance Wt.	28.0	23.8	25.3	25.7	+0.8	27.5	30.7	30.8	29.7	+1.0

Table 3: Comparing with Translation Model Adaptation Techniques – Linear Interpolation and Instance Weighting (Senrich 2012), Fill-up Method (Bisazza et al. 2011), NFM = Neural Fusion Model

models (NFM*). Our models significantly[9] outperformed the baseline system and were also found better than other neural adapted models. We found that our model gets higher weights than the baseline NNJM model (0.054 compared to 0.047 in En-De and 0.043 compared to 0.040 in Ar-En) in MERT training.

Fusion Model-I (NFM-I), which performs deeper fusion (i.e., till the embedding layer) worked better than Fusion Model-II (NFM-II), where backpropagation is restricted only to the out-most hidden layer, showing that there is an additional value in doing a deeper fusion.

Comparing with Phrase-table Weighting: Next we compare our model with the domain adaptation methods available in Moses (See Table 3). Here instead of adapting the NNJM model, we perform adaptation on translation-tables, by interpolating in- and out-domain phrase-tables, through instance weighting or through fill-up method. Instance weighting method gave best improvements among the lot, however, our fusion model outperform these methods in both language pairs and in most of the test-sets. Additional experiments combining phrase-table adaptation and fusion model found gains to be additive in Arabic-to-English language pair. But no further improvements were observed in English-to-German, except for test2011.

Comparing with Data Selection: In Table 4 we experiment with MML-based filtering and probe whether our model can also improve on top of data selection. Firstly, selecting no out-domain data degrades the English-to-German system. On the contrary, the Arabic-to-English system substantially improves. This shows that general domain data is useful for English-to-German and much of the out-domain data (UN corpus) used in these experiments is harmful in the case of Arabic-to-English. In comparison, data selection was found to be less useful in the case of English-to-German. But we found

[9] $p < 0.05$ using bootstrap resampling (Koehn, 2004), with 1000 samples.

	English-to-German				Arabic-to-English			
System	tst11	tst12	tst13	Avg	tst11	tst12	tst13	Avg
Baseline$_{cat}$	27.3	22.9	24.5	24.9	26.1	29.4	30.5	28.7
Baseline$_{ID}$	26.7	22.5	23.6	24.3	27.2	30.0	30.2	29.1
MML	26.9	22.9	24.4	24.7	27.4	30.8	30.9	29.7
+NFM-I	27.6	23.1	25.0	25.2	27.6	31.2	31.1	30.0

Table 4: Comparing with MML (Axelrod et al 2011)

that using our fusion model instead of baseline NNJM in either system still gave improvements (+0.5 and +0.3 in English-German and Arabic-English, respectively).

5 Related Work

Previous work on domain adaptation in MT can be broken down broadly into two main categories namely *data selection* and *model adaptation*.

Data selection has shown to be an effective way to discard poor quality or irrelevant training instances, which when included in an MT system, hurts its performance. Selection based methods can be helpful to reduce computational cost when training is expensive and also when memory is constrained. Data selection was done earlier for language modeling using information retrieval techniques (Hildebrand et al., 2005) and perplexity measures (Moore and Lewis, 2010). Axelrod et al. (2011) further extended the work of Moore and Lewis (2010) to translation model adaptation by using both source- and target-side language models. Duh et al. (2013) used a recurrent neural language model instead of an ngram-based language model to do the same. Translation model features were used recently by (Liu et al., 2014; Hoang and Sima'an, 2014) for data selection. Durrani et al. (2015) performed data selection using operation sequence model and NNJM models.

An alternative to completely filtering out less useful data is to minimize its effect by down-weighting it. It is more robust than selection since it takes advantage of the complete out-domain data with intelligent weighting towards the in-domain. Our work falls in this line of research. Matsoukas et al. (2009) proposed a classification-based sentence weighting method for adaptation. Foster et al. (2010) extended this by weighting phrases rather than sentence pairs. Other researchers have carried out weighting by merging phrase-tables through linear interpolation (Finch and Sumita, 2008; Nakov and Ng, 2009) or log-linear combination (Foster and Kuhn, 2009; Bisazza et al., 2011; Sennrich, 2012) and through phrase training based adaptation (Mansour and Ney, 2013). Chen et al. (2013) used a vector space model for adaptation at the phrase level. Every phrase pair is represented as a vector, where every entry in the vector reflects its relatedness with each domain.

Other work on domain adaptation includes but not limited to studies focusing on topic models (Eidelman et al., 2012; Hasler et al., 2014), dynamic adaptation without in-domain data (Sennrich et al., 2013; Mathur et al., 2014) and sense disambiguation (Carpuat et al., 2013).

6 Conclusion and Future Work

We presented a deep fusion model based on the neural network joint model (NNJM) of Devlin et al. (2014). The model is learned by fusing in- and out-domain NNJM models into a composite model by adjusting their parameters in favor of the in-domain data. When used as a feature during decoding, our model obtains statistically significant improvements on top of a competition grade phrase-based baseline system. We also showed improvements compared to previous adaptation methods. Further gains were obtained when our models were combined with existing methods. Although this study focused on fusing multiple neural models for domain adaptation in phrase-based SMT, the central idea can be adopted in the end-to-end NMT systems (Bahdanau et al., 2015), where the goal will be to fuse multiple NMT systems. We intend to explore this idea in our future work.

References

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–16, San Diego, California, June. Association for Computational Linguistics.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, Edinburgh, United Kingdom.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

Alexandra Birch, Matthias Huck, Nadir Durrani, Nikolay Bogoychev, and Philipp Koehn. 2014. Edinburgh SLT and MT system description for the IWSLT 2014 evaluation. In *Proceedings of the 11th International Workshop on Spoken Language Translation*, IWSLT '14, Lake Tahoe, CA, USA.

Arianna Bisazza, Nick Ruiz, and Marcello Federico. 2011. Fill-up versus interpolation methods for phrase-based SMT adaptation. In *Proceedings of the seventh International Workshop on Spoken Language Translation (IWSLT)*, pages 136–143.

Marine Carpuat, Hal Daume III, Katharine Henry, Ann Irvine, Jagadeesh Jagarlamudi, and Rachel Rudinger. 2013. Sensespotting: Never let your parallel data tie you to an old domain. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, August.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT Evaluation Campaign. *Proceedings of the International Workshop on Spoken Language Translation, Lake Tahoe, US*.

Boxing Chen, Roland Kuhn, and George Foster. 2013. Vector space model for adaptation in statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, August.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '12, Montréal, Canada.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Sofia, Bulgaria, August.

Nadir Durrani, Alexander Fraser, Helmut Schmid, Hieu Hoang, and Philipp Koehn. 2013. Can Markov models over minimal translation units help phrase-based SMT? In *ACL'13*, Sofia, Bulgaria.

Nadir Durrani, Hassan Sajjad, Hieu Hoang, and Philipp Koehn. 2014. Integrating an Unsupervised Transliteration Model into Statistical Machine Translation. In *Proceedings of the 15th Conference of the European Chapter of the ACL (EACL 2014)*, Gothenburg, Sweden, April.

Nadir Durrani, Hassan Sajjad, Shafiq Joty, Ahmed Abdelali, and Stephan Vogel. 2015. Using joint models for domain adaptation in statistical machine translation. In *Proceedings of the Fifteenth Machine Translation Summit (MT Summit XV)*, Florida, USA, To Appear. AMTA.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of NAACL'13*.

Vladimir Eidelman, Jordan Boyd-Graber, and Philip Resnik. 2012. Topic models for dynamic translation model adaptation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Jeju Island, Korea, July.

Andreas Eisele and Yu Chen. 2010. MultiUN: A Multilingual Corpus from United Nation Documents. In *Proceedings of the Seventh conference on International Language Resources and Evaluation*, Valleta, Malta, May.

Andrew Finch and Eiichiro Sumita. 2008. Dynamic model interpolation for statistical machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, Columbus, Ohio, June.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for smt. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07.

George Foster and Roland Kuhn. 2009. Stabilizing minimum error rate training. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, StatMT '09, Athens, Greece.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Cambridge, MA, October.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, and Takehito Utsuro. 2010. Overview of the patent translation task at the ntcir-8 workshop. In *In Proceedings of the 8th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-lingual Information Access*, pages 293–302.

Michel Galley and Christopher D. Manning. 2008. A Simple and Effective Hierarchical Phrase Reordering Model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 848–856, Honolulu, Hawaii, October.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Y.W. Teh and M. Titterington, editors, *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, volume 9 of *JMLR W&CP*, pages 297–304.

Francisco Guzmán, Hassan Sajjad, Stephan Vogel, and Ahmed Abdelali. 2013. The AMARA corpus: Building resources for translating the web's educational content. In *Proceedings of the 10th International Workshop on Spoken Language Technology (IWSLT-13)*, December.

Eva Hasler, Phil Blunsom, Philipp Koehn, and Barry Haddow. 2014. Dynamic topic adaptation for phrase-based mt. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, Gothenburg, Sweden, April.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July.

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Adaptation of the translation model for statistical machine translation based on information retrieval. In *Proceedings of the 10th Conference of the European Association for Machine Translation (EAMT)*, Budapest, May.

Cuong Hoang and Khalil Sima'an. 2014. Latent domain translation models in mix-of-domains haystack. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*.

Shafiq Joty, Hassan Sajjad, Nadir Durrani, Kamla Al-Mannai, Ahmed Abdelali, and Stephan Vogel. 2015. How to Avoid Unwanted Pregnancies: Domain Adaptation using Neural Network Models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, September.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Association for Computational Linguistics (ACL'07)*, Prague, Czech Republic.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, Barcelona, Spain.

Le Liu, Yu Hong, Hao Liu, Xing Wang, and Jianmin Yao. 2014. Effective selection of translation model training data. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Baltimore, Maryland, June.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September. Association for Computational Linguistics.

Saab Mansour and Hermann Ney. 2013. Phrase training based adaptation for statistical machine translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia, June.

Prashant Mathur, Sriram Venkatapathy, and Nicola Cancedda. 2014. Fast domain adaptation of smt models without in-domain parallel data. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, Dublin, Ireland, August.

Spyros Matsoukas, Antti-Veikko I. Rosti, and Bing Zhang. 2009. Discriminative corpus weight estimation for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2*, EMNLP '09.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the International Conference on Machine Learning*.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the Association for Computational Linguistics (ACL'10)*, Uppsala, Sweden.

Preslav Nakov and Hwee Tou Ng. 2009. Improved statistical machine translation for resource-poor languages using related resource-rich languages. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'09)*, Singapore.

Hassan Sajjad, Francisco Guzmn, Preslav Nakov, Ahmed Abdelali, Kenton Murray, Fahad Al Obaidli, and Stephan Vogel. 2013. QCRI at IWSLT 2013: Experiments in Arabic-English and English-Arabic spoken language translation. In *Proceedings of the 10th International Workshop on Spoken Language Technology (IWSLT-13)*, December.

Rico Sennrich, Holger Schwenk, and Walid Aransa. 2013. A multi-domain translation model framework for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Sofia, Bulgaria, August.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, August.

Rico Sennrich. 2012. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, Avignon, France, April.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Inducing Bilingual Lexica From Non-Parallel Data With Earth Mover's Distance Regularization

Meng Zhang[†‡] **Yang Liu**[†‡] **Huanbo Luan**[†] **Yiqun Liu**[†] **Maosong Sun**[†‡]
[†]State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[‡]Jiangsu Collaborative Innovation Center for Language Competence, Jiangsu, China
zmlarry@foxmail.com, liuyang2011@tsinghua.edu.cn
luanhuanbo@gmail.com, {yiqunliu, sms}@tsinghua.edu.cn

Abstract

Being able to induce word translations from non-parallel data is often a prerequisite for cross-lingual processing in resource-scarce languages and domains. Previous endeavors typically simplify this task by imposing the one-to-one translation assumption, which is too strong to hold for natural languages. We remove this constraint by introducing the Earth Mover's Distance into the training of bilingual word embeddings. In this way, we take advantage of its capability to handle multiple alternative word translations in a natural form of regularization. Our approach shows significant and consistent improvements across four language pairs. We also demonstrate that our approach is particularly preferable in resource-scarce settings as it only requires a minimal seed lexicon.

1 Introduction

Bilingual lexica provide word-level semantic equivalence information across languages, and prove to be valuable for a range of cross-lingual natural language processing tasks (Och and Ney, 2003; Levow et al., 2005; Täckström et al., 2013, *inter alia*). As building bilingual lexica from parallel corpora has been solved by word alignment (Och and Ney, 2003), researchers have turned their attention to non-parallel corpora. Accompanied by a small seed lexicon, non-parallel corpora are usually the only resources available in resource-scarce languages and domains, making the task of bilingual lexicon induction both important and challenging. A variety of statistical methods have been proposed to induce bilingual lexica from non-parallel data (Rapp, 1999; Koehn and Knight, 2002; Fung and Cheung, 2004; Gaussier et al., 2004; Haghighi et al., 2008; Ravi and Knight, 2011; Vulić et al., 2011; Vulić and Moens, 2013a; Vulić and Moens, 2013b; Dong et al., 2015). With the surge of word embeddings trained by neural networks, recent approaches that learn bilingual word representations from non-parallel data for bilingual lexicon induction have also shown promise (Mikolov et al., 2013b; Vulić and Moens, 2015).

However, none of the existing methods explicitly considers multiple alternative translation, i.e., the phenomenon that one source language word may have multiple possible translations in the target language. For example, the (romanized) Chinese word "*qiche*" can be translated to "car" or "automobile" in English, while the English word "car" can mean "*qiche*" or "*chexiang*" (railway carriage) in Chinese. Although prevalent among natural languages (Resnik and Yarowsky, 1999), multiple alternative translation is basically ignored by prior bilingual lexicon inducers; instead, they typically impose the one-to-one translation assumption (Vulić and Moens, 2013b) for simplicity. This represents a major drawback of existing bilingual lexicon induction approaches.

There has been one study that shows potential for tackling this issue. It introduces the Earth Mover's Distance (EMD) (Zhang et al., 2016). Given learned bilingual word embeddings, the EMD is used as a post-processing step to match vocabularies cross-lingually, which can be interpreted as word translation. Unlike the traditional K nearest neighbors leaving the determination of the number of translation proposals K to the user, the EMD automatically determines the list of translation candidates for each source word.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3188–3198, Osaka, Japan, December 11-17 2016.

Figure 1: An illustration of bilingual word embeddings for translating from (romanized) Chinese to English. Arrows indicate translations, and solid ones are correct. (a) The nearest neighbor incorrectly translates "*chexiang*" to "automobile", and does not allow finding "car" as the other translation of "*qiche*". (b) The Earth Mover's Distance translates correctly. It associates words with weights, as indicated by the sizes of the shapes.

In this work, we propose to bring the EMD's capability to training. Intuitively, as the EMD in the post-processing step is able to connect a source word with multiple target word translations, it can play a more important role during training by driving the word vectors of these mutual translations to be closer. We therefore expect that the bilingual word embeddings learned this way will be more suitable for encoding multiple alternative translation by harnessing the power of the EMD. Our experiments validate the effectiveness of this strategy. A summary of our contributions is as follows:

- We introduce the Earth Mover's Distance into the training of bilingual word embeddings, and interpret it as a natural form of regularization for the overall learning objective (Section 3).
- We demonstrate significant and consistent performance improvement from our strategy across four language pairs (Sections 6.1 and 6.2).
- We investigate the effect of the number of seed word translation pairs, and find our approach to be most appealing with few seeds, in line with typical resource-scarce scenarios (Section 6.3).

2 Background

As an embedding-based approach to bilingual lexicon induction, the model consists of matrices $W^{\mathrm{S}} \in \mathbb{R}^{D \times V^{\mathrm{S}}}$ and $W^{\mathrm{T}} \in \mathbb{R}^{D \times V^{\mathrm{T}}}$, which pack up D-dimensional word embeddings of source and target languages with vocabulary sizes V^{S} and V^{T}, respectively. After training, these bilingual word embeddings are supposed to properly lie in the D-dimensional space that encodes cross-lingual semantic equivalence. To build a bilingual lexicon, or equivalently, to translate a source word into the target language, the nearest neighbor is typically employed to retrieve the target word embedding that is closest to the source word embedding.

The nearest neighbor has its limitations, as argued by Zhang et al. (2016). For one thing, the retrieval operation is inherently local (Figure 1(a)). Instead, they introduce the Earth Mover's Distance (EMD), which offers to match two sets of points with minimum total cost. For the word translation task, bilingual word embeddings can be naturally viewed as two sets of points. Therefore, given bilingual word embeddings, the EMD can perform word translation by providing optimal vocabulary-level matching in a holistic fashion (Figure 1(b)). This is achieved via the following optimization program:

$$\begin{aligned} &\min_{T} \sum_{t=1}^{V^{\mathrm{T}}} \sum_{s=1}^{V^{\mathrm{S}}} T_{ts} C_{ts} \\ &s.t.\ T_{ts} \geq 0 \\ &\quad \sum_{s=1}^{V^{\mathrm{S}}} T_{ts} \leq f_t^{\mathrm{T}}, t \in \{1, ..., V^{\mathrm{T}}\} \\ &\quad \sum_{t=1}^{V^{\mathrm{T}}} T_{ts} = f_s^{\mathrm{S}}, s \in \{1, ..., V^{\mathrm{S}}\} \end{aligned} \tag{1}$$

where C_{ts} defines the cost of matching the target word w_t^{T} and the source word w_s^{S} (illustrated by the distance between words in Figure 1), and f_t^{T} (resp. f_s^{S}) is the weight associated with w_t^{T} (resp. w_s^{S}) (illustrated by the sizes of the shapes in Figure 1(b)). The weights are chosen to be the number of times a word appears in the corpus. Once the linear program is solved, the matrix T stores the matching information between source and target vocabularies. This cross-lingual matching can be interpreted as translation. For example, a non-zero T_{ts} can be seen as evidence to translate the source word w_s^{S} to the target word w_t^{T}.

Besides the vocabulary-level matching, the EMD program brings an additional benefit. As mentioned in Section 1, it automatically retrieves multiple translations for a source word as long as the program finds it appropriate (cf. Figure 1). In the following section, we will strengthen this desirable capability by bringing the EMD program from a post-processing step to the training phase.

3 Approach

In typical scenarios, resources available to bilingual lexicon inducers include non-parallel corpora $\mathcal{C}^{\mathrm{S}}$ and $\mathcal{C}^{\mathrm{T}}$, and a seed lexicon $\mathbf{d}$. In order to utilize these resources to train bilingual word embeddings, a straightforward idea is to devise a learning objective that combines a monolingual term and a seed term.

The monolingual term $\mathcal{J}_{\mathrm{mono}}$ is responsible for explaining regularities in corpora $\mathcal{C}^{\mathrm{S}}$ and $\mathcal{C}^{\mathrm{T}}$. Since the two corpora are non-parallel, $\mathcal{J}_{\mathrm{mono}}$ consists of two monolingual submodels that are independent of each other:

$$\mathcal{J}_{\mathrm{mono}}\left(W^{\mathrm{S}}, W^{\mathrm{T}}\right) = \mathcal{J}_{\mathrm{mono}}^{\mathrm{S}}\left(W^{\mathrm{S}}\right) + \mathcal{J}_{\mathrm{mono}}^{\mathrm{T}}\left(W^{\mathrm{T}}\right). \tag{2}$$

As the common practice (Gouws et al., 2015), we choose the well established skip-gram model (Mikolov et al., 2013a) for our monolingual term.

The seed term $\mathcal{J}_{\mathrm{seed}}$ encourages embeddings of word translation pairs in a seed lexicon $\mathbf{d}$ to move near, which can be achieved via a L_2 regularizer:

$$\mathcal{J}_{\mathrm{seed}}\left(W^{\mathrm{S}}, W^{\mathrm{T}}\right) = -\sum_{\langle s,t\rangle \in \mathbf{d}} \left\|W_s^{\mathrm{S}} - W_t^{\mathrm{T}}\right\|^2, \tag{3}$$

where $s \in \left\{1, ..., V^{\mathrm{S}}\right\}$ and W_s^{S} is the s-th column of W^{S} (i.e. the embedding of the s-th source word w_s^{S}), and notations are similar for the target side.

However, as shown in our experiment, a simple linear combination of the monolingual term and the seed term is insufficient to provide satisfactory performance. We propose to introduce the Earth Mover's Distance into the training phase, as an additional term in the learning objective:

$$\mathcal{J}_{\mathrm{EMD}}\left(W^{\mathrm{S}}, W^{\mathrm{T}}, T\right) = -\sum_{t=1}^{V^{\mathrm{T}}}\sum_{s=1}^{V^{\mathrm{S}}} T_{ts}C_{ts} \tag{4}$$

with constraints

$$\begin{gathered} T_{ts} \geq 0 \\ \sum_{s=1}^{V^{\mathrm{S}}} T_{ts} \leq f_t^{\mathrm{T}}, t \in \left\{1, ..., V^{\mathrm{T}}\right\} \\ \sum_{t=1}^{V^{\mathrm{T}}} T_{ts} = f_s^{\mathrm{S}}, s \in \left\{1, ..., V^{\mathrm{S}}\right\} \end{gathered}. \tag{5}$$

Note that, unlike the post-processing case (1), the ground distance matrix C is now parametrized by bilingual embeddings W^{S} and W^{T}, and therefore adjustable during training.

Putting everything together, we arrive at our overall learning objective to maximize:

$$\mathcal{J}\left(W^{\mathrm{S}}, W^{\mathrm{T}}, T\right) = \mathcal{J}_{\mathrm{mono}}\left(W^{\mathrm{S}}, W^{\mathrm{T}}\right) + \lambda_{\mathrm{s}}\mathcal{J}_{\mathrm{seed}}\left(W^{\mathrm{S}}, W^{\mathrm{T}}\right) + \lambda_{\mathrm{e}}\mathcal{J}_{\mathrm{EMD}}\left(W^{\mathrm{S}}, W^{\mathrm{T}}, T\right) \tag{6}$$

with constraints (5) inherited from the EMD. The hyperparameters λ_{s} and λ_{e} control the relative weighting of the terms. In this form, we can naturally view the EMD term as a regularizer that can potentially

drive the embedding space to be more suitable for inducing bilingual lexica, especially multiple alternative word translation pairs.

The joint maximization of the overall learning objective (6) is clearly non-convex. In order to take advantage of the efficient solver specialized for the EMD program, we propose an alternating optimization procedure:

1. Fix W^{S} and W^{T}, and optimize with respect to T. This reduces to the usual linear EMD program with fixed ground distance, and the optimization can be achieved with the existing solver.

2. Fix T, and optimize with respect to W^{S} and W^{T}. Now the optimization can be easily achieved with stochastic gradient ascent.

4 Implementation

In this section, we describe details of a practical implementation of our approach.

4.1 Optimization

In our overall learning objective (6), unlike the other two, the EMD term $\mathcal{J}_{\mathrm{EMD}}$ requires an alternating optimization procedure. In order to allow each term to contribute to the learning process, we follow these steps. First, in each pass of the corpus (i.e. an epoch) the monolingual term $\mathcal{J}_{\mathrm{mono}}$ and the seed term $\mathcal{J}_{\mathrm{seed}}$ are optimized with asynchronous stochastic gradient ascent (Gouws et al., 2015). Then, we proceed to optimize the EMD term $\mathcal{J}_{\mathrm{EMD}}$ with the alternating optimization procedure. In Step 2 of the procedure, we take M gradient ascent steps. This hyperparameter is related to λ_{e}, as they jointly affect the strength of the EMD regularization. We are inclined to take small and many gradient ascent steps, so we fix $M = 10,000$ and tune λ_{e} on the validation set. Finally, the learning rate is decayed linearly at the end of each epoch.

4.2 Adding Context Vectors

In the previous section, we have presented our model with word vectors W^{S} and W^{T} as the parameters. In reality, each word is associated with a context vector as well (Mikolov et al., 2013c). While the usual representation of a word for evaluation is simply a word vector, some authors have suggested adding the context vector (Pennington et al., 2014; Levy et al., 2015). Previously this means a simple post-processing step during evaluation, but in our setting we can bring the trick to training. Specifically, using Euclidean distance as the ground distance, we would have parametrized C_{ts} in the EMD term (4) as

$$C_{ts} = \left\| W_t^{\mathrm{T}} - W_s^{\mathrm{S}} \right\|. \tag{7}$$

Considering the context vectors U^{S} and U^{T}, we now reformulate the ground distance as

$$C_{ts} = \left\| \left(W_t^{\mathrm{T}} + U_t^{\mathrm{T}}\right) - \left(W_s^{\mathrm{S}} + U_s^{\mathrm{S}}\right) \right\|. \tag{8}$$

This modification affects both steps in the alternating optimization procedure. In addition, the seed term also encourages corresponding context vectors to be close.

5 Experimental Setup

5.1 Data

In our experiments, the tested systems induce bilingual lexica from Wikipedia comparable corpora[1] on four language pairs: Chinese-English, Spanish-English, Italian-English, and Japanese-Chinese. Following (Vulić and Moens, 2013a), we retain only nouns that occur at least 1,000 times in our corpora.[2] For the Chinese side, we first use OpenCC[3] to normalize characters to be simplified, and then perform

[1] http://linguatools.org/tools/corpora/wikipedia-comparable-corpora

[2] For Spanish-English and Italian-English, the cut-off frequency is 3,000 for a comparably-sized vocabulary.

[3] https://github.com/BYVoid/OpenCC

	zh-en		es-en		it-en		ja-zh	
	zh	en	es	en	it	en	ja	zh
# tokens	21M	53M	57M	90M	65M	88M	38M	16M
vocabulary size	3,349	5,154	2,543	3,557	3,378	3,534	6,043	2,814

Table 1: Training set statistics. Language codes: zh = Chinese, en = English, es = Spanish, it = Italian, ja = Japanese.

	zh-en	es-en	it-en	ja-zh
# test instances	1,938	1,860	2,051	2,320
# with multiple alternative translation	661	1,293	1,338	513

Table 2: Statistics of the test sets obtained by processing the gold standard lexica in the same way as (Zhang et al., 2016). A good portion of the test instances come with multiple alternative translation in the ground truth.

Chinese word segmentation and POS tagging with THULAC[4]. The preprocessing of the English side involves tokenization, POS tagging, lemmatization, and lowercasing, which we carry out with the NLTK toolkit[5] for the Chinese-English pair. For Spanish-English and Italian-English, we choose to use TreeTagger[6] for preprocessing, as in (Vulić and Moens, 2013a). For the Japanese corpus, we use MeCab[7] for word segmentation and POS tagging. The statistics of the preprocessed corpora is given in Table 1.

5.2 Seed Word Translation Pairs

We build our seed lexicon in a way similar to (Vulić and Moens, 2013a). First, we ask Google Translate[8] to translate the source side vocabulary. Then the translations in the target language are queried again in the reverse direction to translate back to the source language, and those that don't match with the original source words are discarded. This helps to ensure the quality of the translations. Finally, a translation pair is discarded if the target word falls out of our target vocabulary. We then take the most frequent S translation pairs as the seed lexicon. We vary S in our experiment to examine the effect of the seed lexicon size.

5.3 Evaluation Method

The limiting factor that prevents us from experimenting with truly resource-scarce language pairs is the unavailability of gold standard lexica for evaluation. Our focus on multiple alternative translation raises a higher demand that the gold standard lexica should include multiple possible translations for source words. For Chinese-English, we use Chinese-English Translation Lexicon Version 3.0[9] as the gold standard. For Spanish-English and Italian-English, we access Open Multilingual WordNet[10] through NLTK. For Japanese-Chinese, we use an in-house lexicon that meets our need. We reserve 10% of each gold standard lexicon for validation, and the remaining 90% for testing. We list test set statistics for each language pair in Table 2.

Following (Zhang et al., 2016), our evaluation metrics include accuracy A, precision P, recall R, and F_1 score. Accuracy is traditionally reported for the bilingual lexicon induction task, but it does not reflect the handling of multiple translations. This evaluative tradition also proves the lack of attention for multiple alternative translation. Therefore, we will be primarily looking at F_1 score in our experiments.

[4]http://thulac.thunlp.org
[5]http://www.nltk.org
[6]http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger
[7]http://taku910.github.io/mecab
[8]https://translate.google.com
[9]https://catalog.ldc.upenn.edu/LDC2002L27
[10]http://compling.hss.ntu.edu.sg/omw

Method	A	P	R	F_1
STAT	0.2430	0.1589	0.1594	0.1591
MONO+SEED	0.2652	0.1983	0.1747	0.1858
Ours	0.5134	0.3770	0.3385	0.3567

Table 3: Performance on Chinese-English lexicon induction with 100 seed word translation pairs.

Method	Spanish-English	Italian-English	Japanese-Chinese
STAT	0.2384	0.2222	0.2117
MONO+SEED	0.2705	0.2350	0.1952
Ours	0.3686	0.3452	0.4111

Table 4: F_1 scores for three language pairs with 100 seed word translation pairs.

5.4 Baselines

We compare our approach to two baselines:

1. Statistics-based (STAT) (Gaussier et al., 2004).

2. Monolingual and seed terms (MONO+SEED).

The first baseline (STAT) is the traditional statistics-based approach, conventionally considered the standard approach to bilingual lexicon induction (Gaussier et al., 2004). It represents each word with a vector that encodes association strength between the word and seed words. We use a smoothed version of positive pointwise mutual information (PPMI) (Turney and Pantel, 2010) as the monolingual association measure.

The second baseline (MONO+SEED) is our system without the EMD term (i.e. $\lambda_{\mathrm{e}} = 0$). Comparison with it allows us to observe the effectiveness of the EMD term.

As we focus on multiple alternative translation but existing methods do not address it, we post-process the baselines by the EMD procedure (Zhang et al., 2016) to grant them the desired capability for a fair comparison with our approach.[11]

5.5 Hyperparameters

Our approach inherits hyperparameters from the monolingual skip-gram model, and includes term weights λ_{s} and λ_{e}. We set these hyperparameters based on tuning on the validation set, and observe little performance difference as long as they lie within a reasonable range. The monolingual hyperparameters are set as follows: embedding size D is 40; window size is 5; 5 negative samples; subsampling threshold is 10^{-5}; initial learning rate is 0.02; 20 training epochs. The statistics-based baseline uses a window size of 5 as well. The seed term weight λ_{s} is set to 0.01, and the EMD term weight λ_{e} is 0.0001.

6 Results

6.1 Performance on Chinese-English

We first report experimental results on Chinese-English lexicon induction with 100 seed word translation pairs, as shown in Table 3. We observe significant performance gains over both baselines, as measured by all evaluation metrics. In particular, comparing our approach with the MONO+SEED baseline highlights the effectiveness of introducing the EMD program into the training phase. As for training time, our approach takes about 4 hours, compared to 2 hours of MONO+SEED, due to the introduction of the EMD regularization.

[11] We found EMD post-processing to be generally superior to nearest neighbors, in line with (Zhang et al., 2016).

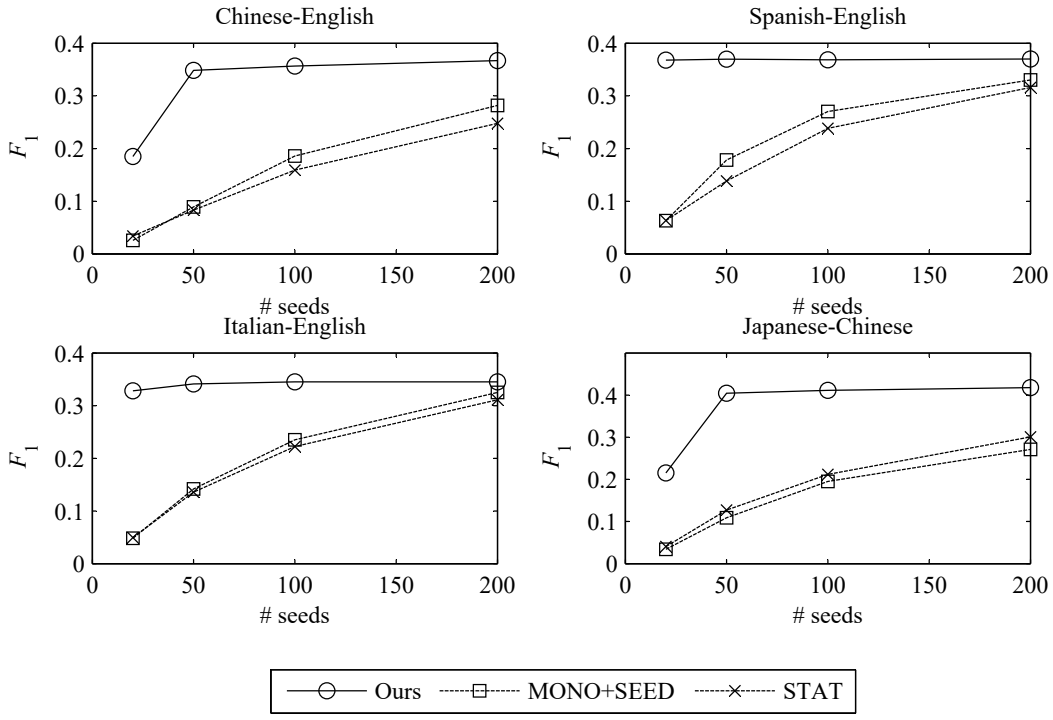


Figure 2: F_1 scores for the four language pairs with varying number of seed word translation pairs.

6.2 Performance on Other Language Pairs

We next experiment with the other three language pairs, i.e., Spanish-English, Italian-English, and Japanese-Chinese. The tested systems are provided with 100 seed word translation pairs as well. We only report the resulting F_1 scores in Table 4, as the other evaluation metrics exhibit similar trends. Relative to Chinese-English, these three language pairs should be more closely related. Nevertheless, the improvements of our method remain large, regardless of language pairs. The consistent performance signifies the generalizability of our approach across different language pairs.

6.3 Effect of Seed Lexicon Size

In this section, we investigate how the number of seed word translation pairs may affect the performance of the bilingual lexicon inducers. We vary the seed lexicon size in $\{20, 50, 100, 200\}$. Figure 2 shows the F_1 scores of the tested systems for the four language pairs. We observe that our system always attains high performance for the closely related language pairs Spanish-English and Italian-English, even when the seeds are as few as 20. For the more distant language pairs Chinese-English and Japanese-Chinese, 50 seeds suffice. In contrast, a limited number of seeds considerably degrades the performance of the baseline systems. Therefore, our system is particularly appealing in realistic resource-scarce scenarios for its minimal requirement for a seed lexicon, which is labor-intensive to compile.

6.4 Qualitative Analysis

In order to obtain a clearer view of the difference between the tested systems, we probe into the embeddings trained by them through a few examples. The Chinese-English translations in Table 5 imply that embeddings trained by our method appear superior. Although the two baselines may output more translations than our system, they often miss the correct ones, as shown by the examples of "*shan*" and "*jianzhu*". These translation differences should be eventually attributed to the quality of the underlying bilingual word embeddings, and in turn to the performance of the systems.

	STAT	MONO+SEED	Ours
qiche	good maker	**automobile** competitor customer luxury	**car** **automobile** **auto**
shan	palm flat waterfall dune barley chestnut citrus	middle part	**hill** **mountain**
jianzhu	architecture monument	foundation interior brick onwards	**building**

Table 5: English translations of three (romanized) Chinese words by the tested systems. The correct translations are in bold. The number of translations in each cell varies because it is automatically determined by the EMD program.

7 Related Work

Following its monolingual counterpart (Mikolov et al., 2013c, *inter alia*), bilingual word representation learning has attracted considerable attention. However, most of the works require parallel data as the cross-lingual signal (Zou et al., 2013; Chandar A P et al., 2014; Hermann and Blunsom, 2014; Kočiský et al., 2014; Gouws et al., 2015; Luong et al., 2015; Coulmance et al., 2015), making them unsuitable for bilingual lexicon induction. Although a few exceptions exist (Mikolov et al., 2013b; Faruqui and Dyer, 2014; Lu et al., 2015; Vulić and Moens, 2015; Shi et al., 2015; Gouws and Søgaard, 2015; Wick et al., 2016; Ammar et al., 2016), they lack a mechanism to deal with the multiple alternative translation prevalent cross-lingually.

The multiple alternative translation across languages is rooted in the polysemy of words within languages. In the monolingual setting, word sense disambiguation stands with a long line of research (Agirre and Rigau, 1996, *inter alia*). Since the advent of word representation learning, there have been some attempts to learn multiple vectors for a word, each dedicated to a single sense of the word, and therefore known as "sense embedding".

Existing sense embeddings can be roughly divided into two categories, depending on whether external resources are utilized. For those that do not rely on external resources, their main idea is to employ unsupervised methods like clustering to differentiate between multiple senses (Reisinger and Mooney, 2010; Huang et al., 2012; Neelakantan et al., 2014; Tian et al., 2014; Li and Jurafsky, 2015). For those that do, they typically retrofit existing word vectors to sense inventories (Jauhar et al., 2015; Rothe and Schütze, 2015), or use the resources to obtain a word sense disambiguation system, and then use it to disambiguate words, so that word representation learning methods can be applied (Chen et al., 2014; Iacobacci et al., 2015). An exception is the work of (Guo et al., 2014). Their external resource is parallel data. They observe that different senses of a word usually have different translations, so disambiguation can be thus achieved.

However, no prior research has shown how to connect sense embeddings cross-lingually, unless multilingual lexical ontologies exist (Camacho-Collados et al., 2015). For bilingual lexicon induction, where only non-parallel data and a seed lexicon are available, it is unclear whether sense embeddings can address multiple alternative translation.

Our work complements (Zhang et al., 2016): Their work applies the Earth Mover's Distance to the post-processing of fixed bilingual word embeddings to retrieve word translation, while ours strives to

train better bilingual word embeddings with the EMD. In addition, we also explore the feasibility of using the EMD for bilingual lexicon induction from non-parallel data. In computer vision, there have been a few works that experiment with trainable ground distance in the EMD program (Wang and Guibas, 2012; Zen et al., 2014). However, they require supervision to properly guide the training. With supervision, their EMD program can stand alone to fit training data, while in our approach the EMD shows up as a regularizer in the learning objective. Besides, their models fully parametrize the ground distance as optimizable variables, whereas our model treats it as the Euclidean distance with adjustable word vectors.

8 Conclusion

In this paper, we look into multiple alternative translations prevalent across natural languages, which are largely neglected in previous bilingual lexicon induction research. We propose to introduce the Earth Mover's Distance into the training of bilingual word embeddings as a natural form of regularization. We provide strong empirical results for four language pairs to demonstrate the effectiveness of our approach. Furthermore, we discover that our method remains reliable with rather few seed word translation pairs, unlike the baselines exhibiting performance degradation. This advantage of our approach is particularly desirable in realistic resource-scarce settings.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. This work is supported by the 863 Program (2015AA011808), the National Natural Science Foundation of China (No. 61522204, No. 61432013 and No. 61303075), and Toshiba Corporation Corporate Research & Development Center.

References

Eneko Agirre and German Rigau. 1996. Word Sense Disambiguation Using Conceptual Density. In *COLING*.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively Multilingual Word Embeddings. In *arXiv:1602.01925 [cs]*.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. A Unified Multilingual Semantic Representation of Concepts. In *ACL-IJCNLP*.

Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An Autoencoder Approach to Learning Bilingual Word Representations. In *NIPS*.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A Unified Model for Word Sense Representation and Disambiguation. In *EMNLP*.

Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. 2015. Trans-gram, Fast Cross-lingual Word-embeddings. In *EMNLP*.

Meiping Dong, Yang Liu, Huanbo Luan, Maosong Sun, Tatsuya Izuha, and Dakun Zhang. 2015. Iterative Learning of Parallel Lexicons and Phrases from Non-Parallel Corpora. In *IJCAI*.

Manaal Faruqui and Chris Dyer. 2014. Improving Vector Space Word Representations Using Multilingual Correlation. In *EACL*.

Pascale Fung and Percy Cheung. 2004. Mining Very-Non-Parallel Corpora: Parallel Sentence and Lexicon Extraction via Bootstrapping and EM. In *EMNLP*.

Eric Gaussier, J.M. Renders, I. Matveeva, C. Goutte, and H. Dejean. 2004. A Geometric View on Bilingual Lexicon Extraction from Comparable Corpora. In *ACL*.

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *NAACL-HLT*.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast Bilingual Distributed Representations without Word Alignments. In *ICML*.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning Sense-specific Word Embeddings By Exploiting Bilingual Resources. In *COLING*.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning Bilingual Lexicons from Monolingual Corpora. In *ACL-HLT*.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual Distributed Representations without Word Alignment. In *ICLR*.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *ACL*.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning Sense Embeddings for Word and Relational Similarity. In *ACL-IJCNLP*.

Sujay Kumar Jauhar, Chris Dyer, and Eduard Hovy. 2015. Ontologically Grounded Multi-sense Representation Learning for Semantic Vector Space Models. In *NAACL-HLT*.

Philipp Koehn and Kevin Knight. 2002. Learning a Translation Lexicon from Monolingual Corpora. In *ACL Workshop on Unsupervised Lexical Acquisition*.

Tomáš Kočiský, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning Bilingual Word Representations by Marginalizing Alignments. In *ACL*.

Gina-Anne Levow, Douglas W. Oard, and Philip Resnik. 2005. Dictionary-based techniques for cross-language information retrieval. *Information Processing & Management*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *TACL*.

Jiwei Li and Dan Jurafsky. 2015. Do Multi-Sense Embeddings Improve Natural Language Understanding? In *EMNLP*.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep Multilingual Correlation for Improved Word Embeddings. In *NAACL-HLT*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual Word Representations with Monolingual Quality in Mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *ICLR Workshop*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting Similarities among Languages for Machine Translation. In *arXiv:1309.4168 [cs]*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013c. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space. In *EMNLP*.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *CL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*.

Reinhard Rapp. 1999. Automatic Identification of Word Translations from Unrelated English and German Corpora. In *ACL*.

Sujith Ravi and Kevin Knight. 2011. Deciphering Foreign Language. In *ACL-HLT*.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-Prototype Vector-Space Models of Word Meaning. In *NAACL-HLT*.

Philip Resnik and David Yarowsky. 1999. Distinguishing systems and distinguishing senses: new evaluation methods for Word Sense Disambiguation. *Natural Language Engineering*.

Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending Word Embeddings to Embeddings for Synsets and Lexemes. In *ACL-IJCNLP*.

Tianze Shi, Zhiyuan Liu, Yang Liu, and Maosong Sun. 2015. Learning Cross-lingual Word Embeddings via Matrix Co-factorization. In *ACL-IJCNLP*.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and Type Constraints for Cross-Lingual Part-of-Speech Tagging. *TACL*.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A Probabilistic Model for Learning Multi-Prototype Word Embeddings. In *COLING*.

Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *JAIR*.

Ivan Vulić and Marie-Francine Moens. 2013a. Cross-Lingual Semantic Similarity of Words as the Similarity of Their Semantic Word Responses. In *NAACL-HLT*.

Ivan Vulić and Marie-Francine Moens. 2013b. A Study on Bootstrapping Bilingual Vector Spaces from Non-Parallel Data (and Nothing Else). In *EMNLP*.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual Word Embeddings from Non-Parallel Document-Aligned Data Applied to Bilingual Lexicon Induction. In *ACL-IJCNLP*.

Ivan Vulić, Wim De Smet, and Marie-Francine Moens. 2011. Identifying Word Translations from Comparable Corpora Using Latent Topic Models. In *ACL-HLT*.

Fan Wang and Leonidas J. Guibas. 2012. Supervised Earth Mover's Distance Learning and Its Computer Vision Applications. In *ECCV*.

Michael Wick, Pallika Kanani, and Adam Pocock. 2016. Minimally-Constrained Multilingual Embeddings via Artificial Code-Switching. In *AAAI*.

Gloria Zen, Elisa Ricci, and Nicu Sebe. 2014. Simultaneous Ground Metric Learning and Matrix Factorization with Earth Mover's Distance. In *International Conference on Pattern Recognition*.

Meng Zhang, Yang Liu, Huanbo Luan, Maosong Sun, Tatsuya Izuha, and Jie Hao. 2016. Building Earth Mover's Distance on Bilingual Word Embeddings for Machine Translation. In *AAAI*.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual Word Embeddings for Phrase-Based Machine Translation. In *EMNLP*.

What Makes Word-level Neural Machine Translation Hard: A Case Study on English-German Translation

Fabian Hirschmann **Jinseok Nam** **Johannes Fürnkranz**
Knowledge Engineering Group
Technische Universität Darmstadt
Darmstadt, Germany

Abstract

Traditional machine translation systems often require heavy feature engineering and the combination of multiple techniques for solving different subproblems. In recent years, several end-to-end learning architectures based on recurrent neural networks have been proposed. Unlike traditional systems, Neural Machine Translation (NMT) systems learn the parameters of the model and require only minimal preprocessing. Memory and time constraints allow to take only a fixed number of words into account, which leads to the out-of-vocabulary (OOV) problem. In this work, we analyze why the OOV problem arises and why it is considered a serious problem in German. We study the effectiveness of compound word splitters for alleviating the OOV problem, resulting in a 2.5+ BLEU points improvement over a baseline on the WMT'14 German-to-English translation task. For English-to-German translation, we use target-side compound splitting through a special syntax during training that allows the model to merge compound words and gain 0.2 BLEU points.

1 Introduction

In the field of machine translation, traditional methods depend on a careful design of useful features obtained by analyzing linguistic properties of the translation tasks. Inspired by recent success of methods that learn vector representation of words from large corpora in an unsupervised way, Devlin et al. (2014) have shown that word representations learned by neural language models can be effectively used as components in a complex translation system. By contrast, *Neural Machine Translation (NMT)* models are end-to-end learning systems that tune model parameters so that the desired output sentence is generated for a given input sentence of arbitrary length with minimal processing steps such as tokenization. Most of recently proposed NMT models are based on the encoder-decoder framework (Sutskever et al., 2014; Cho et al., 2014), where a source sentence is mapped into a fixed-size vector that preserves both the syntactic and semantic structure of the source sentence, from which a decoder starts with generating a sequence of words in a target language. Bahdanau et al. (2015) extend the vanilla encoder-decoder NMT framework by adding a small feed-forward neural network which learns which word in the source sentence is relevant for predicting the next word in the target sequence. It has been shown that the performance of such *soft-attention* NMTs stays consistent as the sequence length increases.

Although NMT models have been applied successfully to translation tasks, it is still challenging to handle *out-of-vocabulary (OOV)* words because only a small number of words are considered to reduce computational overhead. All words not in the vocabulary are assigned to a single special token, e.g., $\langle$UNK$\rangle$ in our case. In order to address the OOV problem, Jean et al. (2015) further extend the model of Bahdanau et al. (2015) with importance sampling so that it can hold a larger vocabulary without increasing training complexity. Luong et al. (2015) address the OOV problem by looking up unknown words in an automatically generated dictionary, and use an external word aligner to map words in the target sequence to ones in the source sequence.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3199–3208, Osaka, Japan, December 11-17 2016.

Regarding OOV words, a fundamental problem of NMT models that take words as inputs is that having a larger vocabulary cannot be a solution for morphologically rich languages such as German and Czech due to the proliferation of word forms resulting from the addition of affixes to word stems. This means that even though we include millions of words in the vocabulary, quite a significant ratio of words remains unknown tokens. We will discuss such properties of German in Section 3.

Therefore, instead of using words as only inputs and outputs, there have been several approaches which take into account more fine-grained units such as characters. Based on the fact that rare words are often composed of frequent words, Sennrich et al. (2016) build a vocabulary of subunits instead of words and view a sentence as a sequence of subunits. Such a vocabulary typically includes meaningful linguistic units, e.g., prefixes, suffixes and frequent stems, but it may also include less meaningful units because the vocabulary building process solely relies on their frequencies. While learning directly from characters of both source and target sentences is interesting since the OOV problem is not an issue anymore, it is more difficult and time-consuming to train such models (Ling et al., 2015). As a compromise, (Luong and Manning, 2016) propose a hybrid model that uses both words and characters. Similarly, (Chung et al., 2016) suggest the use of a character-level decoder which takes words or subunits as inputs but outputs character sequences.

According to the recent developments in NMT, it is highly likely that fine-grained units enable us to get better neural translation models. In this paper we, hence, explore the problems that are typically encountered by word-level neural translation systems, how we can avoid such problems and what remains problematic. We will start with a brief explanation of the word-level NMT architecture proposed by (Bahdanau et al., 2015) in the next section.

2 Background: Neural Machine Translation

Consider that we have a pair of a source sentence and its translation in the target language, e.g., (*Wie geht es dir?*, *How are you?*) when translating German sentences to English. If a word is treated as an atomic unit of translation, we can define both source and target sentences as a sequence of words such that $\boldsymbol{x} = \{x_j\}_{j=1}^{Tx}$ and $\boldsymbol{y} = \{y_i\}_{i=1}^{Ty}$ where T_x is the number of words in the source sentence and T_y is that of the target words. The goal of learning translation models can be expressed as maximizing the joint probability $p\left(\boldsymbol{y}|\boldsymbol{x}\right)$ of the target words $\boldsymbol{y} = \{y_1, \cdots, y_{T_y}\}$ conditioned on the source words $\boldsymbol{x} = \{x_1, \cdots, x_{T_x}\}$. We can factorize it into a product of conditional probabilities of a single target word:

$$p\left(\boldsymbol{y}|\boldsymbol{x}\right) = \prod_{i=1}^{T_y} p\left(y_i|\boldsymbol{y}_{<i}, \boldsymbol{x}\right) \tag{1}$$

where $\boldsymbol{y}_{<i} = \{y_1, \cdots, y_{i-1}\}$ denotes a set of words preceding a word at position i in the target sentence. Each conditional $p(y_i|\boldsymbol{y}_{<i}, \boldsymbol{x})$ can be defined in a parameterized form by

$$p(y_i = k|\boldsymbol{y}_{<i}, \boldsymbol{x}) = \frac{\exp(s_{ik})}{\sum_{v=1}^{V} \exp(s_{iv})} \tag{2}$$

where s_{ik} is a score for predicting the k-th word in the target vocabulary as a word at position i. Since in most cases T_x and T_y are variable across sentence pairs as well as $T_x \neq T_y$ in a single pair of sentences, we use the *encoder-decoder* architecture, also known as *sequence-to-sequence* learning (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015), where two RNNs are trained jointly to handle such variable inputs and outputs. Using the encoder-decoder RNNs the score s_{ik} can be defined by $s_{ik} = f\left(y_{i-1}, \mathbf{h}_i, \mathbf{c}_i\right)$. One RNN, the so-called *encoder*, computes the hidden state $\overline{\mathbf{h}}_j$ of a source word x_j given previous hidden state as $\overline{\mathbf{h}}_j = \text{RNN}_{enc}(\mathbf{x}_j, \overline{\mathbf{h}}_{j-1})$ where $\mathbf{x}_j$ is vector representation of a word x_j. Similarly, we obtain a hidden state $\mathbf{h}_i$ to be used for predicting y_i using the other, so-called *decoder* RNN as $\mathbf{h}_i = \text{RNN}_{dec}\left(\mathbf{y}_{i-1}, \mathbf{h}_{i-1}, \mathbf{c}_i\right)$ where $\mathbf{c}_i = \sum_{j=1}^{T_x} \alpha_{ij} \overline{\mathbf{h}}_j$ s.t. $\sum_j \alpha_{ij} = 1$ is a context vector, which is a weighted sum of input hidden states. Moreover, we also learn the *attention weights* $\alpha_{ij} = f_{att}\left(\mathbf{h}_{i-1}, \overline{\mathbf{h}}_j\right)$. The decoder's initial hidden state $\mathbf{h}_0$ is initialized by the encoder, e.g., using the last

POS/NE	German		English	
	Coverage	Ratio	Coverage	Ratio
TRUNC	36.84%	0.12%		
I-PER	37.03%	2.21%	41.69%	2.43%
I-ORG	51.67%	1.33%	76.70%	2.14%
I-LOC	63.17%	1.30%	76.18%	1.56%
ITJ/UH	66.67%	0.01%	70.00%	0.01%
NN	68.73%	20.50%	93.25%	14.75%
VVIZU	70.37%	0.17%		
FM/FW	70.76%	0.27%	61.90%	0.03%
ADJ	78.99%	7.43%	90.49%	6.89%
VVFIN	86.08%	4.53%		
VVPP	87.13%	2.29%		
CARD/CD	87.62%	1.65%	92.01%	1.85%
VVINF	90.11%	1.80%		

Table 1: Statistics of different tag sets.

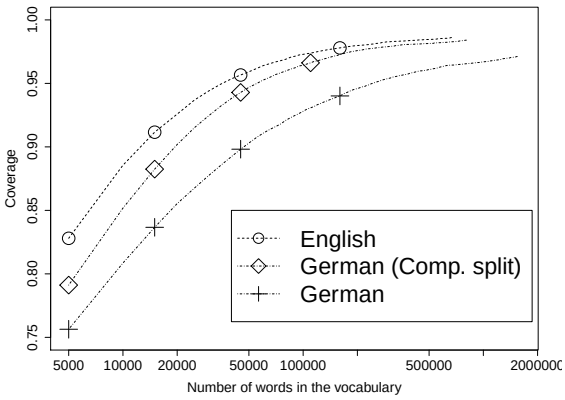


Figure 1: Coverage of the test set w.r.t. the size of vocabulary.

hidden state $\overline{\mathbf{h}}_{T_x}$ of the source sentence $\boldsymbol{x}$ while we can set the initial hidden state of the encoder $\overline{\mathbf{h}}_0 = \mathbf{0}$. For further details, see (Bahdanau et al., 2015).

Given a corpus of N training examples $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)})\}_{n=1}^{N}$, we can iteratively update the model parameters θ by using the gradient of the negative log-likelihood given by

$$\frac{\partial}{\partial\theta}\mathcal{L}(\theta;(\boldsymbol{x}^{(n)},\boldsymbol{y}^{(n)})) = -\sum_{i=1}^{T_y^{(n)}}\frac{\partial}{\partial\theta}s_{ik} - \mathbb{E}_{v\sim\mathcal{V}}\left[\frac{\partial}{\partial\theta}\exp(s_{iv})\right]. \quad (3)$$

As computational complexity of Equation (3) grows linearly in the size of the vocabulary $\mathcal{V}$, it is often the case that tens of thousands of the most frequent words are included in the vocabulary.

3 What is Behind the Out-of-Vocabulary Problem?

In the experimental results of (Jean et al., 2015), we can see that a larger vocabulary does not bring us the performance improvement when the target language is German while English to French translation benefits from the use of a larger vocabulary. This tells us that a larger vocabulary might not be an effective solution to the OOV problem, at least, when we translate into German. To look into the reasons, let us consider what type of words are treated as OOV words and how often such words appear in the corpus. To be more precise, we collect the most frequent $30,000$ words from a large corpus, i.e., the training set being used in our experiments. Then we calculate the following two scores for each of part of speech (POS) and named entity (NE) tags with respect to OOV words as follows

$$\text{coverage(tag)} = \frac{\text{\# of words with } \textit{tag} \text{ in vocabulary}}{\text{\# of words with } \textit{tag}}, \qquad \text{ratio(tag)} = \frac{\text{\# of words with } \textit{tag}}{\text{\# of words}}.$$

For POS and NE tagging, we use the Stanford NLP suite[1].

The results are summarized in Table 1, where we merge the different tagsets for German and English into common tags to make them comparable. In particular, all different types of adjectives fall into a single *ADJ* tag. The ratio refers to the frequency in the test set, e.g., 20.50% for *NN* means that 20.50% of the words in unseen data are nouns. As can be seen in the table, NEs in all forms, i.e., I-PER, I-ORG and I-LOC, are poorly covered. This is intuitively obvious because a fixed-size vocabulary cannot contain the names of all people, places, or organizations. NEs also appear frequently, i.e., 5% of all unknown words are NEs. The coverage of numbers (*CARD*) is also suboptimal, with the English vocabulary covering 92.01% while the German vocabulary covers only 87.62%. For numbers as well as for NEs this may not be a problem per se because OOV words can be copied from the source sentence using the alignment model.

[1] http://stanfordnlp.github.io/CoreNLP

The biggest gap of coverage of a tag set between English and German can be seen for nouns (*NN*). English features a coverage of 93.25% while German covers only 68.73%. The reason for this is most likely due to an interesting property of the German language: compound words. In German, new words can be created by concatenating two or more words.

Interestingly, it is hard to get 95% of German words covered by a vocabulary of even 1M words in contrast to English. Figure 1 shows coverage of words in the test set as a function of vocabulary size. Unless German compound words are split, coverage of German is lower than that of English by $3 \sim 8\%$ depending on the vocabulary size. Splitting compound words enables to cover German words in the test set as much as English ones as the vocabulary size increases.

4 Dealing with Compound Words

Compounds are words (more precisely, lexemes) consisting of more than one stem. Compounding is found in many languages, including Dutch, Finnish, and German. To build a compound word, two or more words are joined resulting in a longer word. For example, the German compound word *Apfelkuchen* is produced by concatenating *Apfel* (apple) and *Kuchen* (pie). Yet in German, it needs to be explicitly added to the vocabulary while its corresponding term, *apple pie*, is likely to be already included in the vocabulary extracted from a large English corpus. As Germanic languages such as German have a high number of compound words, we explore the methodology of compound splitting for German being both on the source and target side, referred to as *compound splitting* and *compound merging*, respectively.

4.1 Compound Splitting

Compound splitters often employed in the literature generally fall into one of two categories (Fritzinger and Fraser, 2010; Popović et al., 2006): *corpus-driven* and *linguistic*. The former is based on word frequencies calculated on a training corpus, the best split being selected among the candidates. The latter splits words through linguistic analysis e.g., by using a morphological analyzer. Hybrid approaches are possible, and corpus-driven splitters found in practice often contain some additional rules instead of relying solely on corpus statistics. Likewise, while it is possible to split words based solely on the output of a morphological analyzer, a splitter may take word frequencies into account.

In this work, we consider the following two splitters:

1. The compound splitter developed by Weller and Heid (2012), herein referred to as *IMS splitter*, is based on the corpus-driven approach by Koehn and Knight (2003). We compute a frequency list of lemmatized words using TreeTagger (Schmid, 1994; Schmid, 1995) on the WMT14 training data as well as a frequency list of words with respect to their POS tag.[2] These two lists are used to improve the performance of the IMS splitter. Different splitting possibilities are ranked by the geometric mean of the frequencies $(\prod_{i=1}^{n} \text{freq}(p_i))^{\frac{1}{n}}$ where p_i are the parts of the respective splittings and n the number of parts. An exemplary output of this splitter can be found in Table 2, which shows different ways to split a word along with the scores for this particular split.

2. Fritzinger and Fraser (2010) studied compound splitters in terms of their effectiveness in statistical machine translation (SMT) systems and propose a hybrid approach that uses corpus statistics as well as a morphological analyzer. Hence, we will refer to this splitter as *hybrid splitter*. Based on split points provided by the morphological analyzer *Smor*, word frequencies from a large training corpus are consulted. Smor is a finite-state based morphological analyzer concerned with the word formation of German, i.e., inflection, derivation, and compounding. By employing a hybrid approach, only linguistically motivated splittings (provided by Smor) are performed according to the frequency lists on the training corpus, hence reducing the number of incorrect splits. For example, The word *Durchschnittsauto* (*standard car*) is split into *Durchschnitt Auto* instead of *Durch Schnitt Auto*. The word *Durchschnitt* has a one-to-one relation with the English word *average*, but is itself a compound word.

[2] We switched to TreeTagger from the Stanford Tagger because of its substantially faster processing speed.

Table 2: Exemplary output of the IMS splitter.

Compound	Split Compound	Score
kühleinrichtung	kühl_ADJ einrichtung_NN	869.3
	kühlen_V einrichtung_NN	251.4
	kühleinrichtung_NN	6.0
gezeitenkraftwerk	gezeiten_NN kraft_NN werk_NN	984.9
	gezeiten_NN kraftwerk_NN	324.44
	gezeitenkraft_NN werk_NN	243.6
	gezeitenkraftwerk_NN	33.0

Another important aspect when talking about compound splitting is the concept of *filler letters*. When a new word is formed through the use of compounding, sometimes filler letters are inserted between two compounds. For German, these filler letters are usually *s, en, n, er, and es*. For example, the German word *Jahresverpflichtung* is made from the words *Jahr* (year) and *Verpflichtung* (commitment), and between these two words the infix *es* is inserted. The easiest way to deal with this issue is to maintain a list of filler letters and a set of rules for testing whether a compound ends with one of them, in which case these filler letters can be removed. Indeed, the IMS splitter uses exactly this type of rules to provide proper splits. The hybrid splitter does not need an explicit definition of filler letters because this knowledge is already encoded in the morphological analyzer Smor.

For the inverse task of translating from English to German, it is not so clear what filler letters should be used because filler letters are needed for reconstructing a correct compound word from its individual elements.

4.2 Compound Merging

When compounds occur on the target side it is necessary to join individual compound elements together. We can deal with this problem by inserting special syntax into the training corpus that eases the process of compound merging. For example, instead of encoding the word *Autofahrer* (car driver) as *Auto* and *Fahrer*, we can encode it as *Auto @@ Fahrer*. By doing so, we hope the model will produce such connection markers in the decoder, which can then easily be joined during postprocessing. This also solves the problem of filler letters for German being on the target side, because filler letters can be encoded using this special syntax as well. For instance, the word *Kraftverkehrsverband* is made of up the words *Kraft*, *Verkehr*, and *Verband* and can be encoded as *Kraft @@ Verkehr @s@ Verband*.

This process has also been proposed by (Williams et al., 2015) using the hybrid splitter by (Fritzinger and Fraser, 2010). We argue that this special syntax can also be produced by a much simpler corpus-driven splitter such as the IMS splitter. Hence, we modified the IMS splitter to output these filler letters between compound words. This modification applies to all compound splitters simply by comparing the splitter's output with its input and extracting filler letters using a regular expression.

5 Experimental Setup

This section describes the foundations of our experiments. In particular, we address preprocessing, which is an important part of the machine learning pipeline, and the setup of our NMT training.

5.1 Dataset & Preprocessing

Our models were trained on the data provided by the 2014 Workshop on Machine Translation (WMT). Specifically, we used the Europarl v7, Common Crawl, and News Commentary corpora. Our development set corresponds to the data in the *newstest2013* files while the test set is given by *newstest2014*.[3] We use Moses' tokenizer[4] and filter out noisy sentences, written in a different language, using a language detection tool.[5]

[3] We use cleaned test sets.

[4] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl

[5] https://github.com/shuyo/language-detection

Since some of the German sentences in the training set were made available before the German orthography reform of 1996, we converted such words to conform to the new spelling rules. For instance, the following rules were used: (muß ⟶ muss), (Phantasie ⟶ Fantasie), (Bestelliste ⟶ Bestellliste), (essentiell ⟶ essenziell) and (geschrieen ⟶ geschrien). Note that this spelling conversion does not affect the development and test sets because no occurrences of words written in the old style were present.

5.2 Training Details

Our NMT implementation is based on the neural network framework `Blocks`[6] (van Merriënboer et al., 2015). While it may be advisable to try different hyperparameters, an elaborate tuning of the RNNs was not practical given that it took about one week to train a single hyperparameter configuration for machine translation on the WMT data set on our hardware. Therefore, we chose the same set of hyperparameters as in (Bahdanau et al., 2015) except for the size of the vocabularies. To be more specific, we set the dimensionality of word vectors to 640, the number of hidden units in GRU-RNN to $1,000$, the maximum sequence length during training to 50, the mini-batch size to 80, and the number of words in both source and target vocabularies to $30,000$. We did not use weight noise regularization or dropout. We also randomly shuffled the training data using a constant seed for all our experiments so that different models are trained on the same order of the data set.

We trained our models on a NVIDIA Tesla K20 and Titan Black GPU. Since our data set contains nearly 4 million instances, a single epoch corresponds to around $48,000$ iterations. Depending on the model, the best score on the development set was usually achieved at around $400,000$ to $500,000$ iterations, or 8 to 10 epochs, or 5 to 7 days. At every checkpoint, $15,000$ iterations in our experiments, we computed the bilingual evaluation understudy (BLEU) score (Papineni et al., 2002), the Metric for Evaluation of Translation with Explicit ORdering (METEOR) (Banerjee and Lavie, 2005) and the translation error rate (TER) (Snover et al., 2006) to monitor the progress of the model on the development set.

6 Experiments

In this section, we present the results for the experiments we conducted with the setup explained in the previous sections. We will first show that German to English translation works considerably better than English to German (Section 6.1), and then try to analyze this observation in more depth (Section 6.2).

6.1 Effect of Compound Splitting and Joining

Table 3 shows the performance of three models on the test set for both translation directions. All our models produce cased outputs. Nonetheless, BLEU scores are given for evaluation performed before and after lowercasing predictions and reference sets.

For English to German, both compound splitters contribute to the improvements of NMT models across all measures except for TER although the gap of the performance between the models is negligible. However, for German to English translation, we can observe huge gains by using the corpus-based compound words splitting method, outperforming the baseline by 2.7 BLEU$^{\text{uncased}}$, 3.3 METEOR, and -0.6 TER. While the use of compound splitters enables to achieve significant performance improvement in terms of BLEU and METEOR, only marginal improvement was observed in TER. This implies that splitting German compound words results in more number of correct n-grams in English whereas it has no significant impact on word ordering. See more details of three metrics in (Birch and Osborne, 2011). Note that German word order can differ from that of English due to separated verb prefixes and long-range movement of verbs in German.

When inspecting the output of both splitters, we noticed that the hybrid/linguistic splitter is more likely to produce a correct splitting. However, the corpus-based splitter provides better results for NMT. This contradiction has also been observed by (Fritzinger and Fraser, 2010) for phrase-based SMT. Corpus-driven methods tend to split more aggressively. For instance, consider the word *überall* which may get split into *über* and *all* by the corpus-driven approach due to these two words appearing very frequently. Conversely, the linguistic splitter is unlikely to split this word.

[6] `https://github.com/mila-udem/blocks`

Table 3: Results of compound split for German words using either a corpus-based splitter or hybrid corpus-based/linguistic-motivated one. Scores in subscripts are on the development set.

Translation Direction	Model	↑ BLEU$^{\text{uncased}}$	↑ BLEU$^{\text{cased}}$	↑ METEOR$^{\text{uncased}}$	↓ TER$^{\text{uncased}}$
English → German	Baseline	$19.21_{19.6}$	$18.79_{19.11}$	$39.40_{39.00}$	$63.40_{61.70}$
	Hybrid Split	$19.34_{19.58}$	$18.95_{19.10}$	$39.90_{39.10}$	$63.70_{62.20}$
	Corpus-based Split	$19.41_{20.22}$	$18.99_{19.78}$	$40.30_{39.80}$	$64.40_{62.10}$
German → English	Baseline	$21.72_{23.30}$	$20.91_{22.36}$	$26.50_{27.90}$	$58.80_{58.70}$
	Hybrid Split	$22.30_{23.55}$	$21.37_{22.50}$	$27.30_{28.40}$	$60.00_{59.30}$
	Corpus-based Split	$24.42_{24.89}$	$23.41_{23.75}$	$29.80_{30.30}$	$58.20_{57.80}$

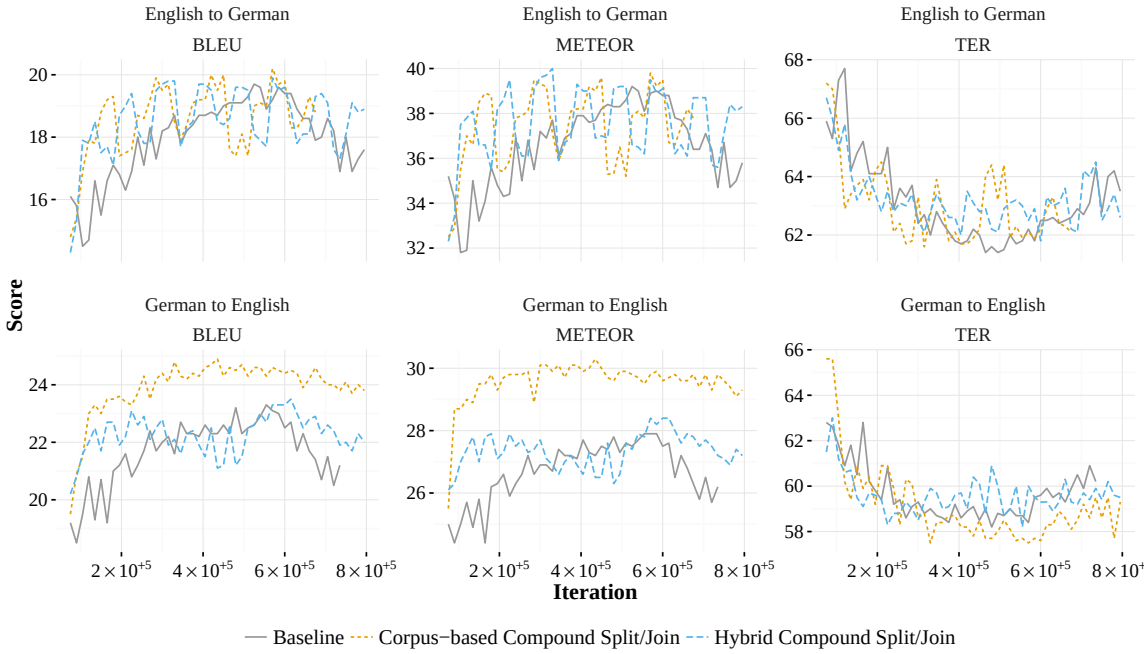


Figure 2: Performance of NMT models handling compound words differently on the development set.

The explanation for a translation system performing better with aggressive splits could be that the system can recover from words that are split too much. This is especially applicable to NMT, where the problem of OOV words leads to many compound words not found in the vocabulary. Hence, the performance hit by unsplit words that may lead to OOV words is larger than the performance hit produced by overaggressive splitting.

We believe that the reason why compound splitting works much better for English than for German is that the vocabulary coverage for English is well above the coverage of German. As noted in Section 3, we found that only 68.73% German nouns are covered when encoding a text corpus using a fixed vocabulary of size $30,000$, whereas English nouns are covered at a ratio of 93.25%. This motivated our experiment that contains a compound splitter. Before we started training a NMT model with compound split data, we first explored the effects of compound splitting in terms of word coverage on the test data. For this analysis, we used the IMS splitter. Our results, which can be seen in Figure 1, confirm that word coverage for German dramatically increases when compound splitting is applied. Another possible reason is that if a compound word at the source side is split into multiple words, the NMT model described in Section 2 is able to learn alignments between English words and multiple subwords of the compound word. We think that this is also related to why the frequency-based vocabulary construction method improves the performance of NMT models (Sennrich et al., 2016).

6.2 Difficulties When Translating into German

When German is on the target side, the model successfully produces the special syntax we used that allows us to concatenate compound words as a postprocessing step (cf. Section 4.2). A translation produced by this model looks as follows:

Source Sentence	The darker the meat, the higher the **pH value**.
Reference Sentence	Je dunkler das Fleisch, desto höher der **ph-Wert**.
Baseline Model	Je dunkler das Fleisch, desto höher der **pH**.
Split/Join Model	Je dunkler das Fleisch, desto höher der **pH @-@ Wert**.
Split/Join Model (merged)	Je dunkler das Fleisch, desto höher der **pH-Wert**.

The translation produced by the model for the phrase *pH value* is *pH @-@ Wert* and can be concatenated by the use of a regular expression. This seems to work even with longer compound words, as shown by the following example:

Source Sentence	A total of four **road safety inspections** were carried out.
Reference Sentence	Insgesamt seien vier **Verkehrsschauen** durchgefhrt worden.
Baseline Model	Insgesamt wurden vier ⟨**UNK**⟩ durchgefhrt.
Split/Join Model	Es wurden insgesamt vier **Straße @n@ Verkehr @s@ Inspektionen** durchgefhrt.
Split/Join Model (merged)	Es wurden insgesamt vier **Straßenverkehrsinspektionen** durchgefhrt.

In this case, the baseline model fails to translate the phrase altogether resulting in an unknown word. For readers not familiar with German, all sentences are correct translations for the source sentence when we disregard the unknown word, i.e., the baseline model and the split model differ in structure, but both are correct, and the split join model is more precisely in translating the word for road safety inspection.

Another interesting observation is that the performance in terms of BLEU fluctuates wildly as splitting is applied. We compared the translation output between iterations showing favorable performance and iterations with a lower BLEU score. We found that sentences were often restructured causing the performance to drop. An example of this restructuring may look as follows:

Source Sentence	However, the new rules put in place will undoubtedly make it more difficult to exercise the right to vote in 2012 .
Reference Sentence	Allerdings werden die neu eingeführten Regeln im Jahr 2012 zweifellos die Ausübung des Wahlrechts erschweren .
Split/Join Model, Iteration 450000	Mit den neuen Regeln wird es zweifellos schwieriger sein, das Wahlrecht im Jahr 2012 auszuüben .
Split/Join Model, Iteration 465000	Die neuen Regeln werden jedoch zweifellos die Ausübung des Wahlrechts für 2012 erschweren .

The system output at iteration $450,000$ in this example receives 35.29 BLEU points, whereas the output at iteration $465,000$ receives 71.43 BLEU points. This discrepancy can be explained with the fact that BLEU as an evaluation measure favors sentences with more n-gram matches. The model's output at iteration $450,000$ receives a 4-gram match for the phrase *zweifellos die Ausübung des Wahlrechts erschweren*, whereas the other model's output does not. Please do note that both outputs are perfect translations of the source sentence and would be evaluated similarly by a human evaluator.

7 Conclusion

In this work, we presented why word-level NMT models have trouble with OOV words and explained why such problems arise. Based on the analysis about the OOV problem, we obtained substantial improvements of 2.7 BLEU points for German to English translations by splitting German compound words on the source side. We also applied compound splitting when German was on the target side by modifying the compound splitter's output to contain special syntax that can later be used to create compound words from their individual components. While we obtained only a modest improvement of 0.2 BLEU, our proposed method is able to merge German words on the target side, allowing us to generate correct target sentences.

Acknowledgements

Calculations on the Lichtenberg high performance computer of the Technische Universität Darmstadt were conducted for this research. The Titan Black GPU used for this research was donated by the NVIDIA Corporation. This work has been supported by the German Institute for Educational Research (DIPF) under the Knowledge Discovery in Scientific Literature (KDSL) program, and the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representation*, San Diego, California.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for machine translation evaluation with improved correlation with human judgments. In *Proceedings of the Association for Computational Linguistics Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, volume 29, pages 65–72.

Alexandra Birch and Miles Osborne. 2011. Reordering metrics for mt. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1027–1035, Portland, Oregon, June. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, Doha, Qatar.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1693–1703, Berlin, Germany.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1370–1380, Baltimore, Maryland.

Fabienne Fritzinger and Alexander Fraser. 2010. How to avoid burning ducks: combining linguistic analysis and corpus statistics for german compound processing. In *Proceedings of the 5th Joint Workshop on Statistical Machine Translation and MetricsMATR*, pages 224–234, Uppsala, Sweden.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1–10, Beijing, China.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 187–193, Budapest, Hungary.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan Black. 2015. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1054–1063, Berlin, Germany.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, Beijing, China.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania.

Maja Popović, Daniel Stein, and Hermann Ney. 2006. Statistical machine translation of german compound words. In *Advances in Natural Language Processing*, volume 4139, pages 616–624. Springer.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, volume 12, pages 44–49, Manchester, UK.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to german. In *Proceedings of the Association for Computational Linguistics (ACL) Special Interest Group for Linguistic Data and Corpus-Based Approaches to Natural Language Processing (EMNLP)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725, Berlin, Germany.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*, pages 223–231, Cambridge, Massachusetts.

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

Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. 2015. Blocks and fuel: frameworks for deep learning. *arXiv preprint arXiv:1506.00619*.

Marion Weller and Ulrich Heid. 2012. Analyzing and aligning german compound nouns. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 2395–2400, Istanbul, Turkey.

Philip Williams, Rico Sennrich, Maria Nadejde, Mathias Huck, and Philipp Koehn. 2015. Edinburgh's syntax-based systems at WMT 2015. In *Proceedings of the 10th of the Association for Computational Linguistics Workshop on Statistical Machine Translation*, pages 199–209, Lisbon, Portugal.

Improving Word Alignment of Rare Words with Word Embeddings

Masoud Jalili Sabet[1] **Heshaam Faili**[1] **Gholamreza Haffari**[2]
[1] School of Electrical and Computer Engineering, University of Tehran, Iran
[2] Faculty of Information Technology, Monash University, Australia
{jalili.masoud,hfaili}@ut.ac.ir
gholamreza.haffari@monash.edu

Abstract

We address the problem of inducing word alignment for language pairs by developing an unsupervised model with the capability of getting applied to other generative alignment models. We approach the task by: *i)* proposing a new alignment model based on the IBM alignment model 1 that uses vector representation of words, and *ii)* examining the use of similar source words to overcome the problem of rare source words and improving the alignments. We apply our method to English-French corpora and run the experiments with different sizes of sentence pairs. Our results show competitive performance against the baseline and in some cases improve the results up to 6.9% in terms of precision.

1 Introduction

Statistical machine translation systems usually break the translation task into two or more subtasks and an important one is finding word alignments over a sentence-aligned bilingual corpus (Brown et al., 1990). One of the common approaches to find word alignment, uses a generative translation model that produces a sentence in one language given the sentence in another language. Most SMT systems use an implementation of the IBM alignment models (Brown et al., 1993). These models use expectation maximization (EM) algorithm in training and only require sentence-aligned bilingual texts. Inducing word alignment from bilingual texts requires large amount of sentence-aligned parallel data which is not available for most of the language pairs. It also causes more problems for rare words which are the key parts of the sentences, but at the same time, are less frequent and therefore have a more biased probability distribution (Moore, 2004).

This paper deals with the alignment task for the rare words by proposing a new alignment model based on the low-dimensional representations of the words. We explore the effect of replacing words with their vector space embeddings in IBM Alignment Model 1. In this model, instead of using a conditional multinomial distribution (to generate a target word $t_i \in T$ given a source word $s_i \in S$), we use a conditional Gaussian distribution and generate a d-dimensional word embedding $V_{t_i} \in \mathbb{R}^d$ given s_i. We then propose a method to **improve the alignments for rare words** by using their similar words and updating their distributions.

The advantages of this model are: *i)* It uses monolingual word embedding which has more available training data; *ii)* By using the extracted knowledge from monolingual data, it has better results in low-resource word alignment tasks; *iii)* The Gaussian model can be applied to all generative alignment models by replacing the conditional distribution, and thus getting even better results.

2 Related Works

2.1 Word Embeddings

Recent works on word embedding show improvements in capturing semantic features of the words (Mikolov et al., 2013; Pennington et al., 2014). Since the word alignment task requires a form of statistics or comparison between words from the source and the target languages, a good translation model

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3209–3215, Osaka, Japan, December 11-17 2016.

can result in better alignments. Bilingual word embedding models like (Zou et al., 2013; Søgaard et al., 2015) train vectors for words in both languages in the same vector space, Hence a translation model can be made by these embeddings and it can be really useful for the task of word alignment. Despite the advantages of bilingual embedding models, to achieve a good performance by these models and building more informative vector representations for words, a large amount of data is required, which is not available in low-resource language pairs. On the other hand, providing a good monolingual corpus for most of the languages needs less effort and building a model that mostly uses monolingual data is more reasonable.

2.2 Alignment Model

IBM alignment models such as model 1, usually have problems with aligning rare words (Moore, 2004). In a sentence pair (S, T), for a target word $t \in T$ and two source words $s, s_{rare} \in S$ when s_{rare} is a rare word and s is a normal word, the translation model will more likely generate $p(t|s_{rare}) > p(t|s)$, and therefore most of the target words will align to s_{rare}. The reason for this problem is the rare word s_{rare} has co-occurrence with only a few target words and it increases the conditional probabilities for those target words. If a source word s which is similar to s_{rare} exists and has co-occurrence with more target words, those target words could be used to improve the distribution of s_{rare}.

There were proposed methods to overcome the problem of low-resource languages based on using different word alignment methods and combining the results like in (Xiang et al., 2010). We try to provide a combination of alignments to get better performance for the rare words.

3 Vector Model

In this section, we develop a conditional model $p(t|s)$ that, given a source word s, assigns probabilities to a target word t using a conditional Gaussian distribution. We explain the required modifications needed in IBM alignment model 1 for using word embedding and briefly review the EM algorithm in this model. We then present the method to use the similar words for updating the distributions of each word and improving the alignment results. The embedding of word $w \in W$ will be written as $\mathbf{v}_w \in \mathbb{R}^d$. The proposed model will be called the *Vector Model*.

3.1 Alignment Model

The IBM alignment models use a translation model in the form of conditional probability $p(t|s)$. In a similar way to (Lin et al., 2015), given a source word $s \in S$, instead of the probability of the target word $t \in T$, the probability of a vector representation $\mathbf{v}_t \in \mathbb{R}^d$ of the word can be calculated. Correspondingly, each source word $s \in S$ is represented by mean μ_s and covariance matrix Σ_s:

$$\begin{aligned} p(t|s) &= p(\mathbf{v}_t|\mu_s, \Sigma_s) \\ &= \frac{\exp\left(\frac{-1}{2}(\mathbf{v}_t - \mu_s)^T \Sigma_s^{-1}(\mathbf{v}_t - \mu_s)\right)}{\sqrt{(2\pi)^d |\Sigma_s|}} \end{aligned} \tag{1}$$

In this model, vector representations are only used for target words and source words are replaced with distributions in the target language vector space.

3.2 EM Algorithm

The EM algorithm should have some modifications to update the Gaussian distributions. The expectation step is similar to the original model. For all sentence pairs (S, T):

$$\forall s \in S, t \in T :$$

$$count(t|s) += \frac{p(t|s)}{\sum_{s' \in S} p(t|s')} \tag{2}$$

$$total(s) += \frac{p(t|s)}{\sum_{s' \in S} p(t|s')} \tag{3}$$

The new conditional probabilities are calculated in the maximization step and for making means and covariances for the source words, these probabilities are used as weights for weighted averaging of the vectors:

$$p(t|s) = \frac{count(t|s)}{total(s)} \tag{4}$$

$$\forall s : \mu_s = \frac{1}{\sum_t p(t|s)} \sum_t p(t|s)\mathbf{v}_t \tag{5}$$

$$\forall s : \Sigma_s = \frac{1}{\sum_t p(t|s)} \sum_t p(t|s)(\mathbf{v}_t - \mu_s)(\mathbf{v}_t - \mu_s)^T \tag{6}$$

In the initialization of the EM algorithm, for all pairs of words, the probability $p(t|s)$ is equal to 1. Therefore, the distributions of source words are made based on their co-occurrences with the target words, which is a similar situation to initialization of the IBM model 1.

3.3 Using Similar Words for Improvement

The proposed model in §3.1 only uses word embedding in the target vector space and each source word corresponds to a distribution. As discussed in §2.2, the distributions of the rare source words will be biased to a small set of target words.

For each source word s, similar source words to s can be found using the cosine similarity between their corresponding vectors. We call the top *k* similar words to each word s, the neighbors of s and represent it as $(N_k(s))$. A weight will be given to each neighbor based on the cosine similarity with the source word:

$$\forall x \in N_k(s) : w_s(x) = \frac{sim(s,x)}{\sum_{x' \in N_k(s)} sim(s,x')} \tag{7}$$

where $w_s(x)$ is the similarity weight of the neighbor word x to the word s and $sim(x,y)$ is the cosine similarity of $\mathbf{v}_x$ and $\mathbf{v}_y$.

In order to use the neighbors for updating the distributions of source words, in the maximization step of the EM algorithm, calculation of the means and covariances will have additional steps:

$$\forall s : \mu_s = \lambda\mu_s + (1-\lambda) \sum_{s' \in N_k(s)} w_s(s')\mu_{s'} \tag{8}$$

$$\forall s : \Sigma_s = \lambda\Sigma_s + (1-\lambda) \sum_{s' \in N_k(s)} w_s(s')\Sigma_{s'} \tag{9}$$

where λ is the linear interpolation parameter and it should be estimated.

4 Experiments

4.1 Data

The language pair used for all the experiments is English-French. The test set is a word-aligned bilingual corpus that contains 447 sentence pairs (Och and Ney, 2000b). Using larger data sets can result in better learning for the model, and in order to create corpora with different sizes, we appended more parallel sentences from the training set to the test data (Note that the proposed model is unsupervised and it does not use the gold alignment in the training). We created corpora with different sizes of sentence pairs and the experiments use these corpora.

For creating the word embeddings, we used the tool word2vec[1] (Mikolov et al., 2013). For the input, we used the English sentences and the French sentences separately and created two sets of vectors. The number of dimensions for vectors was set to 200.

[1] https://code.google.com/p/word2vec/

The models are evaluated using the Alignment Error Rate (AER) and using both possible and sure alignment links. The baseline model is IBM model 1 which has the same learning method and is the most similar model to the proposed one. For making the IBM alignments, the tool Giza++ was used (Och and Ney, 2000a). We believe that the same kind of improvement to this baseline can be achieved by other generative models like IBM Models, by applying the proposed model in this paper.

4.2 Number of Iterations

To see the improvement of the model in each iteration, the 1K corpus is used. Figure 1 illustrates the performance of our vector model compared to IBM model for different number of iterations.

As can be seen from the Figure 1, the proposed vector model has better performance during all iterations of the training and it seems that 5 iterations for training is suitable for both vector model and the IBM model. Therefore, for the rest of the experiments, the models will train for 5 iterations.

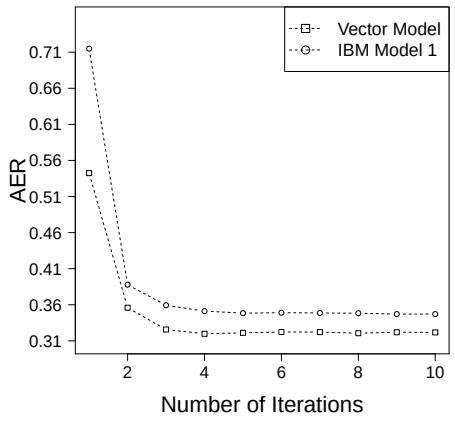


Figure 1: AER as a function of the number of iterations.

4.3 Estimation of the Parameter Lambda

To find the best value for the parameter λ, we used the 1k corpus. The AER is evaluated for different values of λ. Figure 2 shows that the best performance can be achieved by the value 0.4.

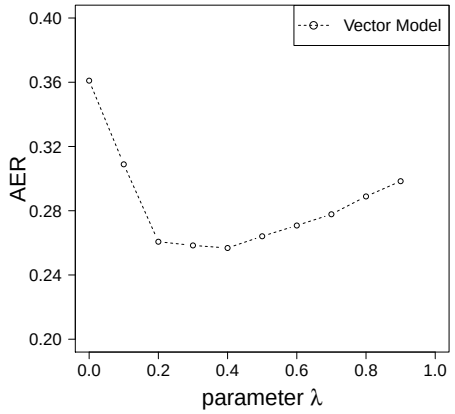


Figure 2: AER of Vector model based on the value of λ.

4.4 Number of Neighbors

To see the improvement with using different number of neighbors for updating the source words, we used the 1k corpus and changed the number of neighbors used for updating the distributions. In this experiment, for the parameter λ we used the value 0.4.

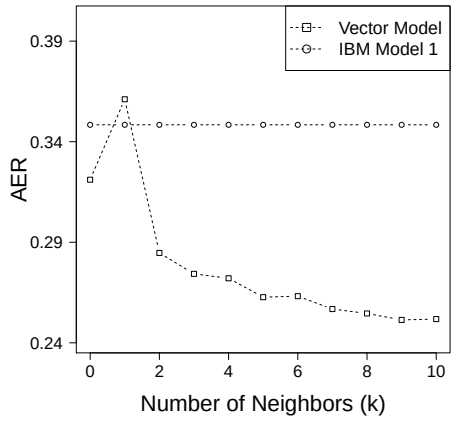


Figure 3: AER based on the number of neighbors.

Figure 3 illustrates the improvement of AER by using more neighbors for updating the distributions. The improvement is up to 9.7% in AER. The results are not improving significantly with more than 7 neighbors, and for the next experiments, the parameter k will be set to 7.

4.5 Size of the Corpus

This experiment is for evaluating the performance of the model based on the size of the corpus. In this experiment, the three models: *a)* IBM model 1, *b)* Vector model using 7 neighbors and *c)* Vector model with no neighbors are trained on the corpora with five different sizes 1k, 5k, 10k, 20k and 100K.

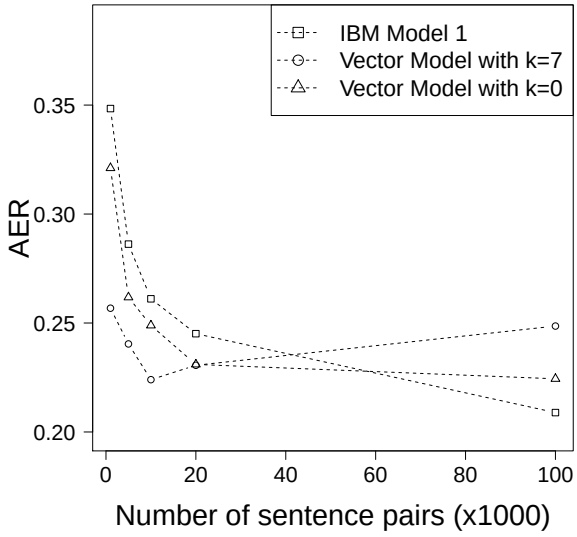


Figure 4: AER of Vector model and baseline based on the size of corpus.

Figure 4 illustrates that the two proposed models are outperforming the baseline for small data sizes. It also shows that the performance of the vector model with 7 neighbors is promising in small corpora

which is useful for low-resource languages. When the size of the corpus grows, the vector model gets to a certain threshold and will stop improving. Using the vector representations for making the translation model can be a good method for small corpora, but in a large corpus, making the probability model is far better than the approximation of the same model.

4.6 Precision on Rare Words

The last experiment is for evaluating the performance of the model based on the precision of the alignments of the rare words. As a definition for rare words, in this experiment, we call a word to be rare if it appears less that 20 times in the corpus. The precision of the alignment for the rare words is calculated by this formula:

$$precision = \frac{\#\text{of correct alignments for rare words}}{\#\text{of alignments for rare words produced by the model}} \tag{10}$$

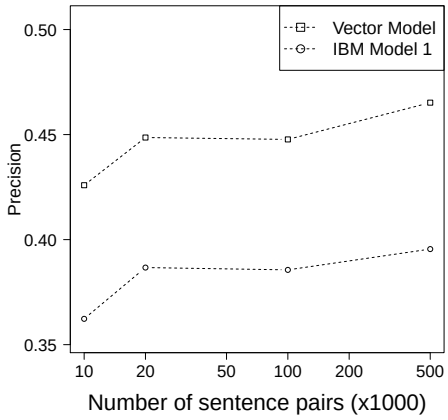


Figure 5: Precision of the Vector model and baseline on rare words based on the size of the corpus.

As shown in Figure 5, the proposed method produces better alignments for the rare words on all different sizes of the corpora.

5 Conclusion

We presented an unsupervised method to find word alignments for language pairs that uses monolingual word embeddings. We then, studied the usage of neighbors for improving the word alignment which made the model more useful for small corpora. We showed that the proposed method finds better alignments for the rare words and outperforms the baseline on different sizes of the corpora. Using the neighbors improved the performance of the model for rare words up to 6.9% compared to the baseline. The proposed Vector model has the capability of being applied to other generative alignment models which is not studied yet. Our work could be extended to other IBM models in the future works.

References

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. 2015. Unsupervised pos induction with word embeddings. *arXiv preprint arXiv:1503.06760*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Robert C Moore. 2004. Improving ibm word-alignment model 1. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 518. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2000a. Giza++: Training of statistical translation models.

Franz Josef Och and Hermann Ney. 2000b. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 440–447. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual nlp. In *The 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2015)*.

Bing Xiang, Yonggang Deng, and Bowen Zhou. 2010. Diversify and combine: Improving word alignment for machine translation on low-resource languages. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 22–26. Association for Computational Linguistics.

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398.

Measuring the Information Content of Financial News

Ching-Yun Chang[1], Yue Zhang[1], Zhiyang Teng[1], Zahn Bozanic[2], Bin Ke[3]
(1) Singapore University of Technology and Design
(2) Fisher College of Business, The Ohio State University
(3) NUS Business School, National University of Singapore
chang.frannie@gmail.com, yue_zhang@sutd.edu.sg,
zhiyang_teng@mymail.sutd.edu.sg, bozanic.1@fisher.osu.edu,
bizk@nus.edu.sg

Abstract

Measuring the information content of news text is useful for decision makers in their investments since news information can influence the intrinsic values of companies. We propose a model to automatically measure the information content given news text, trained using news and corresponding cumulative abnormal returns of listed companies. Existing methods in finance literature exploit sentiment signal features, which are limited by not considering factors such as events. We address this issue by leveraging deep neural models to extract rich semantic features from news text. In particular, a novel tree-structured LSTM is used to find target-specific representations of news text given syntax structures. Empirical results show that the neural models can outperform sentiment-based models, demonstrating the effectiveness of recent NLP technology advances for computational finance.

1 Introduction

Information has economic value because it allows individuals to make choices that yield higher expected payoffs than they would obtain from choices made in the absence of information. A major source of information is text from the Internet, which embodies news events, analyst reports and public sentiments, and can serve as a basis for investment decisions. Measuring the information content of text is hence a highly important task in computational finance. For investors such as venture capitals, information content should reflect a firm's intrinsic value, or potential of future growth. However, measuring the information content of text can be challenging due to uncertainties and subjectivity. Fortunately, for public companies, the stock price can be used as a metric. Our goal is thus to leverage such data on public companies to train a model for measuring the information content of news on arbitrary companies.

Finance theory suggests that stock prices reflect all available information and expectations about the future prospects of firms (Fama, 1970). Based on this, empirical studies in economics and finance literature have exploited statistical methods to investigate how stock returns react to a particular news or event, which is called *event studies* or *information content effect* (Ball and Brown, 1968). A standard analysis is to measure the *cumulative abnormal return* (CAR) of a firm's price over a period of time centered around the event date (termed the *event window*) (MacKinlay, 1997). Conceptually, a daily *abnormal return* represents the performance of a stock that varies from the expectation, normally triggered by an event, and can be positive or negative depending on whether the stock outperforms or underperforms the expected return. A CAR is the sum of all abnormal returns in an event window, formally described in Section 2.

We study how news affects a public firm's CAR for training a model to measure the information content of arbitrary financial news. It is worth noting that this is different from predicting a firm's stock price movements, which aims at maximizing trading profits. Rather, we investigate whether NLP techniques can assist the understanding of an event's economic value. In finance literature sentiment signals have been used as a standard linguistic feature for representing information content (Tetlock, 2007). We build a baseline using frequency-based features derived from sentiment words to represent news text. However, sentiment polarities are subjective and may not fully represent the message conveyed in news text. For example, event information is also influential in determining a firm's stock price.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,
pages 3216–3225, Osaka, Japan, December 11-17 2016.

To address this, we propose a deep neural model to better present news. A typical way of modeling a sentence is to treat it as a sequence and input the sequence to a long short-term memory (LSTM; (Hochreiter and Schmidhuber, 1997) model, which is capable of learning semantic features automatically. However, information may present different impact to individual firms and therefore, we need a way to represent information conveyed in a news sentence depending on a specific firm. We propose a novel Tree-LSTM which incorporates contextual information with target-dependent grammatical relations to embed a sentence. Because of the target-dependent feature of the proposed embedding model, the representation of a sentence varies from firm to firm.

We train our information content measurement model with news text collected from Reuters Business & Financial News[1]. Results show that the proposed model yields significant improvements over a baseline sentiment-based model. Different from existing event studies which focus on predefined events or firms (Davis et al., 2012), our model is general to various news and firms; and one can measure the effect of information content of news on any companies, including private companies.

The contribution of this paper is two-fold:

- First, we show that the information content of news text can be measured automatically. Our results demonstrate the effectiveness of state-of-the-art NLP techniques in computational finance, and introduce *information content effect* in finance to the ACL community. Given the ubiquitous and instantaneous nature of electronic text, text analytics is an obvious approach for information content analysis.
- Second, we design a novel target-dependent Tree-LSTM-based model for representing news sentences. To our knowledge, this is the first open-domain information content effect prediction system using machine learning and NLP technologies.

2 Cumulative Abnormal Return

Formally, the *abnormal return* of a firm j on date t is the difference between the *actual return* R_{jt} and the *expected return* $\hat{R_{jt}}$. $\hat{R_{jt}}$ can be an estimated return based on an asset pricing model, using a long run historical average, or it can be the return on an index, such as the Dow Jones or the S&P 500 during the same period. For example, suppose that a firm's stock price rose by 3%, and the market index increased by 5% over the same period. If the stock is expected to perform equally to what the market does, namely 5%, the stock yielded an abnormal return of -2% even though the firm's actual return is positive.

The *cumulative abnormal return* (CAR) of a firm j in an n-day event window is defined as the sum of abnormal returns of each day:

$$CAR_{jn} = \sum_{t=1}^{n} (R_{jt} - \hat{R_{jt}}) \quad (1)$$

The event window is normally centered at the event date and only trading days are considered in a window. The CARs before and after the event date mimic possible information leaks and delayed response to the information, respectively. Depending on the span of an event window, CARs provide analysts with short- and long-term information about the impact of an event on a stock's price. The most common event window found in research is a three-day window (-1, 1) where an event is centered at day 0 (Tetlock, 2007; Davis et al., 2012), and the corresponding CAR is denoted as CAR_3.

In this paper we model the effect of a news release on a firm's CAR_3. If a news release occurs during trading hours, day 0 is the news release date; otherwise, day 0 is the next trading period. As prior research demonstrates that there is no significant difference between a modeled expected return and the market return for a short-term event window (Kothari and Warner, 2004), we compute the expected return $\hat{R_{jt}}$ by the return of the equally-weighted market index including all the stocks on NYSE, Amex and NASDAQ.

3 Information Content Prediction

Our goal is to estimate the polarity of information content $y \in \{0$: negative effect, 1: positive effect$\}$ given a sentence s in which a target firm is mentioned. Assume that there is an embedding model that

[1] http://www.reuters.com/finance

maps a sentence to a feature vector $g \in \mathbb{R}^{d_g}$. The probability of positive information content effect is defined as:

$$p(y = 1|s) = softmax(W_p g + b_p) \tag{2}$$

W_p and b_p are parameters. We build two methods to obtain g from text, one being a traditional method in the finance literature (Section 4) and the other being a novel neural network (Section 5).

It is possible that there is more than one sentence, $\langle s_1, s_2, \ldots, s_m \rangle$, mentioning a firm of interest in the same event window. In this case, a neural attention mechanism adapted from Bahdanau et al. (2014) is utilized to synthesize the corresponding information embeddings $\langle g_1, g_2, \ldots, g_m \rangle$. To compute the attention vector, we define:

$$u^{(i)} = v^T tanh(W_u g_i) \qquad a^{(i)} = softmax(u^{(i)}) \qquad g' = \sum_{i=1}^{m} a^{(i)} g_i \tag{3}$$

where v and W_u are learnable parameters. $u^{(i)}$ is the score of how much attention should be put on g_i, and $a^{(i)}$ is the normalized score. The final synthetic feature vector g' substitutes for g in Equation 2 to predict the effect.

4 Sentiment-based Representation

A growing body of finance research literature examines the correlation between financial variables, such as stock returns and earning surprises, and the sentiment of corporate reports, press releases, and investor message boards (Li, 2010; Davis et al., 2012), most of which are based on purpose-built sentiment lexicon. A commonly used source is the list compiled by Loughran and McDonald (2011),[2] which consists of 353 positive words and 2,337 negative words. As a baseline we follow prior literature (Mayew and Venkatachalam, 2012) and represent the information content of a sentence based on counts derived from the lexicon of Loughran and McDonald. The feature vector consists of raw frequency counts and sentence length-normalized values of positive words, negative words, and the difference of positives and negatives. In addition, the sentence length is also considered.

5 Deep Neural Representation

As mentioned in the introduction, it is useful to model news information content beyond their sentiment signals. Deep learning has shown being effective in automatically inducing features that capture semantic information over nature language sentences. To verify the relative effectiveness of deep neural models, we first build a baseline neural network model using a popular LSTM structure (Section 5.1) and then develop a novel syntactic tree-structured LSTM that is sensitive to specific target entities (Section 5.2).

5.1 Bidirectional LSTM

A popular way of modeling a sentence s is to represent each word by a vector $x \in \mathbb{R}^{d_x}$ (Mikolov et al., 2013), and sequentially input its word vectors $\langle x_1, x_2, \ldots, x_{|s|} \rangle$ to a long short-term memory (LSTM; Hochreiter and Schmidhuber (1997)) model, which is a form of recurrent neural network (RNN; Pearlmutter (1989)). We take a variation of LSTM with *peephole connections* (Gers and Schmidhuber, 2000), which uses a *input gate* i_t, a *forget gate* f_t and a *output gate* o_t in the same memory block to learn from the current cell state. In addition, to simplify model complexity, we adopt coupled i_t and f_t (Cho et al., 2014). The following equations show how the LSTM *cell state* c_t and the *output* of the memory block h_t are updated given *input* x_t at time step t:

$$\begin{aligned} i_t &= \sigma(W_1 x_t + W_2 h_{t-1} + W_3 c_{t-1} + b_1) & c_t &= f_t \otimes c_{t-1} + i_t \otimes tanh(W_7 x_t + W_8 h_{t-1} + b_3) \\ f_t &= 1 - i_t & h_t &= o_t \otimes tanh(c_t) \\ o_t &= \sigma(W_4 x_t + W_5 h_{t-1} + W_6 c_t + b_2) \end{aligned} \tag{4}$$

The W terms are the weight matrices (W_3 and W_6 are diagonal weight matrices for peephole connections); the b terms denote bias vectors; σ is the logistic sigmoid function; and $\otimes$ computes element-wise multiplication of two vectors.

[2] http://www.nd.edu/~mcdonald/Word_Lists.html

A deep LSTM is built by stacking multiple LSTM layers, with the output memory block sequence of one layer forming the input sequence for the next. At each time step the input goes through multiple non-linear layers, which progressively build up higher level representations from the current level. In our information embedding models, we embody a deep LSTM architecture with 2 layers.

One of our baseline information embedding models is a bidirectional LSTM model (Graves et al., 2013), called *BI-LSTM*, consisting of two 2-layer LSTMs running on the input sequence in both forward and backward directions yielding vectors $\langle \overrightarrow{h_1}, \overrightarrow{h_2}, \ldots, \overrightarrow{h_{|s|}} \rangle$ and $\langle \overleftarrow{h_{|s|}}, \overleftarrow{h_{|s|-1}}, \ldots, \overleftarrow{h_1} \rangle$, respectively. We exclude stopwords and punctuations from each sentence. The final model outputs the information embedding g by concatenating the final outputs of the two LSTMs, namely $\overrightarrow{h_{|s|}}$ and $\overleftarrow{h_{|s|}}$.

5.2 Dependency Tree-LSTM

A syntactic approach for modeling a sentence s is to use a tree-structured LSTM (Tree-LSTM), embedding the parse tree of a sentence (Le and Zuidema, 2015; Tai et al., 2015; Zhu et al., 2015; Miwa and Bansal, 2016). Our hypothesis is that dependency relations between words convey a certain level of information content. For example, a dependency parser can tell *Facebook* is the subject which did the action *acquired* to the object *Whatsapp* in the sentence *Facebook acquired Whatsapp*. We parse sentences with ZPar (Zhang and Clark, 2011)[3] and adopt the N-ary Tree-LSTM of Tai et al. (2015) with peephole connections to run on a binarized dependency-based parse tree.

For the specific task, we leverage the structure of a binary Tree-LSTM, and develop a novel way to represent a dependency relation between two words using this structure (Section 5.2.1). We propose an algorithm to transform a dependency parse tree to a binary tree, where leaf nodes are words and internal nodes are dependency relations, so that the transformed tree can be embedded using binary Tree-LSTM (Section 5.2.2). Finally we explain how the task of information content effect measurement can benefit from the target-dependent feature of the proposed binarization algorithm (Section 5.2.3).

5.2.1 Binary Tree-LSTM for Dependency Arcs

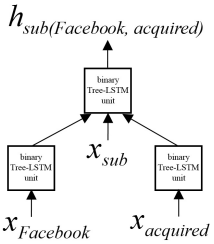
Similar to the LSTM memory block described in Section 5.1, a binary Tree-LSTM unit (Tai et al., 2015) takes *input* x_t at time step t and updates its *cell state* c_t and the *output* of the memory block h_t controlled by *input gate* i_t, *forget gate* f_t and *output gate* o_t. However, instead of depending on only one previous memory block as in a sequential LSTM model, a binary Tree-LSTM unit takes two children units, namely left (l) and right (r), into consideration. In this case, there are two forget gates f_t^l and f_t^r for the left and right children, respectively, so that information from each child can be selectively incorporated. The unit activations are defined by the following set of equations:

$$
\begin{aligned}
i_t &= \sigma(W_9 x_t + \sum_{D\in\{l,r\}} (W_{10}^D h_{t-1}^D + W_{11}^D c_{t-1}^D) + b_4) \\
f_t^l &= \sigma(W_{12} x_t + \sum_{D\in\{l,r\}} (W_{13}^D h_{t-1}^D + W_{14}^D c_{t-1}^D) + b_5) \\
f_t^r &= \sigma(W_{15} x_t + \sum_{D\in\{l,r\}} (W_{16}^D h_{t-1}^D + W_{17}^D c_{t-1}^D) + b_6) \\
o_t &= \sigma(W_{18} x_t + \sum_{D\in\{l,r\}} W_{19}^D h_{t-1}^D + W_{20} c_t + b_7) \\
c_t &= \sum_{D\in\{l,r\}} f_t^D \otimes c_{t-1}^D \\
&\quad + i_t \otimes tanh(W_{21} x_t + \sum_{D\in\{l,r\}} W_{22}^D h_{t-1}^D + b_8) \\
h_t &= o_t \otimes tanh(c_t)
\end{aligned}
\tag{5}
$$

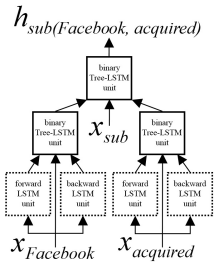
The above binary Tree-LSTM model was proposed to represent binary-branching constituents (Tai et al., 2015). In this paper, however, we show that it can be used to represent dependency arcs. For example, given the subject dependency arc *sub* between *acquired* (head) and *Facebook* (dependent), Figure 1a illustrates the bottom-up information propagation in a binary Tree-LSTM model, where x_{sub}, $x_{acquired}$ and $x_{Facebook}$ are vector representations of *sub*, *acquired* and *Facebook*, respectively. The output of the last unit $h_{sub(Facebook,acquired)}$ is the dependency embedding of *sub(acquired, Facebook)*. We call this model *LEX-TLSTM*.

Miwa and Bansal (2016) incorporate bidirectional LSTMs into the input of a Tree-LSTM unit by concatenating x_i, $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ so that contextual information of individual words can be considered. Instead

[3]We use the default English dependency parser model available at https://github.com/frcchang/zpar/releases/tag/v0.7.5



(a) Lexical input.

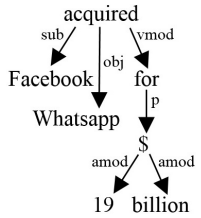


(b) Contextual input.

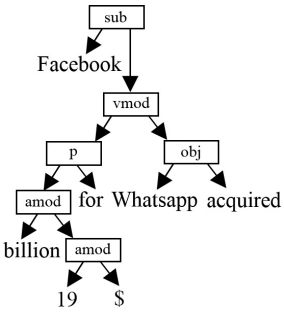
Figure 1: Binary Tree-LSTM models.

Input: target node n, dependency parse tree T_d
Output: binarized dependency tree T_b
$T_b \leftarrow NewBinaryTree$
$T_b.root \leftarrow Binarize(n, T_d)$
Function *Binarize(n, T_d)*
  if T_d.*HasNoDep(n)* **then**
    return n
  end
  if T_d.*HasParent(n)* **then**
    $p \leftarrow T_d.GetParent(n)$
    $btn \leftarrow NewBinaryTreeNode$
    $btn.dep \leftarrow T_d.RemoveDep(p, n)$
    $btn.LChild \leftarrow Binarize(n, T_d)$
    $btn.RChild \leftarrow Binarize(p, T_d)$
    return btn
  end
  if T_d.*HasChild(n)* **then**
    $c \leftarrow T_d.GetOneChild(n)$
    $btn \leftarrow NewBinaryTreeNode$
    $btn.dep \leftarrow T_d.RemoveDep(n, c)$
    $btn.LChild \leftarrow Binarize(c, T_d)$
    $btn.RChild \leftarrow Binarize(n, T_d)$
    return btn
  end

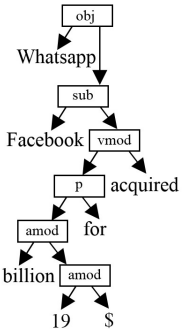
Algorithm 1: Dependency tree Binarization



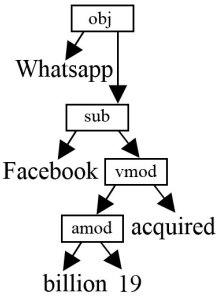
(a) Dependency parse tree.



(b) Binarized dependency tree using the root word *acquired* as the target.



(c) Target dependent tree using *Whatsapp* as the target.



(d) Remove stopwords and punctuations from tree (c).

Figure 2: A dependency parse tree and its binarized versions given different targets.

of inputting the three vectors as a whole into a Tree-LSTM unit, we treat forward and backward 2-layer LSTM units as the left and right children, respectively, while inputting a word vector as shown in Figure 1b, which we refer as *CTX-TLSTM*.

5.2.2 Binarized Dependency Tree

Figure 2a shows the dependency parse tree of the tokenized sentence *Facebook acquired Whatsapp for $ 19 billion*, where the root word is at *acquired* and head words are at the upper ends of dependency branches. To adapt a dependency parse tree to a binary Tree-LSTM model, Algorithm 1 presents a recursive algorithm to binarize a dependency parse tree given a target word.

The algorithm starts from a given target word and creates a binary tree node to represent a dependency relation between the target word and another word, with the dependent and head placed at the left and right children, respectively. The rule for selecting the dependency relation is that the dependency with the target's head is considered first followed by that of the target's dependents. In addition, we sort the dependents of a target word in a way that left context words are always in front of right context words, and both of the left and rigth context words are ordered by the distance to the target word in descending order. For instance, the sorted dependent list of *acquired* in the example is $\langle$*Facebook*, *for*, *Whatsapp*$\rangle$. After deciding a dependency relation to binarize, the head and dependent words become targets of the algorithm to expand the binary tree recursively until all the dependency relations are transformed. After the transformation, a binarized tree has words at leaf nodes, and each internal node represents a dependency relation as shown in Figure 2b.

5.2.3 Target Dependent Tree-LSTM

Recall that our objective is to model the information content effect on a target firm mentioned in a sentence. In the previous example, a bidirectional model would output only one representation for the sentence no matter what the target firm is. In contrast, the proposed tree transformation algorithm outputs different binarized trees when different targets are given. Figure 2b and Figure 2c show the binarized trees using *Facebook*[4] and *Whatsapp* as targets, respectively. As in BI-LSTM, we remove stopwords and punctuations from a tree. Figure 2d demonstrates the result after removing *for* and *$* from Figure 2c. When one child is ignored, the current node is replaced by the other child.

When applying a binary Tree-LSTM model on a binarized dependency tree, information is propagated from the bottommost leaf nodes to the topmost dependency node (e.g. sub in Figure 2b), and the final output h is treated as the information embedding g. As the proposed binarization algorithm tends to leave the target at the top of the binary tree, information effectively flows from context to the target firm. For a binary Tree-LSTM model running on a target-dependent tree, we add the prefix *TGT-* to the model identification; otherwise the target is the root word defined by the parser, and *RT-* is prefixed to the model identification.

5.3 Training

We pre-train skip-gram embeddings (Mikolov et al., 2013) of size 100 on a collection of Bloomberg financial news from October 2006 to November 2013, and the size of the trained vocabulary is 320,618. In addition, firm names and an UNK token for representing any words out of the vocabulary are added to the vocabulary, having an initial embedding as the average of the pre-trained word vectors. The word embeddings are fine-tuned during model training, with dropout (Srivastava et al., 2014) using a probability of 0.5 to avoid overfitting. The other hyperparameters for our models along with dependency type representations are initialized according to the method of Glorot et al. (2010)

For sequential LSTMs we use a 2-layer structure with inputs of size 100 and outputs of size 200. For Tree-LSTM models, only one layer is exploited, and the input and output dimensions are the same as that of sequential LSTMs.

Training is done by maximizing the conditional log-likelihood of the target effect category for 15 iterations. The parameters are optimized by stochastic gradient descent with momentum (Rumelhart et al., 1988) using an initial learning rate of 0.005, with L2 regularization at strength 10^{-6}. Every 1,000 training examples the parameters are evaluated on the development set by the macro-averaged F-score, and those achieving the highest value are kept.

6 Experiments Settings

Data: We collect publicly available financial news from Reuters from October 2006 to December 2015. Instead of taking a whole news article or simply a news title into consideration, we target the section of text which appears in the HTML 'class' attribute of 'focus paragraph' of Reuters news articles. This is invariably the first paragraph of such articles, which provide additional detail to the information contained in the article's title. For example, the focus paragraph of the news titled *Exclusive: Target gets tough with vendors to speed up supply chain* (4 May 2016, 12:22pm EDT) is:

Discount retailer Target Corp (TGT.N) is cracking down on suppliers as part of a multi-billion dollar overhaul to speed up its supply chain and better compete with rivals including Wal-Mart Stores Inc (WMT.N) and Amazon.com Inc (AMZN.O).[5]

As *Target Corp*, *Wal-Mart Stores Inc* and *Amazon.com Inc* are mentioned in the paragraph, we assume that the information content of the paragraph affects CAR_3 of these three firms. We ignore focus paragraphs that do not contain any names of U.S.-based, publicly listed firms. Finally, text are grouped per firm per event date, and for each group a CAR_3 is computed accordingly. This yields 22,317 instances, 5,848 of which have information gathered from more than one news. A total of 1,330 firms are covered

[4]Targeting at *acquired* and *Facebook* happen to have the same binarized tree.

[5]http://www.reuters.com/article/us-target-suppliers-exclusive-idUSKCN0XV096

	+CAR_3	-CAR_3
Train	9,674	9,643
Dev	493	507
Test	995	1,005

Table 1: Numbers of CAR_3 in the datasets.

	+CAR_3	-CAR_3
Sentiment-based	0.53	0.55
BI-LSTM	0.61	0.62
RT-LEX-TLSTM	0.62	0.63
RT-CTX-TLSTM	0.62	0.63
TGT-LEX-TLSTM	0.63	0.63
TGT-CTX-TLSTM	**0.64**	**0.66**

Table 2: AUCs of different embedding methods.

	+CAR_3	-CAR_3	+CAR_3 >+2%	-CAR_3 <-2%
Sentiment-based	0.53	0.55	0.59	0.58
BI-LSTM	0.58	0.58	0.66	0.63
TGT-CTX-TLSTM	**0.63**	**0.62**	**0.70**	**0.68**

Table 3: Final AUCs.

in our data. We randomly select 1,000 and 2,000 instances as development and test sets, respectively, and the rest are used for training. The numbers of positive and negative CAR_3 examples in the training, development and test sets are fairly balanced, as shown in Table 1.

Evaluation Metric: Although the task of information content effect prediction is a binary classification problem, we do not evaluate our models using the accuracy metric because the data are automatically aligned and some CAR_3 values may not reflect the information correctly. Instead, we evaluate the performance of the models by the area under the precision-recall curve (AUC), where *precision* is the fraction of retrieved positive/negative effect instances that really have positive/negative impact, and *recall* is the fraction of positive/negative effect instances that are retrieved.

6.1 Development Experiments

Table 2 summarizes AUCs of both positive and negative effect predictions on the development set for models using different embedding methods. First, the deep neural embedding approaches outperform the conventional sentiment-based representation widely exploited in finance research. This shows that deep neural models are stronger in capturing news information on and beyond sentiment signals. Second, compared with the sequential embedding strategy, namely BI-LSTM, those Tree-LSTM based (TLSTM) dependency embeddings perform better, demonstrating the benefit of syntactic information. Finally, the information content prediction can benefit from the target-dependent tree transformation (TGT) compared with that using the root word (RT). In addition, the performance of target-dependent models can be improved by incorporating an input word embedding (LEX) with its contextual information (CTX). It is worth noting that the main improvement comes from using the neural feature representation instead of the sentiment word statistics.

6.2 Final Results

Table 3 gives the AUCs for the baseline sentiment-based representation, the sequential embedding BI-LSTM, and the targeted dependency tree method TGT-CTX-TLSTM evaluated on the test set. TGT-CTX-TLSTM outperforms the other baselines, and the improvements between models are statistically significant ($p \leq 0.05$).

One possible application of the proposed model is the use as a security recommender in the financial domain. Thus we apply the model to instances with $|CAR_3| > 2\%$, namely information with high impact. A total of 1,021 instances in the test set pass this threshold. As shown in Table 3 the proposed model achieves AUCs of 0.7 and 0.68 on +CAR_3 and -CAR_3, respectively. The results not only show the robustness of our model compared to the baselines but also demonstrate its applicability.

To demonstrate the attention mechanism for weighing individual news, Table 4 shows three sets of news, each of which consists news from the same event window and mentioning a specific firm. Both the CAR_3 and the predicted effect probability for each event window are given, and the modeled weight is shown in front of each news, which meets human expectations. For example, one would expect that the news of Wal-Mart Stores Inc missing its profit expectation is more influential than that of being capable of paying shopping using smartphones at Wal-Mart, as shown in the first news group.

- Target: Wal-Mart Stores Inc; CAR_3: -4.7%; TGT-CTX-TLSTM: 0.21
  - 0.95 Wal-Mart Stores Inc's (WMT.N) full-year profit may miss analysts' expectations as growth slows in its international markets, pressuring the company even as its U.S. discount stores continue to prosper.
  - 0.05 A group of big retailers that includes Wal-Mart Stores Inc, Target Corp and Japan's 7-Eleven is developing a mobile payment network, adding to the proliferation of options that let consumers pay with smartphones.
- Target: Oshkosh Corp; CAR_3: 9.8%; TGT-CTX-TLSTM: 0.81
  - 0.66 Activist investor Carl Icahn offered to buy all the outstanding shares of Oshkosh Corp (OSK.N) Thursday for a 21-percent premium to the U.S. truckmaker's closing price on Wednesday, sending its shares to their highest in more than a year.
  - 0.34 Truck maker Oshkosh Corp (OSK.N) advised its shareholders on Thursday to take no action related to activist investor Carl Icahn's offer to buy all outstanding shares in the company for $32.50 each.
- Target: Sony Corp; CAR_3: -2.3%; TGT-CTX-TLSTM: 0.27
  - 0.86 Sony Corp stuck with its full-year profit forecast after slashing its outlook for TV sales, confident that other units will perform better than earlier anticipated to offset additional losses in the unit.
  - 0.14 Japan's biggest technology conglomerates reported quarterly results, with weak TV demand a common theme at both Sony Corp and Sharp Corp.

Table 4: Learned weights for different news.

7 Related Work

Our work is related to research that applies NLP techniques on financial text to predict stock prices and market activities. In terms of corpora, financial news (Leinweber and Sisk, 2011; Xie et al., 2013; Luss and d'Aspremont, 2015; Ding et al., 2015), firm reports (Kogan et al., 2009; Li, 2010; Lee et al., 2014; Qiu et al., 2014) and web content, such as tweets (Bollen et al., 2011; Vu et al., 2012) and forum posts (Das and Chen, 2007; Gilbert and Karahalios, 2010) have been studied. In terms of linguistic features, existing work can be classified into tree major categories: bag-of-words (Kogan et al., 2009; Lee et al., 2014; Qiu et al., 2014), sentiment-based (Das and Chen, 2007; Li, 2010; Bollen et al., 2011; Vu et al., 2012; Luss and d'Aspremont, 2015), and information-retrieval-based (Schumaker and Chen, 2009; Xie et al., 2013; Ding et al., 2015) methods. Our work falls into the category of information retrieval-based features by exploiting syntax information derived from a dependency parser. However, different from the aforementioned work, our goal is not to predict stock prices but to measure the economic value of news information content.

The proposed Tree-LSTM based model for automatically representing syntactic dependencies is in line with recent research that extends the standard sequential LSTM in order to support more complex structures, such as Grid LSTM (Kalchbrenner et al., 2015), Spatial LSTM (Dyer et al., 2015), and Tree-LSTM (Le and Zuidema, 2015; Tai et al., 2015; Zhu et al., 2015; Miwa and Bansal, 2016). We consider the information content prediction as a semantic-heavy task and demonstrate that it can benefit significantly from a novel target-specific dependency Tree-LSTM model.

8 Conclusion

We showed that the impact of information in news release can be predicted using a firm's CAR, and that a proposed target-depend Tree-LSTM model, incorporating contextual information with syntax dependencies, is more effective in representing information content in news text compared to the classic bidirectional LSTM model and a baseline sentiment-based representation. The proposed model can serve as a security assessment tool for financial analysts, and benefit more comprehensive financial models and studies.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ray Ball and Philip Brown. 1968. An empirical evaluation of accounting income numbers. *Journal of Accounting Research*, pages 159–178.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

Kyunghyun Cho, Bart Van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Conference on EMNLP*, pages 1724–1734, Doha, Qatar.

Sanjiv R Das and Mike Y Chen. 2007. Yahoo! for Amazon: Sentiment extraction from small talk on the web. *Management Science*, 53(9):1375–1388.

Angela Kay Davis, Jeremy Max Piger, and Lisa Marie Sedor. 2012. Beyond the numbers: Measuring the information content of earnings press release language. *Contemporary Accounting Research*, 29(3):845–868.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *Proceedings of the Twenty-Fourth ICJAI*, pages 2327–2333.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Eugene F Fama. 1970. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417.

Felix Gers and Jürgen Schmidhuber. 2000. Recurrent nets that time and count. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, pages 189–194.

Eric Gilbert and Karrie Karahalios. 2010. Widespread worry and the stock market. In *ICWSM*, pages 59–65.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256.

Alan Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *ASRU, 2013 IEEE Workshop on*, pages 273–278.

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

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2015. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*.

Shimon Kogan, Dimitry Levin, Bryan R Routledge, Jacob S Sagi, and Noah A Smith. 2009. Predicting risk from financial reports with regression. In *Proceedings of the Conference on NAACL*, pages 272–280. Association for Computational Linguistics.

SP Kothari and Jerold B Warner. 2004. The econometrics of event studies. *Handbook of Empirical Corporate Finance*.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In *Joint Conference on Lexical and Computational Semantics*.

Heeyoung Lee, Mihai Surdeanu, Bill MacCartney, and Dan Jurafsky. 2014. On the importance of text analysis for stock price prediction. In *LREC*, pages 1170–1175.

David Leinweber and Jacob Sisk. 2011. Event driven trading and the'new news'. *Journal of Portfolio Management*, 38(1).

Feng Li. 2010. The information content of forward-looking statements in corporate filings — a naïve Bayesian machine learning approach. *Journal of Accounting Research*, 48(5):1049–1102.

Tim Loughran and Bill McDonald. 2011. When is a liability not a liability? textual analysis, dictionaries, and 10-Ks. *The Journal of Finance*, 66(1):35–65.

Ronny Luss and Alexandre d'Aspremont. 2015. Predicting abnormal returns from news using text classification. *Quantitative Finance*, 15(6):999–1012.

A Craig MacKinlay. 1997. Event studies in economics and finance. *Journal of Economic Literature*, 35(1):13–39.

William J Mayew and Mohan Venkatachalam. 2012. The power of voice: Managerial affective states and future firm performance. *The Journal of Finance*, 67(1):1–43.

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

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. *Preprint arXiv: 1601.00770*.

Barak A Pearlmutter. 1989. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1(2):263–269.

Xin Ying Qiu, Padmini Srinivasan, and Yong Hu. 2014. Supervised learning models to predict firm performance with annual reports: An empirical study. *Journal of the Association for Information Science and Technology*, 65(2):400–413.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling*, 5:3.

Robert P Schumaker and Hsinchun Chen. 2009. Textual analysis of stock market prediction using breaking financial news: The AZFin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):12.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the Annual Meeting of the ACL*.

Paul C Tetlock. 2007. Giving content to investor sentiment: The role of media in the stock market. *The Journal of Finance*, 62(3):1139–1168.

Tien-Thanh Vu, Shu Chang, Quang Thuy Ha, and Nigel Collier. 2012. An experiment in integrating sentiment features for tech stock prediction in twitter. In *Proceedings of the Conference on COLING*.

Boyi Xie, Rebecca J Passonneau, Leon Wu, and Germán G Creamer. 2013. Semantic frames to predict stock price movement. In *Proceedings of the Annual Meeting of the ACL*, pages 873–883.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational linguistics*, 37(1):105–151.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of International Conference on Machine Learning*, July.

Automatic Generation and Classification of Minimal Meaningful Propositions in Educational Systems

Andreea Godea, Florin Bulgarov and Rodney Nielsen
Department of Computer Science and Engineering
University of North Texas, TX, USA
{AndreeaGodea, FlorinBulgarov}@my.unt.edu
Rodney.Nielsen@colorado.edu

Abstract

Truly effective and practical educational systems will only be achievable when they have the ability to fully recognize deep relationships between a learner's interpretation of a subject and the desired conceptual understanding. In this paper, we take important steps in this direction by introducing a new representation of sentences – Minimal Meaningful Propositions (MMPs), which will allow us to significantly improve the mapping between a learner's answer and the ideal response. Using this technique, we make significant progress towards highly scalable and domain independent educational systems, that will be able to operate without human intervention. Even though this is a new task, we show very good results both for the extraction of MMPs and for classification with respect to their importance.

1 Introduction

Over the last few decades, technology has provided us many powerful tools that have completely changed our daily routines. However, one crucial area where technology has yet to have the significant impact suggested by its true promise is in education. Most students around the world have been learning in the same manner for decades. Nevertheless, in the past few years technology has started to increase its role in the learning process and has begun improving the effectiveness of students and instructors. Several groups are developing tools or systems with the goal of improving the feedback provided to students and instructors, assessing students' understanding of a concept, and facilitating their self-guided learning.

Intelligent Tutoring Systems (ITSs) were created with the goal of improving learning through real-time and personalized feedback for students (Graesser et al., 2001; Rosé et al., 2003; Makatchev et al., 2004; Pon-Barry et al., 2004). ITSs need to be able to interpret complex student responses and improve their feedback as they process more questions and responses, but essentially all existing systems require skilled developers to write new rules or train new classifiers for each additional question. Generally, an ITS only provides feedback to students, and when they do provide feedback to instructors, it is typically just high-level information regarding the correctness of the answer. Much of the prior work in this area originated in educational assessment systems (Mitchell et al., 2002; Sukkarieh et al., 2003; Nielsen et al., 2009). Most such systems investigate similarity or entailment relationships between a learner's answer and the reference answer, and then communicate a score to the teacher.

More recently, a new type of educational technology has emerged with the goal of increasing student engagement in classrooms (Paiva et al., 2014). Classroom Engagement Systems (CES) are meant to replace audience response (clicker) systems (Duncan, 2006; Fies and Marshall, 2006) by allowing students to construct answers to free-response questions. Unlike an ITS, the main goal of a CES is to facilitate teacher-student interaction. However, the system Paiva and associates present has some notable weaknesses: the analysis is strictly lexical, all content words are treated with equal importance, and only a small number of student responses are chosen as representatives.

The lack of tools to precisely identify the importance of concepts in the reference answer without manual intervention for each question, and the lack of tools to analyze the nature of a student's response,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3226–3236, Osaka, Japan, December 11-17 2016.

Q: Explain how mass is different than weight.

RA: Mass tells you how much matter an object has. Weight tells you how gravity affects the mass and it changes when gravity changes.

Reference Answer MMPs:		**Student Answer** MMPs:
1. Mass tells you how much matter an object has.^{P}	$\Longleftarrow$	1. Mass is the amount of matter an object holds.
2. Weight tells you how gravity affects the mass.^{P}		2. Weight is how heavy/light something is.
3. Weight changes when gravity changes.^{S}	$\Longleftarrow$	3. Weight gets heavier in higher gravity.

Figure 1: MMP Overview. **P** refers to a primary MMP while **S** denotes a secondary MMP. The entailment symbol signifies that the student understood that MMP.

again without manual coding per question, are significant weaknesses in existing educational technology. To that end, this paper takes important steps to address those weaknesses, introducing methods that will enable educational systems to effectively analyze deep semantic relationships between a learner's answer and a reference answer. Our primary contributions are:

- We introduce the concept of Minimal Meaningful Propositions (MMP), a decomposition of text, such as a question's answer, into the set of propositions that individually represent single minimal claims or arguments that cannot be further decomposed without losing contextual meaning, and taken as a whole represent the entire meaning of the text.
- We present a computational method for breaking text down into its MMPs.
- We present a method, features and categories to classify a reference answer's MMPs, which will allow educational systems to ensure feedback is focused on the most pertinent points.

MMPs are extracted from the reference answer to enable a thorough comparison between it and a student's response in order to diagnose which concepts were understood, misunderstood, or omitted from the response, as well as to determine the importance of those concepts. The research described in this paper will represent a strong foundation for the next generation of fully automated scoring systems. A complete example showing how MMPs are useful is given in Figure 1. Here we show a question, the reference answer, its MMPs, the student answer MMPs and their entailment relations. As can be seen, the student fails to address primary MMP 2 from the reference answer. However, the student successfully understood the concepts expressed in MMPs 1 and 3.

The final outcome of this approach to analyzing student responses will open a variety of new possibilities for fully automated educational systems. For instance, it will support: improved dynamic analysis of student answers to novel questions, the ability to focus on the most important conceptual misunderstandings, the means to provide meaningful feedback to instructors regarding the classroom understanding of concepts, and a construct for more effectively grouping similar answers either for realtime classroom analysis or for assessment purposes. This will allow such systems to be flexible enough to adapt to individual student and teacher needs and to various pedagogical methods. MMPs could also be an effective level of analysis in a wide variety of other NLP applications such as summarization, translation and more general textual entailment. In the following sections we describe the MMP concept and present our methods and results for MMP Extraction and MMP Classification.

2 Related Work

The goal of our work is not only to research means to better assess students' answers in a classroom environment, but also to research tools for more effective and constructive feedback regarding overall understanding of a subject. Although we are the first to introduce the concept of Minimal Meaningful Proposition, other works in the literature have had relatively similar goals (Burrows et al., 2015).

C-rater, a scoring engine developed by ETS, grades a student's answer to assessment questions (Leacock and Chodorow, 2003). C-rater recognizes paraphrases of a set of reference answers to determine wether the student's answer is correct. Although much of the work done by c-rater has been automated in the past years (Sukkarieh and Stoyanchev, 2009), it still requires an appropriate set of responses that have

Data	#Questions	Answ. Words/Question	#MMP	MMP/Q	Primary	Secondary	Extraneous	Redundant
Train	208	25.5	826	4.0	676 (81.8%)	100 (12.1%)	41 (4.9%)	9 (1.0%)
Test	109	22.1	383	3.5	323 (84.3%)	45 (11.7%)	12 (3.1%)	3 (0.7%)
Total	317	24.3	1209	3.8	999 (82.6%)	145 (11.9%)	53 (4.3%)	12 (0.9%)

Table 1: MMP counts and the average length of the reference answers in words.

already been holistically scored by trained raters.[1] In contrast, our approach is fully automated and can be used in a dynamic setting to recognize the focused relationships between a specific reference answer proposition and the student's response. MMPs are also classified for importance in a fully automated, domain-independent fashion using general linguistic features extracted from the reference answer, the question, and their interrelationships.

Another approach with a similar goal is entailment of semantic facets (Nielsen et al., 2009). Here, rather than checking whether the student's answer is a paraphrase of the reference answer as a whole, the authors break the target conceptual knowledge down into fine-grained *facets*, derived roughly from the typed dependencies in a parse of the reference answer. This might allow pinpointing the facet of the reference answer that the student misunderstood at a very fine-grained level, but unlike Minimal Meaningful Propositions, facets are often not *meaningful* without much more context. Hence, entailment of a semantic facet could be misleading with regard to student understanding.

Other related concepts have been introduced in text summarization, question answering and dialog generation. For example, *Elementary Discourse Units* (EDUs), developed for discourse segmentation, are defined as minimal non-overlapping textual spans representing units of discourse. EDUs are generally used as a precursor to identifying relationships between discourse segments. Previous work extracting EDUs has proposed rule-based approaches (Polanyi et al., 2004), classification of discourse boundaries (Soricut and Marcu, 2003; Subba and Di Eugenio, 2007; Afantenos et al., 2010) and sequence labeling (Hernault et al., 2010). However, in contrast with MMPs, EDUs are not necessarily either minimal or meaningful – for example, conditionals required for *meaningful* interpretation of a proposition are not included in the same EDU as their consequent, and a "minimal" discourse unit text span can often be broken into multiple finer-grained *minimal* propositions.

Nenkove and Passonneau (2004) introduced the *Summary Content Unit* (SCU) as a key component of the Pyramid evaluation method for multi-document summarization. SCUs are defined as semantically-motivated, sub-sentential units of variable length and emerge from the annotation of multiple human summaries for the same input. Previous work extracts SCUs manually (Nenkova and Passonneau, 2004; Nenkova et al., 2007) or uses topic modeling to match topics with manually-extracted SCUs (Hennig et al., 2010). However, to the best of our knowledge, there is no model for automatically constructing SCUs. Another important difference is the input data being used. SCUs are extracted from multiple, well-structured human summaries; whereas, MMPs emerge from a single version of a potentially poorly-structured answer. A review of SCUs also finds that, while they are *meaningful*, they are not necessarily *minimal* – many SCUs are syntactically complex and would be divided into multiple MMPs.

3 Data Description

The data used in our experiments consists of 317 questions that were asked in real science classes from middle school. Each question comes with the teacher's reference answer, which was decomposed into MMPs by two graduate students (from education and science major) and adjudicated by a third (from Education and Linguistics). Each of the first two annotators labeled the data independently and the adjudicator decided the correct label among the existing annotations.

The first stage of the annotation was identifying the MMPs in the instructors' reference answers. Annotators were provided guidelines for restating a reference answer as its corresponding set of minimal meaningful propositions, or distinct stand-alone claims, and given several guiding examples.

The second stage of the annotation process was to classify the MMPs into one of the following classes:

[1] https://www.ets.org/research/topics/as_nlp/written_content/

1. *primary*: fundamental to answering question
2. *secondary*: relevant but not integral to answer – often clarify or qualify a primary MMP
3. *extraneous*: unnecessary or minimally relevant to the question
4. *redundant*: contain information directly or indirectly provided by the question

As can be seen in Table 1, 1209 MMPs were annotated in total, with an average of 3.8 MMPs per answer. 999 MMPs were annotated as primary, 145 as secondary, 53 as extraneous and 12 as redundant. Training and test sets were created by randomly splitting the questions (2/3 in training and 1/3 in test).

4 Minimal Meaningful Propositions

Consider a sentence to be comprised of a set of related propositions. We define an MMP as a *proposition* that cannot be broken down into finer-grained propositions (it is *minimal*) and still be interpretable without further context (it is *meaningful* on its own). A sentence usually contains more than one MMP. Note that the MMPs only state explicit propositions, not any implications, presuppositions, or entailments. Moreover, in our case, an MMP should relate to the question in a way or another, when treated independently. The goal of the present work is to automatically extract MMPs from a question's reference answer and classify them according to their importance. The example below is a real question and reference answer asked in a classroom, and its human-extracted MMPs.

Q: How did Rutherford figure out that atoms are mostly empty space, and that the nucleus is positive?

RA: He used gold foil hammered about an atom thick, and placed radium in a lead lined box that emitted positive alpha particles towards the gold foil.

MMPs:

1. Rutherford used gold foil with the thickness of an atom.
2. Rutherford placed radium in a lead lined box.
3. The lead lined box emitted positive alpha particles towards the gold foil.

Extracted MMPs can contain information that was initially spread out over the sentence. These finer-grained propositions allow us to separate the different pieces of information expected from a student and classify their importance. An educational system that successfully uses MMPs will be able to tell the teacher which concepts were understood, contradicted, or omitted by the students.

5 MMP Extraction

A high level summary of the MMP extraction process is as follows. First, in the learning phase, we learn the unique set of syntactic patterns covering all of the gold-standard human generated MMPs. Then, in the application or testing phase, we process sentences recursively, on each call extracting the MMP associated with the longest matching pattern learned from the training set and recursively processing the remainder of the sentence. If part of the sentence remains and no further patterns match, the remainder forms the final MMP. The details of this process follow.

In the *learning* phase, the algorithm learns structural templates from a shallow parse (i.e., chunks) of the gold-standard MMPs in the training data. These templates will be used to extract MMPs from test set answers. Figure 2 shows an MMP and the structure extracted. Table 2 shows the most frequent structures in the dataset. The frequencies follow a Zipfian distribution, with the five most common structures covering almost 50% of the MMPs.

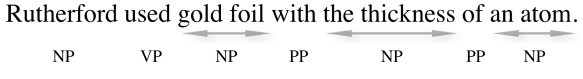


Figure 2: MMP Structure

Rank	Structure	%
1	*NP, VP, NP*	19.5
2	*NP, VP, NP, PP, NP*	11.3
3	*NP, VP, PP, NP*	8.5
4	*NP, VP*	7.6
5	*NP, PP, NP, VP, NP*	2.6

Table 2: Most Frequent MMP Structures

In the *test* phase, the algorithm first splits the answer into sentences, which it parses using Stanford CoreNLP (Manning et al., 2014). Then conjunctions are automatically pre-processed to: replace enu-

	F_1-score			BLEU score			
	Avg. P	Avg. R	F_1	1-grams	2-grams	3-grams	4-grams
MMP Extraction	**0.725**	**0.552**	**0.627**	**0.569**	**0.495**	**0.360**	**0.172**
Predicates Baseline	0.437	0.375	0.404	0.290	0.229	0.160	0.107
Sentence Baseline	0.438	0.449	0.443	0.377	0.315	0.246	0.154

Table 3: MMP Extraction Results

merations with a single base phrase type, and split conjoined SVO (i.e., Subject Verb Object) structures into separate sentences, replicating the subject and verb as appropriate.

Then, for each sentence, the algorithm finds the longest structure matching a pattern learned during training. If the matching pattern only covers a portion of the sentence, the algorithm extracts that portion as an MMP and recursively processes the unmatched portion of the sentence. If any base phrases remain unmatched when the recursion bottoms out, they become the final MMP. Due to the nature of the algorithm, our method is generalizable to different questions types or domains.

Given our example question **Q** in Section 4, the automatically extracted MMPs are:

1. He used gold foil hammered about an atom thick.
2. Placed radium in a lead lined box.
3. Emitted positive alpha particles towards the gold foil.

As can be seen, the automatically extracted MMPs are very similar to the human-generated examples. The major difference being the missing subject in the last 2 MMPs, which we will address in future work.

5.1 Results

To evaluate the performance, we compared the set of system-generated MMPs with the gold standard for each question. We pre-processed both system-generated and gold-standard MMPs to remove stop words and stemmed the remaining words using the Porter Stemmer. We report the Precision, Recall and F_1-score as well as the BLEU score. Precision is computed as the number of matching words divided by the total number of words in the system-generated MMP. Recall is the number of matching words divided by the total number of words in the gold-standard MMP. The BLEU score, introduced in (Papineni et al., 2002), is a highly-adopted method for automatic evaluation of machine translation systems. BLEU is based on a modified computation of *precision*, using the number of matching n-grams. It ranges from 0 to 1, with values closer to 1 representing more similar texts. Using different values of n, we can measure different aspects of the evaluation, *adequacy*: n = 1-2, and *fluency*: n = 3-4.

MMP-level metric values are based on a greedy iterative alignment of system-generated and gold-standard MMPs, where on a given iteration the algorithm aligns, processes, and then removes the pair with the highest F_1-score (or BLEU score). MMP-level values are averaged to get a question-level value, and finally, Table 3 presents the average over all questions.

For comparison we also computed two baselines. The *Sentence Baseline* is a method in which every sentence of the reference answer is treated as an MMP. In the *Predicates Baseline*, we build an MMP for every predicate in a sentence. Using SENNA (Collobert et al., 2011), we then identify the predicate's arguments and attach them to the MMP. As can be seen, the method that we propose, *MMP Extraction*, significantly outperforms the two baselines, achieving an F_1-score of 0.627 and showing that the pattern matching approach generalizes well on new, unseen questions. The precision is higher than the recall, meaning that the system generated MMPs are shorter than the human-generated ones.

The BLEU score achieved for different n-gram sizes are also considerably higher than the baselines'. The *adequacy* scores, unigrams and bigrams, are fairly high for this new task. As we increase the number of consecutive words to be scored, the score drops. This is a normal behavior for tasks where different solutions to a problem exist without any compromises. When using the BLEU metric to score the *fluency*, trigrams and 4-grams, it is strongly recommended that you have more than one reference solution or, in our case, human-annotated MMPs. When a single reference solution exists, as in our case, substantially lower values are expected (Papineni et al., 2002). For comparison, in a translation task, on a test corpus of about 500 sentences, a human translator scored 0.346 against four references and scored 0.257 against

two references, when $n = 4$ (Papineni et al., 2002).

5.2 Error Analysis

Some of the most common errors occurring in the extraction phase are associated with the subjectivity of the task. Consider the following reference answer and its system-generated MMPs:

RA: *Once light reaches our eyes, signals are sent to our brain and our brain deciphers the information in order to detect the appearance, location and movement of objects.*

1. Light reaches our eyes.
2. Signals are sent to our brain.
3. Brain deciphers the information.
4. In order to detect the appearance, location and movement of objects.

First, in the human-annotated set of MMPs, 1 and 2 are joined into a single proposition: *When light reaches our eyes, signals are sent to our brain*. The annotators believed that the two pieces of information were too dependent to be separated. On the other hand, the system found two different claims and therefore, extracted two propositions. In addition, the fourth MMP generated by the system is dependent on the context and thus, not very good. The annotators broke this piece of text into three different MMPs, one for each element detected by the brain. In future, we plan to solve this issue using the semantic roles of constituents. We will also include coreference resolution to improve detecting agents.

Other errors made by the system can be fixed by including more syntactic and semantic information. For example, in a student response where the object does not immediately follow the predicate, our algorithm can get confused, and in some cases will not include the object as part of the MMP. In future work, we will check the validity of verb usages in an external resource such as VerbNet (Schuler, 2005), which will help us distinguish, for example, between transitive and intransitive verbs.

6 MMP Classification

Given a question, reference answer and its MMPs, our goal is to identify the importance of each MMP with respect to the question. We follow a supervised approach to classify MMPs as *primary, secondary, redundant* or *extraneous*. Many of the features the classifier uses to determine this importance compare the MMPs to the question. Hence, in preparation for feature extraction, we pre-process the question to eliminate unnecessary information and identify its key concepts. We then extract a number of features and train a classifier to predict labels for each MMP.

6.1 Question Pre-processing

Questions often include instructions to guide students on issues unrelated to content (e.g. *answer in $\leq$ 2 sentences*). An analysis of such text revealed it can provide unnecessary errors to our model. Therefore, we filtered out text matching patterns indicative of such instructions. Three cases were explored:

Hints. We filtered out sentences starting with the keyword *hint*, if the hint was unrelated to other question content (e.g. Why do models change over time? *Hint: think about why your idea changed.*). The hint sentence was deemed related if the Pointwise Mutual Information (PMI) between any of its words and those of the rest of the question exceeded a threshold of $T = 0.28$, which was learned from annotated word pairs (*related* vs. *not related*) from the training data. In the example above, the hint was removed, since no relationship was found with the preceding sentence.

Punctuated Instructions. While analyzing the training data, several additional patterns indicating content-independent instructions were discovered. For example, in the question: *"Using the tables, describe how you know the object is accelerating."*, the instruction *using the tables* is unrelated to understanding the core *object acceleration* concept. In a substantial number of patterns, the instructions were delimited by punctuation, as in this case. Rules were developed to detect and filter out these punctuation-delimited content-independent instructions.

Parentheticals. Parenthesized text often includes important abbreviations, definitions or examples, but it can also contain content-independent instructions. Based on analysis of the training data, we wrote rules to filter out parentheticals that: 1) contain imperative statements: *... (Do not touch your eyes!)*, 2)

start with negation: *What gives an atom its VOLUME (not its mass)?*, or 3) do not contain a noun, since this is highly correlated with instructions: *... the dependent variable (what we measure).*

6.2 Key Concept Identification

We identify a subset of the key concepts in the question for use in feature extraction. Specifically, we focus on concepts identifiable using part-of-speech tags and grammatical dependency relations, which covers the vast majority of key concepts. Generally key concepts are expressed as nouns, but many are also expressed as adjectives (*Explain what* homogeneous *means*) or gerunds (*Explain what* weathering *means*). The most common dependency types associated with key concepts are: adjectival modifiers (`amod`), noun compound modifiers (`nn`), and copulas (`cop`). However, we only consider dependency word pairs that are collocations as determined by applying the *Likelihood Ratio* statistic, using word counts from a large corpus consisting primarily of *Gigaword* (Graff and Cieri, 2003). If a dependency pair is not a collocation only constituent words matching previous selection criteria are used (e.g., since the dependency concept *equal forces* is not a collocation, only *forces* is retained). Given the question: *What evidence can indicate if a change is physical?*, the system identifies the key concepts: *evidence* and *physical change*.

6.3 Classification

To classify MMPs, we follow a supervised approach, training Random Forests on features indicative of the classes. However, since the original data lacks of *redundant* MMPs (see Table 1), we enriched the *training* data by adding 64 manually generated examples. Following the patterns in the training data, we created redundant MMPs by using information already stated in the question or paraphrasing parts of it.

Feature Engineering. Using information from the training data, we manually designed features based on the question, the MMPs, the reference answer, and the relations between them. From all the features explored, we chose the set that performed best in 10-fold cross-validation (10xCV) on training data. The final set of features is described in Table 4.

Note that two versions of the question were considered in this process: the original question (Q) and a version based strictly on its interrogative and imperative sentences (Q′). Various types of features were explored: 1) *features derived from the question representation* – $\{1, 2, 4\}$, 2) *semantic features* – $\{5, 18, 25, 27\}$, 3) *syntactic features* – $\{6, 7, 8, 9\}$, 4) *features focused on mismatches between the MMP and question* – $\{20, 21, 26\}$ and 5) *features focused on overlapping information* – $\{3, 10, 13, 14, 15\}$.

QUESTION FEATURES

1.	Q contains structures similar to *"tell me what you know"*.
2.	Q has only one concept.
3.	Q has only one concept and it appears in MMP.
4.	Q has only one MMP attached.
5.	Q′'s concepts have hyponyms/hypernyms in MMP. (Fellbaum, 1998)
6.	Q′'s subject is lemma in MMP.
7.	Q′'s subject is concept in MMP.
8.	Q′'s subject is subject in MMP.
9.	Q′'s predicate appears in MMP.

MMP FEATURES

10.	#overlapping lemmas between MMP and Q, excluding stop words.
11.	max PMI of MMP words (not in Q) and Q words.
12.	min PMI of MMP words (not in Q) and Q words.
13.	all MMP words appear in Q.
14.	all MMP words appear in a Q's sentence.
15.	all MMP words appear in a Q's interrogation.
16.	at least one MMP concept addressed by Q.
17.	MMP's predicate occurs in Q.
18.	MMP's predicate has hyponyms/hypernyms in Q.
19.	MMP's subject is subject in an answer sentenceaddressing Q.
20.	MMP provides a reason, although Q did not ask for it.
21.	MMP provides more information than required by Q.
22.	MMP is relevant for Q.
23.	the current MMP embeds other MMP.
24.	MMP's subject in Q′'s concepts.
25.	MMP's subject relates to Q′'s concepts using hyponymy or PMI.
26.	MMP's subject not in Q′, but refers to previous MMPs.
27.	max similarity score between MMP and Q′'s sentences.

Table 4: Description of Features

6.4 Results and Discussion

We report Precision, Recall and F_1-score per class, using both a *strict* evaluation, based on adjudicated labels, and a *relaxed* evaluation, where the system is credited for matching either annotator's label. The results from 10xCV on the training data are presented in Table 5. As can be seen, *primary* and *redundant* are the best performing classes – they follow more recognizable patterns than the others. Whereas, *secondary* is the worst performing class. The vast majority of system errors on the *secondary* class are

predictions of *primary*. Similarly, in analyzing the human annotation disagreements for MMPs adjudicated as *secondary*, the confusion was with *primary* in the vast majority of the cases – discriminating between *secondary* and *primary* is hard for humans as well as the system. This suggests there is more of a continuum between *primary* and *secondary* rather than a sharp decision boundary. This will be explored further in future work.

While the increase in the F_1-score for the *relaxed* evaluation is more than 6% for *secondary*, when viewed as a relative reduction in the error rate, the increase is considerable for all classes, ranging from 10% to 27%. This provides further motivation for a future investigation into the similarity in the errors in system and human judgements. Despite challenges, the high system F_1-score demonstrates the feasibility of the task and the promise of the system.

Next, we compare the performance to the majority class baseline, where each instance is classified as *primary*. Table 6 reports the weighted F_1-score achieved on both the test set and in 10xCV on training. The proposed approach outperforms the baseline in both scenarios. Our method shows an improvement of almost 16% in 10xCV compared with the baseline. Employing the *relaxed* evaluation, our performance is higher than the baseline with almost 16% on 10xCV and 7% on the *test* set.

The system outperformed the baseline under all scenarios (Table 6). It shows a slight improvement on test, even though the baseline results are 12% higher on test than on training. This difference in the baseline is due to the addition of redundant MMPs to the training data, as detailed earlier.

	Strict Matching			Relaxed Matching		
Classes	P	R	F_1	P	R	F_1
Primary	0.840	0.970	0.900	0.884	0.976	0.927
Secondary	0.675	0.250	0.365	0.835	0.303	0.429
Redundant	0.875	0.662	0.753	0.856	0.797	0.810
Extraneous	0.882	0.365	0.517	0.950	0.514	0.588

Table 5: Training Set 10xCV Results

	Training 10xCV		Test	
System/Approach	Strct F_1	Rlxd F_1	Strct F_1	Rlxd F_1
Proposed Approach	**0.810**	**0.857**	**0.815**	**0.876**
Majority Class	0.653	0.701	0.770	0.804

Table 6: System weighted F_1-score vs. Baseline

Ablation study. To assess the contribution of different categories of features, we performed an ablation study. Table 7 reports the weighted F_1-score when training on *all* of our features, and when training on all features except the specified set. The final column shows the increase in the error after removing the feature set, as a percent of 0.185 – the error when training on all features.

All feature categories have a substantial contribution to the results. The features derived from *Mismatching Information* between the question and MMP are especially useful – their removal results in a 21% relative increase in the error. The *secondary* and *extraneous* classes suffer the largest increase in error when *Mismatching Information* is not leveraged. This is logical since the *extraneous* information is not directly related to the question and *secondary* MMPs are optional and less directly tied to the question than *primary* MMPs. *Semantic features* have the second largest impact. While they help all classes, the greatest impact is on *primary* and *secondary* classes. MMPs with these classes contain highly relevant information for the question and the semantic features help identify indirect semantic relationships between parts of the MMP and the question. *Syntactic features* also have a positive contribution, especially for the *redundant* class. This is likely due to a more regular pattern in the mapping between components of a redundant MMP and the syntactic structure of the question. The *Question Representation features* also make a substantial contribution, particularly in classifying *primary*, *secondary* and *extraneous* MMPs. For example, if the question contained a single key concept and the MMP did not address it, the MMP is probably *extraneous*; whereas, if the MMP is strongly related to the key concept, then it is likely *primary* or *secondary*.

Feature Category *Removed*	**Wtd F_1**	**RIE%**
None (trained on *all* features)	0.815	–
Syntactic Features	0.800	8
Question Features (Q)	0.794	11
Question Representation Features	0.790	14
Semantic Features	0.787	15
Mismatching Information	0.776	21

Table 7: Feature ablation: weighted F_1 with *relative* percent increase in error (RIE) on Test

7 Conclusion

This work has resulted in three notable contributions. First, we introduced the concept of Minimal Meaningful Propositions and discussed how it can enhance feedback and accuracy in applications such as educational assessment. Second, we described an effective method for automatically extracting MMPs. The results of this approach were shown to be considerably higher than two meaningful baselines (0.184 absolute improvement on F_1 over a better baseline – 33% error reduction), validating the approach. Third, we successfully classified MMPs with respect to their importance, achieving a weighted F_1-score of 0.815 on the test set. This will enable applications, such as ITS, to respond appropriately based on different types of MMPs. This work introduces a new fully automated, domain-independent foundation for analyzing students' free responses, at a level of granularity that is appropriate for contextual comparison. The corpus described in this paper is publicly available for research purposes and represents a substantial contribution to multiple NLP sub-communities. This will quicken the pace of much related research. The annotated corpus along with the annotation guidelines will be available from the H*i*LT Lab Resources webpage.[2]

Acknowledgements

This research was supported by the Institute of Education Sciences, U.S. Department of Education, Grant R305A120808 to University of North Texas. Opinions expressed are those of the authors.

[2] http://hilt.cse.unt.edu/resources.html

References

Stergos Afantenos, Pascal Denis, Philippe Muller, and Laurence Danlos. 2010. Learning recursive segments for discourse parsing. *arXiv preprint arXiv:1003.5372*.

Steven Burrows, Iryna Gurevych, and Benno Stein. 2015. The eras and trends of automatic short answer grading. *International Journal of Artificial Intelligence in Education*, 25(1):60–117.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Douglas Duncan. 2006. Clickers: A new teaching aid with exceptional promise. *Astronomy Education Review*, 5(1):70–88.

Christiane Fellbaum. 1998. A semantic network of english verbs. *WordNet: An electronic lexical database*, 3:153–178.

Carmen Fies and Jill Marshall. 2006. Classroom response systems: A review of the literature. *Journal of Science Education and Technology*, 15(1):101–109.

Arthur C Graesser, Xiangen Hu, Suresh Susarla, Derek Harter, NK Person, Max Louwerse, Brent Olde, et al. 2001. Autotutor: An intelligent tutor and conversational tutoring scaffold. *10th ICAI in Education*, pages 47–49.

David Graff and C Cieri. 2003. English gigaword corpus. *Linguistic Data Consortium*.

Leonhard Hennig, Ernesto William De Luca, and Sahin Albayrak. 2010. Learning summary content units with topic modeling. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 391–399. Association for Computational Linguistics.

Hugo Hernault, Danushka Bollegala, and Mitsuru Ishizuka. 2010. A sequential model for discourse segmentation. In *Computational Linguistics and Intelligent Text Processing*, pages 315–326. Springer.

Claudia Leacock and Martin Chodorow. 2003. C-rater: Automated scoring of short-answer questions. *Computers and the Humanities*, 37(4):389–405.

Maxim Makatchev, Pamela W Jordan, and Kurt VanLehn. 2004. Abductive theorem proving for analyzing student explanations to guide feedback in intelligent tutoring systems. *Journal of Automated Reasoning*, 32(3):187–226.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Tom Mitchell, Terry Russell, Peter Broomhead, and Nicola Aldridge. 2002. Towards robust computerised marking of free-text responses.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method.

Ani Nenkova, Rebecca Passonneau, and Kathleen McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(2):4.

Rodney d. Nielsen, Wayne Ward, and James h. Martin. 2009. Recognizing entailment in intelligent tutoring systems*. *Nat. Lang. Eng.*, 15(4):479–501, October.

Frank Paiva, James Glenn, Karen Mazidi, Robert Talbot, Ruth Wylie, Michelene TH Chi, Erik Dutilly, Brandon Helding, Mingyu Lin, Susan Trickett, et al. 2014. Comprehension seeding: Comprehension through self explanation, enhanced discussion, and inquiry generation. In *Intelligent Tutoring Systems*, pages 283–293. Springer.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Livia Polanyi, Chris Culy, Martin Van Den Berg, Gian Lorenzo Thione, and David Ahn. 2004. A rule based approach to discourse parsing. In *Proceedings of SIGDIAL*, volume 4.

Heather Pon-Barry, Brady Clark, Karl Schultz, Elizabeth Owen Bratt, and Stanley Peters. 2004. Contextualizing learning in a reflective conversational tutor. In *Advanced Learning Technologies, 2004. Proceedings. IEEE International Conference on*, pages 236–240. IEEE.

Carolyn P Rosé, Antonio Roque, Dumisizwe Bhembe, and Kurt Vanlehn. 2003. A hybrid text classification approach for analysis of student essays. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing-Volume 2*, pages 68–75. Association for Computational Linguistics.

Karin Kipper Schuler. 2005. Verbnet: A broad-coverage, comprehensive verb lexicon.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics.

Rajen Subba and Barbara Di Eugenio. 2007. Automatic discourse segmentation using neural networks. In *Proc. of the 11th Workshop on the Semantics and Pragmatics of Dialogue*, pages 189–190.

Jana Z Sukkarieh and Svetlana Stoyanchev. 2009. Automating model building in c-rater. In *Proceedings of the 2009 Workshop on Applied Textual Inference*, pages 61–69. Association for Computational Linguistics.

Jana Z Sukkarieh, Stephen G Pulman, and Nicholas Raikes. 2003. Auto-marking: using computational linguistics to score short, free text responses. In *th annual conference of the International Association for Educational Assessment (IAEA), Manchester, UK*.

First Story Detection using Entities and Relations

Nikolaos Panagiotou
National and Kapodistrian
University of Athens
npanagio@di.uoa.gr

Cem Akkaya
Yahoo! Research,
Sunnyvale, CA
cakkaya@yahoo-inc.com

Kostas Tsioutsiouliklis
Yahoo! Research,
Sunnyvale, CA
kostas@yahoo-inc.com

Vana Kalogeraki
Athens Univerisity of
Economics and Business
vana@aueb.gr

Dimitrios Gunopulos
National and Kapodistrian
University of Athens
dg@di.uoa.gr

Abstract

News portals, such as Yahoo News or Google News, collect large amounts of documents from a variety of sources on a daily basis. Only a small portion of these documents can be selected and displayed on the homepage. Thus, there is a strong preference for major, recent events. In this work, we propose a scalable and accurate First Story Detection (FSD) pipeline that identifies fresh news. In comparison to other FSD systems, our method relies on relation extraction methods exploiting entities and their relations. We evaluate our pipeline using two distinct datasets from Yahoo News and Google News. Experimental results demonstrate that our method improves over the state-of-the-art systems on both datasets with constant space and time requirements.

1 Introduction

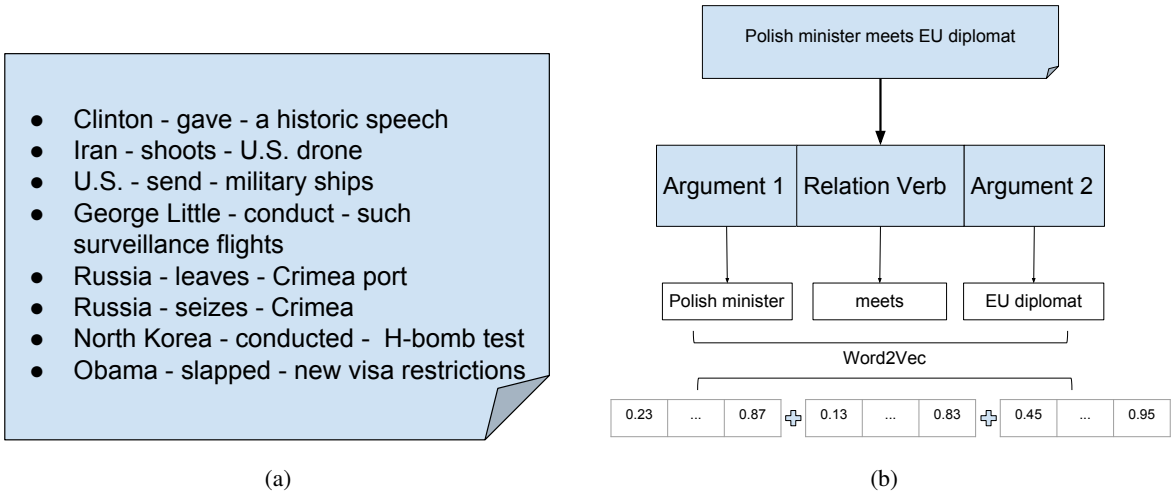
First Story Detection (FSD) is the task of detecting the first document about a new event given a stream of documents (Allan et al., 1998; Allan et al., 2000a). The task is also known as New Event Detection (NED). The problem appears in several real-world applications where news stories are accumulated and presented to users in near real-time. A FSD system should be accurate, scalable, and process a stream of articles in a single pass. The output of such a system is very valuable to news portals such as Yahoo News, Google News, since the rapid detection of a new event is crucial for the service reputation. FSD is a very challenging, if not impossible, task for human operators, since it requires inspecting millions of documents per day. As a result, accurate, automatic solutions are very desirable.

The majority of the FSD systems in the literature attempt to classify a document as a first story if the document differs significantly from those published before and thus may describe a new event. This is accomplished usually in two steps. In the first step, the nearest neighbor of a new document in the previous document stream is identified. In the second step, the similarity between the new document and its nearest neighbor is considered in order to decide if it is a first story or not. In this methodology, the selection of an appropriate similarity metric and the selection of an informative document representation are essential. As a result many different metrics have been studied in the literature, such as the cosine similarity (Petrović et al., 2012), the KL-Divergence (Karkali et al., 2013), and the named entities overlap (Kumaran and Allan, 2004). So far, terms and entities occurring in a document have been the main representation units, boiling down the task to detecting documents with novel terms and entities in a stream.

We believe that existing approaches under-utilize the semantic information present in news articles, specifically, actions performed by entities, or interactions that happened between a pair of entities. These actions and interactions are essential, since they describe events, around which a news story revolves. Our hypothesis is that often the same entities and terms may appear in documents that describe different events – sometimes related – leading to false negatives. For example, when North Korea conducted the hydrogen bomb test in January of 2016, the U.N. security council called for an emergency meeting. This resulted in two different events in news streams in the following order: (1) North Korea conducted the H-bomb test, (2) the U.N. announced a meeting about North Korea. The articles about the two events have a very similar term and entity distribution. Nevertheless, the extracted relations about North Korea and the U.N. differ in the two story lines. For the first story line, relations such as "N.Korea - concerns - U.N." are detected while for the second story line relations such as "U.N. - announces - meeting about North Korea" are present.

Motivated by this observation, we propose to utilize relations between entities when deciding if an article should be classified as a first story. For this purpose, we define a relation as an entity-action-entity triplet (see Figure 1a)

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers,
pages 3237–3244, Osaka, Japan, December 11-17 2016.



(a) (b)

Figure 1: (a) Relations extracted from different news articles. (b) Our proposed relation representation.

describing events and sub-events of a story. In addition, we rely on a distributed representation to represent our relation triplets in order to overcome lexical variation.

To the best of our knowledge, our approach is the first one to integrate relations between entities in a FSD system. Our experiments demonstrate that our system improves the detection error trade-of (DET) on a Yahoo News dataset by up to 29.6% compared to the best unsupervised system and up to 17.3% compared to the best supervised system. At the same time, we show that the space and time requirements remain constant over time suggesting that the method is suitable for very high volume streams.

The contributions of this work can be summarized as follows:

- We build an efficient, stream-based pipeline for detecting fresh news that uses traditional term similarity metrics amplified by relation and entity similarity information
- We propose a novel approach to model a document as a set of named entities and their relations.
- We make annotation and preprocessing tools as well as preprocessed datasets available [1].

The rest of the paper is organized as follows. In Section 2, we briefly describe related work on FSD, as well as recent advances in relation extraction. Section 3 follows with the description of our system. First, we provide details of a basic scalable FSD system. Then, we describe our proposed entity-relation document model and present our system in detail. In Section 4, we describe the evaluation procedure, the datasets, and the systems we compared with. In Section 5, we provide experimental results and analysis. Section 6, concludes the paper.

2 Related Work

First Story Detection has been extensively studied in the literature. One of the best-performing FSD systems proposed was UMASS (Allan et al., 2000b). This system retrieves the nearest neighbour of a new document. Then the system calculates a novelty score for the new document as the cosine distance, using incremental TF-IDF weighted vectors, to its nearest neighbor. This score is used to decide if it is a first story or not. In a work proposed by (Stokes and Carthy, 2001), the authors use two distinct document representations when searching for the nearest neighbor while on (Brants et al., 2003) the authors use different TF-IDF models per document category during the search. All these approaches leverage primarily statistical information about terms found in a document.

The same principle was extended in (Kumaran and Allan, 2005) where the authors incorporated information about entities, topics and also used a supervised classifier to perform the detection. In this work, we also use features that exploit the named entities. However, we extend the entity usage by capturing also the way that the entities interact.

The above approaches are not designed to scale with massive streams. Efficient first story detection was recently studied by (Petrović et al., 2010) where the authors proposed a constant time and space solution. They redesign UMASS (Allan et al., 2000b) utilizing a locality sensitive hashing (LSH) multi-index and apply the system on the Twitter stream. We deploy the same algorithm in our system for scalability. The (Petrović et al., 2010) system was also implemented in (McCreadie et al., 2013) as a storm topology that is able to process the Twitter Firehose in

[1] Supplementary paper material available at: `https://bitbucket.org/npan1990/firststory-annotator`

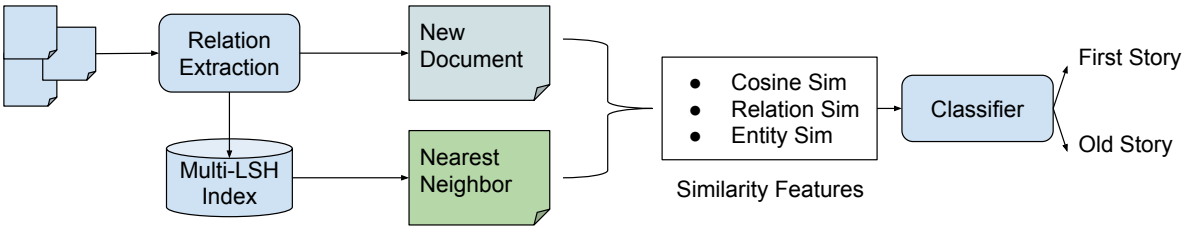


Figure 2: The First Story Detection pipeline we propose.

real-time using 70 processing units according to the author claims. On the same direction, the authors in (Karkali et al., 2013) develop an efficient approach that completely avoids the nearest neighbor identification by defining the novelty of a document as the novelty of its terms.

In addition, (Petrović et al., 2012) provides an extension of the system described in (Petrović et al., 2010) that addresses the synonymy problem by expanding the term vectors with paraphrases. The new system yielded a 13% improvement in detection error trade-of without significantly increasing the computational complexity. In our work we also address the synonymy problem, but from a different perspective through Word2Vec (Mikolov et al., 2013).

The method that we propose highly depends on a robust relation extraction mechanism. Since we are interested in generic relations independent of a predefined taxonomy, we rely on an open information extraction system. These systems detect open domain relations by self-training over a massive corpus (Banko et al., 2007) or heuristic rules (Fader et al., 2011; Etzioni et al., 2011; Schmitz et al., 2012). They allow the development of very scalable systems. The relations extracted often have a generic format of two arguments that are connected by a verb (see Figure 1a). In our work, we use OpenIE 4.1 (Etzioni et al., 2011) the successor of ReVerb (Fader et al., 2011), Ollie (Schmitz et al., 2012) and TextRunner (Banko et al., 2007).

3 The Proposed First Story Detection Pipeline

We design and implement a novel pipeline to solve the FSD task. In this section, we describe the main aspects of our pipeline. We begin by describing the basic approach and a scalable extension that uses locality sensitive hashing in Section 3.1. Then, in Section 3.2, we describe a holistic document representation based on relations and a similarity function that compares documents using this representation. Finally, in Section 3.3, we present how we identify first stories using various similarity features and a supervised classifier.

3.1 A Basic First Story Detection Pipeline

A generic pipeline for detecting first stories consists of two basic steps. In the first step, as soon as a new document arrives, the nearest neighbor is identified. In the second step, a similarity function is considered in order to measure how similar the new document is to its nearest neighbor and decide if it is a new story. This basic design was first instantiated by the UMASS system described in (Allan et al., 2000b). It uses cosine similarity and incremental TF-IDF document vectors and achieved state-of-the-art performance during the topic detection and tracking challenge (Fiscus and Doddington, 2002). Subsequently it was improved along two axes: (i) improved scalability by exploiting approximate nearest neighbor techniques, proposed by (Petrović et al., 2010), that use locality sensitive hashing, (ii) improved accuracy by addressing the synonymy problem through exploiting syntactic paraphrases (Petrović et al., 2012).

Our approach follows the basic First Story Detection paradigm. The main novel aspects of the pipeline we propose are: (i) New features are extracted improving the resolution of the similarity function. These features exploit the **relations** between the entities that appear in a document and we show how they can be obtained and incorporated in a novel similarity measure between documents. (ii) The various similarity features extracted are given to a supervised classifier that learns how to combine them, and decides if a new document is a new story. Our complete First Story Detection pipeline is illustrated in Figure 2.

3.2 Entity-Relation Document Representation

In this section, we describe how we model the entities and their relations in a document. In addition, we propose Relation Similarity (RelSim), a metric for comparing two documents in terms of their named entities and their relations. Our technique uses state-of-the-art relation extraction algorithms that extract the relationship between two arguments in a 3-tuple format (see Figure 1a).

Simplifying the Relations

The relation arguments are n-grams and in many cases consist of large text chunks or even sub-clauses. However,

these large text chunks add noise to the relations and rarely reappear on other documents making relation comparison a difficult task. For that reason, we employ a set of simplification heuristics in order to convert large relation arguments to small n-grams. Here is a summary of the rules:

- We require the first relation argument to contain a simple named entity. If it does, then the argument is replaced by the named entity.
- We require that the second relation argument is not a sub-clause. If it is we remove the relation.
- If the second argument contains a named entity the argument is replaced by the named entity. If it contains more than one named entities the relation is split into multiple relations.
- From the second relation argument we keep only the nouns and the adjectives.
- From the relationship verbs we keep only the core verb that expresses the action. Modals and auxiliary verbs are removed.

During the relation extraction, we also extract the named entities along with their types[2]. We keep only the named entities of type *Person*, *Location*, and *Organization*. In many cases, we observed that an entity was mentioned explicitly only once in the text, and then was referenced implicitly through pronouns in later sentences. So, after we have identified the entities and their types, we propagate them to the following sentences while replacing the pronouns (e.g. He-met-Putin translates to Obama-met-Putin).

Relation Representation

The UMASS system represents the documents as TF-IDF weighted vectors. However, in order to compare documents in terms of their relations we need a relation-oriented representation.

Definition 1 Relation: *A relation is defined as a 3-tuple* $r = (arg1, action, arg2)$. arg1 *is the main entity or actor of the relation.* arg2 *is the recipient of that action (e.g. Putin - meets - Obama) or a preposition that describes the action (Obama - landed - Thursday). The action is a simple verb.*

Definition 2 Bag of Relations: *The relation set* $R_d = \{r_1, \ldots, r_{|R_d|}\}$ *for a document d.*

In many cases the actor of a relation could have multiple textual representations or surface forms (e.g. Obama, Barack Obama, President of US). In addition, multiple relation verbs could express the same action. For example, the relations $r_1 = (A, proposes, X)$ and $r_2 = (A, suggests, X)$ have probably the same semantic meaning. Thus, we decided to exploit Word2Vec, a technique described in (Mikolov et al., 2013) that learns a vector representation for words. All the three relation parts were converted into their corresponding vectors. If the representation of a n-gram (e.g. "U.S. President") was not directly available we used the average vectors of the n unigrams. The resulting three vectors are concatenated to a large vector. The concatenated vector of a relation $r_1 = (A, meets, X)$ will be different than the concatenated vector of the relation $r_2 = (X, meets, A)$. Under this technique, similar relations will have a similar vector representation addressing this way the synonymy problem. The procedure is also illustrated in Figure 1b.

Comparing documents using relations

So far we have described how to represent a document as a set of simplified relations. However, it is unclear how to compare two documents using their relations. Thus, we propose Relation Similarity (RelSim), a metric that compares semantically two documents using their named entities and relations.

Assume that we have two documents d_1 and d_2 that we need to compare, each with its relations R_{d1} and R_{d2}, respectively. For each relation $r_i \in R_{d1}$ the method $simRD(r_i, d_2)$ returns a relation score rs_i that is equal to the cosine similarity with the most similar relation $r_j \in R_{d2}$. While searching for the most similar relation from d_2, $SimRD(r_i, d_2)$ also checks the distance to the inverse of the relation r_i. This decision was taken since in some cases, a relation r_i may be expressed in a reverse form on the document we compare. For example, the inverse relation of $r_i = (Obama, meets, Putin)$ is the relation $r'_i = (Putin, meets, Obama)$. The method $SimRD(r_i, d_2)$ is defined on Equation 1.

The document to document similarity $SimDD(d_1, d_2)$ is the average relation score rs_i for every relation $r_i \in R_{d1}$ with the document d_2 and is defined on Equation 2. Since $SimDD(d_1, d_2)$ is not a symmetric function it is not suitable for a similarity metric. The final relation similarity $RelSim(d_1, d_2)$ defined on Equation 3 is the metric we use to compare two documents in terms of their relations. It is important to note that the computational complexity of RelSim is $O(|R_{d1}| * |R_{d2}|)$. However, the relations identified per document are on average only 10 with a standard deviation of 9. Thus, the computational complexity of RelSim won't affect the system scalability.

$$SimRD(r_i, d_2) = \max_{r_j \in R_{d2}} (max(Cos(r_i, r_j), Cos(r'_i, r_j))) \quad (1)$$

$$SimDD(d_1, d_2) = \frac{sum_{r_i \in R_{d1}}(SimRD(r_i, d_2))}{|R_{d1}|} \quad (2)$$

[2]We used the Stanford CoreNLP tool available at: `http://stanfordnlp.github.io/CoreNLP/ner.html`

Feature	LSH-RelFSD	LSH-RelEntFSD
$CosSim(d, d_n)$	✓	✓
$RelSim(d, d_n)$	✓	✓
$EntOverlap(d, d_n)$	✗	✓
$RelEntOverlap(d, d_n)$	✗	✓

Table 1: The similarity features between d and d_n used by our methods.

$$RelSim(d_1, d_2) = \frac{SimDD(d_1, d_2) + SimDD(d_2, d_1)}{2} \quad (3)$$

3.3 Entity-Relation FSD

We only focused on extracting relations from news articles, simplifying them and using them as a distance metric for document comparison. Having these ingredients, in this section we describe how to use the relation similarity (*RelSim*) discussed above in order to perform first story detection. We present two supervised approaches, *LSH-RelFSD* and *LSH-RelEntFSD*, that address the FSD task as a binary classification problem. The first uses as features the cosine similarity and the relation similarity between and new document d and its nearest neighbor d_n. The latter uses also entity similarity features. The classifier we use, since the method is supervised, is a Logistic Regression. The features used for a new document d and its nearest neighbor d_n are presented on Table 1. CosSim is the term cosine similarity employed also by UMASS. RelSim is the similarity function described in the previous section. EntOverlap is the overlap of the entities that appear on the documents and RelEntOverlap is the overlap of the entities present in the relations.

4 Evaluation Setup

In this section we describe our evaluation framework. We provide details about the datasets, the annotation process, the system configuration, and the pipeline parameters.

4.1 Datasets

We decided to evaluate the scalability and accuracy of our FSD pipeline on three datasets: (1) A Yahoo News dataset (D1) under the category "Politics and Government" that we created using an event-guided annotation procedure described in (Petrović et al., 2012). In order to simplify the annotation process for the dataset (D1) we implemented a user-friendly web interface, which allows the easy labelling of the documents in terms of events. We make the tool available[3]. This dataset consists of 652 documents, 89 are first stories and 563 are non first stories. (2) A Google News dataset (D2) that was proposed in (Karkali et al., 2013). This dataset consists of 2006 documents about "Technology" where 1491 are annotated as first stories and 515 are annotated as non first stories. Clearly, the datasets (D1) and (D2) have different label distributions. (3) A much larger synthetic dataset (D3) of $106,000$ documents from Yahoo News is used in order to evaluate the scalability of our method. However, this dataset is not annotated and thus is not used for evaluating the accuracy.

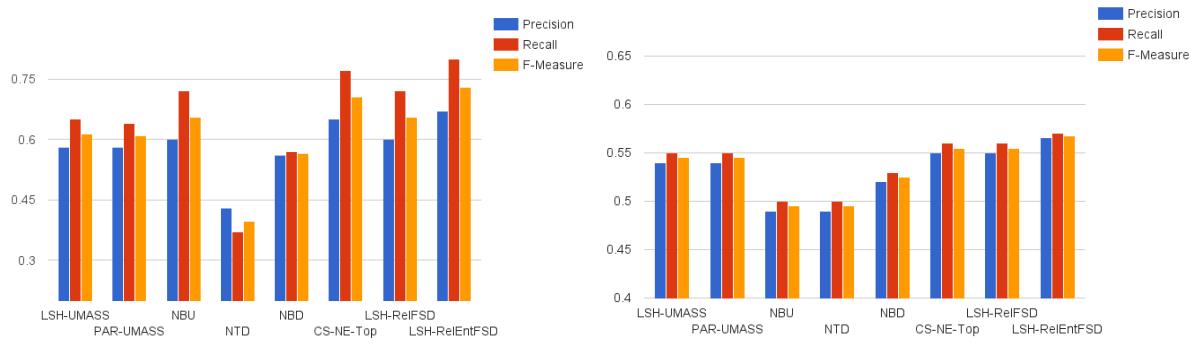
4.2 System Instantiation

In order to compare our system with existing state-of-the-art systems, we implemented to the best of our ability the following systems: (i) LSH-UMASS by (Petrović et al., 2010) and (ii) PAR-UMASS by (Petrović et al., 2012). (Karkali et al., 2013) define several measures to evaluate the novelty of a document's terms; we use here the ones they suggest that perform best for the content, namely (iii) NTD, (iv) NBD, (v) NBU. (vi) CS-NE-Top suggested by (Kumaran and Allan, 2005). LSH-UMASS, PAR-UMASS and CS-NE-Top are similar to our method in that they also initially detect the nearest neighbor of a new document. LSH-UMASS uses the incremental TF-IDF weighted term vectors while PAR-UMASS expands the vectors according to a pool of syntactic paraphrases. CS-NE-Top uses the similarity of the term, the entity and the non-entity (topic) vectors and similarly to our technique uses a classifier in order to combine the multiple similarity features.

LSH-UMASS, LSH-RelFSD and LSH-RelEntFSD use locality sensitive hashing as described in (Petrović et al., 2010), to efficiently identify the nearest neighbor. The required multi-LSH index parameters were set to $K = 5$ and $L = 20$ which result in missing a nearest neighbor of distance 0.3 with probability $\delta = 0.025$. The maximum LSH bucket size was set to 1000. For the PAR-UMASS system paraphrases available online[4] with Precision more than 0.4 are used. For NBU, NTD, and NBD we set the required parameter N to 60.

[3] `https://bitbucket.org/npan1990/firststory-annotator`

[4] `http://paraphrase.org/#/download`



(a) Dataset (D1)

(b) Dataset (D2)

Figure 3: FSD systems comparison in terms of Precision, Recall and F-Measure.

4.3 System Description

We preprocessed our dataset (D3) on a large commercial Hadoop Cluster. The preprocessing was implemented via a Pig script in order to effectively allocate the required resources and run the Map-Reduce job. The job took less than 2 hours to extract the necessary metadata for the January 2016 dataset (D3). Datasets (D1) and (D2) are much smaller than (D3), and the preprocessing was done on a single machine. For evaluating our pipeline accuracy and performance we used a 4-core Intel i7 CPU with 32GB of RAM.

4.4 Evaluation Metrics

In order to evaluate the FSD task we used Precision, Recall, and F-Measure. Datasets (D1) and (D2) are fully labeled, so it is possible to calculate the above metrics averaged over the two classes. Also, a commonly used metric for FSD is the detection error trade-off (DET) score. DET score is defined in Equation 4 and depends on the probability of missing a first story P_{miss} and the probability of a false alarm P_{fa} for a specific threshold τ. For each system we find the threshold τ that minimizes P_{fa} and P_{miss} and achieves the lowest DET score DET_{min}. The costs of a false alarm and missing a first story C_{fa} and C_{miss} are set to 1.0, P_{target} is set to 0.5 similarly to (Karkali et al., 2013). All the evaluation metrics are calculated under 5-fold stratified cross validation.

$$DET = C_{miss} * P_{miss} * P_{target} + C_{fa} * P_{fa} * (1 - P_{target}) \quad (4)$$

5 Experimental Results and Discussion

5.1 First Story Detection Performance

In Figure 3a, we present the performance of various FSD systems on the Yahoo News dataset (D1). Clearly, the systems that incorporate entity or relation information have a significant advantage on this dataset. LSH-RelEntFSD achieves a F-Measure of .73 and CS-NE-Top, which comes second, achieves a F-Measure of .70. LSH-RelFSD and NBU systems that follow on this dataset achieve a F-Measure of .65. In terms of Precision, LSH-RelEntFSD and CS-NE-Top report .67 and .64 while their Recall is .81 and .77 respectively. The rest of the systems score F-Measure values between .40 and .61.

In Figure 3b, we present the performance on the Google News dataset (D2). LSH-RelEntFSD achieves the best performance with a F-Measure of .57. CS-NE-Top and LSH-RelFSD that follow both report a F-Measure of .55. The systems LSH-UMASS and PAR-UMASS that come next achieve a F-Measure of .54. The Precision and the Recall for LSH-RelEntFSD is $\sim$ .57. The remaining systems report F-Measure values up to .52.

The results for the DET_{min} evaluation metric are shown in Table 2. The best performance on both datasets is achieved by LSH-RelEntFSD and CS-NE-Top. On the Yahoo news dataset (D1) our LSH-RelEntFSD system achieves a 17.3% improvement over the state-of-art supervised system CS-NE-Top and a 29.6% improvement over the best unsupervised system NBU. On the Google news dataset (D2), LSH-RelEntFSD achieves a 4.4% improvement in the DET_{min} score over the best unsupervised system.

The improvement in Det_{min} score is statistically significant for the dataset (D1) at the $p < 0.05$ level using a paired t-test. However, for dataset (D2) all methods perform close to the best method (LSH-RelEntFSD) and so the results are not statistically significant. To understand our results in (D2) we explored the nature of this dataset in more detail and discovered that the dataset contains many non-news articles, such as product descriptions, reviews as well as personal opinions. The impact is two-fold: firstly, incorporating entity and relation information on these articles is not as important as in dataset (D1) about "Politics and Government" where many named entities

Method	$Det_{min}(D1)$	$Det_{min}(D2)$
LSH-UMASS	.35	.45
PAR-UMASS	.36	.45
NBU	.27	.5
NTD	.62	.5
NBD	.42	.47
CS-NE-Top	.23	.44
LSH-RelFSD	.27	.44
LSH-RelEntFSD	**.19**	**.43**

Table 2: Det_{min} scores for the Yahoo (D1) and Google (D2) datasets.

participate and interact. In addition, we discovered that for several of the articles in (D2), in particular among those that are reviews, descriptions or opinions, there is significant ambiguity whether they should have been classified as new stories or not. This clearly impacts the performance of all the techniques.

Our proposed pipeline outperforms all the methods used on both datasets. This strengthens the conclusion that a linear classifier that combines multiple similarity features is essential in order to effectively address the task at hand. The supervised system CS-NE-Top is the closest competitor reporting similar performance as LSH-RelEntFSD on dataset (D2) while LSH-RelEntFSD outperforms CS-NE-Top on dataset (D1) by 17.3%. This improvement results mainly from taking into account the entity relations in addition to the entity and term similarity features.

5.2 Space and Time requirements

Since our system should be able to process hundreds of thousands or even millions of documents per day we ensure that the algorithm's time and space requirements do not increase as the stream progresses. Figure 4 illustrates the processing time required per document on the large Yahoo News dataset (D3). Clearly, it shows that the processing time per document remains steady over time and it is less than $20ms$ on average. This result suggests that the processing time per document does not grow with the stream time making the method suitable for high volume streams. The four large spikes on the figure are caused due to the memory allocation and cleaning operations performed by the Java virtual machine. In addition, the memory usage is also steady and did not exceed $18GB$ of which $7GB$ were required by the Word2Vec model.

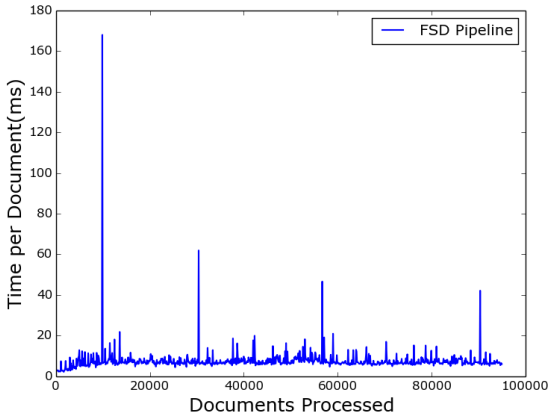


Figure 4: Processing time per document.

6 Conclusions

In this work, we proposed a scalable first story detection pipeline that exploits relations between entities in order to deduce the freshness of a document. To the best of our knowledge, our approach is the first one to integrate relations between entities in a FSD system. We proposed a novel document representation based on relation triplets and described a similarity function on that representation. The positive results on two datasets provide evidence that incorporating relation information helps to detect new events. Another advantage of our method is that it addresses indirectly the synonymy problem through the usage of Word2Vec in the relation representation. Finally, we demonstrated that our system is scalable with constant space and time requirements by investigating the behaviour on a large dataset.

Acknowledgments.

This research has been financed by the European Union through the FP7 ERC IDEAS 308019 NGHCS project, the Horizon2020 688380 VaVeL project and a Yahoo Faculty Research & Engagement Program.

References

James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. 1998. Topic detection and tracking pilot study final report.

James Allan, Victor Lavrenko, and Hubert Jin. 2000a. First story detection in tdt is hard. In *Proc of the ninth international conference on Information and knowledge management*, pages 374–381. ACM.

James Allan, Victor Lavrenko, Daniella Malin, and Russell Swan. 2000b. Detections, bounds, and timelines: Umass and tdt-3. In *Proc of topic detection and tracking workshop*, pages 167–174.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction for the web. In *IJCAI*, volume 7, pages 2670–2676.

Thorsten Brants, Francine Chen, and Ayman Farahat. 2003. A system for new event detection. In *Proc of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 330–337. ACM.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. 2011. Open information extraction: The second generation. In *IJCAI*, volume 11, pages 3–10.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proc of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Jonathan G Fiscus and George R Doddington. 2002. Topic detection and tracking evaluation overview. *Topic detection and tracking*, pages 17–31.

Margarita Karkali, François Rousseau, Alexandros Ntoulas, and Michalis Vazirgiannis. 2013. Efficient online novelty detection in news streams. In *Web Information Systems Engineering–WISE 2013*, pages 57–71. Springer.

Giridhar Kumaran and James Allan. 2004. Text classification and named entities for new event detection. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 297–304. ACM.

Giridhar Kumaran and James Allan. 2005. Using names and topics for new event detection. In *Proc of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 121–128. Association for Computational Linguistics.

Richard McCreadie, Craig Macdonald, Iadh Ounis, Miles Osborne, and Slobodan Petrovic. 2013. Scalable distributed event detection for twitter. In *Big Data, 2013 IEEE Int Conference on*, pages 543–549. IEEE.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to twitter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 181–189. Association for Computational Linguistics.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2012. Using paraphrases for improving first story detection in news and twitter. In *Proc of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 338–346. Association for Computational Linguistics.

Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. 2012. Open language learning for information extraction. In *Proc of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics.

Nicola Stokes and Joe Carthy. 2001. Combining semantic and syntactic document classifiers to improve first story detection. In *Proc of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 424–425. ACM.

Textual complexity as a predictor of difficulty of listening items in language proficiency tests

Anastassia Loukina, Su-Youn Yoon,
Jennifer Sakano, Youhua Wei, Kathleen M. Sheehan
Educational Testing Service
660 Rosedale Road,
Princeton, NJ 08541, USA
{aloukina,syoon,jsakano,ywei,ksheehan}@ets.org

Abstract

In this paper we explore to what extent the difficulty of listening items in an English language proficiency test can be predicted by the textual properties of the item text. We show that a system based on multiple text complexity features can predict item difficulty for several different item types and for some items achieves higher accuracy than human estimates of item difficulty.

1 Introduction

Many language proficiency tests measuring listening or reading comprehension consist of multiple questions or "items" of varying complexity. For simpler items, finding the correct answer may be easy even for relatively low-proficiency speakers while other, more difficult, items require higher levels of proficiency. In addition, multiple sets of items or "forms" are usually created for large-scale language tests to reduce the possibility that the test takers are already familiar with a particular item. In order to be fair to all test takers, such forms need to be comparable in terms of difficulty of the items they contain. Therefore item difficulty is a crucial parameter for every new item added to the test. In test theory, item difficulty is defined by the proportion of the test takers who answer the item correctly (Holland and Thayer, 1985). This value can be estimated empirically by a pilot study before the item is used in an actual test. However, reliable estimates of item difficulty require a substantial amount of test taker responses. There are also concerns about item exposure, especially if an item is to be used in a high-stakes test.

In this study we consider an automated system for item difficulty prediction that can help obtain quick estimates of item difficulty for new items and assist test developers in creating items of varying complexity while retaining form comparability. The novel contribution of this study is the application of the state-of-the-art findings related to predicting the difficulty of spoken texts to predicting the difficulty of listening items in a language proficiency test.

Listening comprehension items often consist of a recorded passage followed by a printed or recorded multiple choice question related to this passage. Several studies have investigated the factors that affect the difficulty of such listening and reading test items. Previous work on item difficulty prediction (Boldt and Freedle, 1996; Freedle and Kostin, 1996; Nissan et al., 1995; Rupp et al., 2001) identified three types of variables that affect item difficulty. The first category are the variables related to the text such as its length or information density. The second group of variables consider various properties of the question, for example, the number of negatives in correct and incorrect responses ("distractors") to the question or lexical overlap between the correct and incorrect responses. Finally, the last group of variables considers interaction between the text and the question, for example, what kind of information a test taker needs to extract from the text to provide the correct answer. In these studies a relatively small number of features from all of these groups were moderately predictive of item difficulty with an R^2 around 0.3. More recently, Hoshino and Nakagawa (2010), Susanti et al. (2016), and Beinborn et al. (2014) developed item difficulty prediction systems for other types of items such as those contained in Cloze tests or vocabulary tests. They used features that assess (a) the difficulty of a passage, (b) the difficulty of a correct answer,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3245–3253, Osaka, Japan, December 11-17 2016.

and (c) the similarity between the correct answer and the distractors, but the detailed implementation of features and the performance in predicting item difficulty substantially varied across different studies. These studies were also conducted on relatively small numbers of items and test takers.

While previous studies found that all three categories of features were predictive of item difficulty, many of the features were designed for particular item types and therefore required new features to be developed for new item types. In this study, we use a large corpus of items to investigate whether item difficulty can be estimated using a large pool of more general text complexity features that can be applicable across a wide range of items. In contrast to previous studies such as Beinborn et al. (2014), which were generally limited to items comprised of short texts, some item-types in this study assess the understanding of relatively long texts, and this further motivates our use of generic text complexity features in the item difficulty prediction.

Compared to the relatively limited number of studies on item difficulty, there exists extensive research on the factors that affect ESL reading and listening comprehension as well as automated text difficulty prediction systems for spoken and written texts. Mature automated text difficulty prediction systems (Landauer et al., 1998; Sheehan et al., 2013) are already available and have been used operationally to evaluate reading materials selected for use in instruction and assessment. For listening materials, a comprehensive list of features known to affect a spoken text difficulty has been recently provided by Bloomfield et al. (2010) who classified these features along four dimensions: length, complexity, organization, and auditory features. Moreover, many studies explored individual features. Thus syntactic complexity and vocabulary difficulty features have been found to have a strong and consistent effect on reading and listening comprehension (Blau, 1990; Nissan et al., 1995). For the auditory dimension, speaker accent, disfluencies, audio quality (background noise), and speaking rate were found as important factors (Blau, 1990; Brindley and Slatyer, 2002). Based on these findings, Kotani et al. (2014) and Yoon et al. (2016) developed fully automated spoken text difficulty prediction systems and reported promising performances.

Of the four dimensions identified by Bloomfield et al. (2010), the auditory dimension of the items in this study was generally held almost constant since acoustic characteristics such as speaker accent, speaking rate, and audio quality were closely monitored during the original recording. As a result, they were very homogenous. Therefore, we focused on the first three dimensions: length, complexity and organization of the item text, and we explored whether these can predict item difficulty as measured empirically.

This study focuses on the following aspects:

- We use a set of generic text complexity features for four largely different item-types.
- We use a large set of items with an objective item difficulty index estimated by students' performance from a large scale language proficiency test.
- We provide further insight about the relationship between item difficulties and passage difficulties.

2 Data and methodology

2.1 Corpus of listening items

We used a large collection of listening items from an international English language proficiency test. The corpus comprised four types of multiple choice items listed here in increasing order of complexity:

- Picture description (*PD*): Test takers are presented with a picture and four recorded statements. Their task is to select the statement that best describes the given picture.
- Dialogue completion (*DC*): Test takers hear a question or a statement and several possible responses. Their task is to choose which response is the most appropriate.
- Conversation (*C*): Test takers listen to a conversation between two speakers and answer a series of printed questions about the content of the conversation.

- Monologue (*M*): Test takers listen to a recording of a monologue (e.g., announcement or advertisement) and answer a series of printed questions about the content of the recording.

Compared to *PD* and *DC*, the *C* and *M* items have a more complex structure. For these two item types the items are clustered into testlets or sets with three items per testlet. All items in the same testlet refer to a common listening passage (conversation or monologue). The listening passages in *C* and *M* items are also much longer than in *PD* and *DC* items; the number of words ranged from 41 to 174 and the mean was 97.17. *C* and *M* item are further classified into gist items that assess understanding of the central idea and detail items that assess understanding of details.

The total number of items used in this study is presented in Table 1. The items were included in 154 operational test forms administered from 2009 to 2013 to two groups of test takers referred to hereafter as Group A and Group B. Both groups included English language learners, but the two groups took test in different countries. The number of test takers for each admin was around 30,000 on average, and the minimum number of test takers was 3,599.

Group	Total	*PD*	*DC*	*C*	*M*
Group A	5092	520	1569	1501	1502
Group B	6436	671	2011	1872	1882
Both	1694	165	511	504	514
Total	9834	1026	3069	2869	2870

Table 1: The number of items of different types across the two populations

2.2 Item difficulty indicators

2.2.1 EQDelta

The difficulty of each item was estimated by delta. It is computed by Eq. 1,

$$\Delta = 13 - 4Z_p \tag{1}$$

where p is the proportion of the test taker population who answered the item correctly and Z_p is the Z value corresponding to a given p value in the standard normal distribution (Holland and Thayer, 1985).

Thus on the delta scale, the higher delta values denote more difficult items. In the testing program, the observed deltas obtained from different test forms were equated to a common scale, so that the equated deltas can be compared across test forms. These empirically obtained equated deltas ("EQDelta") were used as a response variable in our study.

The descriptive statistics of item difficulty are shown in Table 2. The means of different item-types ranged from 11.54 to 13.08.

Group	Item-type	Minimum	Maximum	Mean	Std. Deviation
Group A	*PD*	6.50	17.20	11.54	2.21
	DC	7.50	16.50	12.36	1.43
	C	8.50	16.70	12.99	1.32
	M	9.00	17.30	13.08	1.33
Group B	*PD*	6.20	17.50	11.94	2.14
	DC	7.70	17.30	12.45	1.45
	C	8.40	16.80	12.75	1.42
	M	7.60	17.00	12.88	1.44

Table 2: Descriptive statistics of EQDelta

2.2.2 Item sequence

The Equated Deltas can only be computed after the item has been administered to a certain number of test takers. When creating new items, test developers rely on their experience and prior understanding of item difficulty drivers to vary the difficulty of the items. This predicted difficulty is generally used to sequence the items within each form in increasing order of difficulty. Note that the relationship between the item sequence and item difficulty is less direct for *C* and *M* items where several items are combined in sets. In this case the placing of the items in the form are determined by the overall difficulty of the whole set and therefore item order is a less useful indicator of the perceived difficulty of individual items within each set. Nonetheless in the absence of other criteria, item sequence can be used as a proxy for expert human estimate of item difficulty, a useful benchmark for the automated system for item difficulty prediction.

2.3 Text complexity features

In order to obtain text features, we used an automated text complexity prediction system, *TextEvaluator* (Sheehan et al., 2013; Sheehan et al., 2014; Napolitano et al., 2015). *TextEvaluator* is a fully automated system which was created to assess the complexity of reading passages for both native and non-native speakers including the complexity of reading items in English Language proficiency tests (Chen et al., 2015).

TextEvaluator generates 339 raw features based on vocabulary lists and various NLP technologies such as tagging and automated parsing. The features were designed to measure the degree of complexity along the four dimensions: vocabulary, syntax, cohesion, and discourse. These features can be categorized into the follow subgroups:

- Academic Vocabulary (vocabulary): features in this dimension measure the proportion of academic vocabulary in the text. e.g., the frequency of academic words normalized by text length
- Concreteness (vocabulary): features in this dimension measure the degree of concreteness of the text by aggregating concreteness rating of each word in the text.
- Word unfamiliarity (vocabulary): features in this dimension measure the difficulty of vocabulary used in the text. e.g., the frequency of rare words normalized by text length
- Syntactic Complexity (syntax): features in this dimension measure the complexity of grammatical structures used in the text. e.g., the average frequency of long sentences, and average number of words per sentence.
- Cohesion (cohesion): this dimension includes two different types of features (cohesion): frequency of content word overlap and frequency of casual conjuncts.
- Argumentations (cohesion): this dimension includes two features: the frequency of concessive and adversative conjuncts
- Conversational style (discourse): features in this dimension measure the proportion of words related to conversational text (e.g., conversational verbs, communication verbs, and contractions)
- Degree of narrativity (discourse): features in this dimension calculate the frequency of expressions related to narrativity (e.g., frequency of past tense verbs and frequency of 3rd person singular pronouns.)

TextEvaluator analyzed the features of the entire item: the audio, the question, and the options for item types *PD* and *DC*. For item types *C* and *M*, which consist of an audio stimulus share between the three items and 3 items consisting of a question related to this stimulus and possible responses, *TextEvaluator* extracted separate sets of features for the the stimulus and for each item.

Item	Group A			Group B		
	IS	TC	V	IS	TC	V
PD	0.40	0.49	0.41	0.58	0.52	0.47
DC	0.44	0.40	0.33	0.40	0.40	0.28
C	0.09	0.24	0.16	0.22	0.27	0.19
M	0.22	0.33	0.29	0.20	0.26	0.19

Table 3: Model performance (Pearson's r between predicted and observed EQDelta) based on item sequence (IS), all text complexity features (TC) and vocabulary features only (V) for two populations of test takers

2.4 Experiment design

The goal of the experiment was to evaluate how well *TextEvaluator* features can predict EQDelta for each item. For a benchmark we used the system which predicts the EQDelta based on a single feature: the position in which the item appears in the form, which to some degree reflects the estimate of item complexity assigned to the item by test developers[1].

We built separate models for each of the two populations and four different item types. For *C* and *M* items, which consisted of both recorded propmt and printed questions, we trained 3 types of models: the complexity of a recorded passage only, the complexity of printed questions only, and both sets of features.

For each group, the data were randomly split into training (50%), development (25%) and evaluation (25%) partitions. For *C* and *M* items the partitions were created so that the items related to the same stimulus were always in the same partition. The models were trained on the training set, fine-tuned using evaluations on the development set and finally evaluated on held-out evaluation set.

We used 9 regressors available via SKLL package (Blanchard et al., 2016) to map either the item sequence or the text complexity features to the EQDelta. These included ordinary least squares linear regression, LASSO regression, decision tree regression, elastic net, k-neighbours regression, stochastic gradient descent regression, linear and non-linear support vector regressions and random forest regression. The model performance was evaluated by correlation between predicted and observed EQDelta on new data.

3 Results

The analysis of model performance on the development set showed that for *TextEvaluator* features random forest regressor consistently outperformed other learners. All classifiers produced similar results for the benchmark system based on item sequence. Therefore for consistency we used random forest regressor for both benchmark and complexity-based models.

3.1 Item sequence benchmark

The dependency between item sequence and its empirically established difficulty varied between the item types (see Table 3). Item sequence was the strongest predictor of item difficulty for the two simpler items, *PD* and *DC*, with r varying between 0.40 and 0.58. As expected, the performance was noticeably lower for *C* and *M* (r between 0.09 and 0.22) since these items are combined into sets and sequenced by the overall difficulty of the sets. These patterns were observed for both populations.

3.2 Text complexity

We first looked at the performance of the models based on text complexity of the whole item presented to the test taker. For *PD* and *DC* the complexity features were computed on the script of the recorded item text. For items *C* and *M* these models contained features computed for the recorded passage and printed questions.

[1]For items that have been used in several forms we average both EQDelta and item position across these forms

Item type	Group A		Group B	
	IS	TC	IS	TC
PD	0.50	0.42	0.42	0.42
DC	0.45	0.44	0.34	0.32
C	0.08	0.31	0.20	0.32
M	0.18	0.29	0.12	0.26

Table 4: Model performance (Pearson's r between predicted and observed EQDelta) based on item sequence (IS) and Text complexity (TC) on the held-out evaluation set

The performance of the complexity-based models generally followed the same pattern as was previously observed for sequence-based models: the correlation between predicted and observed items decreased from simpler to more complex items with r around 0.4-0.5 for *PD* items and r around 0.24-0.33 for *M* and *C*.

When compared to human benchmark, for *PD* and *DC* items the models based on text complexity produced similar results. At the same time, the complexity of the text for *C* and *M* items was a better predictor of item difficulty than the item sequence benchmark.

In order to further investigate the relationship between item type and item difficulty prediction, we analyzed the distribution of EQDelta using training data partition.

If EQDelta was not normally distributed for a certain item type (e.g., with a skew towards easy or difficult items), it may have increased the difficulty of the automated prediction and decreased the correlation between predicted difficulty and EQDelta. However, we did not find such tendency; all item types showed a normal distribution. We also found that EQDeltas for *PD* item type were more widely distributed than other items. The standard deviation was approximately 1.4 times greater than for the other item types. This wider distribution of EQDeltas may contribute to higher performance of the automated system for *PD* item type.

Finally, we explored what features contributed most to the final prediction by computing the variable importance for each feature (Breiman, 2001). We found that for all item types the most highly ranked features were related to the lexical content of the item text. These 26 features covered three aspects of vocabulary: first, vocabulary diversity measured as type-to-token ratio in item text. Second, the difficulty of vocabulary in the item text as measured by the frequency of the words in different corpora; Third, the concreteness and imageability of the text (Sheehan et al., 2013). Table 3 shows the performance of all models considered in this study.

3.3 Model performance on the evaluation set

Our analysis of system performance on the development set reported in the previous sections showed the best performance was achieved by the system based on all text complexity features and random forest regressor. We therefore evaluated the performance of these final models on the held-out evaluation set that had not been used for any other analyses.

The results were consistent with what was observed on the development set. These are presented in Table 4.

3.4 Comparison between recorded and printed parts of the item

Two types of items in our study, *C* and *M*, consisted of a listening passage shared between different items within a testlet and printed questions unique for each item. For these two item types we compared performance of text complexity features computed on different parts of the items. *C* and *M* items infer the test takers' understanding of the listening passage. Therefore we initially hypothesized that the text complexity of a common listening passage would be a strong predictor of the item difficulty. However, the empirical results did not support this.

For both populations for the *C* items the performance of the models based on the listening passage was slightly better than the one based on the printed question. There was further improvement from

combining the two parts of the items. At the same time for *M* items the performance of the model based on the listening passage was substantially lower than that of the model based on the printed question. There also was only a very little improvement when the listening passage was added to printed question. These results are presented in Table 5.

Type	Item part	Group A	Group B
C	Listening passage	0.20	0.22
C	Printed question	0.18	0.17
C	Both	0.24	0.27
M	Listening passage	0.19	0.12
M	Printed question	0.30	0.24
M	Both	0.33	0.26

Table 5: Baseline performance (Pearson's r) for models based on different parts of complex items

4 Discussion

In this paper we explored to what extent the text complexity of the item text can be used to predict item difficulty. We compared text complexity-based prediction system to the benchmark system based on item sequence, which we treated as an indication of test developer intuition about item difficulty.

As expected, the benchmark model based on item sequence (experts' judgements) performed worse for *C* and *M* items than for *PD* and *DC* items. As discussed earlier, the former are grouped into sets and are not sequenced according to difficulty of individual items. However, even for simpler items the item sequence was not a strong predictor of item difficulty. One reason for this is that factors other than item difficulty may affect the item sequence in the form and therefore item position may not always accurately reflect the expert judgment about its difficulty. At the same time, this result is also consistent with Beinborn et al. (2014)'s results where English language teachers classified items into four groups (very easy, easy, medium, and difficult). The inter-rater agreement for three raters was not high (Fleiss κ = 0.36). This shows that the item difficulty rating is a challenging task even for experts.

Item text complexity was a stronger predictor of item difficulty for less complex item types (*PD* and *DC*) than for more complex item types. Interestingly, the best performance was observed for *PD* items even though the system had no access to the information about the graphic part of the item. For more complex items, *C* and *M*, the text complexity of the item text appeared to have a smaller contribution to overall item difficulty. Furthermore, we found that for *M* items the textual complexity of the longer listening passage shared between several items was less predictive of item difficulty than the textual complexity of the actual question and responses.

To investigate this further, we conducted the following experiment. First, for each item type, we calculated a range of deltas per each set of items (testlet-condition). Next, we paired three randomly selected items and calculated a range of deltas (random-condition). We conducted a descriptive analysis of delta ranges for these two conditions. Table 6 summarizes the results of this experiment.

The range of difficulty among items within the same testlet (or set) was quite wide. The average difficulty ranges were slightly lower than random-condition, but not largely different. If the overall difficulty of listening passage had substantial impact on the item difficulty, the difficulties of the items that shared the same listening passage might have been within a small range and therefore the difficulty ranges in the testlet condition would be smaller than for the random sets. However, we did not find such a tendency: the difficulty ranges of the testlets were comparable to random sets.

Furthermore, the maximum difficulty ranges within testlet condition were substantially large. For instance, it was 8.00 for *M* items of Group B, and this was only slightly lower than the difficulty range of entire *M* items (9.4 in Table 2). This analysis suggested that both easy and difficult items could be generated from the same listening passage, and only the passage difficulty itself may not be a strong predictor for the item difficulty. Accurate prediction of item difficulty requires a new set of features that capture the interaction between passages and questions.

Country	N	Item Type	Condition	Minimum	Maximum	Mean	standard deviation
Group A	250	*M*	testlet	0.10	6.80	2.26	1.17
			random	0.16	6.44	2.32	1.22
		C	testlet	0.10	6.60	2.00	1.09
			random	0.13	5.94	2.21	1.16
Group B	313	*M*	testlet	0.20	8.00	2.39	1.31
			random	0.12	7.51	2.46	1.29
		C	testlet	0.10	6.40	2.32	1.15
			random	0.13	6.72	2.47	1.27

Table 6: Comparison of difficulty ranges between testlets and random sets

5 Conclusion

Our analyses demonstrated that the generic features measuring item text complexity can be used to predict item difficulty. For simpler item types, the text complexity of the item text accounts for a larger share of variance than for more complex items. Our results also show that the accuracy of the generic system based on the text complexity features is equal to or better than the accuracy of human estimates. This result was consistent for a several types of items with different structure and across two different populations of test takers.

We also found that the most highly ranked features were related to item vocabulary, such as lexical frequency of the words as well as the level of concreteness. However, the system based on vocabulary features performed worse than the system based on all features. Finally, for sets of items which shared a common listening passage, the best performance was achieved by combining the text complexity of the printed questions and the recorded passage.

Acknowledgments

We thank Chi Lu and Diane Napolitano for their help with processing the data. We also thank Ikkyu Choi, Chong Min Lee, Keelan Evanini, Michael Flor and Susan Nissan for their comments and suggestions.

References

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2014. Predicting the difficulty of language proficiency tests. *Transactions of the Association for Computational Linguistics*, 2:517–529.

Dan Blanchard, Nitin Madnani, and Michael Heilman. 2016. Skll: Scikit-learn laboratory.

Eileen K Blau. 1990. The effect of syntax, speed, and pauses on listening comprehension. *TESOL quarterly*, 24(4):746–753.

Amber Bloomfield, Sarah C Wayland, Elizabeth Rhoades, Allison Blodgett, Jared Linck, and Steven Ross. 2010. What makes listening difficult? factors affecting second language listening comprehension. Technical report, DTIC Document.

R.F. Boldt and Roy Freedle. 1996. Using a neural net to predict item difficulty. *ETS Research Report*, RR-96-31.

Leo Breiman. 2001. Random forests. *Machine learning*, 45:5–32.

Geoff Brindley and Helen Slatyer. 2002. Exploring task difficulty in esl listening assessment. *Language Testing*, 19(4):369–394.

Jing Chen, Kathleen M Sheehan, Luke Harding, and Diane Schmitt. 2015. Analyzing and Comparing Reading Stimulus Materials Across the TOEFL Family of Assessments. *ETS Research Report*, RR-15-08.

Roy Freedle and Irene Kostin. 1996. The prediction of TOEFL listening comprehension item difficulty for minitalk passages: Implications for construct validity. *ETS Research Reports*, RR-96-29.

Paul W. Holland and Dorothy T. Thayer. 1985. An alternate definition of the ETS delta scale of item difficulty. program statistics research. *ETS Research Reports*, ETS-RR-85-43.

Ayako Hoshino and Hiroshi Nakagawa. 2010. Predicting the Difficulty of Multiple-Choice Close Questions for Computer-Adaptive Testing. *Special issue: Natural Language Processing and its Applications*, page 279.

Katsunori Kotani, Shota Ueda, Takehiko Yoshimi, and Hiroaki Nanjo. 2014. A listenability measuring method for an adaptive computer-assisted language learning and teaching system. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computation*, pages 387–394.

Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse Processes*, 25(2-3):259–284.

Diane Napolitano, Kathleen M Sheehan, and Robert Mundkowsky. 2015. Online Readability and Text Complexity Analysis with TextEvaluator. In *Proceedings of NAACL-HLT 2015, Denver, Colorado, May 31 - June 5, 2015*, pages 96–100.

Susan Nissan, Felicia Devincenzi, and Linda K. Tang. 1995. An analysis of factors affecting the difficulty of dialogue items in TOEFL listening comprehension. *ETS Research Report*, RR-95-37.

Andre A. Rupp, Paula Garcia, and Joan Jamieson. 2001. Combining multiple regression and CART to understand difficulty in second language reading and listening comprehension test items. *International Journal of Testing*, 1(3-4):185–216.

Kathleen M Sheehan, Michael Flor, and Diane Napolitano. 2013. A two-stage approach for generating unbiased estimates of text complexity. In *Proceedings of the Workshop on Natural Language Processing for Improving Textual Accessibility*, pages 49–58.

Kathleen M. Sheehan, Irene Kostin, Diane Napolitano, and Michael Flor. 2014. The TextEvaluator tool: Helping teachers and test developers select texts for use in instruction and assessment. *The Elementary School Journal*, 115(2):184–209.

Yuni Susanti, Hitoshi Nishikawa, Takenobu Tokunaga, and Obari Hiroyuki. 2016. Item Difficulty Analysis of English Vocabulary Questions. In *Proceedings of the 8th International Conference on Computer Supported Education*, pages 267–274.

Su-Youn Yoon, Yeonsuk Cho, and Diane Napolitano. 2016. Spoken text difficulty estimation using linguistic features. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 267–276.

The Construction of a Chinese Collocational Knowledge Resource and Its Application for Second Language Acquisition

Renfen Hu
Institute of Chinese Information Processing, Beijing Normal University
19 Xinjiekouwai Street, Beijing, China
irishere@mail.bnu.edu.cn

Jiayong Chen and Kuang-hua Chen
Department of Library and Information Science, National Taiwan University
No.1, Sec. 4, Roosevelt Road, Taipei, Taiwan
{d04126001,khchen}@ntu.edu.tw

Abstract

The appropriate use of collocations is a challenge for second language acquisition. However, high quality and easily accessible Chinese collocation resources are not available for both teachers and students. This paper presents the design and construction of a large scale resource of Chinese collocational knowledge, and a web-based application (OCCA, Online Chinese Collocation Assistant) which offers free and convenient collocation search service to end users. We define and classify collocations based on practical language acquisition needs and utilize a syntax based method to extract nine types of collocations. Totally 37 extraction rules are compiled with word, POS and dependency relation features, 1,750,000 collocations are extracted from a corpus for L2 learning and complementary Wikipedia data, and OCCA is implemented based on these extracted collocations. By comparing OCCA with two traditional collocation dictionaries, we find OCCA has higher entry coverage and collocation quantity, and our method achieves quite low error rate at less than 5%. We also discuss how to apply collocational knowledge to grammatical error detection and demonstrate comparable performance to the best results in 2015 NLP-TEA CGED shared task. The preliminary experiment shows that the collocation knowledge is helpful in detecting all the four types of grammatical errors.

1 Introduction

Second language (L2) learners often have problems in appropriate use of word collocations (Bahns and Eldaw,1993; Farghal and Obiedat, 1995; Nesselhauf, 2003). For example, English speakers who are learning Chinese are likely to produce incorrect expressions e.g. 漂亮 的 歌 (piaoliang/beautiful de/* ge/song) because "beautiful song" is a reasonable collocation in English. However, 漂亮 (piaoliang/beautiful) is not an acceptable modifier for 歌 (ge/song) in Chinese (Lu, 1987). Bahns and Eldaw (1993) went deeply into this issue and suggested that leaner's collocational competence does not develop in parallel with general vocabulary knowledge, because words are usually acquired individually without taking note of their immediate environment (Siyanova and Schmitt, 2008).

This problem could be more serious for learning analytic languages such as Chinese, because apart from the negative transfer from the first language (L1), this typical analytic language does not have inflectional morphemes, and uses function words with little lexical meaning to express grammatical relationships. These function words are also highly involved in collocations. The following shows collocation examples with function words:

(a) 在 (zai/*) X 上 (shang/on) → on X
(b) 诚实 (chengshi/honest) 的(de/*) 人 (ren/person) → honest person
(c1) 买 (mai/buy) 东西 (dongxi/something) → buy something
(c2) 买 (mai/buy) 了(le/*) 东西 (dongxi/something) → have bought something
(c3) 买 (mai/buy) 着 (zhe/*) 东西 (dongxi/something) → be buying something

Example (a) is a two-word collocation in which the preposition 在 (zai) collocates with the postposition 上 (shang). The auxiliary word 的 (de) in Example (b) connects an attributive and a head noun. By comparing (c1), (c2) and (c3), the auxiliary words 了 (le) and 着 (zhe) are used to indicate perfective and progressive aspects. These examples illustrate at least three features of Chinese function words: (1) directly constituting specific collocations, e.g. a preposition and a postposition; (2) acting

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3254–3263, Osaka, Japan, December 11-17 2016.

as an auxiliary structural particle in a collocation, e.g. 的 (de); (3) indicating the aspect, e.g. 了 (le) and 着 (zhe). Therefore, teaching and learning Chinese collocations become a considerable challenge.

In order to offer informative reference of collocational knowledge to L2 teachers and learners, this paper discusses the construction of a Chinese collocational knowledge resource and its application for second language acquisition. We define nine types of collocations reflecting Chinese specific grammatical features, and automatically extract collocations from two source corpora: CTC[1], a Chinese text corpus for L2 learners (Yang and Xiao, 2015) and Simplified Chinese Wikipedia Corpus[2]. Sentences in the corpora are firstly processed by LTP-Cloud (Che et al., 2010), a Chinese NLP toolkit to do word segmentation, POS tagging and dependency parsing. Then we compile 37 rules based on word, POS and dependency relation features to extract the target word combinations. After building a collocation database containing over 1,750,000 collocations and the corresponding attributes, we design and construct the OCCA (Online Chinese Collocation Assistant) to offer free and convenient collocation search service to users. To examine the coverage, accuracy and efficiency of our collocation data, we compare it with two traditional collocation dictionaries and apply it to the grammatical error detection task. The experiment shows desirable results from both the comparison and the application.

The rest of the paper is structured as follows. In section 2 we review existing Chinese collocation resources and the automatic extraction methods. Section 3 discusses our definition and operational types for Chinese collocations. Section 4 describes the approach of extracting collocations from text corpora and building the Online Chinese Collocation Assistant. The comparisons and applications of our approach are discussed in Section 5 and 6. We draw brief conclusions in the last section based on the experimental results.

2 Related Work

The definition of collocation varies in previous studies, but most of them emphasize the importance of statistical frequency of word combinations (Firth, 1957; Church, 1990). Benson et al. (1986) defined collocation as "an arbitrary and recurrent word combination", and classified collocations into a lexical group and a grammatical group. Xu et al. (2009) gave a more specific definition: "a collocation is a recurrent and conventional expression containing two or more content word combinations that hold syntactic and/or semantic relations." This definition emphasizes the syntactic and semantic relations between words but excludes the function words.

Existing resources of collocational knowledge mainly fall into two types, i.e. collocation dictionary and collocation bank. Manually compiled dictionaries may have problems in coverage and consistency, and it is quite difficult to add new entries or collocations (Smadja, 1993). Besides, a list of typical collocations without contexts is not sufficient to support language learning. Xu et al. (2009) built Chinese Collocation Bank (CCB) with true collocations annotated in a large-scale news corpus. It is a valuable resource for collocation related research and NLP tasks, but might not be appropriate for language acquisition for two reasons. Firstly, collocations are domain dependent (Smadja, 1993). CCB annotated collocations from the People's Daily corpus (Yu et al., 2000), which consists of news articles of People's Daily, an official newspaper of the Chinese Communist Party. Frequent collocations in this corpus are mostly used in formal texts, and highly related to politics and economy. They might not be suitable to second language learning. Secondly, CCB defined collocations as content word combinations and did not deal with function words that are one of the most difficult parts for L2 learners.

Most previous studies used window-based methods to extract word combinations in a fixed window based on word co-occurrences or distribution scores (Church, 1990; Smadja, 1993; Sun et al., 1997; Xu and Lu, 2006). Obviously, these methods do not target at collocations with syntactic and/or semantic relations. Kilgarriff et al. (2004) used regular expressions over POS-tags to formalize rules of collocation patterns when building Word Sketch Engine. Huang et al. (2005) extended Sketch Engine to Chinese and found POS based rules were efficient in extracting grammatical information. However, they also addressed that the regular expression patterns faced challenges in long-distance collocation extraction and considerable risk of mis-classification, which is more serious in Chinese than in English.

[1] CTC (Corpus of Teaching Chinese as Second Language) contains text data from over 100 classic Chinese textbooks for L2 learning and New HSK (Chinese language proficiency Test) papers since 2009: http://www.aihanyu.org/basic.aspx.
[2] Chinese Wikipedia download: https://dumps.wikimedia.org/zhwiki/

Thus a better strategy should incorporate richer annotation of corpus and take more consideration of Chinese grammatical features.

3 The Proposed Approach

In contrast to definitions of collocation in the previous studies, We introduce function words into our collocation study, and identify four characteristics of collocations: (1) can be word combinations with more than two words; (2) can contain both content words and function words; (3) collocated words can be either adjacent or non-adjacent; (4) collocated words must hold syntactic or semantic relations.

We propose a syntax-based method based on dependency parsing and extract collocations from CTC, a text corpus for L2 learners to meet the domain needs. After preliminary studies, we find Chinese Wikipedia as a good supplement source because it has much bigger corpus size than CTC and the Wikipedia (WIKI hereafter) texts are from various domains. The collocations extracted from WIKI could serve as an important complement to CTC from both the coverage and the domain perspectives. In addition, complicated sentences in WIKI could be good resources for advanced-level learners.

In order to extract collocations based on the dependency trees efficiently and effectively, we define nine types of grammatical collocations as the extraction targets. As shown in Table 1, four of them have universal syntactic relations, while the other five types are Chinese unique collocations with specific syntactic or semantic relations that should be emphasized in second language acquisition. Function words are highly involved in all the types, and in addition to the most common two-word collocations, six types include three-word or four-word collocations.

Types	Language Independent	Function Words	No. of Words
Verb-Object (VO)	Y	direction verb, particle	2, 3, 4
Subject-Predicate (SP)	Y	direction verb, particle	2, 3, 4
Adjective-Noun (AN)	Y	particle	2, 3
Adverb-Predicate (AP)	Y	particle	2, 3
Classifier-Noun (CN)	N	classifier	2
Preposition-Postposition (PP)	N	preposition, direction noun	2
Preposition-Verb (PV)	N	prepostion, direction verb, particle	2, 3, 4
Predicate-Complement (PC)	N	direction verb, particle	2, 3, 4
Connective-Connective (CC)	N	conjunction, conjunctive adverb	2

Table 1. Collocation types

We will briefly introduce the five Chinese-dependent types in the following and the detailed descriptions for 26 forms of word combination based on the nine types are given in OCCA online user guide[3].

- **Classifier-Noun (CN):** Chinese classifier, also called measure word, is used to modify a noun after a number or quantifier. The translation of “a person" is 一个人 (yi/a ge/* ren/person), 个 (ge) is a measure word for 人 (ren/person). Most Chinese nouns require measure words based on their innate semantic and cognitive relations (Zhang, 2007; Her and Hsieh, 2010).
- **Preposition-Postposition (PP):** A preposition usually collocates with specific postpositions (also called direction nouns) to express spatial or temporal relations. They can constitute a prepositional phrase with a noun phrase in between, e.g. 在 桌子 上 (zai/* zhuozi/table shang/on) which means “on the table”.
- **Preposition-Verb (PV):** Both modern Chinese and English are S-V-O languages, however, word order in Chinese is often changed to S-P-O-V by some special prepositions to emphasize a part of the sentence, or to convey a nuance of the meaning (Hu et al., 2014). These prepositions directly introduce the objects of their collocated verbs, e.g. 把水喝了 (ba/* shui/water he/drink le/*) means “drink the water”. They only collocate with verbs that have certain semantic fea-

[3] OCCA user guide: http://occa.xingtanlu.cn/index.php?action=page&pid=11

tures, and most verbs must have one or two adjacent particles to indicate the aspect or direction (Wang, 1985; Lv, 1999; Zhang, 2001).

- **Predicate-Complement (PC):** Complement is a word, phrase or clause following the predicate (a verb or an adjective) to provide additional information, including result, direction, possibility, state, degree, quantity, duration, and location (Liu et al., 2001). To express these rich semantic features, complement has complicated internal structures. We summarize nine forms of Predicate-Complement collocation based on Liu et al. (2001)'s research.
- **Connective-Connective (CC):** Chinese conjunctions and adverbs can both serve as connectives between clauses to indicate various discourse relations, and these connectives are often used in a pair e.g. 因为 (yinwei/because) - 所以 (suoyi/so) for casual relation and 虽然 (suiran/although) – 但是 (danshi/but) for contrastive relation. We identify 45 connective pairs as word collocations based on Liu et al. (2001)'s research of conjunction and adverb usages.

After extracting collocations, we will construct a web-based collocation assistant (OCCA). In order to meet the practical requirements of learning and teaching which we have uncovered based on reviewing previous studies, the OCCA will provide users collocations with the following features: (1) given with context sentences; (2) defined and classified with full consideration of Chinese grammatical features, including syntactic structures and function words; (3) extracted from appropriate texts that are suitable for L2 learners in both domain and degree of difficulty; (4) updated easily with a change of corpus.

4 Extracting Collocations and Constructing OCCA

Figure 1 shows the steps in the process of extracting collocations and building OCCA. The details of each step will be explained as following.

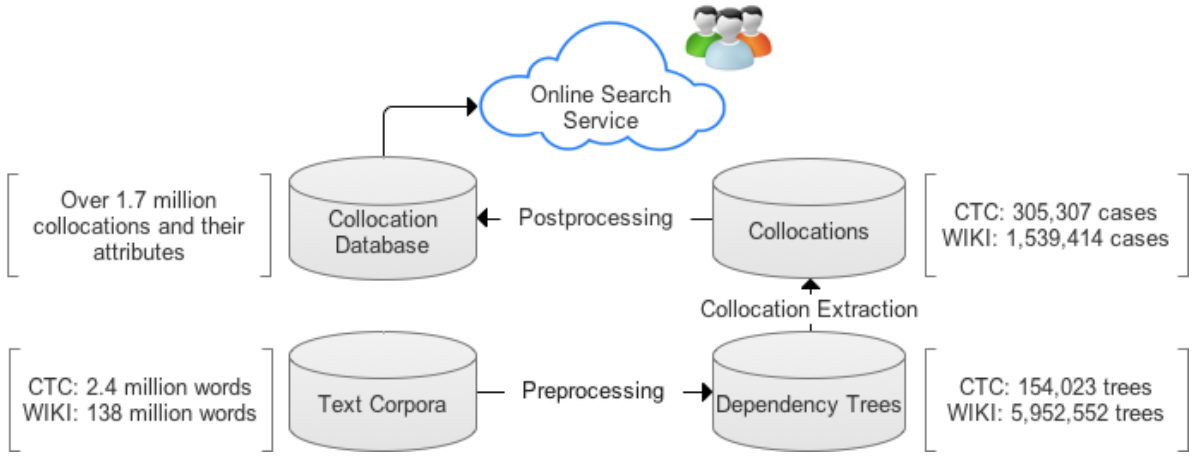


Figure 1. Procedure of our approach

```
<sent id="15213" cont="我们都笑了起来。">
  <word id="0" cont="我们" pos="r" ne="O" parent="2" relate="SBV"/>
  <word id="1" cont="都" pos="d" ne="O" parent="2" relate="ADV"/>
  <word id="2" cont="笑" pos="v" ne="O" parent="-1" relate="HED"/>
  <word id="3" cont="了" pos="u" ne="O" parent="2" relate="RAD"/>
  <word id="4" cont="起来" pos="v" ne="O" parent="2" relate="CMP"/>
  <word id="5" cont="。" pos="wp" ne="O" parent="2" relate="WP"/>
```

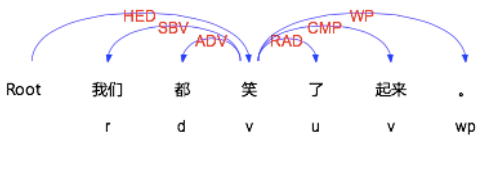


Figure 2. The pre-processing result of a Chinese sentence 我们都笑了起来 (wo'men/we dou/all xiao/laugh le/* qilai/*), which means "we all laugh"

4.1 Pre-processing

LTP-Cloud (Che et al., 2010), a Chinese NLP toolkit, is utilized to carry out word segmentation, POS tagging and dependency parsing for sentences in CTC and WIKI. We choose the latest released version (v3.3.1) of the toolkit to ensure the performance of each NLP module. We obtain over 6.1 million

dependency trees from the two corpora and the results are presented in XML files. Figure 2 shows an XML tree and its dependency graph of a Chinese sentence.

4.2 Collocation Extraction

Dependency tree is an ideal carrier of various grammatical features including dependency relations as well as POS tags that can be used in automatic collocation extraction. As shown in Table 2, we firstly build mappings between dependency relations and collocation types. These relations exist between headwords and their direct modifiers. As they are not in one-to-one mappings with collocation types, POS and word location features are utilized as constraints. In addition to the dependency relations in Table 2, we use the RAD (right adjunct) relation to identify structural and aspect particles that are necessary units in three-word and four-word collocations. Since collocated words in Connective-Connective (CC) collocations do not have direct dependency relations between each other, we only adopt word and POS features for extraction of this type. After analysing all the collocation forms, we manually compile 37 rules to extract the nine types of collocations from the dependency trees. Most rules deal with two-word collocations by extracting dependency triples {headword; modifier; dependency relation}, and also collocations that have more than two words and multiple relations.

Collocation Type	Dependency Relation	Collocation Type	Dependency Relation
VO	VOB (object of verb)	AP	ADV (adverbial)
	IOB (indirect object)	CN	ATT (attribute)
	FOB (fronting object)	PP	POB (preposition-object)
SP	SBV (subject of verb)	PV	ADV (adverbial)
AN	ATT (attribute)	PC	CMP (complement)

Table 2. Mappings between dependency relations and collocation types

Taking the sentence in Figure 2 as an example, three colocations 我们 笑 (wo'men/we xiao/laugh), 都 笑 (dou/all xiao/laugh) and 笑 了 起来 (xiao/laugh le/* qilai/*) will be extracted in this step. The following rule illustrates how the three-word Predicate-Complement collocation 笑了起来 is extracted based on the words' location, POS and dependency features:

if word[relate] == 'CMP' & word[parent] == word[id]-2 & sent[word[id]-2][pos] == v & sent[word[id]-1][relate] == 'RAD' & sent[word[id]-1][parent] == word[id]-2:

extract sent[word[id]-2][cont], sent[word[id]-1][cont], word[cont] as a three-word Predicate-Complement collocation.

4.3 Post-processing

In step 2, we extract 305,307 CTC collocations and 1,539,414 WIKI collocations. To refine the extraction results, we filter out 4,262 CTC and 84,893 WIKI collocations by seven filtering rules, e.g. exclude a collocation if there is a colon between the headword and its direct modifier in the sentence.

After that, we calculate the frequency of each collocation, and identify the relevance between two collocations if they have the same headword, direct modifier and dependency relation, e.g. examples (c1, c2, and c3) mentioned in Section 1. As the only difference is the structural particles, collocations like c2 and c3 are defined as variants of c1. The collocation relevance information could help users gain better understanding of grammatical and semantic roles of Chinese structural particles.

After post-processing, we obtain 1,755,566 collocations in the database. Unlike CCB (Chinese Collocation Bank) that only deals with content word combinations, our data covers a large number of collocations containing function words. Nearly 30% collocations are constituted by one or more function words including prepositions, conjunctions, direction verbs, direction nouns, structural and aspect particles. In addition to the most common two-word collocations, we also extract three-word and four-word collocations, which approximately account for 25% of the total number.

4.4 Building the Online Acquisition Services

We build OCCA based on Apache, PHP and MySQL. The website is http://occa.xingtanlu.cn/. Users can input a single keyword or multiple keywords to search for collocations from CTC (default) or WIKI database. The system outputs collocations with consideration of their relevance and ranks them based on frequency. Users can click a collocation for all the context sentences. The detailed user guide is available on OCCA website. Liu (2010) suggested that word collocations from a corpus can effec-

tively assist language teachers to summarize typical usage patterns and important lexical properties of words. As OCCA offers collocations with statistics and context sentences, it can serve as a good assistant in vocabulary teaching. Students might also use it to retrieve word usages they are uncertain about.

5 Comparison with two Collocation Dictionaries

We evaluate the coverage (collocation type and quantity) and accuracy of OCCA by comparing its searching results with two classic Chinese collocation dictionaries: Modern Chinese Collocation Dictionary (Mei, 1999) and Modern Chinese Content Word Collocation Dictionary (Lin and Zhang, 1992) hereafter referred as D1 and D2.

D1 and D2 collect collocations for over 6,000 and 8,000 entries. In OCCA the CTC and WIKI databases contain 7,632 and 47,475 keywords with occurrences greater than 10 respectively. Considering CTC is a more comparable resource with the dictionaries in vocabulary size, we only use WIKI collocation data as reference in the comparison. As we are building a collocation assistant for second language acquisition, 100 words with the highest error occurrences in the HSK Dynamic Composition Corpus[4] are used as our test words.

After searching for these 100 words in OCCA (the default CTC database), and looking up them in D1 and D2, we find that OCCA contains 98 of them, with higher coverage than D1 (78 words) and D2 (24 words)[5]. The reason for this is that students make mistakes involving function words more often than content words. However, the main lexical items in the dictionaries are content words only, e.g. nouns, verbs, and adjectives.

We also analyse the collocation data of 19 words in the intersection of the three resources. The comparison is conducted from three perspectives: collocation type coverage, collocation quantity and collocation accuracy. As shown in Table 3, for these 19 entries, D1 and D2 both show six types of collocations in our definition, omitting Preposition-Verb (PV) and Preposition-Postposition (PP) collocations that deal with the syntactic and semantic relations of prepositions. As D1 and D2 do not include conjunction entries, the Connective-Connective collocations are not involved in this comparison either. We count the collocations of the 19 entries in these resources and find OCCA is much higher than D1 and D2 in collocation quantities. However, by analysing the collocation data, we also find that OCCA does not contain some collocations in D1 and D2, e.g. "外交 (waijiao/diplomatic) 问题 (wenti/problems)", and "农村 (nongcun/village) 发展 (fazhan/develop)", mainly because of the domain and size limit of the CTC corpus. However, the WIKI database could serve as a good supplement because it covers nearly 63% of the 278 collocations that cannot be retrieved in CTC.

Resources	Types	Quantities	Accuracy
D1	6	622	~100.00%
D2	6	3,234	~100.00%
OCCA	8	9,705	95.19%

Table 3. Collocation data of 19 entries

Assuming D1 and D2 both have 100.00% precision in collocation data, we manually label the correctness of the 9,705 collocations retrieved from the CTC collocation data. 467 of them are annotated as inappropriate collocations, thus the accuracy is 95.19%. Among the 467 collocations, about 6% of them are due to tokenization errors in pre-processing, 87% of them result from parsing errors e.g. 把 X 产生 (ba/* X chansheng/ produce), and 7% of them are extracted from correct parsing results but cannot be taken as appropriate word collocations, e.g. "三 表示" (san/three biaoshi/means). We also find that over 86% of these mistakes occur only one time, and over 8% of them occur two times. Thus OCCA now only outputs collocations with occurrences no less than three times to reduce the negative effect. We will collect collocation errors reported by OCCA users and develop better filtering module based on these data in future work.

[4] HSK dynamic composition corpus is a Chinese leaner corpus released by Beijing Language and Culture University: http://202.112.195.192:8060/hsk/index.asp

[5] Test word list and the retrieval results: http://101.200.121.46/OCCA_test_words.pdf

6 Experiment on Grammatical Error Detection

Collocation data has been proved helpful in a list of Computer Aided Language Learning (CALL) tasks for English learners (Shei and Pain, 2000; Futagi et al., 2008; Wu et al., 2010). To examine the effectiveness of our Chinese collocation resource, we conduct a preliminary experiment on grammatical error detection task using the OCCA collocation data.

NLP technologies for Chinese grammatical error diagnosis (CGED) have received a considerable amount of attention in recent years (Yu et al., 2014; Lee et al., 2015). Various linguistic and computational resources including L1 corpora, L2 corpora and web N-gram data have been employed to identify the grammatical errors (Chang et al., 2012; Cheng et al., 2014; Lee et al., 2014). However, Chinese collocation resource is rarely used in previous studies.

We use the test set of 2015 NLP-TEA CGED shared task in the experiment. This data set consists of 1,000 Chinese sentences collected from the essay section of the computer-based Test of Chinese as a Foreign Language (TOCFL), administered in Taiwan (Lee et al., 2015). Half of these sentences are correct, while the other half include a single defined grammatical error: redundant (132), missing (126), selection (110), and disorder (132).

As the test sentences are in traditional Chinese, while our collocation data is simplified, we carry out the language conversion with the Open Chinese Convert tool[6]. After that, we extract 7,262 word combinations from the 1,000 sentences by taking the same pre-processing and collocation extraction methods described in Section 4.

By looking up each collocation in OCCA including both CTC and WIKI databases, 496 instances in 401 sentences are identified with zero occurrence. We assume these sentences contain improper word combinations and manually label these collocations to check if they are valid to indicate grammatical errors. Table 4 shows examples we consider the zero occurrence tokens are related with the errors.

Type	Examples	Correction
Redundant	可是现在我把什么事都不记得。 (Preposition-Verb: 把 记得)	可是现在我什么事都不记得。 But I can't remember anything now.
Missing	我们到现在都是以写信为连络。 (Verb-Object: 为 联络)	我们到现在都是以写信为连络方式。 We are still writing letters to each other.
Selection	排队了很多时间才轮到我。 (Predicate-Complement: 排队 很多 时间)	排队了很久才轮到我。 I queued for quite a while before my turn.
Disorder	他教书在那个很有名的大学。 (Predicate-Complement: 教书 在)	他在那个很有名的大学教书。 He is teaching at that famous university.

Table 4. Grammatical error examples with zero occurrence collocation

After manual analysis, we find the collocation data could successfully detect word errors in 212 sentences. Among them 41% of the errors are detected by non adjacent word combinations, e.g. 把- 记得 (ba/* jide/remember) in which 把 (ba/*) is a redundant word, and 29% of the errors are detected by 3-word or 4-word combinations, e.g. 排队 很多 时间 (paidui/queue henduo/much shijian/time) in which 很多 时间 is a word selection error.

We also calculate the sentence-level precision, recall and F1 score for four error types. Table 5 shows the test result and the performance of three teams that achieved the best precision (CYUT team run2), recall (NTOU team run1) and F1 score (NTOU team run2) in the error detection part of CGED shared task.

From the data in Table 5, we could see that the collocation data is a helpful resource in the detection of inadequate word combinations. Even though we adopt a very simple data retrieval method, it achieves higher Precision and F1 score than the average of the CGED teams. The precision of our method is very close to the best precision 0.7453 achieved by CYUT team, and we have higher recall and F1 score than they have. NTOU team (Lin and Chen, 2015) proposed to measure sentence likelihood scores with Chinese Web 5-gram data for error detection. Compared with the n-gram dataset that is widely used in grammatical error diagnosis, the most notable advantage of the collocation data is it

[6] Open Chinese Convert Project: https://github.com/BYVoid/OpenCC

captures not only adjacent words but also non-adjacent words with syntactic or semantic relations, thus it could be able to indicate errors that exist in a long distance, e.g. the preposition-verb example 把-记得(ba/* jide/remember) in Table 3.

Type	Precision	Recall	F1 score
Redundant (42)	0.8571	0.3182	0.4641
Missing (48)	0.8136	0.3810	0.5189
Selection (67)	0.7128	0.6091	0.6569
Disorder (55)	0.5392	0.4167	0.4701
Average	**0.7307**	**0.4312**	**0.5424**
CGED bestP Team (CYUT-run2)	**0.7453**	0.2400	0.3631
CGED bestR Team (NTOU-run1)	0.5000	**1.0000**	0.6667
CGED bestF1 Team (NTOU-run2)	0.5164	0.9760	**0.6754**
CGED average	**0.5600**	**0.6066**	**0.5327**

Table 5. Performance of grammatical error detection

It should also be noted that the recall of our method is lower than precision which means that not all the grammatical errors are related with word collocations, especially the word redundant and missing types. To understand the pros and cons of this collocation based method, we analyse the false positive (FP) and false negative (FN) cases and reach the following findings.

False positive cases are correct sentences that are identified with an error because they contain collocations that cannot be retrieved in our database. The most important reason is the word usage difference between traditional and simplified Chinese. Some common words in traditional Chinese e.g. 捷运(jieyun/subway) and 障碍者(zhang'aizhe/disabled person) have different representations e.g. 地铁(ditie/subway) and 残疾人(canjiren/disabled person) in Mandarin. There are also some word combinations incorrectly extracted because of the word segmentation and parsing errors.

False negative cases refer to the incorrect sentences whose errors are not detected by our method. By analysing these sentences, we find most of the errors are relevant with adverb and auxiliary usages. With an adverb or auxiliary missing or using incorrectly, word combinations in the sentence could still be grammatically correct but the whole sentence might not meet the strict requirement of word order or semantic pattern. The following sentence is an example with a word missing error that the collocation based method could not identify. In this sentence an adverb 都 (all) should be used to modify the adjective 差不多 (similar):

Sentence: 我们发现了每个国家人民*差不多。We find people in different countries are similar.

Correction: 我们发现了每个国家人民都差不多。We find people in different countries are all similar.

From the above analysis, we could see that the collocation based method could be effective in detecting a fairy proportion of grammatical errors, especially for those involved non adjacent words. However, the collocation data alone is far from enough, thus we need to combine the dataset with other language resources e.g. the Web 5-gram data and statistical models to explore better strategies.

7 Conclusions

This paper discusses the design and construction of Online Chinese Collocation Assistant (OCCA) for second language teaching and learning. We identify the important roles of function words in collocational knowledge and develop automatic extraction method to extract nine types of Chinese collocations from two text corpora. By searching for keywords in OCCA, users can easily obtain collocations with their statistics and context sentences that are helpful in the vocabulary teaching and learning. We compare the collocation data of OCCA with two traditional dictionaries, and find OCCA has much higher entry coverage and collocation quantity. It also has quite low collocation error rate at less than 5%. In order to investigate the helpfulness of OCCA, we conduct a preliminary experiment to apply the collocation resource to Chinese grammatical error detection. By implementing the simple data retrieval method, OCCA collocations are effective in detecting four types of grammatical errors and demonstrate comparable performance with comparison to the best results in 2015 NLP-TEA CGED shared task.

To minimize the negative impacts of mistakes and to offer much more reliable collocation resources, we will develop more efficient and effective methods for filtering and correcting collocations in the

future. We also hope to conduct further study to verify the effectiveness of OCCA by combining the collocation data with other language resources and methods in CALL tasks.

Acknowledgments

This work was partially supported by the National High Technology Research and Development Program of China (No. 2012AA011104) and National Language Committee Research Program of China (No. YB125-124). This work was also partially supported by Ministry of Science and Technology, Taiwan under the grant number MOST-105-2410-H-002-118. We thank the anonymous reviewers for their insightful comments.

References

Jens Bahns and Moira Eldaw. 1993. Should we teach EFL students collocations?. System, 21(1): 101-114.

Morton Benson, Evelyn Benson and Robert F. Ilson. 1986. The BBI combinatory dictionary of English: A guide to word combinations. John Benjamins, Amsterdam and Philadelphia.

Ru-Yng Chang, Chung-Hsien Wu and Philips Kokoh Prasetyo. 2012. Error diagnosis of Chinese sentences using inductive learning algorithm and decomposition-based testing mechanism. ACM Transactions on Asian Language Information Processing, 11(1): Article 3.

Wanxiang Che, Zhenghua Li and Ting Liu. 2010. Ltp: A chinese language technology platform. In Proceedings of COLING (System Demonstrations), pages 13-16.

Shuk-Man Cheng, Chi-Hsin Yu and Hsin-Hsi Chen. 2014. Chinese Word Ordering Errors Detection and Correction for Non-Native Chinese Language Learners. In Proceedings of COLING, pages 279-289.

Ward K. Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. Computational linguistics, 16(1): 22-29.

Mohammed Farghal and Hussein Obiedat. 1995. Collocations: A neglected variable in EFL. IRAL-International Review of Applied Linguistics in Language Teaching, 33(4): 315-332.

John R. Firth. 1957. Papers in Linguistics, Oxford University Press, London, UK.

Yoko Futagi , Paul Deane , Martin Chodorow and Joel Tetreault. 2008. A computational approach to detecting collocation errors in the writing of non-native speakers of English. Computer Assisted Language Learning, 21(4):353-367.

One-Soon Her and Chen-Tien Hsieh. 2010. On the semantic distinction between classifiers and measure words in Chinese. Language and linguistics, 11(3): 527-551.

Renfen Hu, Zhiying Liu, Lijiao Yang and Yaohong Jin. 2014. Pre-reordering Model of Chinese Special Sentences for Patent Machine Translation. In Proceedings of COLING Workshop on Synchronic and Diachronic Approaches to Analyzing Technical Language, pages 40-47.

Chu-Ren Huang, Adam Kilgarriff, Yiching Wu, Chih-Ming Chiu, Simon Smith, Pavel Rychly, Ming-Hong Bai and Keh-Jiann Chen. 2005. Chinese Sketch Engine and the extraction of grammatical collocations. In Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing, pages 48-55.

Adam Kilgarriff, Pavel Rychly, Pavel Smrz and David Tugwell. The sketch engine. 2004. In Proceedings of Euralex, pages 105-116.

Lung-Hao Lee, Liang-Chih Yu and Li-Ping Chang. 2015. Overview of the NLP-TEA 2015 Shared Task for Chinese Grammatical Error Diagnosis. In Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications, pages 1–6.

Lung-Hao Lee, Liang-Chih Yu, Kuei-Ching Lee, Yuen-Hsien Tseng, Li-Ping Chang and Hsin-Hsi Chen. 2014. A Sentence Judgment System for Grammatical Error Detection. In Proceedings of COLING (System Demonstrations), pages 67-70.

Chuan-Jie Lin and Shao-Heng Chen. 2015. NTOU Chinese Grammar Checker for CGED Shared Task. In Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications, pages 15–19.

Dekang Lin. 1998. Extracting collocations from text corpora. In Proceedings of first workshop on computational terminology, pages 57-63.

Xingguang Lin and Shoukang Zhang. 1992. Dictionary of modern Chinese content word collocations. Commercial Press, Beijing, China.

Fengqin Liu. 2010. Corpus based collocation research and vocabulary teaching for second language learners. Modern Chinese, (6):115-117.

Yuehua Liu, Wenyu Pan and Weihua Gu. 2001. Pratical grammar of modern Chinese. Commercial Press, Beijing, China.

Jianji Lu. 1987. Vocabulary error analysis of Chinese second language leaners. Language Teaching and Research, (4): 122-132.

Shuxiang Lv. 1999. Eight hundred words in modern Chinese. Commercial Press, Beijing, China.

Jiaju Mei. 1999. Dictionary of modern Chinese collocations. Chinese Dictionary Press, Shanghai, China.

Nadja Nesselhauf. 2003. The use of collocations by advanced learners of English and some implications for teaching. Applied linguistics, 24(2): 223-242.

C.-C. Shei and Helen Pain. 2000. An ESL writer's collocational aid. Computer Assisted Language Learning, 13(2): 167-182.

Anna Siyanova and Norbert Schmitt. 2008. L2 learner production and processing of collocation: A multi-study perspective. Canadian Modern Language Review, 64(3): 429-458.

Frank Smadja. 1993. Retrieving collocations from text: Xtract. Computational linguistics, 19(1): 143-177.

Maosong Sun, Changning Huang and Jie Fang. 1997. Quantitative analysis of Chinese collocations. Studies of the Chinese Language, 256(1): 29-38.

Li Wang. 1985. Modern Chinese grammar. Commercial Press, Beijing, China.

Brent Wolter. 2006. Lexical network structures and L2 vocabulary acquisition: The role of L1 lexical/conceptual knowledge. Applied Linguistics, 27(4): 741-747.

Jian-Cheng Wu, Yu-Chia Chang, Teruko Mitamura and Jason S. Chang. 2010. Automatic collocation suggestion in academic writing. In Proceedings of the ACL, pages 115-119.

Ruifeng Xu and Qin Lu. 2006. A multi-stage chinese collocation extraction system. In Advances in Machine Learning and Cybernetics, pages 740-749.

Ruifeng Xu, Qin Lu, Kam-Fai Wong and Wenjie Li. 2009. Building a chinese collocation bank. International Journal of Computer Processing of Languages, 22(01): 21-47.

Lijiao Yang and Hang Xiao. 2015. Contextual information labeling in a corpus for second language learning. Applied Linguistics, (1):107-116.

Liang-Chih Yu, Lung-Hao Lee and Li-Ping Chang. 2014. Overview of grammatical error diagnosis for learning Chinese as a foreign language. 2014. In Proceedings of the 1st Workshop on Natural Language Processing Techniques for Educational Applications, pages 42-47.

Shiwen Yu, Xuefeng Zhu and Huiming Duan. 2000. Guideline of Large-scale Modern Chinese Corpus Annotation. Journal of Chinese Information Processing, 14(6): 58-64.

Hong Zhang. 2007. Numeral classifiers in Mandarin Chinese. Journal of East Asian Linguistics, 16(1): 43-59.

Wangxi Zhang. 2001. The displacement pattern of BA Sentence. Language teaching and research, (3): 1-10.

Joint Inference for Event Coreference Resolution

Jing Lu[1] and **Deepak Venugopal**[2] and **Vibhav Gogate**[1] and **Vincent Ng**[1]
[1]Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75080
[2]Department of Computer Science, The University of Memphis, Memphis, TN 38152
jxl125430@utdallas.edu, dvngopal@memphis.edu,
{vgogate,vince}@hlt.utdallas.edu

Abstract

Event coreference resolution is a challenging problem since it relies on several components of the information extraction pipeline that typically yield noisy outputs. We hypothesize that exploiting the inter-dependencies between these components can significantly improve the performance of an event coreference resolver, and subsequently propose a novel joint inference based event coreference resolver using Markov Logic Networks (MLNs). However, the rich features that are important for this task are typically very hard to explicitly encode as MLN formulas since they significantly increase the size of the MLN, thereby making joint inference and learning infeasible. To address this problem, we propose a novel solution where we implicitly encode rich features into our model by augmenting the MLN distribution with low dimensional unit clauses. Our approach achieves state-of-the-art results on two standard evaluation corpora.

1 Introduction

Within-document event coreference resolution is the task of determining which event mentions in a text refer to the same real-world event. Event coreference is arguably more challenging and less studied than entity coreference. The challenge stems in part from the fact that an event coreference resolver typically lies towards the end of the standard information extraction (IE) pipeline, assuming as input the noisy outputs of its upstream components. Specifically, a standard event coreference resolver takes as input the extracted event triggers, their arguments, and the entity coreference information, and aggregates this information through rules to resolve coreferent event mentions. Each component of this pipeline can introduce errors that naturally propagate to the event coreference resolver, thereby significantly affecting its performance. Further, the aforementioned pipeline architecture also fails to exploit inter-dependencies between the various components that can provide valuable insights to the resolver.

In light of these weaknesses, we propose a novel approach to within-document event coreference resolution based on Markov Logic Networks (MLNs) (Domingos and Lowd, 2009). In our approach, we jointly perform four key tasks in the IE pipeline: trigger identification and subtyping, argument identification and role determination, entity coreference resolution, and event coreference resolution. To our knowledge, this is the first attempt to design an MLN for event coreference resolution. MLNs are particularly well-suited for modeling *joint inference* tasks in natural language processing (NLP) due to the inherent relational structure and uncertainty typically associated with challenging NLP problems.

A major obstacle to the successful application of MLNs to NLP tasks is the high computational complexity of probabilistic inference and learning algorithms. The MLNs used in NLP are so large that even linear time inference algorithms are computationally infeasible. For instance, the rich sets of features that are typically used to solve the four tasks in the IE pipeline for event coreference, are ill-suited for modeling as explicit MLN formulas, since they will yield a large MLN having millions of features. Therefore, a major contribution of our work lies in the proposal of a novel hybrid approach where we embed such features as weighted unit clauses in a low-dimensional space, and then integrate these clauses with the rest of the MLN formulas during inference. Since this idea is generally applicable to modeling NLP tasks using MLNs, we believe that our work will be of interest to other NLP researchers as well.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3264–3275, Osaka, Japan, December 11-17 2016.

Georges Cipriani$_{[Person]}$, a former militant of the French far-left group Action Directe, {left}$_{ev1}$ the prison$_{[Origin]}$ in Ensisheim in northern France on parole on Wednesday$_{[Time]}$. He$_{[Person]}$ {left}$_{ev2}$ Ensisheim$_{[Origin]}$ in a police vehicle$_{[Instrument]}$ bound for an open prison near Strasbourg.

Table 1: Event coreference resolution example.

We evaluate our approach on corpora involving two languages, the new KBP 2015 English corpus and the Chinese portion of the ACE 2005 corpus. On both corpora, our approach performs significantly better than the baseline pipeline-based resolver. In particular, on the KBP corpus, we achieve the best result reported to date surpassing the previous best result by around 0.43 percentage points in average F1-score.

2 Definitions and Corpora

2.1 Definitions

We employ the following definitions in our discussion of event extraction and coreference:

- An **event mention** is an explicit occurrence of an event consisting of a textual trigger, arguments or participants (if any), and the event type/subtype.
- An **event trigger** is a string of text that most clearly expresses the occurrence of event, usually a word or a multi-word phrase
- An **event argument** is an argument filler that plays a certain role in an event.
- An **event coreference chain** (a.k.a. an **event hopper**) is a group of event mentions that refer to the same real-world event. They must have the same event (sub)type.

To understand these definitions, consider first the example shown in Table 1, which contains two event mentions, $ev1$ and $ev2$. Here, *left* is the trigger for both $ev1$ and $ev2$ with subtype Movement.Transport-Person. $ev1$ has three arguments, *Georges Cipriani*, *prison*, and *Wednesday* with roles *Person*, *Origin*, and *Time* respectively. $ev2$ also has three arguments, *He* and *Ensisheim*, and *police vehicle* with roles *Person*, *Origin*, and *Instrument* respectively.

2.2 Corpora

We employ two text corpora in two languages for evaluation. The *English* corpus was used in the Event Nugget Detection and Coreference task in the TAC KBP 2015 Event Track (henceforth the KBP 2015 corpus). This corpus is composed of two types of documents, newswire documents and discussion forum documents. The training set consists of 158 documents with 6538 event mentions distributed over 3335 event coreference chains, and the test set consists of 202 documents with 6438 event mentions distributed over 4125 event coreference chains. The *Chinese* corpus is the Chinese portion of the ACE 2005 training corpus. This corpus is composed of documents taken from six sources, and consists of 633 documents with 3333 event mentions distributed over 2521 event coreference chains. Note that ACE and KBP employ slightly different event ontologies: ACE defines 33 event subtypes and KBP defines 38 event subtypes, among which 31 subtypes are shared by both ontologies.

2.3 Key Differences between ACE and KBP

While both ACE and KBP rely on the aforementioned definitions, the guidelines they employ when annotating triggers and event coreference chains are slightly different. Below we highlight the differences that are relevant to our discussion.[1]

First, there are slight differences w.r.t. the annotation of triggers. ACE only allows single-word triggers, whereas KBP additionally allows multi-word triggers (e.g., *laid off*). Also, each word in ACE may trigger at most one event mention, whereas each (multi-)word in KBP may trigger multiple event mentions (e.g., *murder* can trigger two event mentions with subtypes Life.Die and Conflict.Attack).

[1]For detailed definitions, see http://cairo.lti.cs.cmu.edu/kbp/2015/event/annotation and http://www.itl.nist.gov/iad/mig/tests/ace/2005/ for the definitions of event coreference adopted by KBP 2015 and ACE 2005 respectively.

Second, KBP adopts a more relaxed definition of event coreference than ACE. Specifically, KBP requires that two event mentions be coreferent as long as they *intuitively* refer to the same real-world event. In our running example, $ev1$ and $ev2$ are coreferent according to KBP because they both refer to the same event of Cipriani leaving the prison. ACE, on the other hand, *additionally* requires that the corresponding *arguments* in the two event mentions be coreferent. In the example, $ev1$ and $ev2$ are *not* coreferent according to ACE because their *Origin* arguments are not coreferent (one Origin argument involves a prison and the other involves the city Ensisheim). Note that determining whether two entity mentions are coreferent is the task of entity coreference. Like event mentions, entity mentions have corpus-specific *entity types*.

3 Background

In this section, we give a brief overview of MLNs and discuss related work in event coreference resolution.

3.1 Markov Logic Networks

Formally, an MLN $\mathcal{M}$ is a set of pairs (f_i, θ_i) where f_i is a formula in first-order logic and θ_i is a real number. Given a set of constants, an MLN represents a ground Markov network, in which we have one binary random variable for each possible ground atom and one propositional feature for each possible grounding of each first-order formula. The weight associated with the feature is the weight attached to the corresponding formula. The ground Markov network represents the following probability distribution:

$$P_{\mathcal{M}}(\omega) = \frac{1}{Z} \exp\left(\sum_{f_i} \theta_i N_{f_i}(\omega)\right) \quad (1)$$

where $N_{f_i}(\omega)$ is the number of groundings of f_i that evaluate to True given a world ω (an assignment of $\{0, 1\}$ to all ground atoms). The use of first-order logic enables the user to succinctly represent prior, relational knowledge about the application domain, while the weights help model uncertainty in the truth of the first-order logic sentences.

3.2 Related Work

Existing within-document English event coreference resolvers have been evaluated on different corpora, such as MUC (e.g., Humphreys et al. (1997)), ACE (e.g., Ahn (2006), McConky et al. (2012), Chen and Ji (2009), S. and Arock (2012)), OntoNotes (e.g., Chen et al. (2011)) the (not publicly-available) Intelligence Community (IC) corpus (e.g., Cybulska and Vossen (2012), Araki et al. (2014)); the ECB corpus (e.g., Bejan and Harabagiu (2010; 2014), Lee et al. (2012)) and its extension, ECB+ (e.g.,Yang et al. (2015)); and ProcessBank (e.g., Araki and Mitamura (2015)). The newest event coreference corpus is the one used in the KBP 2015 Event Nugget Detection and Coreference shared task, in which the best performers are RPI's system (Hong et al., 2015), LCC's system (Monahan et al., 2015), and UI-CCG's system (Sammons et al., 2015). Among these corpora, ACE is the only one that is additionally composed of event coreference-annotated Chinese documents. It has been used to train SinoCoreferencer (Chen and Ng, 2014), a publicly-available Chinese event coreference resolver. Not all such corpora were carefully annotated: as Liu et al. (2014) pointed out, OntoNotes and ECB have only been partially annotated with event coreference links, for instance.

4 Baseline System

Our pipeline-based baseline system has five steps:

Step 1: Entity extraction. Our entity extraction model jointly identifies the entity mentions and their entity types. We train this model using CRF++[2], treating each sentence as a word sequence. Specifically, we create one instance for each word w and assign it a class label that indicates whether it begins an entity mention with type t_j (B-t_j), is inside an entity mention with type t_j (I-t_j), or is outside an entity mention

[2] https://taku910.github.io/crfpp/

(O). The features used to represent each instance for training the English CRF and the Chinese CRF are shown in Tables 2(a) and 3(a), respectively.

Step 2: Entity coreference resolution. Our entity coreference classifier is a pairwise classifier that determines whether two entity mentions are coreferent or not. To train this classifier, we employ SVM^{light} (Joachims, 1999), creating training instances using Soon et al.'s (2001) training instance creation method. Each training instance represents two entity mentions in each training document. The class value of a training instance is either positive or negative, depending on whether the two entity mentions are coreferent in the associated text. The features used to represent each instance for training the entity coreference classifiers for English and Chinese are shown in Tables 2(b) and 3(b), respectively.

After training, the resulting classifier can be used to classify each pair of entity mentions extracted in Step 1 as coreferent or not. We select as the antecedent of an entity mention em the closest preceding mention that is classified as coreferent with em.

Step 3: Trigger identification and subtyping. Since ACE allows only single-word triggers, our SVM-based Chinese trigger classifier takes as input a candidate trigger *word* (i.e., a word that survives Li et al.'s (2012) filtering rules) and outputs its event subtype (if it is a true trigger) or *None* (if it is not a trigger). In essence, it jointly (1) identifies event trigger words and (2) assigns a subtype to each identified trigger. To train this classifier, we create one training instance for each word w_i in each training document. If the word does not correspond to a trigger, the class label of the corresponding instance is *None*. Otherwise, the class label is the subtype of the trigger. The features used to represent each instance for training this classifier are shown in Table 3(c).

Because KBP additionally allows multi-word triggers, we recast the task of identifying English triggers as a sequence labeling task, where we train models using CRF++. Recall that since each (multi-)word may trigger multiple event mentions having different (sub)types, we train one CRF for each type. Specifically, to train the CRF for type t_j, we create one instance for each word w_i, assigning it a class label that indicates whether it begins a trigger with subtype s_{jk} (B-s_{jk}), is inside a trigger with subtype s_{jk} (I-s_{jk}), begins a trigger with other types (B-$\mathrm{t}_{m\neq j}$), is inside a trigger with other types (I-$\mathrm{t}_{m\neq j}$) or is outside a trigger (O). The features used to represent each instance for training this CRF are shown in Table 2(c). To improve the recall of event trigger detection, we augment the CRF output with heuristically extracted triggers. Specifically, we first construct a wordlist containing triggers that appear infrequently (less than 10 times) in the training data and do not belong more than one subtype according to the training data. Then, we extract any word as a trigger with the corresponding subtype as long as it appears in the wordlist.

Step 4: Argument identification and role labeling. Our argument identifier and role labeler is a classifier trained using SVM^{light} that jointly learns the tasks of (1) identifying the true arguments of an event mention and (2) assigning a role to each of its true arguments. To train this classifier, we create the training instances by pairing each true event mention em (i.e., event mention consisting of a true trigger) with each of em's candidate event arguments, considering an entity mention extracted in Step 1 a candidate argument of em if it appears in the same sentence as em. If the candidate argument is indeed a true argument of em, the class label of the training instance is the argument's role. Otherwise, its class label is *None*. The features used to represent each instance for training the English classifier and the Chinese classifier are shown in Tables 2(d) and 3(d), respectively.

After training, we can apply this classifier to classify test instances. To create test instances, we pair each *candidate* trigger (extracted in Step 3) with each of its candidate event arguments.

Step 5: Event coreference resolution. The event coreference classifier is a pairwise classifier that determines whether two event mentions are coreferent. To train this classifier, we use SVM^{light}, creating training instances using Soon et al.'s (2001) training instance creation method. The features used to represent each instance for training the event coreference classifier for English and Chinese are shown in Tables 2(e) and 3(e), respectively.

After training, we apply the resulting classifier to classify test instances. We select as the antecedent of an extracted event mention e the closest preceding mention that is classified as coreferent with e.

(a) Features for entity extraction. w is the word under consideration.

Lexical	word unigrams, bigrams, and trigrams formed from w with a window size of five.
Grammatical	w's part-of-speech (POS) tag; whether w is part of a NP; whether w is part of a pronoun, whether w is capitalized.
Semantic	the WordNet synset id of w; the WordNet synset ids of w's hypernym, its parent, and its grandparent.

(b) Features for entity coreference resolution. en_2 is the entity mention to be resolved and en_1 is a candidate antecedent of en_2.

Lexical	whether en_1 is a pronoun; whether en_1 is the subject of the sentence; whether en_1 is a noun; whether en_2 is a pronoun; whether en_1 is a noun; whether en_1 and en_2 have the exactly the same string; whether the modifiers of en_1 and en_2 match; the sentence distance between the strings of $en1$ and en_2.
Grammatical	the number, gender and animacy of en_1 and en_2; whether en_1 and en_2 agree w.r.t. number; whether en_1 and en_2 agree w.r.t. gender; whether en_1 and en_2 agree w.r.t. animacy.

(c) Features for event trigger identification and subtyping. t is the candidate trigger.

Lexical	t's POS tag, lemmatized and unlemmatized word unigrams, word bigrams, and word trigrams formed from t with a window size of five.
Syntactic	depth of t in its syntactic parse tree; path from the leaf node of t to the root in its syntactic parse tree; phrase structure expanded by the parent of t's node; phrase type of t's node.
Semantic	WordNet synset id of t; WordNet synset ids of t's hypernym, its parent, and its grandparent.

(d) Features for event argument identification and role labeling. en is a candidate argument of trigger t.

Basic	t's event subtype; en's entity type; en's head word; event subtype + head word; event subtype + entity type; t's POS tag.
Neighboring words	left/right neighbor word of en; left/right neighbor word of en + the word's POS; left/right neighbor word of en + the word's POS.
Syntactic	the phrase structure obtained by expanding the parent of t in the constituent parse tree; the phrase type of t; the path from en to t in the constituent parse tree; the dependency path from en to t.

(e) Features for event coreference resolution. ev_2 is the event mention to be resolved and ev_1 is a candidate antecedent of ev_2.

Event type features	whether ev_1 and ev_2 agree w.r.t. event type; whether they agree w.r.t. event subtype; the concatenation of their event types; and the concatenation of their event subtypes.
Trigger features	whether ev_1 and ev_2 have the same trigger; whether they have the same lemmatized trigger; whether the triggers of ev_1 and ev_2 or the hypernyms of these triggers are in the same WordNet synset; the concatenation of their triggers; the concatenation of POS tags of their triggers; whether their triggers agree in number if they are nouns; whether their triggers have the same modifiers and they are in the same entity coreference chain if they are nouns; the sentence distance between the triggers of ev_1 and ev_2; whether the triggers of ev_1 and ev_2 appear in a training document as a coreferent event mention pair; whether the triggers of ev_1 and ev_2 appear in the first sentence and headline if this is a newswire document; whether the sentence containing the the triggers of ev_1 and ev_2 are identical if this is a discussion forum document.
Argument features	whether ev_1 and ev_2 have arguments with the same role; whether the arguments have the same head word; whether they are in the same coreference chains; whether they have the same modifiers; the roles and number of the arguments that only appear in ev_1; and the roles and number of the arguments that only appear in ev_2.

Table 2: Features used in the English baseline system. POS tags, constituent parses and dependency parses are provided by CoreNLP (Manning et al., 2014). For all uses of WordNet (Fellbaum, 1998), only the first synset is used.

5 Joint Model

In this section, we describe our MLN-based joint model for event coreference resolution.

5.1 MLN Structure

Figure 1 shows our proposed MLN for event coreference resolution. It has five predicates subdivided into three categories: query, hidden and evidence.

The *query* predicate `EventCoref`(d,t_1,t_2) is true when two event mentions t_1 and t_2 in document d are coreferent. The *hidden* predicates are those that cannot be directly observed in the data. Our model contains three hidden predicates: (1) `Trigger`(d,t,p) is true when mention t in document d has event/trigger subtype p. A special type called "None" indicates that t does not contain a trigger. (2) `Argument`(d,t,a,r) asserts that entity mention a is an argument of event mention t in document d and its role is r. Again, we include a special role called "None", which indicates that the entity mention is not an argument of the event mention. The ! symbol in the predicate definition indicates that every entity mention must take one and only one argument role. (3) `EntityCoref`(d,a_1,a_2) is true when entity mentions a_1 and a_2 in document d are coreferent. The *evidence* predicates represent (ground) random variables

(a) Features for entity extraction. w is the word under consideration.

Lexical	word unigrams, bigrams, and trigrams formed from w with a window size of five.
Grammatical	w's POS tag; whether w is in a NP; whether w is part of a pronoun.
Wordlist-based	whether w can be found in each of the following 10 wordlists: Chinese surnames; famous GPE and location names (three wordlists); Chinese location suffixes; Chinese GPE suffixes; famous international organization names; famous company names; famous person names; and a list of pronouns.

(b) Features for entity coreference resolution. en_2 is an entity mention to be resolved and en_1 is a candidate antecedent of en_2.

Lexical	whether en_1 is a pronoun; whether en_1 is the subject of the sentence; whether en_1 is a noun; whether en_2 is a pronoun; whether en_1 is a noun; whether en_1 and en_2 are the same string; whether the modifiers of en_1 and en_2 match; the sentence distance between $en1$ and en_2.
Grammatical	the number, gender and animacy of en_1 and en_2; whether en_1 and en_2 agree w.r.t. number; whether en_1 and en_2 agree w.r.t. gender; whether en_1 and en_2 agree w.r.t. animacy.

(c) Features for event trigger identification and subtyping. t is a candidate trigger.

Lexical	word and POS n-grams formed from t with a window size of three
Syntactic	depth of t in its syntactic parse tree; path from the leaf node of t to the root in its syntactic parse tree; phrase structure expanded by the parent of t's node; the path from the leaf node of t to the governing clause; phrase type of t's node.
Semantic	whether t exists in a predicate list from the Chinese PropBank (Xue and Palmer, 2009); the entry number of t in a Chinese synonym dictionary.
Closest entity information	entity type of the syntactically/textually nearest entity to t in its syntactic parse tree; entity type of the syntactically/textually left/right nearest entity to t in its syntactic parse tree + entity.

(d) Features for event argument identification and role labeling. en is a candidate argument of trigger t.

Basic	t's event subtype; en's entity type; en's head word; t's subtype + en's head word; t's event subtype + en's entity type; t's POS tag.
Neighboring words	left/right neighbor word of en; left/right neighbor word of en + the word's POS tag; left/right neighbor word of t + the word's POS tag.
Syntactic	the phrase structure obtained by expanding the parent of t in the constituent parse tree; the phrase type of t; the path from en to t in the constituent parse tree; the dependency path from en to t.

(e) Features for event coreference resolution. ev_2 is the event mention to be resolved and ev_1 is a candidate antecedent of ev_2.

Event type features	whether ev_1 and ev_2 agree w.r.t. event type; whether they agree w.r.t. event subtype; the concatenation of their event types; and the concatenation of their event subtypes.
Trigger features	whether ev_1 and ev_2 have the same trigger; whether the trigger of ev_1 and ev_2 partially matched; whether they have the same lemmatized trigger; the concatenation of their triggers; the concatenation of part-of-speech tags of their triggers; whether their triggers agree in number if they are nouns; whether their triggers have the same modifiers if they are nouns; the sentence distance between the triggers of ev_1 and ev_2; the number of words between ev_1 and ev_2; whether the triggers of ev_1 and ev_2 appear in a training document as a coreferent event mention pair.
Argument features	whether ev_1 and ev_2 have arguments of the same role; whether the arguments have the same head word; whether they are in the same coreference chains; whether they have the same modifiers; the roles and number of the arguments that only appear in ev_1; and the roles and number of the arguments that only appear in ev_2.

Table 3: Features used in the Chinese baseline system. POS tags, constituent parses, and dependency parses are provided by CoreNLP (Manning et al., 2014). A detailed description of the wordlists used in the wordlist-based features can be found in Chen and Ng (2016). The Chinese synonym dictionary is HIT-SCIR's Tongyici cilin (extended).[3]

that can be directly observed in the data. In our MLN, we assume that we only observe the words; the predicate `Word`(d,t,w) is true when mention t in document d equals word w.

The MLN formulas are of two types. The first six formulas have infinite weight, which means that they are hard formulas and must always be satisfied. The last two formulas are soft, and their weights will be learned from the data. All logical variables in our formulas are universally quantified and therefore for brevity, we do not use them in the formulas. Formula 1 encodes the hard constraint that if two event mentions are coreferent, then they should share the same trigger subtype. Formula 2 specifies the hard constraint that if event mentions are coreferent, then their triggers subtypes cannot be "None." Formulas 3−6, all of which are hard formulas, specify the commutative and transitive properties of coreferent event and entity mentions. Formula 7, which is a soft formula, specifies the following dependency between coreferent entity mentions and coreferent event mentions: for two event mentions t_1 and t_2 having the same trigger subtype, if there exists an argument role r that is filled by argument a_1 in t_1

[3] `http://ir.hit.edu.cn/`

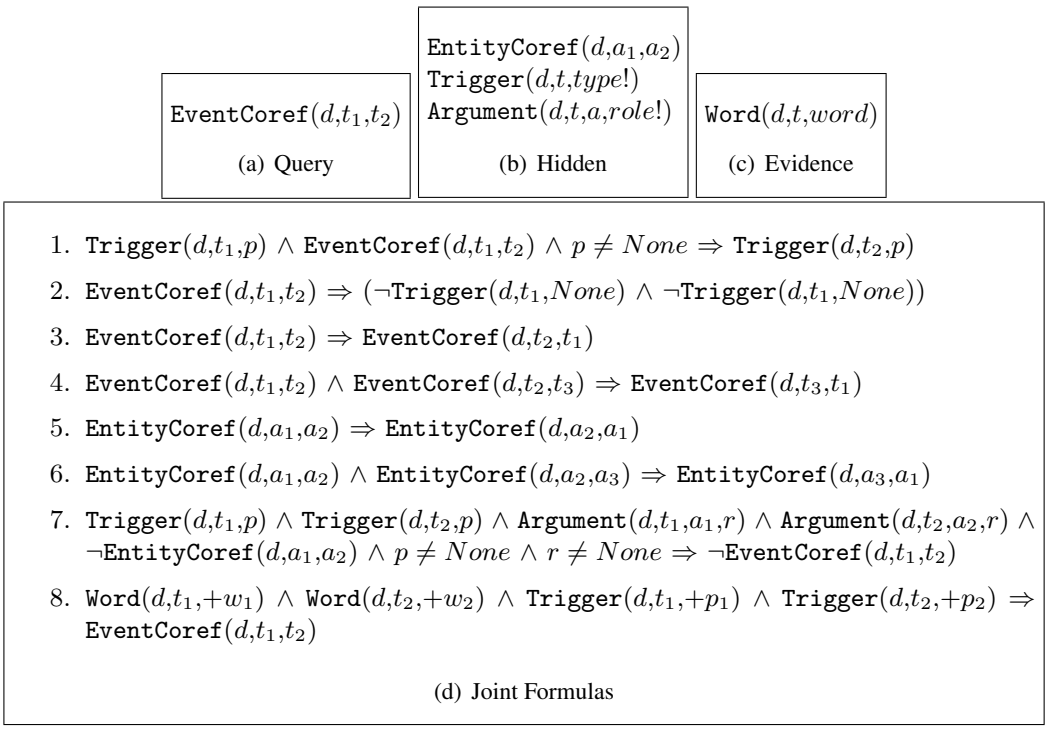


Figure 1: MLN structure.

and by a_2 in t_2, then t_1 and t_2 are not event coreferent if a_1 and a_2 are not entity coreferent.[4] Formula 8, which is also a soft formula, encodes the dependency between words in the text, trigger subtypes and event coreference. The $+$ sign in this formula indicates that for every grounding of the variables marked by the $+$ sign, we use a different weight for the soft formula.

5.2 Augmenting the MLN Distribution

Notice that the MLN shown in Figure 1 does not model the features used in the baseline systems. These features typically have high dimensionality and encoding them directly in the MLN is quite inefficient. For example, describing a trigram as an MLN formula results in d^3 ground formulas, where d is the number of words in our vocabulary. Therefore, the ground Markov network of an MLN that explicitly models all such high dimensional features would be extremely large and infeasible for inference. To address this issue, we implicitly encode the high-dimensional features by embedding them as weighted unit clauses, one for each grounding of the hidden and query predicates. Specifically, for each hidden/query ground atom X_i, we derive a weight $\phi(X_i)$ using the baseline system. This weight is computed as the distance from the hyperplane for the SVM-based classifiers and as a probability value for the CRF-based classifiers in the baseline system. We normalize each weight between the interval $[-1,1]$. The modified MLN distribution incorporating the new unit clauses is given by

$$P_{\mathcal{M}'}(\omega) \propto \exp\left(\sum_{f_i} \theta_i N_{f_i}(\omega)\right) \Phi(\omega) \tag{2}$$

where ω is a world (assignment on every ground atom) and $\Phi(\omega)$ acts as a prior on the set of hidden (**H**) and query (**Y**) ground atoms in the original MLN and is given by,

$$\Phi(\omega) = \exp\left(\sum_{X \in \mathbf{H} \cup \mathbf{Y}} \mathbb{I}_X(\omega)\phi(X)\right)$$

[4] According to the event coreference task definitions, arguments with certain roles cannot satisfy Formula 7. Hence, to reduce memory requirements, we restrict the application of Formula 7 to arguments having the following roles: Position, Person, Entity, Organization, Attack, Defendant, Adjudicator, Giver, Agent, Target, and Thing. In addition, we make it a soft (rather than hard) formula in view of the noisy outputs of our entity coreference resolver.

where $\mathbb{I}_X(\omega)$ is an indicator function that is equal to 1 if X is true in ω and 0 otherwise.

5.3 Setting the Soft Formula Weights

During inference time, we dynamically set the weights for the soft formulas (Formulas 7 and 8 in Figure 1) as follows. For each ground soft formula where its evidence atoms do not make it false, we set its weight to be the sum of the (normalized) SVM weights or CRF probabilities corresponding to its hidden and query atoms. We then multiply the soft weights with hyper-parameters η_1 and η_2 for Formulas 7 and 8 respectively and tune η_1 and η_2 using a grid search over the values $\{0.1, 0.25, 0.5, 0.75, 1.0\}$ to optimize the F1-score of event coreference resolution on the development set.

5.4 Inference

Given the prior-augmented MLN, $\mathcal{M}'$, the key task we are interested in is finding a truth assignment to all ground atoms of `EventCoref` that has the maximum probability given evidence on all ground atoms of `Word`. The following standard MAP inference task, which computes a joint assignment to all hidden and query variables given evidence, can be used to find the desired truth assignment.

$$\arg\max_{\omega} \left\{ \exp\left(\sum_{f_i} \theta_i N_{f_i}(\omega) \right) \Phi(\omega) \right\} \tag{3}$$

Unfortunately, the optimization problem given above is NP-hard in general. Moreover, the number of possible worlds in $\mathcal{M}'$ is extremely large and as a result naively searching over this large space (in order to solve the optimization problem) is computationally infeasible. As a concrete example, for the KBP 15 training dataset, we have 50 million ground atoms.

Fortunately, we can exploit the structure of the MLN given in Figure 1 in order to scale up MAP inference. In particular, the subset of ground atoms corresponding to two distinct documents are independent of each other. More formally, let $\mathbf{X}_i$ and $\mathbf{X}_j$ be the subset of ground atoms corresponding to two documents, say D_i and D_j respectively, then $\mathbf{X}_i$ is conditionally independent of $\mathbf{X}_j$ given evidence. Thus, given D documents in our corpus, the joint distribution represented by our MLN can be expressed as a product of D distributions. We can then perform inference independently over each such distribution, which greatly reduces the complexity of inference. Our inference procedure therefore follows an efficient, lazy, semi-lifted grounding strategy (Gogate and Domingos, 2011) that grounds the MLN for each document independently and solves Eq. (3) for each document separately using Gurobi (2013), a state-of-the-art integer linear programming solver.

6 Evaluation

6.1 Experimental Setup

We perform our evaluation on two corpora, the KBP 2015 English corpus and the Chinese portion of the ACE 2005 training corpus. For English, we train models on 128 of the training documents, tune parameters (the regularization parameters in SVM classifiers and the weights of the soft MLN formulas) on the remaining 30 training documents, and report results on the official test set.[5] For Chinese, since the ACE 2005 test set is not publicly available, we report five-fold cross validation results on the ACE 2005 training corpus. For each fold experiment, we employ three folds for classifier training, one fold for development (parameter tuning), and one fold for testing.

To evaluate event coreference performance on KBP, we follow the official KBP evaluation and employ four commonly-used scoring measures as implemented in version 1.7 of the official scorer provided by the KBP 2015 organizers, namely MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), CEAF_e (Luo, 2005) and BLANC (Recasens and Hovy, 2011), as well as the unweighted average of their F-scores.[6]

[5] Since the KBP 2015 corpus was not annotated with event arguments and entity coreference links, we train our entity mention extractor, our entity coreference resolver, and our event argument identification and role classification model on two LDC corpora provided by the TAC KBP 2015 task organizers (LDC2015E29 and LDC2015E68), as permitted by the rules of the shared task.

[6] The official KBP scorer is available at `http://cairo.lti.cs.cmu.edu/kbp/2015/event/scoring`.

Metric	English/KBP 2015						Chinese/ACE 2005					
	Baseline			**MLNs**			**Baseline**			**MLNs**		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
B^3	53.48	39.21	45.20	50.27	41.63	**45.54**	38.21	37.93	37.66	36.87	42.54	**39.50**
CEAF_e	42.33	38.54	**40.35**	47.53	33.48	39.29	40.28	37.76	38.98	41.02	41.19	**41.10**
MUC	50.52	29.13	36.96	47.07	38.21	**42.18**	40.02	40.27	40.14	39.37	44.70	**41.86**
BLANC	41.16	26.17	32.00	40.61	28.96	**33.30**	24.75	25.67	25.20	22.41	29.07	**25.29**
	Average = 38.64			**Average = 40.08**			CoNLL = 39.02			**CoNLL = 40.82**		

Table 4: Results for event coreference resolution on KBP 2015 and ACE 2005.

English/KBP 2015						Chinese/ACE 2005					
Baseline			**MLNs**			**Baseline**			**MLNs**		
Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
65.05	51.43	57.45	67.97	50.51	**57.95**	67.08	56.44	61.30	66.39	57.37	**61.55**

Table 5: Results for event trigger identification and subtyping on KBP 2015 and ACE 2005.

To evaluate event coreference performance on ACE, we follow previous work on event coreference (e.g., Yang et al. (2015)) and employ the aforementioned four scoring measures as implemented in the latest version (v8) of the CoNLL scorer (Pradhan et al., 2014), as well as the CoNLL score, which is the unweighted average of the MUC, B^3, and CEAF_e F-scores.[7] To our knowledge, there is only one difference between the implementations of the four scoring measures in the two scorers: while the CoNLL scorer considers an event mention correctly detected as long as it has an exact match with a gold event mention in terms of its left and right boundaries, the KBP 2015 scorer is stricter in that it considers an event mention correctly detected by additionally requiring that its event subtype be correctly determined.

6.2 Results and Discussion

The left half of Table 4 shows the results for English event coreference resolution on the KBP 2015 dataset. As can be seen, MLNs outperform the baseline system when evaluated on all but the CEAF_e metrics. W.r.t. the Average metric, MLNs achieve an F-score of 40.08, outperforming the baseline significantly by 1.44 points (paired t-tests, $p < 0.05$). To our knowledge, this is the best result reported to date on this corpus, with the top system in the KBP 2015 shared task achieving an Average F-score of 39.65. In general, the MLN could detect more event coreference chains than the baseline system, as seen from its higher recall in all but the CEAF_e metrics.[8]

The right half of Table 4 shows the results for event coreference resolution on the ACE 2005 Chinese corpus. As can be seen, MLNs outperform the baseline significantly by 1.8 points w.r.t. the CoNLL metric. In fact, MLNs achieve a higher score than the baseline w.r.t. each of the four scoring measures. Similar to what we observed on the KBP corpus, the consistently superior performance achieved by the MLN-based resolver can be attributed to its substantially higher recall accompanied by a slightly lower precision. In particular, since MUC is a link-based metric, the fact that the MLNs achieve a higher MUC recall on both datasets suggest that the MLNs are better at discovering event coreference links than the baseline.

One may argue that the MLNs may *not* be better than the baseline at discovering event coreference links: it may simply be the case that the joint inference process has allowed additional triggers to be extracted, which in turn allowed additional event coreference links to be established. To understand whether this is indeed the case, we compute the results for trigger identification and subtyping in Table 5. As can be seen, fewer English triggers are extracted after joint inference, whereas the reverse is true for Chinese. These results suggest that at least for English, the higher event coreference recall achieved by the MLNs is not attributable to better trigger identification and subtyping.

A closer examination of the outputs reveals that our resolver is comparatively better at extracting two types of coreference links that are traditionally considered difficult to extract. The first type involves triggers that are lexically different. For example, in the text segment "The former mayor of Detroit,

[7] The CoNLL scorer is available at `https://github.com/conll/reference-coreference-scorers`.

[8] As is commonly known, CEAF_e sometimes produces unintuitive scores. Specifically, the CEAF_e F-score may drop as more coreference links are correctly identified. See Moosavi and Strube (2016) for a detailed discussion.

Michigan was sentenced to 28 years in prison . . . Prosecutors asked for a minimum of 28 years for Kilpatrick, who resigned from the mayor's office in 2008 . . .", the link between event mentions triggered by *former* and *resigned*, both of which have type Personnel.End-position, is discovered by our resolver but not the baseline. The second type involves links between event mentions that are far from each other.

6.3 Error Analysis

To better understand how to improve our MLN-based resolver and to provide directions for future work, we conduct a qualitative analysis of its major sources of error in this subsection.

6.3.1 Two Major Types of Precision Error

Erroneous triggers. For both languages, our trigger classifier had difficulties with correctly classifying certain frequently-occurring words that are sometimes used as triggers and sometimes not. Specifically, the classifier misclassified many non-trigger instances of these words as triggers, which were subsequently used to establish coreference links by our resolver. A particularly interesting and challenging example involves the word "violent". Consider two sentences that appear in the same document: "The violent arrest of Ahmed al-Alwani is likely to inflame tensions in Sunni-dominated Anbar" and "Iraq troops arrest leading Sunni MP in violent raid". The first sentence contains two event mentions, one triggered by *violent* with type Conflict.Attack and the other triggered by *arrest* with type Justice.Arrestjail. The second sentence, contains only one event mention: it is triggered by *raid* with type Conflict.Attack and is coreferent with *violent*. While our system successfully detects all three triggers, it also erroneously detects *violent* in the second sentence as a trigger. This error gets propagated to our event coreference resolver, which posits the two occurrences of *violent* as coreferent.

Failure to extract arguments. Recall that our argument classifier does not extract any argument of an event mention that does not appear in the same sentence as its trigger. This severely limits its ability to extract arguments and has caused many spurious event coreference links to be established. For instance, our resolver erroneously posits two *violence* events as coreferent: it does not know that the two events took place in different countries, as the argument classifier failed to extract their location arguments (one is *Honduras* and the other is *Venezuela*).

6.3.2 Two Major Types of Recall Error

Missing triggers. For both languages, the trigger classifier failed to identify trigger words/phrases that are unseen or rarely-occurring in the training data. As a result, many links cannot be established.

Insufficient knowledge. Recall that our MLN-based resolver has achieved a higher recall than the baseline by doing a better job at establishing links between event mentions containing lexically different triggers. However, there are still many links between event mentions with lexically different triggers that our resolver fails to discover owing to the insufficient knowledge made available to it. This type of error is especially prominent on the Chinese corpus.

7 Conclusion

We proposed a novel joint inference based event coreference resolver using MLNs. Since encoding rich NLP features in MLNs is a challenging task, we encoded these features implicitly by adding weighted unit clauses to the MLN distribution. Results on an English corpus (KBP 2015) and a Chinese corpus (ACE 2005) show that our MLN based system achieved statistically significantly better performance than a pipeline-based resolver. Future work includes transferring our approach to other NLP tasks and exploring the possibility of incorporating active learning into our approach.

Acknowledgments

We thank the three anonymous reviewers for their detailed comments. This work was supported in part by NSF Grants IIS-1219142 and IIS-1528037, and by the DARPA PPAML Program under AFRL prime contract number FA8750-14-C-0005. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of NSF, DARPA and AFRL.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Jun Araki and Teruko Mitamura. 2015. Joint event trigger identification and event coreference resolution with structured perceptron. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2074–2080.

Jun Araki, Zhengzhong Liu, Eduard Hovy, and Teruko Mitamura. 2014. Detecting subevent structure for event coreference resolution. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 4553–4558.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the Linguistic Coreference Workshop at The First International Conference on Language Resources and Evaluation*, pages 563–566.

Cosmin Bejan and Sanda Harabagiu. 2010. Unsupervised event coreference resolution with rich linguistic features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1412–1422.

Cosmin Adrian Bejan and Sanda Harabagiu. 2014. Unsupervised event coreference resolution. *Computational Linguistics*, 40(2):311–347.

Zheng Chen and Heng Ji. 2009. Graph-based event coreference resolution. In *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing (TextGraphs-4)*, pages 54–57.

Chen Chen and Vincent Ng. 2014. SinoCoreferencer: An end-to-end Chinese event coreference resolver. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 4532–4538.

Chen Chen and Vincent Ng. 2016. Joint inference over a lightly supervised information extraction pipeline: Towards event coreference resolution for resource-scarce languages. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2913–2920.

Bin Chen, Jian Su, Sinno Jialin Pan, and Chew Lim Tan. 2011. A unified event coreference resolution by integrating multiple resolvers. In *Proceedings of the 4th International Joint Conference on Natural Language Processing*, pages 102–110.

Agata Cybulska and Piek Vossen. 2012. Using semantic relations to solve event coreference in text. In *Proceedings of the LREC Workshop on Semantic Relations-II Enhancing Resources and Applications (SemRel 2012)*, pages 60–67.

Pedro Domingos and Daniel Lowd. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool, San Rafael, CA.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Vibhav Gogate and Pedro Domingos. 2011. Probabilistic theorem proving. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 256–265.

Gurobi. 2013. *Gurobi Optimizer Reference Manual*. Gurobi Inc.

Yu Hong, Di Lu, Dian Yu, Xiaoman Pan, Xiaobin Wang, Yadong Chen, Lifu Huang, and Heng Ji. 2015. RPI BLENDER TAC-KBP2015 system description. In *Proceedings of the Eighth Text Analysis Conference (TAC 2015)*.

Kevin Humphreys, Robert Gaizauskas, and Saliha Azzam. 1997. Event coreference for information extraction. In *Proceedings of the Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, pages 75–81.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In Bernhard Scholkopf and Alexander Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 44–56. MIT Press.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 489–500.

Peifeng Li, Guodong Zhou, Qiaoming Zhu, and Libin Hou. 2012. Employing compositional semantics and discourse consistency in chinese event extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1006–1016.

Zhengzhong Liu, Jun Araki, Eduard Hovy, and Teruko Mitamura. 2014. Supervised within-document event coreference using information propagation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 4539–4544.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Katie McConky, Rakesh Nagi, Moises Sudit, and William Hughes. 2012. Improving event co-reference by context extraction and dynamic feature weighting. In *Proceedings of the 2012 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, pages 38–43.

Sean Monahan, Michael Mohler, Marc Tomlinson, Amy Book, Maxim Gorelkin, Kevin Crosby, and Mary Brunson. 2015. Populating a knowledge base with information about events. In *Proceedings of the Eighth Text Analysis Conference (TAC 2015)*.

Nafise Sadat Moosavi and Michael Strube. 2016. Which coreference evaluation metric do you trust? a proposal for a link-based entity aware metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 632–642.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 30–35.

Marta Recasens and Eduard Hovy. 2011. BLANC: Implementing the Rand Index for coreference evaluation. *Natural Language Engineering*, 17(4):485–510.

Sangeetha S. and Michael Arock. 2012. Event coreference resolution using mincut based graph clustering. *International Journal of Computing and Information Sciences*, pages 253–260.

Mark Sammons, Haoruo Peng, Yangqiu Song, Shyam Upadhyay, Chen-Tse Tsai, Pavankumar Reddyand, Subhro Roy, and Dan Roth. 2015. Illinois CCG TAC 2015 event nugget, entity discovery and linking, and slot filler validation systems. In *Proceedings of the Eighth Text Analysis Conference (TAC 2015)*.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the Sixth Message Understanding Conference*, pages 45–52.

Nianwen Xue and Martha Palmer. 2009. Adding semantic roles to the Chinese Treebank. *Natural Language Engineering*, 15(1):143–172.

Bishan Yang, Claire Cardie, and Peter Frazier. 2015. A hierarchical distance-dependent Bayesian model for event coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:517–528.

Event Detection with Burst Information Networks

Tao Ge[1,2]*, **Lei Cui**[3], **Baobao Chang**[1,2], **Zhifang Sui**[1,2], **Ming Zhou**[3]
[1]Key Laboratory of Computational Linguistics, Ministry of Education,
School of Electronics Engineering and Computer Science, Peking University, Beijing, China
[2]Collaborative Innovation Center for Language Ability, Xuzhou, Jiangsu, China
[3]Microsoft Research, Beijing, China
getao@pku.edu.cn, lecu@microsoft.com, chbb@pku.edu.cn
szf@pku.edu.cn, mingzhou@microsoft.com

Abstract

Retrospective event detection is an important task for discovering previously unidentified events in a text stream. In this paper, we propose two fast centroid-aware event detection models based on a novel text stream representation – Burst Information Networks (BINets) for addressing the challenge, following the *D2N2K (Data-to-Network-to-Knowledge)* paradigm. The BINets are time-aware, efficient and can be easily analyzed for identifying key information (centroids). These advantages allow the BINet-based approaches to achieve the state-of-the-art performance on multiple datasets, demonstrating the efficacy of BINets for the task of event detection.

1 Introduction

Retrospective Event Detection (RED) (sometimes called topic detection) is a core task for text stream analysis, which aims to detect events that are previously unknown to the system (Wayne, 1998; Rajaraman and Tan, 2001) and is useful for many applications such as text stream summarization and evolutionary analysis of events in both news and social streams.

docid	time	text
d_1	Jan 12, 2010	A 7.0 magnitude quake hits the impoverished Caribbean nation of Haiti, killing more than 200,000 people, injuring an estimated 300,000.
d_2	Feb 27, 2010	A huge magnitude 8.8 earthquake strikes near the coast of south-central Chile, shaking buildings, causing blackouts and killing at least 147 people.
d_3	Apr 14, 2010	A 7.1-magnitude earthquake struck Tibetan Autonomous Prefecture of Yushu in southern Qinghai Province on April 14, 2010, killing at least 400 people and injuring more than 10,000.

Table 1: Documents discussing different earthquake events.

Most previous event detection approaches tend to use document- or keyword-based clustering models. Another solution proposed in recent years is to build a keyword graph to model the co-occurrence of keywords for detecting keyword communities as events (Sayyadi and Raschid, 2013). Even though these methods can achieve fair performance in small datasets, they have either of the following limitations:

- No time-awareness: many event detection models do not take into account time information. As a result, it is very likely that the documents that talk about different events (as Table 1 shows) are grouped into one cluster just because their lexical similarity is high.
- Inefficient: clustering-based methods tend to be time-consuming. For example, the time complexity of GAC (group average clustering) – the most commonly used clustering method in event detection – is $O(n^2 \log n)$. The computational challenge makes them difficult to work on a large dataset.

*This work was done when the first author was visiting Microsoft Research Asia

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3276–3286, Osaka, Japan, December 11-17 2016.

- Deviation of cluster centroids: it is likely that the clusters obtained by the methods are not event-centric, which has an adverse effect on the result, as illustrated in Figure 1.

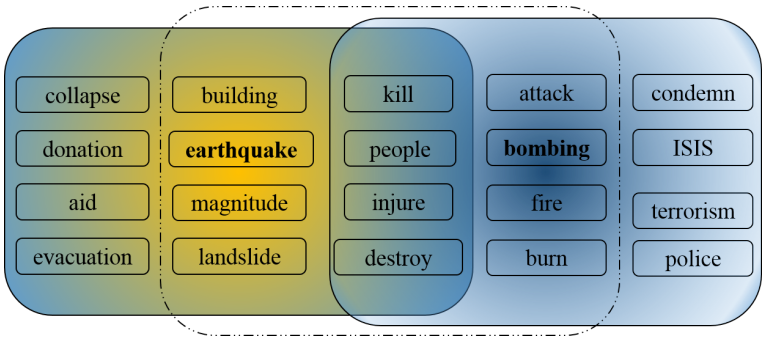


Figure 1: Deviation of cluster centroids: If clusters are not constructed around the centroid of the events (e.g., the dashline cluster is constructed around non-centroids such as *people*, *kill* and *injure* instead of *earthquake* or *bombing*), the performance will be adversely affected.

To offer a better solution to event detection without the above limitations, we propose to use a novel text stream representation: Burst Information Networks (BINets) (Ge et al., 2016a; Ge et al., 2016b). In contrast to the keyword graph which is based on word co-occurrence, a BINet is constructed based on burst co-occurrence. In a BINet (Fig. 2), a node is a burst of one word, which can be represented by the word with one of its burst periods, and an edge between two nodes indicates how strongly they are related (i.e., how frequently they co-occur). Since the nodes in a BINet contains temporal information (e.g., burst period), a BINet is time-aware in which nodes in a community are both topically and temporally coherent. Hence, we can say each community in a BINet corresponds to an event. Based on the BINet representation, we propose two fast centroid-aware event detection models. We show that the BINet-based models are efficient, allowing it to work on a large dataset, and the clusters obtained by the models center around the key information of events. Experiments on multiple datasets show that the BINet-based approaches achieve the state-of-the-art performance in terms of both accuracy and efficiency.

The contributions of this paper are:

- We propose to use BINets – a novel text stream representation for event detection, which is time-aware, can be efficiently built and support event-centric clustering, addressing the typical limitations of previous models.
- We propose two fast centroid-aware algorithms for event detection based on the BINet representation, which not only solve the centroid deviation problem but also are more efficient than traditional approaches.
- We construct and release a dataset for evaluating event detection models on a large text stream during a long time span.

2 Burst Information Networks

2.1 Burst Detection

A word's burst refers to a sharp increase of word frequency during a period. It usually indicates key information, important events or trending topics in a text stream as Figure 3 shows and is useful for many applications. In this paper, we detect a word's burst using the method of Zhao et al. (2012) which is a variant of (Kleinberg, 2003) and models burst detection as a burst state sequence decoding problem where a word w's burst state $s_t(w)$ at time t could be 1 or 0 to indicate if the word bursts or not at t. Specially, if a word w bursts at every time epoch during a period, we call this period a burst period of w and w has a burst during this period. In Figure 3, *earthquake* has 2 burst periods (i.e., Jan 12 - Jan 31, and Feb 27 - Mar 7), which correspond to two famous earthquake events (i.e., 2010 Haiti earthquake and 2010 Chile earthquake).

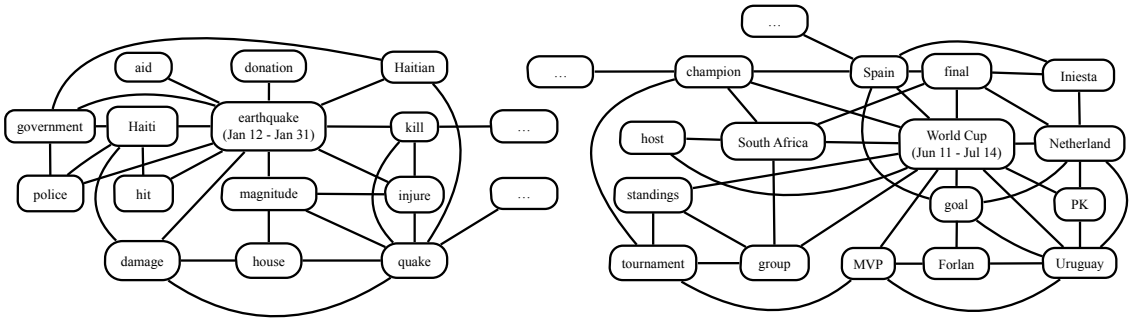


Figure 2: An example of Burst Information Network.

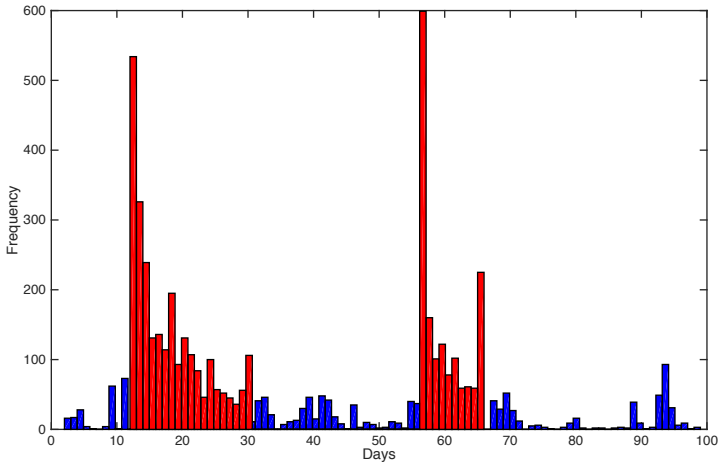


Figure 3: The frequency of *earthquake* during the first 100 days in 2010. There are two burst periods (red) for *earthquake* during the period, corresponding to two strong earthquake events happening in Haiti and Chile respectively.

Formally, we define $\mathcal{P}_i(w)$ as the ith burst period of the word w. It is a time interval, during which the word w bursts at every time epoch:

$$\mathcal{P}_i(w) = [t_i^s(w), t_i^e(w)]$$
$$\forall t \in \mathcal{P}_i(w) \;\; s_t(w) = 1$$

where $t_i^s(w)$ and $t_i^e(w)$ denotes the starting and ending time of the ith burst period of w, and $s_t(w)$ denotes the burst state of w at time t.

2.2 Burst Information Network Construction

A BINet represents associations between key facts in a text stream, which has been proven to be effective in multiple knowledge mining tasks (Ge et al., 2016a; Ge et al., 2016b). The basic component of a BINet is burst elements which are nodes of the information network:

A Burst Element is a burst of a word. It can be represented by a tuple: $\langle w, \mathcal{P}_i(w) \rangle$ where w denotes the word and $\mathcal{P}_i(w)$ denotes one burst period of w. Though a word may have multiple burst periods, a burst element has only one burst period. A word during its different burst periods will be regarded as different burst elements.

There are two main advantages using burst elements as nodes to build the information network:

- A burst element not only includes semantic information but also incorporates the temporal dimension. Nodes in a community are topically and temporally coherent while nodes that are topically or temporally distant cannot be adjacent, which makes it reasonable to consider a community in a BINet corresponds to an event.
- Since a burst element denotes a burst word during one of its burst period, its sense is likely to be consistent. Multiple bursts of a word will be considered as different burst elements. Therefore,

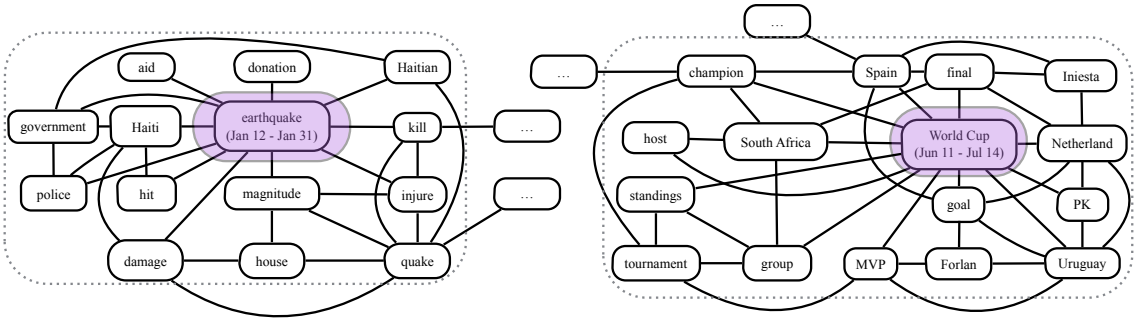


Figure 4: Node based detection model. Shaded nodes denote key nodes.

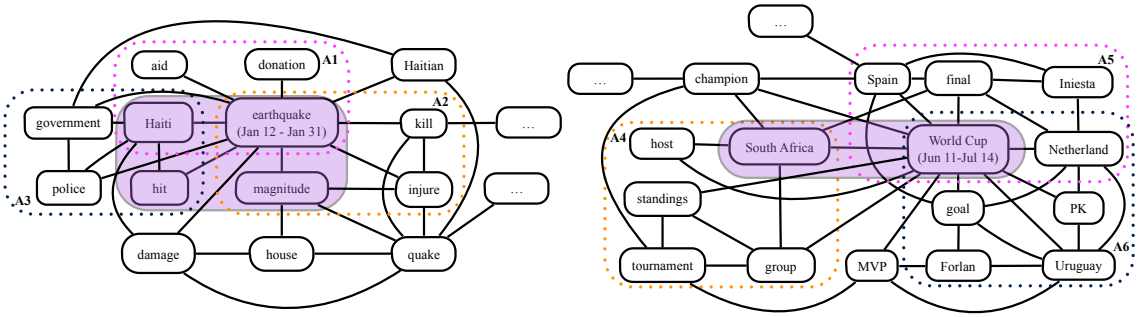


Figure 5: Area based detection model. Shaded areas denote key areas.

nodes in a BINet tends to be less ambiguous.

Formally, a BINet is defined as $G = \langle V, E \rangle$. Each node $v \in V$ is a burst element and each edge $e \in E$ denotes the association between burst elements. Intuitively, if two burst elements frequently co-occur, then they should be highly weighted. We define $\omega_{i,j}$ as the weight of an edge between v_i and v_j, which is equal to the number of documents where v_i and v_j co-occur.

3 Event detection based on the BINet

3.1 Motivation

The goal of event detection is to organize a text stream into multiple document sets, in each of which the documents coherently discuss the same event. The traditional clustering methods are usually inefficient and not time-aware. Moreover, they tend to suffer from the problem of deviation of cluster centroids, as illustrated in Figure 1. In Figure 1, *earthquake* and *bombing* are centroids (i.e., key information) of an earthquake event and a bombing event respectively. If clusters are constructed around the centroids (e.g., solid line clusters), the performance will be good; while if clusters center around non-centroid nodes (e.g., the dashline cluster centers around *kill* and *people*), the results will be poor.

To address the limitations above, we propose to model event detection problem as community detection on the BINet in which each community is both topically and temporally coherent, corresponding to one event. Instead of using popular community detection algorithms in social network analysis whose time complexity is high, we propose two fast centroid-aware event detection model: node-based detection model (NDM) and area-based detection model (ADM). Both of the approaches first identify the key nodes (or key areas) on the BINet, which indicate the centroid (i.e., key information) of events in the text stream, and then construct clusters that center around the key nodes (or key areas). The difference of the models is that NDM attempts to detect a bunch of node communities as clusters while ADM detects the overlapping document areas to form document clusters, as Figure 4 and Figure 5 depict. In some sense, NDM and ADM correspond to the keyword- and document-based clustering model respectively. In the following sections, we will present the details of NDM and ADM.

Community	The word of nodes with top PageRank value	Event
1	Iraq, war, Iraqi, US-led, Baghdad	Iraq war in 2003
2	flu, a/h1n1, health, virus, influenza	2009 A/H1N1 flu pandemic
3	earthquake, quake, Sichuan Province, Sichuan, quake-hit	Sichuan earthquake in 2008
4	Beijing, Olympic_Games, gold, medal, team	2008 Beijing Olympics
5	financial, crisis, global, economy, economic	financial crisis in 2008

Table 2: Example of the communities detected by our approach. Each community corresponds to one event and nodes with the top PageRank values tend to be keywords that are the most situable to describe the events.

3.2 Centroid-aware event detection models

3.2.1 Node-based detection model

The goal of node-based detection model (NDM) is to detect node communities on the BINet each of which corresponds to one event. To guarantee that detected communities center around the key nodes that correspond to key information (i.e,. centroid) of events in the text stream, we first identify the key nodes on the BINet.

Owing to the BINet representation, it is easy to identify the key nodes through the analysis of the network. Among a variety of ways to identify the influential nodes in a network, we simply adopt the Pagerank algorithm (Page et al., 1997). For a node v, its PageRank value $pr(v)$ is computed as follows:

$$pr(v) = d \sum_{v' \in N(v)} \hat{\omega}_{v,v'} \times pr(v') + \frac{1-d}{|V|}$$

where $|V|$ is the number of nodes in the BINet, $N(v)$ denotes the set of nodes adjacent to v, d is the damping factor and is set to 0.85, $\hat{\omega}_{v,v'} = \frac{\omega_{v,v'}}{\omega_{v',*}}$, which is the normalized weight of the edge between v and v'.

Intuitively, a node with a high PageRank value is usually important and likely to be the key node that indicates the key information of an event. Therefore, we rank nodes in the BINet by their PageRank value and choose the node which has the highest PageRank value and does not belong to any community as a key node to construct a community $\mathcal{E}$ around it with its closely related nodes:

$$\mathcal{E} = \{v\} \cup \{u | \hat{\omega}_{v,u} > \sigma_N\}$$

where v is the node with the highest PageRank value and does not belong to any community, $\hat{\omega}_{v,u}$ is the normalized weight of the edge between v and u, and σ_N is the threshold for selecting closely related nodes.

By repeating the process, we can detect multiple communities on the BINet efficiently, each of which centers around a key node. Table 2 shows some communities detected by this approach from 1995-2010 Xinhua news in English Gigaword. One can observe that each community corresponds to one event and nodes with the top PageRank values in a community tend to be key information of the events. We summarize the algorithm in Algorithm 1.

For NDM, we need to infer a document's event after community detection. For a document d, we infer the probability that d discusses the event e_k as follows:

$$P(e_k|d) = \frac{\sum_{v_k \in V_k(d)} pr(v_k)}{\sum_{v \in V(d)} pr(v)} \quad (1)$$

where $V(d)$ denotes the set of nodes that the words of d corresponds to in the BINet, $V_k(d) \subset V(d)$ denotes a subset of $V(d)$ that are in the community of the event e_k, and $pr(v)$ is the PageRank value of a node v. In Eq (1), the PageRank values of nodes in $V(d)$ can be considered as weights. The nodes with high PageRank values are highly weighted because they tend to indicate important topical and event information.

Algorithm 1 Node-based detection model

Input: Ranked list of nodes by PageRank value: $\mathcal{L}$, BINet: $G = \langle V, E \rangle$;
Output: A list of event communities: $\mathcal{C} = [\mathcal{E}_1, \mathcal{E}_2, ..., \mathcal{E}_k]$
while $\|\mathcal{L}\| > 0$ **do**
    $v \leftarrow \mathcal{L}[0]$ (the first element in $\mathcal{L}$)
    $\mathcal{E} \leftarrow \{v\} \cup \{u | \hat{\omega}_{v,u} > \sigma_N\}$
    $\mathcal{L} \leftarrow \mathcal{L} - \mathcal{E}$
    $\mathcal{C}.append(\mathcal{E})$
end while

3.2.2 Area-based Detection Model

A document area is the area (i.e., a set of nodes) on the BINet a document corresponds to. For example, $A3$ in Figure 5 is the area that the document written during the Haiti earthquake about *Haiti*, *government* and *police* corresponds to on the BINet. The idea of area-based detection model (ADM) is discovering the document areas that massively overlap on the BINet to construct clusters so that the documents whose areas are in the same cluster are about the same event. In contrast to NDM in which each item in a cluster is a node, the items in a cluster obtained by ADM is document areas on the BINet.

To guarantee that the clusters center around the centroids of events, we first identify key nodes on the BINet, as NDM does. In ADM, however, we treat a key area as the centroid of an event, which is different from NDM that treats a key node as an event centroid. To identify the key areas on the BINet, we first define the PageRank score of an area A as the normalized sum of the PageRank value of the nodes in it:

$$pr(A) = \frac{\sum_{v \in A} pr(v)}{\sqrt{|A|}}$$

Then, we repeatedly choose the area which has the highest PageRank score and does not belong to any cluster as a key area to construct a cluster with the areas that massively overlap it:

$$\mathcal{E} = \{A\} \cup \{A' | f(A, A') > \sigma_A\} \tag{2}$$

where σ_A is the threshold to construct cluster, $f(A, A')$ is a score to indicate how much A overlaps A' and it is computed as follows:

$$f(A, A') = \frac{|A \cap A'|}{|A \cup A'|} \tag{3}$$

We summarize the algorithm of ADM in Algorithm 2. As NDM, ADM detects events in a greedy manner; hence, the detection process is fast. However, in contrast to NDM, ADM allows one area to belong to multiple communities, which means that one document could belong to multiple events.

Algorithm 2 Area-based detection model

Input: Ranked list of documents areas: $\mathcal{L}$, BINet: $G = \langle V, E \rangle$;
Output: A list of event communities: $\mathcal{C} = [\mathcal{E}_1, \mathcal{E}_2, ..., \mathcal{E}_k]$
while $\|\mathcal{L}\| > 0$ **do**
    $A \leftarrow \mathcal{L}[0]$ (the first element in $\mathcal{L}$)
    $\mathcal{E} \leftarrow \{A\} \cup \{A' | f(A, A') > \sigma_A\}$
    $\mathcal{L} \leftarrow \mathcal{L} - \mathcal{E}$
    $\mathcal{C}.append(\mathcal{E})$
end while

4 Experiments and Evaluation

We conduct experiments to evaluate the performance of our approach. We first evaluate our approach on the TDT4 dataset to compare other event detection approaches. Then, we apply our approach on a larger corpus (2009 – 2010 news corpus) to test its scalability and performance.

For preprocessing, we remove stopwords and conduct lemmatization and name tagging using Stanford CoreNLP toolkit (Manning et al., 2014) before the construction of a BINet.

4.1 Evaluation on TDT4

The TDT4 collection is a well known dataset for comparing methods for event detection. The English part of the dataset includes approximately 29,000 news documents from news agencies such as CNN and BBC from October 2000 to Janurary 2001 (spanning 4 months), while only 1,884 documents[1] are annotated to be related to 71 human identified events (topics). As the setting adopted by previous work (Li et al., 2005; Sayyadi and Raschid, 2013), we use the annotated subset as gold standard for evaluating the performance of our models.

As most of the previous work (Yang et al., 1998; Li et al., 2005) addressing the event detection challenge, we use Micro-Precision, Micro-Recall, Micro-F1 as well as Macro-F1 to evaluate the performance. We compare our approach to the following models whose effectiveness on the TDT4 corpus has been verified by previous work:

- Allan[2] (Allan et al., 1998): A popular online event detection model, which is often used as a baseline to compare event detection models.
- GAC (Yang et al., 1998): A classical but effective approach for event detection using group average clustering.
- KeyGraph (Sayyadi and Raschid, 2013): Betweenness score based community detection approach on KeyGraph. It is notable that the evaluation measures used in Sayyadi and Raschid (2013) are somewhat different from those in this paper and other work – they used Macro-precision, Macro-recall[3] and Macro-F1. We only report its Macro-F1 in Table 3.
- Probabilistic model (Li et al., 2005): A time-aware probabilistic graphical model for event detection. It is the state-of-the-art approach on TDT4 dataset.

Models	Micro-P	Micro-R	Micro-F1	Macro-F1
Allan	0.64	0.57	0.60	0.62
GAC	0.83	0.63	0.72	0.75
KeyGraph	-	-	-	0.69
Probabilistic Model	0.85	0.67	0.75	0.78
BINet-NDM	0.79	0.69	0.74	0.75
BINet-ADM	0.81	0.70	0.75	0.77

Table 3: Performance of various event detection approaches on TDT4.

Table 3 shows the results[4] on the TDT4 dataset. The BINet approaches perform well on the dataset: Both NDM and ADM outperform the classical baselines (i.e., Allan, GAC and Keygraph). The ADM performs better than NDM and even achieves the comparable performance to the state-of-the-art approach by (Li et al., 2005) because the centroid in ADM is a key area that contains more information than a key node in NDM. The reasons for the good performance are two-fold: First, the BINet-based approach is time-aware, which avoid many unnecessary mistakes made by the baseline models that only take into account text content; Second, the BINet-based models are centroid-aware, which guarantee that

[1] Among these 1,884 documents, there are 38 documents belonging to more than one event.

[2] This baseline is often referred as kNN in literature (Li et al., 2005; Sayyadi and Raschid, 2013). However, to avoid the ambiguity with the popular kNN classification model, we simply refer it as Allan.

[3] For reference, the Macro Precision and Recall reported in Sayyadi and Raschid (2013) are 0.82 and 0.59 respectively.

[4] The results of *Allan*, *GAC* and *Probabilistic Model* are from Li et al. (2005) while the results of *KeyGraph* come from Sayyadi and Raschid (2013).

Model	Micro-P	Micro-R	Micro-F1	Macro-F1	Running time
GAC	-	-	-	-	>2 hours
KeyGraph	-	-	-	-	>2 hours
Probabilistic model	-	-	-	-	>2 hours
B-GAC	0.81	0.65	0.72	0.67	7189s (896s)
BINet-NDM	0.85	0.68	0.76	0.69	3591.98 (1350.08s)
BINet-ADM	0.84	0.71	0.77	0.71	3610.03s (1368.13s)

Table 4: Performance and running time of various event detection models on the 2-year news stream. We do not report the precision, recall and f-score for the models that cannot get results within 2 hours. The number in the round bracket is the running time of the model when it is run in 8-way parallel. The running time is measured on a workstation with Intel Xeon 3.5 GHz CPU and 64GB RAM.

the generated clusters center around centroids of events and avoid the problem of deviation of cluster centroids.

4.2 Evaluation on a 2-year news stream

Even though TDT4 is a widely used dataset for event detection, it has several limitations: First, the period of TDT4 dataset is short (only 4 months) as Li et al. (2005) claimed. In TDT4 dataset, hardly can we see multiple events of the same type in the TDT4 dataset (e.g., there is only one flood event in TDT4 dataset). Therefore, even if we just use content-based clustering methods regardless of time information, the performance is not bad. Second, the data size of the TDT4 corpus is so small compared with a real text stream that many stream-based features such as burst cannot function as well as in a real stream. To test the performance of our detection models on a real text stream, we construct a dataset using 2009 – 2010 news from English Gigaword (APW and XIN sections) as a text stream where there are 584,414 news articles in total. We construct a BINet on this dataset, which contains 46,254 nodes and 514,682 edges. For evaluation, we select 83 events that happened during 2009 – 2010 and annotate their relevant documents in the text stream. The selected events are all important events and have their corresponding Wikipedia pages. The annotation process is similar to (Li et al., 2005): we use the Wikipedia title of the events to search the candidate documents using Lucene and then manually identify if the returned documents are actually relevant to the events. Since this annotation process does not guarantee finding all the relevant documents to an event, we call the annotations silver standard[5]. In total, there are 2,584 documents that are annotated as relevant to those 83 events.

Table 4 shows the results of various approaches on the 2-year news stream. Due to the size of the dataset, most traditional event models cannot finish the detection task within two hours since their time complexity is too high. The B-GAC model proposed by (Zhao et al., 2012) is the only one that can finish the task within 2 hours because it adopts the split-merge-clustering strategy that splits[6] the data into multiple small pieces by time for clustering and then merges the clusters. Though such a strategy can alleviate the issue of the scalability, the split of data will affect the global overview of the text stream and have an adverse effect on finding the centroids of events. In contrast, our BINet-based approaches can finish detecting events within 1 hour without splitting the stream and achieve the best result owing to their awareness of both time[7] and event centroids.

We compare the time complexity of our centroid-aware event detection models to other commonly used event detection approaches, as shown in Table 5 where n is the number of documents, K is the event number (K in our BINet-based approaches depends on the selection of σ_N and σ_A), $|W|$ is the size of vocabulary, and $|V|$ and $|E|$ are the number of nodes and edges on the BINet respectively. The running time of NDM and ADM consist of four parts: burst detection, BINet construction, PageRank analysis,

[5]The annotation data can found at http://getao.github.io

[6]We split the 2-year news stream into 8 small pieces, each of which is a 3-month news stream so that it can get the result within 2 hours.

[7]For our BINet-based approaches, only burst detection part is run in parallel in 8-way parallel setting, which is different from B-HAC that splits the stream and clusters documents in parallel.

Models	Time complexity
GAC	$O(n^2 \log n)$
B-GAC	$O(n^2 \log n)$
Keygraph	$O(nK + \|W\|^3)$
NDM	$O(nK + \|V\| \log \|V\| + \|E\| + \|W\|T)$
ADM	$O(n(\log n + L) + \|V\| + \|E\| + \|W\|T)$

Table 5: Time complexity of various event detection models.

and event detection. The first three parts are the same for NDM and ADM, which are preliminary steps for event detection. The time complexity of burst detection algorithm is $O(T)$ for one word where T is the time span of the stream and it can be conducted in parallel for different words because the burst detection processes for different words are independent. The time complexity of the BINet construction and PageRank analysis is $O(n)$ and $O(|V| + |E|)$ respectively. For the event detection part, the time complexity of community detection of NDM is $O(|V| \log |V| + |E| + nK)$. The former term is the time cost for sorting nodes by PageRank value, and the second and the third term are the cost for constructing node communities and assigning events to documents respectively. The time complexity of the detection part in ADM is somewhat different. Its time complexity is $O(n \log n + nL)$. As NDM, the first term is the time for sorting document areas by PageRank value. The second term is the time cost for computing Eq (3) in which L is the average number of times that a document (area) is taken for computing Eq (3) and is affected by the selection of threshold parameter σ_A. In the worst case, $L = K$; while in the best case, $L = 1$, meaning that a document is taken for computing Eq (3) only once. In most cases, L is a small number. The running time of those parts of NDM and ADM is shown in Table 6. Note that, for the part of the PageRank computation, the time is measured by running the PageRank algorithm for 1,000 iterations.

	BINet-NDM	**BINet-ADM**
Burst detection	2,562.17s (320.27s)	2,562.17s (320.27s)
BINet construction	304.56s	304.56s
PageRank computation	716s	716s
Event detection	9.25s	27.3s
Total	3591.98s (1350.08s)	3610.03s (1368.13s)

Table 6: The running time of 4 parts of our BINet-based event detection approaches. The number in the round bracket is the running time of the model when it is run in 8-way parallel.

5 Related Work

Event detection is one of the most popular research topics in recent years and has been extensively studied for the decades (Yang et al., 1998; Swan and Allan, 2000; Allan, 2002; Fung et al., 2005; He et al., 2007; Sayyadi et al., 2009; Zhao et al., 2012; Sayyadi and Raschid, 2013; Ge et al., 2015). They are based on either document- or keyword-based clustering, which usually suffer from either unawareness of time, high expensive computation cost or deviation of cluster centroids. In contrast, our approach is time-aware, centroid-aware and so efficient that it can be run on a large text stream.

In addition, there is much work (Sakaki et al., 2010; Lee et al., 2011; Diao et al., 2012; Aggarwal and Subbian, 2012; Wang et al., 2013; Dong et al., 2015) studying event detection problem in social media. They usually use more or less social media features such as spatio-temporal information, which are not in the same setting with our task.

6 Conclusion and Future Work

This paper proposes to use a novel text stream representation – Burst Information Networks to address the retrospective event detection challenge. Based on the BINet, we propose two fast centroid-aware event

detection models that can effectively overcome the limitations of the previous event detection models and achieve the state-of-the-art performance on both TDT4 and a long-span text stream.

In the future, we plan to study events in a text stream more deeply based on the BINet representation. Since a BINet can offer a global overview of events in the stream level, we plan to use the BINets to derive an event's type, extract its schema and even fill its slots after we detect its corresponding regions on the BINet. Hopefully, this framework could work for endless event knowledge mining if it could be used for monitoring the massive text streams.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work is supported by the National Key Basic Research Program of China (No.2014CB340504), the Research Fund for the Doctoral Program of Higher Education (20130001110027) and the National Natural Science Foundation of China (No. 61375074, 61273318). The contact author is Zhifang Sui.

References

Charu C Aggarwal and Karthik Subbian. 2012. Event detection in social streams. In *SDM*.

James Allan, Ron Papka, and Victor Lavrenko. 1998. On-line new event detection and tracking. In *SIGIR*.

James Allan. 2002. *Topic detection and tracking: event-based information organization*, volume 12. Springer Science & Business Media.

Qiming Diao, Jing Jiang, Feida Zhu, and Ee-Peng Lim. 2012. Finding bursty topics from microblogs. In *ACL*.

Xiaowen Dong, Dimitrios Mavroeidis, Francesco Calabrese, and Pascal Frossard. 2015. Multiscale event detection in social media. *Data Mining and Knowledge Discovery*, 29(5):1374–1405.

Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Philip S Yu, and Hongjun Lu. 2005. Parameter free bursty events detection in text streams. In *VLDB*.

Tao Ge, Wenzhe Pei, Heng Ji, Sujian Li, Baobao Chang, and Zhifang Sui. 2015. Bring you to the past: Automatic generation of topically relevant event chronicles. In *ACL*.

Tao Ge, Lei Cui, Baobao Chang, Sujian Li, Ming Zhou, and Zhifang Sui. 2016a. News stream summarization using burst information networks. In *EMNLP*.

Tao Ge, Qing Dou, Xiaoman Pan, Heng Ji, Lei Cui, Baobao Chang, Zhifang Sui, and Ming Zhou. 2016b. Aligning coordinated text streams through burst information network construction and decipherment. *arXiv preprint arXiv:1609.08237*.

Qi He, Kuiyu Chang, and Ee-Peng Lim. 2007. Using burstiness to improve clustering of topics in news streams. In *ICDM*.

Jon Kleinberg. 2003. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397.

Chung-Hong Lee, Hsin-Chang Yang, Tzan-Feng Chien, and Wei-Shiang Wen. 2011. A novel approach for event detection by mining spatio-temporal information on microblogs. In *ASONAM*.

Zhiwei Li, Bin Wang, Mingjing Li, and Wei-Ying Ma. 2005. A probabilistic model for retrospective news event detection. In *SIGIR*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*.

Larry Page, S Brin, R Motwani, and T Winograd. 1997. Pagerank: Bringing order to the web. Technical report, Stanford Digital Libraries Working Paper.

Kanagasabi Rajaraman and Ah-Hwee Tan. 2001. Topic detection, tracking, and trend analysis using self-organizing neural networks. In *PAKDD*.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW*.

Hassan Sayyadi and Louiqa Raschid. 2013. A graph analytical approach for topic detection. *ACM Transactions on Internet Technology (TOIT)*, 13(2):4.

Hassan Sayyadi, Matthew Hurst, and Alexey Maykov. 2009. Event detection and tracking in social streams. In *ICWSM*.

Russell Swan and James Allan. 2000. Automatic generation of overview timelines. In *SIGIR*.

Xun Wang, Feida Zhu, Jing Jiang, and Sujian Li. 2013. Real time event detection in twitter. In *WAIM*.

Charles Wayne. 1998. Overview of tdt.

Yiming Yang, Tom Pierce, and Jaime Carbonell. 1998. A study of retrospective and on-line event detection. In *SIGIR*.

Wayne Xin Zhao, Rishan Chen, Kai Fan, Hongfei Yan, and Xiaoming Li. 2012. A novel burst-based text representation model for scalable event detection. In *ACL*.

Corpus Fusion for Emotion Classification

Suyang Zhu[1], **Shoushan Li**[1*], **Ying Chen**[2], **Guodong Zhou**[1]
[1]Natural Language Processing Lab, School of Computer Science and Technology,
Soochow University, China
[2]College of Information and Electrical Engineering, China Agricultural University
syzhu@stu.suda.edu.cn, lishoushan@suda.edu.cn,
chenying@cau.edu.cn, gdzhou@suda.edu.cn

Abstract

Machine learning-based methods have obtained great progress on emotion classification. However, in most previous studies, the models are learned based on a single corpus which often suffers from insufficient labeled data. In this paper, we propose a corpus fusion approach to address emotion classification across two corpora which use different emotion taxonomies. The objective of this approach is to utilize the annotated data from one corpus to help the emotion classification on another corpus. An Integer Linear Programming (ILP) optimization is proposed to refine the classification results. Empirical studies show the effectiveness of the proposed approach to corpus fusion for emotion classification.

1 Introduction

Emotion classification aims to recognize human emotions, such as joy, $anger$ or $surprise$ in a given text. Emotion classification has a variety of applications including online chatting (Galik and Rank, 2012), news classification (Liu et al., 2013) and stock marketing (Bollen et al., 2011). In recent years, emotion classification in social media has been greatly popular in the Natural Language Processing (NLP) community (Chen et al., 2010; Purver and Battersby, 2012; Li et al., 2015). Because of the popularity of social media today, the analysis of short text on social media becomes more important (Kiritchenko et al., 2014; Wen and Wan, 2014; Wang et al., 2015). Users express their feelings and emotions on various social media platforms.

Existing emotion classification approaches are based on corpus classification methods where human-annotated emotion corpora are leveraged to train a machine learning-based emotion classification models. Recently, several different emotion corpora have been proposed by different researchers, such as Yao et al. (2014) and Huang et al. (2015). However, the size of each existing labeled corpus might be rather limited due to the high cost of data annotation, which results low performance in traditional supervised emotion classification.

In this paper, we propose a novel task, namely corpus fusion for emotion classification, which aims to leverage the data from different emotion corpora so as to alleviate the data deficiency problem. This task is motivated by the fact that although the emotion corpora are from different resources, they have the same objective of emotion classification. So it is easy to enlarge the size of the corpora by mixing the data of the same emotion category from two corpora. However, corpus fusion for emotion classification is challenging due to the following two factors:

First, the emotion taxonomies are often different between two emotion corpora because of the lack of an accepted standard. As a result, similar instances which express similar or same emotion can be categorized into different types of emotions under different taxonomies. An example from two emotion corpora (Yao et al., 2014; Huang et al., 2015) in Figure 1 expresses this problem. These two instances which express close emotion are labeled with different emotion classes under different emotion taxonomies.

Second, the annotation guidelines are often different between two emotion corpora because of different annotators. For example, the instance from Yao et al. (2014) in Figure 1 which contains a positive

*corresponding author

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3287–3297, Osaka, Japan, December 11-17 2016.

emotion $like$ may not be labeled as $positive$ under the taxonomy of Huang et al. (2015) because it doesn't contain a strong emotional word to express such a positive emotion. The numbers of emotion labels of each instance in two corpora are also different: some instances in Yao et al. (2014) have both primary emotion and secondary emotion, while most of the instances in Huang et al. (2015) are labeled with only one emotion. Because of the difference between emotion taxonomies, it is difficult to directly use the data from different emotion corpora together.

Yao et al. (2014): "我先是震惊，继而敬重，如同听到冯军哥的裸捐一样。" *(English Translation: "I was first shocked, and then respected. Just like when I heard the giving pledge made by Feng Jun.")* *Emotion Categories:* ***Like, Surprise***
Huang et al. (2015): "创意无处不在！令人感到震惊的街头3D艺术！" *(English Translation: "Creativity is everywhere! The shocking 3D street arts!")* *Emotion Categories:* ***Neutral Complex***

Figure 1: An example for two similar instances being categorized into different emotion categories under two emotion taxonomies

In this paper, we propose a corpus fusion approach to leverage the two emotion corpora, i.e., Yao et al. (2014) and Huang et al. (2015) in order to utilize the annotated data from each other. First, we perform supervised emotion classification on two corpora. Second, we refine the predicted emotion labels via a joint inference method, called Integer Linear Programming (ILP). A global objective function is minimized with the obtained posterior probabilities of the test instances. Two types of constraints, namely intra-corpus constraint and extra-corpus constraint are proposed in the ILP approach to address two challenges mentioned above. We use extra-corpus constraint to overcome the first challenge, and intra-corpus constraint is used for overcoming the second challenge. Results of experiments prove that our model makes a promotion on both classification accuracy and F_1-measure, and both intra-corpus constraints and extra-corpus constraints are effective for the corpus fusion task.

The reminder of this paper is organized as follows. Section 2 overviews the related studies. Section 3 introduces two corpora used in this paper. Section 4 proposes the approach to corpus fusion for emotion classification. Section 5 illustrates the experiments to evaluate the proposed approach. Section 6 gives the conclusion and future work.

2 Related Work

In last decade, mainstream approaches for emotion analysis are corpus-based machine learning methods. Several studies construct emotion corpus from social media platform such as blog, microblog and news portal. Gilad (2005) collects blog texts from LiveJournal to construct an emotion corpus with 815,494 blog articles. Quan and Ren (2009) build an emotion corpus from blogs with eight types of emotions on three granularity levels. Pak and Paroubek (2010) establish an emotion corpus by capturing tweets on Twitter. Yao et al. (2014) build an emotion corpus with seven emotion types from SINA microblog. Huang et al. (2015) construct an emotion from TENCENT microblog including both simple and complex emotion annotation.

According to the text granularity, emotion analysis works can be generally divided into three levels: document-level, sentence-level, and word-level. Gilad (2005) uses SVM to model a document-level

emotion classifier with blog articles. Yang et al. (2007) identify the emotion types of blog articles based on SVM and CRF with sentiment lexicon. Lin et al. (2007) use the articles on Yahoo! News to analysis the news readers' emotion.

Sentence-level emotion analysis is mainly based on emotion lexicon. Mohammad and Turney (2010) study the effect of word level emotion lexicons for sentence level emotion analysis. They use word level emotion lexicons based on Word Net and NRC-10 to predict the emotion in sentences with Logistic Regression and SVM. Das and Bandyopadhyay (2010) categorize the emotions on Bengali blog. They first identify the emotion of words in a sentence, then judge the emotion of this sentence according to the words' emotion. Aman and Szpakowicz (2007) implement a knowledge-based sentence level emotion recognition method.

Word-level emotion analysis aims to construct emotion lexicon, which plays an important auxiliary role in emotion analysis. Yang et al. (2014) propose Emotion-aware LDA model to build a domain-specific lexicon. Xu et al. (2010) use language resource, such as synonym dictionary, semantic dictionary, and labeled and unlabeled corpus to construct the similarity matrix between words and seed words. They build an emotion lexicon with five emotion classes using graph-based rules. As a special expression of words, emoticons play an important role in emotion analysis due to the explosion in social media. Tang et al. (2013) annotate data from microblog posts with the help of emoticons.

There are several studies to address corpus adaptation problem in NLP field. Gao et al. (2004) do a pioneer work by describing a transformation-based converter to transfer a certain word segmentation result to another annotation guideline. Jiang et al. (2009) investigate the automatic integration of word segmentation knowledge in different annotated corpora. Similar approaches are applied to constituency parsing (Zhu et al., 2011) and word segmentation (Sun and Wan, 2012)

Unlike all above studies, we propose a corpus fusion approach to emotion classification in order to address the corpus fusion problem to combine two corpora with different emotion taxonomies and annotation guidelines. To the best of our knowledge, this is the first attempt to address this task in emotion analysis.

3 Corpus

Two emotion corpora we used are respectively constructed by Yao et al. (2014) and Huang et al. (2015). We simply denote the two corpora as YAO (2014) and HUANG (2015) in the rest of this paper for convenience.

YAO (2014) is constructed from SINA microblog[1]. It categorizes emotions into seven classes: $happiness$, $anger$, $sadness$, $fear$, $like$, $surprise$ and $disgust$. The corpus consists of 14,000 instances, of which 7,407 instances express emotions. Each instance may include both primary emotion and secondary emotion, or just has one primary emotion. Table 1a illustrates the distribution of primary emotions and secondary emotions in this corpus.

Notation	Emotion Class	Primary Emotion	Secondary Emotion
e_{Y_1}	$happiness$	1460	359
e_{Y_2}	$anger$	669	203
e_{Y_3}	$sadness$	1173	269
e_{Y_4}	$fear$	148	61
e_{Y_5}	$like$	2203	546
e_{Y_6}	$surprise$	362	170
e_{Y_7}	$disgust$	1392	385
-	**Total**	**7407**	**1993**

(a) YAO (2014)

Notation	Emotion Class	Amount
e_{H_1}	joy	1038
e_{H_2}	$anger$	472
e_{H_3}	$sadness$	581
e_{H_4}	$fear$	94
e_{H_5}	$positive$	1178
e_{H_6}	$neutral$	1131
e_{H_7}	$negative$	2175
-	**Total**	**6669**

(b) HUANG (2015)

Table 1: Emotion categories and distribution on two corpora

[1] http://weibo.com

Huang et al. (2015) propose another emotion taxonomy with both basic emotions and complex emotions. Basic emotions include four emotion classes: joy, $anger$, $sadness$ and $fear$. Complex emotions contain three emotion classes: $positive$, $neutral$ and $negative$. This corpus is constructed from TENCENT microblog[2], and it consists of 15,540 instances. 6,669 instances express certain emotion. Although there is a very few multi-label annotation on it, we consider this corpus as single-label annotated. Table 1b shows the distribution of emotions in the two corpora.

4 Approach to Corpus Fusion for Emotion Classification

The corpus fusion approach to emotion classification aims to exploit the relationship between two corpora which have similar emotion taxonomies. Figure 2 illustrates the framework of our model. The testing results generated by the supervised emotion classifier are refined by ILP with label constraints.

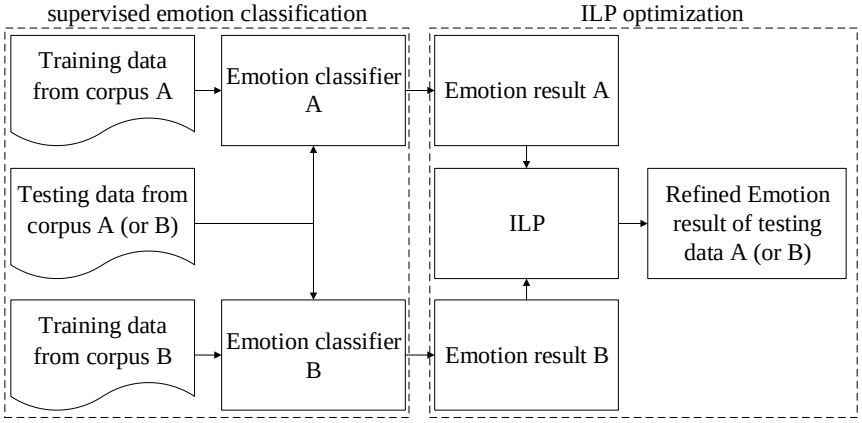


Figure 2: The framework of corpus fusion for emotion classification with ILP

4.1 Supervised Emotion Classification

Supervised classification problem trains a predictor f which maps an input vector x to the corresponding class label y on a set of training data. In emotion classification, a feature vector x is extracted from the instance. Formally, the objective of classification is defined as follows:

$$f(x) \rightarrow y, \quad y \in \{emotion1, emotion2, ...\} \tag{1}$$

In this task, we train plural binary predictors for each emotion class for the testing set from YAO (2014), and a 7-way predictor for the testing set from HUANG (2015). For one sample instance t_i from the testing set, predicting results r_{Y_i} and r_{H_i} indicating the predicted emotion labels, and we get two sets of probabilities P_{Y_i} and P_{H_i} which contain the probabilities of this sample belonging to each category in two emotion taxonomies:

$$\begin{aligned} P_{Y_i} &= \{p(r_{Y_i} = e_{Y_1}),\ p(r_{Y_i} = e_{Y_2})\ ...p(r_{Y_i} = e_{Y_7})\}, \\ P_{H_i} &= \{p(r_{H_i} = e_{H_1}),\ p(r_{H_i} = e_{H_2})\ ...p(r_{H_i} = e_{H_7})\} \end{aligned} \tag{2}$$

where $p(r_{Y_i} = e_{Y_1})$ denotes the probability of t_i belonging to $happiness$ under the emotion taxonomy of YAO (2014), and $p(r_{H_i} = e_{H_1})$ denotes the probability of t_i belonging to joy under the emotion taxonomy of HUANG (2015). The rest can be done in the same manner.

[2]http://t.qq.com/

4.2 Global Optimization with ILP

ILP optimization aims to refine the label result given the probability result. We design objective function and constraints to exploit the similarity between two emotion taxonomies. Like Roth and Yih (2004), we firstly define following assignment costs:

$$\begin{aligned} c_{Y_i} &= -log(p(r_{Y_i} = e_{Y_i})) + log(1 - p(r_{Y_i} = e_{Y_i})), \\ c_{H_i} &= -log(p(r_{H_i} = e_{H_i})) + log(1 - p(r_{H_i} = e_{H_i})), \\ & 1 \leq i \leq 7 \end{aligned} \tag{3}$$

where c_{Y_i} is the cost of t_i belonging to the ith emotion class under the taxonomy of YAO (2014), and c_{H_i} is the cost of t_i belonging to the ith emotion class under the taxonomy of HUANG (2015). For each sample t_i in testing set there can be two cost vectors C_Y and C_H, and two label vectors L_Y and L_H used on storing the refined labels of t_i:

$$C_Y = [c\_Y_1\ c\_Y_2\ ...\ c\_Y_7]^{\mathrm{T}}, \quad C_H = [c\_H_1\ c\_H_2\ ...\ c\_H_7]^{\mathrm{T}} \tag{4}$$

$$L_Y = [y_1\ y_2\ ...\ y_7], \quad L_H = [z_1\ z_2\ ...\ z_7] \tag{5}$$

where y_1 to y_7 indicate the emotion class of t_i under the taxonomy of YAO (2014), and z_1 to z_7 indicate that under the taxonomy of HUANG (2015). For instance, if the label vector L_Y=[0,1,0,0,0,0,1], it indicates that t_i is refined as $anger$ and $disgust$ under the emotion taxonomy of YAO (2014). The ILP optimization aims to acquire the refined emotion labels which are given by two label vectors.

ILP with Intra-corpus Constraints

We employ ILP with intra-corpus constraints to address the issue on annotation guideline. Note that we don't solely apply this type of constraints on the testing set of HUANG (2015) because it is considered to be single-labeled. On YAO (2014), the objective function can be defined as follows:

$$\begin{aligned} \min t &= |L_Y \times C_Y| \\ &= \sum_{i=1}^{7} (c\_Y_i y_i) \end{aligned} \tag{6}$$

Subject to:

$$y_i \in \{0,\ 1\}, \tag{7}$$

$$1 \leq \sum_{i=1}^{7} y_i \leq 2 \tag{8}$$

where formula (8) implies that one or two labels are chosen from the emotion taxonomy of YAO (2014) after optimization. The objective function above aims to minimize the product of cost vector and label vector. Furthermore, an additional constraint aiming to align the emotion classes between two taxonomies is defined as follows:

(C1) Co-occurrence constraint: We filter the emotion pairs with low co-occurrence frequency in YAO (2014). The filtered pairs all occur below 30 times in the corpus according to statistics. For instance, $happiness$ and $disgust$ rarely co-occur in the same instance.

$$\begin{aligned} & y_1 + y_2 \leq 1,\ y_1 + y_4 \leq 1,\ y_2 + y_4 \leq 1,\ y_2 + y_5 \leq 1, \\ & y_2 + y_6 \leq 1,\ y_4 + y_5 \leq 1,\ y_4 + y_6 \leq 1,\ y_4 + y_7 \leq 1 \end{aligned} \tag{9}$$

ILP with Extra-corpus Constraints

We leverage the similarity between two emotion taxonomies with extra-corpus constraints. Firstly, we add specific costs as follows:

$$\begin{aligned} & c\_align\_H_1 = c\_H_1 + c\_H_5,\ c\_align\_H_2 = c\_H_2 + c\_H_7, \\ & c\_align\_H_3 = c\_H_3 + c\_H_7,\ c\_align\_H_4 = c\_H_4 + c\_H_7, \\ & c\_align\_H_i = c\_H_i, 5 \leq i \leq 7 \end{aligned} \tag{10}$$

In formula (10), some costs in the original cost vector defined in formula (4) are added together. For example, c_H_1 and c_H_5 are added into $c\_align\_H_1$. It means that we align $happiness$ under the taxonomy of YAO (2014) to both joy and $positive$ under the taxonomy of HUANG (2015) together with the alignment constraint defined below. The cost vector C_H changes to:

$$C_H^{'} = [c\_align\_H_1\ c\_align\_H_2\ ...\ c\_align\_H_7]^{\mathrm{T}} \tag{11}$$

As a result, the objective function becomes to:

$$\begin{aligned} \min t &= |L_Y \times C_Y| + |L_H \times C_H^{'}| \\ &= \sum_{i=1}^{7}(c\_Y_i y_i + c\_align\_H_i z_i) \end{aligned} \tag{12}$$

Subject to:

$$y_i,\ z_i \in \{0,\ 1\}, \tag{13}$$

$$\sum_{i=1}^{7} y_i = 1, \tag{14}$$

$$\sum_{i=1}^{7} z_i = 1 \tag{15}$$

where formula (14) and (15) unify the number of possible labels on both taxonomies to one because we don't consider any intra-corpus constraints which are derived from the annotation guideline of YAO (2014) when extra-corpus is solely applied. The alignment constraint is defined as follows:

(C2) Alignment constraint: When a sample instance is categorized into a certain emotion e under one taxonomy, it can be categorized into an emotion $e^{'}$ which is same or similar to e under the other taxonomy. For instance, if an instance t is labeled as $disgust$ under the taxonomy of YAO (2014), it can be labeled as $negative$ under the taxonomy of HUANG (2015).

$$y_i = z_i,\ 1 \leq i \leq 7 \tag{16}$$

ILP with Two Types of Constraints

In this subsection, both intra-corpus and extra-corpus constrains are employed. The objective function is defined as follows:

$$\begin{aligned} \min t &= |L_Y \times C_Y| + |L_H \times C_H^{'}| \\ &= \sum_{i=1}^{7}(c\_Y_i y_i + c\_align\_H_i z_i) \end{aligned} \tag{17}$$

Subject to:

$$y_i,\ z_i \in \{0,\ 1\}, \tag{18}$$

$$1 \leq \sum_{i=1}^{7} y_i \leq 2, \tag{19}$$

$$\sum_{i=1}^{7} z_i = 1 \tag{20}$$

Moreover, constraint C1 and C2 are also employed to restrict the labels in both views of intra-corpus and extra-corpus. Additionally, we make a relaxation on the alignment constraint C2.

(C3) Relaxed Alignment constraint: We make a relaxation on C2 to allow more than one chosen label on YAO (2014). C2 makes the numbers of labels on two corpora be the same so that formula (19) becomes meaningless. We employ the following version to replace C2.

$$y_i \geq z_i,\ 1 \leq i \leq 7 \tag{21}$$

5 Experimentation

5.1 Experimental Setting

Features

Bag-of-words feature is adopted in training supervised emotion classifiers. Each instance is represented as a binary vector indicating the presence or absence of word unigrams.

Evaluation Metrics

We employ the widely used accuracy and F_1-measure on the multi-class-single-label emotion classification on HUANG (2015). On multi-class-multi-label emotion classification on YAO (2014), we employ two evaluation metrics to measure the performance. These metrics have been popularly used in multi-label classification problems (Godbole and Sarawagi, 2004).

- ***Accuracy:*** It gives an average degree of the similarity between the predicted and the ground truth label sets of all test examples:

$$Accuracy = \frac{1}{q}\sum_{i=1}^{q}\frac{|y_i \cap y_i^{'}|}{|y_i \cup y_i^{'}|} \tag{22}$$

  where q is the number of all test instances, $y_i^{'}$ is the estimated label and y_i is the true label.

- ***F_1-measure:*** It is the harmonic mean between precision and recall. It can be calculated from true positives, true negatives, false positives and false negatives based on the predictions and the corresponding actual values:

$$F_1 = \frac{1}{q}\sum_{i=1}^{q}\frac{|y_i \cap y_i^{'}|}{|y_i| + |y_i^{'}|} \tag{23}$$

5.2 Experimental Results with ILP Optimization

ILP with Intra-corpus constraints

In this experiment, intra-corpus constraints are applied for refining the predicting results. Note this experiment is only taken place on YAO (2014) in which an instance might have one or two labels. We experimentalize following methods for comparison:

- **Baseline:** We apply Maximum Entropy classifier with BOW feature as one baseline. Seven binary classifiers are trained for each emotion class. We balance the proportion of positive data and negative data for each classifier in order to achieve the best overall performance.

- **ILP with Intra-corpus Constraints:** ILP global optimization approach with defined intra-corpus constraints and objective function.

Table 2 shows the performance of ILP with intra-corpus constraints on the testing set of YAO (2014). According to the results, ILP approach with intra-corpus constraints overcomes the baseline with a 0.050 promotion in accuracy and a 0.013 promotion on F_1-measure, which demonstrates the effectiveness of proposed intra-corpus constraints.

	Accuracy	F_1
Baseline	0.375	0.243
ILP (Intra-corpus)	0.425	0.256

Table 2: Performance of ILP with intra-corpus constraints on YAO (2014)

ILP with Extra-corpus constraints

In this experiment, we apply extra-corpus constraints on ILP optimization to leverage the annotated data from two corpora. We experimentalize following methods for comparison:

- **Baseline:** Max Entropy classifier with BOW feature serves as baseline. In the experiment on YAO (2014), the baseline is same as the one used in the experiment with intra-corpus constraints. In the experiment on HUANG (2015), a 7-way classifier is trained for baseline. The proportion of training data for each emotion class follows its original proportion.
- **ILP with Intra-corpus Constraints:** ILP global optimization approach with defined extra-corpus constraints and objective function.

Table 3 shows the performance of ILP when extra-corpus constraints are utilized on both testing sets. Extra-corpus constraints improve the accuracy on YAO (2014), but the F_1-measure reduces. While on HUANG (2015), both accuracy and F_1-measure improve distinctly, proving the capability of extra-corpus constraints on corpus fusion to leverage the annotated data from other corpus.

	Accuracy	F_1
Baseline	0.375	0.243
ILP (Extra-corpus)	0.430	0.231

(a) On YAO (2014)

	Accuracy	F_1
Baseline	0.405	0.359
ILP (Extra-corpus)	0.431	0.386

(b) On HUANG (2015)

Table 3: Performance of ILP with extra-corpus constraints

ILP with Both Types of constraints

In this experiment, we apply both intra-corpus and extra-corpus constraints on ILP optimization to implement corpus fusion from both views. Following methods are experimentalized:

- **Baseline:** Same as those used in the experiment with extra-corpus constraints.
- **ILP with Intra-corpus Constraints:** ILP approach with only intra-corpus constraints. This method is only employed on YAO (2014).
- **ILP with Extra-corpus Constraints:** ILP approach with only extra-corpus constraints.
- **ILP with Both Types of Constraints:** ILP approach with both intra-corpus and extra-corpus constraints and defined objective function.

Table 4 shows the performance of ILP approach with both intra-corpus and extra-corpus constraints compared to baselines. From these tables, we can see that employing both constraints further improves accuracy and F_1-measure on YAO (2014). The joint use of both constraints avoids the decrease on F_1-measure when only extra-corpus constraints are applied. On HUANG (2015), ILP with both constraints slightly improves the accuracy compared to ILP with only extra-corpus, but the F_1-measure also decreases slightly. Intra-corpus constraints impact a little on HUANG (2015).

	Accuracy	F_1
Baseline	0.375	0.243
ILP (Intra-corpus)	0.425	0.256
ILP (Extra-corpus)	0.430	0.231
ILP (both)	0.440	0.261

(a) On YAO (2014)

	Accuracy	F_1
Baseline	0.405	0.359
ILP (Extra-corpus)	0.431	0.386
ILP (both)	0.435	0.382

(b) On HUANG (2015)

Table 4: Performance of ILP with both types of constraints

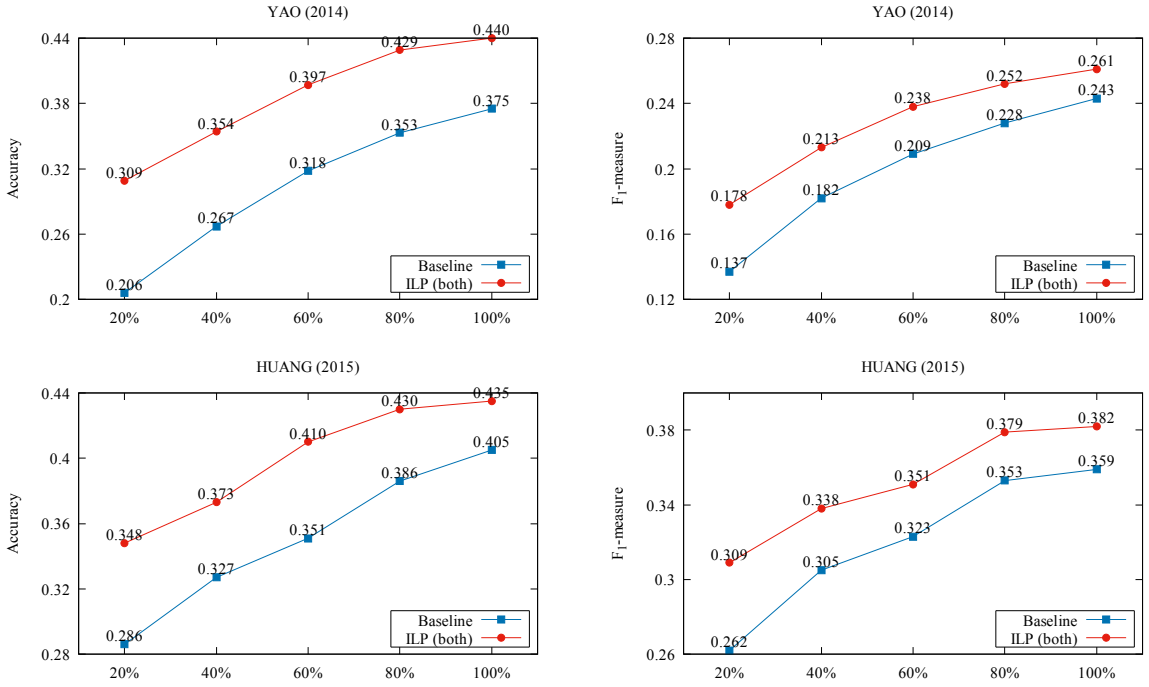


Figure 3: Performance of ILP with both types of constraints with different scales of training set

Figure 3 gives the performance of ILP with both constraints when different scales of training set are used. The improvement of ILP approach decreases with the increase of the scale of training set. It means that a highly performed baseline may reduce the space of promotion achieved by ILP optimization because the amount of error classified instances which can be refined decreases. Even so, ILP approach still improves the performance distinctly when the scale of training set is 100%.

6 Conclusion and Future Work

In this paper, we propose a corpus fusion approach to corpus fusion for emotion classification with ILP optimization. Specifically, we employ intra-task and extra-task constraints to better capture the similarity between two different emotion taxonomies and address the different annotation guidelines. Experiments demonstrate that ILP optimization improves the performance by using annotated data from other corpus, which has a different emotion taxonomy.

In our future work, we would like to seek better modification on ILP for further improvement. Moreover, we will try to adapt this approach to other NLP tasks where two or more corpora are available.

Acknowledgments

This research work has been partially supported by four NSFC grants, No.61273320, No.61331011, No.61375073, and No.61503386.

References

Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. In *Text, Speech and Dialogue, 10th International Conference, TSD 2007, Pilsen, Czech Republic, September 3-7, 2007, Proceedings*, pages 196–205.

Johan Bollen, Huina Mao, and Xiao-Jun Zeng. 2011. Twitter mood predicts the stock market. *J. Comput. Science*, 2(1):1–8.

Ying Chen, Sophia Yat Mei Lee, Shoushan Li, and Chu-Ren Huang. 2010. Emotion cause detection with linguistic constructions. In *COLING 2010, 23rd International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2010, Beijing, China*, pages 179–187.

Dipankar Das and Sivaji Bandyopadhyay. 2010. Sentence level emotion tagging on blog and news corpora. *J. Intelligent Systems*, 19(2):145–162.

Maros Galik and Stefan Rank. 2012. Modelling emotional trajectories of individuals in an online chat. In *Multiagent System Technologies - 10th German Conference, MATES 2012, Trier, Germany, October 10-12, 2012. Proceedings*, pages 96–105.

Jianfeng Gao, Andi Wu, Cheng-Ning Huang, Hongqiao Li, Xinsong Xia, and Hauwei Qin. 2004. Adaptive chinese word segmentation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain.*, pages 462–469.

Mishne Gilad. 2005. Experiments with mood classification in blog posts. In *Proceedings of ACM SIGIR 2005 workshop on stylistic analysis of text for information access*, volume 19, pages 321–327. Citeseer.

Shantanu Godbole and Sunita Sarawagi. 2004. Discriminative methods for multi-labeled classification. In *Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26-28, 2004, Proceedings*, pages 22–30.

Lei Huang, Shoushan Li, and Guodong Zhou. 2015. Emotion corpus construction on microblog text. In *Chinese Lexical Semantics - 16th Workshop, CLSW 2015, Beijing, China, May 9-11, 2015, Revised Selected Papers*, pages 204–212.

Wenbin Jiang, Liang Huang, and Qun Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and POS tagging - A case study. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 522–530.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M. Mohammad. 2014. Sentiment analysis of short informal texts. *J. Artif. Intell. Res. (JAIR)*, 50:723–762.

Shoushan Li, Lei Huang, Rong Wang, and Guodong Zhou. 2015. Sentence-level emotion classification with label and context dependence. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1045–1053.

Kevin Hsin-Yih Lin, Changhua Yang, and Hsin-Hsi Chen. 2007. What emotions do news articles trigger in their readers? In *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, pages 733–734.

Huanhuan Liu, Shoushan Li, Guodong Zhou, Chu-Ren Huang, and Peifeng Li. 2013. Joint modeling of news reader's and comment writer's emotions. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*, pages 511–515.

Saif M. Mohammad and Peter D. Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, CAAGET '10, pages 26–34, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta*.

Matthew Purver and Stuart Battersby. 2012. Experimenting with distant supervision for emotion classification. In *EACL 2012, 13th Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, April 23-27, 2012*, pages 482–491.

Changqin Quan and Fuji Ren. 2009. Construction of a blog emotion corpus for chinese emotional expression analysis. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1446–1454.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning, CoNLL 2004, Held in cooperation with HLT-NAACL 2004, Boston, Massachusetts, USA, May 6-7, 2004*, pages 1–8.

Weiwei Sun and Xiaojun Wan. 2012. Reducing approximation and estimation errors for chinese lexical processing with heterogeneous annotations. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, pages 232–241.

Duyu Tang, Bing Qin, Ting Liu, and Zhenghua Li. 2013. Learning sentence representation for emotion classification on microblogs. In *Natural Language Processing and Chinese Computing - Second CCF Conference, NLPCC 2013, Chongqing, China, November 15-19, 2013, Proceedings*, pages 212–223.

Zhongqing Wang, Sophia Yat Mei Lee, Shoushan Li, and Guodong Zhou. 2015. Emotion detection in code-switching texts via bilingual and sentimental information. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 763–768.

Shiyang Wen and Xiaojun Wan. 2014. Emotion classification in microblog texts using class sequential rules. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 187–193.

Ge Xu, Xinfan Meng, and Houfeng Wang. 2010. Build chinese emotion lexicons using A graph-based algorithm and multiple resources. In *COLING 2010, 23rd International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2010, Beijing, China*, pages 1209–1217.

Changhua Yang, Kevin Hsin-Yih Lin, and Hsin-Hsi Chen. 2007. Emotion classification using web blog corpora. In *2007 IEEE / WIC / ACM International Conference on Web Intelligence, WI 2007, 2-5 November 2007, Silicon Valley, CA, USA, Main Conference Proceedings*, pages 275–278.

Min Yang, Dingju Zhu, and Kam-Pui Chow. 2014. A topic model for building fine-grained domain-specific emotion lexicon. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 421–426.

Yuanlin Yao, Shuwei Wang, Ruifeng Xu, Bin Liu, Lin Gui, Qin Lu, and Xiaolong Wang. 2014. The construction of an emotion annotated corpus on micro blog text. *Journal of Chinese Information Processing*, 28(5):83–91.

Muhua Zhu, Jingbo Zhu, and Minghan Hu. 2011. Better automatic treebank conversion using A feature-based approach. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA - Short Papers*, pages 715–719.

Effective LSTMs for Target-Dependent Sentiment Classification

Duyu Tang, Bing Qin, Xiaocheng Feng, Ting Liu
Harbin Institute of Technology, Harbin, China
{dytang, qinb, xcfeng, tliu}@ir.hit.edu.cn

Abstract

Target-dependent sentiment classification remains a challenge: modeling the semantic relatedness of a target with its context words in a sentence. Different context words have different influences on determining the sentiment polarity of a sentence towards the target. Therefore, it is desirable to integrate the connections between target word and context words when building a learning system. In this paper, we develop two target dependent long short-term memory (LSTM) models, where target information is automatically taken into account. We evaluate our methods on a benchmark dataset from Twitter. Empirical results show that modeling sentence representation with standard LSTM does not perform well. Incorporating target information into LSTM can significantly boost the classification accuracy. The target-dependent LSTM models achieve state-of-the-art performances without using syntactic parser or external sentiment lexicons.[1]

1 Introduction

Sentiment analysis, also known as opinion mining (Pang and Lee, 2008; Liu, 2012), is a fundamental task in natural language processing and computational linguistics. Sentiment analysis is crucial to understanding user generated text in social networks or product reviews, and has drawn a lot of attentions from both industry and academic communities. In this paper, we focus on target-dependent sentiment classification (Jiang et al., 2011; Dong et al., 2014; Vo and Zhang, 2015), which is a fundamental and extensively studied task in the field of sentiment analysis. Given a sentence and a target mention, the task calls for inferring the sentiment polarity (e.g. positive, negative, neutral) of the sentence towards the target. For example, let us consider the sentence: "*I bought a new camera. The picture quality is amazing but the battery life is too short*". If the target string is *picture quality*, the expected sentiment polarity is "positive" as the sentence expresses a positive opinion towards *picture quality*. If we consider the target as *battery life*, the correct sentiment polarity should be "negative".

Target-dependent sentiment classification is typically regarded as a kind of text classification problem in literature. Majority of existing studies build sentiment classifiers with supervised machine learning approach, such as feature based Supported Vector Machine (Jiang et al., 2011) or neural network approaches (Dong et al., 2014; Vo and Zhang, 2015). Despite the effectiveness of these approaches, we argue that target-dependent sentiment classification remains a challenge: how to effectively model the semantic relatedness of a target word with its context words in a sentence. One straight forward way to address this problem is to manually design a set of target-dependent features, and integrate them into existing feature-based SVM. However, feature engineering is labor intensive and the "sparse" and "discrete" features are clumsy in encoding side information like target-context relatedness. In addition, a person asked to do this task will naturally "look at" parts of relevant context words which are helpful to determine the sentiment polarity of a sentence towards the target. These motivate us to develop a powerful neural network approach, which is capable of learning continuous features (representations) without feature engineering and meanwhile capturing the intricate relatedness between target and context words.

[1] Codes are publicly available at `http://ir.hit.edu.cn/~dytang`.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3298–3307, Osaka, Japan, December 11-17 2016.

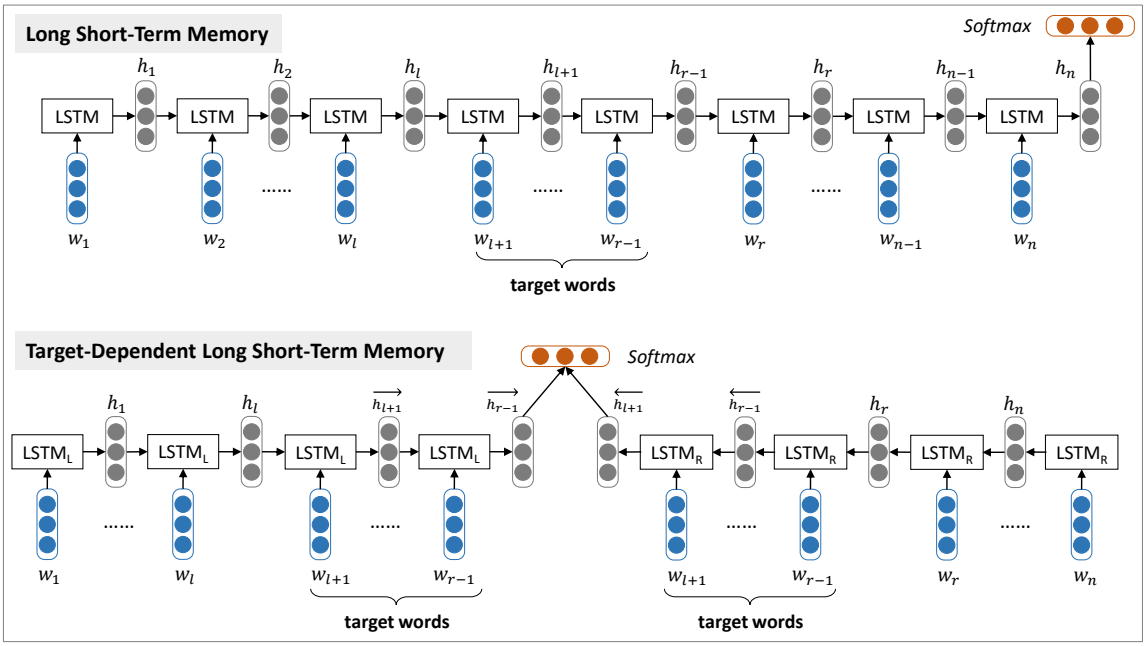


Figure 1: The basic long short-term memory (LSTM) approach and its target-dependent extension TD-LSTM for target-dependent sentiment classification. w stands for word in a sentence whose length is n, $\{w_{l+1}, w_{l+2}, ..., w_{r-1}\}$ are target words, $\{w_1, w_2, ..., w_l\}$ are preceding context words, $\{w_r, ..., w_{n-1}, w_n\}$ are following context words.

In this paper, we present neural network models to deal with target-dependent sentiment classification. The approach is an extension on long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) by incorporating target information. Such target-dependent LSTM approach models the relatedness of a target word with its context words, and selects the relevant parts of contexts to infer the sentiment polarity towards the target. The model could be trained in an end-to-end way with standard backpropagation, where the loss function is cross-entropy error of supervised sentiment classification.

We apply the neural model to target-dependent sentiment classification on a benchmark dataset (Dong et al., 2014). We compare with feature-based SVM (Jiang et al., 2011), adaptive recursive neural network (Dong et al., 2014) and lexicon-enhanced neural network (Vo and Zhang, 2015). Empirical results show that the proposed approach without using syntactic parser or external sentiment lexicon obtains state-of-the-art classification accuracy. In addition, we find that modeling sentence with standard LSTM does not perform well on this target-dependent task. Integrating target information into LSTM could significantly improve the classification accuracy.

2 The Approach

We describe the proposed approach for target-dependent sentiment classification in this section. We first present a basic long short-term memory (LSTM) approach, which models the semantic representation of a sentence without considering the target word being evaluated. Afterwards, we extend LSTM by considering the target word, obtaining the Target-Dependent Long Short-Term Memory (TD-LSTM) model. Finally, we extend TD-LSTM with target connection, where the semantic relatedness of target with its context words are incorporated.

2.1 Long Short-Term Memory (LSTM)

In this part, we describe a long short-term memory (**LSTM**) model for target-dependent sentiment classification. It is a basic version of our approach. In this setting, the target to be evaluated is ignored so that the task is considered in a target independent way.

We use LSTM as it is a state-of-the-art performer for semantic composition in the area of sentiment

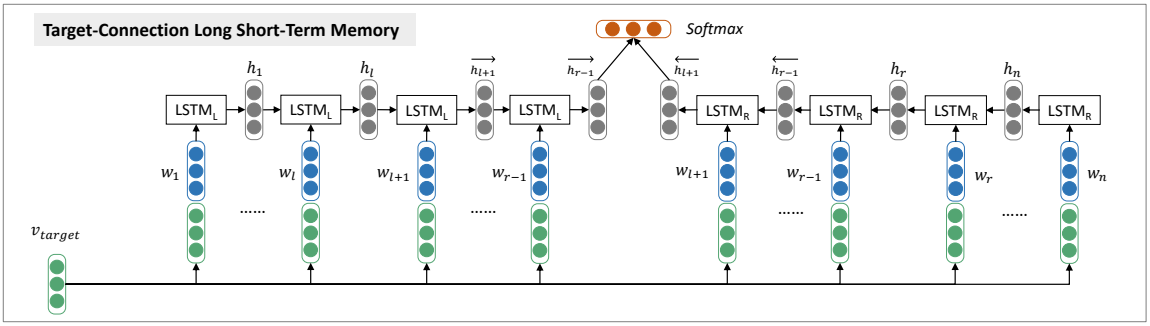


Figure 2: The target-connection long short-term memory (TC-LSTM) model for target-dependent sentiment classification, where w stands for word in a sentence whose length is n, $\{w_{l+1}, w_{l+2}, ..., w_{r-1}\}$ are target words, v_{target} is target representation, $\{w_1, w_2, ..., w_l\}$ are preceding context words, $\{w_r, ..., w_{n-1}, w_n\}$ are following context words.

analysis (Li et al., 2015a; Tang et al., 2015). It is capable of computing the representation of a longer expression (e.g. a sentence) from the representation of its children with multi levels of abstraction. The sentence representation can be naturally considered as the feature to predict the sentiment polarity of sentence.

Specifically, each word is represented as a low dimensional, continuous and real-valued vector, also known as word embedding (Bengio et al., 2003; Mikolov et al., 2013; Pennington et al., 2014; Tang et al., 2014). All the word vectors are stacked in a word embedding matrix $L_w \in \mathbb{R}^{d \times |V|}$, where d is the dimension of word vector and $|V|$ is vocabulary size. In this work, we pre-train the values of word vectors from text corpus with embedding learning algorithms (Pennington et al., 2014; Tang et al., 2014) to make better use of semantic and grammatical associations of words.

We use LSTM to compute the vector of a sentence from the vectors of words it contains, an illustration of the model is shown in Figure 1. LSTM is a kind of recurrent neural network (RNN), which is capable of mapping vectors of words with variable length to a fixed-length vector by recursively transforming current word vector w_t with the output vector of the previous step h_{t-1}. The transition function of standard RNN is a linear layer followed by a pointwise non-linear layer such as hyperbolic tangent function ($tanh$).

$$h_t = tanh(W \cdot [h_{t-1}; w_t] + b) \tag{1}$$

where $W \in \mathbb{R}^{d \times 2d}$, $b \in \mathbb{R}^d$, d is dimension of word vector. However, standard RNN suffers the problem of gradient vanishing or exploding (Bengio et al., 1994; Hochreiter and Schmidhuber, 1997), where gradients may grow or decay exponentially over long sequences. Many researchers use a more sophisticated and powerful LSTM cell as the transition function, so that long-distance semantic correlations in a sequence could be better modeled. Compared with standard RNN, LSTM cell contains three additional neural gates: an input gate, a forget gate and an output gate. These gates adaptively remember input vector, forget previous history and generate output vector (Hochreiter and Schmidhuber, 1997). LSTM cell is calculated as follows.

$$i_t = \sigma(W_i \cdot [h_{t-1}; w_t] + b_i) \tag{2}$$
$$f_t = \sigma(W_f \cdot [h_{t-1}; w_t] + b_f) \tag{3}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}; w_t] + b_o) \tag{4}$$
$$g_t = tanh(W_r \cdot [h_{t-1}; w_t] + b_r) \tag{5}$$
$$c_t = i_t \odot g_t + f_t \odot c_{t-1} \tag{6}$$
$$h_t = o_t \odot tanh(c_t) \tag{7}$$

where $\odot$ stands for element-wise multiplication, σ is sigmoid function, W_i, b_i, W_f, b_f, W_o, b_o are the parameters of input, forget and output gates.

After calculating the hidden vector of each position, we regard the last hidden vector as the sentence representation (Li et al., 2015a; Tang et al., 2015). We feed it to a linear layer whose output length is class number, and add a $softmax$ layer to output the probability of classifying the sentence as positive, negative or neutral. Softmax function is calculated as follows, where C is the number of sentiment categories.

$$softmax_i = \frac{exp(x_i)}{\sum_{i'=1}^{C} exp(x_{i'})} \tag{8}$$

2.2 Target-Dependent LSTM (TD-LSTM)

The aforementioned LSTM model solves target-dependent sentiment classification in a target-independent way. That is to say, the feature representation used for sentiment classification remains the same without considering the target words. Let us again take "*I bought a new camera. The picture quality is amazing but the battery life is too short*" as an example. The representations of this sentence with regard to *picture quality* and *battery life* are identical. This is evidently problematic as the sentiment polarity labels towards these two targets are different.

To take into account of the target information, we make a slight modification on the aforementioned LSTM model and introduce a target-dependent LSTM (**TD-LSTM**) in this subsection. The basic idea is to model the preceding and following contexts surrounding the target string, so that contexts in both directions could be used as feature representations for sentiment classification. We believe that capturing such target-dependent context information could improve the accuracy of target-dependent sentiment classification.

Specifically, we use two LSTM neural networks, a left one LSTM_L and a right one LSTM_R, to model the preceding and following contexts respectively. An illustration of the model is shown in Figure 1. The input of LSTM_L is the preceding contexts plus target string, and the input of LSTM_R is the following contexts plus target string. We run LSTM_L from left to right, and run LSTM_R from right to left. We favor this strategy as we believe that regarding target string as the last unit could better utilize the semantics of target string when using the composed representation for sentiment classification. Afterwards, we concatenate the last hidden vectors of LSTM_L and LSTM_R, and feed them to a $softmax$ layer to classify the sentiment polarity label. One could also try averaging or summing the last hidden vectors of LSTM_L and LSTM_R as alternatives.

2.3 Target-Connection LSTM (TC-LSTM)

Compared with LSTM model, target-dependent LSTM (TD-LSTM) could make better use of the target information. However, we think TD-LSTM is still not good enough because it does not capture the interactions between target word and its contexts. Furthermore, a person asked to do target-dependent sentiment classification will select the relevant context words which are helpful to determine the sentiment polarity of a sentence towards the target.

Based on the consideration mentioned above, we go one step further and develop a target-connection long short-term memory (**TC-LSTM**). This model extends TD-LSTM by incorporating an target connection component, which explicitly utilizes the connections between target word and each context word when composing the representation of a sentence.

An overview of TC-LSTM is illustrated in Figure 2. The input of TC-LSTM is a sentence consisting of n words $\{w_1, w_2, ...w_n\}$ and a target string t occurs in the sentence. We represent target t as $\{w_{l+1}, w_{l+2}...w_{r-1}\}$ because a target could be a word sequence of variable length, such as "*google*" or "*harry potter*". When processing a sentence, we split it into three components: target words, preceding context words and following context words. We obtain target vector v_{target} by averaging the vectors of words it contains, which has been proven to be simple and effective in representing named entities (Socher et al., 2013a; Sun et al., 2015). When compute the hidden vectors of preceding and following context words, we use two separate long short-term memory models, which are similar with the strategy used in TD-LSTM. The difference is that in TC-LSTM the input at each position is the concatenation of word embedding and target vector v_{target}, while in TD-LSTM the input at each position only includes

the embedding of current word. We believe that TC-LSTM could make better use of the connection between target and each context word when building the representation of a sentence.

2.4 Model Training

We train LSTM, TD-LSTM and TC-LSTM in an end-to-end way in a supervised learning framework. The loss function is the cross-entropy error of sentiment classification.

$$loss = -\sum_{s \in S} \sum_{c=1}^{C} P_c^g(s) \cdot log(P_c(s)) \tag{9}$$

where S is the training data, C is the number of sentiment categories, s means a sentence, $P_c(s)$ is the probability of predicting s as class c given by the $softmax$ layer, $P_c^g(s)$ indicates whether class c is the correct sentiment category, whose value is 1 or 0. We take the derivative of loss function through back-propagation with respect to all parameters, and update parameters with stochastic gradient descent.

3 Experiment

We apply the proposed method to target-dependent sentiment classification to evaluate its effectiveness. We describe experimental setting and empirical results in this section.

3.1 Experimental Settings

We conduct experiment in a supervised setting on a benchmark dataset (Dong et al., 2014). Each instance in the training/test set has a manually labeled sentiment polarity. Training set contains 6,248 sentences and test set has 692 sentences. The percentages of positive, negative and neutral in training and test sets are both 25%, 25%, 50%. We train the model on training set, and evaluate the performance on test set. Evaluation metrics are accuracy and macro-F1 score over positive, negative and neutral categories (Manning and Schütze, 1999; Jurafsky and Martin, 2000).

3.2 Comparison to Other Methods

We compare with several baseline methods, including:

In **SVM-indep**, SVM classifier is built with target-independent features, such as unigram, bigram, punctuations, emoticons, hashtags, the numbers of positive or negative words in General Inquirer sentiment lexicon. In **SVM-dep**, target-dependent features (Jiang et al., 2011) are also concatenated as the feature representation.

In **Recursive NN**, standard Recursive neural network is used for feature learning over a transfered target-dependent dependency tree (Dong et al., 2014). **AdaRNN-w/oE**, **AdaRNN-w/E** and **AdaRNN-comb** are different variations of adaptive recursive neural network (Dong et al., 2014), whose composition functions are adaptively selected according to the inputs.

In **Target-dep**, SVM classifier is built based on rich target-independent and target-dependent features (Vo and Zhang, 2015). In **Target-dep**$^+$, sentiment lexicon features are further incorporated.

The neural models developed in this paper are abbreviated as LSTM, TD-LSTM and TC-LSTM, which are described in the previous section. We use 100-dimensional Glove vectors learned from Twitter, randomize the parameters with uniform distribution $U(-0.003, 0.003)$, set the clipping threshold of softmax layer as 200 and set learning rate as 0.01.

Experimental results of baseline models and our methods are given in Table 1. Comparing between SVM-indep and SVM-dep, we can find that incorporating target information can improve the classification accuracy of a basic SVM classifier. AdaRNN performs better than feature based SVM by making use of dependency parsing information and tree-structured semantic composition. We can find that target-dep is a strong performer even without using lexicon features. It benefits from rich automatic features generated from word embeddings.

Among LSTM based models described in this paper, the basic LSTM approach performs worst. This is not surprising because this task requires understanding target-dependent text semantics, while the basic LSTM model does not capture any target information so that it predicts the same result for different

Method	Accuracy	Macro-F1
SVM-indep	0.627	0.602
SVM-dep	0.634	0.633
Recursive NN	0.630	0.628
AdaRNN-w/oE	0.649	0.644
AdaRNN-w/E	0.658	0.655
AdaRNN-comb	0.663	0.659
Target-dep	0.697	0.680
Target-dep$^+$	0.711	**0.699**
LSTM	0.665	0.647
TD-LSTM	0.708	0.690
TC-LSTM	**0.715**	0.695

Table 1: Comparison of different methods on target-dependent sentiment classification. Evaluation metrics are accuracy and macro-F1. Best scores are in bold.

targets in a sentence. TD-LSTM obtains a big improvement over LSTM when target signals are taken into consideration. This result demonstrates the importance of target information for target-dependent sentiment classification. By incorporating target-connection mechanism, TC-LSTM obtains the best performances and outperforms all baseline methods in term of classification accuracy.

Comparing between Target-dep$^+$ and Target-dep, we find that sentiment lexicon feature could further improve the classification accuracy. Our final model TC-LSTM without using sentiment lexicon information performs comparably with Target-dep$^+$. We believe that incorporation lexicon information in TC-LSTM could get further improvement. We leave this as a potential future work.

3.3 Effects of Word Embeddings

It is well accepted that a good word embedding is crucial to composing a powerful text representation at higher level. We therefore study the effects of different word embeddings on LSTM, TD-LSTM and TC-LSTM in this part. Since the benchmark dataset from (Dong et al., 2014) comes from Twitter, we compare between sentiment-specific word embedding (SSWE)[2] (Tang et al., 2014) and Glove vectors[3] (Pennington et al., 2014). All these word vectors are 50-dimensional and learned from Twitter. SSWE$_h$, SSWE$_r$ and SSWE$_u$ are different embedding learning algorithms introduced in (Tang et al., 2014). SSWE$_h$ and SSWE$_r$ learn word embeddings by only using sentiment of sentences. SSWE$_u$ takes into account of sentiment of sentences and contexts of words simultaneously.

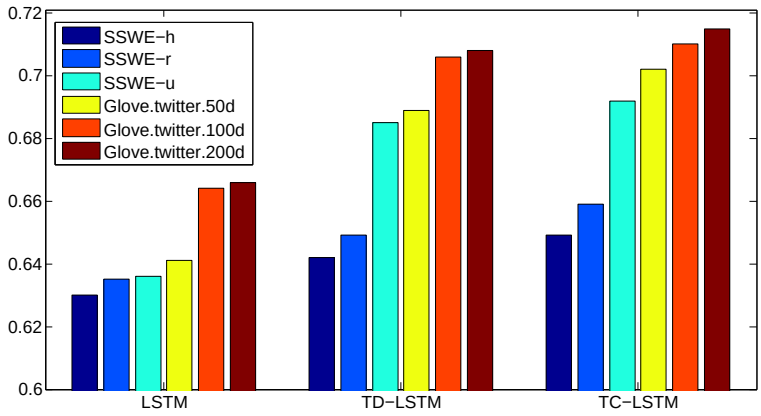


Figure 3: Classification accuracy of LSTM, TD-LSTM and TC-LSTM with different word embeddings. We compare between SSWE$_h$, SSWE$_r$, SSWE$_u$ and Glove vectors.

[2]SSWE vectors are publicly available at `http://ir.hit.edu.cn/~dytang`

[3]Glove vectors are publicly available at `http://nlp.stanford.edu/projects/glove/`

From Figure 3, we can find that $SSWE_h$ and $SSWE_r$ perform worse than $SSWE_u$, which is consistent with the results reported on target-independent sentiment classification of tweets (Tang et al., 2014). This shows the importance of context information for word embedding learning as both $SSWE_h$ and $SSWE_r$ do not encode any word contexts. Glove and $SSWE_u$ perform comparably, which indicates the importance of global context for estimating a good word representation. In addition, the target connection model TC-LSTM performs best when considering a specific word embedding.

	50dms	100dms	200dms
LSTM	27	95	329
LSTM-TD	20	93	274
LSTM-TC	65	280	1,165

Table 2: Time cost of each model with 50dms, 100dms and 200dms Glove vectors. Each value means how many seconds cost in each training iteration.

We compare between Glove vectors with different dimensions (50/100/200). Classification accuracy and time cost are given in Figure 3 and Table 2, respectively. We can find that 100-dimensional word vectors perform better than 50-dimensional word vectors, while 200-dimensional word vectors do not show significant improvements. Furthermore, TD-LSTM and LSTM have similar time cost, while TD-LSTM gets higher classification accuracy as target information is incorporated. TC-LSTM performs slightly better than TD-LSTM while at the cost of longer training time because the parameter number of TC-LSTM is larger.

3.4 Case Study

In this section, we explore to what extent the target-dependent LSTM models including TD-LSTM and TC-LSTM improve the performance of a basic LSTM model.

Example	gold	LSTM
*i hate my **ipod** look at my last tweet before the argh one that 's for you*	-1	0
*okay soooo ... ummmmm .... what is going on with **lindsay lohan**' s face? boring day at the office = perez and tomorrow overload. not good*	0	-1
*i heard ShannonBrown did his thing in the **lakers** game!! got ta love him*	0	1
*Hey **google**, thanks for all these great Labs features on Chromium, but how about " Create Application Shortcut"?!*	1	0

Table 3: Examples drawn from the test set whose polarity labels are incorrectly inferred by LSTM but correctly predicted by both TD-LSTM and TC-LSTM. For each example, target words are in **bold**, "gold" is the ground truth and "LSTM" means the predicted sentiment label from LSTM model.

In Table 3, we list some examples whose polarity labels are incorrectly inferred by LSTM but correctly predicted by both TD-LSTM and TC-LSTM. We observe that LSTM model prefers to assigning the polarity of the entire sentence while ignoring the target to be evaluated. TD-LSTM and TC-LSTM could take into account of target information to some extend. For example, in the 2nd example the opinion holder expresses a negative opinion about his work, but holds a neutral sentiment towards the target "*lindsay lohan*". In the last example, the whole sentence expresses a neutral sentiment while it holds a positive opinion towards "*google*".

We analyse the error cases that both TD-LSTM and TC-LSTM cannot well handle, and find that 85.4% of the misclassified examples relate to neutral category. The positive instances are rarely misclassified as negative, and vice versa. A example of errors is: "*freaky friday on television reminding me to think wtf happened to **lindsay lohan**, she was such a terrific actress , + my huge crush on haley hudson.*", which is incorrectly predicted as positive towards target "*indsay lohan*" in both TD-LSTM and TC-LSTM.

3.5 Discussion

In order to capture the semantic relatedness between target and context words, we extend TD-LSTM by adding a target connection component. One could also try other extensions to capture the connection between target and context words. For example, we also tried an attention-based LSTM model, which is inspired by the recent success of attention-based neural network in machine translation (Bahdanau et al., 2015) and document encoding (Li et al., 2015b). We implement the soft-attention mechanism (Bahdanau et al., 2015) to enhance TD-LSTM. We incorporate two attention layers for preceding LSTM and following LSTM, respectively. The output vector for each attention layer is the weighted average among hidden vectors of LSTM, where the weight of each hidden vector is calculated with a feedforward neural network. The outputs of preceding and following attention models are concatenated and fed to $softmax$ for sentiment classification. However, we cannot obtain better result with such an attention model. The accuracy of this attention model is slightly lower than the standard LSTM model (around 65%), which means that the attention component has a negative impact on the model. A potential reason might be that the attention based LSTM has larger number of parameters, which cannot be easily optimized with the small number of corpus.

4 Related Work

We briefly review existing studies on target-dependent sentiment classification and neural network approaches for sentiment classification in this section.

4.1 Target-Dependent Sentiment Classification

Target-dependent sentiment classification is typically regarded as a kind of text classification problem in literature. Therefore, standard text classification approach such as feature-based Supported Vector Machine (Pang et al., 2002; Jiang et al., 2011) can be naturally employed to build a sentiment classifier. Despite the effectiveness of feature engineering, it is labor intensive and unable to discover the discriminative or explanatory factors of data. To handle this problem, some recent studies (Dong et al., 2014; Vo and Zhang, 2015) use neural network methods and encode each sentence in continuous and low-dimensional vector space without feature engineering. Dong et al. (2014) transfer a dependency tree of a sentence into a target-specific recursive structure, and get higher level representation based on that structure. Vo and Zhang (2015) use rich features including sentiment-specific word embedding and sentiment lexicons. Different from previous studies, the LSTM models developed in this work are purely data-driven, and do not rely on dependency parsing results or external sentiment lexicons.

4.2 Neural Network for Sentiment Classification

Neural network approaches have shown promising results on many sentence/document-level sentiment classification (Socher et al., 2013b; Tang et al., 2015). The power of neural model lies in its ability in learning continuous text representation from data without any feature engineering. For sentence/document level sentiment classification, previous studies mostly have two steps. They first learn continuous word vector embeddings from data (Bengio et al., 2003; Mikolov et al., 2013; Pennington et al., 2014). Afterwards, semantic compositional approaches are used to compute the vector of a sentence/document from the vectors of its constituents based on the principle of compositionality (Frege, 1892). Representative compositional approaches to learn sentence representation include recursive neural networks (Socher et al., 2013b; Irsoy and Cardie, 2014), convolutional neural network (Kalchbrenner et al., 2014; Kim, 2014), long short-term memory (Li et al., 2015a) and tree-structured LSTM (Tai et al., 2015; Zhu et al., 2015). There also exists some studies focusing on learning continuous representation of documents (Le and Mikolov, 2014; Tang et al., 2015; Bhatia et al., 2015; Yang et al., 2016).

5 Conclusion

We develop target-specific long short term memory models for target-dependent sentiment classification. The approach captures the connection between target word and its contexts when generating the representation of a sentence. We train the model in an end-to-end way on a benchmark dataset, and show that

incorporating target information could boost the performance of a long short-term memory model. The target-dependent LSTM model obtains state-of-the-art classification accuracy.

Acknowledgements

We greatly thank Yaming Sun for tremendously helpful discussions. This work was supported by the National High Technology Development 863 Program of China (No. 2015AA015407), National Natural Science Foundation of China (No. 61632011 and No.61273321). According to the meaning given to this role by Harbin Institute of Technology, the contact author of this paper is Bing Qin.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

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

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from rst discourse parsing. In *EMNLP*, pages 2212–2218.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *ACL*, pages 49–54.

Gottlob Frege. 1892. On sense and reference. *Ludlow (1997)*, pages 563–584.

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

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *NIPS*, pages 2096–2104.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. *ACL*, 1:151–160.

Dan Jurafsky and James H Martin. 2000. *Speech & language processing*. Pearson Education India.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196.

Jiwei Li, Dan Jurafsky, and Eudard Hovy. 2015a. When are tree structures necessary for deep learning of representations? *EMNLP*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015b. A hierarchical neural autoencoder for paragraphs and documents. *ACL*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Christopher D Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *EMNLP*, pages 79–86.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Y Ng. 2013a. Reasoning with neural tensor networks for knowledge base completion. *NIPS*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. *IJCAI*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*, pages 1555–1565.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. *EMNLP*, pages 1422–1432.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. *IJCAI*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over tree structures. *ICML*.

Towards assessing depth of argumentation

Manfred Stede
Applied Computational Linguistics
UFS Cognitive Sciences
University of Potsdam / Germany
stede@uni-potsdam.de

Abstract

For analyzing argumentative text, we propose to study the 'depth' of argumentation as one important component, which we distinguish from argument quality. In a pilot study with German newspaper commentary texts, we asked students to rate the degree of argumentativeness, and then looked for correlations with features of the annotated argumentation structure and the rhetorical structure (in terms of RST). The results indicate that the human judgements correlate with our operationalization of depth and with certain structural features of RST trees.

1 Introduction

In recent years, *argumentation mining* has emerged as a discipline that aims to identify portions of argumentation in text and moreover – in some of the work – to relate them to one another, so that the full structure of arguments is being represented. A natural application for this new field is the mining of arguments in customer reviews (e.g., (Villalba and Saint-Dizier, 2012)), where the goal is to move beyond finding positive and negative statements by also finding the reasons that customers provide for their judgements. But the general task can be applied in many other domains as well. For example, the early research of Mochales Palau and Moens (2009) sought to identify premises and conclusions in legal text; another early work on finding theses in student essays (Burstein et al., 2003) has recently been extended to detecting more argumentative structure in such essays (Stab and Gurevych, 2014), (Nguyen and Litman, 2015); other genres that are being tackled include online dialogue (Oraby et al., 2015), multi-party dialogue (Budzynska et al., 2013) and scientific papers (Kirschner et al., 2015).

Despite the manifold recent activities, the field is still young, and a number of basic issues still require attention. This concerns the design of annotation schemes and possibly finding a consensus on them, but also the more fundamental problem of defining the notion of, and identifying instances of, *argumentative text*. While it is tempting to assume that earlier work on text-level subjectivity classification can be used to determine whether some (portion of) text is argumentative, we argue below that this is generally not the case. Going further, we posit that 'argumentativeness' is a matter of degree, and that determining this degree should also be considered a subtask of argumentation mining. We use the genre of newspaper editorials to conduct a small study where human raters are asked to assess how 'argumentative' different texts are, or in other words, what their 'argumentative depth' is. We will defend the introduction of this term by demonstrating that depth is to be distinguished from argument *quality*, which is already an object of study in the community.

Our next task then is to explain how such judgements arise, i.e., to find features that differentiate texts of different argumentative depths. We show that some simple surface features are not sufficient, and then turn to the underlying pragmatics, as it is – to some extent – captured in analyses according to Rhetorical Structure Theory (RST; (Mann and Thompson, 1988)). It was designed as a tool to make the reasons for a text's coherence explicit, and the specific notion of coherence relation used in RST (as opposed to other approaches such as the Penn Discourse Treebank (Prasad et al., 2008) or Segmented Discourse Representation Theory (Asher and Lascarides, 2003)) is decidedly *intentional*. This makes RST a good

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3308–3317, Osaka, Japan, December 11-17 2016.

candidate for checking whether its text representations are able to predict argumentative depth. The corpus we are using in our study has already been annotated with RST trees in earlier work, so we can use this data when looking for correlations with argumentation structure.

The paper is structured as follows. Section 2 introduces the notion of 'argumentativeness' and compares it to that of 'subjectivity', and introduces our approach to characterising the depth of argumentation. Section 3 describes the newspaper corpus we are using, and Section 4 presents the pilot study on annotating and measuring depth. Finally, Section 5 provides a summary and an outlook on future work.

2 Subjectivity versus Argumentation

To define our terminology, we start from the notion of 'text type' (Werlich, 1975) or 'discourse mode' (Smith, 2003), which posits that a passage of text can be of one of the following types: narrative, descriptive, instructive, expository, or argumentative. These classes refer to the central purpose of the passage, and – as especially Smith has shown – they correlate with certain linguistic features (aspect, Aktionsart, tense progression, etc.). The function of argumentative text, in general, is to influence the beliefs or attitudes of the readers by assembling a constellation of propositions that serve to defend a particular standpoint on a controversial issue (cf. (van Eemeren et al., 1996)).

'Subjectivity' in linguistics encompasses several different phenomena, which all serve to distinguish it from the 'objective' statement, but can do so in quite different ways, such as making speculations on future events, reporting on one's feelings, attributing content to third parties or to hearsay, or evaluating some entity in terms of valence (positive/negative). The latter is the notion that is mostly used in Computational Linguistics, and it is the central target of most work on subjectivity classification. This task can be applied on sentence or text level, and it has been done with bag-of-words models, PoS tags, or linguistic features that tend to be more roubst against domain changes (Petrenz and Webber, 2011), (Krüger et al., to appear). Importantly, subjectivity is orthogonal to the text types mentioned above: Texts of all types may be predominantly subjective, and most may not (a likely exception being the argumentative type).

What has – to the best of our knowledge – not been addressed in depth so far is the distinction between the tasks of classifying subjectivity (versus objectivity) on the one hand and argumentativeness (versus non-argumentativeness) on the other. To see why this is necessary, consider the following examples from the Brexit reader-discussion website of the *Irish Times*:[1]

(1) Sir, Born in London of an Irish mother and English father during the second World War and living in Dublin since 1968, I endeavoured to encourage my English relatives not to exit.
It looks like a case for returning my British passport, if Ireland will look favourably on my application for a replacement. Yours, etc.

(2) Sir, A black day for Europe. The age of populism and demagoguery is well and truly upon us, from Donald Trump to Ukip. The centre is under threat and is buckling. We must passionately embrace the centre ground and with it the EU. The EU is undoubtedly flawed but it has guaranteed peace in Europe for 70 years and made the idea of war laughable. It has united countries that for millennia were enemies and all it asks, as per Article 2 of the European Union Treaty, is respect for human dignity, freedom, democracy, equality, among other lofty goals. (...) Yours, etc.

Both are clearly subjective and both express an opinion (incidentally the same) on the issue, but only (2) states a thesis/claim and provides reasons in support (and even grants a possible objection), thus clearly qualifies as presenting an argument. (1), in contrast, gives a very short personal narrative followed by a description of the speaker's attitude.

We find the same difference in product reviews. Consider these camera reviews from an Amazon website:[2]

[1] http://www.irishtimes.com/opinion/letters/brexit-the-readers-have-their-say-1.2698710

[2] https://www.amazon.co.uk/product-reviews/B00IK01PJC/ref=cm_cr_arp_d_paging_btm_51?ie=UTF8

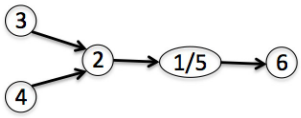


Figure 1: Argumentation structure of example (4)

(3) Camera completely broke whilst on a once in a lifetime holiday and I was nowhere where I could buy a new one. Absolutely gutted as it was first time using camera!

(4) [You'll be lucky if you manage to focus this camera.]$_1$ [Bought to take pictures of jewellery for online sale, the macro function which boasts as having a 5cm macro ability is awful.]$_2$ [For one it is an automatic macro which means that the camera should automatically detect whether the picture requires macro or not - it does not.]$_3$ [For two it should focus on the item in question at even 10cm distance - it does not.]$_4$ [You end up with blurry edges and unfocused pictures.]$_5$ [I was expecting SO much more from an Amazon best seller.]$_6$

Like in the Brexit examples, we sense basically the same opinions (which here resulted in the same low star rating), but the first user gives merely a narration while the second argues (in a fairly complex way with recursive support structure). We added brackets and indices to the sentences so that we can refer to them in our analysis of the argument structure, which was built using the scheme of Peldszus and Stede (2013) and is shown in Figure 1.[3] A brief description of this scheme will be provided in the next section; for now, notice that text segments can be related by SUPPORT and ATTACK relations, and collectively form a tree whose root is the central claim of the text. In our example, all relations are SUPPORT, and we omit the labels here. The node '1/5' indicates that these units in the text convey essentially the same message. Intuitively, to check the representation, a pair of segments in a support relation can be paraphrased with a *why*-link: "I was expecting much more." - "Why?" - "Pictures are unfocused." - "Why?" - "The macro function is awful" - "Why'?" - "Camera doesn't detect macro mode, and camera doesn't focus at close distance."

Using the example, we can begin to make the notion of argumentative depth explicit. For one thing, the measure should contain the proportion of the number of tokens that are taking part in the argumentation analysis: This reflects the amount of non-argumentative information in the text. Furthermore, the measure should reflect the complexity of the tree structure – a flat tree where a claim is supported by a few independent backings is intuitively (and technically) less 'deep' than a structure involving recursion. To measure this, we here simply calculate the average length of the paths from each leaf node to the root. (More elaborate measures could distinguish between the branching factor at the root node and elsewhere, etc.) These two values then constitute our description of argumentative depth – for the time being, we refrain from proposing a way of combining them into a single value. For example (4) and its structure, the pair is: 1.0 / 3.0.

Depth of argumentation, we wish to emphasize, is not the same as quality of argumentation. A writer can build up a fairly complex structure, yet the individual claim/support relations may be weak or even flaky. Here is a (constructed) short camera review to illustrate the point:

(5) I bought this camera yesterday, and I really like it. So even if some people say it's a bad product, you can go ahead and buy it, too. Because it is simply wonderful!

Quality, in this view, is a feature of the reasoning schemes underlying the argumentation, whereas depth is a purely structural measure of the argument as it is presented by the author in the text.

[3]For reasons of space, we cannot discuss all aspects of the analysis here and omit, *inter alia*, a justification of defining the boundaries of segments (argumentative discourse units). See (Peldszus and Stede, 2013).

3 Argumentation in Newspapers: Our Corpus

Our overall goal is to determine whether argumentative depth can be (i) reliably annotated by humans, and (ii) measured automatically. To this end, we wish to correlate it with different layers of linguistic analysis. In order to start this, we favour a corpus of argumentative text that is annotated with argument structure and also with additional layers. The German 'Potsdam Commentary Corpus' (Stede and Neumann, 2014) fulfills this criterion, as it comes with sentence syntax, connectives and their arguments, coreference, and rhetorical structure. We will concentrate here on rhetorical structure; note that our findings do not depend on any specifics of German; a corpus in any other language could be used in the same way.

3.1 The genre of commentary

Most newspapers offer 'opinion' pages with editorials that comment upon current affairs. The specifics of this genre depend to some extent on the different national traditions, but we expect that classifications like that of Schneider and Raue (1996) for German commentary do largely apply to the press in some other countries as well. The authors propose six categories, which include the 'opinion article' (a long piece that slowly builds up a position and encourages readers to reflect), the 'pro and contra' (a crisp and traceable argumentation in favour of a position), the 'on the one/the other hand' (a piece that looks at both sides and remains undecided), and the 'pamphlet' (a strong opinion with little real argumentation). This is, of course, an analytical framework only – actual articles in papers are usually not labelled in this way.

3.2 Data

The Potsdam Commentary Corpus is drawn from two different newspapers: One part is 'pro and contra' (which is in fact their headline) pieces from *Tagesspiegel*; the other is a mix of articles from the opinion page of the local newspaper *Märkische Allgemeine Zeitung* (henceforth: MAZ). All texts are about 10-12 sentences long. The pro/contra articles always come as a pair: One is in favour, one is against the issue under discussion, which may be of local (e.g., Should Berlin build more refugee centers), national (e.g., Should there be mandatory fees for public radio and TV in Germany), or global (e.g., Should we push for a stronger proposal on fighting climate change) relevance. The reader may then decide with whom to side. Given this setting, plus the brevity, we can safely assume a high degree of argumentative depth.

The opinion articles from MAZ, on the other hand, are quite diverse in their argumentative nature: Some merely re-state a piece of news and add some subjective evaluation to it; some try to explain why some event happened; some indulge in a piece of trivia; some take a clear position on a political question and defend it, like a pro/contra text does. Therefore, we can expect to find texts of rather different argumentative depths here, similar to the differences between example texts (1) and (2), and between (3) and (4) above.

For the experiments described below, we selected 15 texts for feature development, and 14 texts for testing; 80% come from MAZ, the rest are 'pro and contra' texts. Both sets were not drawn randomly: They should reflect the spectrum from low to high degree of argumentative depth in roughly equal proportion; these decisions were made by the author (and subject to confirmation by our annotators).

3.3 Annotation layers

For the purposes of this paper, two of the layers provided in the corpus are relevant. The (German) annotation guidelines for these and other layers are freely available.[4]

Rhetorical structure: We are using the RST annotations that were produced for version 2 of the Potsdam Commentary Corpus in 2014. They are largely conforming to the proposals by Mann and Thompson (1988), but the annotation guidelines make some minor modifications. As a preperatory step, the text is first segmented into a sequence of Elementary Discourse Units (EDUs); our guidelines explain which types of clauses constitute such an EDU and which do not. A central concept for our analysis below is

[4] https://publishup.uni-potsdam.de/opus4-ubp/frontdoor/index/index/docId/8276

nuclearity: Most coherence relations in RST distinguish between a central text span (nucleus) and a less important one (satellite); a few treat both (or more) spans as equally important, these are called 'multinuclear'. The result of an RST analysis is a tree structure that joins adjacent EDUs, and then recursively the larger spans.

Using the nuclearity information at each level of that tree, one can read off the central units of the text: those that can be reached from the root of the tree by following only 'nucleus' edges towards the leafs (EDUs). In the second tree of Figure 2 below, EDU 8 is the single central unit; in the the first tree, it is the set of EDUs reached by following all four Joint segments downward.[5] (For an introduction to RST, see also (Taboada and Mann, 2006)).

Argumentation structure: The 'pro/contra' texts in the corpus are already annotated for argumentation, following the scheme proposed by Peldszus and Stede (2013), which is codified in our guidelines. For the MAZ texts, we produced these annotations now, following the same guidelines.

Like with RST, an argumentation structure is also based on a segmentation into elementary units, but these may be larger ('argumentative discourse units' or ADUs). The analysis is a tree whose root represents the 'central claim' unit of the text. The other nodes correspond to ADUs, and the edges to SUPPORT or ATTACK relations, where the latter distinguish between REBUT (challenging the validity of the assertion in the ADU) and UNDERCUT (challenging not the validity of individual ADUs, but the supposition of a SUPPORT relation between two). In contrast to an RST analysis, there is no constraint that only adjacent segments may be conjoined by a relation. Furthermore, the argumentation structure need *not* span the text completely: Parts of it can be deemed irrelevant for the argument made, which is important for our notion of argumentative depth as introduced above.

4 Experimental study

4.1 Features

Simple features. Depth of argumentation is unlikely to be detectable with straightforward surface (or 'near-surface') features. Nonetheless, we tested two simple features that have been used in subjectivity classification (see above) and that might be relevant here: The presence of modal verbs and of 'argumentative' connectives: those that signal a contrastive or causal (in the wide sense) relation.

Argumentation structure: depth measure. The Peldszus/Stede annotation scheme had so far been applied by its authors to pro/contra commentaries and to very short 'microtexts' (Peldszus and Stede, 2016a). Our present annotation is thus its first application to texts which are not as easy to handle: Portions of text might be irrelevant for the argument, and both the central claim and the attachment points of relations can be hard to identify. We will test if our depth measure reflects differences in 'argumentativeness' as perceived by readers.

Rhetorical structure. In the literature there is an ongoing and so far not quite conclusive discussion on the relationship between RST analysis and argumentation analysis (e.g., (Azar, 1999), (Peldszus and Stede, 2013), (Green, 2015), (Peldszus and Stede, 2016b)). Clearly, as mentioned in the beginning, the design of the relations by Mann and Thompson (1988) suggests that the theory be especially amenable to argumentative text. Thus it can be expected that a structural analysis in terms of semantic and pragmatic relations, equipped with nuclearity, might capture what we are characterizing as depth of argumentation here. The first feature that comes to mind is the distribution of semantic versus pragmatic relations (as they have been categorized by the RST and the guideline authors); in particular, we treat Antithesis, Concession, Evaluation, Evidence, Justify, and Reason as 'pragmatic' relations. In the category 'semantic', we include all others except the 'textual' relations Joint, List, Preparation, Conjunction. Here, the hypothesis is "The more pragmatic relations, the more depth" (and conversely for the semantic relations). Then, we use a more elaborate feature to examine some structural properties of the RST tree. The data

[5] The notation in the figure uses straight lines for nucleus links and curved arrows for satellite links. Notice that in the upper tree, EDUs 4-8 have been collapsed in order to fit the tree onto the page. In both trees, segment 1 is missing, because the headline of the text is left out of the analysis.

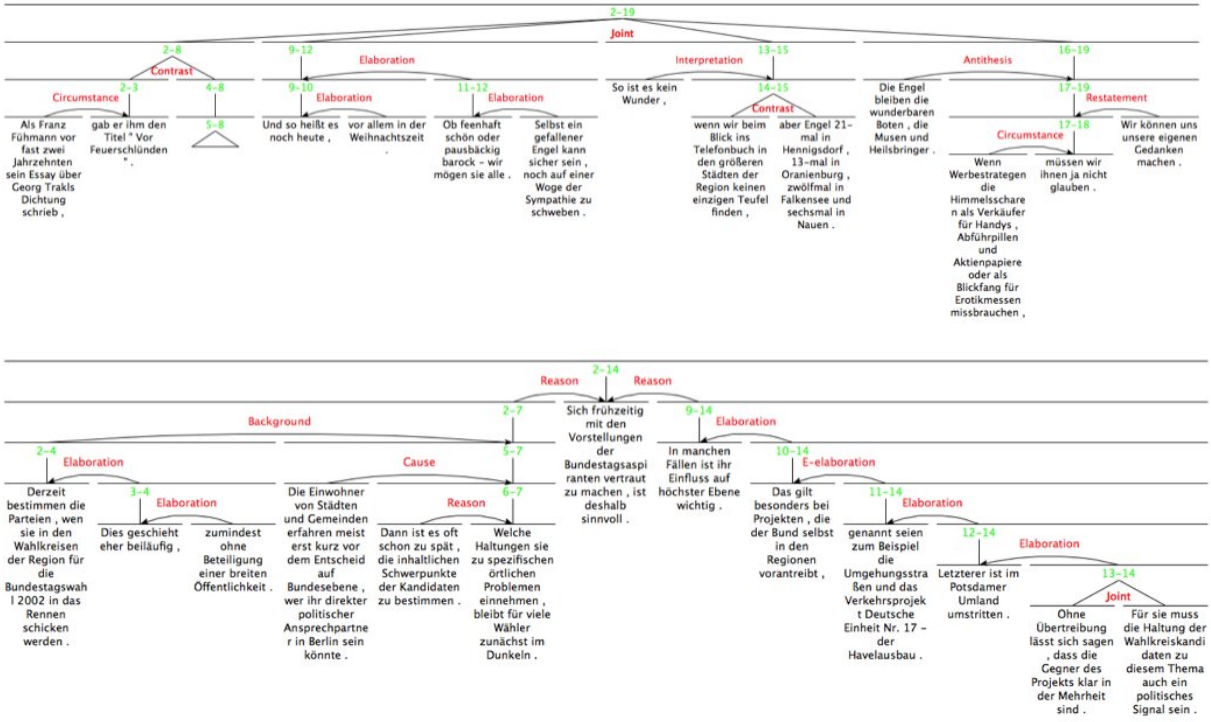


Figure 2: RST trees with very different nuclearity mass distribution

in the development set suggests a tendency for 'deeply argumentative' texts to have a clearly-identifiable central nucleus, whereas other texts can have multinuclear relations toward the top of the tree, which signifies that several points are perceived as being equally important to the author of the text. For illustration, consider Figure 2, which shows the structure of two texts from our development set. We thus seek to measure the distribution of the 'nuclearity mass' (NM) across the tree. We are not aware of such a measure having been described in the literature, and here suggest two variants. Call the number of satellite links on a path from leaf node to root the 'sat value' of that leaf node.

- NM1: The proportion of leaf nodes with a sat value of 0 or 1 (i.e., 'central' units).
- NM2: Let l_i be the length of the path (irrespective of the nuc/sat distinction) from leaf i to the root. Then, NM2 is the sum of those l_i where i has a sat value of 0 or 1, divided by the sum of l_i for all i of the tree.

While NM1 considers just the number of central nodes, NM2 also takes their distance from the root into consideration. For the upper tree in Figure 2, the two values are 0.93 and 0.88; for the lower tree they are 0.23 and 0.11.

4.2 Collecting judgements on argumentative depth

Having obtained hypotheses from the development set, we proceeded to test them on our second set (14 texts). We had students rate these texts for argumentativeness and then split them into two groups of 'low' and 'high' argumentative depth.

It is not trivial to elicit a judgement on 'argumentative depth', since this is obviously not an established concept. We were interested in intuitive judgements of readers that were not influenced by attempts on (mentally) building an explicit representation of the argumentation. For this reason, we worked with first-year students that had not received any training in argumentation analysis. Our procedure was to

Rank	Value	Source	Expect.	Rank	Value	Source	Expect.
1	4.8	ProCon	high	9	2.8	MAZ	high
2	4.2	MAZ	high	10	2.8	MAZ	medium
3	3.8	MAZ	high	11	2.4	MAZ	low
4	3.8	MAZ	high	12	2.2	MAZ	low
5	3.4	ProCon	high	13	2.0	MAZ	low
6	3.4	MAZ	medium	14	1.8	MAZ	low
7	3.4	MAZ	low	15	1.0	News	low
8	3.2	ProCon	high	16	1.0	News	low

Table 1: Text ranking obtained via students' answers to questionnaire

present them with a questionnaire posing the following questions, where the answer was to be given as a position on a 1..5 Likert scale:[6]

- Is the text a news report or an opinion text?
- If the text is a news report, is the reporting clear or unclear?
- If the text is an opinion text, is the position of the author clearly represented?
- If the text is an opinion text, does the author provide clearly recognizable arguments for his or her position?

The first question was meant as a first proxy for the amount of 'opinion' in the texts and in particular to identify non-commentaries. To make it work properly, we added filler items to the texts: two news reports taken also from the *Tagesspiegel*, which were of roughly the same length as the commentaries. The second question then merely served to balance the question set – the answers were not used in determining the ranking, which was based only on the answers to the last two questions. Notice that the last one does not refer to structural depth but merely to the presence of arguments and their clarity; this means that the students did not rate 'depth' directly. (In a future version of the study we plan to add a question to that effect.)

Every student judged three texts, which were mixed according to our expectations on argumentative depth. Most, but not all, students saw one of the news texts (filler) in their set. 30 students participated, so we obtained 90 judgements in total, which amounts to 5 per text (14 commentaries plus 2 news), over which we calculated averages.

4.3 Results

We translated the ratings into a ranking of the texts, ranging from low to high opinion/argument. Table 1 shows for each text its rank and its accumulated value on the Likert scale, its source, and the depth category we had originally expected for it. The first good news is that the news reports (fillers) were easily identified – they are thus discarded from the subsequent evaluation. Then, using the arithmetic mean, we determined the cutoff to form the two groups of eight 'high' and six 'low' argumentative texts (shown in the two halves of the table). Below we call the groups H and L.

As the table shows, our expectations on ranking are mostly confirmed, with two exceptions at rank 8 and 9, which we had estimated higher up.

Simple features. No difference between the two groups can be found for the frequency of modal verbs and of contrastive/causal connectives (neither on the development nor on the test set).[7]

[6] In addition, we asked whether the text was considered generally 'understandable' so that we could discard judgements where the student apparently had not understood the text well enough. This happened only in two instances.

[7] As pointed out by a reviewer, simple features might be indicative of just the first component of our depth measure (the proportion of argumentative tokens); testing this is left for a follow-up study with a larger corpus.

	pragmatic relations	semantic relations	NM1	NM2
group H	d: 43.17% / t: 38.3%	d: 35.0% / t: 45.8%	d: 37.29% / t: 40.79%	d: 28.1% / t: 33.82%
group L	d: 45.13% / t: 28.03%	d: 32.5% / t: 55.88%	d: 49.75% / t: 55.99%	d: 41.22% / t: 46.63%

Table 2: Results for RST features. d = dev set / t = test set

Argumentation structure. On the test set, the average proportion of tokens participating in the argumentation is 0.81 in group H and 0.31 in group L – a very clear difference. The average path lengths in the argumentation tree also confirm the expectation: In group H it is 1.59, and in group L only 1.09. It seems that our depth measure can capture the difference between the groups.

Rhetorical structure. Table 2 summarizes the results for the various RST features. For one thing, it shows that the results on our (more or less equally small) development and test sets can differ considerably, which suggests that more data is needed to obtain more reliable results. However, other tentative conclusions can be drawn: The figures indicate that the distribution of semantic/pragmatic relations is not a good feature for depth of argumentation, whereas both ways of measuring the distribution of nuclearity mass yield clear differences.

5 Summary

Constraining ourselves to monologue text, we suggested assessing the 'depth' of argumentation as a subtask of argumentation mining, when it aims at reconstructing the full structure of the argument. Our measure combines (i) the proportion of the text that contributes to the argument and (ii) a simple account of the complexity of the tree representing the argument: the average length of the paths in that tree. We hinted at the possibility of using more sophisticated variants of (ii). Importantly, we pointed out that depth is a purely structural measure, which is to be distinguished from argument 'quality' as it is being addressed in related work.

To test our measure, we conducted a pilot study with a small corpus of annotated newspaper commentary texts, supplemented by filler items (news reports). Using a set of questions, we obtained judgements on opinionatedness and argumentativity from human raters, which led to ranking the texts and then splitting them into two groups of high and low argumentative depth, respectively. We found that the depth measure can serve to differentiate the groups.

In addition, we were interested to see whether the human judgements correlate with simple surface features and with aspects of the rhetorical structure of the texts. As for the former, we tested modal verbs and connectives (standardly used in subjectivity classification) but could not find an effect. Regarding rhetorical structure, interestingly, the distribution of 'semantic' versus 'pragmatic' relations (as defined in RST) turns out not to differentiate between the groups. However, two measures of distributing the 'nuclearity mass', which we proposed here, do reflect the distinction. We conjecture that these measures can account for the clarity and focus of the argumentation. One next step is to experiment with variants of the RST measures, trying to combine relational and structural features.

To obtain more evidence for the validity of the results, both on the depth measure and the correlations with rhetorical structure, we plan to supplement the study with another experiment on a larger number of texts from the commentary corpus. Given a broader set of data, proper statistical measures can be computed for annotator agreement and the correlations. Furthermore, as noted earlier, the questionnaire will be extended in such a way that raters make judgements on depth more explicitly; in the pilot, we only asked for an intuitive judgement of the presence of clearly recognizable arguments (irrespective of their potential recursive structure).

Afterwards, it will be interesting to apply the approach to other genres, such as student essays, where we expect to find similar differences of argumentative depth. For the genre we studied here, newspaper commentary, we see our work as a contribution to identifying sub-genres (different types of commentary) and explaing the differences between them.

Acknowledgements

Many thanks to the anonymous reviewers for their constructive comments und suggestions on improving the paper.

References

Nicholas Asher and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press, Cambridge.

M. Azar. 1999. Argumentative text as rhetorical structure: An application of Rhetorical Structure Theory. *Argumentation*, 13:97–114.

K. Budzynska, M. Janier, C. Reed, and P. Saint-Dizier. 2013. Towards extraction of dialogical arguments. In *Working Notes of the 13th Workshop on Computational Models of Natural Argument (CMNA 2013)*, Rome.

J. Burstein, D. Marcu, and K. Knight. 2003. Finding the WRITE stuff: automatic identification of discourse structure in student essays. *IEEE Intelligent Systems*, 18(1):32–39.

Nancy Green. 2015. Annotating evidence-based argumentation in biomedical text. In *Proceedings of the IEEE Workshop on Biomedical and Health Informatics*.

Christian Kirschner, Judith Eckle-Kohler, and Iryna Gurevych. 2015. Linking the thoughts: Analysis of argumentation structures in scientific publications. In *Proceedings of the 2015 NAACL-HLT Conference*. Association for Computational Linguistics, June.

K. Krüger, A. Lukowiak, J. Sonntag, S. Warzecha, and M. Stede. to appear. Classifying news versus opinions in newspapers: Linguistic features for domain independence. *Natural Language Engineering*.

William Mann and Sandra Thompson. 1988. Rhetorical structure theory: Towards a functional theory of text organization. *TEXT*, 8:243–281.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: The detection, classification and structure of arguments in text. In *Proceedings of the ICAIL 2009*, pages 98–109. Barcelona, Spain.

Huy Nguyen and Diane Litman. 2015. Extracting argument and domain words for identifying argument components in texts. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 22–28, Denver, CO, June. Association for Computational Linguistics.

Shereen Oraby, Lena Reed, Ryan Compton, Ellen Riloff, Marilyn Walker, and Steve Whittaker. 2015. And that's a fact: Distinguishing factual and emotional argumentation in online dialogue. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 116–126, Denver, CO, June. Association for Computational Linguistics.

Andreas Peldszus and Manfred Stede. 2013. From argument diagrams to argumentation mining in texts: A survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 7(1):1–31.

Andreas Peldszus and Manfred Stede. 2016a. An annotated corpus of argumentative microtexts. In *Argumentation and Reasoned Action: Proceedings of the 1st European Conference on Argumentation, Lisbon 2015 / Vol. 2*, pages 801–816, London. College Publications.

Andreas Peldszus and Manfred Stede. 2016b. Rhetorical structure and argumentation structure in monologue text. In *Proceedings of the Third Workshop on Argumentation Mining*, Berlin. Association for Computational Linguistics.

Philipp Petrenz and Bonnie Webber. 2011. Stable classification of text genres. *Computational Linguistics*, 37(2):385–393.

R. Prasad, N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A. Joshi, and B. Webber. 2008. The Penn Discourse Treebank 2.0. In *Proc. of the 6th International Conference on Language Resources and Evaluation (LREC)*, Marrakech, Morocco.

W. Schneider and P. Raue. 1996. *Handbuch des Journalismus*. Rowohlt, Hamburg.

Carlota Smith. 2003. *Modes of discourse. The local structure of texts*. Cambridge University Press, Cambridge.

Christian Stab and Iryna Gurevych. 2014. Annotating argument components and relations in persuasive essays. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, pages 1501–1510, Dublin.

Manfred Stede and Arne Neumann. 2014. Potsdam commentary corpus 2.0: Annotation for discourse research. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 925–929, Reikjavik.

Maite Taboada and William Mann. 2006. Rhetorical Structure Theory: Looking back and moving ahead. *Discourse Studies*, 8(4):423–459.

F.H. van Eemeren, R. Grootendorst, and A.F. Snoeck Henkemans, editors. 1996. *Fundamentals of argumentation theory*. Lawrence Erlbaum, Mahwah/NJ.

M.G. Villalba and P. Saint-Dizier. 2012. Some facets of argument mining for opinion analysis. In *Computational Models of Argument. Proceedings of COMMA 2012*, pages 23–34, Amsterdam. IOS Press.

Egon Werlich. 1975. *Typologie der Texte*. Quelle und Meyer, Heidelberg.

Video Event Detection by Exploiting Word Dependencies from Image Captions

Sang Phan[†], **Yusuke Miyao**[†], **Duy-Dinh Le**[†§], **Shin'ichi Satoh**[†]
[†]National Institute of Informatics, Japan
[§]Multimedia Communications Lab, University of Information Technology, Vietnam
{plsang,yusuke,ledduy,satoh}@nii.ac.jp

Abstract

Video event detection is a challenging problem in information and multimedia retrieval. Different from single action detection, event detection requires a richer level of semantic information from video. In order to overcome this challenge, existing solutions often represent videos using high level features such as concepts. However, concept-based representation can be confusing because it does not encode the relationship between concepts. This issue can be addressed by exploiting the co-occurrences of the concepts, however, it often leads to a very huge number of possible combinations. In this paper, we propose a new approach to obtain the relationship between concepts by exploiting the syntactic dependencies between words in the image captions. The main advantage of this approach is that it significantly reduces the number of informative combinations between concepts. We conduct extensive experiments to analyze the effectiveness of using the new dependency representation for event detection on two large-scale TRECVID Multimedia Event Detection 2013 and 2014 datasets. Experimental results show that i) Dependency features are more discriminative than concept-based features. ii) Dependency features can be combined with our current event detection system to further improve the performance. For instance, the relative improvement can be as far as 8.6% on the MEDTEST14 10Ex setting.

1 Introduction

Detecting event from videos has been an important research topic due to the explosion of internet videos. In order to build a reliable event detection system, one must rely on the video content rather than simply using textual metadata (Davidson et al., 2010). However, internet videos are often captured under arbitrary conditions, which makes the large content variation among videos of the same event. To handle this problem, we can represent videos using multimodal features such as Dense trajectories (Wang and Schmid, 2013), SIFT (Lowe, 2004), and audio MFCC (Lee et al., 1988). Another approach to tackle this problem is to represent videos using the concept-based representation, i.e., video is represented by the concept detection scores indicating the presence of the concepts. This approach is particularly helpful when the number of training videos are scarce (Chen et al., 2014; Habibian et al., 2014b; Habibian et al., 2013; Ma et al., 2013; Ye et al., 2015). Some other works further select a subset of informative concepts for event detection (Jiang et al., 2015; Mazloom et al., 2013).

Nevertheless, the concept-based representation has two drawbacks i) It is non-trivial to obtain a large concept vocabulary. That is why people often combine concepts from multiple vocabularies to construct a larger one, in the hope to cover a wide range of concepts in real world videos (Yu et al., 2014). ii) The relationship between concepts, e.g., co-occurrence, are not captured. However, these relationships can convey a richer level of semantic information which can not be found from encoding individual concepts. Figure 1 illustrates the benefit of exploiting these relationships for event detection.

One can obtain the relationships between concepts by harvesting the social-tagged images (Li et al., 2012), however, tag information is often far from the actual content of the image. Visual phrases (Sadeghi and Farhadi, 2011) models a person and an interacting object through the use of a composite template.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3318–3327, Osaka, Japan, December 11-17 2016.

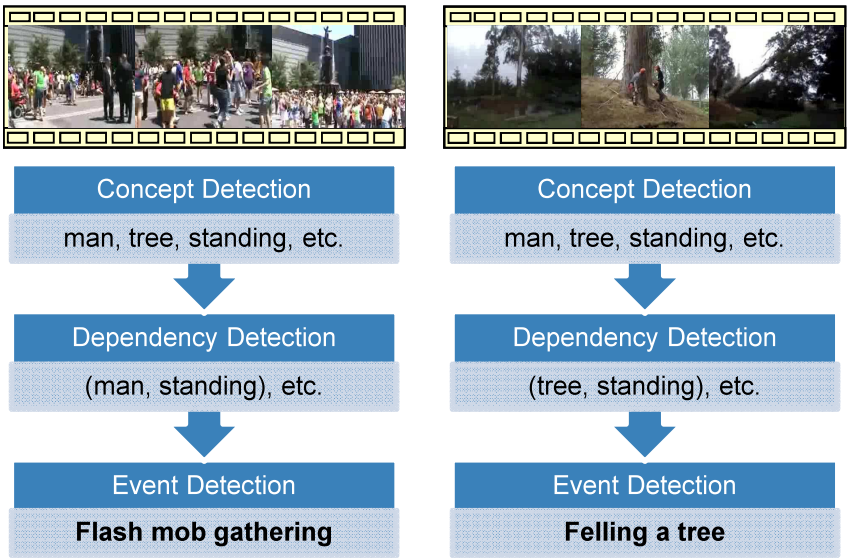


Figure 1: An illustrative example of limitation when using concept-based representation for video. In both videos, "man", "tree", and "standing" can be detected. However, if the dependency *(man, standing)* is also detected, it is more likely a "flash mob gathering". On the other hand, if *(tree, standing)* is detected, it is probably a "Felling a tree" event.

This approach might not be applicable to real world application because there are only 17 visual phrases being used and they are manually selected. Singh et al. (Singh et al., 2015) select relevant concept pairs for event detection by discovering from the event text query. Habibian et al. (Habibian et al., 2014a) also utilize the event query to discover the co-existence or exclusive existence relationships for zero-shot event detection. Assari et al. (Assari et al., 2014) and Can et al. (Can and Manmatha, 2014) model the concept co-occurrences or dependencies based on the joint distributions of two concepts, which relies on the concept detection scores. Borth et al. (Borth et al., 2013) have exploited some syntactic dependencies, such as the combination of adjectives and nouns, for training a sentiment analysis model.

The main drawback of the existing works is that the concept relationships are either manually defined or learned from the concept-based representation, which may face the effect of cumulative error. We address the former issue by exploiting more relationships between concepts such as syntactic dependency relations like subject and object. Such kind of syntactic relationships can also automatically eliminate irrelevant combinations between words or concepts. Typically, we propose a new approach to obtain the relationship between concepts by extracting the word dependencies from the image captions using standard natural language processing technologies. We then train a dependency model directly from the captioned images and use these dependencies as features for event detection.

What differentiates our approach from the previous works is that we explore a comprehensive set of relationship between concepts that is based on the syntactic dependencies. The immediate benefit of our approach is that we can easily obtain a large and informative dependency vocabulary from a much smaller concept vocabulary. In short, the contributions of this paper are twofold: i) A pilot study to exploit word dependencies as features for video event detection. These dependencies encode not only the co-occurrences but also various types of co-occurrences between concepts. ii) We report comprehensive experiments to demonstrate the benefit of the new dependency-based representation.

2 Word Dependency

The central idea in the present paper is to exploit word dependencies as features for event detection. Therefore, the development of the model to predict dependencies for a given video is a key issue. In the following section, we describe a method for obtaining training data for dependency prediction, and our model for dependency prediction.

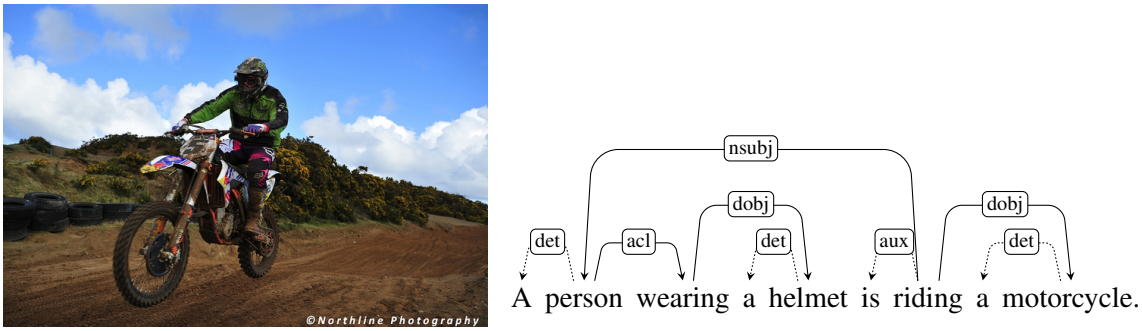


Figure 2: Example of word dependencies. Solid arrows show dependencies included in our vocabulary, while dotted arrows indicate dependencies excluded.

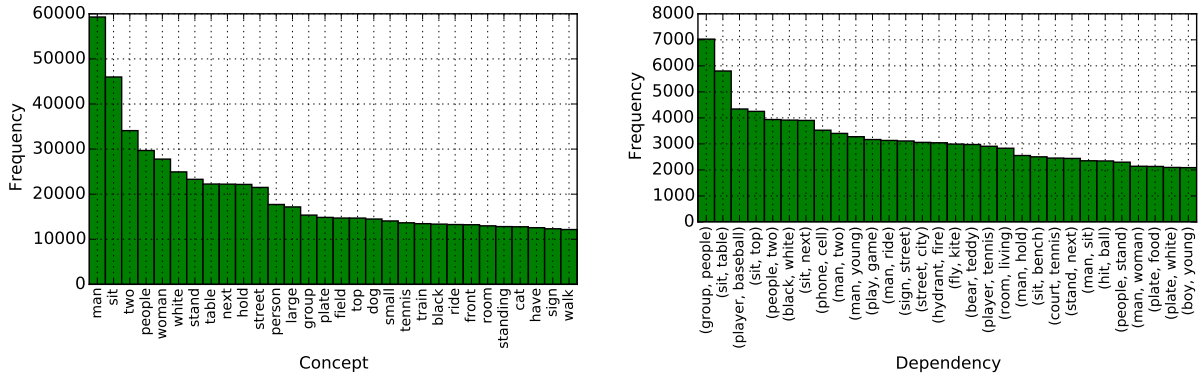


Figure 3: Top 30 frequent concepts and dependencies extracted from the MSCOCO captions.

2.1 Dependency Extraction

Dependency tree represents relationships among words in a sentence, such as subject-verb and verb-object relations. Figure 2 shows an image, its caption, and the dependency tree of the caption obtained by applying Stanford Parser (Manning et al., 2014). For example, an arrow from "riding" to "person" is labeled with "nsubj", which means "person" is a *nominal subject* of "riding".[1] Dependency trees can represent more precise semantic information than a bag of concepts. In this example, the dependecy tree indicates that "person" is the subject of "riding", not the object. This distinction is crucial in event detection as exemplified in Figure 1.

Dependency tree is defined as a set of dependencies, each of which is a triple (*label*, *head*, *dependent*), where *label* is a dependency label (e.g. "nsubj"), *head* is a source word of the dependency (e.g. "riding"), and *dependent* is the other side of the dependency (e.g. "person").

As dependencies consist of pair of words, the simplest way to construct a vocabulary of dependecies is to enumerate all pairs of words. However, this is obviously infeasible. When we have n words and k dependency labels, the total number of dependencies is kn^2. As in the typical situation where $n = 20,000$ and $k = 40$, we need to consider 16 billion dependencies.

In fact, most of the dependencies are not useful for event detection. One reason is that dependencies that appear in real text are very sparse; i.e., most of the triples do not appear. For example, the dependency ("dobj", "wear", "motorcycle"), which means *something is wearing a motorcycle*, is unlikely to appear in real text, because it does not make sense. The distribution of dependencies is highly skewed, and it is not clever to consider all combinations of words and dependency labels. Another reason is that several types of dependencies represent only grammatical relations, and do not convey semantic information.

[1]Dependency labels are defined in the guideline of Universal Dependencies. For details, see: `http://universaldependencies.org/`.

Table 1: Comparison of our model prediction with human evaluation on the COCO validation dataset.

Mean AP	Model prediction	Human evaluation
Concept	0.4347	**0.4410**
Dependency	0.1282	**0.2033**

For example, the dependency ("aux", "riding", "is") describes that "is" is an auxiliary verb of "riding", but this dependency is purely grammatical and does not include any semantic information.

These observations lead us to the idea that we extract a vocabulary of semantically informative dependencies from real texts by excluding unnecessary dependencies. Our solution here is to process dependencies obtained by dependency parsing in the following manner.

1. Apply a dependency parser to caption texts and obtain dependency trees.
2. Remove function words, such as determiners and punctuations, and select top n frequent words as a concept vocabulary.
3. Select dependencies in which both words are included in the concept vocabulary.
4. Cluster dependency labels that have similar relations.

Because our dependencies are extracted from real texts, our vocabulary of dependencies does not include semantically meaningless dependencies. Step 2 and 3 further exclude purely grammatical dependencies and less frequent dependencies. Step 4 is intended to further reduce semantically subtle distinctions. For example, two labels, "nsubj" and "csubj", are defined to represent subject relations, but their distinction (nominal vs. clausal) is not particularly important for our purpose. Therefore, we collapse such semantically similar dependency labels to get a reduced number of dependency types.

In Figure 2, solid arrows show dependencies included in our vocabulary, while dotted arrows indicate dependencies excluded. This example demonstrates that our method selects dependencies that capture semantic information of original texts while excluding less informative dependencies.

2.2 Dependency Modeling

Images and caption texts for the development of the dependency prediction model are obtained from the training set of Microsoft COCO (Lin et al., 2014), which contains 82,783 images and 414,113 captions constructed by crowdsourcing. Stanford CoreNLP 2015-04-20 is used for dependency parsing. We selected $n = 1,000$ concepts, which covers 89.4% of words (excluding function words) in the caption texts of the training data. Using this concept vocabulary, the method described in Section 2 extracted 20,931 dependencies that have more than 10 occurrences in the training data, which cover 63.3% of all the dependencies in the original caption texts. Top 30 frequent concepts and dependencies extracted from the training captions are shown in Fig. 3.

We consider the problem of predicting dependencies as a multi-label classification task because multiple dependencies can present in one image. To train the dependency model, we finetune a deep neural network on the COCO images from the VGG (16 layers) network (Simonyan and Zisserman, 2014). We add a cross-entropy loss layer on top of a sigmoid layer to account for the loss function. The entire network is trained using the Stochastic Gradient Descent (SGD) optimizer.

We compare the model predictions with the human performance to evaluate the effectiveness of our concept and dependency models. In order to measure the human accuracy, we use concepts/dependencies of one gold caption as the predicted labels, and use concepts/dependencies from the remaining captions of the same image as gold labels. If an image has multiple captions, we repeat this procedure so that every caption is used as the prediction, and report the average performance.

The detailed comparison of our model prediction with human performance is shown in Table 1. We found that our concept prediction model has comparable performance with agreement among human-annotated captions, while our dependency prediction is inferior to the human performance. The low agreement between annotators on the dependency evaluation also demonstrates the inherent challenge of dependency modeling.

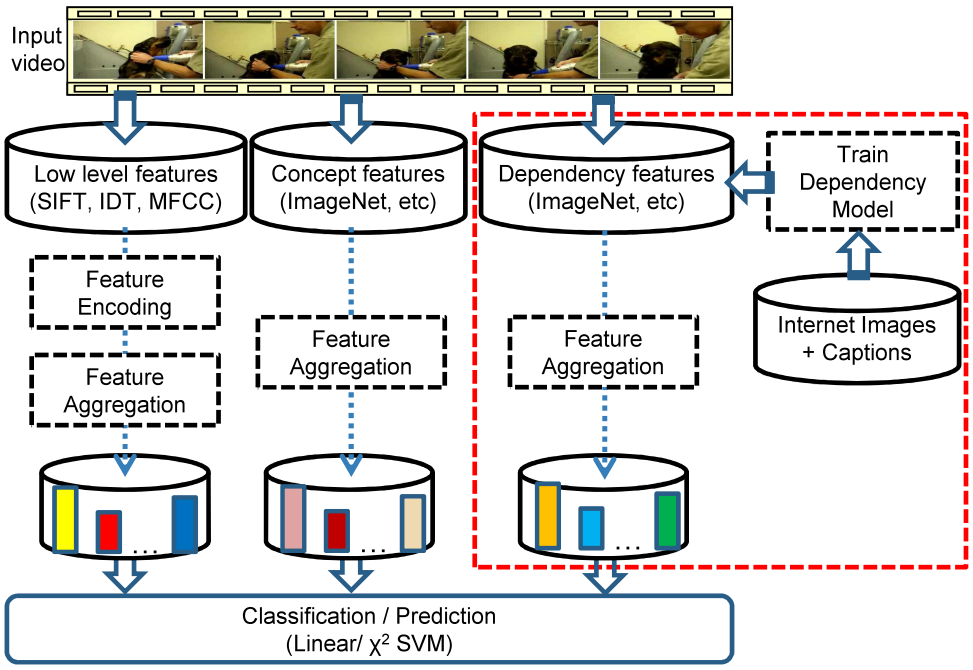


Figure 4: Overview of our event detection framework.

3 Event Detection Framework

In this section, we present a common event detection system, which consists of four main components: feature extraction, feature encoding, feature classification, and feature fusion (Fig. 4).

3.1 Feature extraction

Our framework supports a large variety of features including low-level and high-level features. Low-level features such as audio MFCC and Dense trajectories (Wang and Schmid, 2013) can be extracted from temporal windows spanning of several frames, while still image features such as SIFT (Lowe, 2004) is extracted from sampled frames of video. These raw low-level features require a special encoding technique to generate video-level representation, which will be described in Section 3.2.

We also extract high-level features such as visual concept features extracted from deep visual models. For example, our framework supports state-of-the-art deep learning features which are extracted from pre-trained deep models (Simonyan and Zisserman, 2014). These models are trained on large image collection such as ImageNet (Deng et al., 2009). Using these models, we can extract concept scores for each sampled frame and aggregate them into the final representation for each video. One can also utilize features from the previous fully-connected layers, such as `fc6` and `fc7`, to train the event detectors because they often retain richer information than the last full-connected layer.

Our concept and dependency features can be also considered as high-level features. However, different from the aforementioned high-level features that are extracted from pre-trained visual models, our concept and dependency features are extracted from our concept and dependency models respectively.

3.2 Feature encoding

We use Fisher Vector encoding to map local descriptors extracted from low-level features into the video-level representation. The Fisher vector (FV) has been successfully employed for various image and action classification problems (Perronnin and Dance, 2007; Wang and Schmid, 2013). It can be considered as an extension of Bag-of-Words (BoW) encoding where both first- and second-order statistics between the local descriptors and their assigned codewords are also encoded. Therefore, Fisher vector can achieve comparable performance to that of BoW while using a much smaller codebook.

In our experiment, we first randomly selected one million local descriptors for training a Gaussian mixture model (GMM), which is later used as the codebook for encoding. As suggested in (Perronnin et

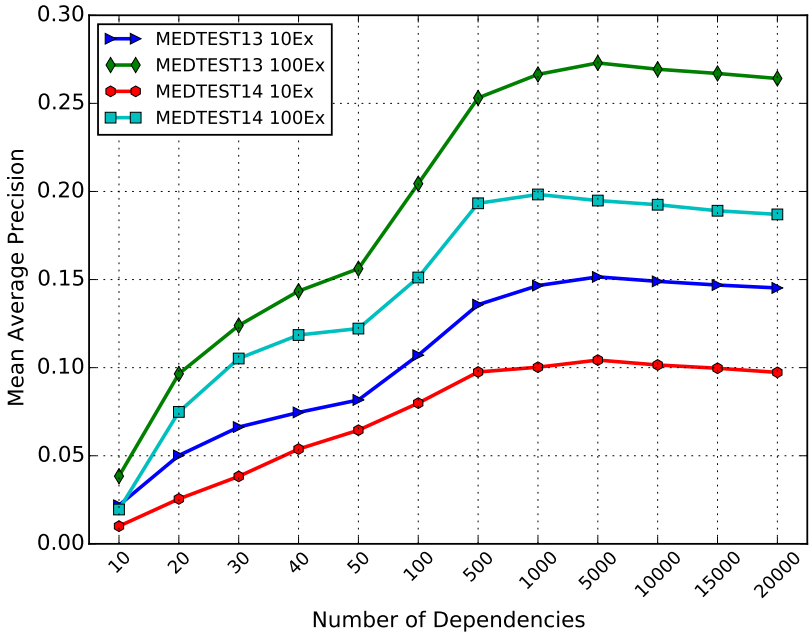


Figure 5: Results of event detection performance with respect to various vocabulary sizes.

al., 2010), it is better to reduce the local feature dimension by using principal component analysis (PCA) and apply power normalization, e.g., with $\alpha = 0.5$, followed by L2-normalization to the Fisher vector.

3.3 Feature classification

We use LibSVM (Chang and Lin, 2011) for training event detectors. The pre-computed kernel technique is utilized to reduce the training time. This technique is especially useful when the number of events are large. For low-level features, which are encoded using Fisher vector encoding, we use linear SVM for training and testing. For high-level features, since its feature dimension is rather small, we employ the χ^2 kernel SVM to obtain a better recognizing performance.

3.4 Feature fusion

Different features cover various characteristics of multimedia data. Hence it is natural to combine these features to get the benefit from multi-modal features. For the sake of simplicity, we use the average late fusion strategy for feature combination.

4 Experiments

We conduct experiments on the TRECVID Multimedia Event Detection (MED) 2013 and 2014 benchmarks under the 10Ex and 100Ex settings, in which only 10 and 100 training videos are given for each event respectively (Over et al., 2014). Dependency features are extracted from the final layer (fc8) of the dependency network presented in Section 2.2 for each sampled frame in video (1 frame every 4 seconds) and then aggregated to form the video representation. We employ LibSVM (Chang and Lin, 2011) with the exponential-χ^2 kernel for event training and testing. All the results are reported in terms of mean average precision (mean AP).

4.1 How many dependencies?

We first study how different number of dependencies affect to the event detection performance. We found that selecting the most frequent dependencies contributes more than randomly selecting the same number of dependencies. Following this observation, we only consider the top frequent dependencies, and report the results in Fig. 5. Small vocabulary sizes often result in a poor outcome. When increasing the vocabulary size, the performance also increases rapidly. The highest performance can be obtained

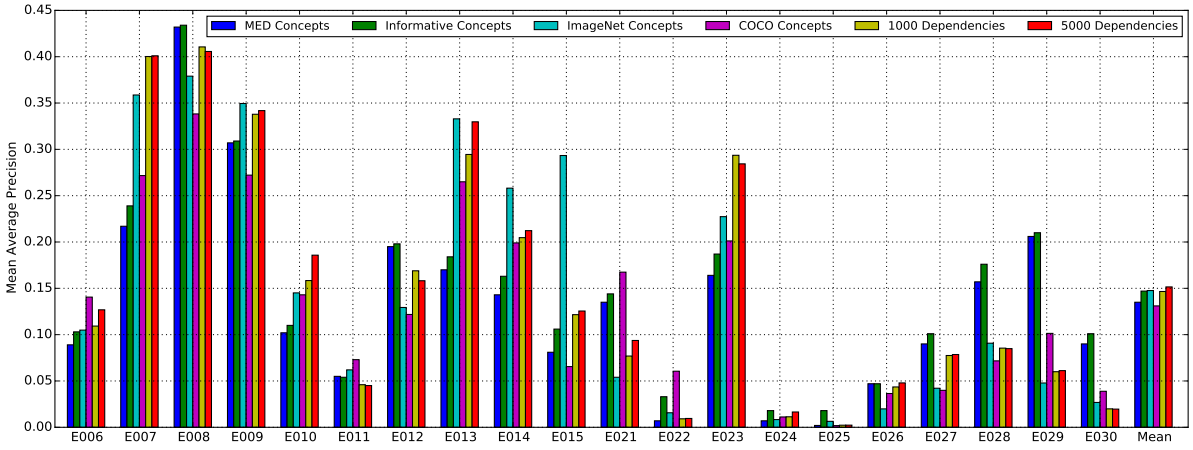


Figure 6: Performance comparison of dependency features with other methods on the MEDTEST13 10Ex setting. Performance of each event is reported in each group, the last group shows the average performance over all events. *MED Concepts* refers to concepts obtained from video annotations (Habibian et al., 2013), and *Informative Concepts* refers to selected concepts in (Mazloom et al., 2013). The mean average precisions from left to right are 0.1350 (MED Concepts), 0.1470 (Informative Concepts), 0.1476 (ImageNet Concepts), 0.1310 (COCO Concepts), 0.1466 (1000 Dependencies) and **0.1515** (5000 Dependencies).

by using around thousands of dependencies. However, if more than ten thousands of dependencies are added to the vocabulary, the performance tends to slightly drop. The reason is that at this point, the dependencies become much less frequent, so their detectors might become less reliable as well.

4.2 Comparison with other methods

Figure 6 shows the performance of our dependency features on the MEDTEST13 10Ex. We include the reported results on the same setting in (Habibian et al., 2013) and (Mazloom et al., 2013) for comparison. The performance of ImageNet concepts (Deng et al., 2009) are produced using the same network that was used to fine-tune our dependency model (Simonyan and Zisserman, 2014). We also select 1,000 COCO concepts as described in Section 2 and report its performance. The best dependency run can outperform the COCO concept run with a relative improvement of **15.6%**. Our dependency features only achieve a marginal improvement over the ImageNet baseline. However, this result is already encouraging because our dependency model was trained with around 80K labeled images, compared to about one million labeled images of the ImageNet model.

4.3 Contribution to our MED system

Finally, we conduct experiments to demonstrate the contributions of our dependency feature to our event detection system. State-of-the-art event detection systems (Yu et al., 2014; Yu et al., 2015) often employ features from multiple modalities such as audio and visual features. Following this strategy, we implemented a number of features in our system including audio MFCC (Lee et al., 1988), SIFT (Lowe, 2004), and Dense trajectories (Wang and Schmid, 2013). In light of the success of deep learning, we also use features extracted from (Simonyan and Zisserman, 2014), which is a pre-trained network on ImageNet (Deng et al., 2009). We also include the reported performance of recent state-of-the art systems in (Yu et al., 2014) and (Yu et al., 2015) for comparison.

Table 2 compares performance of all aforementioned features. Our (5,000-dimensional) dependency feature outperforms ImageNet concepts in all settings, while still inferior to the performance of the Dense trajectories features (Wang and Schmid, 2013). When combining our existing system with the new dependency feature, we achieve a relative improvement of **6.8%** and **8.6%** on the MEDTEST13 10Ex and MEDTEST14 10Ex settings respectively. We report lower results than the ones reported in (Yu et al., 2014; Yu et al., 2015). The reason can be due to the fact that they included more concept features

Table 2: Contribution of the new dependency features to our event detection system.

	MEDTEST13		MEDTEST14	
	10Ex	100Ex	10Ex	100Ex
MFCC (Lee et al., 1988)	0.0440	0.1010	0.0449	0.0776
SIFT (Lowe, 2004)	0.0893	0.2235	0.0730	0.1796
IDT (MBH) (Wang and Schmid, 2013)	0.1550	0.2812	0.0937	0.2138
IDT (HOGHOF) (Wang and Schmid, 2013)	0.1743	0.3198	0.1184	0.2595
ImageNet (Simonyan and Zisserman, 2014)	0.1476	0.2632	0.1037	0.1930
Dependencies	0.1515	0.2729	0.1043	0.1948
Late fusion (w/o dependencies)	0.2420	0.4101	0.1707	0.3449
Late fusion (w/ dependencies)	**0.2584**	**0.4244**	**0.1853**	**0.3571**
DMSY (Yu et al., 2015)	0.2800	0.3860	0.2330	0.3260
CMU (Yu et al., 2014)	0.3130	0.4640	0.2850	0.4190

in their system such as Google Sports (Karpathy et al., 2014) and YFCC concepts (Thomee et al., 2015). Example of some detected concepts for each event can be found in Table 3.

5 Conclusions and Future Work

We exploited word dependencies as a new semantic video representation for recognizing complex events. Different from the existing works, this representation encodes the relationship between concepts based on the syntactic dependencies between words. Therefore it captures semantically richer information, which is crucial for video event detection. We demonstrated that the dependency-based representation is more discriminative than the concept-based representation. Moreover, it also helps improve the detection performance of our existing event detection system.

One limitation of the current work is that we exploited image captions rather than video captions. The main reason is that we have not found any large-scale video captioning corpus for this project. Also the accuracy of dependency prediction from video would be less accurate since video is more challenging to model. In the future work, we plan to extend this work to further modeling word dependencies from video captions.

Acknowledgements

This research was partially supported by JST, PRESTO.

References

Shayan Assari, Amir Zamir, and Mubarak Shah. 2014. Video classification using semantic concept co-occurrences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2529–2536.

Damian Borth, Rongrong Ji, Tao Chen, Thomas Breuel, and Shih-Fu Chang. 2013. Large-scale visual sentiment ontology and detectors using adjective noun pairs. In *Proceedings of the 21st ACM international conference on Multimedia*.

Ethem F Can and R Manmatha. 2014. Modeling concept dependencies for event detection. In *Proceedings of International Conference on Multimedia Retrieval*, page 289. ACM.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Table 3: Top 3 dependencies detected by our system for each event in the MEDTEST13 collection.

ID	Event name	Top dependencies
E006	Birthday party	(group, people), (group, woman), (stand, front)
E007	Changing a vehicle tire	(black, white), (man, young), (man, ride)
E008	Flash mob gathering	(crowd, people), (group, people), (walk, down)
E009	Getting a vehicle unstuck	(black, white), (photo, black), (man, young)
E010	Grooming an animal	(black, white), (man, young), (sit, chair)
E011	Making a sandwich	(stand, front), (man, young), (black, white)
E012	Parade	(crowd, people), (group, people), (walk, down)
E013	Parkour	(black, white), (man, young), (building, tall)
E014	Repairing an appliance	(take, picture), (man, young), (black, white)
E015	Working on a sewing project	(man, young), (take, picture), (phone, cell)
E021	Attempting a bike trick	(black, white), (man, ride), (board, skate)
E022	Cleaning an appliance	(black, white), (man, young), (take, picture)
E023	Dog show	(crowd, people), (group, people), (black, white)
E024	Giving directions to a location	(black, white), (man, young), (building, tall)
E025	Marriage proposal	(man, young), (crowd, people), (group, people)
E026	Renovating a home	(man, young), (black, white), (stand, front)
E027	Rock climbing	(man, young), (black, white), (hold, up)
E028	Town hall meeting	(crowd, people), (group, people), (group, woman)
E029	Winning a race without a vehicle	(group, people), (crowd, people), (man, young)
E030	Working on a metal crafts project	(phone, cell), (man, young), (man, wear)

Jiawei Chen, Yin Cui, Guangnan Ye, Dong Liu, and Shih-Fu Chang. 2014. Event-driven semantic concept discovery by exploiting weakly tagged internet images. In *Proceedings of International Conference on Multimedia Retrieval*.

James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, et al. 2010. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

Amirhossein Habibian, Koen EA van de Sande, and Cees GM Snoek. 2013. Recommendations for video event recognition using concept vocabularies. In *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*, pages 89–96. ACM.

Amirhossein Habibian, Thomas Mensink, and Cees GM Snoek. 2014a. Composite concept discovery for zero-shot video event detection. In *Proceedings of International Conference on Multimedia Retrieval*, page 17. ACM.

Amirhossein Habibian, Thomas Mensink, and Cees GM Snoek. 2014b. Videostory: A new multimedia embedding for few-example recognition and translation of events. In *Proceedings of the ACM International Conference on Multimedia*, pages 17–26. ACM.

Lu Jiang, Shoou-I Yu, Deyu Meng, Yi Yang, Teruko Mitamura, and Alexander G Hauptmann. 2015. Fast and accurate content-based semantic search in 100m internet videos. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 49–58. ACM.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732.

Chin-Hui Lee, F.K. Soong, and Bing-Hwang Juang. 1988. A segment model based approach to speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing, 1988. ICASSP-88.*, pages 501 –541 vol.1.

Xirong Li, Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. 2012. Harvesting social images for bi-concept search. *IEEE Transactions on Multimedia*, 14(4):1091–1104.

T.Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *European conference on computer vision*.

David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Zhigang Ma, Yi Yang, Zhongwen Xu, Shuicheng Yan, Nicu Sebe, and Alexander G Hauptmann. 2013. Complex event detection via multi-source video attributes. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL*, pages 55–60.

Masoud Mazloom, Efstratios Gavves, Koen van de Sande, and Cees Snoek. 2013. Searching informative concept banks for video event detection. In *Proceedings of International Conference on Multimedia Retrieval*.

Paul Over, Georges Awad, Martial Michel, Johnatan Fiscus, Greg Sanders, Wessel Kraaij, Alan F Smeaton, and Georges Quénot. 2014. Trecvid 2014- an overview of the goals. *TRECVID*.

Florent Perronnin and Christopher Dance. 2007. Fisher kernels on visual vocabularies for image categorization. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*.

Florent Perronnin, Jorge Sánchez, and Thomas Mensink. 2010. Improving the fisher kernel for large-scale image classification. In *European conference on computer vision*, pages 143–156. Springer.

Mohammad Amin Sadeghi and Ali Farhadi. 2011. Recognition using visual phrases. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1745–1752. IEEE.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Bharat Singh, Xintong Han, Zhe Wu, Vlad I Morariu, and Larry S Davis. 2015. Selecting relevant web trained concepts for automated event retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4561–4569.

Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. 2015. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*.

Heng Wang and Cordelia Schmid. 2013. Action recognition with improved trajectories. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3551–3558.

Guangnan Ye, Yitong Li, Hongliang Xu, Dong Liu, and Shih-Fu Chang. 2015. Eventnet: A large scale structured concept library for complex event detection in video. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, pages 471–480. ACM.

S Yu, L Jiang, Z Mao, XJ Chang, XZ Du, C Gan, ZZ Lan, ZW Xu, XC Li, Y Cai, et al. 2014. Cmu-informedia@ trecvid 2014 multimedia event detection (med). In *TRECVID Workshop*.

Shoou-I Yu, Lu Jiang, Zhongwen Xu, Yi Yang, and Alexander G Hauptmann. 2015. Content-based video search over 1 million videos with 1 core in 1 second. In *Proceedings of International Conference on Multimedia Retrieval*.

Predicting Restaurant Consumption Level through Social Media Footprints

Yang Xiao, Yuan Wang, Hangyu Mao, and Zhen Xiao
School of Electronics Engineering and Computer Science, Peking University, China
{xiaoyang, wangyuan, mhy, xiaozhen}@net.pku.edu.cn

Abstract

Accurate prediction of user attributes from social media is valuable for both social science analysis and consumer targeting. In this paper, we propose a systematic method to leverage user online social media content for predicting offline restaurant consumption level. We utilize the social login as a bridge and construct a dataset of 8,844 users who have been linked across Dianping (similar to Yelp) and Sina Weibo. More specifically, we construct consumption level ground truth based on user self-report spending. We build predictive models using both raw features and, especially, latent features, such as topic distributions and celebrities clusters. The employed methods demonstrate that online social media content has strong predictive power for offline spending. Finally, combined with qualitative feature analysis, we present the differences in words usage, topic interests and following behavior between different consumption level groups.

1 Introduction

Over the past decade, microblogging services like Twitter and Sina Weibo have built up a huge user base. For example, by the end of 2014, Sina Weibo has accumulated more than 500 million users, out of which 167 million are monthly active users. With the growing popularity of microblogging service, businesses are also looking for new opportunities on social media, e.g., identifying target consumers and marketing their products. Compared with traditional consumer targeting scenarios, social media has several factors in its favour. Firstly, traditional consumer targeting techniques are mainly based on users' query logs or web access histories, and it is generally limited by session length (Dasgupta et al., 2012). While on social media, users have accumulated much more abundant traces, such as tweets, relationships and profiles. Secondly, according to a recent study (Nielson, 2012), 92% of consumers believe recommendations from friends over all other forms of advertising and 64% of salesperson believe word of mouth is the most effective way of marketing. Since social network links both friends and families, it becomes a natural platform for business to take advantage of word of mouth utility (Trusov et al., 2009).

Demographic profile is the starting point of defining target consumers for marketing. Demographic includes multiple aspects such as simple attributes like gender and age, and more complicated attributes like income, personality and consumption level. Since a large number of users provide their profiles on social media, inferring user attributes such as gender (Ciot et al., 2013; Liu and Ruths, 2013; Rao et al., 2011), age (Al Zamal et al., 2012; Nguyen et al., 2013), political polarity (Volkova et al., 2014), or occupation(Preoţiuc-Pietro et al., 2015a) from social media has already been widely studied. In this paper, we focus on the consumption level attribute prediction.

Consumption behavior reflects one's economic capacity and living standards (Brewer et al., 2012). An accurate consumption level prediction model can help business dealers identify their target consumers and recommend suitable products. Moreover, since consumption level is an important factor of economic status (Stutzer, 2004), effective prediction of consumption level will facilitate social economic researches on social media. However, unlike the gender attribute that is displayed on one's page, or political orientation that is unequivocally stated in one's tweets, consumption behavior related attributes are hard to acquire automatically from microblogging service.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3328–3338, Osaka, Japan, December 11-17 2016.

In this paper, we take advantage of the "socialization" of third party websites to build the ground truth collection. Specifically, we map one's Weibo (the largest microblogging service in China) account to his or her corresponding Dianping (the largest consumer review site in China) account. We carefully construct a dataset of 8,844 users who have been linked across Dianping and Weibo, and use one's self-report spending history to build the consumption level ground truth.

We propose a systematic method that takes social media features for consumption level prediction. We hypothesize that consumption level is correlated with various features, mainly including three aspects. The first aspect is user profiles including gender, age and education. Secondly, we hypothesize that language use in social media is a predictive factor for consumption level. We take textual features as the second aspect for prediction. Finally, we hypothesize that people of different economic statuses have their own unique tastes and interests. We take the following links which reflect one's interests and tastes as the third aspect for prediction. Owing to the noisy tweet content and sparse following relationships, this problem is technically challenging. We make use of topic modeling and LIWC categories to generate dense representations of text features. For graph features, due to the large quantities of users on Weibo, using raw following relationships directly has sparsity problem. To address it, we propose a matrix factorization algorithm on following matrix and naturally generate dense representations of user following preferences.

We take the raw features, and especially latent features as input features. We adopt the gradient boosted decision tree that can generate nonlinear combinations of input features, to predict user's consumption level. Empirical experiments demonstrate that rich features on social media have strong predictive power for consumption level, and latent features have best prediction performance. We conduct Spearman correlation test between topic preference and restaurant spending. We have found that the topic preference of a user is significantly correlated with the consumption level. We also report several new findings about consumption level and behavior on social media, e.g., users who follow topics such as luxury brands and politics, or talk more about money (e.g., audit, cash, owe) tend to be of higher consumption level, while users who follow topics about popular stars, use more character expressions and more assent words (e.g., agree, OK, yes) tend to be of lower consumption level. These findings will facilitate future attempts to consumer targeting, and may suggest extension application to spending prediction in other domains. The flexibility of our approach lies in that we identify important and general types of correlations that are easy to leverage from external social media sites.

2 Dataset Description

We focus on Sina Weibo, the largest Chinese microblogging service, as the studied microblogging service. We select a popular crowd-sourced review site, Dianping as the external website to help construct user consumption level ground truth. We do not adopt the automatic user linking methods but use "self-disclosure" to identify the same user across these two medias: some Weibo users provide their Dianping links in their tweets. This approach generates an accurate linking of across-website users.

Weibo Dataset: Sina Weibo is the largest Chinese microblogging service. We crawl the Weibo search page[1] to find those who declare their Dianping pages in their tweets. In this way, we get 62,015 users and then we crawl all the detailed information of these linked users, including profiles, tweets, followers and following links.

Dianping dataset: Dianping[2] is the largest social based crowd-sourced review site in China, which is similar to Yelp in terms of overall design and service. Dianping has two major components: users and local businesses. Users on Dianping can comment on and grade for local businesses, e.g., restaurants and hotels. Besides score and comment, user can also provide how much an average person spend for a meal in the restaurant. Each restaurant is usually assigned a small set of taste categories, price, score and starts a thread of reviews. We crawl all the reviews of 62,015 linked users mentioned above. The linked users have posted 286,069 reviews for 35,650 restaurants. We also crawl the 35,650 restaurants pages. The 35,650 restaurants are located in 45 cities in China, covering 98 different taste categories.

[1] http://s.weibo.com/

[2] http://www.dianping.com/

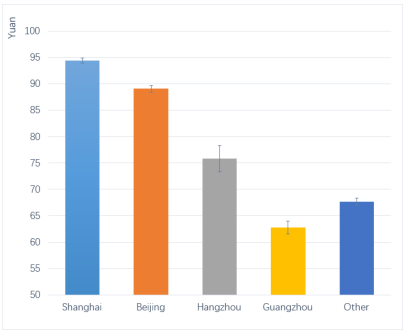


Figure 1: Average spending in different cities

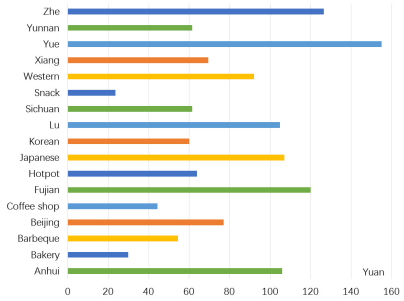


Figure 2: Average spending of different categories

We calculate the average restaurant spending in different cities and the result is shown in Figure 1. As shown in the figure, big cities such as Beijing and Shanghai are more expensive than other cities such as Guangzhou and Hangzhou. Moreover, users in Beijing and Shanghai take a share of 78.8% of our users. To make the population in our dataset more homogeneous, we only keep those who are located in Beijing or Shanghai. Dianping contains many categories of restaurants and different categories have different prices. The prices of different categories are shown in Figure 2. Since categories like cafe shop mainly provide drinks and can not be taken as real restaurants, we filter out restaurants in such categories. In order to gain a reliable estimation of consumption level, we only keep those who have post more than ten reviews. Finally, we obtain a total of 8,844 users. We find that these users are active on Weibo. They have posted 13,026,078 tweets and have 3,078,497 following links in total. We summarize the data statistics in Table 2.

Users	Tweets	Followings	Restaurants	Reviews
8,844	13,026,078	3,078,497	35,650	286,069

Table 1: Data statistics of our dataset for linked across-media users.

3 Problem Definition

If we can estimate one's consumption level according to his or her microblogging account, we can help the businesses design suitable marketing strategies, e.g., sending coupons to those who are more sensitive to price. We formulate this task as a typical prediction task: it aims to estimate the user consumption level. We assume the following general definition of the task: given a user's accumulated spending history $h^{(i)}$ and corresponding social media data s, we would like to test with varying type of s how accurate introduction of s can predict the label of $h^{(i)}$.

To formulate the consumption level prediction task, we assume that there are a set of m users $U = \{u^{(1)}, u^{(2)}, u^{(3)}, ...u^{(m)}\}$. For each user, let $y^{(i)}$ denotes the consumption level of user $u^{(i)}$. Higher value of $y^{(i)}$ means higher consumption level. A feature vector $x^{(i)}$ can be constructed for each user $u^{(i)}$. The aim of the learning task is to derive a prediction function f such that, for each feature vector $x^{(i)}$, it outputs a consumption level $f(x^{(i)})$.

4 Features

In this section, we discuss features extracted from Weibo service. In particular, we study how to derive effective latent features for the task.

4.1 Raw Features

Raw features are features that can be directly extracted from one's Weibo homepage. In this category, we consider the following three aspects:

I. METADATA Demographic fields of users. Fields such as gender and education level are binary features. Education level with value one indicates the user has accessed to university education. Tags

are a list of self reported words that users describe themselves. A separate binary feature is included for each unique tag. A user has 4 tags on average. For example, a user use tags such as music and shopping to represent one's interest.

II. RAWWORDS Unigrams in one's tweets and retweets. We use binary features for each unique unigram.

III. RAWFOLLOW All users that one follows. We use binary features for each unique followee.

4.2 Latent Features

In addition to raw features, we consider leveraging the latent features to improve predictions. We attempt to find semantics from the sparse user word matrix and user following matrix.

LIWC of Tweets (LIWCT)

Linguistic Inquiry and Word Count is a dictionary that classifies English words into psychological meaningful categories (Tausczik and Pennebaker, 2010). LIWC demonstrates its ability to detect psychological meaning in a wide range of applications such as emotionality investigation (Alpers et al., 2005) and personality prediction (Pennebaker and King, 1999). Previous study on social media shows that LIWC provides effective psychological features to determine the relationship between users (Adali et al., 2012) and the credibility of comments (Ott et al., 2011). Chinese LIWC (Huang et al., 2012) is a Chinese version LIWC that classifies 7444 Chinese words into 71 dimensions. Chinese LIWC have some slight differences from English one due to the difference in language, e.g., stemming is not needed for Chinese and verbs in Chinese do not have tense form. In general, the function of Chinese LIWC is similar to the English one.

We hypothesize that people at different consumption levels have different scores on LIWC features. For each user, we calculate the count of words that fall into each LIWC category, and get a vector of 71 dimensions. The vector is then normalized by the sum of the all values in it. Each value in the vector represents the user's usage preference on the linguistic category.

Topic Modeling of Tweets (LDAT)

In addition to employing raw text features, i.e., RAWWORD, we also use a high level representation of words to discover latent semantics. In order to distill topics from tweets, we adopt Latent Dirichlet Allocation method (Blei et al., 2003), which is an unsupervised learning method to discover latent topic distribution using large amounts of documents.

The model has two parameters, i.e., the document topic distribution θ_i and the topic word distribution φ_i. By learning θ_i and φ_i, document's topic distribution can be obtained and hence we get user's preferences on each topic. Since we care about user's topic distribution instead of a single tweet's topic distribution, we aggregate the user's tweets and retweets into documents and take the documents as the input of LDA model.

We take the resulted vector θ_i to describe user u_i's topic distribution. In this study, we set topic number k to 200 and run LDA with 500 iterations using Gibbs[3] sampling.

SVD Following Matrix (SVDF)

Because economic status is the central concern of hierarchical society, people of different economic statuses have their own unique tastes and interests (Wong and Ahuvia, 1998). Intuitively, if one is interested in some areas, it is natural that he or she will follow some related celebrities on social media. Therefore, celebrities that one follows can reflect the interest of the user (Lim and Datta, 2012). Since celebrities have large quantities of followers, in this paper, we take the account that has more than 30,000 followers as a celebrity account. The most direct way of utilizing celebrity features is to take each individual celebrity as a distinct feature aspect, i.e., RAWFOLLOW. However, since the celebrity accounts are of a huge number, using the celebrity features directly can result in sparsity problem. To tackle this problem, we use matrix decomposition to capture the hidden interest of one user. The row of the matrix denotes users in our dataset, the column of the matrix denotes the celebrity accounts and the element

[3]http://gibbslda.sourceforge.net/

f_{ij} is set to 1 if user u_i follows celebrity c_j. Different from traditional matrix decomposition task that has elements ranging from 1 to 5, in our case, the element of matrix only have two values representing whether user follows the celebrity. Hence, we choose the following logistic loss as our optimization goal.

$$U(i) = \arg\min_{w} \sum_{j} \log(1 + \exp(-f_{ij} \cdot w^T C(j))) + \lambda \|w\|^2$$

where f_{ij} denotes whether user u_i follows celebrity c_j, $U(i)$ denotes the latent vector of u_i, $C(j)$ denotes the latent vector of c_j and λ is the regularization coefficient.

We take the resulted $U(i)$ with varying length to describe one's interest. In this study, we employ stochastic gradient descent for matrix factorization parameter inference (Rendle, 2012) .

5 Clustering and Labeling

In this section, we introduce how we derive user consumption level label $y^{(i)}$ from the consumption history $h^{(i)}$. We use Gaussian mixture model to cluster users over their price space. The motivation for clustering is to find the natural structure of consumption prices and avoid manual threshold settings. This makes the labels of users more reliable and applicable to other dataset. The following part introduces the Gaussian mixture model and how we apply it to our dataset.

Gaussian mixture model is a probabilistic model which assumes that data is generated from finite number of Gaussian distributions. Given n data points $\mathbf{x} = \{x_1, x_2, x_3, ..., x_n\}$, the probability of generating the data x_i is as follows:

$$p(x_i|\pi, \Theta) = \sum_{z=1}^{k} p(z|\pi) p(x_i|\theta_z)$$

where π denotes the distribution over components, $p(x_i|\theta_z)$ is normal distribution where $\theta_z = (\mu_z, \sigma_z)$. $p(x_i|\theta_z)$ can be formulated as $p(x_i|\theta_z) = \frac{1}{\sqrt{2\pi}\sigma_z} e^{-\frac{(x_i-\mu_z)^2}{2\sigma_z^2}}$

In our task, we calculate average of the user u_i's spending history $h^{(i)}$ for each user, and in this way we get a list containing average spending of all users, i.e., $L = \{avg(h^{(1)}), ...avg(h^{(m)})\}$. $L^{(i)}$ ranges from 33 Yuan to 436 Yuan. We calculate average spending irrespectively of restaurant category because we find that most users visit diverse categories of restaurants. Since we focus on the relative consumption level of users instead of the real spending of users, we formalize the task as classification task. We apply the Gaussian mixture model to the spending list L. We assume that users come from k (k=2) different consumption levels, and the user's label $y^{(i)}$ is the cluster number the user belongs to.

6 Experiment

In the above section, we have shown how to extract features from social media and how to derive labels from spending history. We are going a step further to figure out the feasibility of using these social media features to predict the user's consumption level. We conduct experiments on the collected Weibo and Dianping dataset as described in Section 2. We set the component number of Gaussian mixture model to 2, i.e., users are labeled either one or zero indicating whether they are of high consumption level. We take 60% of users as training portion, 20% as validation portion and the remaining 20% as the test portion. For the evaluation, we take the accuracy, precision, recall and F1 measure as the evaluation metrics.

6.1 Quantitative Evaluation

We employ GBDT[4] and logistic regression[5] as the prediction models. Since traditional survey-based consumption behavior analysis task mainly focus on demographic attributes (Jang et al., 2004; Fodness, 1994), in this paper, we refer to the results obtained with feature METADATA as *baseline*. Specifically,

[4]http://github.com/dmlc/xgboost

[5]https://www.csie.ntu.edu.tw/ cjlin/liblinear/

Category	Name	Accuracy	Precision	Recall	F1
BASELINE	Age	0.5471	0.5547	0.5108	0.5318
	EDU	0.5507	0.5564	0.5324	0.5441
	TAG	0.5655	0.5629	0.6408	0.5993
	ALL	0.5889	0.5775	0.6715	0.6210
RAW	RAWWORD	0.6574	0.6544	0.6715	0.6628
	RAWFOLLOW	0.6945	0.6783	0.7529	0.7137
	ALL	0.7118	0.6969	0.7610	0.7276
LATENT	LIWCT	0.6066	0.5908	0.6982	0.6400
	LDAT	0.7451	0.7303	0.7863	0.7573
	SVDF	0.7673	0.7760	0.7635	0.7697
	ALL	0.8012	0.7821	0.8413	0.8106

Table 2: Prediction results with different feature sets.

for *baseline* features, we use logistic regression to combine all features together, which is a one layer classifier; for the RAW features and LATENT features, we construct a two layer classifier to combine heterogeneous features, i.e., in the first layer, we construct base classifiers using single feature sources, then we build a second layer classifier on top of the first layer classifier which takes the output of base classifiers as input. In our study, for the second layer classifier, we employ logistic regression to combine heterogeneous features. Since features such as RAWWORD and RAWFOLLOW are of high dimensions, we use L1 regularized logistic regression to construct base classifier; for the LATENT features, we use GBDT as the prediction model to construct non-linear base classifier. For LATENT feature base classifier selection, we have compared logistic regression with GBDT and found that GBDT preforms better than logistic regression. Therefore, we choose GBDT as the base classifier. The prediction results with different set of features are listed in Table 2.

For *baseline* features, we conduct experiments on age, gender, education and tags. As shown in Table 2, tags are the most predictive features and perform better than education and age. This is reasonable because tags contain much richer information than age and education and are related to user's profession and interests. Furthermore, the gender feature performs the same as random guess, hence we do not incorporate the gender feature into baselines. This suggest that the man and the woman do not have significant difference in restaurant consumption.

For RAW feature source, the amount of feature candidates is very large, e.g., hundreds of thousands of words. By borrowing the idea of feature selection in previous text classification task (Forman, 2003), we use χ^2 test to select representative features for classification. We select 4715 features for RAWWORD and 7820 features for RAWFOLLOW, which achieves best performance for prediction. Results in Table 2 suggest that RAW features are competitive for consumption level prediction, e.g., while baseline features can achieve 58.89% accuracy and 62.10% F1, RAWWORD alone can achieve 65.74% accuracy and 66.28% F1, and RAWFOLLOW alone can achieve 69.45% accuracy and 71.37% F1.

For LDAT feature source, we set the vocabulary size to 31514 and the number of topics to 200. By distilling topic semantics from tweets of users, the prediction accuracy can achieve 74.51%, which has been improved by 13.34% in contrast to RAWWORD. For LIWCT feature source, we would have expected LIWCT performs better than RAWWORD, since it categorizes the words into psychological and linguistic meaningful categories. A closer analysis of LIWCT features reveal that the vocabulary of LIWCT has relatively small overlap with the most distinguishing words in RAWWORD. For SVDF feature source, we conduct experiments with varying length of SVDF features. We find that when the length is more than 25, the performance does not increase, hence we set the length to 25 in our experiments. As presented in Table 2, accuracy of SVDF method achieves 76.73% and is much better than the result of RAWFOLLOW feature. This is also because SVDF features can capture high level interest of users.

Topic ID	Label	**Topic** (most frequent words, *translations*)	ρ	p value
13	Seafood	三文鱼,刺身,生蚝,日料,海胆,金枪鱼,鲍鱼,大闸蟹,鲜美,米其林*(salmon, sashimi, oyster, Japanese cooking, urchins, tuna, abalone, steamed crab, tasty, Michelin)*	0.85	0.0001
32	Politics	反腐,受贿,公职,公安局长,批捕,缓刑,查清,名下,收受*(anti-corruption, accept bribes, public employment, public security bureau chief, ratify the arrest, probation, investigation, name, take)*	0.82	3.81E-05
71	Luxury brands	vogue, victoria, miranda, chanel, kerr, alexander, dior, collection, louis, mcqueen *(vogue, victoria, miranda, chanel, kerr, alexander, dior, collection, louis, mcqueen)*	0.75	0.0017
198	Driving	牌照,高架,成品油,中环,远光,私车,93号,车友会,立交,油门*(vehicle license, elevated highway, product oil, median cycle, high beam, private car, No. 93 gasoline, car club, Interchange, gas)*	0.74	0.0014
120	Tennis	roger,莎拉波娃,罗杰,马卡洛娃,彭帅,阿扎伦卡,彭帅,郑洁,oba *(Roger, Sharapova, Roger, Makarova, Peng Shuai, Azarenka, Peng Shuai, Azarenka, Zheng Jie, oba)*	0.71	0.0001
45	Shanghai dialect	哪能,阿拉,今朝,老早,腔调,模子,白相,事体,闲话,辰光,喔唷*(how, I, today, previously, cool, personal loyalty, play, thing, talk, time, ugh)*	0.69	0.0260
192	Auto	车展,发动机,suv,保时捷,别克,沃尔沃,引擎,凯迪拉克,雷克萨斯,比亚迪*(auto show, engine, suv, Porsche, Buick, Volvo, engine, Cadillac, Lexus, BYD)*	0.61	0.0180
135	Mass brands	美宝莲,宝洁,阿芙,origins,美优,olay,多芬,spa,玉兰油,梦妆*(Maybelline, P&G, AFU, origins, beaubeau.com, olay, dove, spa, olay, mamonde)*	-0.77	0.0054
19	Cooking	关火,八角,豆瓣酱,土豆丝,豆角,切末,桂皮,鸡丁,炸酱面,葱油*(take off heat, aniseed, thick broad-bean sauce, shredded potato, French bean, mince, cinnamon, chicken cubes, Noodles)*	-0.81	0.0008
112	Stars	吴亦凡,朴灿烈,张艺兴,吴世勋,exo-m,金钟仁,边伯贤,黄子韬,exo-k, 泰妍*(exo Kris, Park Chan Yeol, exo Lay, Oh Se-hoon, exo-m, exo-k Kai, Baekyun, exo-m Tao, exo-k, Taeyeon)*	-0.81	9.19E-06
142	Character expression	2333, wwww, hhhh, OwO, hhhhh, 233333, QvQ, QuQ, wwwww, 0v0 *(2333, wwww, hhhh, OwO, hhhhh, 233333, QvQ, QuQ, wwwww, 0v0)*	-0.57	0.0322

Table 3: Topics sorted by absolute of Spearman correlation coefficient ρ. Topic labels are manually created.

6.2 Qualitative Analysis

In the above, we demonstrated that latent features, such as LDAT and SVDF, are the most effective features for predicting user consumption level. In this section, we conduct further hypothesis test to analyze the language divergence and the interest divergence between users at different consumption levels.

To select topics that are most correlated with consumption level, we present the formal Spearman correlation coefficient[6] test. The Spearman's coefficient ρ lies in the interval $[-1, 1]$, and a value of "+1" or "-1" indicates a perfect, positive or negative Spearman correlation. Intuitively, it is straightforward to generate two rankings of users, either by average spending or by topic preference. However, it is noted that ρ is sensitive to small value differences of both measures, and it will be difficult to obtain robust correlation values in this case. To capture the general trend, therefore, we group users according to their average spending. We sort users according to their average spending in descending order, split users equally into 100 buckets and examine the correlation at the group level.

Table 3 demonstrates the most correlated topics sorted by absolute of ρ. It is worth noting that topics with positive correlation coefficients reflect interests of high consumption level users, while topics with negative correlation coefficients reflect interests of low consumption level users. As shown in the table, users of higher consumption level prefer topics such as "Luxury Brands", while in contrast users of lower consumption level care more about "Mass Brands". This is consist with previous study on consumer behavior (Wong and Ahuvia, 1998), which shows that people buy luxury brands to take it as publicly visible markers of their economic status. Interestingly, we also find that users who speak Shanghai dialect are more likely to be of higher consumption level. We also conduct the correlation test within only Shanghai users, and we find that Shanghai dialect is also significantly correlated with high spending.

[6]http://en.wikipedia.org/wiki/Spearman's_rank_correlation_coefficient

Topic ID	Label	t (Age)	t (Gender)
13	Seafood	0.8837	$2.2599^{\ddagger}$
32	Politics	$10.1372^{\S}$	$-30.1144^{\S}$
71	Luxury brands	$-1.8778^{\dagger}$	$9.5550^{\S}$
198	Driving	$7.8684^{\S}$	$-7.2142^{\S}$
120	Tennis	$-2.5192^{\ddagger}$	-0.8891
45	Shanghai dialect	$4.7150^{\S}$	$5.9072^{\S}$
192	Auto	$2.8303^{\S}$	$-13.6531^{\S}$
135	Mass brands	-0.7032	$7.0779^{\S}$
19	Cooking	$-2.8099^{\S}$	$7.3084^{\S}$
112	Stars	$-2.7430^{\S}$	$2.7556^{\S}$
142	Character expression	$-7.4935^{\S}$	1.0283

Table 4: Topic preference t-test between different age and gender groups. "†","‡", "§" indicate the t test is significant at the level of 0.1, 0.05 and 0.01 respectively.

While on the other hand, users who use more web trending "Character expressions" tend to be of lower consumption level. Since our ground truth dataset is based on spending in restaurants, the top topics also cover food related topics, i.e., "Seafood" and "Cooking". "Seafood" is generally more expensive and hence it is an indicator of higher consumption level, while users who love "Cooking" may prefer dine in and consume less in restaurants. Moreover, high consumption level users talk more about "Politics", "Driving" and "Auto".

We conduct t test to analyze the interaction between topic preference and profile factors. Table 4 demonstrates the t test results. According to user self report age, we divide users into two groups, i.e., older than 30 years old or younger than 30 years old. A positive t indicates that elder group has higher preference on the topic. As shown in the table, elder users prefer topics such as "Politics", "Driving", "Shanghai dialect" and "Auto", and younger users prefer topics such as "Character expression", "Stars", "Luxury brands", "Cooking" and "Tennis". Generally speaking, elder people have higher spending power. Therefore, topics that elder users prefer are positively correlated with spending, while most topics that younger users prefer are negatively correlated with spending. Similarly, we conduct t test of topic preference between female users and male users. Topics such as "Character expression" and "Tennis" have no significant difference between females and males. Female users have significantly higher preference on topics such as "Luxury brands", "Cooking" and "Seafood", and male users have higher preference on topics such as "Driving" and "Auto".

Celebrity		Celebrity		Celebrity	
Dianping Coupon Shanghai	-	Beijing subway	-	Tourism related company	+
Beijing TV cusine programme	-	Reciting words app	-	International radio anchor	+
Comic dialogue player	-	Beijing SKP	+	Waldorf astoria	+
UK shopping	+	Wine related magazine	+	Charity fund	+

Table 5: Top celebrity features selected by χ^2. '+' or '-' indicates the sign of correlation.

For the celebrity features, we select the celebrities with highest χ^2 scores and present them in Table 5. As listed in the table, users of low consumption level follow celebrity such as coupon, subway, TV cuisine programme and reciting words app. On the contrary, high consumption level users follow celebrity such as traveling, wine, high grade hotels and international radio host.

For the LIWCT features, we also select the categories that have strongest correlation coefficient ρ with consumption level. Interestingly, we found that high consumption level users talk more about money (e.g., audit, cash, owe). On the contrary, low consumption level users talk more about time (e.g, end, until, season) and assent words (e.g., agree, OK, yes).

7 Related Work

User profiling aims to infer attributes of users from massive online data. Demographic attributes are widely used for ad targeting (Cheng and Cantú-Paz, 2010) and product recommendation (Wang et al., 2015). Traditional user profiling is mainly based on users' search logs or web access histories (Weber and Castillo, 2010; Hu et al., 2007). Recently, more and more researchers focus on user profiling on social media (Fink et al., 2012; Goswami et al., 2009; Tu et al., 2015).

In addition to simple demographic attributes such as gender or age, recently, researchers focus on complicated attributes such as political orientation (Pennacchiotti and Popescu, 2011), tags (Feng and Wang, 2012), locations (Backstrom et al., 2010; Pavalanathan and Eisenstein, 2015), occupation (Preoţiuc-Pietro et al., 2015a) and personal interests (Yang et al., 2011). However, the economic status related attributes have not been fully explored, which is partially due the difficulty in ground truth collection. Preoţiuc-Pietro et al. (2015b)'s work on income prediction from social media is the most relevant work to ours. Though consumption and income are related, as previous work on social economics (Brewer et al., 2012) has pointed out, " the amount of consumption in any period is not constrained to be equal to income in that period". Recent work on social economic classification (Lampos et al., 2016) is also related to our work. Their work focus on behavior features while we focus on language features and latent features. Furthermore, besides prediction task, we conduct an exploratory data analysis of language use patterns between users of different consumption levels. This work is also related to the study of food consumption on twitter (Abbar et al., 2015), and the work showed that foods mentioned in tweets are correlated with national obesity and diabetes statistics. The authors (Abbar et al., 2015) conduct experiments mainly from nutrition and health aspects, while we conduct experiments from the social economic aspect.

Our work is also related to mining heterogeneous social networks (Deng et al., 2012; Wang et al., 2011; Deng et al., 2011). Recently, many researchers focus on mapping accounts on different sites to one person (Zafarani and Liu, 2009; Liu et al., 2013) in real world. By utilizing these studies, we can link more users from Dianping and Weibo, and hence scale our task to larger dataset. Moreover, there are also works that utilize user linking feature to leverage social media knowledge for solving the cold start problem on third party website (Xiao et al., 2014; Zhang and Pennacchiotti, 2013). Since we estimate user consumption level accurately, it can also be used to solve the cold start problem in recommendation scenario.

8 Conclusion

In this paper, we focus on understanding the relationship between user's online social media behavior and offline restaurant spending. We link user's social media account and corresponding review site account, and then build consumption level ground truth based on user self report spending in their reviews. We propose the topic modeling methods and the matrix factorization methods to tackle the feature sparsity problem. We demonstrate that raw features and latent features on social media can predict consumption level with strong accuracy. The empirical analysis measures the correlation between social media features and consumption levels, and sheds light on language use differences across users at different consumption levels.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments. This work was supported by the National Grand Fundamental Research 973 Program of China under Grant No.2014CB340405 and the National Natural Science Foundation of China under Grant No.61572044. The contact author is Zhen Xiao.

References

Sofiane Abbar, Yelena Mejova, and Ingmar Weber. 2015. You tweet what you eat: Studying food consumption through twitter. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*.

Sibel Adali, Fred Sisenda, and Malik Magdon-Ismail. 2012. Actions speak as loud as words: Predicting relationships from social behavior data. In *Proceedings of the 21st International Conference on World Wide Web*.

Faiyaz Al Zamal, Wendy Liu, and Derek Ruths. 2012. Homophily and latent attribute inference: Inferring latent attributes of twitter users from neighbors. In *International AAAI Conference on Weblogs and Social Media*.

G. Alpers, A. Winzelberg, C. Classen, H. Roberts, P. Dev, C. Koopman, and C. Barr Taylor. 2005. Evaluation of computerized text analysis in an internet breast cancer support group. *Computers in Human Behavior*, 21(2).

Lars Backstrom, Eric Sun, and Cameron Marlow. 2010. Find me if you can: improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th International Conference on World Wide Web*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan).

Mike Brewer, Cormac O'Dea, et al. 2012. *Measuring living standards with income and consumption: evidence from the UK*. Institute for Social and Economic Research, University of Essex.

Haibin Cheng and Erick Cantú-Paz. 2010. Personalized click prediction in sponsored search. In *WSDM*.

Morgane Ciot, Morgan Sonderegger, and Derek Ruths. 2013. Gender inference of twitter users in non-english contexts. In *Empirical Methods in Natural Language Processing*.

Anirban Dasgupta, Maxim Gurevich, Liang Zhang, Belle Tseng, and Achint O Thomas. 2012. Overcoming browser cookie churn with clustering. In *WSDM*.

Hongbo Deng, Jiawei Han, Bo Zhao, Yintao Yu, and Cindy Xide Lin. 2011. Probabilistic topic models with biased propagation on heterogeneous information networks. In *KDD*.

Hongbo Deng, Jiawei Han, Michael R Lyu, and Irwin King. 2012. Modeling and exploiting heterogeneous bibliographic networks for expertise ranking. In *JCDL*.

Wei Feng and Jianyong Wang. 2012. Incorporating heterogeneous information for personalized tag recommendation in social tagging systems. In *KDD*, pages 1276–1284. ACM.

Clayton Fink, Jonathon Kopecky, and Maksym Morawski. 2012. Inferring gender from the content of tweets: A region specific example. In *Sixth International AAAI Conference on Weblogs and Social Media*.

Dale Fodness. 1994. Measuring tourist motivation. *Annals of tourism research*, 21(3).

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305.

Sumit Goswami, Sudeshna Sarkar, and Mayur Rustagi. 2009. Stylometric analysis of bloggers age and gender. In *Third International AAAI Conference on Weblogs and Social Media*.

Jian Hu, Hua-Jun Zeng, Hua Li, Cheng Niu, and Zheng Chen. 2007. Demographic prediction based on user's browsing behavior. In *Proceedings of the 16th International Conference on World Wide Web*.

CL Huang, CK Chung, N Hui, YC Lin, YT Seih, WC Chen, and JW Pennebaker. 2012. The development of the Chinese linguistic inquiry and word count dictionary. *Chinese Journal of Psychology*, 54(2).

SooCheong Shawn Jang, Billy Bai, Gong-Soog Hong, and Joseph T O Leary. 2004. Understanding travel expenditure patterns: a study of japanese pleasure travelers to the united states by income level. *Tourism Management*.

Vasileios Lampos, Nikolaos Aletras, Jens K Geyti, Bin Zou, and Ingemar J Cox. 2016. Inferring the socioeconomic status of social media users based on behaviour and language. In *ECIR*. Springer.

Kwan Hui Lim and Amitava Datta. 2012. Following the follower: Detecting communities with common interests on twitter. In *Proceedings of the 23rd ACM Conference on Hypertext and Social Media*.

Wendy Liu and Derek Ruths. 2013. What's in a name? using first names as features for gender inference in twitter. In *AAAI Spring Symposium: Analyzing Microtext*.

Jing Liu, Fan Zhang, Xinying Song, Young-In Song, Chin-Yew Lin, and Hsiao-Wuen Hon. 2013. What's in a name?: an unsupervised approach to link users across communities. In *WSDM*.

Dong-Phuong Nguyen, Rilana Gravel, RB Trieschnigg, and Theo Meder. 2013. How old do you think i am? a study of language and age in twitter. In *International AAAI Conference on Weblogs and Social Media*.

AC Nielson. 2012. Global trust in advertising. *NY: USA, Nielsen Media Research, ACNielsen*.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *ACL*, pages 309–319.

Umashanthi Pavalanathan and Jacob Eisenstein. 2015. Confounds and consequences in geotagged twitter data. *arXiv preprint arXiv:1506.02275*.

Marco Pennacchiotti and Ana-Maria Popescu. 2011. Democrats, republicans and starbucks afficionados: user classification in twitter. In *KDD*. ACM.

James W Pennebaker and Laura A King. 1999. Linguistic styles: language use as an individual difference. *Journal of Personality and Social Psychology*, 77(6):1296.

Daniel Preoţiuc-Pietro, Vasileios Lampos, and Nikolaos Aletras. 2015a. An analysis of the user occupational class through twitter content. In *ACL*.

Daniel Preoţiuc-Pietro, Svitlana Volkova, Vasileios Lampos, Yoram Bachrach, and Nikolaos Aletras. 2015b. Studying user income through language, behaviour and affect in social media. *PloS one*, 10(9).

Delip Rao, Michael J Paul, Clayton Fink, David Yarowsky, Timothy Oates, and Glen Coppersmith. 2011. Hierarchical bayesian models for latent attribute detection in social media. In *ICWSM*, pages 598–601.

Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57.

Alois Stutzer. 2004. The role of income aspirations in individual happiness. *Journal of Economic Behavior & Organization*, 54(1):89–109.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: LIWC and computerized text analysis methods. *Journal of Language and Social Psychology*, 29(1):24–54.

Michael Trusov, Randolph E Bucklin, and Koen Pauwels. 2009. Effects of word-of-mouth versus traditional marketing: findings from an internet social networking site. *Journal of Marketing*, 73(5):90–102.

Cunchao Tu, Zhiyuan Liu, and Maosong Sun. 2015. Prism: Profession identification in social media with personal information and community structure. In *Chinese National Conference on Social Media Processing*.

Svitlana Volkova, Glen Coppersmith, and Benjamin Van Durme. 2014. Inferring user political preferences from streaming communications. In *ACL*.

Chi Wang, Rajat Raina, David Fong, Ding Zhou, Jiawei Han, and Greg J. Badros. 2011. Learning relevance from heterogeneous social network and its application in online targeting. In *SIGIR*.

Jinpeng Wang, Wayne Xin Zhao, Yulan He, and Xiaoming Li. 2015. Leveraging product adopter information from online reviews for product recommendation. In *ICWSM*.

Ingmar Weber and Carlos Castillo. 2010. The demographics of web search. In *SIGIR*, pages 523–530. ACM.

Nancy Y Wong and Aaron C Ahuvia. 1998. Personal taste and family face: Luxury consumption in confucian and western societies. *Psychology and Marketing*, 15(5):423–441.

Yang Xiao, Wayne Xin Zhao, Kun Wang, and Zhen Xiao. 2014. Knowledge sharing via social login: Exploiting microblogging service for warming up social question answering websites. In *COLING*.

Shuang-Hong Yang, Bo Long, Alex Smola, Narayanan Sadagopan, Zhaohui Zheng, and Hongyuan Zha. 2011. Like like alike: joint friendship and interest propagation in social networks. In *WWW*, pages 537–546. ACM.

Reza Zafarani and Huan Liu. 2009. Connecting corresponding identities across communities. In *ICWSM*.

Yongzheng Zhang and Marco Pennacchiotti. 2013. Predicting purchase behaviors from social media. In *WWW*.

A Novel Fast Framework for Topic Labeling Based on Similarity-preserved Hashing

Xian-Ling Mao[♠], Yi-Jing Hao[♠], Qiang Zhou[♠], Wen-Qing Yuan[♡], Liner Yang[♣], Heyan Huang[♠*]
[♠]Department of Computer Science, Beijing Institute of Technology, China
[♡]Beijing YanZhiYouWu Technology Co., LTD, China
[♣]Department of Computer Science and Technology, Tsinghua University, China
{maoxl, hyj, qzhou, hhy63}@bit.edu.cn
yuan@yzyw-inc.com, lineryang@gmail.com

Abstract

Recently, topic modeling has been widely applied in data mining due to its powerful ability. A common, major challenge in applying such topic models to other tasks is to accurately interpret the meaning of each topic. Topic labeling, as a major interpreting method, has attracted significant attention recently. However, most of previous works only focus on the effectiveness of topic labeling, and less attention has been paid to quickly creating good topic descriptors; Meanwhile, it's hard to assign labels for new emerging topics by using most of existing methods. To solve the problems above, in this paper, we propose a novel fast topic labeling framework that casts the labeling problem as a k-nearest neighbor (KNN) search problem in probability distributions. Our experimental results show that the proposed sequential interleaving method based on locality sensitive hashing (LSH) technology is efficient in boosting the comparison speed among probability distributions, and the proposed framework can generate meaningful labels to interpret topics, including new emerging topics.

1 Introduction

A wealth of topic models have been proposed to extract interesting topics in the form of multinomial distributions from the corpus automatically, which are useful data mining tools for the statistical analysis of document collections and other discrete data. A common, major challenge in applying all such topic models is to accurately interpret the meaning of each topic. In general, it is very difficult for users to understand a topic merely based on the multinomial word distribution, especially when they are not familiar with the background knowledge. Topic labeling, which generates meaningful labels for a topic so as to facilitate topic interpretation, has attracted increasing attention recently.

Early research on topic labeling generally either select top words in the distribution as primitive labels (Blei et al., 2003; Ramage et al., 2009), or generate labels manually in a subjective manner (Mei et al., 2006; Mei and Zhai, 2005). However, it is highly desirable to automatically generate meaningful labels. Several automatical labeling methods have been proposed recently (Mei et al., 2007; Lau et al., 2010; Lau et al., 2011; Magatti et al., 2009; Mao et al., 2012; Hulpus et al., 2013; Mehdad et al., 2013; Cano et al., 2014). These existing approaches generally take following steps to generate meaningful labels for a given topic: (1) extract candidate labels; (2) rank candidate labels. First, most of existing methods extract candidate labels from a document collection by natural language processing techniques for a given topic, which is time-consuming; for example, the candidate labels in the method proposed by Mei et al. (2007) are extracted from a reference collection using chunking and statistically important bigrams, which is time-consuming. Second, most of existing approaches depend on external knowledge sources such as Wikipedia and Google Directory etc, which cannot be used to label new emerging topics learned from a stream, such as Twitter, because there is no timely information about the new emerging topics in Wikipedia or Google Directory; for example, the method proposed by Lau et al. (2011) uses Wikipedia article titles as candidate labels, which is hard to assign correct labels for the new emerging topics in Twitter because there may not be the timely articles about the new emerging topics in Wikipedia. Thus, despite the success of these works, they are either time-consuming or hard to assign labels for new emerging topics.

*Corresponding author.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3339–3348, Osaka, Japan, December 11-17 2016.

Thus, it is highly desirable to rapidly generate meaningful labels for a topic word distribution while assign labels for new emerging topics as correctly as possible. However, to the best of our knowledge, no existing method has been proposed to satisfy the demand, except that the simplest method which uses top-n words in the distribution to interpret a topic. In this paper, we study this fundamental problem which most topic models suffer from, and propose a labeling framework to rapidly label a topic while assign labels for new emerging topics as correctly as possible.

Topics can be represented as vectors, i.e. j^{th} topic, T_j = (w_{1j}, w_{2j}, ..., w_{tj})^T. Each dimension corresponds to a separate word, and its value in the vector is the probability value. Due to the characters of distributions, we have two observations: (1) Similar topics all have higher probability values over the relevant words, thus a topic is near its similar topics in a probability vector set; For example, the topic "Military" and "Air Force" have both higher probability values over words like "soldier", "missile" and "death" than that over other words, except that the topic "Air Force" has higher probability values over words like "warcraft", "F22" and "pilot", than the topic "Military"; (2) Two different probability distributions might have the same label, because the inherent semantics of a distribution (i.e. topic) is not only reflected by the concrete probability values, but also by the focused words; for example, the following two topics don't have the same distributions, but they are the same topic, and the label is "Military".

	tank	bomb	...	death	take	cup	gun
$topic_1$	0.18	0.10	...	0.11	0.0002	0.007	0.08
$topic_2$	0.17	0.09	...	0.12	0.0003	0.009	0.07

Intuitively, when an event occurs, the information about the event exists in all kinds of sources (labeled data and unlabeled data), such as news, forums and social networks. Thus, these sources are parallel information channels. It is believed that the labels in labeled data will cover most topics in unlabeled data. Through the idea of parallel information channels, in a domain, if there is a database where each record consists of a topic word distribution and its corresponding label [1], and the database is updated uninterruptedly by learning from the latest labeled data as soon as possible; we can interpret a given topic by the labels of k-nearest distributions in the database (i.e. probability vector set) while assign labels for new emerging topics as correctly as possible. If the database is large, timely, and cover most kinds of topics in a domain, it's a reasonable labeling solution to label topics in the domain.

There are two challenges to achieve the intuition: (1) how to quickly find the k-nearest neighbours in a large probability vector set, given a high-dimensional topic word distribution; (2) how to update the database from the latest data as soon as possible.

We proposed a fast labeling framework to solve the two challenges: (1) to finding k-nearest distributions quickly, we use locality sensitive hashing (LSH) as a dimensionality reduction technique to accelerate distribution similarity comparison; (2) to accelerate the update speed of the database, we modify the batch learning algorithm for Labeled LDA to obtain an online learning algorithm.

2 Related work

In most existing research effects on statistical topic modeling, people generally either select top words in the distribution as primitive labels (Blei et al., 2003; Ramage et al., 2009; Ramage et al., 2011), or generate more meaningful labels manually in a subjective manner (Mei et al., 2006; Mei and Zhai, 2005). However, extracting top terms is not very useful to interpret the coherent meaning of a topic (Mei et al., 2007). Meanwhile manually generated labels require lots of human effort to generate such labels, and can easily be biased towards the user's subjective opinions.

Thus, several automatical labeling methods have been proposed recently. In 2007, Mei et al. first proposed probabilistic approaches to automatically label multinomial topics by extracting a set of candidate labels from a reference collection using chunking and statistically important bigrams, and the top ranked labels were chosen to represent the topic (Mei et al., 2007). Magatti et al. (2009) introduced an algorithm to label topics automatically according to a given category hierarchy. The hierarchy was obtained from the Google Directory and the OpenOffice English Thesaurus. The most similar label is assigned to the topic. Lau et al. (2010) proposed to label a topic via selecting one of the top-10 topic terms to label the

[1] Different distributions can have same label

overall topic by a reranking model. Different with their previous method (Lau et al., 2010), Lau et al. (2011) enlarged the candidate labels by making use of Wikipedia article titles.

Mao et al. (2012) proposed two effective algorithms that automatically assign concise labels to each topic in a hierarchy by exploiting sibling and parent-child relations among topics. The structured data in DBpedia is used to label topics (Hulpus et al., 2013). Mehdad et al. (2013) introduced a topic labeling approach that assigns the most representative phrases for a given set of sentences covering the same topic. Cano et al. (2014) proposed a summarisation framework to label the topics learned from Twitter, which is independent of external sources and only relies on the identification of dominant terms in documents related to the latent topic. Word embedding is also used to help label topics (Jin et al., 2016). Interestingly, Aletras and Stevenson (2013) proposed to label topics with images rather than text. Candidate images for each topic are retrieved from the web by querying a search engine using the top-n terms. The most suitable image is selected by using a graph-based algorithm.

Overall, these methods first extract the candidate labels, then rank these labels according to corresponding scoring function. The labeling framework is showed in Figure 1 (a). Despite the success of these works, they are either time-consuming or hard to assign labels for new emerging topics. Thus, in this paper, we will focus on the problems above, and proposed our fast topic labeling framework.

Figure 1: (a) Traditional topic labeling framework and (b) Fast topic labeling framework.

3 Fast Topic Labeling Framework

In this paper, we will study topic labeling problem from a different perspective, i.e. cast it as k-nearest neighbor (KNN) search problem in a probability vector set. As shown in Figure 1 (b), our framework consists of two main components described in the following sections.

3.1 Online Labeled LDA

One important component of our proposed framework is to construct a "topic-label" database which consists of probability distributions over words and corresponding labels, denoted as $DB_{m,n}$, m is the number of topics, and n is the size of vocabulary. We can use Labeled LDA (LLDA) proposed by Ramage et al. (2009) to construct the "topic-label" database. LLDA models the documents with labels, and obtains a probability distribution for each label. The training algorithm for LLDA runs in batch mode, and cannot update the database timely. However, in order that the framework can assign labels for new emerging topics as correctly as possible, the proposed framework needs to process latest labeled data timely, and add gradually new learned records into the "topic-label" database. Thus, we propose a online algorithm for Labeled LDA, called OLLDA, to accelerate the update speed of "topic-label" database.

We define the vector of corresponding labels of document d to be $\boldsymbol{\psi}^{(d)}$, and the number of labels to be M_d, i.e. $M_d = |\boldsymbol{\psi}^{(d)}|$. Similar to the batch variational inference for LDA (Blei et al., 2003), the batch variational inference for Labeled LDA approximates the true posterior by a simpler distribution $q(z, \theta, \beta)$, which is indexed by a set of free parameters. These parameters are optimized by maximizing the lower bound:

$$\log p(\boldsymbol{w}|\alpha, \eta, \Psi) \geq L(\boldsymbol{w}, \phi, \gamma, \boldsymbol{\lambda}, \Psi) \triangleq \mathbb{E}_q[\log p(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\beta}, \Psi|\alpha, \eta)] - \mathbb{E}_q[\log q(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\beta})]. \quad (1)$$

To maximize the lower bound, we have to minimize the KL divergence between $q(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\beta})$ and the

posterior $p(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\beta}, \Psi|\boldsymbol{w}, \alpha, \eta)$. The distribution $q(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\beta})$ can be fully factorized into the form:

$$q(z_{di} = l) = \phi_{dw_{di}l}; \qquad q(\theta_d) = \text{Dir}(\theta_d; \gamma_d); \qquad q(\beta_l) = \text{Dir}(\beta_l; \gamma_l), \tag{2}$$

where l is the index of a label, and also the index of corresponding topic. The posterior over the per-word topic assignments z is parameterized by ϕ, the posterior over the per-document topic weights θ is parameterized by $\boldsymbol{\gamma}$, and the posterior over the topics β is parameterized by $\boldsymbol{\lambda}$. Equation (1) factorizes to

$$\begin{aligned}
L(\boldsymbol{w}, \boldsymbol{\phi}, \boldsymbol{\gamma}, \boldsymbol{\lambda}, \Psi) = & \sum_{d, z_d \in \Psi} \{\mathbb{E}_q[\log p(w_d|\theta_d, z_d, \beta)] + \mathbb{E}_q[\log p(z_d|\theta_d)] - \mathbb{E}_q[\log q(z_d)] + \mathbb{E}_q[\log p(\theta_d|\alpha)] \\
& - \mathbb{E}_q[\log q(\theta_d)] + (\mathbb{E}_q[\log p(\beta|\eta)] - \mathbb{E}_q[\log q(\beta)])/D\} \\
= & \sum_d \sum_w n_{dw} \sum_{l \in \boldsymbol{\psi}^{(d)}} \phi_{dwl}(\mathbb{E}_q[\log \theta_{dl}] + \mathbb{E}_q[\log \beta_{lw}] - \log \phi_{dwl}) - \log \Gamma(\sum_{l \in \boldsymbol{\psi}^{(d)}} \gamma_{dl}) + \sum_{l \in \boldsymbol{\psi}^{(d)}} \\
& (\alpha - \gamma_{dl})\mathbb{E}_q[\log \theta_{dl}] + \log \Gamma(\gamma_{dl}) + (\sum_{l \in \boldsymbol{\psi}^{(d)}} - \log \Gamma \sum_w \lambda_{lw}) + \sum_w (\eta - \lambda_{lw})\mathbb{E}_q[\log \beta_{lw}] \\
& + \log \Gamma(\lambda_{lw}))/D + \log \Gamma(M_d \alpha) - M_d \log \Gamma(\alpha) + (\log \Gamma(W\eta) - W \log \Gamma(\eta))/D \\
\triangleq & \sum_d l(n_d, \phi_d, \gamma_d, \boldsymbol{\lambda}, \psi^{(d)}),
\end{aligned} \tag{3}$$

where W is the size of the vocabulary and D is the number of documents. $l(n_d, \phi_d, \gamma_d, \boldsymbol{\lambda}, \psi^{(d)})$ denotes the contribution of document d to the lower bound of Formula (1). This reveals that the variational objective relies only on n_{dw}, the number of times word w appears in document d. Thus, an online inference algorithm for Labeled LDA can be derived.

We can use coordinate ascent to optimize L over the variational parameters $\boldsymbol{\phi}$, $\boldsymbol{\gamma}$, $\boldsymbol{\lambda}$:

$$\phi_{dwl} \propto exp\{\mathbb{E}_q[\log \theta_{dl}] + \mathbb{E}_q[\log \beta_{lw}]\}; \qquad \gamma_{dl} = \alpha + \sum_w n_{dw}\phi_{dwl}; \qquad \lambda_{lw} = \eta + \sum_d n_{dw}\phi_{dwl}. \tag{4}$$

As the d^{th} vector of word counts n_d is observed, we perform an EM algorithm to obtain optimal parameter values. Similar to the work (Hoffman et al., 2010), we use the weight $\rho_d \triangleq (\tau_0 + d)^{-\kappa}$ to control the rate at which old values of $\boldsymbol{\lambda}$ are forgotten and $\tau_0 \geq 0$ slows down the early iterations of the algorithm, where $\kappa \in (0.5, 1]$ is needed to guarantee convergence. The proposed online variational inference for Labeled LDA (OLLDA) is described in Algorithm 1.

Algorithm 1 Online Variational Inference for Labeled LDA

```
1: ρ_d = (τ_0 + d)^{-κ}
2: Initialize λ randomly.
3: for d = 0 to ∞ do
4:     E step:
5:     Initialize γ_dl = 1.
6:     repeat
7:         for each word w in document d do
8:             for each label l of document d do
9:                 Set φ_dwl ∝ exp{E_q[log θ_dl] + E_q[log β_lw]}
10:                Set γ_dl = α + Σ_w φ_dwl n_dw
11:            end for
12:        end for
13:    until (1/M_d) Σ_l |change in γ_dl| < 0.00001
14:    M step:
15:    Compute λ̃_lw = η + D n_dw φ_dwl
16:    Set λ = (1 − ρ_d)λ + ρ_d λ̃.
17: end for
```

3.2 Distribution Similarity Ranking

After constructing the "topic-label" database, we have to rank distributions for a given topic. To compute similarity between two distributions, there are many metrics, such as Kullback-Leibler divergence

(KL divergence, KL) and Jensen-Shannon Divergence (JSD). Generally speaking, the vocabulary of a distribution is large (over 100K) while the number of records in database is also large, thus it's costly in computing and storage. For example, assume that the number of vocabulary is 100,000, the number of topics is 100,000, and the size of a float is 4 bytes, we need at least 40G space to store distributions.

Locality Sensitive Hashing (LSH), as a dimensionality reduction technique, has been widely applied in large-scale data mining, such as near-duplicate webpage detection (Manku et al., 2007) and image retrieval (Vogel and Schiele, 2007). The key idea of LSH is to assign a number to each point in a metric space by a function h uniformly selected from a family of hashing functions H so that the probability of collision (i.e., assigned by an identical number) is much higher for points close to each other than those far apart (Charikar, 2002). LSH functions involve the creation of short signatures (fingerprints) for each vector (point) in space such that those vectors that are closer to each other are more likely to have similar fingerprints. LSH functions are generally based on randomized algorithms and are probabilistic.

Based on LSH technology, the proposed distribution similarity ranking method will take the following three steps, described below.

3.2.1 Initial Ranking

To accelerate distribution similarity comparison and decrease the storage space, we will choose two representative types of LSH, cosine hash family (**Simhash** (Charikar, 2002)) and the Euclidean hash family (**P-stable LSH** (Datar et al., 2004)), as initial distribution similarity metrics.

Given a probability distribution (i.e. a topic), after obtained the ranked list of topics by a distribution similarity metric (such as Simhash, P-stable LSH), the corresponding labels of returned topics can be used to label the given topic. Because it's useless to assign too many labels to a topic, our systems will return top-20 topics for each given topic.

3.2.2 Re-ranking

A distribution similarity metric does not consider the inherent property of topics. For example, two different points in a probability vector set might have the same label. Thus, we have to consider the nature of topics. Because top-n words of similar topics should be similar, so we will re-rank the order of topics by making use of overlap similarity between top-n word set of the given topic and each in top-20 returned topics. Top-n, as a parameter, will be selected by experiments.

3.2.3 Sequential Interleaving of Two Lists

Given two ranking lists generated by different methods, to obtain benefit from the both results, we consider an interleaving process on these two ranking lists to obtain a better ranking list. Lots of works have been done to interleave two ranking lists of documents in information retrieval area (Hofmann et al., 2011; Hofmann et al., 2012; Chukllin et al., 2015). We borrow the interleaving idea in information retrieval area to handle this problem, and propose a a novel interleaving algorithm, called Sequential Interleaving, to obtain a better ranking list by merging two ranking lists.

In a retrieved list generated by an algorithm, the ranking position of a topic in the list can be treated as a reflection of "confidence level", which is how "good" the algorithm thinks the topic is similar to the given topic. Intuitively, the confidence level $\mathrm{CL}(p, L_r)$ for the probability distribution p of the position r in the ranking list L, should be an inverse function form of the ranking position r. To define the function form of $\mathrm{CL}(p, L_r)$, we inspect the metrics used to evaluating a retrieved list in information retrieval. DCG is a ranking-aware metric which can effectively evaluate how relevant a ranking list is for the query. Its widely-used binary value form (Chapelle et al., 2012) is defined as:

$$\mathrm{DCG}(L) = \sum_{r=1}^{len(L)} \frac{rel_{L_r}}{\log_2(r+1)} \tag{5}$$

where rel_{L_r} is the relevance of the document ranked at r in the ranking list L. In this formula, DCG reflects total confidence level of all documents in the ranking list, thus we can define $\mathrm{CL}(p, L_r)$ as:

$$\mathrm{CL}(p, L_r) = \begin{cases} \frac{1}{\log_2(r+1)} & p \in L, \\ 0 & p \notin L. \end{cases} \tag{6}$$

where p is a probability distribution, and r is the ranking postion of p in the ranking list L.

Given a topic, there are two ranking lists of top-k similar probability distributions (i.e. topics), and the goal is to combine the two ranking lists to obtain a better top-k ranking list. We first obtain the union of the two ranking lists L_1 and L_2, then for each probability distribution p in the union, to compute the total confidence level of p by the following simple formula:

$$\text{TotalCL}(p, L_1, L_2) = \alpha \sum_{L_{1_i}=p} \text{CL}(p, L_{1_i}) + (1-\alpha) \sum_{L_{2_j}=p} \text{CL}(p, L_{2_j}) \tag{7}$$

where $L_{1_i} = p$ denotes the probability distribution p is at the position i in the ranking list L_1, and α is a prior weighting factor. Finally, we sort all the probability distributions by using $\text{TotalCL}(p, L_1, L_2)$, and then give the top-k similar interleaved probability distributions as final ranking result; and thus the labels of these distributions are the top-k candidate labels for the given topic. The proposed algorithm is called Sequential Interleaving. Because the proposed framework requires the high speed of k-nearest neighbor search for a given topic, only locality sensitive hashing algorithms satisfy the condition, thus we combine the re-ranking lists of Simhash and P-stable by proposed Sequential Interleaving method to obtain a ranking list, and abbreviated as si. In this paper, we fairly treat the re-ranking results of Simhash and P-stable by setting α = 0.5.

4 Experiments and results

In this section, we present the results of the efficiency and effectiveness of the proposed method over three data sets.

4.1 Experiment Setup

Data Sets: We explore three different genres of data sets: the Simulated data (**SIMU**), the Conference proceedings (**CONF**), the Twitter dataset (**TW**). To construct the first dataset, we simulated 10,000 distributions over 10,000 words ($DB_{10000,10000}$), and 100,000 distributions over 10,000 words ($DB_{100000,10000}$). After collecting 2,924 fullpapers of four conference (SIGIR, SIGKDD, CIKM, and WWW) proceedings from the year 2010 to 2013, from Google Scholar, conference u-disks and authors' homepage, we obtained the second dataset. The last dataset contains 2.1G microblogs with 3503 hashtags, which removed microblogs without hashtag and hashtags whose idf is less than 50, downloaded from Twitter website in six days. We built "topic-label" database over the last two data sets by our OLLDA algorithm. Specifically, for **CONF**, we trained OLLDA over the data from CIKM2013 to obtain probability distributions with labels as test queries, and trained OLLDA over the remaining data as "topic-label" database; for **TW**, to increase the number of points with same labels in a probability vector set, we split the dataset into 12 pieces according to the time order, and trained separately OLLDA over the $1^{th} \sim 11^{th}$ pieces as "topic-label" database, and over the 12^{th} piece as test queries. All test queries in two datasets can be as the new emerging topics. After training, the statistics of data sets are shown in Table 1 [2].

Baselines: Because proposed framework is totally different with existing methods, e.g., the input of most of existing methods is a topic and a collection which generates the topic, and the input of the proposed framework is only a topic, thus, we cannot compare them directly. Meanwhile there is no related work that focus on efficiency of topic labeling. Essentially, the core of the proposed framework is the comparison among distributions, thus, we choose KL divergence (KL) and Jensen-Shannon Divergence (JSD) as distribution similarity metrics in our framework, as baselines. All methods in this paper are denoted as $FR_{metric,ranking\_stage}$, where *metric* means which distribution similarity metric to choose, and *ranking_stage* means *Initial Ranking* or *Re-ranking*. We use "1" to denote "Initial Ranking" and "2" denote "Re-ranking". For example, assume that the distribution similarity metric is JSD, and just use initial ranking, the method can be denoted as $FR_{jsd,1}$. The sequential interleaving result of the ranking lists of $FR_{sh,2}$ and $FR_{ps,2}$ is denoted as FR_{si}.

All experiments were conducted on a server with dual 6-core Intel i7 cpus with 3.4Ghz, 32G memory.

[2] All "topic-label" databases have been published at "https://github.com/TopicLabeling/tldb".

4.2 Efficiency of Distribution Similarity Metrics over Simulated Data

We sample respectively 20 probability distributions from $DB_{10000,10000}$ and $DB_{100000,10000}$ as test queries and rank the distributions in corresponding SIMU dataset, then compute avarage time per query and space cost. Because the main cost in our distribution similarity ranking is spent in "Initial Ranking" stage, and the cost of re-ranking is little, thus only report the results in "Initial Ranking" stage. Table 2 shows a detailed comparison of $FR_{kl,1}$, $FR_{jsd,1}$, $FR_{sh,1}$ and $FR_{ps,1}$ on two SIMU datasets. We can make two observations from the table: the computing time and storage space of $FR_{ps,1}$ and $FR_{sh,1}$, are significantly less than that of baseline methods. In a word, the efficiency of LSH significantly outperforms the baselines in terms of time and space.

Table 1: The statistics of CONF and TW data sets.

	CONF	TW
Num. of Vocabulary	25,160	189,841
Num. of Labels	586	3,503
Num. of Distributions	957	12,139

Table 2: The efficiency of four methods over the SIMU datasets.

	$\mathbf{DB_{10000,10000}}$		$\mathbf{DB_{100000,10000}}$	
Methods	**Time (s)**	**Space (M)**	**Time (s)**	**Space (M)**
$FR_{kl,1}$	5.761	1,230.830	546.514	12,0206.272
$FR_{jsd,1}$	5.051	1,206.972	656.120	12,211.392
$FR_{sh,1}$	0.078	**115.996**	**0.504**	**674.750**
$FR_{ps,1}$	**0.075**	411.028	0.689	4,020.030

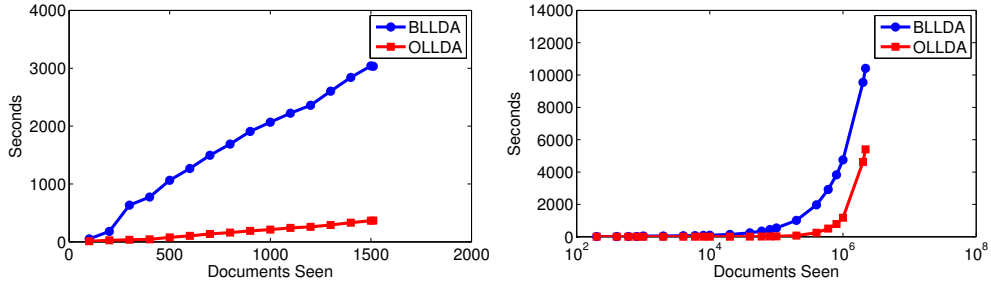
4.3 Performance of Proposed Framework over Real-world Data

As described above, the distributions in our "topic-label" database has their corresponding labels, which are used as standard labels (ground truth). We first show some sample results of our fast labeling methods in Table 4. For comparison, we also show the baselines' labels and standard labels for the same topics. It is clear that the rapidly generated labels can all capture the meaning of the topic to some extent; indeed, most of them are as good as standard labels (e.g., "social network" and "michael jackson"), though some are not (e.g., "text classification") but they are similar to standard labels.

4.3.1 Efficiency of OLLDA

OLLDA has several learning parameters: $\kappa \in (0.5, 1]$ and $\tau_0 \geq 0$. Similar to Hoffman's parameter choosing method (Hoffman et al., 2010), we also set $\kappa = 0.5$ and $\tau_0 = 64$ as the best learning parameter settings for both corpora.

In this experiment, we evaluated the efficiency of OLLDA, compared with batch LLDA (BLLDA). We simulated the situation that documents are coming in a stream. For our OLLDA, the model parameters are updated every time a new document arrives. We computed total training time cost at some points of the seen documents using OLLDA. As for BLLDA, we computed the training time cost at the same points, and note that each run has to use all of the documents previously seen.



(a) Conf (b) Twitter

Figure 2: Training Time On Two Real Datasets

When a new document arrives, BLLDA has to run over all observed documents for many iterations again. Since the time for each iteration grows with the number of documents, the total time for BLLDA grows fast. However, OLLDA only processes the new coming document and update parameters. Thus, as Figure 2 shows, BLLDA costs much more time to get the new parameters compared with OLLDA,

especially when the number of observed documents is large. When running over the entire Conf corpus, the training time of OLLDA is less than 400 seconds, while BLLDA takes more than 3,000 seconds, which is about 8 times longer. Over TW dataset, BLLDA takes more than 10,000 seconds, while OLLDA takes less than 6,000 seconds.

In Figure 2 (b), when the number of observed documents is large, the time cost increases quickly for OLLDA and BLLDA, because of the greater number of parameters and IO cost. However, OLLDA still performs better than BLLDA. If we use mini-batch trick and other optimize technologies in OLLDA, the efficiency of OLLDA will get greater improvement.

4.3.2 Effectiveness of Distribution Similarity Ranking

We compute the following performance metrics over CONF and TW dataset: (1) *Match at top N results (**Match@N**)*, which indicates whether the top N results contain any correct labels; (2) *Precision at top N results (**P@N**)*.

Table 3: The effectiveness of four methods over the CONF and TW datasets.

Datasets	Methods	Match@N				P@N			
		N=1	N=3	N=5	N=10	N=1	N=3	N=5	N=10
CONF	$FR_{kl,1}$	0.0000	0.0000	0.0069	0.0069	0.0000	0.0000	0.0014	0.0007
	$FR_{jsd,1}$	**0.1458**	**0.2569**	**0.2847**	**0.3819**	**0.1458**	**0.0949**	**0.0667**	**0.0458**
	$FR_{sh,1}$	0.0347	0.0556	0.0764	0.1667	0.0347	0.0208	0.0181	0.0181
	$FR_{sh,2}$	0.0833	0.1458	0.1597	0.1806	0.0833	0.0556	0.0375	0.0236
	$FR_{ps,1}$	0.0000	0.0208	0.0278	0.0556	0.0000	0.0069	0.0056	0.0056
	$FR_{ps,2}$	0.0556	0.0694	0.0903	0.1042	0.0556	0.0255	0.0208	0.0125
	FR_{si}	0.0972	0.1528	0.1736	0.2153	0.0972	0.0556	0.0389	0.0271
TW	$FR_{kl,1}$	0.0500	0.0500	0.1000	0.1000	0.0500	0.0500	0.0500	0.0350
	$FR_{jsd,1}$	0.9000	0.9000	**0.9500**	**1.0000**	0.9000	0.6000	**0.5300**	0.4550
	$FR_{sh,1}$	0.9000	**0.9500**	**0.9500**	0.9500	0.9000	**0.6167**	**0.5300**	**0.4600**
	$FR_{sh,2}$	0.9000	**0.9500**	**0.9500**	0.9500	0.9000	**0.6167**	**0.5300**	0.4500
	$FR_{ps,1}$	0.8500	0.9000	0.9000	0.9000	0.8500	0.6000	0.5200	0.4550
	$FR_{ps,2}$	0.9000	0.9000	0.9000	0.9000	0.9000	0.5833	0.5200	0.4500
	FR_{si}	**0.9500**	**0.9500**	**0.9500**	**1.0000**	**0.9500**	**0.6167**	**0.5300**	**0.4600**

Table 4: Sample topics and algorithm-generated labels from TW dataset.

	CONF			TW		
Stand. Label	**recommender_systems**	**social_network**	**text_classification**	**michaeljackson**	**treat**	**rugby**
$\mathbf{FR_{kl,1}}$ **(Top-3)**	personal_information_management recommender location_selection	location_selection sensemaking spreadsheets	personal_information_management image_clustering recommender	glamourkills twitdraw 140kingofpop	voss bonjovi tinychat	ff brazilmissesdemi dietalk
$\mathbf{FR_{jsd,1}}$ **(Top-3)**	collaborative_filtering matrix_factorization collaborative_filtering	social_networks social_networks privacy	query_classification graph_regularization active_learning	michaeljackson michaeljackson thisisit	trick fb happyhalloween	fb nhl rugby
$\mathbf{FR_{sh,1}}$ **(Top-3)**	collaborative_filtering matrix_factorization recommender_systems	social_networks social_networks social_search	query_classification domain_adaptation hierarchical_classification	michaeljackson michaeljackson uknowurblackwhen	trick story09 yeg	rugby rugby rugby
$\mathbf{FR_{ps,1}}$ **(Top-3)**	diversity personalization evaluation	topic_model social-network_analysis social_network	semi-supervised_learning machine_learning evaluation	michaeljackson mj fb	trick fb love	fb rugby fb
$\mathbf{FR_{si}}$ **(Top-3)**	collaborative_filtering matrix_factorization personalization	social_networks social-network_analysis social_networks	query_classification semi-supervised_learning hierarchical_classification	michaeljackson michaeljackson michaeljackson	trick trick treat	rugby rugby rugby
Top-10	users item rating items ratings recommendation collaborative filtering methods recommender	social network graph users content group nodes labels such people	training documents domain classification articles method text test used terms	michaeljackson jackson michael shirts ringtone jacko ringtones movie kingofpop our	brother's wapo today incredible trife strokes things when somebody sense	rugby davidarchie uia game beginnings try great after fatal be4

From Table 3, we can make several observations: (1) For both datasets, $FR_{x,2}$ performs better than $FR_{x,1}$, *x* denotes *sh* or *ps*, which shows that the step "Re-ranking" is effective. (2) For both datasets, FR_{si} performs better than $FR_{sh,2}$ and $FR_{ps,2}$, which shows that the step "Sequential Interleaving" is effective. (3) For both datasets, $FR_{jsd,1}$ performs better than $FR_{sh,1}$ and $FR_{ps,1}$ in most cases, which shows that JSD is good distribution similarity comparision method. However, the time cost of $FR_{jsd,1}$ is higher than LSH methods, showed in Table 5. (4) For both datasets, the performance of $FR_{ps,1}$ is alway poorer than that of $FR_{sh,1}$. Given a probability distribution $\vec{p}$, assume that $\mathbf{P}$ is the set of all probability

distributions, and the set $S_1 = \{p_i | \cos^{-1} \frac{\vec{p}_i \cdot \vec{p}}{\|\vec{p}_i\| \|\vec{p}\|} \leq \theta, p_i \in \mathbf{P}\}$, and assume that $\vec{p}_0$ is a probability distribution, which satisfies $p_0 = argmin_{p_i \in S_1} \|\vec{p}_i - \vec{p}\|$, and the set $S_2 = \{p_i | \ \|\vec{p}_i - \vec{p}\| \leq \ \|\vec{p}_0 - \vec{p}\|\}$. It's easy to prove that $S_2 \subseteq S_1$. That means: the point p_0 locates in the bound of S_1 and S_2, however, the set for Euclidean-based methods is subset of the one for angle-based methods, which means that the ability of finding KNN points for angle-based methods are better than the one for Euclidean-based methods in probability distributions. Thus, $FR_{sh,1}$ always performs better than $FR_{ps,1}$. (5) For CONF dataset, the metric values of all methods are not high. Because our metrics only count the labels which are same as standard labels, and ignore the similar labels. Furthermore, the ratio ($\frac{Num.of Distributions}{Num.of Labels}$) in CONF is small (showed in Table 1). Small ratio means that a lable has fewer points in a probability vector set, thus it's natural that the values of all metrics are very low. Thus, we evaluate all ranking results judged manually by three students, and take use of voting method to obtain the final results, showed in Table 6. The table shows that the metric values increase, and FR_{si} performs best. (6) For TW dataset, the performance of FR_{si} is the best among all methods. The improvements are significant by t-test at the 95% significance level. Thus, the overall results show that proposed distribution similarity ranking is effective. Meanwhile, because the test queries in two datasets can be as the new emerging topics, thus the results also show that the proposed method can handle the new emerging topics.

Table 5: The efficiency over the CONF and TW datasets.

	CONF	TW
Methods	**Time (s)**	**Time (s)**
$FR_{kl,1}$	0.0727	1.9823
$FR_{jsd,1}$	0.2310	10.0959
$FR_{sh,1}$	**0.0024**	**0.0541**
$FR_{ps,1}$	0.0260	0.2594

Table 6: The results of human judge over CONF dataset.

Methods	**Match@N**				**P@N**			
	N=1	**N=3**	**N=5**	**N=10**	**N=1**	**N=3**	**N=5**	**N=10**
$FR_{kl,1}$	0.100	0.200	0.200	0.500	0.100	0.067	0.050	0.065
$FR_{jsd,1}$	0.350	0.650	0.700	0.900	0.350	0.467	0.360	0.295
$FR_{sh,1}$	0.350	0.600	0.700	0.750	0.350	0.300	0.280	0.210
$FR_{ps,1}$	0.200	0.250	0.450	0.800	0.200	0.167	0.150	0.175
FR_{si}	**0.467**	**0.750**	**0.831**	**0.950**	**0.467**	**0.530**	**0.455**	**0.375**

4.3.3 Efficiency of Distribution Similarity Ranking

For each query in test set, we rank the distributions in corresponding dataset, then compute avarage time per query. Table 5 shows a detailed comparison of $FR_{sh,1}$, $FR_{ps,1}$, $FR_{kl,1}$ and $FR_{jsd,1}$ on **TW** and **CONF**. We can make following observations: the computing time of $FR_{ps,1}$ and $FR_{sh,1}$, are significantly less than other methods. In a word, the efficiency of the proposed distribution similarity ranking method based on LSH significantly outperforms the baselines in terms of time.

5 Conclusion

In this paper, we cast the rapid labeling problem as a k-nearest neighbor (KNN) search problem among existing "topic-label" database, which is updated uninterruptedly by online learning from the latest data. The experimental results show that the proposed framework can generate meaningful labels that are useful for interpreting topics in real time.

Acknowledgements

The work was supported by 863 Program of China (No. 2015AA015404) and NSFC (No. 61402036).

References

Nikolaos Aletras and Mark Stevenson. 2013. Representing topics using images. In *Proceedings of NAACL-HLT*, pages 158–167.

D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

A Elizabeth Cano, Yulan He, and Ruifeng Xu. 2014. Automatic labelling of topic models learned from twitter by summarisation. *ACL 2014*.

Olivier Chapelle, Thorsten Joachims, Filip Radlinski, and Yisong Yue. 2012. Large-scale validation and analysis of interleaved search evaluation. *ACM Transactions on Information Systems (TOIS)*, 30(1):6.

Moses S Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM.

Aleksandr Chukllin, ANNE SCHUTH, KE ZHOU, and MAARTEN DE RIJKE. 2015. A comparative analysis of interleaving methods for aggregated search. *ACM Trans. Inf. Syst*, 33(2):5.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM.

Matthew Hoffman, Francis R Bach, and David M Blei. 2010. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864.

Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2011. A probabilistic method for inferring preferences from clicks. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 249–258. ACM.

Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2012. Estimating interleaved comparison outcomes from historical click data. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1779–1783. ACM.

Ioana Hulpus, Conor Hayes, Marcel Karnstedt, and Derek Greene. 2013. Unsupervised graph-based topic labelling using dbpedia. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 465–474. ACM.

Zhipeng Jin, Qiudan Li, Can Wang, Daniel D Zeng, and Lei Wang. 2016. Labelling topics in weibo using word embedding and graph-based method. In *2016 International Conference on Information Systems Engineering (ICISE)*, pages 34–37. IEEE.

J.H. Lau, D. Newman, S. Karimi, and T. Baldwin. 2010. Best topic word selection for topic labelling. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 605–613. Association for Computational Linguistics.

J.H. Lau, K. Grieser, D. Newman, and T. Baldwin. 2011. Automatic labelling of topic models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1536–1545. Association for Computational Linguistics.

D. Magatti, S. Calegari, D. Ciucci, and F. Stella. 2009. Automatic labeling of topics. In *Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on*, pages 1227–1232. IEEE.

Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. 2007. Detecting near-duplicates for web crawling. In *Proceedings of the 16th international conference on World Wide Web*, pages 141–150. ACM.

Xian-Ling Mao, Zhao-Yan Ming, Zheng-Jun Zha, Tat-Seng Chua, Hongfei Yan, and Xiaoming Li. 2012. Automatic labeling hierarchical topics. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2383–2386. ACM.

Yashar Mehdad, Giuseppe Carenini, Raymond T Ng, and Shafiq Joty. 2013. Towards topic labeling with phrase entailment and aggregation. In *Proceedings of NAACL-HLT*, pages 179–189.

Q. Mei and C.X. Zhai. 2005. Discovering evolutionary theme patterns from text: an exploration of temporal text mining. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 198–207. ACM.

Q. Mei, C. Liu, H. Su, and C.X. Zhai. 2006. A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In *Proceedings of the 15th international conference on World Wide Web*, pages 533–542. ACM.

Q. Mei, X. Shen, and C.X. Zhai. 2007. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 490–499. ACM.

D. Ramage, P. Heymann, C.D. Manning, and H. Garcia-Molina. 2009. Clustering the tagged web. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 54–63. ACM.

D. Ramage, C.D. Manning, and S. Dumais. 2011. Partially labeled topic models for interpretable text mining. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–465. ACM.

Julia Vogel and Bernt Schiele. 2007. Semantic modeling of natural scenes for content-based image retrieval. *International Journal of Computer Vision*, 72(2):133–157.

Sequence to Backward and Forward Sequences: A Content-Introducing Approach to Generative Short-Text Conversation

Lili Mou,[1] **Yiping Song,**[2] **Rui Yan,**[3] **Ge Li,**[1,*] **Lu Zhang,**[1] **Zhi Jin**[1,*]
[1]Key Laboratory of High Confidence Software Technologies (Peking University), MoE, China
Institute of Software, Peking University, China *Corresponding authors
[2]Institute of Network Computing and Information Systems, Peking Univerity, China
[3]Institute of Computer Science and Technology, Peking University, China
doublepower.mou@gmail.com
{songyiping,ruiyan,lige,zhanglucs,zhijin}@pku.edu.cn

Abstract

Using neural networks to generate replies in human-computer dialogue systems is attracting increasing attention over the past few years. However, the performance is not satisfactory: the neural network tends to generate safe, universally relevant replies which carry little meaning. In this paper, we propose a content-introducing approach to neural network-based generative dialogue systems. We first use pointwise mutual information (PMI) to predict a noun as a keyword, reflecting the main gist of the reply. We then propose `seq2BF`, a "*sequence to backward and forward sequences*" model, which generates a reply containing the given keyword. Experimental results show that our approach significantly outperforms traditional sequence-to-sequence models in terms of human evaluation and the entropy measure, and that the predicted keyword can appear at an appropriate position in the reply.

1 Introduction

Automatic human-computer conversation is a hot research topic in natural language processing (NLP). In past decades, researchers have developed various rule- or template-based systems, which are typically in vertical domains, e.g., transportation (Ferguson et al., 1996) and education (Graesser et al., 2005). In the open domain, data-driven approaches play an important role, because the diversity and uncertainty make it virtually impossible for humans to design rules or templates. Isbell et al. (2000) and Wang et al. (2013) use information retrieval methods to search for a reply from a pre-constructed database; Ritter et al. (2011) formalize conversation as a statistical machine translation task.

Recently, the renewed prosperity of neural networks brings new opportunities to open-domain conversation (Vinyals and Le, 2015; Shang et al., 2015; Serban et al., 2016a; Li et al., 2016a). In these studies, researchers leverage sequence-to-sequence (`seq2seq`) models to encode a *query* (user-issued utterance) as a vector and to decode the vector into a *reply*. In both encoders and decoders, an RNN keeps one or a few hidden layers; at each time step, it reads a word and changes its state accordingly. RNNs are believed to be well capable of modeling word sequences, benefiting machine translation (Sutskever et al., 2014), abstractive summarization (Rush et al., 2015) and other tasks of natural language generation. Contrary to retrieval methods, neural network-based conversation systems are *generative* in that they can synthesize new utterances; results in the literature also show the superiority of `seq2seq` to phrase-based machine translation for dialogue systems (Shang et al., 2015). In our study, we focus on neural network-based generative *short-text conversation*, where we do not consider context information, following Wang et al. (2013) and Shang et al. (2015).

Despite these, neural networks' performance is far from satisfactory in human-computer conversation. A notorious problem is the *universal reply*: the RNN prefers to generate safe, universally relevant sentences with little meaning, e.g., "something" (Serban et al., 2016a) and "I don't know" (Li et al., 2016a). One problem may lie in the objective of decoding. If we choose a reply with the maximal estimated

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3349–3358, Osaka, Japan, December 11-17 2016.

probability (either greedily or with beam search), it is probable to obtain such universal replies, because they do appear frequently in the training set. Another potential problem is that, the query may not convey sufficient information for the reply, and thus the encoder in `seq2seq` is less likely to obtain an informative enough vector for decoding.

In this paper, we propose a content-introducing approach to generative short-text conversation systems, where a reply is generated in a two-step fashion: (1) First, we predict a keyword, that is, a noun reflecting the main gist of the reply. This step does not capture complicated semantic and syntactic aspects of natural language, but estimates a keyword with the highest pointwise mutual information against query words. The keyword candidates are further restricted to nouns, which are not as probable as universal words (e.g., *I* and *you*), but can introduce substantial content to reply generation. (2) We then use a modified encoder-decoder model to synthesize a sentence containing the keyword. In traditional `seq2seq`, the decoder generates the reply from the first word to the last in sequence, which prevents introducing certain content (i.e., a given word) to the reply. To tackle this problem, we propose `seq2BF`, a novel "*sequence to backward and forward sequences*" model, based on our previous work of backward and forward language modeling (Mou et al., 2015). The `seq2BF` model decodes a reply starting from a given word, and generates the remaining previous and future words subsequently. In this way, the predicted keyword can appear at an arbitrary position in the generated reply.

The rest of this paper is organized as follows. Section 2 describes the proposed approach; Section 3 presents experimental results. Section 4 briefly reviews related work in the literature. Finally we conclude our paper and discuss future work in Section 5.

2 Our Approach

In this section, we present our content-introducing generative dialogue system in detail. Subsection 2.1 provides an overview of our approach, Subsection 2.2 introduces the keyword predictor, and Subsection 2.3 elaborates the proposed `seq2BF` model. We describe training methods in Subsection 2.4.

2.1 Overview

Figure 1 depicts the overall architecture of our approach, which comprises two main steps:

Step I: We first use PMI to predict a keyword for the reply, as shown in Figure 1a.

Step II: After keyword prediction, we generate a reply conditioned on the keyword as well as the query. More specifically, we propose the `seq2BF` model, which generates the backward half of the sequence (Figure 1b) and then the forward half (Figure 1c).

Notice that, the RNNs in Step II do not share parameters (indicated by different colors in the figure) because they differ significantly from each other. Moreover, the encoder and decoder do not share parameters either, which is standard in `seq2seq`. For clarity, we do not assign different colors for encoders and decoders, but separate them with a long arrow in Figures 1b and 1c.

2.2 Keyword Prediction

In this step, we use pointwise mutual information (PMI) to predict a keyword for the reply. We leverage such surface statistics because this step outputs a single keyword, which does not capture complicated syntax and semantics of queries and replies. Our goal of content introducing is to suggest a word that is especially suited to the query, instead of predicting a most likely (common) word. Hence, the pointwise mutual information is an appropriate statistic for keyword prediction.

Formally, we compute PMI of a query word w_q and a reply word w_r using a large training corpus by

$$\mathrm{PMI}(w_q, w_r) = \log \frac{p(w_q, w_r)}{p(w_q)p(w_r)} = \log \frac{p(w_q|w_r)}{p(w_q)} \tag{1}$$

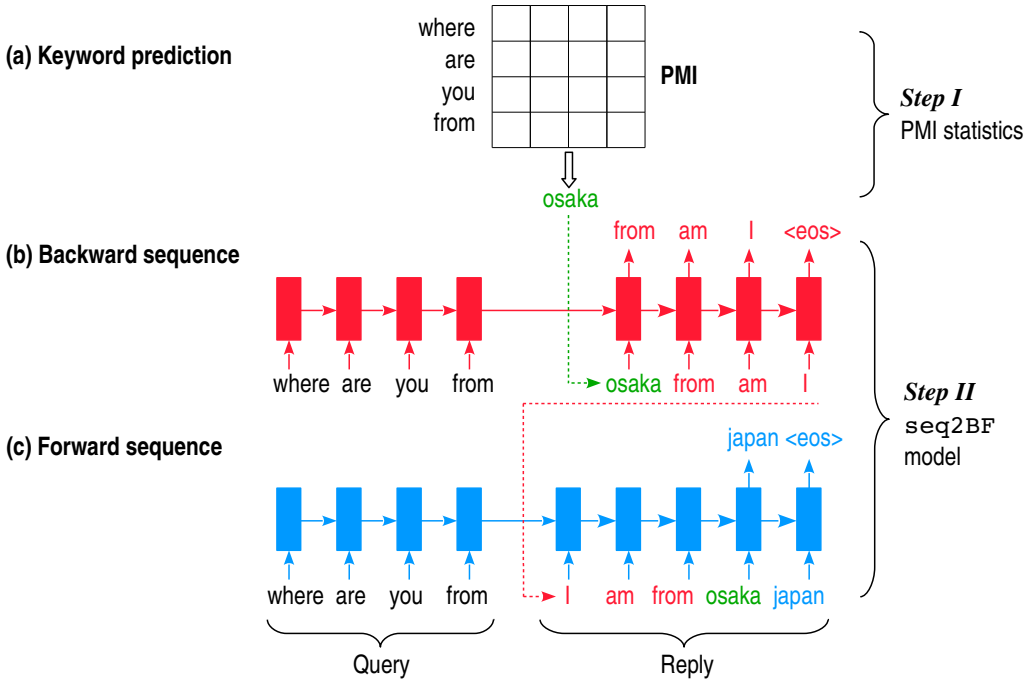


Figure 1: An overview of our content-introducing approach to generative dialogue systems.

When predicting, we choose the keyword w_r^* with the highest PMI score against query words $w_{q_1}, \cdots w_{q_n}$, i.e., $w_r^* = \text{argmax}_{w_r} \text{PMI}(w_{q_1} \cdots w_{q_n}, w_r)$, where

$$\text{PMI}(w_{q_1} \cdots w_{q_n}, w_r) = \log \frac{p(w_{q_1} \cdots w_{q_n}|w_r)}{p(w_{q_1} \cdots w_{q_n})} \tag{2}$$

$$\approx \log \frac{\prod_{i=1}^n p(w_{q_i}|w_r)}{\prod_{i=1}^n p(w_{q_i})} = \sum_{i=1}^n \log \frac{p(w_{q_i}|w_r)}{p(w_{q_i})} = \sum_{i=1}^n \text{PMI}(w_{q_i}, w_r) \tag{3}$$

The approximation is due to the independency assumptions of both the prior distribution $p(w_{q_i})$ and posterior distribution $p(w_{q_i}|w_r)$. While the two assumptions may not be true, we use them in a pragmatic way so that the word-level PMI is additive for a whole utterance. Experiments show that this treatment generally works well.

Different from choosing the most likely word, PMI penalizes a common word by dividing its prior probability; hence, PMI prefers a word that is most "mutually informative" with the query. Moreover, we manually restrict keyword candidates to nouns, so that this step can introduce substantial content to reply generation, which will be discussed in the next part.

2.3 The seq2BF Model

To insert the predicted keyword into sequence generation, we cannot use the traditional seq2seq model. In existing approaches, we usually decompose the probability of an output sentence $\boldsymbol{r} = r_1 r_2 \cdots r_m$ given an input sentence $\boldsymbol{q} = q_1 q_2 \cdots q_n$ by

$$p(r_1, \cdots, r_m|\boldsymbol{q}) = p(r_1|\boldsymbol{q}) p(r_2|r_1, \boldsymbol{q}) \cdots p(r_m|r_1 \cdots r_{m-1}, \boldsymbol{q}) = \prod_{i=1}^m p(r_i|r_1 \cdots r_{i-1}, \boldsymbol{q}) \tag{4}$$

The output sentence is thus predicted in sequence from r_1 up to r_m either greedily or with beam search. I personally believe such decomposition is mainly inspired by the observation that humans always say a sentence from the beginning to the end.

However, in our content-introducing approach to generative dialogue systems, the predicted keyword could appear at the beginning (r_1), the middle (r_2 to r_{m-1}), or the end (r_m) of the reply. It is then natural

to decompose the probability starting from the given word. In particular, the predicted keyword k splits a reply into two (sub-)sequences:

$$\begin{aligned} \text{Backward sequence:} \quad & r_{r-1}, \cdots, r_1 \\ \text{Forward sequence:} \quad & r_{k+1}, \cdots, r_m \end{aligned}$$

and the joint probability of remaining words can be written as

$$p\left(\begin{matrix} r_{k-1} \cdots r_1 \\ r_{k+1} \cdots r_m \end{matrix} \middle| r_k, \boldsymbol{q} \right) = \prod_{i=1}^{k-1} p^{(\text{bw})}(r_{k-i}|r_k, \boldsymbol{q}, \cdot) \prod_{i=1}^{m-k} p^{(\text{fw})}(r_{k+i}|r_k, \boldsymbol{q}, \cdot) \tag{5}$$

where $p(\text{- - -}|r_k, \boldsymbol{q})$ refers to the probability of the backward and forward subsequences given the split word r_k and an encoded query $\boldsymbol{q}$. Notice that both the backward and forward sequence generators include a wildcard allowing rich inner-subsequence and/or inter-subsequence dependencies. In our previous study of backward-and-forward (B/F) language modeling (Mou et al., 2015), we propose three variants: (1) **sep-B/F**: The backward and forward sequences are generated separately. (2) **syn-B/F**: The backward and forward sequences are generated synchronously using a single RNN, two output layers at each time step for the two sequences. (3) **asyn-B/F**: The two sequences are generated asynchronously, that is, we first generate the backward "half" sequence, conditioned on which we then generate the forward "half." Our previous experiments show the asyn-B/F is the most natural way of modeling backward and forward sequences, and thus we adopt this variant in `seq2BF`.

Specifically, our `seq2BF` model works as follows. A `seq2seq` neural network encodes a query and decodes a "half" reply, that is, the first set of factors in Equation 5 becomes $p^{(\text{bw})}(r_{k-i}|r_k, \boldsymbol{q}, \cdot) = p^{(\text{bw})}(r_{k-i}|r_k \cdots r_{k-i+1}, \boldsymbol{q})$, where $1 \leq i \leq k-1$. The decoder here outputs words in a reversed order from r_{k-1}, r_{k-2} to r_1, so that the reversed "half" sequence is fluent with respect to the given word, at least from a mathematical perspective. (Please see Figure 1b.)

Then another `seq2seq` model encodes the query again, but decodes the entire reply, provided that the first half of the reply is given (Figure 1c), i.e., $p^{(\text{fw})}(r_{k+i}|r_k, \boldsymbol{q}, \cdot) = p^{(\text{fw})}(r_{k+i}|r_1 \cdots r_k \cdots r_{k+i-1}, \boldsymbol{q})$, $(1 \leq i \leq m-k)$. Here, the forward generator is aware of the backward half sequence $r_1 \cdots r_{k-1}$, where the word order is reversed again, so that they are in a normal order for fluent forward generation.

In both backward and forward `seq2seq` models, we use RNNs with gated recurrent units (GRUs) for information processing (Cho et al., 2014), given by

$$\boldsymbol{r}_t = \sigma(W_r \boldsymbol{w}_t + U_r \boldsymbol{h}_{t-1} + \boldsymbol{b}_r) \tag{6}$$

$$\boldsymbol{z}_t = \sigma(W_z \boldsymbol{w}_t + U_r \boldsymbol{h}_{t-1} + \boldsymbol{b}_z) \tag{7}$$

$$\widetilde{\boldsymbol{h}}_t = \tanh\Big(W_h \boldsymbol{w}_t + U_h(\boldsymbol{r} \circ \boldsymbol{h}_{t-1}) + \boldsymbol{b}_h\Big) \tag{8}$$

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \circ \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \circ \widetilde{\boldsymbol{h}}_t \tag{9}$$

where W's and U's are weights and $\boldsymbol{b}$'s are bias terms. $\boldsymbol{w}_t$ is the word embedding; $\boldsymbol{h}_t$ is the hidden state at the time step t. "$\circ$" denotes element-wise product.

2.4 Model Training

Training (i.e., parameter estimation) is always a most important thing in the neural network regime, and oftentimes, problems arise when we prepare the dataset.

Fortunately, the `seq2BF` model can be trained without additional labels. We randomly sample a word in a reply as the split word, take the first half, and reverse its word order; in this way, we obtain the training data for the backward sequence generator. The forward sequence generator is essentially a `seq2seq` encoder and decoder from queries to replies. The difference between the pure `seq2seq` and the forward generator of `seq2BF` lies in the inference stage: in our scenario, we ignore the query-reply `seq2seq` generator's output at the beginning steps, but feed it with the "half" reply obtained by our backward sequence generator as well as the predicted keyword (red and green words in Figure 1c); then we let the `seq2seq` model generate remaining future words (blue words in Figure 1c).

It should be emphasized that the backward sequence generator requires "half" replies starting from the split word as training data, and that we cannot train the model with a full reversed sentence. Otherwise, the backward part will undesirably generate an entire reversed reply, and the forward part cannot add much to it.

3 Experiments

3.1 Dataset

We evaluated our approach on a Chinese dataset of human conversation crawled from the Baidu Tieba[1] forum. We used 500,000 query-reply pairs to train the `seq2BF` model. We had another unseen 2000 and 27,871 samples for validation and testing, respectively. To obtain PMI statistics in the first step (Figure 1a) of our method, we use a much larger dataset containing 100M query-reply pairs.

Chinese language is different from English in that a Chinese character carries more semantics than an alphabet. For example, the characters 黑 and 板 mean "black" and "board" in English, respectively; the term 黑板 means "blackboard." Because we have far more Chinese terms than English words, our `seq2BF` is trained in the character level out of efficiency concerns. But we train the keyword predictor with noun phrases (Chinese terms), by noticing that *blackboard* is different from *board*, despite some subtle relations. Fortunately, the two granularities can be integrated together straightforwardly: during backward sequence generation, we only need to condition the model on the character sequence in the key term instead of a single keyword, that is, we have several green inputs in Figure 1b. In our study, we kept 2.5k noun terms as candidate keywords and 4k characters for `seq2BF` generation.

3.2 Hyperparameters

In our experiments, word embeddings and recurrent layers were 500d. We used rmsprop to optimize all parameters except embeddings, with initial weights uniformly sampled from $[-.08, .08]$, initial learning rate 0.002, moving average decay 0.99, and a damping term $\epsilon = 10^{-8}$. Because word embeddings are sparse in use (Peng et al., 2015), we optimized embeddings asynchronously by stochastic gradient descent with the learning rate divided by $\sqrt{\epsilon}$. We set the mini-batch size to 50. These values were mostly chosen empirically by following Karpathy et al. (2015) and Mou et al. (2015); they generally work well in our scenarios. We did not tune the hyperparameters in this paper, but are willing to explore their roles in dialogue generation as future work.

The validation set (containing 2k query-reply samples) was used for early stop only. We chose the parameters yielding the highest character-level BLEU-2 score on our validation set.

3.3 Performance

We evaluated our results in terms of the following criteria:

- **Human Evaluation**. We had six volunteers[2] to annotate the results of our content-introducing `seq2BF` and baselines. To ensure the quality of human evaluation, we randomly sampled 200 queries and replies in the test set. The samples and volunteers were further split into two equal-sized groups of different annotation protocols:
  - *Pointwise annotation*. The volunteers were asked to annotate a score indicating the appropriateness of a reply to a given query: 0 = bad reply, 2 = good reply, and 1 = borderline.
  - *Pairwise annotation*. Given a certain query, the volunteers were asked to judge whether pure `seq2seq` is better than, equal to, or worse than `seq2BF` (with content introducing). If they could not understand both replies, they were asked to choose "equal."

  All our human evaluation were conducted in a random, blind fashion, i.e., we randomly shuffled the samples and volunteers did not know which system generated a particular reply. Notice that we did not define what a "good" or "bad" reply is; otherwise, the annotation may be biased towards certain systems. Rather, annotators had their own subjective discretion. While criteria may differ from

[1] http://tieba.baidu.com

[2] All volunteers are well-educated native speakers of Chinese and have received a Bachelor's degree or above.

(a)

Method	PointHuman	Length	Entropy
seq2seq	0.58	**5.61**	6.960
seq2BF$_-$	0.46	5.60	6.971
seq2BF$_+$	**0.67**	5.31	**9.139**
Groundtruth	–	9.19	8.832

(b)

PairHuman			
Method	**Wins**	**Ties**	**Loses**
seq2seq	24.7	26.0	49.3
seq2BF$_+$	**49.3**	26.0	24.7

Table 1: The performance of our content-introducing seq2BF (denoted as seq2BF$_+$) dialogue system in comparison with pure seq2seq and seq2BF without predicted keywords (seq2BF$_-$). **PointHuman**: Pointwise human evaluation. (Annotator agreement: std=0.33, Fleiss' κ=0.27.) **PairHuman**: Pairwise human evaluation, which shows the percentage at which a system wins, loses, or ties in comparison with the other (κ=0.29).

person to person, the ranking of average scores reflects the comparison of different dialog systems. (In our study, the ranking is consistent among all six annotators.)

- **Length**. The length of an utterance is an objective, surface metric that reflects the substance of a generated reply.

- **Entropy**. Entropy is another objective metric, which shows the serendipity of a reply by measuring the amount of information contained in the utterance. We computed the average character-level entropy, given by

$$-\frac{1}{|R|}\sum_{w\in R}\log_2 p(w) \tag{10}$$

where R refers to all replies, $|R|$ is the number of words in all replies, and $p(\cdot)$ is the unigram probability of a character in the training set.

The latter two metrics are "intrinsic," by which we mean no reference (groundtruth reply) is needed to compute the metric. They are used in Serban et al. (2016b).[3] In our experiments, objective metrics were assessed with all test samples.

We do not include BLEU scores as our evaluation criteria, which are used in Li et al. (2016a). As a pilot study, we also asked two volunteers to write their own replies to 50+ queries. One obtained 1.69 BLEU-4 score. (The result is lower than 1.74 obtained by an automatic dialogue system in Li et al. (2016a); this may be caused by different datasets and languages.) What surprises us is that the other volunteer obtained 0 BLEU-2 score, indicating that no bi-gram overlaps between his replies and the references. This result provides evidence of the diversity among human replies, and thus we abandoned BLEU scores as evaluation criteria. We reported this case study in our paper so as to shed more light on the research of evaluation metrics in dialogue systems.

Table 1 compares our content-introducing seq2BF with the seq2seq baseline. Also included is the seq2BF model without content introducing, that is, we slacked the keyword constraint in the backward sequence generator (Figure 1b), and let it generate a reversed half reply freely.

As seen in Table 1a, the content-introducing seq2BF model yields the highest pointwise human annotation result. For seq2BF without content introducing, we obtain low scores, showing that artificially splitting a sequence into two parts itself is not a fancy way of modeling natural language sentences. However, given a keyword predicted by PMI statistics, the backward and forward sequence generation can significantly improve the dialogue system in comparison with pure seq2seq. In the pairwise annotation protocol, the content-introducing seq2BF model also outperforms traditional seq2seq to a large extent. We computed standard deviation (0.33) and discrete-class Fleiss' κ scores (0.27–0.29), showing fair annotator agreement (Fleiss, 1971). While the agreement is comparable to previous results, e.g., 0.2–0.4 reported in Shang et al. (2015), we find the disagreement mainly comes from the discretion of

[3]The entropy equation in Serban et al. (2016b, v3) has a minor error. We confirmed this with Serban via personal emails.

	Chinese	English (translated)
Query	李有男友公开过了	It's known that *Li*† has a boyfriend.
Grountruth	都已经分了 之前李的贴吧都在讨论了	Broken up. There's discussion in her *Tieba*.‡
seq2seq	我是男的	I am a male
seq2BF$_-$	我的男友	My boyfriend
seq2BF$_+$	有**绯闻**男友	Has a **rumor**ed boyfriend
Query	人大复试飘过	Passed second-round exam of Renming Univ.
Grountruth	这么牛，什么专业	Cool, what's your major
seq2seq	我也是！	Me too!
seq2BF$_-$	我也是！	Me too!
seq2BF$_+$	**分数**是什么	What is your **score**
Query	挺漂亮的 祝福祝福	So beautiful, congratulations
Grountruth	下手慢了哈。现在	You're late till now
seq2seq	谢谢	Thank you
seq2BF$_-$	是的，谢谢	Yes, thank you
seq2BF$_+$	谢谢**夸奖**	Thank you for **praising**
Query	我要换头像了！	Want to change a photo
Grountruth	打算换成什么啊～	What do you like...
seq2seq	我是你的头像	I'm in your photo
seq2BF$_-$	我是你的头像	I'm in your photo
seq2BF$_+$	第一张**图像**是谁	Who is in your first **photo**

Table 2: Examples of generated replies. Predicted keywords are in **bold**. †*Li* is a Chinese actress; we anonymized her first name. ‡*Tieba* is a Chinese forum where our datasets are crawled.

Model	seq2seq	seq2BF$_-$	seq2BF$_+$	
			keyword	remaining
Entropy	6.960	6.971	11.630	7.422

Table 3: Fine-grained analysis of character-level entropy. In the seq2BF$_+$ model, we analyze the average entropy of keywords and remaining words separately.

the "goodness" of a reply (so that the annotation complies with the subjective nature of human evaluation). All annotators yielded the same system-level ranking order, providing consistent evidence of the effectiveness of our approach.

Regarding intrinsic metrics, our seq2BF with a predicted keyword generates slightly shorter replies than seq2seq, but contains far richer information, as the entropy increases by 30%. The results verify that content introducing is particularly useful in generative human-computer dialogue systems.

3.4 Case Studies and Discussion

We provide case studies in Table 2.[4] As we see, the seq2seq responder prefers safe, universally relevant utterances like "me too." In these examples, the replies generally match the queries in meaning, but such universal replies are too boring and thus undesirable in real applications. In the seq2BF model with content introducing, we predict a keyword of the reply with PMI. This yields meaningful words/terms like *rumor* and *score*, serving as the gist of the reply. Then the seq2BF model generates previous and future words to obtain a more complete utterance (maybe not a whole sentence because of the casualness in human conversation). The proposed "backward and forward sequences" mechanism ensures that the predicted keyword can appear at an arbitrary position in the utterance.

We delve deep into the question: why seq2seq models (or variants) tend to generate universal

[4]Due to Chinese-English translation, some characteristics cannot be fully presented in the English text, e.g., the position of the given word, the length of the reply, and even the part-of-speech of a word. Nevertheless, we present the predicted keyword and its translated counterpart in bold and thus the aforementioned characteristics can be visually demonstrated to some extent.

replies? We may have two conjectures: (1) The seq2seq model cannot capture rich enough semantics other than "yes," "me too," etc. (2) The sequence generator is able to capture rich semantics, but starting from a high-level universal word at the beginning, it is unlikely to fall into concrete concepts.

We present in Table 3 the average character-level entropy of keywords and non-keywords. We find that, provided with a noun term, seq2BF can generate meaningful remaining words (keyword excluded) with an entropy of 7.422, higher than 6.971 given by seq2BF without keywords. Noticing that the seq2BF model is exactly the same in content-introducing and non-content-introducing settings, we believe the second conjecture holds. Choosing the most likely reply yields universal utterances; moreover, RNN sequence generators are reluctant to introduce concrete concepts, provided with one or a few universal words/terms (like *I* and *you*) that are greedily chosen at the beginning.

Fortunately, our content-introducing seq2BF works in an opposite fashion. We first predict a meaningful but not that probable noun term as the keyword; then we feed seq2BF with such concrete keyword that provides substantial content. In this way, our approach significantly outperforms pure seq2seq generation in short-text conversation systems.

4 Related Work

4.1 Dialogue Systems

Automatic human-computer conversation has long attracted the attention of researchers. In early decades, people design rule- or template-based systems, but they are mainly in vertical domains (Ferguson et al., 1996; Misu and Kawahara, 2007). Although such approaches can also be extended to the open domain (Han et al., 2015), their generated sentences are subject to 7 predefined forms and thus are highly restricted. For open dialogues, researchers have applied data-driven approaches, including retrieval methods (Isbell et al., 2000; Wang et al., 2013), phrase-based machine translation (Ritter et al., 2011), and recurrent neural networks (Sordoni et al., 2015; Shang et al., 2015).

A hot research topic in human-computer conversation is mixed-initiative systems, for example, the TRAINS-95 system for route planning (Ferguson et al., 1996) and AutoTutor for learner advising (Graesser et al., 2005). Li et al. (2016b) propose a proactive dialogue system that can introduce new content when a stalemate occurs. The system is chatbot-like and in the open domain; an external knowledge base is used for searching related entities as new content. They propose a random walk-like reranking algorithm based on retrieval results. Different from Li et al. (2016b)'s work, our paper addresses the problem of content introducing in open-domain generative dialogue systems.

4.2 Neural Networks for Sentence Generation

Sutskever et al. (2014) propose seq2seq for machine translation; the idea is to encode a source sentence as a vector by a recurrent neural network (RNN) and to decode the vector to a target sentence by another RNN. Bahdanau et al. (2015) enhance it with an attention mechanism. These approaches largely benefit natural language generation tasks such as abstractive summarization (Rush et al., 2015), question answering (Yin et al., 2016), and poetry generation (Wang et al., 2016).

For neural network-based dialogue systems, Sordoni et al. (2015) summarize a query and context as bag-of-words features, based on which an RNN decodes the reply. Shang et al. (2015) generate replies for short-text conversation by seq2seq-like neural networks with local and global attention. Yao et al. (2015) and Serban et al. (2016a) design hierarchical neural networks for multi-turn conversation.

To address the problem of universal replies, Li et al. (2016a) propose a mutual information training objective. Serban et al. (2016b) apply a variational Bayes approach that imposes a probabilistic distribution on the hidden variables and encodes the parameters of the posterior distribution. A very recent study similar to ours is Xing et al. (2016), where replies are augmented with topic information. In a dialogue-like question-answering system, Yin et al. (2016) query a knowledge base and insert a selected triple into an answer sentence by a soft logistic unit. In such approach, however, the answer may not actually appear in the generated sentence, especially when the test patterns are different from training ones. Unlike existing work, our seq2BF model guarantees the predicted keyword can appear in the reply at an appropriate position.

5 Conclusion and Future Work

In this paper, we proposed a content-introducing approach to generative short-text conversation systems. Instead of generating a reply sequentially from the beginning word to the end as in existing approaches, we used pointwise mutual information to predict a keyword, i.e., a noun term, for the reply. Then we proposed a "sequence to backward and forward sequences" (`seq2BF`) model to generate a reply containing the predicted keyword. The `seq2BF` mechanism ensures the keyword can appear at an arbitrary position in the reply, but the generated utterance is still fluent. Experimental results show that our approach consistently outperforms the pure `seq2seq` model in dialogue systems in terms of human evaluation and the entropy measure.

In future work, we would like to apply different keyword prediction techniques (e.g., neural sentence models) to improve the performance; the proposed `seq2BF` model can also be extended to other applications like generative question answering, where the answer may be given by searching an external database or knowledge base.

Acknowledgments

We thank Jiexiu Zhai, Hao Meng, Siqi Wang, Zhexi Hong, Junling Chen, and Zhiliang Tian for evaluating our dialogue systems. We also thank all reviewers for their constructive comments. This research is supported by the National Basic Research Program of China (the 973 Program) under Grant No. 2015CB352201 and the National Natural Science Foundation of China under Grant Nos. 61232015, 91318301, 61421091, 61225007, and 61502014.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of 8th Workshop on Syntax, Semtnatics and Structure in Statistical Translation*, pages 103–111.

George Ferguson, James Allen, and Brad Miller. 1996. TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of Artificial Intelligence Planning Systems Conference*, pages 70–77.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378.

Arthur C Graesser, Patrick Chipman, Brian C Haynes, and Andrew Olney. 2005. AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education*, 48(4):612–618.

Sangdo Han, Jeesoo Bang, Seonghan Ryu, and Gary Geunbae Lee. 2015. Exploiting knowledge base to generate responses for natural language dialog listening agents. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 129–133.

Charles Lee Isbell, Michael Kearns, Dave Kormann, Satinder Singh, and Peter Stone. 2000. Cobot in LambdaMOO: A social statistics agent. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pages 36–41.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Xiang Li, Lili Mou, Rui Yan, and Ming Zhang. 2016b. StalemateBreaker: A proactive content-introducing approach to automatic human-computer conversation. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 2845–2851.

Teruhisa Misu and Tatsuya Kawahara. 2007. Speech-based interactive information guidance system using question-answering technique. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume IV, pages 145–148.

Lili Mou, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2015. Backward and forward language modeling for constrained natural language generation. *arXiv preprint arXiv:1512.06612*.

Hao Peng, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2015. A comparative study on regularization strategies for embedding-based neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2106–2111.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 583–593.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016a. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 3776–3783.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016b. A hierarchical latent variable encoder-decoder model for generating dialogues. *arXiv preprint arXiv:1605.06069*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1577–1586.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205.

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

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 935–945.

Zhe Wang, Wei He, Hua Wu, Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. 2016. Chinese poetry generation with planning based neural network. In *Proceedings the 26th International Conference on Computational Linguistics*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2016. Topic augmented neural response generation with a joint attention mechanism. *arXiv preprint arXiv:1606.08340*.

Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. 2015. Attention with intention for a neural network conversation model. *arXiv preprint arXiv:1510.08565*.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 2972–2978.

Disfluent but effective? A quantitative study of disfluencies and conversational moves in team discourse

Felix Gervits[1] Kathleen Eberhard[2] Matthias Scheutz[1]
[1]Tufts University, Human-Robot Interaction Laboratory, Medford, MA
[2]University of Notre Dame, Department of Psychology, Notre Dame, IN
{felix.gervits, matthias.scheutz}@tufts.edu
eberhard.1@nd.edu

Abstract

Situated dialogue systems that interact with humans as part of a team (e.g., robot teammates) need to be able to use information from communication channels to gauge the coordination level and effectiveness of the team. Currently, the feasibility of this end goal is limited by several gaps in both the empirical and computational literature. The purpose of this paper is to address those gaps in the following ways: (1) investigate which properties of task-oriented discourse correspond with effective performance in human teams, and (2) discuss how and to what extent these properties can be utilized in spoken dialogue systems. To this end, we analyzed natural language data from a unique corpus of spontaneous, task-oriented dialogue (CReST corpus), which was annotated for disfluencies and conversational moves. We found that effective teams made more self-repair disfluencies and used specific communication strategies to facilitate grounding and coordination. Our results indicate that truly robust and natural dialogue systems will need to interpret highly disfluent utterances and also utilize specific collaborative mechanisms to facilitate grounding. These data shed light on effective communication in performance scenarios and directly inform the development of robust dialogue systems for situated artificial agents.

1 Introduction

Effective coordination is fundamental to teamwork in many fields, particularly for teams working under stress. Though a variety of team structures exist, actions teams are among the most demanding, due to the need for the teammates to engage in goal-oriented, interdependent tasks, and to dynamically adapt their decision-making, communication, and planning strategies (Serfaty et al., 1993; Sundstrom et al., 2000). Recently, action teams have begun to incorporate artificial agents in mixed-initiative roles (see Sycara and Sukthankar (2006) for a thorough review). Such teams are largely employed in performance scenarios (e.g., search and rescue missions, military squads, surgical teams), and require strong team communication to achieve complex objectives in a time-sensitive context. Among the many elements needed for coordination in human-agent teams, perhaps the most important for team success is establishing *common ground*. Common ground is a mutual understanding between teammates, involving shared knowledge of the task environment, goals, perspectives, and other factors. For common ground to be established, it is important that teammates not only share information, but also ensure that the information was understood the way it was intended. This process, known as *grounding*, results in a mutual recognition by both parties of the shared information being a part of their common ground and forms the basis for coordination in both dialogue and action (Clark, 1996).

1.1 Task-oriented remote dialogue

Though there is no objective way to measure common ground, evidence for it may be found in the team communication channels. However, grounding in task-oriented remote communication is complicated by a number of factors, including time pressure/workload (Entin and Serfaty, 1999; Khawaja et

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3359–3369, Osaka, Japan, December 11-17 2016.

al., 2012; Urban et al., 1996), mode of interaction (Clark and Brennan, 1991; Doherty-Sneddon et al., 1997; Krauss and Weinheimer, 1966), and team structure (Bortfeld et al., 2001; Clark and Krych, 2004). Given the difficulty of grounding exchanges in task-oriented dialogue, it is not surprising that communication channels in remotely-communicating action teams are very noisy, often containing features such as disfluency, overlapping speech, and ambiguity.

Disfluencies are particularly interesting because they are very common in spontaneous speech, and have been implicated in various interpersonal and cognitive functions (Nicholson et al., 2003). One view holds that disfluencies are noise resulting from increased production difficulty due to cognitive workload. Several studies have found support for this view by showing that disfluency rates tend to increase with higher workload (Berthold and Jameson, 1999; Lindström et al., 2010). Another position is that disfluencies may not be solely due to workload, but may reflect underlying coordination processes such as monitoring one's addressee (Clark and Krych, 2004) or soliciting help in the dialogue (Bortfeld et al., 2001). In support of this view are studies showing that speakers detect and utilize disfluencies to help process surrounding speech (Brennan and Schober, 2001), resolve reference ambiguities (Arnold et al., 2007), hold the conversational floor (Smith and Clark, 1993), improve recall (Corley et al., 2007), and mark discourse structure (Swerts, 1998). Barr (2001) has likened disfluencies to a form of "vocal gestures" due to their ability to provide insight into a speaker's metacognitive state. Despite these findings, no prior studies have examined the role of speech disfluencies with regard to performance in an unscripted collaborative task. Thus, it remains unclear whether the interpersonal benefit of disfluency overrides the cognitive drawback also associated with disfluent speech. Moreover, most of the existing literature on the benefit of disfluency deals with filled pauses, whereas relatively little is known about other types, such as self-repairs. It is also unclear what types of coordination strategies (if any) successful teams use to overcome the various grounding constraints associated with task-oriented communication, and how these strategies interact with a team's ability to establish common ground. Our study was designed to address these gaps in the literature.

1.2 Present study

The present study consists of a quantitative investigation of factors that influence effective grounding and performance in remotely-communicating action teams. Given that grounding is often constrained by factors such as coordination strategy, speaker role, and time pressure/workload, our aim was to explore how these factors interact with team communication and, ultimately, performance. The data consisted of the linguistic- and dialogue-level annotations in the Cooperative Remote Search Task (CReST) corpus (Eberhard et al., 2010) obtained from unscripted communication between a (remotely-located) director and a searcher, who was located in an indoor environment and collaboratively performed a number of tasks involving objects in the environment (see Section 2 for more information). Since the task was specifically designed to simulate the structure of action teams in which a robot may play a vital role (e.g., urban search and rescue), the results can inform our understanding of the kinds of communication and coordination strategies that artificial agents will need to adopt to be effective partners in these hierarchical, mixed-initiative teams.

1.3 Predictions

One specific prediction is that effective teams would be those in which the director plays a greater role in managing the task, and searchers are the more receptive party. Additionally, effective teams will minimize *joint collaborative effort* (Clark and Wilkes-Gibbs, 1986) by establishing common ground with respect to objects and locations in the environment. Evidence for this would be the use of various communication strategies, including: establishing shared referents and shorthand conventions (Clark and Wilkes-Gibbs, 1986), completing one another's utterances (Clark and Schaefer, 1989), taking a partner's perspective (Brennan et al., 2010), and breaking up longer utterances into installments (Clark and Brennan, 1991).

Evidence of collaborative dialogue may also manifest in increased disfluency rate. Though some disfluent speech may be expected due to the difficult nature of the task, an increase in self-repairs may also indicate that a speaker is self-monitoring and adjusting their speech for clarity and accuracy (Clark and

Krych, 2004). In this way, self-repairs may serve to minimize collaborative effort by fixing a problematic utterance before one's partner needs to intervene (Levelt, 1983). This can minimize the number of dialogue turns and lead to more efficient exchanges. Thus, we predict that effective teams would make more disfluencies due to the need to plan and coordinate at higher speeds. Disfluencies would be mainly self-repairs caused by monitoring one's listener and dynamically adjusting one's speech to aid comprehension.

2 Method

The CReST corpus (Eberhard et al., 2010) was used to evaluate speech patterns in 10 teams (20 individuals) of people performing a collaborative, remote, search task. Approximately 8 minutes of language data were extracted for each team. Contained in the corpus is dialogue that occurred before and after a time-limit warning which allows us to examine the effects of time pressure. The corpus also provides an objective measure of the pairs' task performance, which we used to operationalize "effectiveness of communication". Additionally, the members of each team had asymmetrical roles, which we included as an additional factor. Lastly, the corpus was annotated for various linguistic and dialogue events in the speech, including conversational moves and disfluencies (see Eberhard et al. (2010) for additional details about the corpus).

2.1 Task description

The members of each pair were randomly assigned to the *Director* role and *Searcher* role. The director was seated in front of a computer that displayed a floor plan map of the search environment and wore a headset for remotely communicating with the searcher. The searcher also wore a headset and was situated in the search environment which consisted of a hallway and 6 connected office rooms. Neither was familiar with the environment. Distributed throughout the environment were 8 blue boxes, each with three colored blocks, 8 empty green boxes (numbered 1-8), 8 empty pink boxes, and a cardboard box that was at the furthest point from entrance at the end of a hallway. Some of the colored boxes were partially hidden behind a door, on a chair, under a table, etc.

The pairs were informed that the director's map showed the locations of all the boxes except the green ones and that the locations of some of the blue boxes were inaccurate. They were told that the searcher was to retrieve the cardboard box, put the blue blocks from the blue boxes into it, and report the locations of the green boxes to the director, who was to mark them on the map by dragging green icons numbered 1-8. They were told that instructions for the pink boxes would be given to them later. Five minutes into the task, the director's communication with the searcher was put on hold and the director was told that each blue box contained a yellow block which was to be put into each of the pink boxes. To examine effects of time pressure, the director also was told that they had 3 minutes to complete all of the objectives, and a timer that counted down the 3 minutes was displayed next to the map.

2.2 Disfluency annotation

Disfluencies were coded according to the HCRC Disfluency Coding Manual (Lickley, 1998), which includes categories for prolongations, pauses (filled and silent), and self-repairs: *repetitions* (e.g., "L-look in the box"), *substitutions* (e.g., "the pink- uh, blue box"), *insertions* (e.g., "go into the room- the nearby room") and *deletions* (e.g., "we don't have- let's hurry up"). Disfluency rates were calculated for each participant as a proportion per every 100 words. *Speech rate* (words per minute, or w.p.m.) and *mean length of utterance* (average number of words per turn at talk, or MLU) also were calculated.

All annotations were carried out using the open-source EXMARaLDA Partitur-Editor (Schmidt and Wörner, 2009). For extracting disfluency data from the annotated files, we used a custom-built search tool called DeepSearch9[1].

[1]Our custom search tool, DeepSearch9, will be made available for research purposes

2.3 Dialogue annotation

The transcribed utterances were hand-annotated for type of conversational move using Carletta et al. (1997)'s scheme. *Initiation* moves include Instruct, Explain, Wh- and Yes/No questions. Two other Initiation moves are subcategories of Yes/No questions, namely, Check and Align. Checks seek confirmation that one has correctly understood what the partner recently said, often by repeating or paraphrasing the partner's utterance. Aligns explicitly request confirmation that a partner has understood what was just said and is ready to move on. They typically are in the form of an "okay?" or "right?" appended to the end of an Instruct or Explain move. *Response* moves include Acknowledge, Wh-, Yes- and No- Replies. Utterance-initial "okays" and "alrights" were coded as *Ready* moves; they serve as a preparation for the following initiation move (e.g., "*Okay*, now go into the next room"). The rates of producing each type of move were calculated by dividing them by the total number of utterances.

2.4 Team effectiveness

Performance was scored with respect to the number of colored boxes whose task was completed, with a maximum score of 24. The average score was 9.9 (range 1 - 19) and the median was 8. The median score was used to divide the 10 teams into an *effective* and *ineffective* group with average scores of 14.8 (S.D. = 4.0) and 5.0 (S.D. = 2.5), respectively.

3 Results

To test our hypotheses of the factors that influence effective team communication, we examined differences in disfluency rate and dialogue moves between teams in the effective and ineffective performance groups. Time pressure and speaker role were used as factors in the analysis. Some relevant dialogues from the corpus are also discussed below to show the various communication and grounding strategies that teams used.

3.1 Grounding strategies

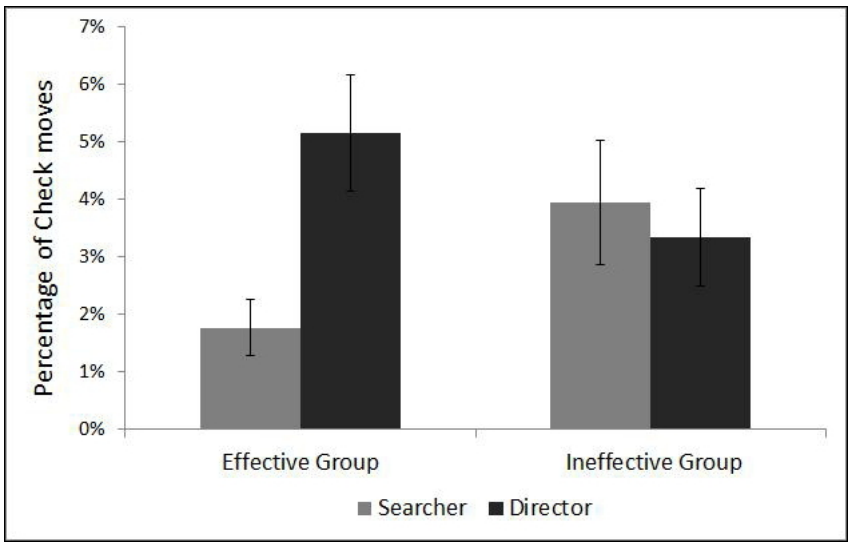


Figure 1: Group x Speaker interaction for Check moves. Error bars represent standard error of the mean.

First, we analyzed group differences in the dialogue moves to test our hypothesis that team effectiveness was related to successful grounding and collaboration. The rates of types of dialogue moves were analyzed with 2x2x2 mixed ANOVAs with Time Pressure and Speaker as within-subjects factors and Group as a between-subjects factors. There was a Group x Speaker interaction for *Check* moves ($F(1,32) = 7.053$, $p = .012$), with effective directors producing more than the ineffective directors, and ineffective searchers producing more than the effective searchers (see Fig. 1). The 3-way interaction was also significant in the analysis of the *Ready* moves ($F(1,32) = 4.657$, $p = .039$). Effective directors'

rate of *Ready* moves was higher than the effective searchers' with and without time pressure, whereas ineffective directors' rate of *Ready* moves decreased to the level of the ineffective searchers' under time pressure (see Fig. 2). Together, the results support our prediction that, compared to ineffective directors, effective directors sought confirmation of their searcher's understanding more often (*Check* moves) and maintained consistent control over the dialogue structure (*Ready* moves).

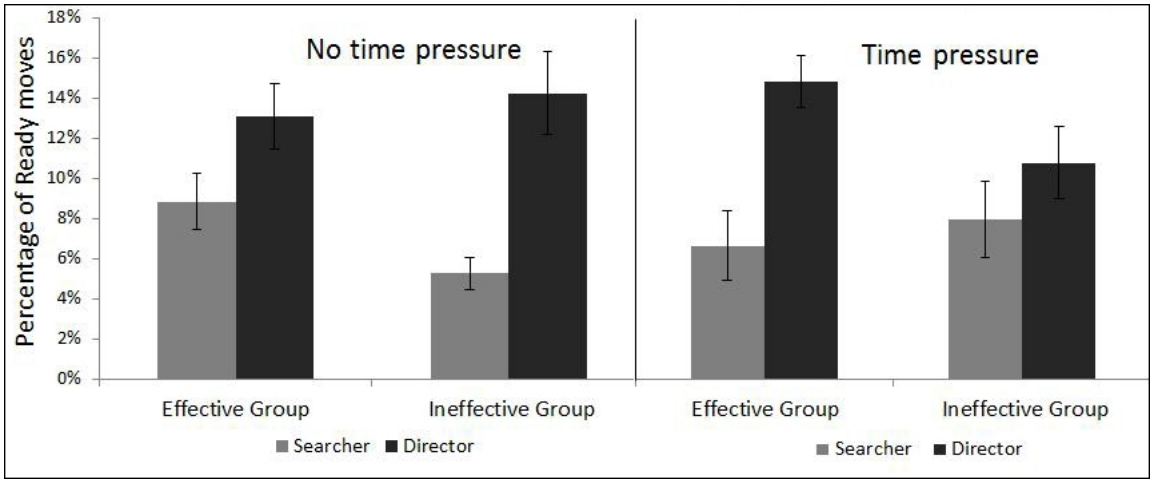


Figure 2: Group x Speaker x Time Pressure interaction for Ready moves. Error bars represent standard error of the mean.

One particular communication strategy used by effective teams was grounding of referents - particularly in room labeling. Since the rooms were not labeled on the map or in the environment, some of the teams developed and utilized a shorthand way to describe them to each other. Consider the following example from an effective team:[2]

D: Okay you're going into the next room
S: Room two
D: Room two, we'll say room two
S: Alright

This is an example of the searcher using a *replacement* (Clark and Wilkes-Gibbs, 1986) to ground a referent. The director initially said "next room", but the searcher proposed "room two" instead. Notice how the director agreed to this shorthand label, and then this was again confirmed by the searcher. This contribution served to establish "room two" as part of the team's common ground. They were able to use this later to minimize joint effort, as in:

S: I'm walking back into [pause] room two
D: Okay
—
S: I'm going back into room one
D: Okay room one, like the very first starting room?
S: Yeah
D: Okay

Here, the label "room one" was used by the searcher, although this was not previously established. The director initiated a *Check* utterance to confirm that they were talking about the same room. After

[2]In the dialogue examples, hyphens (-) indicate repaired segments, colons (:) indicate prolongations, and commas (,) indicate brief silent pauses, with longer pauses contained in brackets. For readability, the director will be referred to as male and the searcher as female in this and all subsequent dialogue examples.

confirmation by the searcher, the grounding was successful and the interaction continued along. Ineffective teams exhibited difficulty grounding referents as seen in the following example:

S: So green number 3 [pause] that was um
D: In the booth
S: Yeah that was in the- [pause] the little room
D: Where in the booth? Where?
S: It wasn't in the big room, it was in the little room, it was right next to the blue box on the chair

Here, the director referred to "the booth", while the searcher referred to a "little room". Since they did not establish common terminology for these referents, the team lost efficiency by taking additional turns.

Additional conversational turns were not always signs of inefficient coordination, however. For example, effective teams facilitated grounding by presenting a task subgoal in installments (Clark and Brennan, 1991) as illustrated here:

D: If you: turn around go out of that room
S: Okay
D: Straight in front of you should be a chair
S: Yes
D: At a table, there's a blue box there
S: Yes
D: Okay, get that

The searcher's acknowledgement following each installment allowed the director to continue on. This strategy also identifies the particular point of grounding difficulty, which is not possible when an entire subgoal is communicated in a single complex turn as illustrated by the following example from an ineffective team:

D: If you look completely straight- straight- straight [pause] like keep walking straight before you even hit the wall, there should be some shelving it looks like. Open the blue box there
S: Wait w- where- where? Sorry {laughs}

Overall, these results suggest that effective teams were better at coordinating their actions by using various dialogue moves and communication strategies to enhance grounding. Effective directors also played a central role in managing the task responsibilities, and sustained this initiative even under time pressure.

3.2 Disfluencies as collaborative tools

To test our hypothesis that disfluencies serve collaborative functions, we examined group differences in disfluency rate. A MANOVA was conducted on the rates of the four types of self-repairs, with Group and Time Pressure as factors. There was a significant effect of Group ($F(4,33) = 2.787$, $p = .042$) on rates of self-repair disfluencies (see Fig. 3): *Insertions* ($F(1,36) = 4.292$, $p = .046$), *Deletions* ($F(1,36) = 4.414$, $p = .043$), and a trending effect for *Substitutions* ($F(1,36) = 2.826$, $p = .101$). In all cases, the effective group had higher disfluency rates, which was not due to longer utterances because the groups' MLU did not differ ($F < 1$). For hesitation disfluencies, a MANOVA conducted on prolongations, filled- and silent-pauses found no effect of performance group on these disfluency measures ($F < 1$). This was expected because these disfluencies (especially filled pauses) were the most common in the corpus, and did not exhibit speaker differences in previous analyses (Nicholson et al., 2003). Overall, the finding that self-repairs increased for the effective teams supports our hypothesis that these types of disfluencies are not solely due to production difficulty, but rather may serve an interpersonal function.

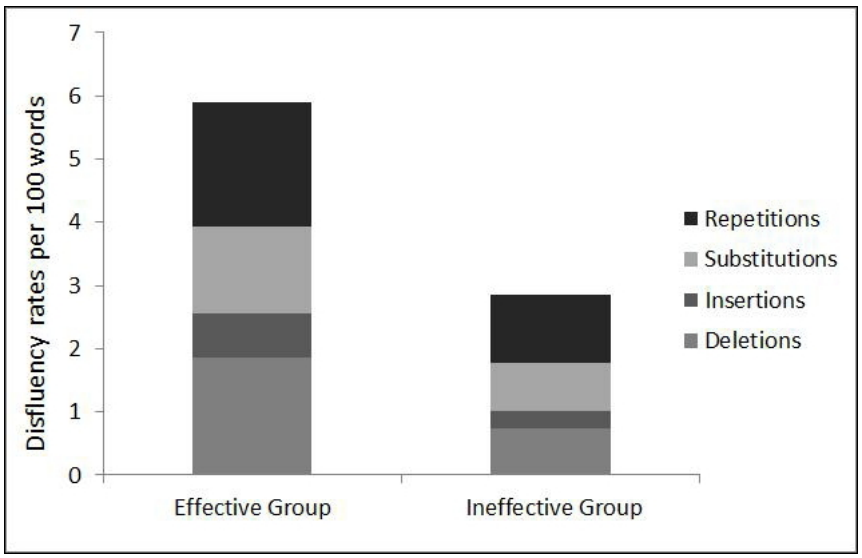


Figure 3: Group effect for disfluency rate

Examples from the corpus further show how disfluencies were used by effective teams as collaborative tools to enhance grounding. Consider the following dialogue exchange from one of the top performing teams under time pressure:

S: Yeah, there's one all the way to the right
D: Okay, you should see- you should [pause] be- go to that end hallway

The director's mixed substitution/deletion self-repair served to make the utterance more accurate and clear for the searcher. Since the director was unsure of exactly what the searcher was seeing, he repaired his utterance and changed it to an instruction that did not provide false information, and that was simpler to understand. This is in line with prior evidence (e.g., Clark and Wilkes-Gibbs (1986)) showing that self-repairs increase when people accommodate their partner's perspective.

Due to the fast-paced nature of the task, some searchers tended to "think out loud" and update their directors with new information as they obtained it. This naturally led to an increase in disfluency, as in the following example from an effective team:

S: But I'm out of- I'm out of- uh [pause] actually, there's a blue one to my left
D: Yes there should be- yes pick that up

In this exchange, the searcher started to say that she is out of blocks, but then changed this mid-sentence (via deletion disfluency) to share new information ("there's a blue one to my left"). Similarly, the director used a substitution to confirm the prior utterance ("yes there should be") and also as a new instruction ("yes pick that up"). The disfluencies here maximized the information that each speaker shared in a single turn, leading to increased efficiency under time pressure.

Effective teams were also able to resolve reference ambiguities through the use of disfluencies, as in the following example:

D: There's also one in the second- [pause] uh, we only have three minutes to do this, okay
S: Okay, second cubicle I got that

Here, the searcher was able to predict that the director was referring to the cubicle, even though this was part of the deleted reparandum, and the word "cubicle" was not even explicitly uttered. The silent pause combined with the non-lexical filler "uh" may have signaled that the director was referring to an object, namely, the cubicle. Despite the fact that this segment of the utterance was deleted, the

searcher was still able to resolve the intended referent by drawing on the mutual knowledge that both teammates share a similar floor plan of the room.

For effective teams in the study, the benefit of clarifying an utterance may have outweighed the cost of disfluent speech because it is cheaper for a person to repair their own speech rather than to have a partner do it for them (Levelt, 1983). In cases where an error was not self-repaired (such as the miscommunication of an important goal), this could have led to misinformation resulting in confusion or loss of efficiency. As a whole, these data suggest that self-repairs served as collaborative tools in the discourse, and were utilized by effective teams to enhance coordination and performance.

4 Discussion

4.1 Summary of results

Overall, the results of our investigation revealed a number of novel effects of disfluencies and dialogue moves in task-oriented remote communication. In our analysis of dialogue moves, we found that the best performing directors used more Check utterances than the ineffective directors (see Fig. 1). Check utterances are used as a means to gauge a partner's understanding, and are especially useful for reducing uncertainty when interlocuters are remotely separated and need to establish shared knowledge. Effective directors also produced more Ready utterances under time pressure than ineffective directors (see Fig. 2), indicating that they showed receptiveness to their partner and maintained control over guiding the team through the changing task conditions. We also observed that more effective teams produced twice as many self-repair disfluencues as the less effective teams. Although previous studies have demonstrated some effects of disfluency on dialogue (Arnold et al., 2007; Brennan and Schober, 2001), ours is the first to link these effects with improved performance in a joint task involving spontaneous speech. Our results indicate that effective teammates were monitoring their own speech and repairing it for clarity and accuracy in order to accommodate their partner's perspective and facilitate grounding. In this way, disfluencies suggest a greater team awareness, which may have contributed to improved performance (see Fig. 3). Overall, the ability to establish common ground through collaborative exchanges seemed to be the key factor in effective team performance. Effective teams were able to accomplish this through particular communication strategies that involved: self-monitoring one's speech for clarity (e.g., self-repairs), being responsive to one's teammate (e.g., Ready moves), as well as monitoring them for understanding (e.g., Check moves).

4.2 Application to spoken dialogue systems

Our results shed light on task-oriented communication and directly inform the development of robust dialogue systems that rely on interactive, collaborative mechanisms. It is important that such systems are able to utilize the information contained in team discourse to gain insight into speakers' cognitive and performance states, and to make better predictions about the course of the interaction. Specifically, the finding that self-repairs are strong indicators of collaborative processes and are increasingly utilized in effective teams, suggests that the detection and interpretation of speech disfluency can be of great benefit to dialogue systems. While there have been numerous approaches to automated disfluency detection (many of which used statistical models trained on the Switchboard corpus, e.g., Qian and Liu (2013)), our findings suggest that these approaches lack several specific components needed for use in robust dialogue systems, including: (1) incremental, on-line processing, (2) identifying the function of the repair, (3) use of the repair to make predictions about subsequent dialogue, (4) use of additional non-linguistic information in the model (e.g., speaker role, cognitive state, etc.), and (5) generalizing to domains other than two-party telephone conversations. While some existing systems attempt to solve these problems individually (e.g., domain generality: Georgila et al. (2010); repair function: Hough et al. (2013); incrementality: Zwarts et al. (2010)), no current approach combines these elements into a unified system. Such an integrated approach is important going forward because it will allow systems to handle the kinds of natural utterances that people make in task-oriented discourse.

Importantly, our findings also highlight the need for dialogue systems to handle incremental input. Given the abundance of time-sensitive information that can be extracted from discourse channels (dis-

fluency, turn-taking, back-channel feedback, etc.), this is an important next step to improve dialogue processing. Though some progress has recently been made on dialogue managers that can handle incremental semantic input (Buß et al., 2010; Schlangen et al., 2009), there has been little to no work on integrating these mechanisms with the planning and reasoning (not to mention vision and motor) capabilities of artificial agents that coordinate their actions with humans. This is a necessary step in order to test the benefit of the collaborative mechanisms our study revealed, and to advance the state of the art of spoken dialogue systems.

4.3 Future directions

Future work will be necessary on both the experimental and computational ends. Future studies will need to evaluate larger samples that are carefully controlled for variables that might influence performance (e.g., gender, age, familiarity, etc.). In addition, workload will need to be objectively measured in order to determine if the effects of time pressure were experienced differently by teams of varying performance. An objective workload measure will also allow for investigation into the *cause* of speech disfluency, specifically in terms of whether it is due to task demands or to intentional signaling (Nicholson et al., 2003).

On the computational end, our results support the development of spoken dialogue systems that can track various dimensions of "team cohesion" based on discourse measures. Such systems can be used in team training exercises in order to identify when a team is under-performing, to find areas of weakness, and to improve overall coordination. An artificial agent with such capacity can also take appropriate actions to repair a problematic interaction through a range of strategies (e.g., *Check* and *Ready* moves, frequent acknowledgments, establishing shared referents, etc.) that we and others found to minimize joint collaborative effort. As a whole, these abilities would improve on the capabilities of existing dialogue systems, and would enable artificial agents to function as more effective teammates.

5 Conclusions

This first study connecting spontaneous speech, dialogue, and task performance supports previous literature and introduces novel findings about the effects of self-repair disfluencies and dialogue moves on coordination and performance in remotely-communicating action teams. The best-performing teams in our study utilized a variety of communication strategies to enhance coordination, including monitoring for understanding, perspective-taking, self-repairs and others. Importantly, we showed that these strategies can be identified by discourse properties available in team communication channels. Applying these empirical results to spoken dialogue systems is an important future direction that can lead to more natural and improved human-agent interaction. By better understanding effective team communication we can design more robust systems that take advantage of the kinds of interactive, collaborative mechanisms inherent in dialogue, to improve coordination and performance in all kinds of human teams.

6 Acknowledgments

This work was supported in part by grants N00014-07-1-1049, N00014-11-1-0493, and N00014-14-1-0149 from the US Office of Naval Research. We would also like to thank the anonymous reviewers for their helpful feedback and suggestions.

References

Jennifer E. Arnold, Carla L. Hudson Kam, and Michael K. Tanenhaus. 2007. If you say thee uh you are describing something hard: the on-line attribution of disfluency during reference comprehension. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 33(5):914–930.

Dale Barr. 2001. Trouble in mind: paralinguistic indices of effort and uncertainty in communication. In C. Cavé, I. Guaïtella, and S. Santi, editors, *Oralité et gestualité: interactions et comportements multimodaux dans la communication: actes du colloque ORAGE 2001*, pages 597–600. Harmattan, Paris.

Andre Berthold and Anthony Jameson. 1999. Interpreting symptoms of cognitive load in speech input. In Judy Kay, editor, *UM99, User Modeling: Proceedings of the Seventh International Conference*, pages 235–244. Springer Wien New York.

Heather Bortfeld, Silvia D. Leon, Jonathan E. Bloom, Michael F. Schober, and Susan E. Brennan. 2001. Disfluency rates in conversation: Effects of age, relationship, topic, role, and gender. *Language and speech*, 44(2):123–147.

Susan E. Brennan and Michael F. Schober. 2001. How Listeners Compensate for Disfluencies in Spontaneous Speech. *Journal of Memory and Language*, 44(2):274–296.

Susan E. Brennan, Alexia Galati, and Anna K. Kuhlen. 2010. Two minds, one dialog: Coordinating speaking and understanding. *Psychology of Learning and Motivation*, 53:301–344.

Okko Buß, Timo Baumann, and David Schlangen. 2010. Collaborating on utterances with a spoken dialogue system using an isu-based approach to incremental dialogue management. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 233–236.

Jean Carletta, Stephen Isard, Gwyneth Doherty-Sneddon, Amy Isard, Jacqueline C. Kowtko, and Anne H. Anderson. 1997. The reliability of a dialogue structure coding scheme. *Computational Linguistics*, 23(1):13–31.

Herbert H. Clark and Susan E. Brennan. 1991. Grounding in communication. *Perspectives on Socially Shared Cognition*, 13:127–149.

Herbert H. Clark and Meredyth A. Krych. 2004. Speaking while monitoring addressees for understanding. *Journal of Memory and Language*, 50(1):62–81.

Herbert H. Clark and Edward F. Schaefer. 1989. Contributing to discourse. *Cognitive science*, 13(2):259–294.

Herbert H. Clark and Deanna Wilkes-Gibbs. 1986. Referring as a collaborative process. *Cognition*, 22(1):1–39.

Herbert H. Clark. 1996. *Using language*. Cambridge university press.

Martin Corley, Lucy J. MacGregor, and David I. Donaldson. 2007. Its the way that you, er, say it: Hesitations in speech affect language comprehension. *Cognition*, 105(3):658–668.

Gwyneth Doherty-Sneddon, Anne Anderson, Claire O'Malley, Steve Langton, Simon Garrod, and Vicki Bruce. 1997. Face-to-face and video-mediated communication: A comparison of dialogue structure and task performance. *Journal of Experimental Psychology: Applied*, 3(2):105–125.

Kathleen M. Eberhard, Hannele Nicholson, Sandra Kübler, Susan Gundersen, and Matthias Scheutz. 2010. The Indiana "Cooperative Remote Search Task" (CReST) corpus. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23*.

Elliot E. Entin and Daniel Serfaty. 1999. Adaptive Team Coordination. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 41(2):312–325.

Kallirroi Georgila, Ning Wang, and Jonathan Gratch. 2010. Cross-domain speech disfluency detection. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 237–240.

Julian Hough, Matthew Purver, et al. 2013. Modelling expectation in the self-repair processing of annotat-, um, listeners. In *Proceedings of the 17th SemDial workshop on the Semantics and Pragmatics of Dialogue*.

M Asif Khawaja, Fang Chen, and Nadine Marcus. 2012. Analysis of collaborative communication for linguistic cues of cognitive load. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 54(4):518–529.

Robert M. Krauss and Sidney Weinheimer. 1966. Concurrent feedback, confirmation, and the encoding of referents in verbal communication. *Journal of personality and social psychology*, 4(3):343.

Willem Levelt. 1983. Monitoring and self-repair in speech. *Cognition*, 14(1):41–104.

Robin Lickley. 1998. *HCRC Disfluency Coding Manual*. Human Communication Research Centre, University of Edinburgh.

Anders Lindström, Jessica Villing, Staffan Larsson, Alexander Seward, and Cecilia Holtelius. 2010. The effect of cognitive load on disfluencies during in-vehicle spoken dialogue. In *Proceedings of the International Conference on Spoken Language Processing, Interspeech, 17-23*.

Hannele Nicholson, Ellen Gurman Bard, Rohin Lickley, Anne H. Anderson, Jim Mullin, David Kenicer, and Lucy Smallwood. 2003. The intentionality of disfluency: Findings from feedback and timing. In *ISCA Tutorial and Research Workshop on Disfluency in Spontaneous Speech*.

Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In *Proceedings of the NAACL-HLT*.

David Schlangen, Timo Baumann, and Michaela Atterer. 2009. Incremental reference resolution: The task, metrics for evaluation, and a bayesian filtering model that is sensitive to disfluencies. In *Proceedings the 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 30–37.

Thomas Schmidt and Kai Wörner. 2009. Exmaralda-creating, analyzing and sharing spoken language corpora for pragmatics research. *Pragmatics*, 19(4):565–582.

Daniel Serfaty, Elliot E. Entin, and Catherine Volpe. 1993. Adaptation to stress in team decision-making and coordination. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 37, pages 1228–1232.

Vicki L. Smith and Herbert H. Clark. 1993. On the course of answering questions. *Journal of Memory and Language*, 32(1):25–38.

Eric Sundstrom, Michael McIntyre, Terry Halfhill, and Heather Richards. 2000. Work groups: From the Hawthorne studies to work teams of the 1990s and beyond. *Group Dynamics: Theory, Research, and Practice*, 4(1):44–67.

Marc Swerts. 1998. Filled pauses as markers of discourse structure. *Journal of Pragmatics*, 30:485–496.

Katia Sycara and Gita Sukthankar. 2006. Literature review of teamwork models. Technical Report CMU-RI-TR-06-50, Robotics Institute, Pittsburgh, PA.

Julie M. Urban, Jeanne L. Weaver, Clint A. Bowers, and Lori Rhodenizer. 1996. Effects of workload and structure on team processes and performance: Implications for complex team decision making. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 38(2):300–310.

Simon Zwarts, Mark Johnson, and Robert Dale. 2010. Detecting speech repairs incrementally using a noisy channel approach. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1371–1378.

A Neural Network Approach for Knowledge-Driven Response Generation

Pavlos Vougiouklis **Jonathon Hare** **Elena Simperl**
Electronics and Computer Science
University of Southampton
Southampton, UK
{pv1e13, jsh2, e.simperl}@ecs.soton.ac.uk

Abstract

We present a novel response generation system. The system assumes the hypothesis that participants in a conversation base their response not only on previous dialog utterances but also on their background knowledge. Our model is based on a Recurrent Neural Network (RNN) that is trained over concatenated sequences of comments, a Convolution Neural Network that is trained over Wikipedia sentences and a formulation that couples the two trained embeddings in a multimodal space. We create a dataset of aligned Wikipedia sentences and sequences of Reddit utterances, which we we use to train our model. Given a sequence of past utterances and a set of sentences that represent the background knowledge, our end-to-end learnable model is able to generate context-sensitive and knowledge-driven responses by leveraging the alignment of two different data sources. Our approach achieves up to 55% improvement in perplexity compared to purely sequential models based on RNNs that are trained only on sequences of utterances.

1 Introduction

Over the recent years, the level of users' engagement and participation in public conversations on social media, such as Twitter, Facebook and Reddit has substantially increased. As a result, we now have large amounts of conversation data that can be used to train computer programs to be proficient conversation participants. Automatic response generation could be immediately deployable in social media as an auto-complete response suggestion feature or a conversation stimulant that adjusts the participation interest in a dialogue thread (Ritter et al., 2011). It should also be beneficial in the development of Question-Answering systems, by enhancing their ability to generate human-like responses (Grishman, 1979).

Recent work on neural networks approaches shows their great potential at tackling a wide variety of Natural Language Processing (NLP) tasks (Bengio et al., 2003; Mikolov et al., 2010). Since a conversation can be perceived as a sequence of utterances, recent systems that are employed in the automatic response generation domain are based on Recurrent Neural Networks (RNNs) (Sordoni et al., 2015; Shang et al., 2015; Vinyals and Le, 2015), which are powerful sequence models. These systems base their generated response explicitly on a sequence of the most recent utterances of a conversation thread. Consequently, the sequence of characters, words, or comments, in a conversation, depending on the level of the model, is the only means with which these models achieve contextual-awareness, and in open-domain, realistic, situations it often proves inadequate (Vinyals and Le, 2015).

In this paper we address the challenge of context-sensitive response generation. We build a dataset that aligns knowledge from Wikipedia in the form of sentences with sequences of Reddit utterances. The dataset consists of sequences of comments and a number of Wikipedia sentences that were allocated randomly from the Wikipedia pages to which each sequence is aligned. The resultant dataset consists of $\sim$ 15k sequences of comments that are aligned with $\sim$ 75k Wikipedia sentences. We make the aligned corpus available at `github.com/pvougiou/Aligning-Reddit-and-Wikipedia`.

We propose a novel model that leverages this alignment of two different data sources. Our architecture is based on coupling an RNN using either Long Short-Term Memory (LSTM) cells (Hochreiter and

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3370–3380, Osaka, Japan, December 11-17 2016.

Schmidhuber, 1996) or Gated Recurrent Units (GRUs) (Cho et al., 2014) that processes each sequence of utterances word-by-word, and a Convolutional Neural Network (CNN) that extracts features from each set of sentences that corresponds to this sequence of utterances. We pre-train the CNN component (Kim, 2014) on a subset of the retrieved Wikipedia sentences in order to learn filters that are able to classify a sentence based on its referred topic. Our model assumes the hypothesis that each participant in a conversation bases their response not only on previous dialog utterances but also on their individual background knowledge. We use Wikipedia[1] as the source of our model's knowledge background and align Wikipedia pages and sequences of comments from Reddit[2] based on a predefined topic of discussion.

Our work is inspired by recent developments in the generation of textual summaries from visual data (Socher et al., 2014; Karpathy and Li, 2014). Our core insight stems from the idea that a system that is able to learn how to couple information from aligned datasets in order to produce a meaningful response, would be able to capture the context of a given conversation more accurately.

Our model achieves up to 55% improved perplexity compared to purely sequential equivalents. It should also be noted that our approach is domain independent; thus, it could be transferred out-of-box to a wide variety of conversation topics.

The structure of the paper is as follows. Section 2 discusses premises of our work regarding both automatic response generation and neural networks approaches for Natural Language Processing (NLP). Section 3 presents the components of the network. Section 4 describes the structure of the dataset. Section 5 discusses the experiments and the evaluation of the models. Section 6 summarises the contributions of the current work and outlines future plans.

2 Related Work

Since Bengio's introduction of neural networks in statistical language modelling (Bengio et al., 2003) and Mikolov's demonstration of the extreme effectiveness of RNNs for sequence modelling (Mikolov et al., 2010), neural-network-based implementations have been employed for a wide variety of NLP tasks. In order to sidestep the *exploding* and *vanishing gradients* training problem of RNNs (Bengio et al., 1994; Pascanu et al., 2012), multi-gated RNN variants, such as the GRU (Cho et al., 2014) and the LSTM (Hochreiter and Schmidhuber, 1996), have been proposed. Both GRUs and LSTMs have demonstrated state-of-the-art performance for many generative tasks, such as SMT (Cho et al., 2014; Sutskever et al., 2014), text (Graves, 2013) and image generation (Gregor et al., 2015).

Despite the fact that CNNs had been originally employed in the computer vision domain (LeCun et al., 1998), models based on the combination of the convolution operation with the classical Time-Delay Neural Network (TDNN) (Waibel et al., 1989) have proved effective on many NLP tasks, such as semantic parsing (Yih et al., 2014), Part-Of-Speech Tagging (POS) and Chunking (Collobert and Weston, 2008). Furthermore, sentence-level CNNs have been used in sentiment analysis and question type identification (Kalchbrenner et al., 2014; Kim, 2014).

The concept of a system capable of participating in human-computer conversations was initially proposed by Weizenbaum (Weizenbaum, 1966). Weizenbaum implemented ELIZA, a keyword-based program that set the basis for all the descendant *chatterbots*. In the years that followed, many template-based approaches (Isbell et al., 2000; Walker et al., 2003; Shaikh et al., 2010) have been suggested in the scientific literature, as a way of transforming the computer into a proficient conversation participant. However, these approaches usually adopt variants of the nearest-neighbour method to facilitate their response generation process from a number of limited sentence paradigms and, as a result, they are limited to specific topics or scenarios of conversation. Recently, models for Statistical Machine Translation have been used to generate short-length responses to a conversational incentive from Twitter utterances (Ritter et al., 2011). In the recent literature, RNNs have been used as the fundamental component of conversational response systems (Sordoni et al., 2015; Shang et al., 2015; Vinyals and Le, 2015). Even though these systems exhibited significant improvements over SMT-based methods (Ritter et al., 2011), they either adopt the length-restricted-messages paradigm or are trained on idealised dataset that undermines the

[1] http://www.wikipedia.com
[2] https://www.reddit.com

generation of responses in open domain realistic scenarios.

We propose a novel architecture for context-sensitive response generation. Our model is trained on a dataset that consists of realistic sequences of Reddit comments that aligned with sets of Wikipedia sentences. We use an RNN and a CNN components to process the sequence of comments and their corresponding set of sentences respectively and we introduce a learnable coupling formulation. The coupling formulation is inspired by the Multimodal RNN that generates textual description from visual data (Karpathy and Li, 2014). However, unlike Karpathy's approach, we do not allow the feature that is generated by the CNN component to diminish between distant timesteps (i.e. Section 3.3).

3 Our Model

Our task is to generate a context-sensitive response to a sequence of comments by incorporating background knowledge. The proposed model is based on the assumption that each participant in a conversation phrases their responses by taking into consideration both the past dialog utterances and their individual knowledge background. We train the model on a set of M sequences of Reddit comments and N summaries of Wikipedia pages that are related to the main discussed topic of a conversation. During training our models takes as an input a sequence of one-hot[3] vector representations of words $x_1, x_2, ..., x_T$ from a sequence of comments and a group of sentences S that is aligned with this sequence of utterances. We use a sentence-level CNN, which processes the group of sentences, in parallel with a word-level RNN that processes the sequence of comments in batches and propose a formulation that learns to couple the two networks to produce a meaningful response to the preceding comments. We experiment with two different commonly used RNN variants that are based on: (i) the LSTM cell and (ii) the GRU. We pre-train our CNN sentence model on a subset of the Wikipedia-sentences dataset in order for it to learn to classify a sentence based on the topic-keyword that was matched for its corresponding page acquisition.

Please note that since *bias* terms can be included in each weight-matrix multiplication (Bishop, 1995), they are not explicitly displayed in the equations that describe the models of this section.

3.1 Sentence Modelling

Models based on CNNs achieve their basic functionality by convolving a sequence of inputs with a set of filters in order to extract local features. We adopt the Convolutional Sentence Model from (Kim, 2014) and we expand it in order to meet our specific needs for a multi-class, rather than binary classification. Let $t_{1:l}$ the concatenation of the vectors of all the words that exist in a sentence s. A *narrow* type convolution operation with a filter $m \in \mathbb{R}^{k \times m}$ is applied to each m-gram in the sentence s in order to produce a *feature map* $\mathbf{c_{mf}} \in \mathbb{R}^{l-m+1}$ of features:

$$c_{mf_j} = tanh(m^T t_{j-m+1:j}) \ , \tag{1}$$

$$\mathbf{c_{mf}} = \begin{bmatrix} c_{mf_1} \\ \vdots \\ c_{mf_3} \end{bmatrix} \ , \tag{2}$$

with $l \geq m$. Shorter sentences are *padded* with zero vectors when necessary. The most relevant feature from each feature map is captured by applying the max-over-time pooling operation (Collobert and Weston, 2008). The consequent matrix is the result of concatenating the max values from each feature map that has been produced by applying an f number of m length filters over the sentence s. The network results in a fully-connected layer and a $softmax$ that carries out the classification of the sentences. The architecture of the sentence model is illustrated on the left side of Figure 1.

3.2 Sequence Modelling

We describe two commonly used RNN variants that are based on: (i) the LSTM cell and (ii) the GRU. We experiment with both of them in order to explore which one serves better the sequential-modelling

[3]Each x_t refers to the one-hot representation vector of a vocabulary token. One-hot is a vector that contains a 1 at the index of this particular x_t token in the vocabulary with all the other values set to zero.

needs of our full architecture.

Let $h_t^l \in \mathbb{R}^n$ be the aggregated output of a hidden unit at timestep $t = 1...T$ and layer depth $l = 1...L$. The vectors at zero layer depth, $h_t^0 = \mathbf{W_{x \rightarrow h}} x_t$, represent vectors that are given to the network as an input. The parameter matrix $\mathbf{W_{x \rightarrow h}}$ has dimensions $[|X|, n]$, where $|X|$ is the cardinality of all the potential one-hot input vectors. All the matrices that follow have dimension $[n, n]$.

3.2.1 Long Short-Term Memory

Our LSTM cells' architecture is adopted from (Zaremba and Sutskever, 2014):

$$in_t^l = sigm(\mathbf{W_{in}^l} h_t^{l-1} + \mathbf{W_{h \rightarrow in}^l} h_{t-1}^l) \ , \tag{3}$$

$$f_t^l = sigm(\mathbf{W_f^l} h_t^{l-1} + \mathbf{W_{h \rightarrow f}^l} h_{t-1}^l) \ , \tag{4}$$

$$cell_t^l = f_t^l \odot cell_{t-1}^l + in_t^l \odot tanh(\mathbf{W_{cell}^l} h_t^{l-1} + \mathbf{W_{h \rightarrow cell}^l} h_{t-1}^l) \ , \tag{5}$$

$$out_t^l = sigm(\mathbf{W_{out}^l} h_t^{l-1} + \mathbf{W_{h \rightarrow out}^l} h_{t-1}^l) \ , \tag{6}$$

$$h_t^l = out_t^l \odot tanh(cell_t^l) \ , \tag{7}$$

where in_t^l, f_t^l, out_t^l and $cell_t^l$ are the vectors at timestep t and layer depth l that correspond to the *input gate*, the *forget gate*, the *output gate* and the *cell* respectively.

3.2.2 Gated Recurrent Unit

The Gated Recurrent Unit was proposed as a less-complex implementation of the LSTM (Cho et al., 2014).

$$reset_t^l = sigm(\mathbf{W_{reset}^l} h_t^{l-1} + \mathbf{W_{h \rightarrow reset}^l} h_{t-1}^l) \ , \tag{8}$$

$$update_t^l = sigm(\mathbf{W_{update}^l} h_t^{l-1} + \mathbf{W_{h \rightarrow update}^l} h_{t-1}^l) \ , \tag{9}$$

$$\widetilde{h_t^l} = tanh(\mathbf{W_{in}^l} h_t^{l-1} + \mathbf{W_{h \rightarrow h}^l} (reset_t^l \odot h_{t-1}^l)) \ , \tag{10}$$

$$h_t^l = (1 - update_t^l) \odot h_{t-1}^l + update_t^l \odot \widetilde{h_t^l} \ , \tag{11}$$

where $reset_t^l$, $update_t^l$ and $\widetilde{h_t^l}$ are the vectors at timestep t and layer depth l that represent the values of the *reset gate*, the *update gate* and the *hidden candidate* respectively.

3.3 Coupling

After the pre-training of the CNN is complete, and the fully-connected and $softmax$ layers are removed, the CNN is connected to the hidden units of the last layer L of the recurrent component. This is illustrated in Figure 1. The recurrent component is implemented with either LSTMs or GRUs. At each timestep, the RNN is processing a word from a sequence of comments and the CNN is extracting local features by convolving this sequence's corresponding sentences with groups of differently sized filters. The red-coloured edges in Figure 1 represent the learnable parameters during training.

The coupling formulation that follows is inspired by the Multimodal RNN that generates textual description from visual data (Karpathy et al., 2014). Since, we do not want to allow the effect of sentence features, which represent the background knowledge of our model, to diminish between distant timesteps we differentiate from Karpathy's approach; and instead of providing the feature that is generated by the CNN to the RNN only at the first timestep, we provide it at every timestep. Furthermore, Karpathy employs the *simple* RNN or Elman network (Elman, 1990) as the sequence modelling component of his architecture whereas we adopt multi-gated RNN variants.

It should be noted that in the equations that follow, the term $CNN_h \in \mathbb{R}^{\sum \mathbb{MF}}$ would refer to the output of the sentence-level CNN with its fully-connected and $softmax$ layers disconnected, where $\mathbb{MF}$ is the group of all the feature maps that are generated for each different filter size. During training the resultant embeddings, which are computed by the CNN_h processing a group of sentences S and the RNN variant

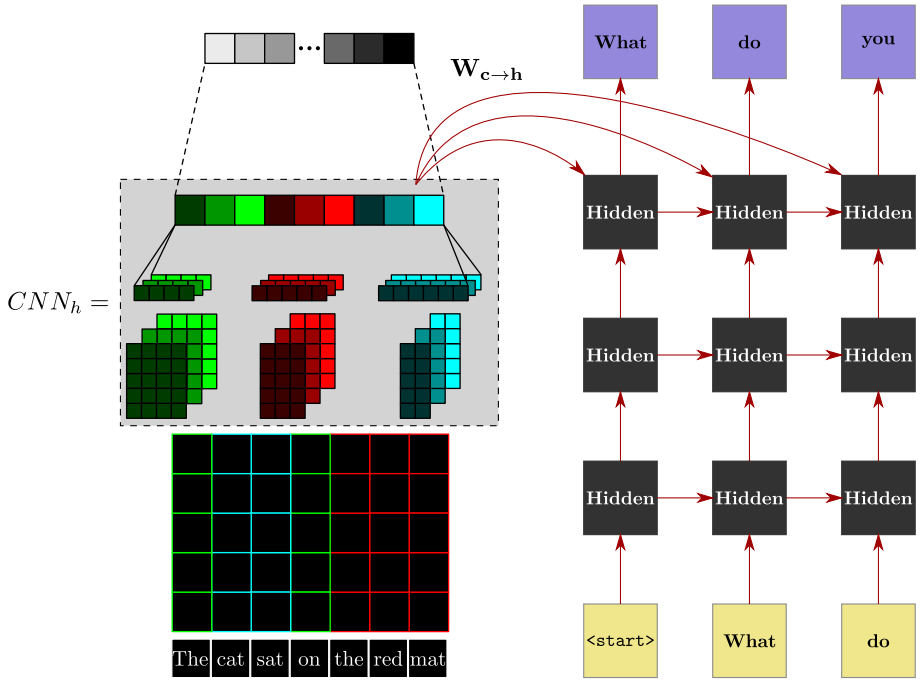


Figure 1: The architecture of our generative model. At each timestep, the RNN is processing a word from a sequence of comments and the CNN_h is extracting local features by convolving this sequence's corresponding sentences with a set of three differently sized filters. The red-coloured edges are the learnable parameters during training. Each comment in a sequence is augmented with start-of-comment `<start>` and end-of-comment `<end>` tokens.

processing the sequence of one-hot input word vectors $x_1, x_2, ..., x_T$ from the corresponding sequence of comments, are coupled in a hidden state $h_1, h_2, ..., h_T$. The prediction for the next word is computed by projecting this hidden state h_t to sequence of outputs $y_1, y_2, ..., y_T$:

$$c_S = \mathbf{W_{c \to h}} CNN_h(S) \ , \tag{12}$$

$$h_t = h_t^L \odot c_S \ , \tag{13}$$

$$y_t = softmax(\mathbf{W_y} h_t). \tag{14}$$

4 Dataset

We create a dataset[4] of aligned sentences from Wikipedia and sequences of utterances from Reddit. A shared, fixed, vocabulary was used for both data sources. We treat Wikipedia as a "cleaner" data source and we formed our vocabulary in the following manner. First, we included all the words that occur 2 or more times in the Wikipedia sentences. Subsequently, from the Reddit sequences of comments, we included any words that occur at least 3 times across both data sources. The resultant shared dictionary includes 56280 of the most frequent words. Every out-of-vocabulary word is represented by a special `NaN` token.

In constructing our dataset, our goal is to align Wikipedia sentences with sequences of comments from Reddit. We found that topics related to philosophy and literature are discussed on Reddit with the exchange of longer and more elaborative messages than the responses of the majority of conversational subjects on social media. A dataset that consists of long and detailed responses would provide more room for conversation incentives and would allow us to investigate the performance of our architectures against dialog exchanges with longer comments. We compiled a list of 35 predetermined topic-keywords from the philosophical and literary domain. By utilising the `search` feature of both the Reddit API[5]

[4] `github.com/pvougiou/Aligning-Reddit-and-Wikipedia`
[5] reddit.com/dev/api

and the MediaWiki API[6], we extracted: (i) sequences of comments from conversational threads most related to each keyword and (ii) the 300 Wikipedia pages most related to each keyword after carrying out Wikipedia's automatic disambiguation procedure. In order to increase the homogeneity of the dataset in terms of the length of both the sequences of comments and the sentences, we excluded sequences and sentences whose length exceeded: (i) $\overline{len} - \sigma = 1140$ and (ii) $\overline{len} - 2 \cdot \sigma = 54$ words respectively. The resultant dataset consists of 15460 sequences of Reddit comments and 75100 Wikipedia sentences.

4.1 Reddit

Reddit is absolved from the length-restricted-messages paradigm, facilitating the generation of longer and more meaningful responses. Furthermore, Reddit serves as an openly-available question-answering platform. Our research hypothesis is that a neural network trained on sequences of questions and their corresponding answers will be able to generate responses that escape the concept of daily-routine expressions, such as "good luck" or "have fun", and facilitate a playground for more detailed and descriptive dialogue utterances.

Each different conversation on Reddit starts with a user submitting a *parent-comment* on a subreddit. A sequence of utterances then succeeds this parent comment. Since we wanted to investigate how our model performs against long-term dependant dialog components, we set the depth of conversation to 5. Starting from the parent-comment, we follow the direction of the un-ordered tree of utterances until the fourth-level child-comment. If a comment (node) leads to n responses (children), we copy the observed sequence n times and for each sequence, we continue until the fourth response (leaf). Based on the above structural paradigm, we extracted sequences with at least four children-comments, of which we retained the only first four utterances along with the original parent-comment of the sequence. Note that each comment in a sequence is augmented with the respective start-of-comment `<start>` and end-of-comment `<end>` tokens.

Topic	Reddit Sequence of Comments	Wikipedia Sentences
Noam Chomsky	<start> Noam Chomsky: Bernie Sanders is Not a Radical. He has Mass Support for Positions on Healthcare & Taxes <end> <start> Funny, because Bernie Sanders's idol Eugene Debs ran against FDR <end> <start> Maybe Clinton will be FDR <end> <start> Watch out, Japanese. <end> <start> Japanese You misspelled Syrians <end>	For Chomsky, there are minimalist questions but the answers can be framed in any theory. ⋮ Minimalism in structured writing or topic-based authoring is based on the ideas of John Millar Carroll. Minimalism is about reducing the interference of the information with the users sense-making process. An error, in fact, is the teachable moment that the content can exploit.

Table 1: Example of the alignment of our dataset. One sequence of comments is coupled with a set of sentences. The sentences are randomly allocated from the Wikipedia pages which have been extracted based on the same search term (**Noam Chomsky**) as the corresponding sequence.

4.2 Wikipedia

Wikipedia sentences are used as the knowledge background of our model. We chose to include only sentences from the Wikipedia summaries, since in preliminary experiments, we found that including all the textual material of a page introduces a lot of noise to our data. The 13410 Wikipedia summaries that matched the search criteria were split into sentences. Each sentence was labelled with the initial topic-keyword that was matched for its corresponding page acquisition. A subset of 30000 labelled sentences was used for pre-training the CNN component of our architecture.

[6] mediawiki.org/wiki/API:Main_page

4.3 Dataset Alignment

We choose to align each sequence of Reddit utterances with 20 Wikipedia sentences. Both the Wikipedia pages to which the sentences correspond and the sequence of comments have been extracted using the same search-term. An example of the structure of the dataset is displayed in Table 1.

5 Experiments

The full network was regularised by introducing a dropout (Zaremba et al., 2014) value of 0.4 to the non-recurrent connections between the last hidden state, h_t, and the softmax layer of the network. In order to avoid any potential exploding gradients training problems, we enforce an l_2 constraint on the gradients of the weights in order for them to be no greater than 5 (Sutskever et al., 2014).

- The CNN component is trained with narrow convolutional filters of widths 3, 4, 5 and 6, with 300 feature maps each. We use the rectifier as activation function. All of the parameters were initialised with a random uniform distribution between -0.1 and 0.1. The network was trained for 10 epochs using stochastic gradient descent with a learning rate of 0.2. We regularised the network by introducing a dropout (Hinton et al., 2012) value of 0.7 to the connections between the pooling and the softmax layer of the network.

- For the recurrent component of our networks, we use 2 layers of (i) 1000 LSTM cells and (ii) 1000 GRUs, resulting in approximately 16M and 12M recurrent connections respectively. All of the parameters are initialised with a random uniform distribution between -0.08 and 0.08. The networks were trained for 10 epochs, using stochastic gradient descent with a learning rate of 0.5. After the 7th epoch in the LSTM case and 3rd epoch in the GRU case, the learning rate was decayed by 0.2 every half epoch.

The dataset is split into training, validation and test with respective portions of 80, 10 and 10. A sample of responses that is generated by our proposed systems is shown in Table 2.

5.1 Experimental Results

Examples of responses that are generated by our proposed systems and their respective purely sequential equivalents are shown in Table 2. The sequences of comments and their corresponding sentences are sampled randomly from the test set. Our architectures learn to couple information that exists in the sequence of comments with knowledge that is contained in the Wikipedia sentences and is, potentially, related to context of those comments.

When a piece of information in the sequence of comments is successfully aligned with the content of its corresponding Wikipedia sentences a knowledgeable, context-sensitive response is generated. A representative example of this functionality is provided in the last sequence of comments in Table 2, where the context of the sequence is coupled with the fact that *Chomsky supported Bernie Sanders in the United States presidential election* (i.e. from the allocated to that sequence of Reddit utterances Wikipedia sentence: "In late 2015, Chomsky announced his support for Vermont U.S. senator Bernie Sanders in the upcoming 2016 United States presidential election."[7]). In case no information alignment is identified between the content of the sequence of comments and the Wikipedia sentences, the generation procedure is based almost explicitly on the sequence of utterances, and a response is generated in a similar to the purely sequential models' fashion.

5.2 Automatic Evaluation

We use perplexity on the test set to evaluate our proposed models against their purely sequential equivalents. Perplexity measures the cross-entropy between the predicted sequence of words and the actual, empirical, sequence of words. The results are illustrated in the top part of Table 3. Our proposed architectures achieve 55% and 45% improvement in perplexity compared to their respective purely LSTM- and GRU-based equivalents.

[7] en.wikipedia.org/wiki/Noam_Chomsky

Sequence of Comments	<start> My grandfather had a miniature set of all of Shakespeare's plays. It's 111 years old. <end> <start> Had? What happened to them? <end> <start> Less what happened to them, more what happened to him. <end> <start> I may have gone to your grandfather's estate sale. <end> <start> Literally as far away as you could be and stay in the same country. Also, we aren't having an estate sale. <end>	<start> Stephen Colbert & Stephen King <end> <start> This post made me realise that King played Bachman in Sons of Anarchy. He was fantastic in the role as well. <end> <start> Fun Fact: Stephen King writes under the pseudonym "Richard Bachman". <end> <start> Fun Fact: Richard NaN cause of death, per his creator, was "cancer of the pseudonym, a rare form of NaN." <end> <start> [Oh, here's a fun fact:] <end>
LSTM	<start> I don't. <end>	NaN.
GRU	<start> You are not so bad. <end>	<start> I meant that I was the one that was the first time to point. <end>
LSTM Coupled With ConvNet	<start> I was thinking the same. I saw it as a kid and I found Shakespeare and I was working in the head for a long time. <end>	<start> I was just thinking the same thing. <end>
GRU Coupled With ConvNet	<start> I don't know what I heard. <end>	<start> I remember searching for other people [spoiler] ("Apparently nothing is not "in" the books"). My personal favorite authors, not a whole story, but as a try to cause it to me. <end>
Sequence of Comments	<start> Can you write a six word story? According to legend, Ernest Hemingway once won a bet by writing a six word story comprised of these words: For sale. Baby shoes. Never worn. What's the best six word story you can write? Mine is, "She lied about having the procedure." <end> <start> I came; I saw; I conquered. <end> <start> "Veni, vidi, vici" said it better <end> <start> A better love story than twilight <end> <start> I'm fucking tired of this joke. <end>	<start> Noam Chomsky: The US would be recognized as a top terrorist state if international law was applied. <end> <start> You're not connecting the dots enough. The "regular" America gets cheap goods and oil through the bellicose actions of the "harsh" America. Now increasingly the actions of "harsh" America tend to benefit a tiny subset of the American people, but let's not pretend like our actions around the world don't have a multitude of practical effects on the American people as a whole. <end> <start> The thinking that if we didn't oppress people around the globe we wouldn't be able to maintain our lifestyle is misguided at best. <end> <start> So you're saying there's no economic benefit to our actions? I think you're mistaken. <end>
LSTM	<start> I think the "cause" is a good idea. <end>	<start> I don't think you are saying that it is a good idea. <end>
GRU	<start> No, it's not. <end>	<start> I would say that the "best" in the way we could have to do, and it is a big one. <end>
LSTM Coupled With ConvNet	<start> I don't think it's a mischaracterization to attribute this man's suicide to paranoia brought on by shady government surveillance. He was depressed, probably because of the holy few stories, the first step is to not like the world of his relationships, because he talks to the first one. <end>	<start> I don't think the majority of society has nothing to do with Sanders terrorism. <end>
GRU Coupled With ConvNet	<start> I think some of his memories was a family. <end>	<start> This is very radical I think Chomsky is an anarcho-syndicalist. An elementary linguistics student, he has no choice who is where he is. He does not need to be a "free market". He has to believe that the potential for American interests would be impossible, but they would be appropriate to charge what the outcome of American foreign interests is. <end>

Table 2: Sample of responses that are generated by our proposed systems and their sequential equivalents. The sequences of comments and their corresponding sentences are sampled randomly from the test set.

Model	LSTM	LSTM Coupled With ConvNet	GRU	GRU Coupled With ConvNet
Perplexity	4.301	**1.905**	3.749	2.051
Average Rating (σ)	**2.65** ($\pm$1.167)	2.4 ($\pm$1.27)	2.5 ($\pm$1.359)	**2.65** ($\pm$1.561)

Table 3: **Top:** Automatic evaluation with the perplexity metric on the test set. **Bottom:** Average rating of the responses that are generated by each model based on human evaluation.

5.3 Human Evaluation

Human evaluation was conducted using research students and staff from the School of Electronics and Computer Science of the University of Southampton. The evaluators were provided with a table of 10 randomly selected sequences of Reddit comments along with the response that is generated by our proposed models and their purely sequential equivalents. In order to simplify our task, we included only sequences of comments with a length less than 100 words. The name of the models to which each response corresponds were anonymised. The authors excluded themselves from this evaluation procedure. The evaluators were asked to rate each generated response from 1 to 5, with 1 indicating a very bad response, based on how well it fits the context of the corresponding sequence of comments.

Table 3 presents the average rating of each model's responses based on human evaluation. Even though our decision to apply a restriction over the length of the sequences of utterances, which were included in the human evaluation experiment, brings us in an agreement with literature that challenges the reliability of automatic evaluation methods, such as perplexity, in the domain of short-length responses (Ritter et al., 2011), we argue that an experiment at a larger scale, absolved from significant simplification choices, would demonstrate an alignment between the human judgements and the automatic evaluation results.

6 Conclusion

To the best of our knowledge this work constitutes the first attempt for building an end-to-end learnable system for automatic context-sensitive response generation by leveraging the alignment of two different data sources. We proposed a novel system that incorporates background knowledge in order to capture the context of a conversation and generate a meaningful response.

This paper made the following contributions: We built a dataset that aligns knowledge from Wikipedia in the form of sentences with sequences of Reddit utterances; and, we designed a neural network architecture that learns to couple information from different types of textual data in order to capture the context of a conversation and generate a meaningful response. Our approach achieved up to 55% improvement in perplexity compared to purely sequential models based on RNNs that are trained only on sequences of utterances. It should also be noted that despite the fact that our dataset focuses on the philosophical and literary domain, the design procedure could be transferred out-of-the-box to a great variety of domains.

Arguments could be made against the performance gain of our architectures against human evaluation. Based on the reported low performance of purely LSTM-based models on very long-term dependant datasets (Sutskever et al., 2014), we believe that an experiment at a larger scale without a restriction over the length of the sequences of utterances (Section 5.3) would emphasise the superiority of our approach.

We believe that further work on the coupling formulation that is proposed in Section 3.3 could provide additional improvements to the results of this work. An additional direction for future work could be the introduction of a complimentary, to the current procedure, task that would enhance the quality of the information alignment from the two data sources.

Acknowledgements

This research is partially supported by the Answering Questions using Web Data (WDAQua) project, a Marie Skłodowska-Curie Innovative Training Network, part of the Horizon 2020 programme. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research.

References

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, March.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

Christopher M. Bishop. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623.

Ralph Grishman. 1979. Response generation in question answering systems. In *Proceedings of the 17th Annual Meeting on Association for Computational Linguistics*, ACL '79, pages 99–101, Stroudsburg, PA, USA. Association for Computational Linguistics.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Sepp Hochreiter and Jrgen Schmidhuber. 1996. Bridging long time lags by weight guessing and "long short term memory". In *Spatiotemporal Models in Biological and Artificial Systems*, pages 65–72. IOS Press.

Charles Lee Isbell, Jr., Michael J. Kearns, Dave Kormann, Satinder P. Singh, and Peter Stone. 2000. Cobot in lambdamoo: A social statistics agent. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 36–41. AAAI Press.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *CoRR*, abs/1404.2188.

Andrej Karpathy and Fei-Fei Li. 2014. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306.

Andrej Karpathy, Armand Joulin, and Fei Li. 2014. Deep fragment embeddings for bidirectional image sentence mapping. In Z. Ghahramani, M. Welling, C. Cortes, N.d. Lawrence, and K.q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1889–1897. Curran Associates, Inc.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Japan, September 26-30, 2010*, pages 1045–1048.

Razvan Pascanu, Tomáš Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 583–593, Stroudsburg, PA, USA. Association for Computational Linguistics.

Samira Shaikh, Tomek Strzalkowski, Sarah Taylor, and Nick Webb. 2010. Vca: An experiment with a multiparty virtual chat agent. In *Proceedings of the 2010 Workshop on Companionable Dialogue Systems*, CDS '10, pages 43–48, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1577–1586, Beijing, China, July. Association for Computational Linguistics.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *TACL*, 2:207–218.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205, Denver, Colorado, May–June. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.

A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. 1989. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, Mar.

Marilyn Walker, Rashmi Prasad, and A Stent. 2003. A trainable generator for recommendations in multimodal dialog. In *Proceedings of EUROSPEECH*, pages 1697–1701.

Joseph Weizenbaum. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, January.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 643–648.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *CoRR*, abs/1410.4615.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

PersoNER: Persian Named-Entity Recognition

Hanieh Poostchi
University of Technology Sydney
Capital Markets CRC
hpoostchi@cmcrc.com

Ehsan Zare Borzeshi
Capital Markets CRC
ezborzeshi@cmcrc.com

Mohammad Abdous
Iran University of Science and Technology
md.abdous@gmail.com

Massimo Piccardi
University of Technology Sydney
massimo.piccardi@uts.edu.au

Abstract

Named-Entity Recognition (NER) is still a challenging task for languages with low digital resources. The main difficulties arise from the scarcity of annotated corpora and the consequent problematic training of an effective NER pipeline. To abridge this gap, in this paper we target the Persian language that is spoken by a population of over a hundred million people world-wide. We first present and provide ArmanPerosNERCorpus, the first manually-annotated Persian NER corpus. Then, we introduce PersoNER, an NER pipeline for Persian that leverages a word embedding and a sequential max-margin classifier. The experimental results show that the proposed approach is capable of achieving interesting MUC7 and CoNNL scores while outperforming two alternatives based on a CRF and a recurrent neural network.

1 Introduction

Named-Entity Recognition (NER), introduced in the sixth Message Understanding Conference (MUC-6) (Grishman and Sundheim, 1996), concerns the recognition of Named Entities (NE) and numeric expressions in unstructured text. Since 1996, great effort has been devoted to NER as a foundational task for higher-level natural language processing tasks such as summarization, question answering and machine translation.

Shortage of gold standards has initially limited NER investigation to high-resource languages such as English, German and Spanish (Tjong Kim Sang and De Meulder, 2003). Gradually, publicly available encyclopediae have enabled combinations of semi-supervised and distant supervision approaches for other languages (Althobaiti et al., 2015). However, low-resource languages still face a significant scarcity of public repositories. For instance, only 8.8% of Wikipedia articles in Hindi are identified as entity-based articles in Freebase (Al-Rfou et al., 2015). In this work, we aim to enable supervised NER for a low-resource language, namely Persian, by providing the first manually-annotated Persian NE dataset. The Persian language, despite accounting for more than a hundred million speakers around the globe, has been rarely studied for NER (Khormuji and Bazrafkan, 2014) and even text processing (Shamsfard, 2011). In addition, we present PersoNER, a Persian NER pipeline consisting of a word embedding module and a sequential classifier based on the structural support vector machine (Tsochantaridis et al., 2005). The proposed pipeline achieves interesting MUC7 and CoNNL scores and outperforms two alternatives based on a CRF and a recurrent neural network.

2 Related Work

Early research on NER was mostly devoted to handcrafted rule-based systems which are intrinsically language-dependent, and thus laborious to be extended to new languages. As a consequence, recent studies are mainly focused on language-independent machine learning techniques that attempt to learn

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3381–3389, Osaka, Japan, December 11-17 2016.

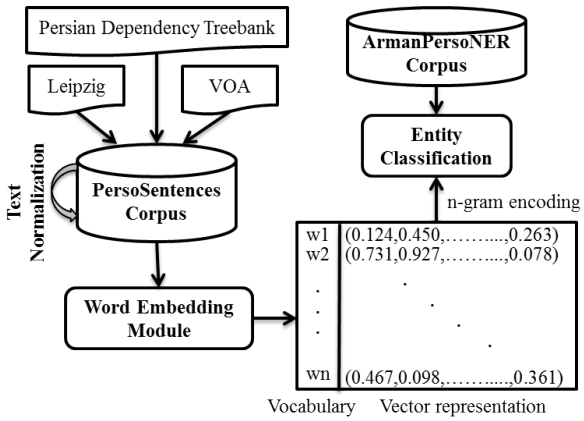


Figure 1: PersoNER workflow.

statistical models for NER from data (Nadeau and Sekine, 2007). Moreover, replacement of manually-annotated gold standards with very large "silver standard" corpora mollifies the scarcity of supervised data. Silver standards are NE annotated corpora derived from processing Wikipedia's text and meta-information alongside entity databases such as Freebase (Nothman et al., 2013; Al-Rfou et al., 2015).

Existing NER approaches mainly divide over two categories: in the first, the task is decoupled into an initial step of word embedding, where words are mapped to feature vectors, followed by a step of word/sentence-level classification. The feature vector can be as simple as a binary vector of text features like '*word is all uppercased*' or a more complex, real-valued vector capturing semantic and syntactic aspects of the word. Word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and Hellinger-PCA (Lebret and Collobert, 2014) are well-known examples of unsupervised word embeddings applied successfully to the NER task. For classification, sequential classifiers such as HMMs (Zhou and Su, 2002), CRFs (Lafferty et al., 2001; Finkel et al., 2005) and deep neural networks (Al-Rfou et al., 2015) have been amongst the most popular choices.

The second category, proposed by (Collobert et al., 2011) and recently followed by many including (Mesnil et al., 2013; Mesnil et al., 2015) and others, leverages recurrent neural networks (RNNs) to deliver end-to-end systems for NER. With this approach, an implicit word embedding is automatically extracted in the network's early layers by initializing the training with random values or a preliminary embedding. In this paper, we apply and compare approaches from both categories.

3 The Proposed Approach

The workflow of PersoNER is illustrated in Figure 1. The steps include data collection, text normalization, word embedding and entity classification. In this section, we focus on the two technical modules, word embedding and classification, while data collection and text normalization are described in Section 4.

3.1 Word Embedding

Term-frequency (tf), term-frequency inverse-document-frequency (tf-idf), bag of words (bow) and word co-occurrence are general statistics intended to characterize words in a collection of documents. Out of them, word co-occurrence statistics have the ability to represent a word by the frequencies of its surrounding words which well aligns with the requirements of NER. Recently, Lebret and Collobert (2014) have shown that a simple spectral method analogous to PCA can produce word embeddings as useful as those of neural learning algorithms such as word2vec. Given an unsupervised training corpus and a vocabulary, V, the co-occurrence matrix, $C_{|V|\times|D|}$, in (Lebret and Collobert, 2014) is computed as:

$$C(v_i, d_j) = p(d_j|v_i) = \frac{n(v_i, d_j)}{\sum_d n(v_i, d)} \quad (1)$$

where $v_i \in V; i = 1 \dots |V|$ and $d_j \in D \subseteq V; j = 1 \dots |D|$. $n(v_i, d_j)$ is the count of occurrences of context word d_j in the neighborhood of reference word v_i. Thus, $C(v_i, :)$ represents discrete probability distribution $p(d|v_i)$ and is used to characterize v_i. Since words are represented as discrete distributions, Lebret and Collobert (2014) argue that it is more appropriate to measure their distances in a Hellinger space. Accordingly, $H(C)$ is the transformation of C into Hellinger space where the distance between any two discrete probability distributions, P and Q, is given by:

$$dist(P, Q) = \frac{1}{\sqrt{2}} ||\sqrt{P} - \sqrt{Q}||_2. \tag{2}$$

Eventually, PCA is applied to reduce the dimensionality of $H(C) \in \mathbb{R}^{|V| \times |D|}$ to $h(C) \in \mathbb{R}^{|V| \times m}$, where $m \ll |D|$.

3.2 Classification

In this subsection, we first briefly introduce sequential labeling as a formal problem and then describe the sequential classifier based on the structural support vector machine.

3.2.1 Sequential Labeling

Sequential labeling predicts a sequence of class labels, $y = \{y_1, \dots y_t, \dots y_T\}$, based on a corresponding sequence of measurements, $x = \{x_1, \dots x_t, \dots x_T\}$. It is a very common task in NLP for applications such as chunking, POS tagging, slot-filling and NER. A widespread model for sequential labeling is the hidden Markov model (HMM) that factorizes the joint probability of the measurements and the labels, $p(x, y)$, by arranging the latter in a Markov chain (of order one or above) and conditioning the measurement at frame t on only the corresponding label. For an HMM of order one, $p(x, y)$ is expressed as:

$$p(x, y) = p(y_1) \prod_{t=2}^{T} p(y_t|y_{t-1}) \prod_{t=1}^{T} p(x_t|y_t) \tag{3}$$

where $p(y_1)$ is the probability of the initial class, terms $p(y_t|y_{t-1})$ are the transition probabilities and terms $p(x_t|y_t)$ are the emission, or measurement, probabilities. By restricting the emission probabilities to the exponential family, i.e., $p(x_t|y_t) \propto exp(w^T f(x_t, y_t))$, the logarithm of probability $p(x, y)$ can be expressed as the score of a *generalized linear model*:

$$\begin{aligned} &\ln p(x, y) \propto w^T \phi(x, y) = \\ &w_{in} f(y_1) + \sum_{t=2}^{T} w_{tr}^T f(y_t, y_{t-1}) + \sum_{t=1}^{T} w_{em}^T f(x_t, y_t) \end{aligned} \tag{4}$$

where w_{in}, w_{tr} and w_{em} are the linear models for assigning a score to the initial classes, transitions and emissions, respectively. Functions $f(y_1)$, $f(y_t, y_{t-1})$ and $f(x_t, y_t)$ are arbitrary, fixed "feature" functions of the measurements and the labels.

The generalized linear model in (4) is more suitable for discriminative training than the generative probabilistic model in (3). Notable discriminative approaches are conditional random fields (CRFs) (Lafferty et al., 2001) and structural SVM (Tsochantaridis et al., 2005). In particular, structural SVM has built a very strong reputation for experimental accuracy in NLP tasks (Joachims et al., 2009; Tang et al., 2013; Qu et al., 2014) and for this reason we exploit it in our NER pipeline.

Eventually, given a measurement sequence x in input, inference of the optimal label sequence can be obtained as:

$$\bar{y} = \underset{y}{\operatorname{argmax}}\, p(x, y) = \underset{y}{\operatorname{argmax}}(w^T \phi(x, y)) \tag{5}$$

This problem can be efficiently solved in $O(T)$ time by the Viterbi algorithm working in either the linear or logarithmic scale (Rabiner, 1989).

3.2.2 Structural SVM

From a supervised training set of sequences, $\{X, Y\} = \{x^i, y^i\}, i = 1 \dots N$, *structural SVM* finds the model's parameters, w, by minimizing the usual SVM trade-off between the hinge loss and an $L2$ regularizer (Tsochantaridis et al., 2005). Its learning objective can be expressed as:

$$\begin{aligned} &\underset{w,\xi}{\operatorname{argmin}} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi^i \quad s.t. \\ &w^T\phi(x^i, y^i) - w^T\phi(x^i, y) \geq \Delta(y^i, y) - \xi^i, \\ &\qquad i = 1 \dots N, \ \ \forall y \in \mathcal{Y} \end{aligned} \tag{6}$$

In the objective function, the first term is the regularizer while the second term, $\sum_{i=1}^{N}\xi^i$, is the hinge loss, i.e. a convex upper bound over the total loss on the training set. Hyperparameter C is an arbitrary, positive coefficient that balances these two terms. In the constraints, $w^T\phi(x, y)$ computes the generalized linear score for a (x, y) pair. In the case of sequential labeling, such a score is given by Eq. (4). Eventually, $\Delta(y^i, y)$ is the loss function chosen to assess the loss over the training set.

For an NER task with M entity classes, each sequence of length T adds $(M + 1)^T$ constraints to (6). Due to their exponential number, exhaustive satisfaction of all constraints is infeasible. However, (Tsochantaridis et al., 2005) has shown that it is possible to find ϵ-correct solutions with a subset of the constraints of polynomial size consisting of only the "most violated" constraint for each sequence, i.e. the labeling with the highest sum of score and loss:

$$\begin{aligned} &\xi^i = \max_y(-w^T\phi(x^i, y^i) + w^T\phi(x^i, y) + \Delta(y^i, y)) \\ &\rightarrow \bar{y}^i = \underset{y}{\operatorname{argmax}}(w^T\phi(x^i, y) + \Delta(y^i, y)) \end{aligned} \tag{7}$$

This problem is commonly referred to as "loss-augmented inference" given its resemblance with the common inference of Eq. (5) and is the core of structural SVM. In the case of scores and losses that can be computed frame by frame (such as the 0-1 loss or the Hamming loss), the Viterbi algorithm with appropriate weights can still be used to compute the loss-augmented inference in $O(T)$ time.

4 Data Collection

In this section, we describe the collection and preprocessing of the Persian corpora. The datasets consist of 1) an unsupervised corpus, called PersoSentencesCorpus, that we use for the word embedding module and 2) a manually named-entity annotated data set of Persian sentences, called ArmanPersoNERCorpus, that we use for supervised classification. Alongside this publication, we release ArmanPersoNERCorpus[1] as the first ever publicly-available Persian NER dataset.

4.1 PersoSentencesCorpus

A very large corpus of documents covering a variety of contexts is required to populate an effective co-occurrence matrix. We fulfill this requirement by accumulating the following three datasets of Persian sentences:

- The *Leipzig corpora* [2] with 1,000,000 sentences from news crawling and 300,000 from Wikipedia.
- The *VOA*[3] news dataset with 277,000 sentences.
- The *Persian Dependency Treebank*[4] with 29,982 sentences (Rasooli et al., 2013).

The aggregated corpus, called PersoSentencesCorpus, holds more than 1.6 million sentences and seems of adequate size to train the co-occurrence matrix.

[1] http://poostchi.info/hanieh/NLP/ArmanPersoNERCorpus.txt
[2] http://corpora2.informatik.uni-leipzig.de/download.html
[3] http://www.ling.ohio-state.edu/~jonsafari/corpora/index.html\#persian
[4] http://dadegan.ir/en/perdt/

Entity type	**Person**	**Organization**	**Location**	**Facility**	**Event**	**Product**	**Other**
Number of Tokens (NT)	5,215	10,036	4,308	1,485	2,518	1,463	224,990
Percentage	2.08%	4.01%	1.72%	0.59%	1.00%	0.58%	89.99%
Number of Unique-Tokens (NUT)	1,829	1,290	832	548	556	634	15,677
Percentage (NUT/NT)	35.07%	12.85%	19.31%	36.90%	22.08%	43.33%	6.96%

Table 1: Class percentages in ArmanPersoNERCorpus.

4.2 ArmanPersoNERCorpus

To create an NE dataset, in collaboration with ArmanSoft[5], we have decided to manually annotate NEs in a subset of the **BijanKhan**[6] (Bijankhan et al., 2011) corpus which is the most-established tagged Persian corpus, yet lacking entity annotation. We selected the subset from news sentences since they are the most entity-rich. Before the annotation, a comprehensive manual was designed based on the definition of Sekine's extended named entities (Sekine, 2007) adapted to the Persian Language. The annotation task was led by an experienced lead annotator who instructed the front-end annotators (two native post-graduate students) and revised their annotations. The guidelines were very clear and we expected minimal subjectivity. We have verified this hypothesis in two ways: by a sample of 500 already annotated NEs chosen randomly, and by another sample of 500 already annotated NEs from the two most semantically-close classes (location and organization). Both samples were revised by three other, independent native annotators and the percentages of corrections have been only 1.8% and 1.9%, respectively.

All NEs have been annotated in IOB format. The annotated dataset, **ArmanPersoNERCorpus**, contains 250,015 tokens and 7,682 sentences (considering the full-stop as the sentence terminator). It can be used to train NER systems in future research on Persian NER, but it also offers an ideal test set for evaluation of NER systems trained on silver standards. The NEs are categorized into six classes: *person*, *organization* (such as banks, ministries, embassies, teams, nationalities, networks and publishers), *location* (such as cities, villages, rivers, seas, golfs, deserts and mountains), *facility* (such as schools, universities, research centers, airports, railways, bridges, roads, harbors, stations, hospitals, parks, zoos and cinemas), *product* (such as books, newspapers, TV shows, movies, airplanes, ships, cars, theories, laws, agreements and religion), and *event* (such as wars, earthquakes, national holidays, festivals and conferences); *other* are the remaining tokens. It is worth noting that annotation was not trivial since individual tokens have been categorized according to the context. For instance, *"Tokyo"* is a different type of entity in sentence *"$Tokyo_{loc}$ is a beautiful city"* versus sentence *"$London_{org}$ and $Tokyo_{org}$ sign flight agreement"*. Table 1 summarizes the number of tokens for each entity class in ArmanPersoNERCorpus.

Figure 2 shows a snapshot of the dataset together with an English transliteration of the tokens. Each line contains five tab-separated columns. In order from left to right, they are ez$\bar{a}$fe, POS-tag, inflexion, token and NER-tag. The first three columns are inherited from the BijanKhan corpus. Ez$\bar{a}$fe [7] is a grammatical particle in the Persian language that connects words of a phrase, usually noun-phrase, together. It is pronounced as an unstressed i vowel between the linked words, but generally not indicated in writing.

4.3 Text Normalization

As the preprocessing phase, the PersoSentencesCorpus has been normalized and tokenized following the approach proposed in (Feely et al., 2014) that suggests applying a pipeline of useful tools to deal with written Persian. The pipeline starts with PrePer (Seraji, 2013) which maps Arabic specific characters to their Persian Unicode equivalent. In addition, it replaces the full space between a word and its affix with a zero-width-non-joiner character. Then, a Farsi text normalizer (Feely, 2013) omits Arabic and Persian diacritics and unifies variant forms of some Persian characters to a single Unicode representation. Finally,

[5] http://armansoft.ir

[6] http://ece.ut.ac.ir/dbrg/bijankhan/

[7] https://en.wikipedia.org/wiki/Ezafe

Ezāfe	POS-tag	Inflexion	Token	NER-tag	Transliteration
O	N	N,SING,SURN	سید	B-PERS	Seyed
EZ	N	N,SING,PR,GEN	محمود	I-PERS	Mahmoud
O	N	N,SING,PR	محدث	I-PERS	Mohadess
EZ	N	N,SING,COM,GEN	مدیر	O	manager
EZ	N	N,SING,COM,GEN	اکتشاف	O	discovery
EZ	N	N,SING,COM,GEN	شرکت	B-ORG	Company
EZ	AJ	ADJ,SIM,GEN	ملی	I-ORG	National
EZ	N	N,SING,COM,GEN	نفت	I-ORG	Oil
O	N	N,SING,LOC,PR	ایران	I-ORG	Iranian
O	P	P	در	O	in
O	N	N,SING,COM	مصاحبه	O	interview
O	P	P	با	O	with
EZ	AJ	ADJ,SIM,GEN	واحد	B-ORG	Unit
EZ	AJ	ADJ,SIM,GEN	مرکزی	I-ORG	Central
O	N	N,SING,COM	خبر	I-ORG	News
O	P	P	با	O	with
EZ	N	N,SING,COM,GEN	اعلام	O	declaring
O	DET	DET	این	O	this
O	N	N,SING,COM	خبر	O	announcement
O	V	V,PA,SIM,POS,3	افزود	O	adds
O	PUNC	DELM	:	O	:
O	P	P	با	O	With
EZ	N	N,PL,COM,GEN	حفاریهای	O	diggings
O	AJ	ADJ,CMPR	بیشتر	O	more
O	P	P	در	O	in
EZ	N	N,SING,LOC,GEN	میدان	B-LOC	field
EZ	AJ	ADJ,SIM,GEN	نفتی	I-LOC	oil
O	N	N,SING,LOC,PR	چنگوله	I-LOC	Changuleh
O	N	N,SING,COM	انتظار	O	expectation
O	V	V,PRS,POS,4	داریم	O	have
EZ	N	N,PL,COM,GEN	ذخائر	O	reservoirs
O	DET	DET	این	O	In
O	N	N,SING,LOC	میدان	O	field
O	N	N,SING,COM	افزایش	O	increase
O	V	V,SUB,POS,3	یابد	O	will
O	PUNC	DELM	.	O	.

Figure 2: A snapshot of ArmanPersoNERCorpus.

	Entities													
	Person		**Organization**		**Location**		**Facility**		**Event**		**Product**		**Overall**	
Methods	MUC7	CoNLL	MUC7	CoNLL	MUC7	CoNLL	MUC7	CoNLL	MUC7	CoNLL	MUC7	CoNLL	MUC7	CoNLL
CRF	76.98	64.10	60.59	42.25	66.98	57.97	61.46	41.09	59.98	22.48	33.75	20.00	60.89	49.92
Jordan-RNN	79.13	72.13	67.31	57.28	69.90	62.70	63.49	51.92	62.30	39.79	49.50	**42.08**	68.53	60.52
SVM-HMM	**82.40**	**75.65**	**71.65**	**61.59**	**72.92**	**66.67**	**72.22**	**61.20**	**71.63**	**52.58**	**50.90**	41.37	**72.59**	**65.13**

Table 2: F_1 score comparison between three different classifiers based on MUC7 and CoNLL score functions for NER task on ArmanPersoNERCorpus. The F_1 score achieved by structural SVM is higher overall and for all classes but one, with the Jordan-RNN as the second best.

tokenization is performed by using three tokenizers in a cascade: the Farsi verb tokenizer of (Manshadi, 2013), SetPer (Seraji et al., 2012) and tok-tok (Dehdari, 2015).

5 Experiments

In this section, we report NER results based on the PersoSentencesCorpus and ArmanPersoNERCorpus datasets. The classification task is challenging given the much lower frequencies of the entity classes versus the non-entity class (*other*), as shown in Table 1. For this task, we have not used any of the additional linguistic information that is available from the dataset (such as POS tag, inflexion etc).

To calculate the co-occurrence matrix, C, we have used a context window of radius 5. The size of the dictionary, V, from the PersoSentencesCorpus is $|V| = 49,902$ and that of subset D is $D = 7,099$, obtained by selecting only the words with count greater than 15. The word embedding matrix $h(C)$ has been computed by heuristically setting $m = 300$. For classification, each word has been encoded as a 3-gram that includes the previous and following feature vectors. All the models used for classification share the same word embeddings.

For classification, we have compared the proposed SVM-HMM with a CRF and a deep learning approach based on the Jordan-RNN (Mesnil et al., 2013). For the SVM-HMM we have used structural SVM from (Joachims, 2008) with a Markov chain of order 3 and learning constant $C = 0.5$. The CRF is from the HCRF library (Morency et al., 2010) and is trained with an $L2$ regularizer of weight 100. The Jordan-RNN is a recurrent neural network from (Mesnil et al., 2013) trained with 100 hidden states and initialized using the same features vectors. All parameters were chosen by 3-fold cross-validation over a reasonable range of values. The indices for the three folds are available in the dataset to allow for future result comparison. We have also tried continuous bag of words (Mikolov et al., 2013), skip-grams (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) as embeddings, and the Elman-RNN (Mesnil et al., 2013) as classifier, but results have proved generally less accurate.

Table 2 shows the comparison of the average MUC7 and CoNLL scores from the 3-fold cross-validation for the three classifiers. The MUC7 and CoNLL scores are F_1 values adapted to the NER task, with the CoNLL score generally stricter than MUC7 (Nadeau and Sekine, 2007). As shown in Table 2, the scores achieved by the SVM-HMM are higher overall and for all classes but one, with the Jordan-RNN as the second best. To verify statistical significance, we have also run a paired t-test over the results from the six individual classes and confirmed statistical significance of the differences even at $p = 0.02$. The relative ranking between SVM-HMM and the CRF is supported by similar results in the literature, including (Nguyen and Guo, 2007; Tang et al., 2013; Lei et al., 2014), showing that regularized minimum-risk classifiers tend to outperform equivalent models trained under maximum conditional likelihood. The relative ranking between SVM-HMM and the RNN is instead somehow in contrast with the recent results in the literature, and a possible explanation for it is the relatively small size of the dataset compared to the number of free parameters in the models. We plan future comparative experiments with larger corpora to further probe this assumption.

6 Conclusion

In this paper, we have presented and released ArmanPersoNERCorpus, the first manually-annotated Persian NE dataset, and proposed an NER pipeline for the Persian language. The main components

of the pipeline are word embedding by Hellinger PCA and classification by a structural SVM-HMM classifier. Experiments conducted over the ArmanPersoNERCorpus dataset have achieved interesting overall F_1 scores of 72.59 (MUC7) and 65.13 (CoNNL), higher than those of a CRF and a Jordan-RNN. The released dataset can be used for further development of Persian NER systems and for evaluation of systems trained on silver-standard corpora, and the achieved accuracy will provide a baseline for future comparisons.

References

Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. 2015. Polyglot-ner: Massive multilingual named entity recognition. In *Proceedings of 2015 SIAM International Conference on Data Mining*.

Maha Althobaiti, Udo Kruschwitz, and Massimo Poesio. 2015. Combining minimally-supervised methods for arabic named entity recognition. *TACL*, 3:243–255.

Mahmood Bijankhan, Javad Sheykhzadegan, Mohammad Bahrani, and Masood Ghayoomi. 2011. Lessons from building a persian written corpus: Peykare. *Language resources and evaluation*, 45(2):143–164.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

Jon Dehdari. 2015. A fast, simple, multilingual tokenizer. `https://github.com/jonsafari/tok-tok/`.

Weston Feely, Mehdi Manshadi, Robert E Frederking, and Lori S Levin. 2014. The cmu metal farsi nlp approach. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 4052–4055, Reykjavik, Iceland.

Weston Feely. 2013. Open-source dependency parser, part-of-speech-tagger, and text normalizer for farsi (persian). `https://github.com/wfeely/farsiNLPTools/`.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics*, pages 363–370.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 466–471, Kopenhagen.

Thorsten Joachims, Thomas Hofmann, Yisong Yue, and Chun-Nam Yu. 2009. Predicting structured objects with support vector machines. *Communications of the ACM*, 52(11):97–104.

Thorsten Joachims. 2008. SVM$^{\text{hmm}}$: Sequence tagging with structural support vector machines. `https://www.cs.cornell.edu/people/tj/svm\_light/svm\_hmm.html/`.

Morteza Kolali Khormuji and Mehrnoosh Bazrafkan. 2014. Persian named entity recognition based with local filters. *International Journal of Computer Applications*, 100(4):1–6.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Remi Lebret and Ronan Collobert. 2014. Word embedding through hellinger pca. In *14th Conference of the European Chapter of the Association for Computational Linguistics*, Gothenburg, Sweden.

Jianbo Lei, Buzhou Tang, Xueqin Lu, Kaihua Gao, Min Jiang, and Hua Xu. 2014. A comprehensive study of named entity recognition in chinese clinical text. *Journal of the American Medical Informatics Association*, 21:808–814.

Mehdi Manshadi. 2013. Farsi verb tokenizer. `https://github.com/mehdi-manshadi/Farsi-Verb-Tokenizer/`.

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

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, and Geoffrey Zweig. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, March.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Louis-Philippe Morency, C. Mario Christoudias, Ariadna Quattoni, Hugues Salamin, Giota Stratou, and Sybor Wang. 2010. Hidden-state conditional random field library. `http://multicomp.ict.usc.edu/?p=790`.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January. Publisher: John Benjamins Publishing Company.

Nam Nguyen and Yunsong Guo. 2007. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 681–688, New York, NY, USA. ACM.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artif. Intell.*, 194:151–175, January.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. `http://nlp.stanford.edu/projects/glove/`.

Lizhen Qu, Yi Zhang, Rui Wang, Lili Jiang, Rainer Gemulla, and Gerhard Weikum. 2014. Senti-lssvm: Sentiment-oriented multi-relation extraction with latent structural SVM. *TACL*, 2:155–168.

L.R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *IEEE Proc.*, 77:257–286.

Mohammad Sadegh Rasooli, Manouchehr Kouhestani, and Amirsaeid Moloodi. 2013. Development of a persian syntactic dependency treebank. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 306–314, Atlanta, Georgia, June. Association for Computational Linguistics.

Satoshi Sekine. 2007. The definition of sekines extended named entities. `http://nlp.cs.nyu.edu/ene/version7_1_0Beng.html`. New York University.

Mojgan Seraji, Megyesi Beta, and Nivre Joakim. 2012. A basic language resource kit for persian. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 2245–2252, Istanbul, Turkey.

Mojgan Seraji. 2013. Preper: A pre-processor for persian. In *Proceedings of Fifth International Conference on Iranian Linguistics*, Bamberg, Germany.

Mehrnoush Shamsfard. 2011. Challenges and open problems in persian text processing. *Proceedings of LTC*, 11.

Buzhou Tang, Hongxin Cao, Yonghui Wu, Min Jiang, and Hua Xu. 2013. Recognizing clinical entities in hospital discharge summaries using structural support vector machines with word representation features. *BMC Medical Informatics and Decision Making*, 13(SUPPL1).

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 473–480, Stroudsburg, PA, USA. Association for Computational Linguistics.

OCR++: A Robust Framework For Information Extraction from Scholarly Articles

Mayank Singh, Barnopriyo Barua, Priyank Palod, *Manvi Garg,
Sidhartha Satapathy, Samuel Bushi, Kumar Ayush, Krishna Sai Rohith, Tulasi Gamidi,
Pawan Goyal and Animesh Mukherjee
Department of Computer Science and Engineering
*Department of Geology and Geophysics
Indian Institute of Technology, Kharagpur, WB, India
mayank.singh@cse.iitkgp.ernet.in

Abstract

This paper proposes *OCR++*, an open-source framework designed for a variety of information extraction tasks from scholarly articles including metadata (title, author names, affiliation and e-mail), structure (section headings and body text, table and figure headings, URLs and footnotes) and bibliography (citation instances and references). We analyze a diverse set of scientific articles written in English to understand generic writing patterns and formulate rules to develop this hybrid framework. Extensive evaluations show that the proposed framework outperforms the existing state-of-the-art tools by a large margin in structural information extraction along with improved performance in metadata and bibliography extraction tasks, both in terms of accuracy (around 50% improvement) and processing time (around 52% improvement). A user experience study conducted with the help of 30 researchers reveals that the researchers found this system to be very helpful. As an additional objective, we discuss two novel use cases including automatically extracting links to public datasets from the proceedings, which would further accelerate the advancement in digital libraries. The result of the framework can be exported as a whole into structured TEI-encoded documents. Our framework is accessible online at *http://www.cnergres.iitkgp.ac.in/OCR++/home/*.

1 Introduction

Obtaining structured data from documents is necessary to support retrieval tasks (Beel et al., 2011). Various scholarly organizations and companies deploy information extraction tools in their production environments. Google scholar[1], Microsoft academic search[2], Researchgate[3], CiteULike[4] etc. provide academic search engine facilities. European publication server (EPO)[5], ResearchGate and Mendeley[6] use *GROBID* (Lopez, 2009) for header extraction and analysis. A similar utility named *SVMHeaderParse* is deployed by $CiteSeerX$[7] for header extraction.

Through a comprehensive literature survey, we find comparatively less research in document structure analysis than metadata and bibliography extraction from scientific documents. The main challenges lie in the inherent errors in OCR processing and diverse formatting styles adopted by different publishing venues. We believe that a key strategy to tackle this problem is to analyze research articles from different publishers to identify generic patterns and rules, specific to various information extraction tasks. We introduce *OCR++*, a hybrid framework to extract textual information such as (i) metadata – title, author names, affiliation and e-mail, (ii) structure – section headings and body text, table and figure headings, URLs and footnotes, and (iii) bibliography – citation instances and references from scholarly articles.

[1] http://scholar.google.com
[2] http://academic.research.microsoft.com
[3] https://www.researchgate.net
[4] http://www.citeulike.org/
[5] https://data.epo.org/publication-server
[6] https://www.mendeley.com
[7] http://citeseerx.ist.psu.edu/

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3390–3400, Osaka, Japan, December 11-17 2016.

The framework employs a variety of Conditional Random Field (CRF) models and hand-written rules specially crafted for handling various tasks. Our framework produces comparative results in metadata extraction tasks. However, it significantly outperforms state-of-the-art systems in structural information extraction tasks. On an average, we record an accuracy improvement of 50% and a processing time improvement of 52%. We claim that our hybrid approach leads to higher performance than complex machine learning models based systems. We also present two novel use cases including extraction of public dataset links available in the proceedings of the NLP conferences available from the ACL anthology.

2 Framework overview

OCR++ is an extraction framework for scholarly articles, completely written in Python (Figure 2). The framework takes a PDF article as input, 1) converts the PDF file to an XML format, 2) processes the XML file to extract useful information, and 3) exports output in structured TEI-encoded[8] documents. We use open source tool *pdf2xml*[9] to convert PDF files into rich XML files. Each token in the PDF file is annotated with rich metadata, namely, x and y co-ordinates, font size, font weight, font style etc. (Figure 1).

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENT><METADATA><PDFFILENAME>/var/www/html/OCR++/myproject/media/documents/input.pdf</PDFFILENAME><PROCESS name="pdftoxml" cmd="
-noImage -noImageInline "><VERSION value="1.2"><COMMENT/></VERSION><CREATIONDATE>Fri Oct  7 07:11:18 2016
</CREATIONDATE>
</PROCESS>
    </METADATA>
    <PAGE width="595.276" height="841.89" number="1" id="p1">
        <MEDIABOX x1="0" y1="0" x2="595.276" y2="841.89"/>
        <CROPBOX x1="0" y1="0" x2="595.276" y2="841.89"/>
        <BLEEDBOX x1="0" y1="0" x2="595.276" y2="841.89"/>
        <ARTBOX x1="0" y1="0" x2="595.276" y2="841.89"/>
        <TRIMBOX x1="0" y1="0" x2="595.276" y2="841.89"/>
        <TEXT width="397.117" height="12.8972" id="p1_t1" x="100.215" y="65.4231">
            <TOKEN sid="p1_s3" id="p1_w1" font-name="NimbusRomNo9L" symbolic="yes" fixed-width="yes" bold="no" italic="no" font-size="
            14.3462" font-color="#000000" rotation="0" angle="0" x="100.215" y="65.4231" base="75.322" width="53.0092" height="12.8972">
            OCR++:</TOKEN>
            <TOKEN sid="p1_s4" id="p1_w2" font-name="nimbusromno9l" symbolic="yes" fixed-width="yes" bold="no" italic="no" font-size="
            14.3462" font-color="#000000" rotation="0" angle="0" x="157.672" y="65.4231" base="75.322" width="10.358" height="12.8972">A</
            TOKEN>
            <TOKEN sid="p1_s5" id="p1_w3" font-name="nimbusromno9l" symbolic="yes" fixed-width="yes" bold="no" italic="no" font-size="
            14.3462" font-color="#000000" rotation="0" angle="0" x="171.616" y="65.4231" base="75.322" width="43.5551" height="12.8972">
            Robust</TOKEN>
            <TOKEN sid="p1_s6" id="p1_w4" font-name="nimbusromno9l" symbolic="yes" fixed-width="yes" bold="no" italic="no" font-size="
            14.3462" font-color="#000000" rotation="0" angle="0" x="218.758" y="65.4231" base="75.322" width="72.3622" height="12.8972">
            Framework</TOKEN>
            <TOKEN sid="p1_s7" id="p1_w5" font-name="nimbusromno9l" symbolic="yes" fixed-width="yes" bold="no" italic="no" font-size="
            14.3462" font-color="#000000" rotation="0" angle="0" x="294.706" y="65.4231" base="75.322" width="21.9497" height="12.8972">
            For</TOKEN>
            <TOKEN sid="p1_s8" id="p1_w6" font-name="nimbusromno9l" symbolic="yes" fixed-width="yes" bold="no" italic="no" font-size="
            14.3462" font-color="#000000" rotation="0" angle="0" x="320.243" y="65.4231" base="75.322" width="74.5572" height="12.8972">
            Information</TOKEN>
            <TOKEN sid="p1_s9" id="p1_w7" font-name="nimbusromno9l" symbolic="yes" fixed-width="yes" bold="no" italic="no" font-size="
            14.3462" font-color="#000000" rotation="0" angle="0" x="398.386" y="65.4231" base="75.322" width="65.3469" height="12.8972">
            Extraction</TOKEN>
            <TOKEN sid="p1_s10" id="p1_w8" font-name="nimbusromno9l" symbolic="yes" fixed-width="yes" bold="no" italic="no" font-size="
            14.3462" font-color="#000000" rotation="0" angle="0" x="467.32" y="65.4231" base="75.322" width="30.0123" height="12.8972">
            from</TOKEN>
            </TEXT><TEXT width="109.576" height="12.8972" id="p1_t2" x="243.984" y="81.3631"><TOKEN sid="p1_s11" id="p1_w9" font-name="
            nimbusromno9l" symbolic="yes" fixed-width="yes" bold="no" italic="no" font-size="14.3462" font-color="#000000" rotation="0"
            angle="0" x="243.984" y="81.3631" base="91.262" width="58.1882" height="12.8972">Scholarly</TOKEN>
            <TOKEN sid="p1_s12" id="p1_w10" font-name="nimbusromno9l" symbolic="yes" fixed-width="yes" bold="no" italic="no" font-size="
            14.3462" font-color="#000000" rotation="0" angle="0" x="305.759" y="81.3631" base="91.262" width="47.8015" height="12.8972">
            Articles</TOKEN>
```


Figure 1: Screenshot of *pdf2xml* tool output for the current paper.

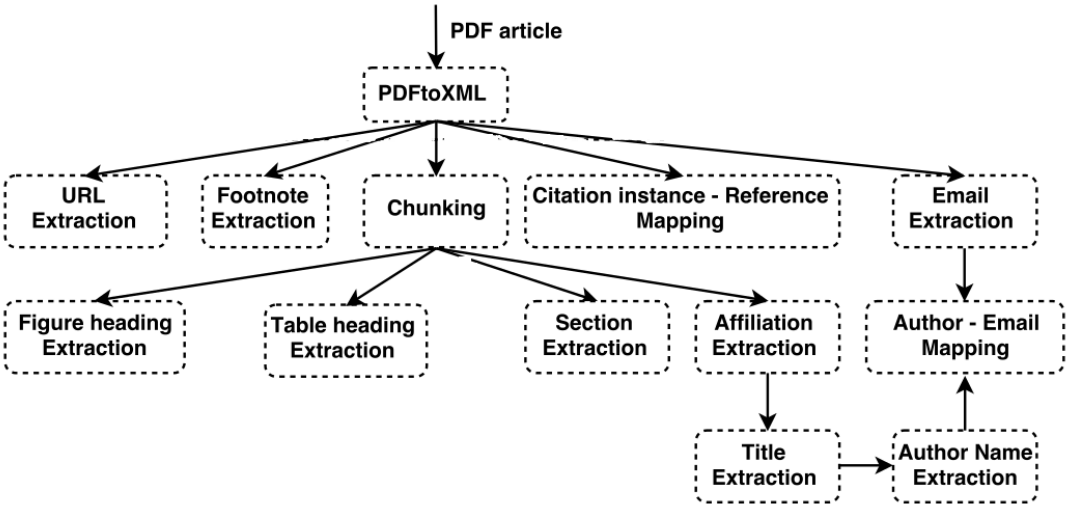
Figure 2(a) describes the sub-task dependencies in *OCR++*. The web interface of the tool is shown in Figure 2(b). We leverage the rich information present in the XML files to perform extraction tasks. Although each extraction task described below is performed using machine learning models as well as hand written rules/heuristics, we only include the better performing scheme in our framework. Next, we describe each extraction task in detail.

2.1 Chunking

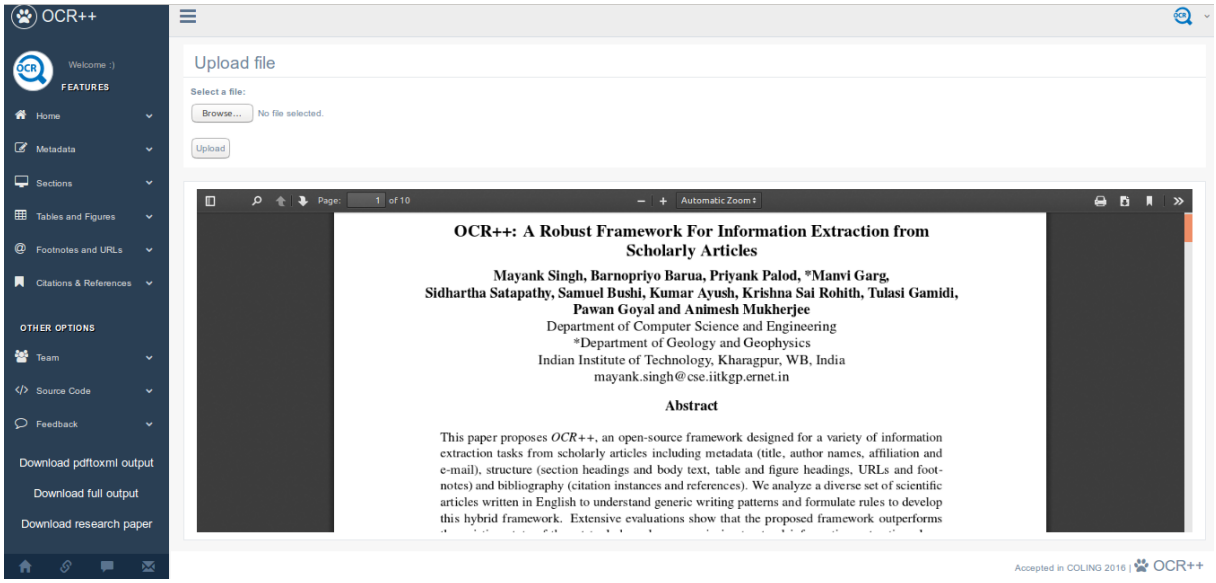
As a first step, we segment XML text into *chunks* by measuring distance from neighboring text and differentiating from the surrounding text properties such as font-size and bold-text.

[8] http://www.tei-c.org/index.xml

[9] URL: http://sourceforge.net/projects/pdf2xml/. We employ version 1.2.7 developed for 64 bit Linux systems.



(a) Sub-tasks dependencies in *OCR++*.



(b) Screenshot of OCR++ web interface.

Figure 2: OCR++ framework overview and user interface

2.2 Title extraction

We train a CRF model to label the token sequences using 6300 training instances. Features are constructed based on generic characteristics of formatting styles. Token level feature set includes boldness, relative position in the paper, relative position in the first chunk, relative size, case of first character, boldness + relative font size overall, case of first character in present and next token and case of first character in present and previous token.

2.3 Author name extraction

In this sub-task, we use the same set of features as described in the title extraction sub-task to train the CRF model along with a heuristic that the tokens eligible for the author names are either present in the first section or within 120 tokens after the title. Different author names are distinguished using heuristics, such as, difference in y-coordinates, tab separation etc. Further, false positives are removed using heuristics such as length of consecutive author name tokens, symbol or digit in token and POS tag. The first word among consecutive tokens is considered as the first name, the last word as the last name, and all the remaining words are treated together as the middle name.

2.4 Author e-mail extraction

An e-mail consists of a user name, a sub-domain name and a domain name. In case of scholarly articles, usually, the user names are written inside brackets separated by commas and the bracket is succeeded by the sub-domain and domain name. On manual analysis of a set of scholarly articles, we find four different writing patterns, author1@cse.domain.com, {author1, author2, author3}@cse.domain.com, [author4, author5, author6]@cse.domain.com and [author7@cse, author7@ee].domain.com. Based on these observations, we construct hand written rules to extract e-mails.

2.5 Author affiliation extraction

We use hand written rules to extract affiliations. We employ heuristics such as presence of country name, tokens like "University", "Research", "Laboratories", "Corporation", "College", "Institute", superscript character etc.

2.6 Headings and section mapping

We employ CRF model to label section headings. Differentiating features (the first token of the chunk, the second token, avg. boldness of the chunk, avg. font-size, Arabic/Roman/alpha-enumeration etc.) are extracted from chunks to train the CRF.

2.7 URL

We extract URLs using a single regular expression described below:

```
http[s]?://(?:[a-zA-Z]|[0-9]|[\$-_@.&+]|[!*\(\),]|(?:\%[0-9a-fA-F][0-9a-fA-F]))
```

2.8 Footnote

Most of the footnotes have numbers or special symbols (like asterisk etc.) at the beginning in the form of a superscript. Footnotes have font-size smaller than the surrounding text and are found at the bottom of a page – average font size of tokens in a chunk and y-coordinate were used as features for training the CRF. Moreover, footnotes are found in the lower half of the page (this heuristic helped in filtering false positives).

2.9 Figure and table headings

Figure and table heading extraction is performed after chunking (described in Section 2.1). If the chunk starts with the word "FIGURE" or "Figure" or "FIG." or "Fig.", then the chunk represents a figure heading. Similarly, if the chunk starts with the word "Table" or "TABLE", then the chunk represents a table heading. However, it has been observed that table contents are also present in the chunk. Therefore, we use a feature "bold font" to extract bold tokens from such chunks.

2.10 Citations and references

The bibliography extraction task includes extraction of citation instances and references. All the tokens succeeding the reference section are considered to be part of references and further each reference is extracted separately. Again, we employ hand written rules to distinguish between two consecutive references. On manual analysis, we found 16 unique citation instance writing styles (Table 1). We code these styles into regular expressions to extract citation instances.

2.11 Mapping tasks

Connecting author name to e-mail: In general, each author name present in the author section associates with some e-mail. *OCR++* tries to recover this association using simple rules, for example, sub-string match between username and author names, abbreviated full name as username, order of occurrence of e-mails etc.

Citation reference mapping: Each extracted citation instance is mapped to its respective reference. Since, there are two different styles of writing citation instances, *Indexed* and *Non-indexed*, we define mapping tasks for each style separately. Indexed citations are mapped directly to references with the

Table 1: Generic set of regular expressions for citation instance identification. Here, AN represents author name, Y represents year and I represent reference index within citation instance.

Citation Format	Regular Expression
<AN> et al. [<I>]	`([A-Z][a-zA-Z]* et al[.][\string\d\{1,3\}])`
<AN> [<I>]	`([A-Z][a-zA-Z]* [\string\d\{2\}])`
<AN> et al.<spaces> [<I>]	`([A-Z][a-zA-Z]* et al[.][ ]*[\string\d\{1\}])`
<AN> et al., <Y><I>	`([A-Z][a-zA-Z]* et al[.],\string\d\{4\}[a-z])`
<AN> et al., <Y>	`([A-Z][a-zA-Z]* et al[.][,] \string\d\{4\})`
<AN> et al., (<Y>)	`([A-Z][a-zA-Z]* et al[.][,] (\string\d\{4\}))`
<AN> et al. <Y>	`([A-Z][a-zA-Z]* et al[.] \string\d\{4\})`
<AN> et al. (<Y>)	`([A-Z][a-zA-Z]* et al[.] (\string\d\{4\}))`
<AN> and <AN> (<Y>)	`([A-Z][a-zA-Z]* and [A-Z][a-zA-Z]*(\string\d\&{4\}))`
<AN> & <AN> (<Y>)	`([A-Z][a-zA-Z]* & [A-Z][a-zA-Z]*(\string\d\&{4\}))`
<AN> and <AN>, <Y>	`([A-Z][a-zA-Z]* and [A-Z][a-zA-Z]*[,] \d\{4\})`
<AN> & <AN>, <Y>	`([A-Z][a-zA-Z]* & [A-Z][a-zA-Z]*[,] \d\{4\})`
<AN>, <Y>	`([A-Z][a-zA-Z]*[,] \string\d\{4\})`
<AN> <Y>	`([A-Z][a-zA-Z]* \string\d\{4\})`
<AN>, (<Y><I>)	`([A-Z][a-zA-Z]*(\string\d\{4\}[a-z]*))`
< multiple indices separated by commas >	`.*?[(.*?)]`

index inside enclosed brackets. The extracted index is mapped with the corresponding reference. Non-indexed citations are represented using the combination of year of publication and author's last name.

3 Results and discussion

Following an evaluation carried out by Lipinski et al. (2013), GROBID provided the best results over seven existing systems, with several metadata recognized with over 90% precision and recall. Therefore, we compare *OCR++* with the state-of-the-art *GROBID*. We compare results for each of the sub-tasks for both the systems against the ground-truth dataset. The ground-truth dataset is prepared by manual annotation of title, author names, affiliations, URLs, sections, subsections, section headings, table headings, figure headings and references for 138 articles from different publishers. The publisher names are present in Table 3. We divide the article set into training and test datasets in the ratio of 20:80. Note that each of the extraction modules described in the previous section also has separate training sample count, for instance, 6300 samples have been used to train the title extraction. Also, we observe that both the systems provide partial results in some cases. For example, in some cases, only half of the title is extracted or the author names are incomplete. In order to accommodate partial results from extraction tasks, we provide evaluation results at token level, i.e, what fraction of the tokens are correctly retrieved.

Table 2: Micro-average F-score for GROBID and OCR++ for different extractive subtasks.

	GROBID			OCR++		
Subtask	Precision	Recall	F-Score	Precision	Recall	F-Score
Title	0.93	**0.94**	**0.93**	**0.96**	0.85	0.90
Author First Name	0.81	**0.81**	**0.81**	**0.91**	0.65	0.76
Author Middle Name	N/A	N/A	N/A	**1.0**	**0.38**	**0.55**
Author Last Name	0.83	**0.82**	**0.83**	**0.91**	0.65	0.76
Email	0.80	0.20	0.33	**0.90**	**0.93**	**0.91**
Affiliation	0.74	0.60	0.66	**0.80**	**0.76**	**0.78**
Section Headings	0.70	**0.87**	**0.78**	**0.80**	0.72	0.76
Figure headings	0.59	0.42	0.49	**0.96**	**0.75**	**0.84**
Table headings	0.77	0.17	0.28	**0.87**	**0.74**	**0.80**
URLs	N/A	N/A	N/A	**1.0**	**0.94**	**0.97**
Footnotes	0.80	0.42	0.55	**0.91**	**0.63**	**0.74**
Author-Email	0.38	0.24	0.29	**0.93**	**0.44**	**0.60**

Table 2 presents comparative results for GROBID and *OCR++*. It shows that in terms of precision,

OCR++ outperforms *GROBID* in all the sub-tasks. Recall is higher for *GROBID* for some of the metadata extraction tasks. In *OCR++*, since title extraction depends on the first extracted chunk from section extraction, the errors in chunk extraction lead to low recall in title extraction. Similar problem results in lower recall in author name extraction. Due to the presence of variety of white space length between author first, middle and last name in various formats, we observe low recall overall in author name extraction subtasks. We also found that in many cases author-emails are quite different from author names resulting in lower recall for author-email extraction subtask. *OCR++* outperforms *GROBID* in majority of the structural information extraction subtasks in terms of both precision and recall. We observe that *GROBID* performs poorly for table heading extraction due to intermingling of table text with heading tokens and unnumbered footnotes. A similar argument holds for the figure heading as well. URL extraction feature is not implemented in *GROBID*, while *OCR++* extracts it very accurately. Similarly, poor extraction of non-indexed footnotes resulted in lower recall for footnote extraction subtask.

Table 3: Micro-average F-score for GROBID and OCR++ for different publishing styles.

		GROBID			OCR++		
Publisher	paper count	Precision	Recall	F-Score	Precision	Recall	F-Score
IEEE	30	0.82	0.61	0.70	**0.9**	**0.69**	**0.78**
ARXIV	25	0.75	0.63	0.68	**0.91**	**0.73**	**0.81**
ACM	35	0.69	0.49	0.58	**0.89**	**0.71**	**0.79**
ACL	16	0.89	0.59	0.71	**0.91**	**0.79**	**0.85**
SPRINGER	17	0.78	0.6	0.68	**0.85**	**0.63**	**0.72**
CHI	3	0.13	0.20	0.16	**0.5**	**0.36**	**0.42**
ELSEVIER	6	0.58	0.6	0.59	**0.82**	**0.74**	**0.78**
NIPS	3	0.82	0.68	0.74	**0.83**	**0.72**	**0.77**
ICML	1	**0.6**	**0.6**	**0.6**	0.59	0.54	0.56
ICLR	1	0.49	**0.55**	0.52	**0.67**	0.52	**0.59**
JMLR	1	0.58	0.55	0.56	**0.86**	**0.83**	**0.85**

Similarly, Table 3 compares *GROBID* and *OCR++* for different publishing formats. Here the results seem to be quite impressive with *OCR++* outperforming *GROBID* in almost all cases. This demonstrates the effectiveness and robustness of using generic patterns and rules used in building *OCR++*. As our system is more biased towards single (ACL, ARXIV, ICLR, etc.) and double column formats (ACM, ELSEVIER etc.), we observe lower performance on three column formats (CHI). Similarly, non-indexed sections format (SPRINGER, ELSEVIER, etc.) show less performance than indexed sections format (IEEE, ACM, etc.).

Since citation instance annotation demands a significant amount of human labour, we randomly select eight PDF articles from eight different publishers from ground-truth dataset PDFs. Manual annotation produces 328 citation instances. We also annotate references to produce 187 references in total. Table 4 shows performance comparison for bibliography related tasks. As depicted from Table 4, *OCR++* performs better for both citation and reference extraction tasks. GROBID does not provide Citation-Reference mapping, which is an additional feature of *OCR++*.

Table 4: Micro-average accuracy for GROBID and OCR++ bibliography extraction tasks.

	GROBID			OCR++		
	Precision	Recall	F-Score	Precision	Recall	F-Score
Citation	0.93	0.81	0.87	**0.94**	**0.97**	**0.95**
Reference	0.94	0.94	0.94	**0.98**	**0.99**	**0.98**
Citation-Reference	N/A	N/A	N/A	**0.94**	**0.97**	**0.95**

Next, we investigate whether better formatting styles over the years lead to higher precision by the proposed tool. Also, we compare *OCR++* with *GROBID* in terms of processing time.

3.1 Effect of formatting style on precision

We select International Conference on Computational Linguistics (COLING) as a representative example to understand the effect of evolution in formatting styles over the years on the accuracy of the extraction task. We select ten random articles each from six different years of publications. *OCR++* is used to extract the title for each year. Figure 3 presents title extraction accuracy for each year, reaffirming the fact that the recent year publications produce higher extraction accuracy due to better formatting styles and advancement in converters from Word/LaTeX to PDF. We also notice that before the year 2000, ACL anthology assumes that PDFs do not have embedded text, resulting into a lower recall before 2000.

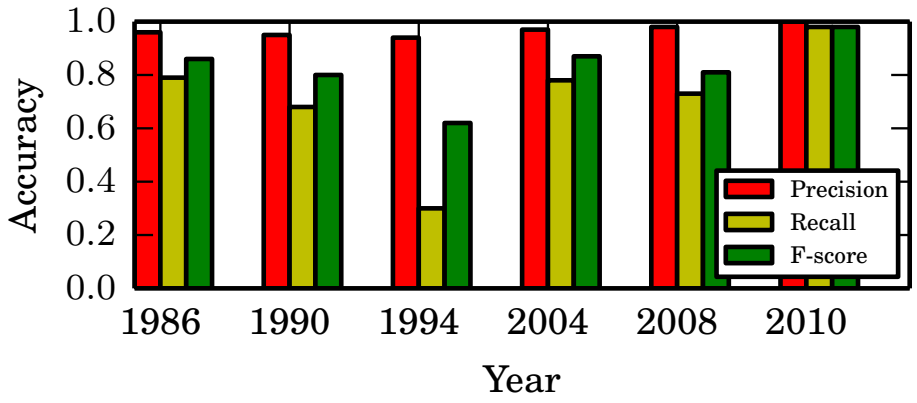


Figure 3: Title extraction accuracy calculated at six different years for COLING.

3.2 Processing time

To compare the processing times, we conducted experiments on a set of 1000 PDFs. The evaluation was performed on a single 64-bit machine, eight core, 2003.0 MHz processor and CentOS 6.5 version. Figure 4 demonstrates comparison between processing time of *GROBID* and *OCR++*, while processing some PDF articles in batch mode. There is significant difference in the execution time of *GROBID* and *OCR++*, with *OCR++* being much faster than *GROBID* for processing a batch of 100 articles.

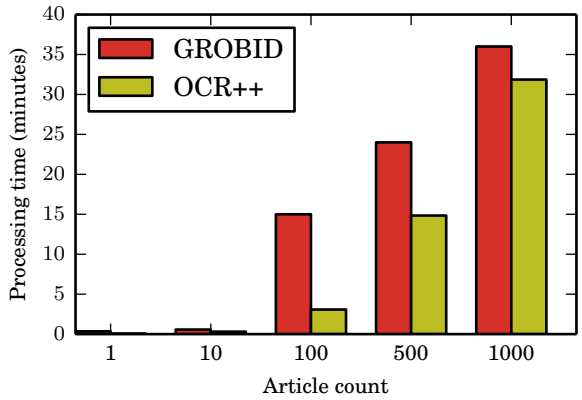


Figure 4: Comparison between batch processing time of *GROBID* and *OCR++*.

3.3 User experience study

To conduct a user experience study, we present *OCR++* to a group of researchers (subjects). Each subject is given two URLs: 1) OCR++ server URL and 2) Google survey form[10]. A subject can upload

[10]http://tinyurl.com/juxq2bt

any research article in PDF format on the server and visualize the output. In the end, the subject has to fill in a response sheet on the Google form. We ask subjects questions related to their experience such as, a) which extraction task did you like the most? b) have you found the system to be really useful? c) have you used similar kind of system before, d) do you find the system slow, fast or moderate, e) comments on the overall system experience and f) drawbacks of the system and suggestions for improvements.

A total of 30 subjects participated in the user experience survey. Among the most liked sub-tasks, title extraction comes first with 50% of votes. Affiliation and author name extraction tasks come second and third respectively. All the subjects found the system to be very useful. Only two of the subjects had used a similar system before. As far as the computational speed is concerned, 50% subjects found the system performance to be fast while 33% felt it to be moderate.

4 Use Cases

4.1 Curation of dataset links

With the community experiencing a push towards reproducibility of results, the links to datasets in the research papers are becoming very informative sources for researchers. Nevertheless, to the best of our knowledge, we do not find any work on automatic curation of dataset links from the conference proceedings. With *OCR++*, we can automatically curate dataset related links present in the articles. In order to investigate this in further detail, we aimed to extract dataset links from the NLP venue proceedings. We ran *OCR++* on four NLP proceedings, ACL 2015, NAACL 2015, ACL 2014, and ECAL 2014, available in PDF format. We extract all the URLs present in the proceedings. We then filter those URLs which are either part of *Datasets* section's body or are present in the footnotes of *Datasets* section, along with the URLs that consist of one of the three tokens: *datasets, data, dumps*. Table 5 presents statistics over these four proceedings for the extraction task. From the dataset links thus obtained, precision was found by human judgement as to whether a retrieved link corresponds to a dataset. One clear trend we saw was the increase in the number of dataset links from year 2014 to 2015. In some cases, the retrieved link corresponds to project pages, tools, researcher's homepage etc. resulting in lowering of precision values.

Table 5: Proceedings dataset extraction statistics: Article count represents total number of articles present in the proceedings. Total links and Dataset links correspond to total number of unique URLs and total number of unique dataset links extracted by OCR++ respectively. Precision measures the fraction of dataset links found to be correct.

Venue	Year	Articles Count	Total links	Dataset links	Precision
ACL	2015	174	345	38	0.74
NAACL	2015	186	186	18	0.50
ACL	2014	139	202	16	0.50
EACL	2014	78	141	12	0.67

4.2 Section-wise citation distribution

Citation instance count plays a very important role in determining future popularity of a research paper. An article's text is distributed among several sections. Some sections have more fraction of citations than the rest. In the second use case, we plan to study the section-wise citation distribution. Section-wise citation distribution refers to how citations are distributed over multiple sections in the article's text. This is an important characteristic of the citations and has recently been used for developing a faceted recommendation system (Chakraborty et al., 2016). We group specific sections to 5 generic sections, Background, Datasets, Method, Result/Evaluation and Discussion/Conclusion. Table 6 shows an example mapping from specific to generic section names. Note that this mapping can be changed as per the requirement. Figure 5 shows citation distribution for article dataset consisting of the 138 articles mentioned earlier. Maximum number of citations are present in the method section, followed by background and discussion and conclusion. Result section comprises the least number of citations.

Table 6: Specific to generic section mapping

Generic section	Specific sections
Background	Introduction, Related Work, Background
Method	Methodology, *Method Specific names*
Result/Evaluation	Results, Evaluation, Metrics
Discussion/Conclusion	Discussion, Conclusion, Acknowledgment

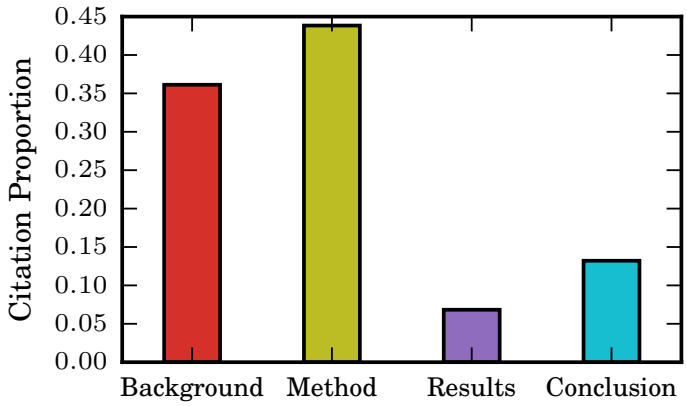


Figure 5: Section-wise citation distribution in article dataset

5 Deployment

The current version of *OCR++* is deployed at our research group server [11]. The present infrastructure consists of single CentOS instance[12]. We make the entire source code publicly available [13]. We also plan to make annotated dataset publicly available in the near future.

6 Related work

Researchers follow diverse approaches for individual extraction tasks. The approaches based on image processing segment document image into several text blocks. Further, each segmented block is classified into a predefined set of logical blocks using machine learning algorithms. Gobbledoc (Nagy et al., 1992) used X-Y tree data structure that converts the two-dimensional page segmentation problem into a series of one-dimensional string-parsing problems. Dengel and Dubiel (1995) employed the concept language of the GTree for logical labeling. Similar work by Esposito et al. (1995) presented a hybrid approach to segment an image by means of a top-down technique and then bottom-up approach to form complex layout component.

Similarly, current state-of-the-art systems use support vector machine (SVM) (Han et al., 2003) and conditional random field (CRF) (Councill et al., 2008; Luong et al., 2012; Lafferty et al., 2001) based machine learning models for information extraction. A study by Granitzer et al. (2012) compares *Parsecit* (a CRF based system) and *Mendeley Desktop*[14] (an SVM based system). They observed that SVMs provide a more reasonable performance in solving the challenge of metadata extraction than CRF based approach. However, Lipinski et al. (2013) observed that *GROBID* (a CRF based system) performed better than *Mendeley Desktop*.

[11] CNeRG. http://www.cnergres.iitkgp.ac.in
[12] OCR++ server. http://www.cnergres.iitkgp.ac.in/OCR++/home/
[13] Source code. http://tinyurl.com/hs9oap2.
[14] https://www.mendeley.com/

7 Conclusions

The goal of this work was to develop an open-source information extraction framework for scientific articles using generic patterns present in various publication formats. In particular, we extract metadata information and section related information. The framework also performs two mapping tasks, author and e-mail mapping and citations to reference mapping. Despite OCR errors and the great difference in the publishing formats, the framework outperforms the state-of-the-art systems by a high margin. We find that the hand-written rules and heuristics produced better results than previously proposed machine learning models.

The current framework has certain limitations. As described in Section 2, we employ *pdf2xml* to convert a PDF article into a rich XML file. Even though the XML file consists of rich metadata, it suffers from common errors generated during PDF conversion. Example of such common errors are the end-of-line hyphenation and character encoding problem. This is a common problem especially in two-column articles. Secondly, the current version of *pdf2xml* lacks character encoding for non English characters. It also suffers from complete omission or in some cases mangling of ligatures, i.e. typical character combinations such as 'fi' and 'fl' that are rendered as a single character in the PDF, which are not properly converted and often lost, for example, resulting in the non-word 'gure' for 'figure'. The three mentioned major problems along with other minor PDF conversion errors are directly reflected in the *OCR++* output.

As discussed in the previous section, a considerable amount of work has already been done to extract reference entities (title, publisher name, date, DOI, etc.) with high accuracy. Therefore, *OCR++* does not aim to extract reference entities. In future, we aim to extend the current framework by extracting information present in figures and tables. Figures and tables present concise statistics about dataset and results. Another possible extension is to add a mapping from unstructured reference to structured reference record. We are also currently in process to extend the functionality for non-English articles.

References

Joeran Beel, Bela Gipp, Stefan Langer, Marcel Genzmehr, Erik Wilde, Andreas Nürnberger, and Jim Pitman. 2011. Introducing Mr. Dlib, a machine-readable digital library. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, pages 463–464. ACM.

Tanmoy Chakraborty, Amrith Krishna, Mayank Singh, Niloy Ganguly, Pawan Goyal, and Animesh Mukherjee. 2016. FeRoSA: A faceted recommendation system for scientific articles. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 528–541. Springer.

Isaac G. Councill, C. Lee Giles, and Min-Yen Kan. 2008. ParsCit: an open-source CRF reference string parsing package. In *LREC*.

Andreas Dengel and Frank Dubiel. 1995. Clustering and classification of document structure-a machine learning approach. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 2, pages 587–591. IEEE.

Floriana Esposito, Donato Malerba, and Giovanni Semeraro. 1995. A knowledge-based approach to the layout analysis. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 466–471. IEEE.

Michael Granitzer, Maya Hristakeva, Kris Jack, and Robert Knight. 2012. A comparison of metadata extraction techniques for crowdsourced bibliographic metadata management. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 962–964. ACM.

Hui Han, C. Lee Giles, Eren Manavoglu, Hongyuan Zha, Zhenyue Zhang, Edward Fox, et al. 2003. Automatic document metadata extraction using support vector machines. In *Digital Libraries, 2003. Proceedings. 2003 Joint Conference on*, pages 37–48. IEEE.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Mario Lipinski, Kevin Yao, Corinna Breitinger, Joeran Beel, and Bela Gipp. 2013. Evaluation of header metadata extraction approaches and tools for scientific PDF documents. In *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*, pages 385–386. ACM.

Patrice Lopez. 2009. GROBID: Combining automatic bibliographic data recognition and term extraction for scholarship publications. In *Research and Advanced Technology for Digital Libraries*, pages 473–474. Springer.

Minh-Thang Luong, Thuy Dung Nguyen, and Min-Yen Kan. 2012. Logical structure recovery in scholarly articles with rich document features. *Multimedia Storage and Retrieval Innovations for Digital Library Systems*, page 270.

George Nagy, Sharad Seth, and Mahesh Viswanathan. 1992. A prototype document image analysis system for technical journals. *Computer*, 25(7):10–22, July.

Efficient Data Selection for Bilingual Terminology Extraction from Comparable Corpora

Amir Hazem[1] **Emmanuel Morin**[1]

[1] LINA - UMR CNRS 6241, Université de Nantes, France
{amir.hazem,emmanuel.morin}@univ-nantes.fr

Abstract

Comparable corpora are the main alternative to the use of parallel corpora to extract bilingual lexicons. Although it is easier to build comparable corpora, specialized comparable corpora are often of modest size in comparison with corpora issued from the general domain. Consequently, the observations of word co-occurrences which are the basis of context-based methods are unreliable. We propose in this article to improve word co-occurrences of specialized comparable corpora and thus context representation by using general-domain data. This idea, which has been already used in machine translation task for more than a decade, is not straightforward for the task of bilingual lexicon extraction from specific-domain comparable corpora. We go against the mainstream of this task where many studies support the idea that adding out-of-domain documents decreases the quality of lexicons. Our empirical evaluation shows the advantages of this approach which induces a significant gain in the accuracy of extracted lexicons.

1 Introduction

Comparable corpora are the main alternative to the use of parallel corpora for the task of bilingual lexicon extraction, particularly in specialized and technical domains for which parallel texts are usually unavailable or difficult to obtain. Although it is easier to build comparable corpora (Talvensaari et al., 2007), specialized comparable corpora are often of modest size (around 1 million words) in comparison with general-domain comparable corpora (up to 100 million words) (Morin and Hazem, 2016). The main reason is related to the difficulty to obtain many specialized documents in a language other than English. For example, a single query on the Elsevier portal[1] of documents containing in their title the term "breast cancer" returns 40,000 documents in English, where the same query returns 1,500 documents in French, 693 in Spanish and only 7 in German.

The historical context-based approach dedicated to the task of bilingual lexicon extraction from comparable corpora, and also known as the *standard approach*, relies on the simple observation that a word and its translation tend to appear in the same lexical contexts (Fung, 1995; Rapp, 1999). In this approach, each word is described by its lexical contexts in both source and target languages, and words in translation relationship should have similar lexical contexts in both languages. To enhance bilingual lexicon induction, recent approaches use more sophisticated techniques such as topic models based on bilingual latent dirichlet allocation (*BiLDA*) (Vulic and Moens, 2013b; Vulic and Moens, 2013a) or bilingual word embeddings based on neural networks (Gouws et al., 2014; Chandar et al., 2014; Vulic and Moens, 2015; Vulic and Moens, 2016) (approaches respectively noted: *Gouws*, *Chandar* and *BWESG+cos*). All these approaches require at least sentence-aligned/document aligned parallel data (*BiLDA*, *Gouws*, *Chandar*) or non-parallel document-aligned data at the topic level (BWESG+cos). Since specialized comparable corpora are of small size, sentence-aligned (document aligned) parallel data are unavailable and non-parallel document-aligned data at the topic level can't be provided since specialized comparable corpora usually deal with one single topic. Based on the recent comparison in (Vulic and Moens, 2015; Vulic and Moens, 2016) where the standard approach (noted in there article as *PPMI+cos*) performed better in most cases while compared to *BiLDA*, *Gouws* and *Chandar*, and due to the unavailability of non parallel

[1] www.sciencedirect.com

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3401–3411, Osaka, Japan, December 11-17 2016.

document aligned data at the topic level, we only deal with the standard approach and show at least that our approach improve drastically bilingual terminology extraction while adding well selected external data.

The small size of specialized comparable corpora renders unreliable word co-occurrences which are the basis of the standard approach. In this paper, we propose to improve the reliability of word co-occurrences in specialized comparable corpora by adding general-domain data. This idea has already been successfully employed in machine translation task (Moore and Lewis, 2010; Axelrod et al., 2011; Wang et al., 2014, among others). The approach of using adapted external data, also known as *data selection* is often applied in Statistical Machine Translation (SMT) to improve the quality of the language and translation models, and hence, to increase the performance of SMT systems. If data selection has become a mainstream in SMT, it is still not the case in the task of bilingual lexicon extraction from specialized comparable corpora. The majority of the studies in this area support the principle that the quality of the comparable corpus is more important than its size and consequently, increasing the size of specialized comparable corpora by adding out-of-domain documents decreases the quality of bilingual lexicons (Li and Gaussier, 2010; Delpech et al., 2012). This statement remains true as long as the used data is not adapted to the domain. We propose two data selection techniques based on the combination of a specialized comparable corpus with external resources. Our hypothesis is that word co-occurrences learned from a general-domain corpus for general words (as opposed to the terms of the domain) improve the characterization of the specific vocabulary of the corpus (the terms of the domain). By enriching the general words representation in specialized comparable corpora, we improve their characterization and therefore improve the characterization of the terms of the domain for better discrimination.

The remainder of this article is organized as follows: Section 2 describes the standard approach to bilingual lexicon extraction from comparable corpora. Section 3 presents previous works related to the improvements of the standard approach for specialized comparable corpora. Section 4 describes our strategies to improve the characterization of lexical contexts. Section 5 presents the different textual resources used for our experiments: the specialized and general comparable corpora, the bilingual dictionary and the terminology reference lists. Section 6 evaluates the influence of using lexical contexts built from general comparable corpora on the quality of bilingual terminology extraction. Section 7 presents our conclusions.

2 Standard Approach

Bilingual lexicon extraction from comparable corpora relies on the simple assumption that a word and its translation tend to appear in the same lexical contexts. Based on this assumption, the standard approach can be carried out by applying the following steps:

1. Build for each word w of the source and the target languages a context vector (resp. $\mathbf{s}$ and $\mathbf{t}$ for source and target languages) by identifying the words that appear in a window of n words around w normalized according to the measure of association of each word in the context of w. The association measures studied are Mutual Information (Fano, 1961), Log-likelihood (Dunning, 1993), and the Discounted Odds-Ratio (Evert, 2005).

2. Translate with a bilingual dictionary the context vector of a word to be translated from the source to the target language ($\bar{\mathbf{i}}$ the translated context vector).

3. Compare the translated context vector $\bar{\mathbf{i}}$ to each context vector of the target language $\mathbf{t}$ through a similarity measure and rank the candidate translations according to this measure. The similarity measures employed are Cosine (Salton and Lesk, 1968) and weighted Jaccard (Grefenstette, 1994)

3 Related Work

In the past few years, several contributions have been proposed to improve each step of the standard approach. Prochasson et al. (2009) enhance the representativeness of the context vectors by strengthening the context words that happen to be transliterated and scientific compound words in the target

language. Ismail and Manandhar (2010) also suggest that context vectors should be based on the most important contextually relevant words (in-domain terms), and thus propose a method for filtering the noise of the context vectors. Bouamor et al. (2013) propose an adaption of the standard approach that exploits Wikipedia to improve the context vector representation. From the context vector of a word to be translated, they build a vector of Wikipedia concepts using the ESA inverted index (Explicit Semantic Analysis). This vector of concepts is then translated into the target language. The candidate translations are found by projecting the translated vector of concepts using the ESA direct index onto the context vector of the target language. Prochasson and Fung (2011) propose to use a machine learning approach based both on the context-vector similarity and the co-occurrence features to learn a model for rare words from one pair of languages and this model can be used to find translations from another pair of languages. Hazem and Morin (2013) study different word co-occurrence prediction models in order to make the observed co-occurrence counts in specialized comparable corpus more reliable by re-estimating their probabilities. Morin and Hazem (2016) show the unfounded assumption of the balance in terms of quantity of data of the specialized comparable corpora and that the use of unbalanced corpora significantly improves the results of the standard approach.

Other improvements to the standard approach have been proposed by introducing other paradigms. For instance, Gaussier et al. (2004) propose to apply Canonical Correlation Analysis (CCA) which is a bilingual extension of Latent Semantic Analysis (LSA) whereas Hazem and Morin (2012) propose to use Independent Component Analysis (ICA) which is basically an extension of the Principal Component Analysis (PCA). Vulić et al. (2011) also propose an extension of the Latent Dirichlet Allocation (LDA) taking into account bilinguality and called bilingual LDA (BiLDA), improvements of this latter can be found in (Vulic and Moens, 2013b; Vulic and Moens, 2013a). Gouws et al. (2014) and Chandar et al. (2014) use multilingual word embeddings based on sentence-aligned parallel data and/or translation dictionaries whereas Vulić and Moens. (2015; 2016) learn bilingual word embeddings from non-parallel document aligned data based on skip-gram model. These approaches are beyond the scope of this study because even if they improve the standard approach they are intended for large comparable corpora of general language and/or require parallel aligned data or non parallel aligned documents which are unavailable for specialized corpora. In this paper, we give a particular interest to the massive amount of general domain data that can be found on the web and discuss ways of taking advantage of these resources in order to enrich word context representation and improve the standard approach.

4 Adapted Standard Approach

We propose two adaptations of the standard approach. Based on the assumption that general domain information can benefit the task of bilingual lexicon extraction from specialized corpora, we enhance the standard approach for that purpose by jointly exploiting data from specialized and general domains.

4.1 Global Standard Approach

The first adaptation of the standard approach can be described as basic. It consists to build the context vectors from a comparable corpus composed of the specialized and the general comparable corpora. This adaptation is inspired by the work of Morin et al. (2010) that shows that the discourse categorization (scientific versus popular scientific documents) of the documents in a specialized comparable corpus increases the quality of the extracted French/Japanese lexicon composed of single-word terms despite the data sparsity. For alignment of multi-word terms, the discourse categorisation of documents is not relevant. This work suggests that increasing the size of the specialized comparable corpora by adding popular scientific documents is interesting.

4.2 Selective Standard Approach

In the second adaptation, we first build independently word' context vectors of the two corpora (specialized and general) and then, for each word that belongs to the specialized domain corpus, if it appears in the general domain corpus, we merge its specialized and general context vectors. This allows to filter general domain words that are not part of the specialized corpus and renders the selective standard ap-

proach much less time consuming than the global standard approach. The merging process carried out before the normalization of context vectors of the standard approach (see step 1 - Section 2) is done as follows:

Increasing word co-occurrence counts (Hyp1) if a word w_i co-occurs p times with w in the specialized domain and q times in the general domain, we simply add the two co-occurrence counts so that the merged context vector of w will contain w_i with a co-occurrence count of $p + q$.

Reducing the vector space model sparseness (Hyp2) if a word w_j co-occurs r times with w in the general domain but does not co-occur with w in the specialized domain, we add w_j to the merged context vector of w. In that case, w_j is considered to be as new information that is added to the context vector of w to enrich it.

By enhancing word co-occurrence counts, the context vectors of the words become more reliable. Whereas, by increasing the density of the vector space model, the context vectors of the words become more precise. This twofold strategy enables us to better characterize the words of the specialized comparable corpus without increasing the number of words to characterize. In this way, the candidate translations of a word are always selected from the vocabulary of the specialized comparable corpus.

In the same way that Hazem and Morin (2013), we use a general language corpus to make the observed word co-occurrence counts in a specialized comparable corpus more reliable. Like them, we modify the initial word co-occurrence counts, but unlike them, we introduce new words learned from the general corpus in the vector space model.

5 Data and Resources

In this section, we describe the data and resources we used in our experiments which are conducted on the French/English language pair.

5.1 Comparable Corpora

The specialized comparable corpora were selected in terms of bilingual terminology access of technical domains. For this purpose, comparable corpora gather texts sharing common features such as domain, topic, genre, discourse and period without having a source text-target text relationship which guarantees access to the original vocabulary. For our experiments, we used three French/English specialized comparable corpora:

Breast cancer corpus (BC) is composed of documents collected from the Elsevier website[1]. We have selected the documents published between 2001 and 2008 where the title or the keywords contain *cancer du sein* in French and *breast cancer* in English.

Volcanology corpus (VG) was manually built by gathering documents dedicated to volcanology such as web documents, academic textbooks, popular science books, general newspapers, popular and semi-popular science magazines, travel magazines, and glossaries.

Wind energy corpus (WE) has been released in the TTC project[2]. This corpus has been crawled from the web using *Babouk* crawler (Groc, 2011) based on several keywords such as *vent*, *énergies*, *éolien*, *renouvelable* in French and *wind*, *energy*, *rotor* in English.

In order to evaluate our approach , we explored different types and size of external data. Most of them are parallel corpora often used in multiple evaluation campaigns such as WMT[3]. It is to note that we do not take advantage of the parallel information. Using parallel corpora only insures a good degree of comparability. We briefly describe each corpus:

[2] www.ttc-project.eu
[3] www.statmt.org

Corpus	# content words		# distinct words		Comp.
	FR	EN	FR	EN	
Breast cancer	521,262	525,934	6,630	8,221	79.07
Volcanology	399,828	405,286	9,142	8,623	83.69
Wind energy	313,954	314,551	5,346	6,378	81.61
NC	5.7M	4.7M	23,597	29,489	88.52
EP7	61.8M	55.7M	40,861	46,669	87.90
JRC	70.3M	64.2M	100,004	93,104	85.30
CC	91.3M	81.1M	250,999	259,226	86.13
GW	353.4M	291.8M	299,784	323,280	85.56
UN	421.7M	361.9M	158,647	137,411	84.73

Table 1: Characteristics of the specialized corpora and the external data.

News commentary corpus (NC) is a twelve language parallel corpus of news commentaries provided by the WMT workshop for SMT[4].

Europarl corpus (EP7) is a parallel corpus for SMT extracted from the proceedings of the European Parliament. It contains about 21 languages. We used the French-English version 7 used for the WMT translation task[3]

JRC acquis corpus (JRC) is a collection of legislative European union texts[4]. We used the French-English aligned version at OPUS provided by JRC (Tiedemann, 2012).

Common crawl corpus (CC) is a petabytes of data collected over 7 years of web crawling set of raw web page data and text extracts[5].

Gigaword corpus (GW) is a set of monolingual newswire corpora provided by LDC[6].

United nations corpus (UN) is a six language parallel text of the United Nations originally provided as translation memory (Rafalovitch and Dale, 2009).

The French/English corpora were then normalized through the following linguistic pre-processing steps: tokenization, part-of-speech tagging, and lemmatization. Finally, the function words were removed and the words occurring less than twice in the French and in the English parts were discarded. Table 1 shows the size of the comparable corpora and also indicates the comparability degree in percentages (Comp.) between the French and the English parts of each comparable corpus. The comparability measure (Li and Gaussier, 2010) is based on the expectation of finding the translation for each word in the corpus and gives a good idea about how two corpora are comparable. We can notice that all the comparable corpora have a high degree of comparability.

5.2 Bilingual Dictionary

The bilingual dictionary used in our experiments is the French/English dictionary ELRA-M0033[7]. This resource is a general language dictionary which contains around 244,000 entries.

5.3 Gold Standard

To evaluate the quality of bilingual terminology extraction from comparable corpora, a bilingual terminology reference list that reflects the technical vocabulary of the comparable corpus is required. The

[4] opus.lingfil.uu.se
[5] commoncrawl.org
[6] www.ldc.upenn.edu
[7] www.elra.info

list is usually composed of more or less 100 single words: 95 single words in Chiao and Zweigenbaum (2002), 100 in Morin et al. (2010), 125 and 79 in Bouamor et al. (2013a). We build a reference list for each of the three comparable corpora using specialized glossaries available on the Web. For instance, the list is derived from the UMLS[8] for the breast cancer corpus. Concerning wind energy, the list is provided with the corpus[1]. In order to focus only on the vocabulary characteristic of the specialized corpus we remove technical terms that have a common meaning in the general domain such as *analysis*, *factor*, *method*, *result*, *study*, etc. Without this precaution, these terms would be mechanically better identified in a larger corpus. To discard these terms, we use for French the list of the ScienTexT Project[9] and for English the Academic Keyword List[10]. Each word of the reference lists appears at least 5 times in the specialized comparable corpus. The reference lists are composed of 248 terms for breast cancer, 156 terms for volcanology and 139 terms for wind energy.

6 Experiments

Table 2 shows the results of the standard approach (noted SA) using only specialized comparable corpora (BC, VG and WE) or using only external data (NC, EP7, JRC, CC, GW and UN). It also shows the two adapted standard approaches (noted GSA and SSA) using the combination of each specialized comparable corpus with each corpus of the external data. The scores are measured in terms of the Mean Average Precision (MAP). We also used the three most exploited association and similarity measure configurations: Mutual Information with Cosine (noted MI-COS), Discounted Odds-Ratio with Cosine (noted OR-COS) and finally, Log-likelihood with weighted Jaccard (noted LL-JAC).

The first column of Table 2 shows the results of the SA for the three specialized comparable corpora. We can see that for each corpus the results differ according to a given measure configuration. Overall, for SA, the best results are obtained using the LL-JAC configuration. The SA for instance obtains a MAP score of 34.6% using BC corpus and a MAP score of 50.4% using VG corpus.

From the second to the seventh column, Table 2 shows the results of the SA using external data only, and our two adapted approaches (GSA and SSA). Column four for instance, shows the results of SA that uses the JRC corpus only. It also shows the results of GSA and SSA that combine the JRC corpus with each specialized corpus. GSA for instance obtains a MAP score of 63.3% and SSA a MAP score of 66.8% while combining the BC corpus with the JRC corpus (MI-COS configuration). Comparatively, and for the same configuration, SA using JRC corpus only, obtains a MAP score of 53.2%.

The first comment concerns the SA where surprisingly, using external data only, almost always improves its performance. This is particularly noticed when using external data of large size such as CC, GW and UN corpora. The good results obtained using these latter corpora can be explained by their characteristics. The Common crawl corpus (CC) for instance which has been crawled from the web, contains many scientific and specialized documents that can improve context representation. In addition, its large size makes co-occurrence counts more reliable. According to Table 3 we can see that more than 90% of the distinct words of the specialized corpora are present in the large general domain corpora.

The second comment concerns GSA and SSA where both always outperform SA for all the configurations. For the BC corpus for instance, we can notice that GSA obtains a MAP score of 81.5% and SSA obtains a MAP score of 83.4% using the GW corpus (LL-JAC configuration) while SA obtains a MAP score of 34.6% using BC and a MAP score of 78.3% using the GW corpus. Using other external data also improves the results of SA using the BC corpus. For instance, SSA obtains a MAP score of 65.9% using JRC, 57.5% using EP7 and 57.8% using NC. This means that adding external data always benefits bilingual lexicon extraction. If both GSA and SSA always improve bilingual lexicon extraction for the three specialized corpora, the results of Table 2 show that SSA outperforms GSA for almost all the configurations. This means that enriching the words that belong to the specialized domain corpus, if they appear in the general domain corpus (by merging context vectors) is more efficient than using a global combination (GSA). In addition, it should be noted that SSA is much faster than GSA.

[8] www.nlm.nih.gov/research/umls

[9] scientext.msh-alpes.fr

[10] www.uclouvain.be/en-372126.html

	BC	NC	EP7	JRC	CC	GW	UN	
SA	25.9	44.9	49.8	53.2	75.8	83.6	57.9	MI-COS
GSA	-	55.8	60.1	63.3	80.7	85.0	66.7	
SSA	-	**57.8**	**60.9**	**66.8**	**81.6**	**85.6**	**67.1**	
SA	27.0	45.3	48.5	52.0	75.5	81.1	55.7	OR-COS
GSA	-	**58.9**	58.3	61.7	80.2	83.2	58.9	
SSA	-	**58.9**	**60.8**	**66.6**	**82.3**	**85.5**	**67.2**	
SA	34.6	45.4	45.4	49.3	72.8	78.3	50.7	LL-JAC
GSA	-	57.4	56.3	63.0	77.2	81.5	62.0	
SSA	-	**57.8**	**57.5**	**65.9**	**78.7**	**83.4**	**65.5**	

(a) Breast cancer corpus

	VG	NC	EP7	JRC	CC	GW	UN	
SA	22.7	47.9	50.0	51.7	77.5	75.0	62.7	MI-COS
GSA	-	55.1	58.1	61.3	**78.3**	**78.7**	68.6	
SSA	-	**57.5**	**60.7**	**64.4**	78.1	76.0	**68.7**	
SA	37.9	49.7	50.2	49.3	75.6	73.9	59.4	OR-COS
GSA	-	61.6	60.4	59.0	77.2	76.3	67.5	
SSA	-	**62.2**	**61.3**	**62.3**	**78.5**	**78.8**	**68.5**	
SA	50.4	48.4	45.8	45.1	71.2	68.7	50.9	LL-JAC
GSA	-	63.0	60.6	58.3	**73.3**	70.6	59.3	
SSA	-	**64.0**	**62.4**	**58.9**	73.2	**72.8**	**61.2**	

(b) Volcanology corpus

	WE	NC	EP7	JRC	CC	GW	UN	
SA	15.6	41.0	51.0	63.4	72.1	67.4	60.4	MI-COS
GSA	-	47.3	**54.6**	65.3	73.2	69.1	64.1	
SSA	-	**50.5**	53.2	**67.8**	**74.9**	**70.8**	**66.9**	
SA	19.4	45.4	50.0	60.8	71.3	68.1	58.3	OR-COS
GSA	-	52.3	51.8	64.2	72.3	70.3	60.8	
SSA	-	**52.8**	**53.9**	**66.8**	**74.8**	**72.5**	**63.7**	
SA	28.0	43.6	45.1	60.0	65.0	62.5	48.6	LL-JAC
GSA	-	42.9	46.0	59.7	64.7	63.0	50.8	
SSA	-	**43.8**	**48.7**	**61.6**	**66.2**	**65.7**	**53.6**	

(c) Wind energy corpus

Table 2: Results (MAP %) of the *Standard Approach* (SA), the *Global Standard Approach* (GSA) and the *Selective Standard Approach* (SSA) for the breast cancer corpus (BC), the volconalogy corpus (VG) and the wind energy corpus (WE) using the news commentary corpus (NC), the Europarl corpus (EP7), the JRC acquis corpus (JRC), the common crawl corpus (CC), the Gigaword corpus (GW) and the united nation corpus (UN) (the improvements indicate a significance at the 0.001 level using the Student t-test).

		BC+NC	BC+EP7	BC+JRC	BC+CC	BC+GW	BC+UN
# Hyp1(∩)	FR	3,939	4,366	4,789	5,502	**5,907**	5,142
	EN	4,315	4,668	5,451	6,303	**7,103**	5,701
# Hyp2(∪)	FR	721	1,931	1,067	3,211	**4,503**	3,330
	EN	746	1,767	1,013	2,833	**4,952**	3,215
		VG+NC	VG+EP7	VG+JRC	VG+CC	VG+GW	VG+UN
# Hyp1(∩)	FR	6,472	7,184	6,808	**8,426**	8,330	7,901
	EN	6,190	6,581	6,214	7,825	**7,864**	7,142
# Hyp2(∪)	FR	556	1,480	861	2,872	**3,910**	2,700
	EN	614	1,436	904	2,829	**4,866**	2,827
		WE+NC	WE+EP7	WE+JRC	WE+CC	WE+GW	WE+UN
# Hyp1(∩)	FR	3,804	4,136	4,535	4,909	**4,944**	4,770
	EN	4,246	4,582	5,071	5,546	**5,767**	5,331
# Hyp2(∪)	FR	790	2,135	1,204	3,842	**5,531**	3,663
	EN	784	1,901	1,174	3,422	**6,350**	3,715

Table 3: Number of distinct context vectors that have been augmented (enriched).

SSA translation candidates are those of the specialized domain only (around 6,600 candidates for the BC corpus) and GSA translation candidates are those of the specialized domain plus those of the general domain (around 250,000 candidates for CC corpus - see Table 1) which render the computation of vector similarity much more time consuming. Overall, we can see that the results differ according to the configuration measures used. If for SA, the best results are always obtained using LL-JAC, this is not the case for GSA and SSA. For the BC corpus for instance, SSA obtains the highest MAP score of 85.6% using GW and the MI-COS configuration while for the VG corpus combined with GW, we can see that the best MAP score of 78.8% is obtained by SSA using the OR-COS configuration. These differences are mainly due to the measure properties. If the MI measure shows poor results on small corpora, it is mainly because it overestimates low counts and underestimates high counts. This disadvantage is smoothed when using more data. The differences between MI and OR measures are too low to conclude which is the most appropriate one to use as we obtain more or less equivalent results for the used corpora.

Table 3 shows the number of distinct words of each specialized corpus that have been enriched using each general-domain corpus. Hyp1 corresponds to the first hypothesis of SSA in which we assume that only the context vectors of the specialized corpora should be enriched. So the Hyp1 column shows the number of distinct words that appear in both the specialized and the general domain corpora. For instance, Hyp1 of the BC corpus and the NC corpus noted BC+NC, shows that there are 4,315 words in common for their English parts and 3,939 in common for their French parts. One can notice that a high amount of specialized context vectors are enriched thanks to general-domain corpus. Hyp2 corresponds to the mean of the number of new words that have been added to each context vector of the specialized domain words. For instance, Hyp2 for BC+NC shows that in average we add 746 new English words and 721 new French words for each context vector of the BC corpus. Here also we can see that many new words are added to the specialized context vectors. The experimental results previously shown in Table 2 confirm the usefulness of Hyp1 and Hyp2.

7 Conclusion

We have shown in this article how the problem of adding external data could be achieved for improving bilingual lexicon extraction from specialized comparable corpora. We have proposed two approaches that use external data in an adapted way to preserve the original vocabulary. Even if our selective standard approach goes against the mainstream which states that adding out-of-domain data decreases the quality of bilingual lexicons, we never denature the initial specialized comparable corpus. The results obtained

by the selective standard approach show significant improvements for alignment of single-word terms while using any of the external data and confirm the usefulness of exploiting as much data as we have to better characterize context vector representation and thus bilingual lexicon extraction.

Acknowledgments

The research leading to these results has received funding from the French National Research Agency under grant ANR-12-CORD-0020 (CRISTAL project) and the French Projet OCEAN (Outil de Concordance bilinguE libre pour l'Aide la traductioN) of the General Delegation for the French Language and in languages of France.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'11)*, pages 355–362, Edinburgh, Scotland, UK.

Dhouha Bouamor, Adrian Popescu, Nasredine Semmar, and Pierre Zweigenbaum. 2013a. Building Specialized Bilingual Lexicons Using Large Scale Background Knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*, pages 479–489, Seattle, WA, USA.

Dhouha Bouamor, Nasredine Semmar, and Pierre Zweigenbaum. 2013b. Context vector disambiguation for bilingual lexicon extraction from comparable corpora. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*, pages 759–764, Sofia, Bulgaria.

A. P. Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M. Khapra, Balaraman Ravindran, Vikas C. Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1853–1861, Montreal, Quebec, Canada.

Yun-Chuang Chiao and Pierre Zweigenbaum. 2002. Looking for candidate translational equivalents in specialized, comparable corpora. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, pages 1208–1212, Tapei, Taiwan.

Estelle Delpech, Béatrice Daille, Emmanuel Morin, and Claire Lemaire. 2012. Extraction of domain-specific bilingual lexicon from comparable corpora: compositional translation and ranking. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING'12)*, pages 745–762, Mumbai, India.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Stefan Evert. 2005. *The statistics of word cooccurrences : word pairs and collocations*. Ph.D. thesis, University of Stuttgart.

Robert M. Fano. 1961. *Transmission of Information: A Statistical Theory of Communications*. MIT Press, Cambridge, MA, USA.

Pascale Fung. 1995. Compiling bilingual lexicon entries from a non-parallel english-chinese corpus. In *Proceedings of the 3rd Annual Workshop on Very Large Corpora (VLC'95)*, pages 173–183, Cambridge, MA, USA.

Eric. Gaussier, Jean-Michel Renders, Irena. Matveeva, Cyril. Goutte, and Hervé Déjean. 2004. A geometric view on bilingual lexicon extraction from comparable corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL'04)*, pages 526–533, Barcelona, Spain.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2014. Bilbowa: Fast bilingual distributed representations without word alignments. *CoRR*, abs/1410.2455.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publisher, Boston, MA, USA.

Clément De Groc. 2011. Babouk : Focused Web Crawling for Corpus Compilation and Automatic Terminology Extraction. In *Proceedings of 10th International Conferences on Web Intelligence (WIC'11)*, pages 497–498, Lyon, France.

Amir Hazem and Emmanuel Morin. 2012. ICA for Bilingual Lexicon Extraction from Comparable Corpora. In *Proceedings of the 5th Workshop on Building and Using Comparable Corpora (BUCC'12)*, pages 126–133, Istanbul, Turkey.

Amir Hazem and Emmanuel Morin. 2013. Word co-occurrence counts prediction for bilingual terminology extraction from comparable corpora. In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP'13)*, pages 1392–1400, Nagoya, Japan.

Azniah Ismail and Suresh Manandhar. 2010. Bilingual lexicon extraction from comparable corpora using in-domain terms. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10)*, pages 481–489, Beijing, China.

Bo Li and Éric Gaussier. 2010. Improving corpus comparability for bilingual lexicon extraction from comparable corpora. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10)*, pages 644–652, Beijing, China.

Robert C. Moore and William D. Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL'10)*, pages 220–224, Uppsala, Sweden.

Emmanuel Morin and Amir Hazem. 2016. Exploiting unbalanced specialized comparable corpora for bilingual lexicon extraction. *Natural Language Engineering*, 22(4):575–601.

Emmanuel Morin, Batrice Daille, Koichi Takeuchi, and Kyo Kageura. 2010. Brains, not brawn: The use of "smart" comparable corpora in bilingual terminology mining. *ACM Transactions on Speech and Language Processing*, 7(1):1–23.

Emmanuel Prochasson and Pascale Fung. 2011. Rare Word Translation Extraction from Aligned Comparable Documents. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL'11)*, pages 1327–1335, Portland, OR, USA.

Emmanuel Prochasson, Emmanuel Morin, and Kyo Kageura. 2009. Anchor points for bilingual lexicon extraction from small comparable corpora. In *Proceedings of the 12th Conference on Machine Translation Summit (MT Summit XII)*, pages 284–291, Ottawa, Canada.

Alexandre Rafalovitch and Robert Dale. 2009. United nations general assembly resolutions: A six-language parallel corpus. In *Proceedings of the 12th Conference on Machine Translation Summit (MT Summit XII)*, Ottawa, Canada.

Reinhard Rapp. 1999. Automatic Identification of Word Translations from Unrelated English and German Corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, pages 519–526, College Park, MD, USA.

Gerard Salton and Michael E. Lesk. 1968. Computer evaluation of indexing and text processing. *Journal of the Association for Computational Machinery*, 15(1):8–36.

Tuomas Talvensaari, Jorma Laurikkala, Kalervo Järvelin, Martti Juhola, and Heikki Keskustalo. 2007. Creating and exploiting a comparable corpus in cross-language information retrieval. *ACM Transactions on Information Systems (TOIS)*, 25(1).

Jörg Tiedemann. 2012. Parallel Data, Tools and Interfaces in OPUS. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey.

Ivan Vulic and Marie-Francine Moens. 2013a. Cross-lingual semantic similarity of words as the similarity of their semantic word responses. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'13)*, pages 106–116, Atlanta, GA, USA.

Ivan Vulic and Marie-Francine Moens. 2013b. A study on bootstrapping bilingual vector spaces from non-parallel data (and nothing else). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*, pages 1613–1624, Seattle, WA, USA.

Ivan Vulic and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL'15)*, pages 719–725, Beijing, China.

Ivan Vulic and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *J. Artif. Intell. Res. (JAIR)*, 55:953–994.

Ivan Vulić, Wim De Smet, and Marie-Francine Moens. 2011. Identifying word translations from comparable corpora using latent topic models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL'11)*, pages 479–484, Portland, OR, USA.

Longyue Wang, Derek F. Wong, Lidia S. Chao, Yi Lu, and Junwen Xing. 2014. A Systematic Comparison of Data Selection Criteria for SMT Domain Adaptation. *The Scientific World Journal*, 2014:10.

TweetGeo – A Tool for Collecting, Processing and Analysing Geo-encoded Linguistic Data

Nikola Ljubešić
Dept. of Knowledge Technologies
Jožef Stefan Institute
nikola.ljubesic@ijs.si

Tanja Samardžić
CorpusLab, URPP Language and Space
University of Zurich
tanja.samardzic@uzh.ch

Curdin Derungs
GISLab, URPP Language and Space
University of Zurich
curdin.derungs@geo.uzh.ch

Abstract

In this paper we present a newly developed tool that enables researchers interested in spatial variation of language to define a geographic perimeter of interest, collect data from the Twitter streaming API published in that perimeter, filter the obtained data by language and country, define and extract variables of interest and analyse the extracted variables by one spatial statistic and two spatial visualisations. We showcase the tool on the area and a selection of languages spoken in former Yugoslavia. By defining the perimeter, languages and a series of linguistic variables of interest we demonstrate the data collection, processing and analysis capabilities of the tool.

1 Introduction

Geographic distribution of linguistic features is traditionally studied in dialectology (regarding closely-related varieties) and in language typology (regarding different languages and language families), with the goal of identifying the patterns of language change. The potential for studying the geographic spread of linguistic features increased with the development of computer-mediated-communication (CMC). Short and long texts produced by the users of social media communication platforms constitute large samples of authentic language use that can be automatically retrieved and analysed to address a range of questions about human behavior, including spatial linguistic patterns analysed in this paper.

The social network Twitter made an especially important contribution to the development of new methods of collecting linguistic data from the Internet by allowing access to the content produced by their users through an API (application programming interface). One interesting feature of Twitter is that tweets are often associated with spatial information, explicitly (GPS coordinates) or implicitly (place names). The data collected from Twitter can either be used as a valuable complement to linguistic data already collected by traditional means or, if traditional data is not available, as a replacement. This opportunity, however, comes with considerable challenges. First, using the data collection interface requires technical skills that researchers interested in studying language variation usually cannot be expected to have. Second, once collected, the data often turns out to be noisy, difficult to annotate and unevenly distributed in space. Observing patterns therefore requires advanced processing methods, such as spatial statistics and geographic information science (GIScience).

In this paper, we present a tool set that combines computational linguistics and GIScience methods in order to facilitate the collection, visualisation and analysis of georeferenced tweets. Our major goal is to allow the wider linguistic research community access to data automatically collected from the Internet (Twitter data in this particular case). We provide a configurable tool set for speeding up the research process without limiting researchers in their choice of input data and analysis techniques. The user is, for instance, free to specify language(s), regions, and linguistic features of interest. Spatial analysis tools

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3412–3421, Osaka, Japan, December 11-17 2016.

can be applied to the extracted and annotated data for visualization and as a help in reasoning about the spatial patterns. In short, we present a tool intended to enable researchers with basic programming skills to perform advanced computational and spatial linguistic analysis.

Throughout the paper, we illustrate the functioning of the tool on an example data set collected for the territory of former Yugoslavia. We choose this region as a case where collecting new linguistic data is especially important. Due to the recent linguistic proliferation,[1] the current situation in this region provides an opportunity for all interested researchers to observe the impact of historical developments on language change. Collecting and analysing samples of computer-mediated-communication becomes particularly important since conducting large-scale linguistic surveys is impeded by the current political and economical situation.

2 Related Work

Computational analyses of the geographic distribution of linguistic features have a long tradition in dialectological research. The data for studying the spread of linguistic features are traditionally collected through questionnaires and field work: a number of potentially informative categories are selected and their realisations are elected from a number of informants selected to represent a linguistic variety in a particular area. Data collected in this way are then stored in databases that can be queried and used for different kinds of quantitative analyses (Nerbonne, 2009; Bauernschuster et al., 2014; Szmrecsanyi, 2012; Wieling et al., 2011). The knowledge about the distribution of linguistic features on the world-wide scale has recently become available in the form of databases. For instance, the data stored in a well-known typological database, WALS (Dryer and Haspelmath, 2013) is often used in large-scale computational studies of language universals (Dunn et al., 2011).

The trends in computational analysis of spatial linguistic data sets led to the development of specialised software such as GeoLing[2], a tool, written in Java, that enables researchers to visualise the spatial distribution of linguistic features using methods such as kernel density estimation (for smoothing data points representation on maps), factor analysis (for reducing the dimensionality) and clustering (for grouping similar areas). This tool requires a previously prepared data set, which can be collected using traditional methods.

User-generated content available on the Internet is used in computational linguistics mostly to study demographic characteristics of speakers based on the linguistic variety they use (Eisenstein et al., 2011; Nguyen et al., 2011; Danescu-Niculescu-Mizil et al., 2013), often including a geographic component (Doyle, 2014; Hovy and Johannsen, 2016). We concentrate here on the work where associated software was made available. Doyle (2014), proposes a method based on conditional probability to estimate the geographical distribution of linguistic features using Twitter. This method can be used to overcome the problem of uneven geographic distribution of collection points.[3] The software associated with this work is SeeTweet[4], a Python tool that uses the Twitter search API to collect tweets containing terms of interest, as well as base terms used for estimating the prior spatial frequency of tweets. The tool does not perform any visualisation of the collected data or any inference.

A recently developed web-based tool called Humboldt[5] (Hovy and Johannsen, 2016) provides a search interface that allows the user to query for lexical phenomena in five languages, and to get both statistical analysis and map representations of the results along two demographic factors: age and sex. This tool uses a data set previously collected by the authors from one source of online reviews of companies.

The tool that we propose in this article differs from the existing tools in its scope and flexibility. Previous tools are mostly focused either on data collection (SeeTweet) or analysis (GeoLing, Humboldt). We integrate these two components allowing the researchers to set up their own criteria both for collecting

[1] Following the war and the separation of SFR Yugoslavia's constitutive republics in the nineties, one of the official languages, Serbo-Croatian, was divided into four languages: Croatian, Bosnian, Montenegrin, and Serbian.

[2] `https://www.uni-ulm.de/en/mawi/geoling/home.html`.

[3] Note that the problem of uneven distribution does not concern traditional data collection methods, where balanced sampling (based on ZIP codes, for instance) is usually part of the design.

[4] `https://github.com/gabedoyle/seetweet`.

[5] `http://www.languagevariation.com`

and analysing data sets depending on their hypotheses. With our tool, the user can define a wide range of potential features to be extracted from a large set of messages. Unlike SeeTweet, where data collection is limited to searching for specific terms, we provide the possibility for capturing the whole (available) data stream, storing all the messages produced in a given region during the collection time. The extraction of the features of interest is again controlled by the user who applies predefined generic functions on his linguistic description of specific phenomena, consisting of either regular expressions or a lexical resource. Regarding the analysis, our tool offers flexibility by allowing users to dynamically switch between spatial summary statistics, simple visualisations and more sophisticated analysis. Importantly, all analysis steps are independent from the underlying spatial distribution of the collected data. This is important since CMC geo-encoded data is known to be biased towards places with high population and sparse in rural regions (Hecht and Stephens, 2014).

In the following three sections we describe the tool and the research set-up that it supports. Each section describes one of the three main components of the tool. With an example data set we illustrate how each component is configured and what it gives as a result. The tool, accompanied with the exemplary dataset, is made available on GitHub[6].

3 Data Collection

The data collection component communicates with the Public Twitter Streaming API and stores the messages to a given location.

It is written in Python and relies on the *tweepy* Twitter API wrapper. In order to start data collection, the user needs to edit the configuration file by entering his Twitter API credentials (obtained from the Twitter Developer site), the project name (arbitrarily defined by the user) and the perimeter of interest (defined by the longitude and latitude bounds). Once the process is launched, a database is created and all the messages obtained from the Public Streaming API are stored. Messages not containing explicit longitude and latitude are discarded.[7]

Each of the retrieved objects is stored in an *sqlite* database as a BLOB structure. Parts of these structures, the `lang` and `screen_name` attributes, are also explicitly stored in the database outside the BLOB for reporting purposes. From these entries, the user can get a collection update at any time, specifying the number of tweets collected, the number of speakers (Twitter users) who published the tweets, and the head of the frequency distribution of tweets per speaker and tweets per the `lang` attribute value.

For our use case we started the data collection procedure in January 2016. For illustration purposes, we use the data collected up to July 2016, while continuing to run the collection process. We defined the perimeter over the countries of former SFR Yugoslavia, namely Slovenia, Croatia, Bosnia, Montenegro, Serbia and FYR Macedonia. During the 6 month period we collected 526,658 tweets from 50,783 speakers.

4 Data Processing

The data processing module is written in Python. Its two main functionalities are data filtering and extraction of linguistic variables.

4.1 Data Filtering

At this point, we provide three speaker-level (i.e. Twitter-user-level) filtering criteria:

- the minimum number of tweets published by a speaker,
- the most prominent language(s) used by a speaker, and
- the most prominent countries from which the speaker tweets.

[6] `https://github.com/clarinsi/tweetgeo`

[7] There are two ways of encoding spatial information on Twitter, explicit position (longitude, latitude), or a location (ranging from a town to a country). We discard the latter as in many areas locations are too general to be useful for spatial analysis.

While implementing the first and last filtering functionalities was quite straightforward (the country from which the tweet was sent can be obtained directly from the `Status` object), the language filtering functionality required additional engineering. Namely, although Twitter messages are tagged with the `lang` attribute, this attribute is known to be very unreliable, especially for the so called smaller languages. To filter only the messages in the language(s) of interest, we perform additional language identification on the level of speaker by applying the off-the-shelf language identification tool *langid.py*[8] (Lui and Baldwin, 2012) on a concatenation of all tweets of a speaker. Before performing language identification, we remove mentions, hashtags and URLs , as such elements were not seen in the language identification training data.

Only the tweets produced by the speakers that satisfy the language constraints are passed to the variable extraction module.

The language filtering criteria are set in a configuration file which is shared with the feature extraction process. In our use case we allowed three languages known to *langid.py*: Croatian, Bosnian and Serbian. While we were collecting data published in Slovenia and FYR Macedonia as well, in this analysis we are not interested in Slovene nor Macedonian data, but want to retain the speakers of the languages of interest from these countries. For this use case we did not define any restrictions regarding the minumum number of tweets per user, and we allowed tweets from our countries of interest and their neighbouring countries.

By running the speaker-level language identification over the fifty thousand Twitter users in our example collection, we have identified 3,854 speakers of the languages of interest. Given that only 7% of the total of collected speakers were identified as writing in the defined languages, we have performed an evaluation of the language identification output by manually checking 200 random entries. While the precision on this sample was 1.0 (16 out of 200 cases), recall was 0.89 as two users were not identified as speakers of the languages of interest. However, each of the two missed users actually produced just one tweet consisting of two words beside smileys and mentions, making the loss negligible.

While there are four times more English speakers than those of the languages of interest, there is a comparable number of Italian speakers and a smaller number of Russian, Spanish, Turkish and Slovene speakers.

4.2 Variable Extraction

Variables represent the user's linguistic features of interest. Our tool allows for a great flexibility in defining the variables, allowing the researchers to express their theoretical insight and creativity in formal description of the linguistic phenomena, putting more weight on individual linguistic insights than it is usually the case in quantitative approaches which tend to use aggregate linguistic data. Deeper exploration of linguistic features and their interactions is in line with the current trends in spatial linguistic research (Wieling and Nerbonne, 2015).

We showcase the variable extraction module on five nominal variables relevant for our area of interest. The first variable, *yat* (illustrated in Table 1), covers the Proto-Slavic vowel which has a different reflex in different dialects, having two levels, *e* for text containing forms of the Ekavian dialect (*dete*, 'child') and *je* for that containing forms of the Jekavian dialect (*dijete*). The second variable, *štošta* focuses on the variation in the interrogative pronoun *what*, with two levels, Standard Croatian *što* and Standard Serbian *šta*. The third variable *daje* covers two levels of variation in the interrogative clitics, *je li* prescribed in Standard Croatian, and *da li* allowed in the remaining varieties. The fourth variable, *month* covers a two-level lexical variation, where Croatian contains specific names for months (*siječanj*, *veljača*...) encoded via variable level *hr*, while the remaining varieties use international ones (*januar*, *februar*...) encoded via variable level *int* (see Table 1 for some examples). The fifth variable, *rdrop* (also given in Table 1), covers the frequent drop of the ending *r* in Standard Serbian like *jučer* (encoded with level *r*) vs. *juče* (encoded with level *nor*).

Our variable extraction component is defined in the configuration file mentioned in the previous subsection in the form of four lists of functions.

[8] `https://github.com/saffsd/langid.py`

yat		month		rdrop	
Word	Value	Word	Value	Word	Value
dječak	je	juni	int	juče	nor
dečak	e	lipanj	hr	jučer	r
dječaka	je	august	int	naveče	nor
dečaka	e	kolovoz	hr	navečer	r

Table 1: A sample of a feature extraction lexicons; each column represent one file used by the extraction function.

The first list of functions operates on the `Status` object of the *tweepy* module, enabling extraction of metadata such as the number of retweets, posting time, whether the tweet is a reply to another tweet etc. The formalism requires for the user to define the location of the metadata in the *tweepy* `Status` object, like `user.screen_name` for the speaker's screen name or `favorited_count` for the number of times a status was favorited. The user can additionally define a function to be applied on the metadata value, such as extracting the posting year from the posting time, like `lambda x:str(x.year)`.

The remaining three function lists operate on the text of the tweet. While the second list of functions operates on the original text of the tweet, the third list of functions operates on the lowercased text, and the fourth one on the normalised text of the tweet.

The normalisation process can be defined by the user and covers, in our use case, removal of repeating characters, generalising spaces and removal of diacritics.

The choice of the text representation level from which the variable will be extracted depends on how important the literal representation of the writing is for extracting a particular variable. For instance, when idetifying the *što* pronoun, we use second level text representation – lowercased text. We want to easily take into account titlecased or uppercased versions of the pronoun, but do not want for diacritics to be removed as the form *što* clashes with the numeral *one hundred*. On the other side, when identifying names of months, we use the third level of text representation, covering with the form *ozujak* various forms like *Ožujak*, *ozujak* or *ožuJAAAAAK*.

The functions that can be run on any of the three mentioned text representations are divided into two types: the `lexicon_choice` and the `regex_choice` function.

4.2.1 Lexicon Choice

In the functions of the `lexicon_choice` type, the desired feature to be extracted is encoded by the user in the form of a lexicon file, where each line consists of a (word, value) pair. An example of such a lexicon in our use case is the lexicon of words containing the already mentioned Proto-Slavic vowel *yat*, as illustrated in the first column of the Table 1.

The *e* reflex is characteristic in the eastern variants (mostly Serbian), while the *je* reflex is found more to the western side of our target perimeter (Croatian, Bosnian, Montenegrin).

The lexicon illustrated in Table 1 was automatically generated from the Croatian and Serbian inflectional morphological lexicons hrLex and srLex (Ljubešić et al., 2016) by searching for pairs of words having the same morphosyntactic description and the word forms identical except the transformations (*ije* vs. *e*) or (*je* vs. *e*)), and both word forms having just one possible canonical form (lemma).

The `lexicon_choice` function iterates through the tokens of each of the collected tweets. If any of the tokens matches any word in the lexicon, the variable value associated with the word is added to the set of potential values. If, at the end of the tweet, there is only one value in the set of potential values, the function assigns that value to the tweet. In all other cases (no coverage, multiple values) the function returns the `NA` value.

The tokenisation function can be also modified by the user. It currently considers tokens to be hashtags, mentions, URLs or greedy alphanumeric sequences.

Similar lexicons can be specified by the user to extract any lexical variation features. Such lexicons can be written by hand or extracted automatically from other resources such in our case. Once they are stored in a location required by the function, they can be used for extraction.

In our use case, we have extracted four such lexicons, illustrated in Table 1. We have chosen these variables based on their varied use, as discussed in comparisons of the language varieties (Meša, 2011). Detailed documentation of the extracted variables is available in the tool documentation.

4.2.2 Regex Choice

The `regex_choice` function operates similarly to the `lexicon_choice` function, it just does not rely on a lexicon, but a list of pairs of regular expressions and variable values. If a regular expression is applicable to the text of a tweet, the corresponding variable value is added to the set of potential values. The decision which value should be returned is identical as in the `lexicon_choice` function. An example of such a function in our use case is the variation of the particles *je li* and *da li*, each being covered by the corresponding regular expressions `"\bje li\b"` and `"\bda li\b"`. These regular expressions are applied on the third level of text representation, namely normalised text.

5 Data Analysis

The analysis module is written in R, an open-source programming language which incorporates a wealth of packages for spatial data handling. At this level of the tool development we decided to limit ourselves to three functionalities: point visualisation, spatial trend detection and the identification of dominant regions per variable level. In the remainder of this section, each of the three functionalities is illustrated.

Here we stress one more time that in this analysis we only consider tweets associated with explicit geolocation, which is only given for some 1-3% of all Twitter messages (Leetaru et al., 2013). We do not attach geolocation to unlocated tweets by, for instance, using the place of domicile of the user location prediction procedures, as it has been shown that such procedures can lead to wrong assumptions (Hahmann et al., 2014).

5.1 Point Visualisation

The point visualisation allows to gain an initial impression of the spatial distribution of all levels of a linguistic feature. It can thus be considered a visual analytics tool (Andrienko et al., 2010).

For visualisation we use the leaflet framework [9], which allows to dynamically change the spatial focus by zooming and panning. This functionality proves to be vital for representing tweets as spatial points, due to the uneven spatial distribution discussed above. On small scales (e.g. country level) tweets are cluttered in populated places and it is thus often difficult to identify the exact distribution of feature levels without having the option of dynamically changing the scale and extent of the map.

Additionally, leaflet allows to activate an HTML popup option, which we use to allow access to the text content of each tweet through mouseclick. The user can for instance iterate through a subsample of the data and thus gain an impression on the data quality in terms of the spatial precision or linguistic variable extraction.

This functionality is also intended to be used alternately with the variable extraction module. Namely, besides analysing the output of the variable extraction process in pure text format, it is often easier to analyse the extracted variables in space and therefore get a faster insight in potential problems in the variable extraction process.

Examples of point visualisation on the *yat* and *štošta* variables can be seen on the left side of Figure 1 and Figure 2.

5.2 Spatial Trend Detection

Spatial linguistic analysis is often concerned with first-order effects, such as distributional patterns in the data (Diggle, 2014). With the spatial trend detection tool we intended to go one step further and introduce a simple measure that allows to quantify the spatial dependency in the data, often referred to as spatial autocorrelation or second-order effect. The quantification of spatial autocorrelation for continuous variables (e.g. temperature) is well established and measures such as *Moran's I* can be used (Moran, 1950). Quantifying spatial autocorrelation in nominal data, which we deal with in this paper, is

[9] `http://leafletjs.com/`

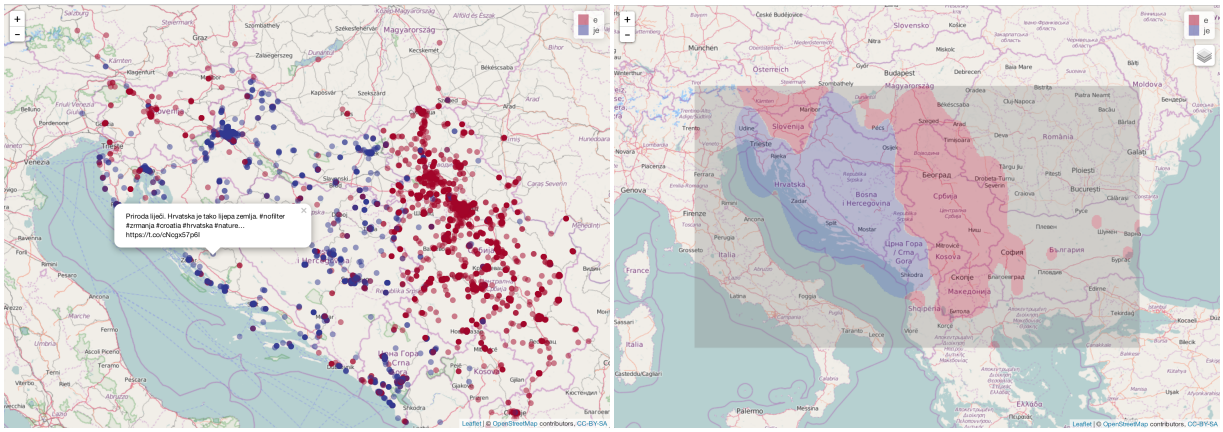


Figure 1: Result of point visualisation (left) and dominance map (right) for the variable *yat*

variable	level	spatial trend	frequency
yat	e	0.457	3298
	je	0.812	1702
stosta	što	1.079	1097
	šta	0.961	2205
daje	dali	0.902	497
	jeli	1.684	58
month	hr	1.252	16
	int	0.906	416
rdrop	r	0.402	79
	nor	0.713	619

Table 2: Statistical description of each variable and level

slightly less common. We compare the spatial distances as computed between all tweets of one linguistic feature (expected distances) with the distances as calculated for each feature level separately (observed distances). Aggregating these two sets of distances into what we call a *relative distance measure* allows us to distinguish feature levels that are spatially clustered (observed distance $<$ expected distance) from levels that are scattered in space (observed distance $>$ expected distance).

The results of the spatial trend detection applied to our five features is given in Table 2. We can observe that two variables having a strong spatial trend (low value equals strong trend), namely *yat* and *rdrop*.

In the *yat* variable the *e* level shows a higher spatial trend than the *ije* level, mostly due to the fact that the use of Ekavian is focused around Belgrade while the use of Jekavian is much more scattered around. This trend can also (partially) be observed in the point visualisation in Figure 1.

In the *rdrop* variable, the *r* level shows a much stronger spatial trend, which goes back to the fact that these variants are mostly used in Croatia only while in the remainder of the region the *nor* variants are used.

A somewhat surprising spatial trend is that of the *month* variable for which we would expect to have a strong spatial trend especially the *hr* level as Croatian month names are used in Croatia only. This result can be followed back to a low observation frequency of the variable in general, especially of the *hr* level. This result shows that, as most measures, the spatial trend heuristic is prone to outliers for small sample sizes.

The remaining two variables, *štošta* and *daje* show an expected weak spatial trend (higher is weaker) as these variants are used intermittently in the whole area of interest.

As shown with these examples, the spatial trend detection tool serves as a simple heuristic for deciding which linguistic features bear the potential of segregating space into larger linguistic areas. Ideally,

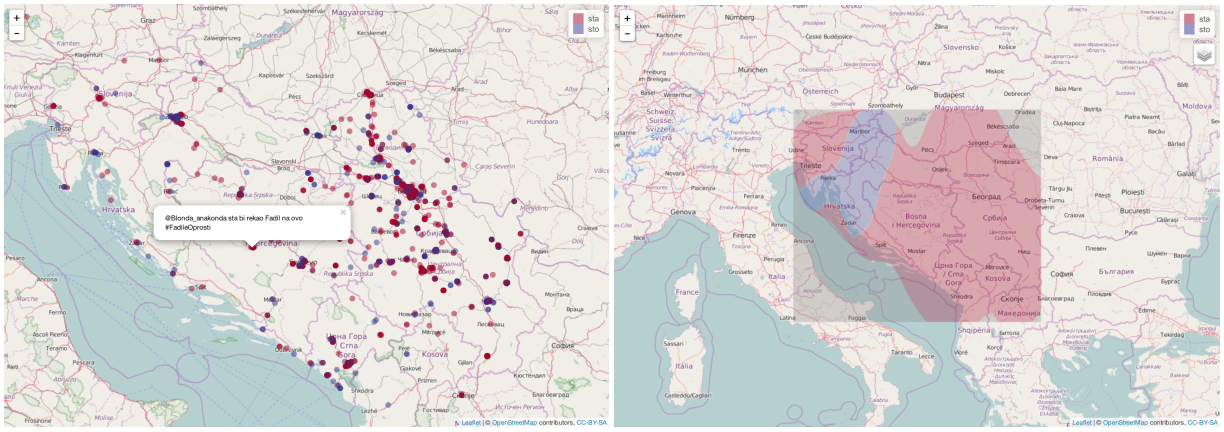


Figure 2: Result of point visualisation (left) and dominance map (right) for the variable *štošta*

candidate features will be passed forward to the dominance map tool, described in the next section.

5.3 Dominance Maps

The dominance map functionality provides the means for calculating continuous surfaces from point observations, in our case georeferenced tweets. Surfaces are calculated for each feature level separately, using kernel density estimation (KDE), a well established method for representing point observations as density surfaces. The local value of a density surface represents the number of observations of the respective feature level proximate to this location. A kernel function is applied for smoothing the signal and to thus account for local noise. The application of KDE to linguistic data is well represented in literature, e.g. (Bart et al., 2013). After computing density surfaces for each feature level individually, local intensities are compared and only the level with maximum local intensity is preserved and mapped as the dominant level. Hence, the dominance map function visually represents linguistic areas *dominated* by individual feature levels.

The two variables presented via point visualisation on the left side of Figure 1 and Figure 2 have their dominance maps depicted on the right side of the respective figures. While for the *yat* variable the point visualisation was already informative, due to a strong spatial trend as reported in Table 2, for the *štošta* variable, showing a weaker spatial trend, the visual identification of regionally dominant levels turns out to be considerably difficult. Therefore the dominance map comes in very handy, showing that only in central Croatia the *što* level is dominant while in the rest of the study area *šta* dominates.

For both variables the dominance map shows that the value dominant in Bosnia and Serbia is also dominant in Slovenia. This is due to large Bosnian and Serbian communities living in this country.

The results of the analysis of the *štošta* variable will benefit from more extensive data collection as the point visualisation shows that the areas of Croatia and Bosnia are only sparsely covered. We would therefore like to emphasize the preliminary nature of these results.

6 Conclusion and Future Work

In this paper, we have presented a configurable, flexible tool set for working with geo-encoded linguistic data automatically collected from Twitter. We have demonstrated through a use case how the tool facilitates to monitor language use in a region of interest. We have extracted a sample of five features with varied linguistic properties (phonetic, lexical, syntactic) and analysed their spatial distribution using spatial methods of varied complexity.

The results of our initial analyses indicate that the flexibility offered by our tool is important for gaining important insights into the data: higher level analyses and visualisation can reveal patterns not visible in a simpler point visualisation (as in the case of the *štošta* variable). On the other hand, point visualisation can serve as a good tool for manual checkups of the reliability of both the extracted data and higher-level analyses. With the possibility to obtain different representations quickly and in a relatively simple

way, researchers can use our tool to address important questions regarding geographical distributions of linguistic features. As the tool is written in popular languages, it is also easy to extend.

In future work, we will use the presented tool to perform further analyses of the language use in the region addressed in our initial study. We will address some of the key points in the ongoing debate about the differences between Bosnian, Croatian, Montenegrin and Serbian and the potential process of linguistic separation.

Furthermore we will continue extending the presented TweetGeo tool with additional analysis capabilities, as well as work on merging the tool with previously developed tools – TweetCat[10] (Ljubešić et al., 2014) which focuses on data acquisition through the Twitter Search API, and TweetPub[11] which is meant for preparing linguistically annotated Twitter collections for publishing while following the Twitter Developer agreement – into a unified toolkit for gathering, analysing and redistributing Twitter data.

While the presented tool is designed for spatial linguistic analysis, we would argue that it is also suited for studying the spatial distribution of other phenomena that can be studied using Twitter and associated metadata. Examples are demographic characteristics, relations between the speakers or particular ways of using the social network. We would therefore argue that without major changes, our tool could be applied to the broader context of the humanities.

Acknowledgements

We would like to thank our collaborator Dolores Batinić for her help in the linguistic part of our work. The work presented in this paper was supported by the URPP 'Language and Space' individual grant and by the Swiss National Science Foundation grant No. 160501.

References

Gennady Andrienko, Natalia Andrienko, Urska Demsar, Doris Dransch, Jason Dykes, Sara Irina Fabrikant, Mikael Jern, Menno-Jan Kraak, Heidrun Schumann, and Christian Tominski. 2010. Space, time and visual analytics. *International Journal of Geographical Information Science*, 24(10):1577–1600.

Gabriela Bart, Robert Weibel, Pius Sibler, and Elvira Glaser. 2013. Analysis of Swiss German syntactic variants using spatial statistics. *Álvarez Pérez, Xosé Afonso, Ernestina Carrilho & Catarina Magro (red.). Current Approaches to Limits and Areas in Dialectology. Newcastle upon Tyne: Cambridge Scholars Publishing*.

Stefan Bauernschuster, Oliver Falck, Stephan Heblich, Jens Suedekum, and Alfred Lameli. 2014. Why are educated and risk-loving persons more mobile across regions? *Journal of Economic Behavior & Organization*, 98:56 – 69.

Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. No country for old members: User lifecycle and linguistic change in online communities. In *Proceedings of WWW*.

Peter J . Diggle. 2014. *Statistical Analysis of Spatial and Spatio-Temporal Point Patterns, Third Edition*. Chapman and Hall/CRC 2013.

Gabriel Doyle. 2014. Mapping dialectal variation by querying social media. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 98–106, Gothenburg, Sweden, April. Association for Computational Linguistics.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Michael Dunn, Simon J. Greenhill, Stephen C. Levinson, and Russell D. Gray. 2011. Evolved structure of language shows lineage-specific trends in word-order universals. *Nature*, pages 79–82.

Jacob Eisenstein, Noah A. Smith, and Eric P. Xing. 2011. Discovering sociolinguistic associations with structured sparsity. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1365–1374, Portland, Oregon, USA, June. Association for Computational Linguistics.

[10] https://github.com/clarinsi/tweetcat
[11] https://github.com/clarinsi/tweetpub

Stefan Hahmann, Ross S Purves, and Dirk Burghardt. 2014. Twitter location (sometimes) matters: Exploring the relationship between georeferenced tweet content and nearby feature classes. *Journal of Spatial Information Science*, 2014(9):1–36.

Brent Hecht and Monica Stephens. 2014. A tale of cities: Urban biases in volunteered geographic information. *ICWSM*, 14:197–205.

Dirk Hovy and Anders Johannsen. 2016. Exploring language variation across Europe - a web-based tool for computational sociolinguistics. In Nicoletta Calzolari et al., editor, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Kalev Leetaru, Shaowen Wang, Guofeng Cao, Anand Padmanabhan, and Eric Shook. 2013. Mapping the global twitter heartbeat: The geography of Twitter. *First Monday*, 18(5).

Nikola Ljubešić, Darja Fišer, and Tomaž Erjavec. 2014. TweetCaT: a Tool for Building Twitter Corpora of Smaller Languages. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Nikola Ljubešić, Filip Klubička, Željko Agić, and Ivo-Pavao Jazbec. 2016. New Inflectional Lexicons and Training Corpora for Improved Morphosyntactic Annotation of Croatian and Serbian. In Nicoletta Calzolari et al., editor, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *ACL (System Demonstrations)*, pages 25–30.

Emira Mešanović Meša. 2011. *Kontrasivna analizabosanskog, hrvatskog i srpskog jezika u zakonima Federacije Bosne i Hercegovine*. Slavistički komitet, Sarajevo.

P. A. P. Moran. 1950. Notes on continuous stochastic phenomena. *Biometrika*, 37(1/2):17–23.

John Nerbonne. 2009. Data-driven dialectology. *Language and Linguistics Compass*, page 175–198.

Dong Nguyen, Noah A. Smith, and Carolyn P. Rosé. 2011. Author age prediction from text using linear regression. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 115–123, Portland, OR, USA, June. Association for Computational Linguistics.

Benedikt Szmrecsanyi. 2012. Geography is overrated. In Sandra Hansen, Christian Schwarz, Philipp Stoeckle, and Tobias Streck, editors, *Dialectological and folk dialectological concepts of space*, pages 215–231, Berlin. de Gruyter.

Martijn Wieling and John Nerbonne. 2015. Advances in dialectometry. *Annual Review of Linguistics*, pages 243–264.

Martijn Wieling, John Nerbonne, and R. Harald Baayen. 2011. Quantitative social dialectology: Explaining linguistic variation geographically and socially. *PLoS ONE*, 6(9):1–14, 09.

Extending WordNet with Fine-Grained Collocational Information via Supervised Distributional Learning

Luis Espinosa-Anke[1], **Jose Camacho-Collados**[2], **Sara Rodríguez-Fernández**[1], **Horacio Saggion**[1], **Leo Wanner**[3,1]
[1]NLP Group, Universitat Pompeu Fabra
firstname.lastname@upf.edu
[2]Department of Computer Science, Sapienza University of Rome
collados@di.uniroma1.it
[3]Catalan Institute for Research and Advanced Studies (ICREA)

Abstract

WordNet is probably the best known lexical resource in Natural Language Processing. While it is widely regarded as a high quality repository of concepts and semantic relations, updating and extending it manually is costly. One important type of relation which could potentially add enormous value to WordNet is the inclusion of collocational information, which is paramount in tasks such as Machine Translation, Natural Language Generation and Second Language Learning. In this paper, we present ColWordNet (CWN), an extended WordNet version with fine-grained collocational information, automatically introduced thanks to a method exploiting linear relations between analogous sense-level embeddings spaces. We perform both intrinsic and extrinsic evaluations, and release CWN for the use and scrutiny of the community.

1 Introduction

The embedding of cues about how we perceive concepts and how these concepts relate and generalize across different domains gives knowledge resources the capacity of generalization, which lies at the core of human cognition (Yu et al., 2015) and is also central to many Natural Language Processing (NLP) applications (Jurgens and Pilehvar, 2015). It is general practice to identify and formalize conceptual relations using a reference knowledge repository. As such a repository, WordNet (Miller et al., 1990) stands out as the *de facto* standard lexical database, containing over 200k English senses with 155k word forms. Over the years, WordNet has become the cornerstone of agglutinative resources such as BabelNet (Navigli and Ponzetto, 2012) and Yago (Suchanek et al., 2007). It is also used in semantically intensive tasks such as Word Sense Disambiguation (Navigli, 2009), Query Expansion and IR (Fang, 2008), Sentiment Analysis (Esuli and Sebastiani, 2006), semantic similarity measurement (Pilehvar et al., 2013), development and evaluation of word embeddings models (Huang et al., 2012; Faruqui et al., 2015), and Taxonomy Learning Evaluation (Bordea et al., 2015).

While the value of WordNet for NLP is indisputable, it is generally recognized that enriching it with additional information makes it an even more valuable resource. Thus, there is a line of research aimed at extending it with novel terminology (Jurgens and Pilehvar, 2016), cross-predicate relations (Lopez de la Calle et al., 2016), and so forth. Nonetheless, one type of information has been largely neglected so far: collocations, i.e., idiosyncratic binary lexical co-occurrences. As a standalone research topic, however, collocations have been the focus of a substantial amount of work, e.g. for automatically retrieving them from corpora (Choueka, 1988; Church and Hanks, 1989; Smadja, 1993; Kilgariff, 2006; Evert, 2007; Pecina, 2008; Bouma, 2010; Gao, 2013), and for their semantic classification according to different typologies (Wanner et al., 2006; Gelbukh and Kolesnikova., 2012; Moreno et al., 2013; Wanner et al., 2016). However, to the best of our knowledge, no previous work attempted the automatic enrichment of WordNet with collocational information. The only related attempt consisted in designing a schema for

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3422–3432, Osaka, Japan, December 11-17 2016.

the manual inclusion of lexical functions from Explanatory Combinatorial Lexicology (ECL) (Mel'čuk, 1996) into the Spanish EuroWordNet (Wanner et al., 2004).

Given the importance of collocations for a series of NLP applications (e.g. machine translation, text generation or paraphrasing), we propose to fill this gap by putting forward a new methodology which exploits intrinsic properties of state-of-the-art semantic vector space models and leverages the transformation matrix introduced by Mikolov et al. (2013b) in a word-level machine translation task. As a result, we release an extension of WordNet with detailed collocational information, named ColWordNet (CWN). This extension is carried out by means of the inclusion of *novel edges*, where each edge encodes a *collocates-with* relation, as well as the semantics of the collocation itself. For example, given the pair of synsets `desire.n.01` and `ardent.a.01`, a novel relation $\xrightarrow[x]{col:intense}$ is introduced, where 'intense' is the *semantic category* denoting *intensification*, and x is the confidence score assigned by our algorithm.

The remainder of the paper is organized as follows: In Section 2, we provide some background on collocations and the vector space models on which we base our approach. Section 3 describes the methodology followed to construct CWN. Then, Section 4 presents both intrinsic and extrinsic experimental results. And, finally, Section 5 summarizes the main contributions of our paper and outlines potential avenues for future work.

2 Background

In what follows, we first present relevant background on the semantic categories of collocations we use in our work (Section 2.1) and then on the resources used in our experiments (Section 2.2).

2.1 Collocations

Collocations are restricted lexical co-occurrences of two syntactically related lexical items, the base and the collocate. In a collocation, the base is freely chosen by the speaker, while the choice of the collocate depends on the base; see, e.g., (Cowie, 1994; Mel'čuk, 1996; Kilgariff, 2006) for a theoretical discussion. For instance, in the collocations *take* [*a*] *step*, *solve* [*a*] *problem*, *pay attention*, *deep sorrow*, and *strong tea*, *step*, *problem*, *attention*, *sorrow* and *tea* are the bases and *take*, *solve*, *pay*, *deep* and *strong* their respective collocates.

Besides a syntactic dependency, between the base and the collocate a semantic relation holds. Some of these semantic relations, such as 'intense', 'weak', 'perform', 'cause', etc. can be found across a large number of collocations. For instance, an 'intense' *applause* is a *thundering applause*, an 'intense' *emotion* is *deep*, 'intense' *rain* is *heavy*, and so on. In our experiments, we focused on the subset of the most prominent eight semantic collocation relations (or categories), which are listed in the first column of Table 1. These semantic categories are a generalization of the *lexical functions* (LFs) from ECL already used in Wanner et al. (2004). We have decided to use somewhat more generic categories instead of LFs because, on the one hand, some of the LFs differ only in terms of their syntactic structure (i.e. they capture the same semantic relation), and, on the other hand, LFs pose a great challenge for annotation due to their syntactic granularity.

2.2 Resources

The CWN lexical database is generated thanks to the exploitation of word and sense-based vector space models stemming from BabelNet (Navigli and Ponzetto, 2012),[1] which currently constitutes the largest semantic repository of both concepts and named entities.[2] In BabelNet, just like in WordNet, concepts are represented as *synsets* (i.e., set of synonym senses[3]). This allows us to exploit BabelNet's direct mapping with WordNet so that when our algorithm yields a candidate collocate $synset^{n}_{bn}$, we may retrieve

[1] `http://babelnet.org/`

[2] In its 3.6 release version, BabelNet is composed of 6.1M concepts and 7.7M named entities.

[3] For example, the concept defined as *principal activity in your life that you do to earn money* is represented by the synset {*occupation, business, job, line of work, line*}, where *occupation*, *business*, *job*, *line of work*, and *line* are senses/lexicalizations of the given synset.

its corresponding $synset^n_{wn}$, provided there exists one. In what follows, we briefly describe two different vector space models that are used in this paper for the task of synset-level collocation discovery.

SENSEMBED[4] (Iacobacci et al., 2015) is a knowledge-based approach for obtaining latent continuous representations of individual word senses based on Word2Vec (Mikolov et al., 2013a). Unlike other sense-based embeddings approaches, such as, e.g., Huang et al. (2012), which address the inherent polysemy of word-level representations relying solely on text corpora, SENSEMBED exploits the structured knowledge of BabelNet along with distributional information gathered from the Wikipedia corpus. In this paper, we used SENSEMBED for automatically disambiguating our training data, and as our **bases model**.

SHAREDEMBED. For this model we exploit distributional information from a 3B-word corpus extracted from the web (Han et al., 2013),[5] arguably richer in collocations than the encyclopedic style of Wikipedia. Similarly to SENSEMBED, this model is based on a pre-disambiguation of text corpora using BabelNet as sense inventory. However, unlike SENSEMBED, which learns vector representations for individual word senses, for this model we are interested in obtaining fine-grained information in the form of both plain text words and synsets[6] in a shared vector space (see Section 3.2 for the motivation behind this choice, and its application). To this end, we follow Chen et al. (2014) and modify the objective function of Word2Vec,[7] so that words and synsets can be learned jointly in a single training. The output is a vector space of word and synset embeddings that we use as **collocates model**.

3 Methodology

In this section, we provide a detailed description of the algorithm behind the construction of CWN. The system takes as input the WordNet lexical database and a set of collocation lists pertaining to predefined semantic categories, and outputs CWN. First, we collect training data and perform automatic disambiguation (Section 3.1). Then, we use this disambiguated data for training a linear *transformation matrix* from the base vector space, i.e., SENSEMBED, to the collocate vector space, i.e., SHAREDEMBED (Section 3.2). Finally, we exploit the WordNet taxonomy to select input base collocates to which we apply the transformation matrix in order to obtain a sorted list of candidate collocates (Section 3.3).

3.1 Collecting and Disambiguating Training Data

As is common in previous work on semantic collocation classification (Moreno et al., 2013; Wanner et al., 2016), our training set consists of a list of manually annotated collocations. For this purpose, we randomly selected nouns from the Macmillan Dictionary and manually classified their corresponding collocates with respect to their semantic categories.[8] Note that there may be more than one collocate for each base. Since collocations with different collocate meanings are not evenly distributed in language (e.g., we may tend to use more often collocations conveying the idea of 'intense' and 'perform' than 'begin to perform'), the number of instances per category in our training data also varies significantly (see Table 1).

Our training dataset consists at this stage of pairs of plain words, with the inherent ambiguity this gives raise to. We surmount this challenge by applying a disambiguation strategy based on the notion that, from all the available senses for a collocation's base and collocate, their correct senses are those which are most similar. This is a strategy that has been proved effective in previous concept-level disambiguation tasks (Delli Bovi et al., 2015). Formally, let us denote the SENSEMBED vector space as $\mathcal{S}$, and our original text-based training data as $\mathbf{T}$. For each training collocation $\langle b, c \rangle \in \mathbf{T}$ we consider all the available lexicalizations (i.e., senses) for both the base b and the collocate c in $\mathcal{S}$, namely $L_b = \{l^1_b ... l^n_b\}$, and $L_c = \{l^1_c ... l^m_c\}$, and their corresponding set of sense embeddings $\mathbf{V}_b = \{\vec{v}^1_b, ..., \vec{v}^n_b\}$ and $\mathbf{V}_c =$

[4] We downloaded the pre-trained sense embeddings at `http://lcl.uniroma1.it/sensembed/`.

[5] `ebiquity.umbc.edu/blogger/2013/05/01/umbc-webbase-corpus-of-3b-english-words/`

[6] As explained above, a synset is a set composed of synonym senses.

[7] We used the Continous Bag-Of-Words (CBOW) model with standard hyperparameters: 300 dimensions and a window size of 8 words.

[8] We do not consider phrasal verb collocates, e.g. *stand up*, *give up* or *calm down*.

Sem. Category	Example	# instances
'intense'	*absolute certainty*	586
'weak'	*remote chance*	70
'perform'	*give chase*	393
'begin to perform'	*take up a chase*	79
'increase'	*improve concentration*	73
'decrease'	*limit* [*a*] *choice*	73
'create', 'cause'	*pose* [*a*] *challenge*	195
'put an end'	*break* [*the*] *calm*	79

Table 1: Semantic categories and size of training set

$\{\vec{v}_c^1, ..., \vec{v}_c^m\}$. Our aim is to select, among all possible pairs of senses, the pair $\langle l'_b, l'_c \rangle$ that maximizes the cosine similarity between the corresponding embeddings v'_b and v'_c, which is computed as follows:

$$\langle \vec{v}'_b, \vec{v}'_c \rangle = \text{argmax}_{\vec{v}_b \in \mathbf{V}_b,\, \vec{v}_c \in \mathbf{V}_c} \frac{\vec{v}_b \cdot \vec{v}_c}{\|\vec{v}_b\| \, \|\vec{v}_c\|} \quad (1)$$

Our disambiguation strategy yields a set of disambiguated pairs **D**. This is the input for the next module of the pipeline, the learning of a *transformation matrix* aimed at retrieving WordNet synset collocates for any given WordNet synset base.

3.2 Training a Sense-Level Transformation Matrix for each Semantic Category

Among the many properties of word embeddings (Mikolov et al., 2013a; Mikolov et al., 2013c) that have been explored so far in the literature (e.g., modeling analogies or projecting similar words nearby in the vector space), the most pertinent to this work is the linear relation that holds between semantically similar words in two analogous spaces (Mikolov et al., 2013b). Mikolov et al.'s original work learned a linear projection between two monolingual embeddings models to train a word-level machine translation system between English and Spanish. Other examples include the exploitation of this property for language normalization, i.e. finding *regular English* counterparts of Twitter language (Tan et al., 2015), or hypernym discovery (Espinosa-Anke et al., 2016).

In our specific case, we learn a linear transformation from $\vec{v}'_b$ to $\vec{v}'_c$, aiming at reflecting an inherent condition of collocations. Since collocations are a linguistic phenomenon that is more frequent in the narrative discourse than in formal essays, they are less likely to appear in an encyclopedic corpus (recall that SENSEMBED vectors, which we use, are trained on a dump of the English Wikipedia). This motivates the use of $\mathcal{S}$ as our *base space*, and our SHAREDEMBED $\mathcal{X}$ as the *collocate model*, as it was trained over more varied language such as blog posts or news items.

Then, we construct our linear transformation model as follows: For each disambiguated collocation $\langle l'_b, l'_c \rangle \in \mathbf{D}$, we first retrieve the corresponding base vectors $\vec{v}'_b$. Next, we exploit the fact that $\mathcal{X}$ contains both BabelNet synsets and words, and derive for each l'_c two items, namely the vectors associated to its lexicalization (word-based) and its BabelNet synset. For example, for the training pair $\langle$`ardent_bn:00097467a`, `desire_bn:00026551n`$\rangle \in \mathbf{D}$, we learn two linear mappings, namely `ardent_bn:00097467a` $\mapsto$ `desire` and `ardent_bn:00097467a` $\mapsto$ `bn:00026551n`. We opt for this strategy, which doubles the size of the training data in most lexical functions (depending on coverage), due to the lack of resources of manually-encoded classification of collocations. By following this strategy we obtain an extended training set $\mathbf{D}* = \{\vec{b}_i, \vec{c}_i\}_{i=1}^n$ ($\vec{b}_i \in \mathcal{X}$, $\vec{c}_i \in \mathcal{S}, n \geq |\mathbf{D}|$). Then, we construct a *base matrix* $\mathbf{B} = \left[\vec{b}_1 \dots \vec{b}_n\right]$ and a collocate matrix $\mathbf{C} = [\vec{c}_1 \dots \vec{c}_n]$ with the resulting set of training vector pairs. We use these matrices to learn a linear transformation matrix $\Psi \in \mathbb{R}^{d_\mathcal{S} \times d_\mathcal{X}}$, where $d_\mathcal{S}$ and and $d_\mathcal{X}$ are, respectively, the number of dimensions of the base vector space (i.e., SENSEMBED) and the collocate vector space (SHAREDEMBED).[9] Following the notation in Tan et al. (2015), this transformation can be depicted as:

[9] In our setting the numbers of dimensions are $d_\mathcal{S} = 400$ and $d_\mathcal{X} = 300$.

$$\mathbf{B}\Psi \approx \mathbf{C}$$

As in Mikolov et al.'s original approach, the training matrix is learned by solving the following optimization problem:

$$\min_{\Psi} \sum_{i=1}^{n} \|\vec{b}_i - \vec{c}_i\|^2$$

Having trained Ψ, the next step of the pipeline is to apply it over a subset of WordNet's base concepts and their hyponyms. For each synset in this branch, we apply a scoring and ranking procedure which assigns a *collocates-with* score. If such score is higher than a predefined threshold, tuned over a development set, this relation is included in CWN.

3.3 Retrieving and Sorting WordNet Collocate Synsets

During the task of enriching WordNet with collocational information, we first gather a set of base WordNet synsets by traversing WordNet hypernym hierarchy starting from those base concepts that are most fit for the input semantic category.[10] Then, the *transformation matrix* learned in Section 3.2 is used to find candidate WordNet synset collocates (mostly verbs or adjectives) for each base WordNet synset.

As explained in Section 3, WordNet synsets are mapped to BabelNet synsets, which in turn map to as many vectors in SENSEMBED as their associated lexicalizations. Formally, given a base synset b, we apply the transformation matrix to all the SENSEMBED vectors $\mathbf{V}_b = \{\vec{v}_b^1, ..., \vec{v}_b^n\}$ associated with its lexicalizations. For each $\vec{v}_b^i \in \mathbf{V}_b$, we first get the vector $\vec{\psi}_b^i = \vec{v}_b^i\Psi$ obtained as a result of applying the transformation matrix and then we gather the subset $W_b^i = \{\vec{w}_b^{i,1} \dots \vec{w}_b^{i,10}\}$ ($\vec{w}_b^{i,j} \in \mathcal{X}$) of the top ten closest vectors by cosine similarity to $\vec{\psi}_b^i$ in the SHAREDEMBED vector space $\mathcal{X}$. Each $\vec{w}_b^{i,j}$ is ranked according to a scoring function $\lambda(\cdot)$, which is computed as follows[11]: $\lambda(\vec{w}_b^{i,j}) = \frac{cos(\vec{\psi}_b^i, \vec{w}_b^{i,j})}{j}$. This scoring function takes into account both the cosine similarity as well as the relative position[12] of the candidate collocate with respect to other neighbors in the vector space. Apart from sorting the list of candidate collocates, this scoring function is also used to measure the confidence of the retrieved collocate synsets in CWN.

4 Evaluation

We evaluate CWN both intrinsically and extrinsically. Our intrinsic evaluation consists of a manual scoring of the correctness of the newly introduced relations (Section 4.1). Extrinsic evaluation assesses the quality of CWN as an input resource for introducing collocational information into a word embeddings model (Section 4.2).

4.1 Intrinsic: Precision of Collocate Relations

Sampling and evaluation are carried out as follows. First, for each semantic category, we retrieve 50 random bases included in the aforementioned base concepts (see Section 3.3) and all their hyponym branch. This results in an evaluation set $Test$ of 800 collocations, as for each base we retrieve the 5 highest scoring candiates. These collocations are evaluated in terms of correctness, i.e., if the associated synset is an appropriate collocate for the input base. Note that not all bases in the test set may be suitable for the given semantic category, and that is why we also perform an evaluation on the test data restricted to only those bases manually selected for being suitable for having at least one collocate. We denote the restricted test data as $Test*$. For example, the base synset `putt.n.01` defined as *hitting a golf ball*

[10] These are: For 'intense' and 'weak', `attitude.n.01`, `feeling.n.01` and `ability.n.02`. For the rest of them, we select `cognition.n.01`, `act.n.02` and `action.n.01`.

[11] If $w_b^{i,j}$ appears in a different W_b^j set ($j \neq i$), its scores are averaged.

[12] Position is arguably an important factor as there may be dense areas where cosine similarity alone may not reflect entirely the fitness of a candidate.

	'intense'				**'perform'**				**'put an end'**				**'increase'**			
	Baseline		CWN		Baseline		CWN		Baseline		CWN		Baseline		CWN	
	Test	*Test**	*Test*	*Test**	*Test*	*Test**	*Test*	*Test**	*Test*	*Test**	*Test*	*Test**	*Test*	*Test**	*Test*	*Test**
P@1	0.00	0.00	0.35	**0.46**	0.15	0.16	0.20	**0.36**	0.05	0.08	0.15	**0.50**	0.05	0.14	0.15	**0.42**
P@5	0.03	0.30	0.43	**0.57**	0.06	0.06	0.13	**0.23**	0.02	0.03	0.12	**0.40**	0.04	0.11	0.18	**0.51**
MRR	0.05	0.41	0.48	**0.65**	0.18	0.19	0.32	**0.59**	0.07	0.12	0.20	**0.68**	0.07	0.21	0.22	**0.65**
MAP	0.05	0.45	0.48	**0.64**	0.15	0.18	0.32	**0.59**	0.07	0.12	0.19	**0.64**	0.07	0.20	0.22	**0.64**
	'decrease'				**'create/cause'**				**'weak'**				**'begin to perform'**			
	Baseline		CWN		Baseline		CWN		Baseline		CWN		Baseline		CWN	
	Test	*Test**	*Test*	*Test**	*Test*	*Test**	*Test*	*Test**	*Test*	*Test**	*Test*	*Test**	*Test*	*Test**	*Test*	*Test**
P@1	0.00	0.00	0.30	**0.46**	0.05	0.16	0.10	**0.50**	0.00	0.00	0.10	**0.22**	0.00	0.00	0.00	0.00
P@5	0.02	0.03	0.19	**0.29**	0.04	0.13	0.04	**0.20**	0.02	0.03	0.04	**0.08**	0.03	0.07	0.02	**0.20**
MRR	0.02	0.04	0.39	**0.61**	0.07	0.25	0.10	**0.50**	0.03	0.04	0.01	**0.22**	0.05	0.12	0.04	**0.41**
MAP	0.02	0.03	0.38	**0.58**	0.06	0.20	0.10	**0.50**	0.03	0.04	0.01	**0.22**	0.05	0.12	0.04	**0.41**

Table 2: Summary of the manual evaluation of the performance of CWN and of the baseline

that is on the green using a putter does not admit any 'decrease' collocate, and therefore its collocations are not considered in $Test*$.

Since our algorithm returns a list of candidate collocate synsets for an input base synset, the task naturally becomes that of a ranking problem, and therefore ranking metrics such as Precision@K (P@K), Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) are appropriate for evaluating this experiment. These measures provide insights on different aspects of the outcome of the task, e.g. how often valid collocates were retrieved in the first positions of the rank (MRR), and if there were more than one valid collocate, whether this set was correctly retrieved, (MAP and R-P)[13]. In Table 2 we provide a detailed summary of the performance of our system (CWN), as compared with a competitor unsupervised baseline which exploits word analogies (as in $\vec{man} - \vec{king} + \vec{woman} = \vec{queen}$). This baseline, which we deploy on the SHAREDEMBED space, takes as input a prototypical collocation of a given semantic category (e.g. *thunderous applause* for 'intense') and an input base, and collects the top 10 Nearest Neighbours (NNs) to the vector resulting of the aforementioned analogy operation. This approach was recently used in a similar setting (Rodríguez-Fernández et al., 2016). Due to the difficulty of the task, and the restriction it imposes for collocates to be disambiguated synsets rather than any text-based word, the unsupervised approach fails short when compared to our supervised method, which is capable to find more and better disambiguated collocates.

Note that for half of the semantic categories under evaluation, our approach correlated well with human judgement, with the highest ranking candidates being more often correct than those ranked lower. This is the case of 'put an end', 'decrease', 'create/cause' and 'weak'. In fact, it is in 'put an end', where our system achieves the highest MRR score, which we claim to be the most relevant measure, as it rewards cases where the first ranked returned collocation is correct without measuring in the retrieved collocates at other positions. Moreover, let us highlight the importance of two main factors. First, the need for a well-defined semantic relation between bases and collocates. It has been shown in other tasks that exploit linear transformations between embeddings models that even for one single relation there may be clusters that require certain specificity in the *domain* or *semantic* of the data (see Fu et al. Fu et al. (2014) for a discussion of this phenomenon in the task of taxonomy learning). Second, the importance of having a reasonable amount of training pairs so that the model can learn the idiosyncrasies of the semantic relation that is being encoded (e.g., Mikolov et al. (2013b) report a major increase in performance as training data increases in several orders of magnitude). This is reinforced in our experiments, where we obtain the highest MAP score for 'intense', the semantic category for which we have the largest training data available.

[13] See Bian et al. (2008) for an in-depth analysis of these metrics.

	'intense'			'weak'			'perform'			'create/cause'		
	correct	*dist.*	*diff.*	*correct*	dist.	*diff.*	*correct*	*dist.*	*diff.*	*correct*	*dist.*	*diff.*
original	0.22	0.04	+0.18	0.17	0.05	+0.12	0.15	0.05	+0.10	0.17	0.06	+0.11
retrofitted	0.27	0.06	**+0.21**	0.19	0.06	**+0.13**	0.25	0.11	**+0.14**	0.28	0.12	**+0.16**

Table 3: Comparison of collocational sensitivity between original and retrofitted embeddings models over four semantic categories.

4.2 Extrinsic evaluation: Retrofitting Vector Space Models to CWN

We complement our manual evaluation with an extrinsic experiment, where we assess the extent to which our newly generated lexical resource can be used to *introduce collocational sensitivity* to a generic word embeddings model.[14] To this end, we extract collocation clusters by extracting all the synsets associated lemmas (e.g. for *heavy.a.01 rain.n.01*, we would extract the cluster [*heavy*, *rain*, *rainfall*]). These are used as input for the Retrofitting Word Vectors algorithm (Faruqui et al., 2015).[15] This algorithm takes as input a vector space and a semantic lexicon which may encode any semantic relation, and puts closer in the vector space words that are related in the lexicon.

Previous approaches have encoded semantic relations by introducing some kind of bias into a vector space model (Yu et al., 2015; Pham et al., 2015; Mrkšić et al., 2016; Nguyen et al., 2016). For instance, Yu et al. (2015) encode (term, hypernym) relations by grouping together terms and their hypernyms, rather than semantically related items. In this way, their *biased* model puts closer to *jaguar* terms like *animal* or *vehicle*, while an unbiased model would put nearby terms such as *lion*, *bmw* or *jungle*. We aim at introducing a similar bias, but in terms of collocational information. This is achieved, for each lexical function and each synset in CWN-*st*, by obtaining its top 3 collocate candidates and incorporate information on their *collocationality* into the model.

4.2.1 Collocational Sensitivity

In this experiment, we assess the extent to which a retrofitted model with collocational bias is able to discriminate between a correct collocation and a random combination of the same base with an unrelated collocate. To this end, we manually constructed two datasets, one for *noun+adjective* ('intense' and 'weak' semantic categories) and one for *noun+verb* combinations, which we evaluate on the two most productive semantic categories, namely 'perform' and 'create/cause'. The datasets consist of 50 bases and one of their correct collocates according to the Macmillan Collocations Dictionary, accompanied by four *distractor* (*dist.* in Table 3) collocates. For instance, given the correct 'perform' collocation *make a pledge*, we expect our 'perform'-wise retrofitted model to increase the score in $\vec{make} + \vec{pledge}$ substantially more than a combination $\vec{pledge} + \vec{distractor}$. For each evaluated semantic category, we computed the average increase of the cosine similarity between all correct collocations and all distractors (*diff.* in Table 3). As shown in Table 3, there is a consistent increase over the four evaluated semantic categories, namely 'intense', 'weak', 'perform' and 'create/cause'. This proves the potential of our retrofitted model to discern between correct and wrong collocates. In the following section, we explore the possibility to use this vector space for finding collocates giving a base as input.

4.2.2 Exploring Nearest Neighbours for Collocate Discovery

Inspired by Yu et al.'s (2015) work on introducing hypernymic bias into a word embeddings model, we explore the extent to which our retrofitted models can be used to discover *alternative collocates* given the composition of the words involved in a collocation as input. In order to discover these collocates, we compose the base and the collocate by averaging their respective word embeddings and retrieve its closest words in the vector space according to cosine similarity. In Table 4, we show a sample of five NNs for several input adjective+noun collocations. These examples reveal how the vector space model retroffited using our collocations tends to bring closer in the space modifiers (i.e., collocates), providing

[14] We use the Google News pre-trained Word2Vec vectors, available at `code.google.com/archive/p/word2vec/`, as input for retrofitting.

[15] We used the code available at `https://github.com/mfaruqui/retrofitting`

	'intense'			'weak'	
	original	retrofitted		original	retrofitted
ferocious + hatred	**vicious**	fierce	dim +light	bright	**faint**
	fury	**fearsome**		dimmed	**unaccented**
	ferocity	fury		dimmer	**dense**
	savage	hate		dimming	bright
	hostility	**savage**		lights	centaur
intense + sympathy	fierce	**considerable**	mild + comment	milder	**modest**
	empathy	**tremendous**		NamedEntity	**meek**
	admiration	**enormous**		NamedEntity	NamedEntity
	anger	encouragement		NamedEntity	NamedEntity
	grudging respect	**immense**		NamedEntity	NamedEntity
sheer + delight	amazement	**immense**	modest + progress	progress	**mild**
	sheer unadulterated	**colossal**		pro gress	**meek**
	sheer joy	delectation		**Modest**	dissatisfaction
	joy	disgust		NamedEntity	progess
	astonishment	**stupendous**		strides	**slight**

Table 4: Comparison of the five NNs of six sample adj+noun collocations between a generic word embeddings model and a *retrofitted* version with semantic collocation information ('intense' and 'weak'). Note the increase in plausible collocates in retrofitted models (in bold). NamedEntity refers to noisy entities appearing among the top 5 NNs.

an interesting method for automatic collocation discovery. Despite its simplicity, this collocational discovery approach extracts a considerable amount of suitable fine-grained collocates for a given base. For example, given the collocation *intense sympathy*, the retrofitted space extracts *considerable*, *tremendous*, *enormous* and *immense* as candidate collocates of intensity among the five nearest neighbours. As future work we plan to further exploit and evaluate the impact of this property.

5 Conclusions and Future Work

We have described a system for an automatic enrichment of the WordNet lexical database with fine-grained collocational information, yielding a resource called ColWordNet (CWN). Our approach is based on the intuition that there is a linear transformation in vector spaces between bases and collocates of the same *semantic category*, e.g. between *heavy* and *rain*, or between *ardent* and *desire*. We have exploited sense-based embedding models to train an algorithm designed to retrieve valid collocates for a given input base. This pipeline is carried out at the *sense* level (rather than the word level), by leveraging models which use BabelNet as a reference sense inventory. We evaluated CWN both intrinsically and extrinsically, and verified that our algorithm is able to encode fine-grained *collocates-with* relations at synset level.

Release. We release CWN at several different confidence levels. The version with the highest confidence includes over 100k collocational edges, which connect over 8k unique base and collocate WordNet synsets. These connections are further enriched by two pieces of information, namely (1) the type of collocation (e.g. 'intense' or 'perform'), and (2) a confidence score derived from our approach. Moreover, in addition to CWN, we also release four modified versions of the well-known Word2Vec Google News vector space model, retrofitted with collocational information, which we constructed for the extrinsic evaluation of CWN. These models can be exploited both for assessing the correctness of a collocation and for the discovery of alternative collocates for a given collocation. Finally, we also make available the evaluation datasets built as part of the Collocational Sensitivity experiment. All data associated with this publication is publicly available at http://www.taln.upf.edu/colwordnet.

Future work. In the future, we plan to design a method to retrieve the best *bases* for a given semantic category, which would allow us not to rely on predefined manually built base concepts. Finally, we are currently investigating the potential of applying neural approaches recasting the task as a sequence classification problem for including collocational information in WordNet clusters.

Acknowledgements

This work was partially funded by the following projects: MULTISENSOR (FP7-ICT-610411), KRISTINA (H2020-645012-RIA), HARENES (FFI201130219-CO2-02), through a predoctoral grant (BES-2012-057036), the María de Maeztu Units of Excellence Program (MDM-2015-0502) and TUNER (TIN2015-65308-C5-5-R, MINECO/FEDER, UE). Jose Camacho-Collados is supported by a Google Doctoral Fellowship in Natural Language Processing.

References

Jiang Bian, Yandong Liu, Eugene Agichtein, and Hongyuan Zha. 2008. Finding the right facts in the crowd: factoid question answering over social media. In *Proceedings of the 17th international conference on World Wide Web*, pages 467–476. ACM.

Georgeta Bordea, Paul Buitelaar, Stefano Faralli, and Roberto Navigli. 2015. Semeval-2015 task 17: Taxonomy extraction evaluation (texeval). In *Proceedings of the SemEval workshop*.

G. Bouma. 2010. Collocation Extraction beyond the Independence Assumption. In *Proceedings of ACL, Short paper track*, pages 109–114, Uppsala, Sweden.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of EMNLP*, pages 1025–1035, Doha, Qatar.

Y. Choueka. 1988. Looking for Needles in a Haystack or Locating Interesting Collocational Expressions in Large Textual Databases. In *Proceedings of the RIAO*, pages 34–38.

K. Church and P. Hanks. 1989. Word Association Norms, Mutual Information, and Lexicography. In *Proceedings of ACL*, pages 76–83.

A. Cowie. 1994. Phraseology. In R.E. Asher and J.M.Y. Simpson, editors, *The Encyclopedia of Language and Linguistics, Vol. 6*, pages 3168–3171. Pergamon, Oxford.

Claudio Delli Bovi, Luis Espinosa Anke, and Roberto Navigli. 2015. Knowledge base unification via sense embeddings and disambiguation. In *Proceedings of EMNLP*, pages 726–736, Lisbon, Portugal.

Luis Espinosa-Anke, Jose Camacho-Collados, Claudio Delli Bovi, and Horacio Saggion. 2016. Supervised distributional hypernym discovery via domain adaptation. In *Proceedings of EMNLP*.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC*, volume 6, pages 417–422.

S. Evert. 2007. Corpora and Collocations. In A. Lüdeling and M. Kytö, editors, *Corpus Linguistics. An International Handbook*. Mouton de Gruyter, Berlin.

Hui Fang. 2008. A re-examination of query expansion using lexical resources. In *ACL*, volume 2008, pages 139–147.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*, pages 1606–1615.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of ACL*, volume 1.

Z.M. Gao. 2013. Automatic Identification of English Collocation Errors based on Dependency Relations. *Sponsors: National Science Council, Executive Yuan, ROC Institute of Linguistics, Academia Sinica NCCU Office of Research and Development*, page 550.

A. Gelbukh and O. Kolesnikova. 2012. *Semantic Analysis of Verbal Collocations with Lexical Functions*. Springer, Heidelberg.

Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. UMBC EBIQUITY-CORE: Semantic textual similarity systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, volume 1, pages 44–52.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*, pages 873–882, Jeju Island, Korea.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning Sense Embeddings for Word and Relational Similarity. In *Proceedings of ACL*, pages 95–105, Beijing, China.

David Jurgens and Mohammad Taher Pilehvar. 2015. Reserating the awesometastic: An automatic extension of the wordnet taxonomy for novel terms. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies, Denver, CO*, pages 1459–1465.

David Jurgens and Mohammad Taher Pilehvar. 2016. SemEval-2016 Task 14: Semantic taxonomy enrichment. *Proceedings of SemEval*, pages 1092–1102.

Adam Kilgariff. 2006. Collocationality (and How to Measure it). In *Proceedings of the Euralex Conference*, pages 997–1004, Turin, Italy. Springer-Verlag.

Maddalen Lopez de la Calle, Itziar Aldabe, Egoitz Laparra, and German Rigau. 2016. Predicate Matrix. Atomatically extending the semantic interoperability between predicate resources. *Language Resources and Evaluation*, 50(2):263–289.

I.A. Mel'čuk. 1996. Lexical Functions: A tool for the description of lexical relations in the lexicon. In L. Wanner, editor, *Lexical Functions in Lexicography and Natural Language Processing*, pages 37–102. Benjamins Academic Publishers, Amsterdam.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for Machine Translation. *arXiv preprint arXiv:1309.4168*.

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

George A. Miller, R.T. Beckwith, Christiane D. Fellbaum, D. Gross, and K. Miller. 1990. WordNet: an online lexical database. *International Journal of Lexicography*, 3(4):235–244.

Pol Moreno, Gabriela Ferraro, and Leo Wanner. 2013. Can we determine the semantics of collocations without using semantics? In I. Kosem, J. Kallas, P. Gantar, S. Krek, M. Langemets, and M. Tuulik, editors, *Proceedings of the eLex 2013 conference*, Tallinn & Ljubljana. Trojina, Institute for Applied Slovene Studies & Eesti Keele Instituut.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. *Artificial Intelligence*, 193:217–250.

Roberto Navigli. 2009. Word Sense Disambiguation: A Survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating Distributional Lexical Contrast into Word Embeddings for Antonym-Synonym Distinction. In *Proc. of ACL*, pages 454–459.

Pavel. Pecina. 2008. A Machine Learning Approach to Multiword Expression Extraction. In *Proceedings of the LREC 2008 Workshop Towards a Shared Task for Multiword Expressions (MWE 2008)*, pages 54–57, Marrakech.

Nghia The Pham, Angeliki Lazaridou, and Marco Baroni. 2015. A Multitask Objective to Inject Lexical Contrast into Distributional Semantics. In *Proceedings of ACL*, pages 21–26.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, Disambiguate and Walk: A Unified Approach for Measuring Semantic Similarity. In *Proceedings of ACL*, pages 1341–1351.

Sara Rodríguez-Fernández, Roberto Carlini, Luis Espinosa-Anke, and Leo Wanner. 2016. Example-based Acquisition of Fine-grained Collocation Resources. In *Proceedings of LREC*, Portoroz, Slovenia.

Frank Smadja. 1993. Retrieving Collocations from Text: X-Tract. *Computational Linguistics*, 19(1):143–177.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *WWW*, pages 697–706. ACM.

Luchen Tan, Haotian Zhang, Charles L.A. Clarke, and Mark D. Smucker. 2015. Lexical Comparison Between Wikipedia and Twitter Corpora by Using Word Embeddings. In *Proceedings of ACL (2)*, pages 657–661, Beijing, China, July.

Leo Wanner, Margarita Alonso Ramos, and Antonia Martí. 2004. Enriching the Spanish EuroWordNet by Collocations. In *LREC*.

Leo Wanner, Bernd Bohnet, and Mark Giereth. 2006. Making Sense of Collocations. *Computer Speech and Language*, 20(4):609–624.

Leo Wanner, Gabriela Ferraro, and Pol Moreno. 2016. Towards Distributional Semantics-based Classification of Collocations for Collocation Dictionaries. *International Journal of Lexicography, doi:10.1093/ijl/ecw002*.

Zheng Yu, Haixun Wang, Xuemin Lin, and Min Wang. 2015. Learning Term Embeddings for Hypernymy Identification. In *Proceedings of IJCAI*, pages 1390–1397.

A News Editorial Corpus for Mining Argumentation Strategies

Khalid Al-Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen, Benno Stein
Faculty of Media, Bauhaus-Universität Weimar, Germany
<firstname>.<lastname>@uni-weimar.de

Abstract

Many argumentative texts, and news editorials in particular, follow a specific strategy to persuade their readers of some opinion or attitude. This includes decisions such as when to tell an anecdote or where to support an assumption with statistics, which is reflected by the composition of different types of argumentative discourse units in a text. While several argument mining corpora have recently been published, they do not allow the study of argumentation strategies due to incomplete or coarse-grained unit annotations. This paper presents a novel corpus with 300 editorials from three diverse news portals that provides the basis for mining argumentation strategies. Each unit in all editorials has been assigned one of six types by three annotators with a high Fleiss' κ agreement of 0.56. We investigate various challenges of the annotation process and we conduct a first corpus analysis. Our results reveal different strategies across the news portals, exemplifying the benefit of studying editorials—a so far underresourced text genre in argument mining.

1 Introduction

News editorials define a genre of written argumentative discourse whose main goal is persuasiveness. In a news editorial, an author states and defends a thesis that conveys his or her stance on a controversial topic usually related to the public interest. Editorials do not only aim to persuade readers of some opinion, but they often also propagate particular ideologies or recommend certain attitudes to the community, e.g., a specific action towards an upcoming event (van Dijk, 1992). To achieve persuasion, a news editorial follows a particular *argumentation strategy* that the author expects to be most suitable for the target audience, i.e., the author composes a series of claims, assumptions, and different types of evidence while using argumentative language and structure (van Dijk, 1995). This does not only cover the resort to quantitative features of text (e.g., related to lexical style, cohesion, or rhetorical structure), but it also refers to the roles, positions, flows, and relations of specific argumentative discourse units.

The study of news editorials is beneficial for several tasks, such as the creation of persuasive writing strategies for writing assistance systems and qualitative media content analysis. At the same time, the rapid expansion of online news portals increases the need for algorithms that can analyze an editorial's discourse *automatically*. The needed analyses include argumentation mining and evidence detection, both of which are studied in computational argumentation, an emergent area of computational linguistics. While several text corpora for such analyses have recently been published for different domains and genres, a respective resource with news editorials is missing to this day. Moreover, existing corpora stick to coarse-grained and/or incomplete annotations of the units of an argumentative discourse (see Section 2 for details), which renders the mining of an author's argumentation strategy impossible.

In this paper, we present a novel corpus with 300 news editorials evenly selected from three diverse online news portals: *Al Jazeera*, *Fox News*, and *The Guardian*. The aim of the corpus is to study (1) the mining and classification of fine-grained types of argumentative discourse units and (2) the analysis of argumentation strategies pursued in editorials to achieve persuasion. To this end, each editorial contains manual type annotations of all units that capture the role that a unit plays in the argumentative discourse,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3433–3443, Osaka, Japan, December 11-17 2016.

Figure 1: Example news editorial from the presented corpus. Each argumentative discourse unit of all 300 editorials in the corpus has been manually assigned one of six types, four of which are shown here.

such as *assumption* or *statistics*. The corpus consists of 14,313 units of six different types, each annotated by three professional annotators from the crowdsourcing platform *upwork.com*. Figure 1 shows the type annotations of one editorial in the corpus. The editorial has been taken from The Guardian.

Based on the results of the annotation process, we analyze the agreement between annotators in order to scrutinize the major cases of disagreement as well as to designate the complex issues that humans face in classifying types of argumentative discourse units in editorials. Considering the number and complexity of the types, the obtained inter-annotator agreement of 0.56 in terms of Fleiss' κ can be seen as high. In a first brief statistical analysis of the corpus, we investigate differences in the type distribution between the three portals, which indicate divergent argumentation strategies.

To conclude, the contribution of this paper is three-fold: (1) We propose the first annotation scheme for mining argumentation strategies that captures the type of each unit of an argumentative text. Unlike previous schemes, it relies on a complete annotation of the text on a fine-grained level. (2) We introduce a novel corpus with 300 news editorials which are manually annotated according to the scheme. Despite the resort to a so far under-resourced text type, the corpus contains a considerably larger number of unit annotations than comparable existing corpora. In addition, we provide a detailed analysis of the inter-annotator agreement and of the reliability of the resulting annotations. The corpus is freely available in order to foster research on computational argumentation.[1] (3) We present insightful findings that indicate the potential of our corpus for analyzing the argumentation strategies of news editorials. As far as we know, our corpus is the first which allows studying such strategies in monological opinionated text.

2 Related Work

Many recent publications in the area of computational argumentation are concerned with the construction of annotated resources, which is as a fundamental step towards building automatic systems that analyze the argumentative structure of texts. Unlike most previous corpora, the annotations of our corpus provide a classification of argumentative units based on their content. In the most simple case, other works distinguish only argumentative discourse units from other text, as we do in (Al-Khatib et al., 2016). Some

[1] Webis-Editorials-16 corpus, http://www.uni-weimar.de/en/media/chairs/webis/corpora/

corpora contain unit annotations based on the role units take in arguments, e.g., premise or conclusion (Stab and Gurevych, 2014; Habernal and Gurevych, 2015). Such annotations represent the general argumentative structure, but they do not encode the means an author uses to persuade the readers.

Previous corpora that contain content-based unit annotations are not suitable for analyzing argumentation strategies due to their annotation method or to the genre of the contained texts. Similar to our corpus, the CE corpus of Wikipedia articles (Aharoni et al., 2014; Rinott et al., 2015) also considers types of evidence (anecdotes, statistics, and testimony). However, evidence is annotated only where it relates to a set of given topics, so no complete annotation of the article's discourse is provided. Park and Cardie (2014) use annotations to distinguish verifiable from non-verifiable evidence, which is not related to argumentation strategies. The corpus of Habernal and Gurevych (2016) includes logos and pathos annotations. While this would be appropriate for studying argumentation strategies, the corpus consists of short forum discussions, which follow no strategic plan.

Several corpora serve to study relations between the argumentative discourse units in a text (Boltužić and Šnajder, 2014; Ghosh et al., 2014; Peldszus and Stede, 2015; Stab and Gurevych, 2014), e.g., whether a unit supports or attacks another unit or the thesis of the text. Some patterns of such relations would certainly be relevant for an analysis of argumentation strategies, like the rebuttal of a common ground that seems to counter the author's stance. However, similar to the low number of attack relations in persuasive essays (Stab and Gurevych, 2014), we found few insightful patterns of this kind in editorials.

While we focus on the argumentation strategy of an entire text, the 60 schemes of Walton et al. (2008) can be seen as representing the strategy of single arguments, although at a much finer granularity. Annotating a complete argumentative text according to all these schemes is very difficult. The only available resource with scheme annotations we are aware of, the Araucaria corpus (Reed and Rowe, 2004), contains annotations of single arguments but not of whole texts.

The construction of the corpus introduced in this paper has already been sketched in (Kiesel et al., 2015). Due to the pure form of argumentation found in news editorials, we discuss suitability of the corpus as a resource for shared tasks on argument mining there. Most existing work on news editorials in computational linguistics studies sentiment and opinions (Yu and Hatzivassiloglou, 2003; Wilson and Wiebe, 2003; Bal, 2009). A first conceptual study of the relation between the opinions in a news editorial and its argumentative structure is described in (Bal and Saint-Dizier, 2009). Besides, to our knowledge, the only work in this regard is the very recent work of Chow (2016) on Chinese editorials. Unfortunately, the annotation of this corpus is restricted to the argumentativeness of paragraphs as a whole, which makes the corpus unsuitable for analyzing argumentation strategies.

3 Model

We now introduce the model (in terms of annotation scheme) that we propose for analyzing the argumentation strategy of a news editorial. The model is focused on the sequential structure of argumentative discourse, which benefits a reliable statistical recognition of strategy patterns. To this end, we separate an editorial into argumentative discourse units of six different types where each type represents a particular role in the discourse. While our model is in line with related work on evidence types (Rinott et al., 2015), we assign a type to each unit in order to capture an editorial's overall argumentation strategy.

In particular, we see argumentative discourse units as the smallest elements of the argumentative discourse of an editorial. They represent the propositions stated by the editorial's author to discuss, directly or indirectly, his or her thesis. In general, propositions affirm or deny that certain entities have certain attributes. An entity may be an object (e.g., *milk*), a being (e.g., *Obama* or *we*), or an abstract concept (e.g., *learning to cooperate*). Technically, we define a unit based on the notion of propositions as follows:

> **Argumentative Discourse Unit:** An argumentative discourse unit is the minimum text span that completely covers one or more propositions. It always includes a subject (or a placeholder, such as "which") and a verb, and it needs to include an object if grammatically required. It spans at most one sentence.

The units of a news editorial play different roles in the editorial's argumentative discourse. E.g., some

represent knowledge or beliefs of the author or other people, and some serve as evidence in favor or against the truth of other units. We assume each unit to refer to exactly one of six types:

1. **Common Ground:** The unit states common knowledge, a self-evident fact, an accepted truth, or similar. It refers to general issues, not to specific events. Even if not known in advance, it will be accepted without proof or further support by all or nearly all possible readers.

   Example: *"History warns us what happens when empires refuse to teach known values that strengthen societies and help protect them from enemies intent on their destruction."*

2. **Assumption:** The unit states an assumption, conclusion, judgment, or opinion of the author, a general observation, possibly false fact, or similar. To make readers accept it, it is, or it would need to be supported by other units.

   Example: *"For too long young people have relied on adults who have done too little to stop the violation of the rights of the children for whom they were responsible."*

3. **Testimony:** The unit gives evidence by stating or quoting that a proposition was made by some expert, authority, witness, group, organization, or similar.

   Example: *"According to The Yazidi Fraternal Organization (YFO), thousands of young Yazidi women and children are being used by ISIL as sex slaves."*

4. **Statistics:** The unit gives evidence by stating or quoting the results or conclusions of quantitative research, studies, empirical data analyses, or similar. A reference may but needs not always be given.

   Example: *"Of the total of 779 men and boys that have been detained at Guantanamo Bay since 2002, only nine have been convicted of any crime."*

5. **Anecdote:** The unit gives evidence by stating personal experience of the author, an anecdote, a concrete example, an instance, a specific event, or similar.

   Example: *"In 1973, it deployed 18,000 troops with 300 tanks to save Damascus during the 'October War'."*

6. **Other:** The unit does not or hardly adds to the argumentative discourse or it does not match any of the above classes.

   Example: *"Happy New Year!"*

Our hypothesis is that these six types suffice to capture the main structural characteristics of argumentation strategies in news editorials. At the same time, they define an annotation scheme for a fine-grained mining of argumentative discourse units. This scheme represents the basis for the corpus we constructed.

4 Corpus Construction

This section describes the construction and annotation process of the news editorial corpus based on the proposed model (Section 3). The purpose of the corpus is to study different argumentation strategies in news editorials in terms of the argumentative discourse units they use.

4.1 Data Acquisition and Preparation

Before the annotation, the editorials are selected from three diverse news portals and decomposed into clause-like segments in order to ease the annotation process to achieve scale.

Selection of Argumentative News Editorials The corpus consists of editorials from *aljazeera.com*, *foxnews.com*, and *theguardian.com*. This selection of news portals cover diverse cultures and styles. They are internationally well-known and have separate editorial sections. We randomly selected 100 editorials from each portal that (1) are published within the same short time interval (December 2014 and January 2015) to facilitate a topical overlap, (2) sparked at least a small discussion (had at least 5 comments), and (3) contain at least 250 words (to filter out texts that just pose a question instead of arguing).

Type	Editorials	Total	Mean	Std. dev.	Median	Min	Max
Tokens	**All editorials**	**287364**	**957.88**	**257.28**	**932**	**298**	**1894**
	Al Jazeera	106430	1064.30	236.05	1033	440	1671
	Fox News	86415	864.15	226.36	855	298	1613
	Guardian	94519	945.19	267.13	906	481	1894
Sentences	**All editorials**	**11754**	**39.18**	**13.00**	**37**	**12**	**114**
	Al Jazeera	3962	39.62	10.55	38	16	75
	Fox News	3912	39.12	13.45	39	12	104
	Guardian	3880	38.80	14.65	36	18	114
Paragraphs	**All editorials**	**4664**	**15.55**	**6.48**	**15**	**2**	**45**
	Al Jazeera	1896	18.96	5.15	19	7	33
	Fox News	1689	16.89	6.71	16	2	45
	Guardian	1079	10.79	4.29	10	5	31
Segments	**All editorials**	**35665**	**118.88**	**38.21**	**116**	**28**	**309**
	Al Jazeera	11521	115.21	31.68	113	32	218
	Fox News	11315	113.15	35.4	112	28	231
	Guardian	12829	128.29	44.58	122	59	309

Table 1: Distribution of tokens, sentences, paragraphs, and segments in the corpus before annotation.

Pre-Segmentation of Argumentative Discourse Units To allow for an annotation at larger scale, we automatically segmented the editorials before the annotation but then allowed annotators to merge adjacent segments to discard incorrect unit boundaries. In this setup, the annotators do not have to choose the exact unit boundaries, which simplifies the annotation process while making the evaluation of the annotator-agreement more intuitive. A similar manual approach was used by Park and Cardie (2014).

In detail, the applied segmentation algorithm, which we will make publicly available, starts a new segment at the beginning or end of every clause not preceded by a relative pronoun. Clauses were identified using a state-of-the-art dependency parser (Manning et al., 2014) and the clause tags from the Penn Treebank Guidelines (Bies et al., 1995). The heuristic behind the segmentation was chosen based on a careful analysis of news editorials as well as of the persuasive essays corpus from (Stab and Gurevych, 2014), since essays resemble editorials in the way they compose argumentative discourse units. An evaluation of the segmentation algorithm on that corpus yielded very satisfying results: The algorithm segmented the 90 essays into 5132 segments. Only nine of these segments should have been split further, as they overlapped with several ground-truth units from the essay corpus. On the other hand, the segmentation was somewhat too fine-grained, namely, the 1552 ground-truth units were split into 3637 segments. In our setup, however, the annotators then perform the necessary segment merges. Table 1 shows statistics about the size of the corpus and its three sub-corpora after segmentation.

4.2 Annotation Process

Given the 300 selected news editorials, an annotation process was performed in order to identify all argumentative discourse units in each newspaper editorial, including an assignment of one of the six types from Section 3 to each unit. The main steps of this process are summarized in the following.

Task Definition First, each editorial had to be read as whole in order to understand the main topic and to follow the stance of the editorial's author.

As the annotation task, one out of eight classes had to be chosen for each segment of each editorial (see pre-segmentation above): (1–6) Any of the six types of argumentative discourse units of our model from Section 3, (7) *no unit*, when the segment does not belong to a unit, and (8) *continued*, when the segment needs to be merged with subsequent segments in order to obtain a unit. In case (8), the class assigned to the last segment determines the class of the merged unit.

The annotation guidelines given to the annotators contained the type definitions from Section 3 and a few clear and controversial examples for each type. In addition, we pointed out that the correct classification of a segment may require looking at surrounding segments. Also, no distinction should be made between true and false propositions (e.g., a wrong testimony should still be classified as testimony).

	Common ground	Assumption	Anecdote	Testimony	Statistics	Other	No unit	Continued	Overall
Fleiss' κ	0.114	0.613	0.399	0.591	0.582	0.152	0.365	0.684	**0.560**

Table 2: Inter-annotator agreement in the main annotation process, quantified in terms of Fleiss' κ.

Pilot Annotation Study A pilot study was conducted on nine editorials to evaluate the guidelines and to select the annotators. For this, three editorials were chosen from each portal. They are not part of the corpus, but were acquired and segmented in the same fashion.

We decided to conduct the annotation process via the professional crowdsourcing platform *upwork.com* in order both to increase scalability and to obtain independent annotators. The list of candidate annotators comprised ten freelancers. All of them were native English speakers, had at least a bachelor's degree, and had already knowledge about argumentation theories from their education.

Seven annotators completed all nine editorials, taking around 30 minutes per editorial on average. The Fleiss' κ agreement score for all seven annotators was a moderate 0.433 (J. Richard Landis, 1977). As we observed remarkable drops in the agreement caused by either of three specific annotators, we decided to exclude those annotators and keep the remaining four for the main annotation study.

An error analysis of the annotation of the four annotators revealed insightful hard cases. For instance, the annotators had difficulties to distinguish between *common ground* and *anecdote* for units discussing a specific event that is well-known universally. Also, there was notable disagreement between *common ground* and *assumption*. This was expected, though, since the distinction of these two types appears more subjective than for other type combinations. Nevertheless, the agreement between the four annotators for all types was substantial with $\kappa = 0.606$. Therefore, we decided not to modify our scheme, but only to clarify the type definitions and to add some additional examples that clarify these hard cases.

Main Annotation Process The 300 corpus editorials were evenly distributed among the four annotators. Each annotator got 225 editorials to annotate, 75 from each news portal. Accordingly, each editorial was annotated by three annotators.

4.3 Annotation Results

We analyzed the results of the annotation process in order to examine (1) the reliability of the corpus and (2) the major disagreements in units and types between the annotators. Our main findings are as follows:

Inter-Annotator Agreement In terms of Fleiss' κ, the overall agreement is 0.56. As broken down in Table 2, however, the types *common ground* and *other* have only a slight agreement, while the annotators achieved fair agreement for *no unit* and *anecdote* as well as moderate or substantial agreement for the remaining four types. Moreover, for 94.4% of all segments at least two of three annotators agree on one type, suggesting that a resort to majority agreement is very adequate. Considering that the annotators had to decide among eight different classes for every segment, such agreement seems high in overall terms. Therefore, we conclude that the annotations of the corpus can be seen as reliable.

Disagreement Analysis To analyze the disagreement between the annotators, we created the confusion probability matrix (CPM, Cinková et al. (2012)) for all classes shown in Table 3. Each matrix cell shows the probability of choosing the column's class, given that another annotator chose row's class. Table 3 reveals the five class-pairs where annotators are most confused between:

1. Disagreement between *other* and *assumption* (0.324). An explanation may be that the annotators interpreted the intention of the author of a respective editorial differently in some segments.

   An example unit that led to confusion is *"I just don't get it"* after another unit *"the rave reviews for the first episode make me feel like a teetotaller at a lock-in"*. The first unit could be interpreted as an implicit assumption about the reviews in the second unit, say, that the review is corrupt or hard to understand. However, it could also simply be seen as an interjection not belonging to any argument.

2. Disagreement between *common ground* and *assumption* (0.562). Although we revised our guidelines to resolve the ambiguity of these types, their distinction still seems to be hard in practice.

	Common ground	Assumption	Testimony	Statistics	Anecdote	Other	No unit	Continued
Common ground	0.129	**0.562**	0.012	0.005	**0.163**	0.012	0.075	0.042
Assumption	0.035	0.701	0.017	0.010	0.075	0.014	0.066	0.083
Testimony	0.006	0.134	0.618	0.016	0.087	0.002	0.034	0.104
Statistics	0.006	0.195	0.042	0.603	0.074	0.002	0.037	0.040
Anecdote	0.037	0.277	0.041	0.013	0.451	0.016	0.059	0.104
Other	0.018	**0.324**	0.006	0.003	0.101	0.166	**0.310**	0.073
No unit	0.008	0.114	0.008	0.003	0.027	0.023	0.440	**0.377**
Continued	0.001	0.036	0.006	0.001	0.012	0.001	0.094	0.849

Table 3: Probability confusion matrix for all pairs of annotated types of argumentative discourse units.

For example, the unit *"To see a movie legally you must leave your house, queue up, ask someone for a ticket and then sit down in the company of others"* can be viewed as *common ground* if the annotator believes that most people agree with this statement, meaning there is no need for justification. In contrast, it is viewed as an *assumption* if people are assumed to disagree to some extent, e.g., because a DVD can be bought and watched legally at home.

3. Disagreement between *common ground* and *anecdote* (0.163). Confusion between these types occurred in cases where there was a distinct fact that the editorial's author uses to support his stance.

   For example, *Iraq's Sunnis were the leading force within the Iraqi army since its foundation on January 6, 1921*. This declaration was used to support the author's claim that the Sunnis respect their army and see it as a national institution of unrivaled prestige.

4. Disagreement between *other* and *no unit* (0.310). Without clear reason, these classes seem to have been used interchangeably sometimes.

5. Disagreement between *no unit* and *continued* (0.377). The main reason for such disagreement was that the annotators dealt with discourse markers and connectives inconsistently.

   E.g., in case of the subsequent segments (1) *"According to the administration"* and (2) *"the film by Nakoula Basseley Nakoulahad sparked spontaneous riots to defend Muhammad's honor"*, the first was partly seen as *no unit*, although our guideline specified to consider such segments as one unit.

Post-Processing of the Annotations For the final version of the corpus, the corpus segments were consolidated using the *majority vote* for each segment: If at least two workers agreed on the class of a segment, the segment was classified accordingly. Else, one of this paper's authors selected one of three suggested classes. Based on the disagreement analysis and a manual inspection of the annotations, we found a few general misclassifications that could be fixed semi-automatically. While overruling some decisions of the annotators, we thereby achieve a more consistent annotation, which is crucial for learning based on the corpus. In particular, we conducted the following post-processing steps:

- A considerable number of segments was annotated as *no unit*, although it should have been merged with the next segment. We reviewed several instances of this problem, such as conditional statements (e.g., of the form "if A then B") or relations that are not argumentative but temporal or spatial (e.g., of the form "when A then B"). Where necessary, we then merged the respective segments.

- According to our definitions, only non-rhetorical questions should be labeled as *no unit*. However, many rhetorical questions were also classified as *no unit*, even though they had, in our view, a clear argumentative function: most times implicitly conveying claims, recommendations, or similar. Following our definitions, we reclassified them as *assumption*.

- Second person voice segments were often classified as *no unit*, possibly due to the unintended interpretation that a unit requires an *explicit* subject. Nearly all of them are appeals, recommendations, or similar. As above, we thus reclassified them as *assumption*.

In addition to the corrections above, we excluded periods, commas, or similar punctuation at the end of segments and put them in separate *no unit* segments. This is important to prevent unit type classifiers from misleadingly learning to identify particular types based on these characters.

Type	Total	Mean	Std. dev.	Median	Min	Max	Percent
Common ground	241	0.80	1.53	0	0	13	1.7%
Assumption	9792	32.64	12.42	32	3	86	68.4%
Anecdote	2603	8.68	9.12	7	0	77	18.2%
Testimony	1089	3.63	5.42	2	0	44	7.6%
Statistics	421	1.40	2.76	0	0	19	2.9%
Other	167	0.56	1.64	0	0	24	1.2%
All units	14313	47.71	14.28	46	14	132	100%

Table 4: The distribution of types of argumentative discourse units in the created corpus.

4.4 The Corpus

Table 4 presents some statistics of the final corpus, obtained after post-processing. We observe that the most frequent type of argumentative discourse unit is *assumption* covering almost 68.4% of all units. The *anecdote* type represents about 18.2%, surpassing the *testimony* (7.6%), *statistics* (2.9%), and *common ground* (1.7%). *Other*, finally, only refers to a very low percentage of units (1.2%). On one hand, this supports the hypothesis that editorials are a rich source for argumentation. On the other hand, it serves as strong evidence that the six proposed types of units cover most units found in editorials.

5 Argumentation Strategies

Aside from the general use of our corpus for the development and evaluation of approaches to argumentation mining, the corpus serves to investigate how authors argue in news editorials in order to persuade the readers. We do not actually analyze such argumentation strategies here. Rather, we present some basic findings that indicate the potential of our corpus for analysis in this regard.

In particular, Table 5 shows detailed statistics about the types of argumentative discourse units in the corpus. Overall, we see that the length of news editorials is quite stable across the three news portals, with a mean between 48.76 (*The Guardian*) and 52.34 units (*Fox News*). Some very short (minimum 14 units) and very long editorials (maximum 132 units) exist, though.

Regarding the distribution of the types, some general tendencies as well as some insightful differences can be observed. Generally, more than two third of an editorial usually comprises assumptions. This is not surprising, as the type *assumption* covers both claims and any other propositions that may require justification. While *The Guardian* has the highest proportion of assumptions (71.7%), it represents the median for most other types. Fox News more strongly relies on *common ground*, with more than one unit of that type on average. Even more clearly, 8.7% of all units in Fox News editorials is *testimony* evidence, about twice as many on average as in The Guardian (4.55 vs. 2.53). In contrast, Al Jazeera seems to put more emphasis on *anecdote*. At least, it spreads anecdotes across more units (21.0% of all). Interestingly, all three portals behave very similar in their resort to *statistics* at the same time.

Altogether, these numbers suggest that our corpus is worthy of being analyzed regarding argumentation strategies. While we leave such analyses to future research, we expect that especially the sequence of types of argumentative discourse units in an editorial can be decisive. Similar findings have been reported for the impact of discourse functions on the quality of essays (Persing et al., 2010), the inference of intentions from rhetorical moves (Teufel, 2014), and the generality of sentiment flows across review domains (Wachsmuth et al., 2015). Possibly, an adequate granularity (editorial level vs. paragraph level) and abstraction (all types vs. majority in paragraphs vs. ...) may have to be found for editorials, though, because the high average number of units allows for significant variance in respective sequences.

6 Discussion and Conclusion

Although news editorials are considered as one of the purest forms of argumentative text, so far few works exist in computational linguistics that study them. In this paper, we have presented the development of an annotated corpus for the mining of an editorial's argumentative discourse and the analysis of its argumentation strategy. We expect that such an analysis will contribute to the computational assessment of the quality and persuasiveness of monological argumentation. In recent work, we have analyzed the argumentative structure of persuasive essays in order to assess their argumentation quality (Wachsmuth

Type	Editorials	Total	Mean	Std. dev.	Median	Min	Max	Percent
Common ground	**All editorials**	**241**	**0.80**	**1.53**	**0**	**0**	**13**	**1.7%**
	Al Jazeera	59	0.59	0.97	0	0	5	1.2%
	Fox News	104	1.04	2.04	0	0	13	2.0%
	Guardian	78	0.78	1.36	0	0	10	1.6%
Assumption	**All editorials**	**9792**	**32.64**	**12.42**	**32**	**3**	**86**	**68.4%**
	Al Jazeera	3294	32.94	10.79	33	3	65	66.9%
	Fox News	3002	30.02	13.16	30	5	86	57.4%
	Guardian	3496	34.96	12.68	32	6	73	71.7%
Anecdote	**All editorials**	**2603**	**8.68**	**9.12**	**7**	**0**	**77**	**18.2%**
	Al Jazeera	1036	10.36	10.19	8	0	71	21.0%
	Fox News	727	7.27	6.67	6	0	37	13.9%
	Guardian	840	8.40	9.82	6	0	77	17.2%
Testimony	**All editorials**	**1089**	**3.63**	**5.42**	**2**	**0**	**44**	**7.6%**
	Al Jazeera	381	3.81	4.61	3	0	22	7.7%
	Fox News	455	4.55	7.42	2	0	44	8.7%
	Guardian	253	2.53	3.09	2	0	16	5.2%
Statistics	**All editorials**	**421**	**1.40**	**2.76**	**0**	**0**	**19**	**2.9%**
	Al Jazeera	141	1.41	2.60	0	0	17	2.9%
	Fox News	143	1.43	3.23	0	0	19	2.7%
	Guardian	137	1.37	2.37	0	0	12	2.8%
Other	**All editorials**	**167**	**0.56**	**1.64**	**0**	**0**	**24**	**1.2%**
	Al Jazeera	12	0.12	0.41	0	0	2	0.2%
	Fox News	83	0.83	1.20	0	0	5	1.6%
	Guardian	72	0.72	2.49	0	0	24	1.5%
All units	**All editorials**	**14313**	**47.71**	**14.28**	**46**	**14**	**132**	**100.0%**
	Al Jazeera	4923	49.23	12.23	48	21	81	100.0%
	Fox News	5234	52.34	15.64	50	17	123	100.0%
	Guardian	4876	48.76	16.55	46	22	132	100.0%

Table 5: Distribution of types of argumentative discourse units in the complete corpus and in the subcorpus of each news portal. Percentages refer to the proportions of units in the respective (sub-) corpus.

et al., 2016). While we focused on the logos means of persuasion there, argumentation strategies actually bring together logos, pathos, and ethos (Aristotle, 2007).

Our corpus is based on a fine-grained model (in terms of an annotation scheme) that we propose for capturing the different types of argumentative discourse units found in news editorials and in similar argumentative texts. We have detailed the annotation process and we have presented empirical evidence for typical characteristics of editorials and their variance across news portals. While not being considered in the model, we point out that units sometimes may have different types, e.g., they may represent both testimonial and statistical evidence. To create a clear classification setting, we decided to assign exactly one type to each unit, though, and to give the annototators guidelines about what type to prefer in what context. Partly, they are already encoded in the presented type definitions. Aside from that, we observed rather low agreement for the infrequent type *common ground*. Still, the resulting annotations may be valuable for research questions related to argumentation quality or persuasiveness. For instance, some authors use specific terms such as "in fact" or "for sure" before assumptions to let them appear as common ground. Our corpus helps to detect such cases.

In general, there is room to extend our corpus in future work. Most evidently, a deeper analysis of the argumentative discourse of news editorials will need to consider the relations between units that make up arguments. Among others, the relations will also reveal what are the main claims of an editorial. Unlike related text genres such as persuasive essays, however, editorials usually have no clear hierarchical argumentative structure, partly due to a frequent resort to enthymemes. This makes a consistent annotation of relations very challenging. We thus decided to focus on the types of units in the given form of the corpus, thereby ending up with a reliable corpus of reasonable size that we will make freely available. Recently, the need for according benchmark corpora was discussed in the emerging community of computational argumentation in order to allow for shared tasks and similar (Gurevych et al., 2016).

References

Ehud Aharoni, Anatoly Polnarov, Tamar Lavee, Daniel Hershcovich, Ran Levy, Ruty Rinott, Dan Gutfreund, and Noam Slonim. 2014. A Benchmark Dataset for Automatic Detection of Claims and Evidence in the Context of Controversial Topics. In *Proceedings of the First Workshop on Argumentation Mining*, pages 64–68, Baltimore, Maryland. Association for Computational Linguistics.

Khalid Al-Khatib, Henning Wachsmuth, Matthias Hagen, Jonas Köhler, and Benno Stein. 2016. Cross-Domain Mining of Argumentative Text through Distant Supervision. In *15th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 16)*, pages 1395–1404. Association for Computational Linguistics, June.

Aristotle. 2007. *On Rhetoric: A Theory of Civic Discourse (George A. Kennedy, Translator)*. Clarendon Aristotle series. Oxford University Press.

Bal Krishna Bal and Patrick Saint-Dizier. 2009. Towards and Analysis of Argumentation Structure and the Strength of Arguments in News Editorials. In *AISB Symposium on Persuasive Technologies*, pages 55–63.

Bal Krishna Bal. 2009. Towards an analysis of opinions in news editorials: How positive was the year? In *Proceedings of the Eighth International Conference on Computational Semantics*, IWCS-8 '09, pages 260–263, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. 1995. Bracketing Guidelines for Treebank II Style Penn Treebank Project. Technical report, University of Pennsylvania.

Filip Boltužić and Jan Šnajder. 2014. Back up your Stance: Recognizing Arguments in Online Discussions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 49–58, Baltimore, Maryland, June. Association for Computational Linguistics.

Marisa Chow. 2016. Argument identification in chinese editorials. In *NAACL Student Research Workshop 2016*. Association for Computational Linguistics.

Silvie Cinková, Martin Holub, and Vincent Kríž. 2012. Managing Uncertainty in Semantic Tagging. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 840–850, Stroudsburg, PA, USA. Association for Computational Linguistics.

Debanjan Ghosh, Smaranda Muresan, Nina Wacholder, Mark Aakhus, and Matthew Mitsui. 2014. Analyzing Argumentative Discourse Units in Online Interactions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 39–48, Baltimore, Maryland, June. Association for Computational Linguistics.

Iryna Gurevych, Eduard H. Hovy, Noam Slonim, and Benno Stein. 2016. Debating Technologies (Dagstuhl Seminar 15512). *Dagstuhl Reports*, 5(12):18–46.

Ivan Habernal and Iryna Gurevych. 2015. Exploiting Debate Portals for Semi-Supervised Argumentation Mining in User-Generated Web Discourse. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2127–2137.

Ivan Habernal and Iryna Gurevych. 2016. Argumentation mining in user-generated web discourse. *Computational Linguistics*, page (in press). Submission received: 2 April 2015; revised version received: 20 April 2016; accepted for publication: 14 June 2016. Pre-print available at http://arxiv.org/abs/1601.02403.

Gary G. Koch J. Richard Landis. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159–174.

Johannes Kiesel, Khalid Al-Khatib, Matthias Hagen, and Benno Stein. 2015. A Shared Task on Argumentation Mining in Newspaper Editorials. In *Proc. of the 2nd Workshop on Argumentation Mining*, pages 35–38.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Joonsuk Park and Claire Cardie. 2014. Identifying Appropriate Support for Propositions in Online User Comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38, Baltimore, Maryland, June. Association for Computational Linguistics.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948, Lisbon, Portugal, September. Association for Computational Linguistics.

Isaac Persing, Alan Davis, and Vincent Ng. 2010. Modeling Organization in Student Essays. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 229–239.

Chris Reed and Glenn Rowe. 2004. Araucaria: Software for Argument Analysis, Diagramming and Representation. *International Journal on Artificial Intelligence Tools*, 13.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show Me Your Evidence – An Automatic Method for Context Dependent Evidence Detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 440–450.

Christian Stab and Iryna Gurevych. 2014. Annotating Argument Components and Relations in Persuasive Essays. In *Proceedings of the 25th International Conference on Computational Linguistics, COLING 2014*, pages 1501–1510.

Simone Teufel. 2014. Scientific Argumentation Detection as Limited-Domain Intention Recognition. In *Proceedings of the Workshop on Frontiers and Connections between Argumentation Theory and Natural Language Processing*, pages 101–109.

Teun A. van Dijk. 1992. Racism and argumentation:"Race Rio" Rhetoric in Tabloib Editorials . In *In van Emeren et al. (Eds.), Argumentation illuminated*.

Teun A. van Dijk. 1995. Opinions and Ideologies in Editorials. In *Proceedings of the 4th International Symposium of Critical Discourse Analysis, Language, Social Life and Critical Thought*, Athens, June.

Henning Wachsmuth, Johannes Kiesel, and Benno Stein. 2015. Sentiment Flow – A General Model of Web Review Argumentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 601–611, Lisbon, Portugal.

Henning Wachsmuth, Khalid Al-Khatib, and Benno Stein. 2016. Using Argument Mining to Assess the Argumentation Quality of Essays. In *Proceedings of the 26th International Conference on Computational Linguistics*.

Douglas Walton, Christopher Reed, and Fabrizio Macagno. 2008. *Argumentation schemes*. Cambridge University Press.

Theresa Wilson and Janyce Wiebe. 2003. Annotating opinions in the world press. In *Proceedings of the Fourth SIGdial Workshop of Discourse and Dialogue*.

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, pages 129–136, Stroudsburg, PA, USA. Association for Computational Linguistics.

Universal Dependencies for Turkish

Umut Sulubacak ⋆ and **Memduh Gökırmak** †
Department of Computer Engineering
Istanbul Technical University
34469 Istanbul, Turkey
{⋆sulubacak|†gokirmak}@itu.edu.tr

Francis M. Tyers
HSL-fakultehta
UiT Norgga árktalaš universitehta
N-9018 Tromsø, Norway
francis.tyers@uit.no

Çağrı Çöltekin
Department of Linguistics
University of Tübingen
72074 Tübingen, Germany
ccoltekin@sfs.uni-tuebingen.de

Joakim Nivre
Department of Linguistics and Philology
Uppsala University
75126 Uppsala, Sweden
joakim.nivre@lingfil.uu.se

Gülşen Eryiğit
Department of Computer Engineering
Istanbul Technical University
34469 Istanbul, Turkey
gulsen.cebiroglu@itu.edu.tr

Abstract

The Universal Dependencies (UD) project was conceived after the substantial recent interest in unifying annotation schemes across languages. With its own annotation principles and abstract inventory for parts of speech, morphosyntactic features and dependency relations, UD aims to facilitate multilingual parser development, cross-lingual learning, and parsing research from a language typology perspective. This paper presents the Turkish IMST-UD Treebank, the first Turkish treebank to be in a UD release. The IMST-UD Treebank was automatically converted from the IMST Treebank, which was also recently released. We describe this conversion procedure in detail, complete with mapping tables. We also present our evaluation of the parsing performances of both versions of the IMST Treebank. Our findings suggest that the UD framework is at least as viable for Turkish as the original annotation framework of the IMST Treebank.

1 Introduction

The Universal Dependencies (UD)[1] project is an international collaborative project to make cross-linguistically consistent treebanks available for a wide variety of languages. Currently in version 1.3, the UD project covers 40 languages, including two Turkic languages: Kazakh, which was annotated from scratch, and Turkish, the creation of which is described in this paper.

The universal annotation guidelines of UD are based on the Google Universal Part-of-Speech Tagset (Petrov et al., 2012) for parts of speech, the Interset framework (Zeman, 2008) for morphological features, and Stanford Dependencies (De Marneffe et al., 2006; Tsarfaty, 2013; De Marneffe et al., 2014) for dependency relations. The objective of harmonizing annotation guidelines as far as possible is to make comparison of parsing results and investigating cross-linguistic methods across languages easier. This is achieved by a number of principles, including the primacy of content words, distinguishing core arguments from modifiers and distinguishing clausal constituents from nominals.

The IMST-UD Treebank was first released in UD version 1.3 and became the first Turkish treebank to be included in a UD release. The treebank was created by automatic conversion of the IMST

[1] http://universaldependencies.org/

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3444–3454, Osaka, Japan, December 11-17 2016.

Treebank (Sulubacak et al., 2016), which is itself a reannotation of the METU-Sabancı Turkish Treebank (Oflazer et al., 2003; Atalay et al., 2003). Although the annotation framework of the IMST Treebank was revised, it is still fundamentally similar to that of the METU-Sabancı Treebank and radically different from the UD framework in both morphology and syntax.

In this paper, we describe the procedures employed in converting the annotation schemes of the IMST Treebank to the corresponding UD-compliant schemes. We also provide comparative statistics on the composition of the IMST Treebank before and after the conversion. Afterwards, we report our initial parsing results on the new IMST-UD Treebank in comparison with the original IMST Treebank. The paper is structured as follows: Section 2 discusses the conversion procedure, Section 3 describes the IMST Treebank and the relevant statistics, Section 4 explains the parsing tests and their analysis, and finally, Section 5 presents the conclusion.

2 Mapping

In this section, we describe the procedure we employed in mapping the original IMST Treebank to a UD-compliant framework. The UD-compliant grammatical representations to which we mapped the original annotation schemes were largely adapted from previous work in the subject (Çöltekin, 2015; Çöltekin, 2016). The original treebank was available in the CoNLL-X data format (Buchholz and Marsi, 2006), where sentences are bounded by empty lines, and every word has a separate row, each containing a tab-delimited array of morphosyntactic data pertaining to the word. In compliance with the UD standard, the converted sentences were output in the CoNLL-U format.[2]

The sections to follow present explanations and discussions on the procedures of mapping morphological and syntactic data, as well as some idiosyncratic linguistic phenomena. Quick reference tables were also provided where applicable, showing what conditions on the *source* unit are required to assign which properties to the *target* unit.

2.1 Segmentation

The inflectional group (IG) formalism (Oflazer, 1999; Hakkani-Tür et al., 2002) was designed to make the highly agglutinative typology of Turkish tractable for language processing. Since then, it has seen usage in many influential works (Oflazer, 2003; Eryiğit and Oflazer, 2006) and has become the de facto standard in parsing Turkish. According to the formalism, orthographic tokens are divided into morphosyntactic words from *derivational boundaries*.[3] These units are called the inflectional groups (IGs) of the token. The IG formalism establishes these, rather than orthographic tokens, as the syntactic units of the sentence.

The original IMST treebank also follows its predecessors in using the IG formalism. The rightmost IG governs the word, while every other IG depends on the next one in line with the exclusive relation DERIV. Though a computationally effective representation, IGs are in contradiction with the UD principles. The representation dictates that the rightmost IG (which is, more often than not, a function word) be the head, whereas the leftmost IG (which is always a content word) is made to be the deepest dependent. As this does not comply with the principle of the primacy of content words, IGs have been removed during the conversion to UD. As a substitute, some derivational morphemes were treated as unbound enclitics, segmented off of their host words, assigned parts of speech such as ADP and AUX, and made to depend on their stems. Other morphemes were merged with their stems and were either fully lexicalized or marked for complex morphology. By a lexicalized derivation we mean tokens for which the grammatical process of derivation is not represented, and the result of the derivation is considered to be the lemma. An example for this is shown with *küreselleşme* in Figure 1.

Table 1a outlines the derivations that were segmented off of their stems. The surface forms for each such segment was constructed with the help of a morphological synthesizer, by 1) compiling the morphological analysis of the whole token, then 2) removing the part that corresponds to the derivation and any

[2]The CoNLL-U format is itself a revised version of the CoNLL-X data format. A description of the format is maintained (at the time of writing) on the official UD website (`http://universaldependencies.org/format.html`).

[3]In this context, a *derivational boundary* is the boundary between a POS-changing derivational suffix (or zero morpheme) and the stem that it is added to.

Source		*Target*			
CPOSTAG	POSTAG	LEMMA	FORM	UPOSTAG	DEPREL
ADJ	AGT	*ci*	**SYNTHESIZE**	ADP	CASE
ADJ	FITFOR	*lik*	**SYNTHESIZE**	ADP	CASE
ADJ | NOUN	REL	*ki*	**SYNTHESIZE**	ADP	CASE
ADJ	WITH	*li*	**SYNTHESIZE**	ADP	CASE
ADJ	WITHOUT	*siz*	**SYNTHESIZE**	ADP	CASE
ADVERB	LY	*ce*	**SYNTHESIZE**	ADP	CASE
ADVERB	SINCE	*dir*	**SYNTHESIZE**	ADP	CASE
NOUN	NESS	*lik*	**SYNTHESIZE**	ADP	CASE
VERB	ZERO	*i*	**SYNTHESIZE**	AUX	COP

(a)

CPOSTAG	POSTAG
ADJ	INBETWEEN
ADJ	JUSTLIKE
ADJ	RELATED
NOUN	AGT
NOUN	DIM
VERB	ACQUIRE
VERB	BECOME

(b)

Table 1: *(a)* Segmentation of copulas and other derivations, and *(b)* lexicalized derivations.

following inflection, and finally 3) synthesizing the new form from this partial analysis. The segments were also assigned the lemmas and parts of speech given in the LEMMA and UPOSTAG columns of the table, and made to depend on their stems with the relation specified in the DEPREL column.

The derivations given in Tables 1a, 1b and 3 are made via the addition of various derivational suffixes. Each of these suffixes has several allomorphs according to vowel harmony (e.g. the agent-deriving suffix may have the following 16 forms: *–cı, –ci, –cu, –cü, –çı, –çi, –çu, –çü, –ıcı, –ici, –ucu, –ücü, –yıcı, –yici, –yucu, –yücü*), and sometimes there is no overt suffix (as in the third person singular copula, which is a zero morpheme). Moreover, words are often further inflected after derivation, or may be multiply derived, and the analysis of these cascading and overlapping suffixes is an ambiguous and unreliable process. Therefore, instead of derivational morphemes, the minor part-of-speech tags assigned to each word (given in the POSTAG column) were used to identify derivations.

Table 1b lists the derivations that were not considered sufficiently productive and merged with their stems. Although these derivations have varying degrees of productivity, words derived by them are largely confined to a limited group of fairly common derivations. The fact that these words were more often than not lexicalized in the original treebank served as our justification for the lexicalization. The *lexicalized token* was made to inherit the surface form, lemma, and all morphological and syntactic data from the *derivation*, as well as its dependents, before replacing both the *stem* and the *derivation*.

Table 3 summarizes the participle (verbal adjective), transgressive (verbal adverb) and gerund (verbal noun) derivations in the same manner. In compliance with the UD standard of encoding verb forms, the *merged token* was made to inherit the lemma of the *stem*, as well as the surface form, the CASE, PERSON[PSOR], NUMBER[PSOR] and TENSE features, the head index, and the dependents of the

Source		*Target*	
CPOSTAG	POSTAG	UPOSTAG	FEATS
ADJ	NUM	NUM	—
ADJ	—	ADJ	—
ADVERB	—	ADV	—
DET	—	DET	—
DUP	—	X	ECHO=RDP
CONJ	—	CONJ	—
INTERJ	—	INTJ	—
NOUN	NUM	NUM	—
NOUN	PROP | ABR	PROPN	—
NOUN	—	NOUN	—
POSTP	NEG	VERB	—
POSTP	QUES	AUX	—
POSTP	—	ADP	—
PRON	DEMONS	PRON	PRONTYPE=DEM
PRON	PERS	PRON	PRONTYPE=PRS
PRON	QUANT	PRON	PRONTYPE=IND
PRON	REFLEX	PRON	REFLEX=YES
PRON	—	PRON	—
PUNC	—	PUNCT	—
VERB	ZERO	AUX	—
VERB	—	VERB	—

Table 2: Part-of-speech tag mapping.

Source		*Target*	
CPOSTAG	POSTAG	UPOSTAG	FEATS
ADVERB	ADAMANTLY	VERB	VERBFORM=TRANS
ADVERB	AFTERDOINGSO	VERB	VERBFORM=TRANS
ADVERB	ASIF	VERB	VERBFORM=TRANS
ADVERB	ASLONGAS	VERB	VERBFORM=TRANS
ADVERB	BYDOINGSO	VERB	VERBFORM=TRANS
ADVERB	SINCEDOINGSO	VERB	VERBFORM=TRANS
ADVERB	WHILE	VERB	VERBFORM=TRANS
ADVERB	WHEN	VERB	VERBFORM=TRANS
ADVERB	WITHOUTBEINGABLETOHAVEDONESO	VERB	MOOD=ABIL | NEGATIVE=NEG | VERBFORM=TRANS
ADVERB	WITHOUTHAVINGDONESO	VERB	NEGATIVE=NEG | VERBFORM=TRANS
ADJ	AORPART	VERB	TENSE=AOR | VERBFORM=PART
ADJ	NARRPART	VERB	ASPECT=PERF | TENSE=PAST | VERBFORM=PART
ADJ	PASTPART	VERB	TENSE=PAST | VERBFORM=PART
ADJ	PRESPART	VERB	TENSE=PRES | VERBFORM=PART
ADJ	FUTPART	VERB	TENSE=FUT | VERBFORM=PART
NOUN	INF1	VERB	VERBFORM=GER
NOUN	INF2	VERB	VERBFORM=GER
NOUN	INF3	VERB	VERBFORM=GER

Table 3: Merging of verbal derivations (transgressives, participles and gerunds).

derivation. The *merged token* was also assigned a VERBFORM feature as designated by the mapping, along with ASPECT, MOOD, TENSE and NEGATIVE features, before replacing the *stem* and the *derivation*.

In addition to the derivations discussed previously in this section, there were some zero derivations in the original treebank that were immediately derived into other parts of speech without any inflection inbetween, such as when adjectives were derived into zero nouns before copular (verbal) derivations. These intermediate derivations held no morphosyntactic information and were eliminated in conversion.

2.2 Part-of-Speech Tags

The mapping of the UD part-of-speech tags are displayed in Table 2. Most parts of speech were mapped in a straightforward, one-to-one fashion, with a small number of exceptions. In some cases, extra morphological features were used for an expressive conversion.

2.3 Morphological Features

Table 4 shows the mapping of the morphological features. Derivational information was mostly kept in the minor part of speech (POSTAG) field in the original IMST Treebank. These tags were retained in the XPOSTAG field in the CoNLL-U output after the conversion. Using either a directly corresponding UD feature or a combination of other UD features, we were able to represent most of the information kept in these fields.

The TENSE, ASPECT and MOOD features are closely related and often fused in Turkish. In some cases, a multiply derived token may have more than one value for one of these features. Moreover, although the UD guidelines enforce these features for finite verbs, they were occasionally omitted in the IMST Treebank so that they would defer to a neutral value. Whenever one of these features had more than one corresponding value, we concatenated these values with a hyphen delimiter, except for multiple occurrences of the same feature value, and the cases specified in Table 4. If one of these features had no directly corresponding value, we assigned the implied default value (TENSE=PRES, ASPECT=PERF, and MOOD=IND). For instance, the feature sequence HASTILY | PROG1 was converted to ASPECT=PROG-RAPID | MOOD=IND | TENSE=PRES.

Source	*Target*
FEATS	FEATS
A1SG	PERSON=1 | NUMBER=SING
A2SG	PERSON=2 | NUMBER=SING
A3SG	PERSON=3 | NUMBER=SING
A1PL	PERSON=1 | NUMBER=PLUR
A2PL	PERSON=2 | NUMBER=PLUR
A3PL	PERSON=3 | NUMBER=PLUR
PNON	—
P1SG	PERSON[PSOR]=1 | NUMBER[PSOR]=SING
P2SG	PERSON[PSOR]=2 | NUMBER[PSOR]=SING
P3SG	PERSON[PSOR]=3 | NUMBER[PSOR]=SING
P1PL	PERSON[PSOR]=1 | NUMBER[PSOR]=PLUR
P2PL	PERSON[PSOR]=2 | NUMBER[PSOR]=PLUR
P3PL	PERSON[PSOR]=3 | NUMBER[PSOR]=PLUR
ABL	CASE=ABL
ACC	CASE=ACC
DAT	CASE=DAT
EQU	CASE=EQU
GEN	CASE=GEN
LOC	CASE=LOC
INS	CASE=INS
NOM	CASE=NOM
CARD	NUMTYPE=CARD
DIST	NUMTYPE=DIST
ORD	NUMTYPE=ORD

Source	*Target*
FEATS	FEATS
AOR	TENSE=AOR
FUT	TENSE=FUT
PAST | PAST	TENSE=PQP | REGISTER=INF
PAST	TENSE=PAST
PRES	TENSE=PRES
NARR | PAST	TENSE=PQP
NARR | NARR	TENSE=PQP | EVIDENTIALITY=NFH
NARR	TENSE=PAST | EVIDENTIALITY=NFH
HASTILY	ASPECT=RAPID
PROG1	ASPECT=PROG | REGISTER=INF
PROG2	ASPECT=PROG | REGISTER=FORM
REPEAT	ASPECT=DUR
STAY	ASPECT=DUR-PERF
ABLE	MOOD=ABIL
ALMOST	MOOD=PRO
COND	MOOD=CND
COP	MOOD=GEN
DESR	MOOD=DES
IMP	MOOD=IMP
NECES	MOOD=NEC
OPT	MOOD=OPT
NEG	NEGATIVE=NEG
POS	NEGATIVE=POS
CAUS	VOICE=CAU
PASS	VOICE=PASS

Table 4: Morphological feature mapping.

2.4 Dependency Relations

The mapping rules used in converting dependency relations are outlined in Tables 5, 6, 7, and 8. The conditions for these mapping rules are considerably more complex than for the parts of speech and the morphological features. More often than not, besides the original dependency relations, additional morphosyntactic and lexical data must be considered for an accurate mapping. Furthermore, the entire analysis of a given dependent may sometimes not suffice, and further data pertaining to the head token that governs that dependent must be considered as well (as specified under columns with *(head)* labels).

Table 5 shows the mappings for dependency relations that are essentially types of modifiers and determiners. The mapping conditions are exactly as arranged on the table, except for the mapping to the ADVCL relation, where if the word had the feature VERBFORM=GER, it was also required to have an adpositional dependent with a CASE dependency. This means having a CASE dependent on a verbal head, which is incompatible with the UD guidelines for the moment. However, as this is an issue that will be discussed in the future, we decided to wait and see whether a change in the guidelines will be made. Table 6 displays the rules for dependencies that denote multiword expressions and other compounds. Multiword expressions (MWEs) were mapped to five different UD relations dependending on their context. The remaining MWEs were converted according to their syntactic role in the sentence. For both of the groups covered in Tables 5 and 6, certain cases were only distinguishable by their lemmas. These cases are given in additional rows below each table.

Tables 7 and 8 show the mappings for the remaining dependency relations. These tables also give exact mapping conditions, except for tokens with OBJECT dependencies (Table 7), which were still mapped to CCOMP dependencies without a VERBFORM=GER feature if they had a copular dependent with a COP dependency. Table 8 is reserved for dependency conversions whose head indices were adjusted

Source				*Target*
DEPREL *(dep)*	CPOSTAG *(dep)*	FEATS *(dep)*	FEATS *(head)*	DEPREL
INTENSIFIER	ADV	—	—	ADVMOD:EMPH
NOT INTENSIFIER	ADV	—	—	ADVMOD
MODIFIER	—	—	VERBFORM=PART	ACL
MODIFIER	NUM	NUMTYPE=ORD | DIST	—	AMOD
MODIFIER	VERB	VERBFORM=GER | TRANS	—	ADVCL
MODIFIER	NUM	NUMTYPE=CARD	—	NUMMOD
MODIFIER	NOUN | PRON | PROPN	—	—	NMOD
POSSESSOR	NOUN	CASE=ABL	**NO** PERSON[PSOR]	NMOD
POSSESSOR	NOUN	—	PERSON[PSOR]	NMOD:POSS
DEPREL *(dep)*	CPOSTAG *(dep)*	LEMMA *(dep)*	LEMMA *(head)*	DEPREL
MODIFIER	ADJ	**NOT** *(hangi | nasıl | ne | nere)*	—	AMOD
MODIFIER	ADJ	*(hangi | nasıl | ne | nere)*	—	DET
DETERMINER	—	*(her | hiçbir | ne)*	**NOT** *(şey | yer | zaman)*	DET

Table 5: Dependency mapping: Modifiers and determiners.

Source					*Target*
DEPREL *(dep)*	CPOSTAG *(dep)*	CPOSTAG *(head)*	FEATS *(dep)*	FEATS *(head)*	DEPREL
POSSESSOR	NOUN	NOUN	CASE=NOM	**NO** PERSON[PSOR]	COMPOUND
MWE | MODIFIER	NUM	NUM	—	—	COMPOUND
MWE	X	X	ECHO=RDP	ECHO=RDP	COMPOUND:REDUP
MWE	PROPN	PROPN	—	—	NAME
DEPREL *(dep)*	CPOSTAG *(dep)*	CPOSTAG *(head)*	LEMMA *(dep)*	LEMMA *(head)*	DEPREL
MWE | DETERMINER	—	—	*(her | hiçbir | ne)*	*(şey | yer | zaman)*	MWE
MWE | MODIFIER	—	VERB	—	*(bulun | et | ol | kıl)*	COMPOUND:LVC

Table 6: Dependency mapping: Multiword expressions and other compounds.

Source			*Target*
DEPREL	CPOSTAG	POSTAG	DEPREL
APPOSITION	NOUN	—	APPOS
APPOSITION	VERB	—	PARATAXIS
OBJECT	VERB	VERBFORM=GER	CCOMP
OBJECT	—	**NO** VERBFORM=GER	DOBJ
PREDICATE	—	—	ROOT
SUBJECT	VERB	VERBFORM=GER	CSUBJ
SUBJECT	—	**NO** VERBFORM=GER	NSUBJ

Table 7: Dependency mapping: Other dependencies, keeping the typology.

along with their dependency relations. For the mappings marked **SWAP** in the HEAD column, the direction of the dependency was also reversed. The original dependent became the new head and vice versa, and the dependents of these tokens were swapped. For those marked **CLAUSAL**, the head of the dependency (usually the sentence root in the original IMST Treebank) was updated to the head of the clause in which the token occurs. If no such clause exists, the head of the sentence was assigned instead.

Source				*Target*	
DEPREL *(dep)*	CPOSTAG *(dep)*	POSTAG *(dep)*	LEMMA *(dep)*	DEPREL	HEAD
ARGUMENT	ADP	—	—	CASE	**SWAP**
ARGUMENT	AUX	QUES	—	AUX:Q	**SWAP**
ARGUMENT	VERB	NEG	—	COP:NEG	**SWAP**
CONJUNCTION	—	—	*(de | ki | mi)*	MARK	**CLAUSAL**
CONJUNCTION	—	—	**NOT** *(de | ki | mi)*	CC	**CLAUSAL**
COORDINATION	—	—	—	CONJ	**CLAUSAL**
INTENSIFIER	NUM	—	*ise*	DISCOURSE	**CLAUSAL**
PUNCTUATION	SMILEY	—	—	DISCOURSE	**CLAUSAL**
PUNCTUATION	—	—	—	PUNCT	**CLAUSAL**
VOCATIVE	INTJ | SYM	—	—	DISCOURSE	**CLAUSAL**
VOCATIVE	NOUN | PROPN	—	—	VOCATIVE	**CLAUSAL**

Table 8: Dependency mapping: Other dependencies, adjusting the typology.

For any remaining tokens whose dependencies were not updated by any of the given mapping rules, a catch-all UD relation was assigned according to its converted part of speech. Tokens with the part-of-speech tags ADP, CONJ, INTJ and PUNCT were respectively attached the dependency relations CASE, CC, VOCATIVE and PUNCT. Those with the tags ADJ, ADV, DET and NUM were respectively given the AMOD, ADVMOD, DET and NUMMOD relations. Any other token was assigned the NMOD relation.

2.5 Postprocessing

After the adjustments to segmentation and the conversion of part-of-speech tags, morphological features and dependency relations, we applied postprocessing routines to each sentence to ensure they constitute valid dependency trees. This step was also necessary in order to circumvent some cases in the original IMST Treebank where sentences did not have a unique token with the sentence root as the head. These cases were often due to dependencies such as CONJUNCTION, PUNCTUATION and VOCATIVE, which depended on the sentence root in certain contexts as required by the dependency grammar. Otherwise, a small number of annotation errors which broke the unique root constraint were also present in the original treebank, and these warranted addressing as well.

Initially, every token depending on the sentence root with a non-ROOT dependency was reassigned the clausal head (or, if not applicable, the sentential head) as its new head. The remaining sentences that still broke the constraint were artifacts of annotation errors. For these sentences, an additional treeification procedure was applied to break all cycles and ensure the possibility of reaching the root from any token.

For sentences with no rooted token (and at least one obligatory cycle), the rightmost token that was part of a cycle was considered the sentential head and connected to the sentence root with the dependency relation ROOT. For any other cycles, the token with the most dependents in the cycle was considered a clausal head and connected to the sentential head, keeping its original dependency relation. Finally, if a sentence had multiple rooted tokens, the rightmost rooted token with a VERB category (or, in the absence of rooted VERB tokens, simply the rightmost rooted token) was considered the sentential head, and the other rooted tokens were connected to that token with the dependency relation CONJ.

3 The IMST Treebank

The IMST Treebank is a Turkish dependency treebank of well-edited sentences from a wide range of domains, fully annotated for morphological analyses and dependency relations. The treebank underwent substantial changes since its unofficial conception in 2014 and was at version 1.3 when it was officially released.[4]

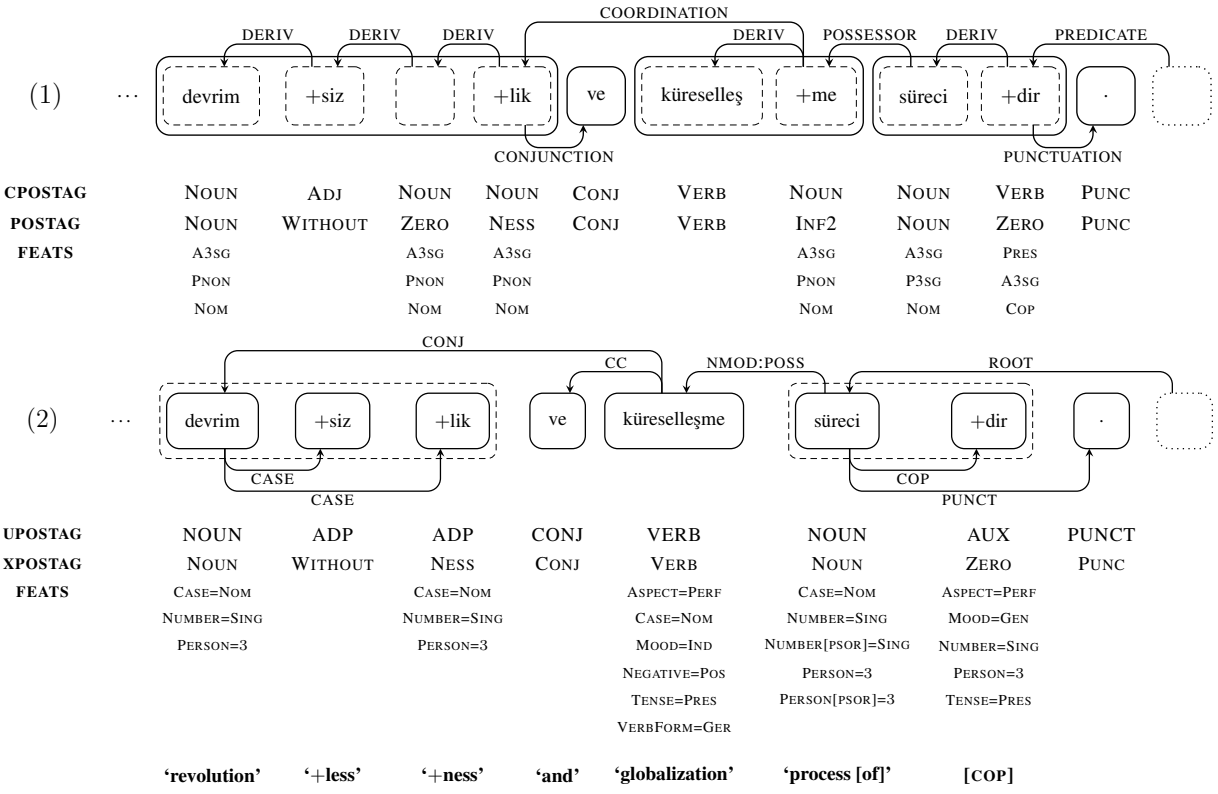


Figure 1: An example of a partial sentence, "*...devrimsizlik ve küreselleşme sürecidir.*" ("**...is the process of revolutionlessness and globalization.**"), before (1) and after (2) the conversion, extracted from the IMST and IMST-UD treebanks.

The IMST Treebank was annotated using its own annotation framework, which is based on that of the METU-Sabancı Treebank and radically different from the UD framework. Figure 1 compares a partial sentence from the IMST Treebank before and after the conversion. The + sign is used for convenience as a suffix marker, and does not actually occur in the treebank. The token enclosures denote either IG sets (in the original treebank sentences) or multi-word tokens (in converted sentences). As shown in the example, these multi-word groups were converted to a head-first typology, whereas coordination structures remained head-final. This is because the final token in a coordination structure always retains all inflection, whereas suffixes shared by all the tokens may be dropped in the others.

Table 9 presents a selection of comparative statistics, including the total numbers of sentences, tokens and dependency counts as well as the counts of unique part-of-speech tags, morphological features and dependency relations for the baseline and converted versions of the IMST Treebank, as a preamble to the parsing tests described in Section 4. We use the treebank's version 1.3.1 as the baseline for the UD conversion. For this reason, the statistics provided in this section are slightly different from those given in the IMST Treebank's original publication (Sulubacak et al., 2016).

[4]The latest version of the treebank is available for research purposes on `http://tools.nlp.itu.edu.tr`

	IMST	IMST–UD
# *Sentences*	5635	5635
# *(Orthographic) Tokens*	56423	56423
# *(Syntactic) Words*	63072	58085
# *Dependencies*	56423 *(excl.* DERIV*)*	58085
	63072 *(incl.* DERIV*)*	
# *Projective Dependencies*	61849	55043
# *Non-projective Dependencies*	1223	3042
# *(Unique) Parts of Speech*	11	14
# *(Unique) Morphological Features*	47	67
# *(Unique) Dependency Relations*	16	29

Table 9: Comparative statistics for the IMST Treebank and the IMST-UD Treebank.

4 Evaluation

In this section, we present our statistical analysis on the parsing performances of the original and converted versions of the IMST Treebank.

4.1 Preliminaries

For our parsing tests, we employ the same MaltParser (Nivre et al., 2007) configuration as in many previous studies on the METU-Sabancı Treebank (Eryiğit, 2006; Eryiğit et al., 2008; Eryiğit et al., 2011; Sulubacak and Eryiğit, 2013) and the IMST Treebank (Sulubacak et al., 2016). In compliance with the parsing procedures used in the cited studies, we eliminate non-projective sentences from each training set, as this practice was shown to boost overall performance[5] (Eryiğit et al., 2008; Eryiğit et al., 2011).

In further accordance with the cited studies, we use the conventional labeled and unlabeled attachment scores as our evaluation metrics. Although both scores are essentially based on the ratio of correct predictions to all tokens, they differ in which predictions they accept as correct. While a correct prediction of the head token suffices for the unlabeled attachment score (UAS), the labeled attachment score (LAS) also requires the dependency relation to be correctly predicted. Furthermore, dependencies with the relation DERIV[6] are excluded from evaluation for the baseline version, as they are considered trivial.

4.2 Parsing Scores

The parsing scores given in Table 10 were calculated via ten-fold cross-validation on the baseline *(left)* and the UD *(right)* versions of the IMST Treebank. A comparison of the scores before and after the conversion to UD shows that there has been a noticeable improvement in the labeled attachment score, despite the consequential increase in the number of unique POS tags, morphological features and dependency labels, as previously shown in Table 9. However, there has been no apparent progress in the unlabeled attachment score. Since head indices had also been adjusted as part of the mapping procedure, the similarity in the scores is likely a favorable coincidence. Considering both scores, it is evident that the UD framework has been more accommodating for the IMST Treebank over the current parsing setup.

	IMST	IMST–UD
LAS	$75.4 \pm 0.2\%$	$77.1 \pm 0.2\%$
UAS	$83.8 \pm 0.3\%$	$83.8 \pm 0.2\%$

Table 10: Attachment scores.

[5] We tested this on the UD version of the IMST Treebank as well, and including non-projective sentences in training indeed caused a drop of 2.9 percentage points in the average labeled attachment score compared to training without non-projective sentences.

[6] The DERIV relation is used in the annotation framework of the original IMST Treebank to mark intra-token dependencies between morphosyntactic units. Each such unit depends on the next, and the rightmost unit is considered to be the head.

5 Conclusion

In this paper, we described our procedure for converting the morphological and syntactic tagset of the IMST Treebank to comply with the UD standard. In doing so, we presented a specific application of the UD guidelines to the annotation of parts of speech, morphological features and dependency relations in Turkish. We also introduced the IMST-UD Treebank, which was automatically converted from the IMST Treebank and became the first Turkish treebank to be in a UD release. We also evaluated the parsing performances on the IMST and IMST-UD treebanks and found that there is a noticeable improvement in parsing performances after conversion, which suggests that the UD framework is at least as viable for Turkish as the original annotation framework of the IMST Treebank.

Acknowledgements

We would like to thank Birsel Karakoç, Hüner Kaşıkara and Tuğba Pamay for their valuable insights and discussions, and the Uppsala University for kindly hosting a meeting for us during the initial stages of our effort. We would also like to offer our gratitude to our anonymous reviewers for meticulously proofreading our manuscript.

References

Nart B. Atalay, Kemal Oflazer, and Bilge Say. 2003. The annotation process in the Turkish Treebank.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X Shared Task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 149–164. Association for Computational Linguistics.

Çağrı Çöltekin. 2015. A grammar-book treebank of Turkish. In Markus Dickinson, Erhard Hinrichs, Agnieszka Patejuk, and Adam Przepiórkowski, editors, *Proceedings of the 14th workshop on Treebanks and Linguistic Theories (TLT 14)*, pages 35–49.

Çağrı Çöltekin. 2016. (when) do we need inflectional groups? In *Proceedings of The 1st International Conference on Turkic Computational Linguistics*, page (to appear).

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of 5th International Conference on Language Resources and Evaluation (LREC)*, volume 6, pages 449–454.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford Dependencies: a cross-linguistic typology. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC)*, Reykjavík, Iceland, May. European Language Resources Association (ELRA).

Gülşen Eryiğit and Kemal Oflazer. 2006. Statistical dependency parsing of Turkish. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 89–96, Trento, April.

Gülşen Eryiğit. 2006. *Dependency Parsing of Turkish*. Ph.D. thesis, Istanbul Technical University.

Gülşen Eryiğit, Joakim Nivre, and Kemal Oflazer. 2008. Dependency parsing of Turkish. *Computational Linguistics*, 34(3):357–389.

Gülşen Eryiğit, Tugay Ilbay, and Ozan Arkan Can. 2011. Multiword expressions in statistical dependency parsing. In *Proceedings of the 2nd Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL)*, Dublin, Ireland, October. Association for Computational Linguistics.

Dilek Zeynep Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2002. Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities*, 36(4):381–410.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13:95–135, 6.

Kemal Oflazer, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gökhan Tür. 2003. Building a Turkish treebank. In Anne Abeille, editor, *Building and Exploiting Syntactically-Annotated Corpora*. Kluwer Academic Publishers.

Kemal Oflazer. 1999. Dependency parsing with an extended finite state approach. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 254–260, College Park, Maryland, USA. Association for Computational Linguistics.

Kemal Oflazer. 2003. Dependency parsing with an extended finite-state approach. *Computational Linguistics*, 29(4):515–544.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*, pages 2089–2096.

Umut Sulubacak and Gülşen Eryiğit. 2013. Representation of morphosyntactic units and coordination structures in the Turkish dependency treebank. In *Proceedings of the 4th Workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL)*, pages 129–134, Seattle, Washington, USA, October. Association for Computational Linguistics.

Umut Sulubacak, Tuğba Pamay, and Gülşen Eryiğit. 2016. IMST: A revisited Turkish dependency treebank. In *Proceedings of the 1st International Conference on Turkic Computational Linguistics (TurCLing)*, Konya, Turkey, 3 April.

Reut Tsarfaty. 2013. A unified morpho-syntactic scheme of Stanford Dependencies. In *Proceedings of 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 578–584.

Daniel Zeman. 2008. Reusable tagset conversion using tagset drivers. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC)*, pages 213–218.

Creating Resources for Dialectal Arabic from a Single Annotation: A Case Study on Egyptian and Levantine

Ramy Eskander, Nizar Habash†, Owen Rambow and Arfath Pasha
Columbia University, USA
†New York University Abu Dhabi, UAE
rnd2110@columbia.edu, nizar.habash@nyu.edu
rambow@ccls.columbia.edu, arfath@ccls.columbia.edu

Abstract

Arabic dialects present a special problem for natural language processing because there are few Arabic dialect resources, they have no standard orthography, and they have not been studied much. However, as more and more written dialectal Arabic is found on social media, natural language processing for Arabic dialects has become an important goal. We present a methodology for creating a morphological analyzer and a morphological tagger for dialectal Arabic, and we illustrate it on Egyptian and Levantine Arabic. To our knowledge, these are the first analyzer and tagger for Levantine.

1 Introduction

The goal of this paper is to show how a particular type of annotated corpus can be used to create a morphological analyzer and a morphological tagger for a dialect of Arabic, without using any additional resources. A morphological analyzer is a tool that returns all possible morphological analyses for a given input word taken out of any context. A morphological tagger is a tool that identifies the single morphological analysis which is correct for a word given its specific context in a text. We illustrate our work using Egyptian Arabic and Levantine Arabic. Egyptian Arabic is in fact relatively resource-rich compared to other Arabic dialects, but we use a subset of available data to simulate a resource-poor dialect.

For Arabic and its dialects, we are faced with a set of well-known challenges (Habash, 2010): they have a rich morphology, the orthography encourages ambiguity, and the dialects do not have standard orthographies (and are generally not well documented linguistically). However, we can also exploit similarities among the dialects and between Dialectal Arabic (DA) and Modern Standard Arabic (MSA), and in fact the writing system, while encouraging ambiguity, also reduces the differences between these variants.

The approach we present in this paper is based on the morphological annotation of a text corpus. The annotator provides for each word a conventional orthography, a segmentation, a set of features, and a lemma. We use this information to hypothesize unseen morphological forms, and use all forms to build an analyzer for the new dialect. Because of the close relationship among the variants of Arabic, we also make use of an existing analyzer for MSA, and (in the case of Levantine), an existing Egyptian analyzer. The resulting analyzer is then used in a tagger, which uses the annotated corpus to learn classifiers that choose among all possible analyses. Crucially, we do not require the corpus to be fully annotated, allowing the annotator to concentrate on the most frequent words. We are currently annotating five more dialects with this sort of corpus, with initial corpora available for two dialects (Al-Shargi et al., 2016).

The primary contribution of this paper is that we describe a methodology for creating morphological analyzers and taggers for any Arabic dialect. We show the effectiveness of our approach by measuring performance based on training corpora of different sizes. A secondary contribution of this paper is that we present new resources for Levantine Arabic (a morphological analyzer and a morphological tagger), which to our knowledge are the first of their kinds.

While we restrict our attentions to Arabic and its dialects, we believe our approach may be relevant to other situations in which we face a large number of related low-resource languages or language variants.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3455–3465, Osaka, Japan, December 11-17 2016.

This paper is structured as follows: Section 2 presents related work. Section 3 presents the data we use. Section 4 discusses the creation of morphological analyzers, and Section 5 the morphological taggers. Section 6 evaluates the morphological analysis and tagging. Finally, Section 7 concludes and outlines future plans.

2 Related Work

There has been a considerable amount of research on MSA morphological analysis, disambiguation, part-of-speech (POS) tagging, tokenization, lemmatization and diacritization, see for example (Habash and Rambow, 2005; Zitouni et al., 2006; Diab et al., 2007; Pasha et al., 2014).

In the early efforts on DA processing, researchers focused on exploiting MSA resources and tools. Duh and Kirchhoff (2005) adopted a minimally supervised approach that requires raw data from several DAs, and an MSA morphological analyzer. They reported a POS accuracy of $\sim 71\%$ on a coarse-grained POS tagset (17 tags). Similarly, Chiang et al. (2006) were the first to attempt to do parsing on Arabic dialects using MSA training data. Other notable efforts to create dialectal morphological analyzers using MSA morphological analyzers include work reported by Abo Bakr et al. (2008) and Salloum and Habash (2011).

In the last few years, an important shift in the research on DA processing occurred, as researchers started to create resources that target DA and focused less on exploiting MSA resources. This can be seen in the rise of dialectal corpora and annotations (Gadalla et al., 1997; Diab et al., 2010; Al-Sabbagh and Girju, 2012b; Mohamed et al., 2012; Maamouri et al., 2014; Bouamor et al., 2014; Jarrar et al., 2014; Masmoudi et al., 2014; Smaïli et al., 2014; Voss et al., 2014; Khalifa et al., 2016).

In the context of morphological analysis and tagging for Arabic dialects, recent efforts focused principally on Egyptian Arabic (Mohamed et al., 2012; Al-Sabbagh and Girju, 2012a; Habash et al., 2012b; Habash et al., 2013). Mohamed et al. (2012) annotated a small corpus of Egyptian Arabic for morphological segmentation and learned tokenization models using memory-based learning (Daelemans and van den Bosch, 2005). Their system achieves a 91.90% accuracy on the task of morpheme-segmentation. Al-Sabbagh and Girju (2012a) describe a supervised tagger for Egyptian Arabic social media corpora using transformation-based learning (Brill, 1995). They report 87.6% on POS tagging. Finally, we previously created a morphological analyzer for Egyptian (Habash et al., 2012b) using the lexicon of Kilany et al. (2002); then we used this analyzer to create a morphological tagger (Habash et al., 2013). Our previous work used rich Egyptian-specific resources — the lexicon derived from the CallHome corpus for Egyptian Arabic (Kilany et al., 2002) and the Egyptian Arabic Treebank (Maamouri et al., 2014). In contrast, we now wish to explore how much can be done with a small annotation effort. In both our previous work and our new one, we use an analyze-and-choose approach to morphological tagging, following the work of Hajič (2000) (also used by Habash and Rambow (2005) for MSA). We also compare against our previous work in our evaluation.

3 Data

3.1 Orthography

Arabic dialects do not have a standard orthography. This is a big challenge to the annotation process as it allows the coexistence of uninteresting orthographic variations. To address this challenge, we previously developed the *Conventional Orthography for Dialectal Arabic* (CODA) (Habash et al., 2012a). The CODA choices aim at reducing differences between variants (DA and MSA) when possible while maintaining the distinctive morphological inventories of the different variants. The first CODA specifications were developed for Egyptian Arabic (henceforth EGY) and utilized in the EGY corpus which we also use (Maamouri et al., 2012). The EGY CODA guidelines were extended to Levantine Arabic (henceforth LEV) by the creators of the LEV corpus we use (Jarrar et al., 2014). Since the LEV corpus was annotated without diacritics, all diacritics were also stripped from the EGY corpus for the study we present in this paper. The only exception is that lemmas are represented using diacritics in the corpora for both dialects so that fine-grained distinctions between different lexemes can be made.

Word	CODA	Lemma	Gloss	BW Tag	POS	POS5	Stem
سألو sÂlw	سأله sÂlh	saÂal	ask	PV+PVSUFF_SUBJ:3MS+PVSUFF_DO:3MS	verb	VRB	sÂl
ابوه Abwh	ابوه Abwh	Ab	father	NOUN+POSS_PRON_3MS	noun	NOM	Ab
ليش lyš	ليش lyš	layš	why	INTERROG_ADV	adv_interrog	PRT	lyš
اتأخرت AtÂxrt	اتأخرت AtÂxrt	AitÂax∼ar	be late	PV+PVSUFF_SUBJ:2MS	verb	VRB	AtÂxr
لهلوكت lhlwkt	لهالوقت lhAlwqt	waqt	time	PREP+DEM_PRON+DET+NOUN	noun	NOM	wqt
؟ ?	؟ ?	?	?	PUNC	punc	PNX	?

Table 1: An example Levantine sentence سألو ابوه ليش اتأخرت لهلوكت؟ *sÂlw Abwh lyš AtÂxrt lhlwkt?* 'His father asked him why he was late?'. The various columns are for the CODA spelling and different morphological features: lemma, gloss, Buckwalter POS tag, two reduced POS tags and stem.

3.2 Egyptian Data

Corpus We use the Egyptian Arabic corpora developed by the Linguistic Data Consortium (LDC) (Maamouri et al., 2012; Eskander et al., 2013a). The corpora are morphologically annotated in a similar style to the annotations done at the LDC for MSA. Words are provided with contextually appropriate CODA form, lemmas, POS tags (Buckwalter, 2004), and English glosses.

We ran the EGY corpus through our EGY morphological analyzer CALIMA$_{\text{Egy}}$ (Habash et al., 2012b) in order to generate morphological features similar to the ones described in MADAMIRA (Pasha et al., 2014). In this process, we replaced the human-annotated corpus analysis by the closest CALIMA$_{\text{Egy}}$ analysis when it does not match any of the CALIMA$_{\text{Egy}}$ analyses for a given word. In the cases where CALIMA does not produce an analysis, a word is analyzed as a proper noun as a back-off. The back-off happened in 4.9% of the words. Finally, we removed diacritics from the corpus except for the lemmas as described above.

Data Splits We follow the corpus splits described in (Diab et al., 2013). The whole DEV (45K words) and TEST (46K words) are used, while only a portion of 135K words of TRAIN is utilized for the purpose of this work.

3.3 Levantine Data

Corpus We use the Curras Corpus of Palestinian Arabic developed at Birzeit University (Jarrar et al., 2014) as the LEV data. Palestinian Arabic is a sub-dialect of Levantine Arabic. The Corpus is around 57,000 words, half of which come from transcripts of a TV show and the rest of which comes from a mix of sources such as Facebook, Twitter, blogs and web forums.

The corpus is morphologically annotated in a similar style to the annotations in the EGY corpus. Procedurally, the developers of the Curras Corpus used MADAMIRA Egyptian (Pasha et al., 2014) to provide a starting point for the manual annotation. In this paper, we used a version of Curras that is only 82% manually annotated. Gaps in annotation exist only in the training corpus, but not in the development or test corpora, which are fully manually annotated. At the time of this publication, the Curras corpus is fully annotated.

Table 1 presents an example of a single Levantine sentence with the following associated annotations:[1] the CODA spelling, the lemma, its gloss, the full Buckwalter POS tag, two POS tags from different tagsets of Arabic and the stem. We discuss them further and evaluate against them in Section 6.

Data Splits We divided the provided corpus into three data sets; TRAIN, DEV and TEST, corresponding to 78% (45K words), 11% (6K words) and 11% (6K words) of the corpus size, respectively. DEV is selected to be the first 371 sentences of the Facebook threads in Curras, in addition to the first four documents from the TV show *Watan Aa Watar*, while TEST represents the rest of the Facebook threads and the next four documents from the TV show. The remaining part of the Curras corpus forms TRAIN.

[1] Arabic transliteration in this paper is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007).

4 Creating the Morphological Analyzers

We build two morphological analyzers, one for EGY and one for LEV, based on the corpus annotations. The two analyzers are built in the same manner based on completing the inflectional classes (ICs) generated from TRAIN. We follow our work on the automatic extraction of morphological lexicons from corpora to build morphological analyzers given the corpus annotations (Eskander et al., 2013b). However, the work described in (Eskander et al., 2013b) was performed only on verbs with no clitics. This would not allow us to build wide-coverage morphological analyzers. In this paper, we extend the work to cover words of any POS types, whether with clitics or without, so that we obtain complete morphological analyzers. We also apply the approach to Levantine for the first time.

For each POS, we collect all the possible morphosyntactic feature combinations found in TRAIN for words of that POS. This list of feature combinations defines the set of slots for inflected forms found in all inflectional classes (ICs) for lemmas with that POS. For each lemma in TRAIN and for all of its inflected forms found in TRAIN, we then create an inflectional class (IC) that lists the prefix, stem and suffix information in the appropriate slots in the IC. Typically, many slots remain empty for these ICs. The stem is represented as an abstraction, where the letters in the stem are replaced by placeholders, except for the letters ا *A*,أ *Â*, إ *Ǎ*, آ *Ā*, ؤ *ŵ*, ئ *ŷ*, ء *'*, و *w*, ي *y* and ى *ý*. This approach simulates in a simple manner the templatic (or "root and pattern") morphology of Semitic langauges. We then automatically complete the ICs to fill in the missing slots, and obtain all inflections of all the lexemes. Each IC in this set of complete ICs is associated with a set of compatible roots, such that each lexeme corresponds to a root and an IC. We do the completion process for each POS type separately (a total of 33 POS types), as every POS type has its own set of features with which it is compatible.

After completing the ICs, we create ALMOR databases (Habash, 2007) that represent the EGY and LEV morphological analyzers. The list of prefixes and suffixes are generated directly from the ICs. For the construction of the stems, each of the roots associated to an IC is plugged into the stem templates to generate the concrete stems. Finally, the compatibility tables are generated according to the correlation among the prefixes, stems and suffixes in the ICs.

We create morphological analyzers for different training sizes. In the case of EGY, the sizes of the analyzers are 5K, 15K, 45K and 135K, while the sizes of the LEV analyzers are 5K, 15K and 45K. Thus, we can evaluate the performance of the EGY and LEV analyzers at different sizes and compare them. Additionally, the big EGY analyzer of 135K words allows us to see the performance when more data is available.

The orthographic transformations between the input words and the surface-form annotations that appear in TRAIN are added into the analyzers as extensions. This allows the analyzers to convert a input in spontaneous orthography that is not in CODA into a CODA-compliant form that the analyzers know how to handle. The orthographic extensions are added for each of the prefix, stem and suffix entries, separately. However, a transformation that only appears once in TRAIN is omitted. This avoids having over-generating extensions that are due to infrequent typos.

If an input word is not given an analysis by the analyzer, we perform a back-off to a proper-noun analysis. In this case, the lemma, CODA, and stem are given the form of the input word.

5 Creating the Morphological Taggers

The morphological taggers were created by extending MADAMIRA for the EGY data and for LEV. MADAMIRA (Pasha et al., 2014) is a system for morphological analysis and disambiguation of Arabic text. It utilizes a morphological analysis component, a feature modeling component and an analysis ranking component to produce a list of analyses for each word in a given sentence. The morphological analyzer returns a list of all possible analyses (independent of context) for each word. The feature modeling component applies classifiers to derive predictions for the word's morphological features in context. The analysis ranking component then scores each word analysis list based on how well each analysis agrees with the model predictions, and then sorts the analyses based on that score.

For the purpose of this paper, two sets of MADAMIRA systems were developed using EGY and LEV

analyzers and classifiers built from different sizes of data sets, as described in Section 4. We note that a MADAMIRA system already exists for EGY, which uses a large amount of training data. In this paper, we do not use that system, and artificially reduce the amount of training data to simulate a resource-poor dialect.

Each system is trained on the TRAIN sets described in 3. Tuning of the individual classifiers was conducted by randomly selecting about 10% of the total word volume from TRAIN to be used as a tuning set. The tuning set was used to generate a set of feature weights that are required by the Analysis Ranking component. The tuning set was later merged back into TRAIN and the classifiers were then trained using all the training data.

6 Evaluation

6.1 Components of the Evaluation

In our evaluations for both the morphological analyzer and the tagger, we evaluate for the following components of the output of the analyzer or tagger:

- **POS** is the core POS tag of the word. We use the stem tagset in MADAMIRA, whose size is 36.
- **POS5** is a reduced tag set: *NOM* (all nominals including adjectives and adverbs), *PROP* (proper nouns), *VRB* (verbs), *PRT* (all particles), and *PNX* (punctuation). This tagset is a variant of the Columbia Arabic Treebank tagset, which is based on traditional Arabic grammar (Habash et al., 2009).
- **Lemma** is the fully diacritized lemma.
- **CODA** is the undiacritized conventional spelling of the input word with normalized Alifs (Habash et al., 2012a).
- **Stem** is the undiacritized stem of the word with normalized Alifs. The evaluation of this component represents the ability of the analyzer or tagger to segment a word into three parts, corresponding to all prefixes conjoined, a stem, and all suffixes conjoined.
- **ALL** represents the conjunction of all five preceding metrics, i.e., they all need to be correct in the same answer.

We describe the specific evaluations for the analyzers and for the taggers in the next subsections.

6.2 Evaluating the Analyzers

We now present the evaluation of the dialectal morphological analyzers created from the annotated corpora and compare them to existing state-of-the-art analyzers. Tables 2 and 3 present the results on DEV for EGY and LEV, respectively.

Metrics We use several evaluation metrics to measure the effectiveness of the analyzers. First is the **Analyzer Token Recall**, which measures for each token and for each analysis criterion whether the hand-tagged analysis is generated by the morphological analyzer (possibly among others). It is an upper limit on our ability to correctly tag a word. As the name implies, this metric counts all tokens, not just unique types. We apply this metric to all components listed in Section 6.1. The second metric is **OOV**, which represents the cases that are not recognizable by the analyzer, i.e., out of vocabulary. Finally, we present the number of **Analyses per Word**. This is a measure of the degree of ambiguity that should be considered together with the other two (recall and OOV). Overall we want the recall to be high, but we want the ambiguity and OOV rates to be low. Recall that if a word is OOV, a proper-noun back-off analysis is assigned, where the lemma, CODA and stem are given the form of the input word.

Systems For both EGY and LEV, we compare the use of the SAMA analyzer (Graff et al., 2009) (our MSA baseline); a rule-based dialect-affix extended version of SAMA (SAMA$_{ext}$) based on the work of Salloum and Habash (2014); and a combination of CALIMA$_{\text{Egy}}$ with SAMA$_{ext}$. For EGY, the latter is a state-of-the-art comparison point that we do not expect to beat in all of the metrics since it was carefully

developed over years, and using more data than we are. For LEV, we expect CALIMA$_{\text{Egy}}$+SAMA$_{ext}$ to be a very good baseline (because of the similarities between EGY and LEV), but we expect to improve on it.

For both EGY and LEV, we compare different training data sizes, namely 5K, 15K and 45K words. For EGY only, we add an extra step of 135K words since more data is available. For each training size, we compare three settings: (a) a **Lookup** baseline that assumes no learning by paradigm completion, (b) a **ParaFill** setting, which uses paradigm completion as discussed in Section 4, and (c) a combination of **ParaFill** with additional pre-existing resources. The additional resources we use for **ParaFill** differ by dialect. In the case of EGY, we only use SAMA$_{ext}$ (since we are using EGY to simulate a low-resource dialect, we cannot use CALIMA$_{\text{Egy}}$); but we use the CALIMA$_{\text{Egy}}$ analyzer and also SAMA$_{ext}$ for LEV (since for dialects other than EGY, we can always make use of the richer resources available for EGY).

		Analyzer Token Recall							
System		**POS**	**POS5**	**Lemma**	**CODA**	**Stem**	**All**	**OOV**	$\frac{Analyses}{Word}$
SAMA		79.8	97.9	62.2	94.7	85.3	57.1	8.4	11.1
SAMA$_{ext}$		83.2	98.3	64.3	95.8	87.9	58.9	5.0	14.7
CALIMA$_{\text{Egy}}$+SAMA$_{ext}$		94.6	99.5	86.9	97.3	94.8	81.9	1.7	22.0
Train 5K	Lookup$_{\text{Egy}}$	60.6	81.1	57.6	89.9	67.9	55.9	43.2	1.0
	ParaFill$_{\text{Egy}}$	74.5	87.9	69.9	92.6	80.2	67.2	26.1	2.2
	ParaFill$_{\text{Egy}}$+SAMA$_{ext}$	91.9	98.9	83.8	97.1	93.6	79.9	3.1	12.2
Train 15K	Lookup$_{\text{Egy}}$	69.9	86.2	67.6	91.6	74.7	65.7	32.9	1.3
	ParaFill$_{\text{Egy}}$	85.3	93.8	81.5	95.0	88.3	78.7	14.8	3.9
	ParaFill$_{\text{Egy}}$+SAMA$_{ext}$	93.6	99.3	88.4	97.4	94.6	85.1	2.4	13.9
Train 45K	Lookup$_{\text{Egy}}$	78.9	91.2	77.1	93.3	81.8	75.5	23.1	1.8
	ParaFill$_{\text{Egy}}$	91.9	97.4	89.2	96.6	93.4	86.6	7.9	6.9
	ParaFill$_{\text{Egy}}$+SAMA$_{ext}$	94.8	99.6	91.3	97.6	95.5	88.5	1.9	16.9
Train 135K	Lookup$_{\text{Egy}}$	85.7	94.5	84.0	94.4	87.1	82.7	15.9	2.4
	ParaFill$_{\text{Egy}}$	94.5	98.7	92.6	97.1	95.4	90.1	4.6	10.2
	ParaFill$_{\text{Egy}}$+SAMA$_{ext}$	95.4	99.8	93.1	97.8	96.2	90.5	1.5	20.1

Table 2: EGY Morphological Analysis Recall on DEV

		Analyzer Token Recall							
System		**POS**	**POS5**	**Lemma**	**CODA**	**Stem**	**All**	**OOV**	$\frac{Analyses}{Word}$
SAMA		77.7	92.7	72.5	91.5	87.2	62.1	8.7	9.3
SAMA$_{ext}$		81.2	93.5	74.9	92.7	89.6	64.0	6.0	12.3
CALIMA$_{\text{Egy}}$+SAMA$_{ext}$		86.8	94.8	87.6	93.4	92.5	77.4	3.9	17.3
Train 5K	Lookup$_{\text{Lev}}$	44.7	69.6	46.3	84.1	59.0	43.6	54.1	0.6
	ParaFill$_{\text{Lev}}$	57.4	76.5	57.5	87.9	71.4	52.9	38.8	1.2
	ParaFill$_{\text{Lev}}$+CALIMA$_{\text{Egy}}$+SAMA$_{ext}$	90.3	95.6	91.3	94.9	94.8	84.1	3.6	13.2
Train 15K	Lookup$_{\text{Lev}}$	54.0	75.2	55.1	85.5	65.6	52.5	44.7	0.8
	ParaFill$_{\text{Lev}}$	68.4	83.4	67.3	89.9	78.6	62.6	26.9	2.2
	ParaFill$_{\text{Lev}}$+CALIMA$_{\text{Egy}}$+SAMA$_{ext}$	91.1	95.9	91.7	95.1	94.8	85.3	3.5	14.2
Train 45K	Lookup$_{\text{Lev}}$	70.1	85.7	70.2	88.8	75.7	67.9	28.2	1.1
	ParaFill$_{\text{Lev}}$	79.9	90.6	78.9	93.2	86.5	74.5	15.4	4.0
	ParaFill$_{\text{Lev}}$+CALIMA$_{\text{Egy}}$+SAMA$_{ext}$	92.3	96.5	92.8	95.7	95.4	87.0	3.1	16.0

Table 3: LEV Morphological Analysis Recall on DEV

Results For DEV, for both EGY and LEV, as expected, among the pre-existing systems, SAMA$_{ext}$ outperforms SAMA; and CALIMA$_{\text{Egy}}$+SAMA$_{ext}$ does best of the three options. Also, across all training sizes, paradigm completion outperforms lookup; and using a combination of paradigm completion with a pre-existing state-of-the-art system for MSA (in the case of EGY) or for MSA and EGY (in the case of LEV) does best in terms of token recall. There is of course no guarantee that the tagger later on will select the analysis correctly: while we see that the search space is expanding as needed, we also see that as the training size increases and as additional resources are added, the number of analyses per word increases, adding more ambiguity for the tagger to select from.

System		All	OOV	$\frac{Analyses}{Word}$
SAMA		58.5	7.7	11.3
SAMA$_{ext}$		60.3	4.3	14.7
CALIMA$_{\text{Egy}}$+SAMA$_{ext}$		82.8	1.6	21.7
Train 5K	Lookup$_{\text{Egy}}$	54.4	45.8	0.9
	ParaFill$_{\text{Egy}}$	65.8	28.8	2.1
	ParaFill$_{\text{Egy}}$+SAMA$_{ext}$	80.2	2.7	12.0
Train 15K	Lookup$_{\text{Egy}}$	63.5	36.5	1.2
	ParaFill$_{\text{Egy}}$	77.0	17.3	3.8
	ParaFill$_{\text{Egy}}$+SAMA$_{ext}$	84.7	2.1	13.7
Train 45K	Lookup$_{\text{Egy}}$	72.9	27.2	1.7
	ParaFill$_{\text{Egy}}$	84.7	9.8	6.7
	ParaFill$_{\text{Egy}}$+SAMA$_{ext}$	87.9	1.7	16.6
Train 135K	Lookup$_{\text{Egy}}$	80.5	19.5	2.3
	ParaFill$_{\text{Egy}}$	89.0	5.5	9.8
	ParaFill$_{\text{Egy}}$+SAMA$_{ext}$	90.2	1.3	19.7

Table 4: EGY Analyzer Recall on TEST

System		All	OOV	$\frac{Analyses}{Word}$
SAMA		61.7	9.6	9.2
SAMA$_{ext}$		63.4	7.0	12.1
CALIMA$_{\text{Egy}}$+SAMA$_{ext}$		77.7	4.7	17.2
Train 5K	Lookup	43.3	54.2	0.6
	ParaFill$_{\text{Lev}}$	50.8	41.1	1.2
	ParaFill$_{\text{Lev}}$ +CALIMA$_{\text{Egy}}$+SAMA$_{ext}$	84.0	4.4	13.4
Train 15K	Lookup	51.5	45.6	0.8
	ParaFill$_{\text{Lev}}$	61.3	28.6	2.2
	ParaFill$_{\text{Lev}}$ +CALIMA$_{\text{Egy}}$+SAMA$_{ext}$	85.1	4.2	14.4
Train 45K	Lookup	66.7	29.3	1.1
	ParaFill$_{\text{Lev}}$	73.9	16.0	4.0
	ParaFill$_{\text{Lev}}$ +CALIMA$_{\text{Egy}}$+SAMA$_{ext}$	87.0	3.7	16.2

Table 5: LEV Analyzer Recall on TEST

		DEV						TEST					
System		**POS**	**POS5**	**Lemma**	**CODA**	**Stem**	**All**	**POS**	**POS5**	**Lemma**	**CODA**	**Stem**	**All**
MADAMIRA-MSA		71.0	78.6	55.2	91.0	82.3	48.4	72.4	79.9	56.6	91.1	83.0	49.5
MADAMIRA-EGY		86.6	91.4	71.8	94.1	89.1	63.8	86.3	91.6	72.2	94.1	88.8	64.2
Train 5K	ParaFill$_{\text{Egy}}$	70.8	74.7	65.1	91.3	77.3	59.5	68.6	72.5	63.8	90.8	75.4	58.0
	ParaFill$_{\text{Egy}}$+SAMA$_{ext}$	70.6	74.6	65.0	91.4	77.2	59.4	68.5	72.4	63.7	91.0	75.2	58.0
Train 15K	ParaFill$_{\text{Egy}}$	76.7	82.8	71.3	90.0	84.5	62.9	74.9	80.9	70.1	89.9	82.7	61.6
	ParaFill$_{\text{Egy}}$+SAMA$_{ext}$	80.6	87.4	72.8	91.5	86.7	64.0	80.4	87.6	72.6	91.6	86.0	63.8
Train 45K	ParaFill$_{\text{Egy}}$	84.3	88.8	74.6	92.5	89.4	67.8	82.6	87.2	73.4	92.4	87.5	66.6
	ParaFill$_{\text{Egy}}$+SAMA$_{ext}$	84.3	89.6	74.9	92.4	89.2	67.4	83.8	89.5	74.3	92.4	88.7	66.7
Train 135K	ParaFill$_{\text{Egy}}$	85.9	91.7	76.1	94.4	90.4	68.3	85.3	91.3	75.6	94.4	89.0	67.8
	ParaFill$_{\text{Egy}}$+SAMA$_{ext}$	84.0	91.4	76.6	94.5	90.2	66.9	83.8	91.5	76.3	94.7	89.3	66.4

Table 6: EGY Tagger Results on DEV and TEST

For some metrics, such as CODA and POS5, the baseline is rather high. Our EGY system trained on 45K words in conjunction with SAMA$_{ext}$ beats the highly engineered CALIMA$_{\text{Egy}}$ system on all metrics. In the case of LEV, our baselines are lower and we can incorporate CALIMA$_{\text{Egy}}$ as a pre-existing system. Our best system beats the best baseline on all metrics starting with 5K training data. We now discuss the performance on the harsh All metric, which is really the best indicator of quality overall, in more detail. For EGY, using paradigm completion on only 5K training data improves above the SAMA and SAMA$_{ext}$ baselines. By 15K we are able to beat the state-of-the-art system CALIMA$_{\text{Egy}}$+SAMA$_{ext}$ and consistently provide less ambiguity than it. This is due to the power of paradigm completion, which adds thousands of unseen inflected forms. We also get a very similar pattern for Levantine, where the error rate in All token recall can be cut by 30% against the high CALIMA$_{\text{Egy}}$+SAMA$_{ext}$ baseline by using a combination of paradigm completion and existing systems at the 5K level; and by 42% at the 45K level.

Tables 4 and 5 present the TEST results for EGY and LEV, respectively (only showing the All results). The results on TEST have the same pattern as those on DEV.

6.3 Evaluating the Taggers

We now discuss the performance of the morphological taggers that we build.

Metrics We use a single metric, **Accuracy**, where we ask whether our predicted result is correct. The evaluation was conducted across all the components described in 6.1 at the word token level.

Systems For each of EGY and LEV, we trained two sets of MADAMIRA systems using classifiers built from different sizes of data sets as described in Section 5. The two sets differed in the underlying

		DEV						TEST					
System		**POS**	**POS5**	**Lemma**	**CODA**	**Stem**	**All**	**POS**	**POS5**	**Lemma**	**CODA**	**Stem**	**All**
MADAMIRA-MSA		68.6	75.2	64.5	87.7	83.3	50.4	67.7	74.6	64.6	87.2	82.2	50.3
MADAMIRA-EGY		75.9	84.6	65.8	88.6	84.1	53.3	75.3	84.2	65.1	88.8	83.8	52.7
Train 5K	$\text{ParaFill}_{\text{Lev}}$	55.8	57.6	53.1	86.0	68.8	46.2	54.0	55.7	51.3	86.1	68.5	45.2
	$\text{ParaFill}_{\text{Lev}}$++	74.6	85.0	70.7	89.3	84.8	55.6	74.7	85.1	69.6	88.4	84.2	55.5
Train 15K	$\text{ParaFill}_{\text{Lev}}$	65.0	68.4	60.6	86.5	75.9	53.0	64.4	67.5	59.1	86.4	75.8	52.3
	$\text{ParaFill}_{\text{Lev}}$++	78.2	85.9	74.6	89.9	85.6	60.6	78.5	85.9	74.0	90.0	86.1	61.1
Train 45K	$\text{ParaFill}_{\text{Lev}}$	75.7	80.2	69.2	89.9	83.7	61.6	75.6	79.7	69.7	89.7	83.7	62.9
	$\text{ParaFill}_{\text{Lev}}$++	81.8	89.4	77.8	91.7	88.4	65.5	82.3	89.2	77.2	91.7	88.5	65.7

Table 7: LEV Tagger Results on DEV and TEST; $\text{ParaFill}_{\text{Lev}}$++ refers to $\text{ParaFill}_{\text{Lev}}$+$\text{CALIMA}_{\text{Egy}}$+$\text{SAMA}_{ext}$

morphological analyzer: the result of paradigm completion only ($\text{ParaFill}_{\text{Egy}}$ or $\text{ParaFill}_{\text{Lev}}$), or this system augmented with additional resources from other variants (MSA for EGY, MSA and EGY for LEV). These are the same analyzers evaluated above in Section 6.2. In addition, evaluations were conducted on MADAMIRA-MSA and MADAMIRA-EGY in order to compare the performance of the EGY and LEV systems to the standard systems. Note that for EGY, MADAMIRA-EGY represents a carefully engineered contrastive system, while for LEV, it represents a plausible alternative to dialect-specific efforts and thus a true baseline.

Results The results for EGY are shown in Table 6 for DEV and TEST. Looking at the DEV results, we see as expected that MADAMIRA-EGY provides a high level of performance across the components of the evaluation. We do not beat this system on POS even with 135K training data. However, we beat it on POS5 and CODA at 135K; 45K is enough training data to beat MADAMIRA-EGY on Stem, and 15K is enough on Lemma and All. We see that performance increases with more training data, as expected. Interestingly, the increase in performance from 45K to 135K (a substantial amount of additional annotation) is much smaller than the previous increments, for all evaluation criteria. We also see that the addition of SAMA_{ext} to the morphological analyzer helps only at the medium amount of 15K (on All); we suspect that this is because at 5K, there is not enough training data to counteract the increased ambiguity that comes from adding these additional analyses, while at larger amounts of training data, the analyzer based on paradigm completion alone generates sufficiently rich databases. Results are roughly similar on TEST.

The results for LEV are shown in Table 7 for DEV and TEST. For LEV, MADAMIRA-EGY represents a baseline. Our system beats this baseline even with only 5K training data, however only if we include +$\text{CALIMA}_{\text{Egy}}$+$\text{SAMA}_{ext}$ (and not on POS). In fact, the results for using the $\text{ParaFill}_{\text{Lev}}$+$\text{CALIMA}_{\text{Egy}}$+$\text{SAMA}_{ext}$ analyzer are consistently better than those for $\text{ParaFill}_{\text{Lev}}$ analyzer alone (the one we get from paradigm completion), though this effect is stronger at smaller training sizes. This shows that the tagger is able to make the right choice despite the steep increase in ambiguity (as seen in Table 3), for example from 2.2 analyses per word for $\text{ParaFill}_{\text{Lev}}$ to 14.2 for $\text{ParaFill}_{\text{Lev}}$+$\text{CALIMA}_{\text{Egy}}$+$\text{SAMA}_{ext}$ at 15K training data. Finally, we see again that more data helps, and we do not see a major flattening of the learning curve at 45K words yet. As with EGY, the results on TEST mirror the ones on DEV for LEV.

Error Analysis We conducted error analyses for the $\text{ParaFill}_{\text{Egy}}$ and $\text{ParaFill}_{\text{Lev}}$ taggers, trained on 45K-word TRAIN and tested on DEV. For EGY, 20.7% of the errors occur because the gold answer is not provided by the morphological analyzer. An additional 5.6% of the errors are back-off cases at the time of preparing the data where $\text{CALIMA}_{\text{Egy}}$ does not produce an answer. OOV cases contribute a further 2.6% of the errors. The rest of the errors are cases where the analyzer provides the correct analysis, but the tagger fails to pick it in context. For LEV, the absence of the gold answer in the analyzer and the OOV entries contribute to 21.7% and 6.6% of the errors, respectively. The other errors occur as the tagger fails to select the correct answer provided by the analyzer.

7 Conclusions and Future Work

This paper has presented a methodology for deriving a morphological analyzer and a morphological tagger for an Arabic dialect. We have shown that this can be done successfully, even with a small amount of data. The approach requires a single type of annotation: a morphological annotation on running text which identifies the normalized spelling, the segmentation, the morphological features, and the lemma for each word. Both the analyzer and the tagger evaluations show the importance of using the paradigm completion approach to creating full inflectional classes: we obtain important error reductions over using just the morphological information supplied in the annotation. Furthermore, we have shown that for Levantine, using Egyptian resources helps performance in both analysis and tagging, even though using such resources greatly increases ambiguity, thus making the tagging task harder.

This paper has also presented a morphological analyzer and a morphological tagger for Levantine. To our knowledge, these are the first of their kind. We plan on using the complete version of the Levantine corpus and then making these resources available.

Our error analysis showed that most of the errors in the tagger come from the tagger itself, not the analyzer. This is understandable, as usually taggers are trained on much larger corpora. In future work, we will investigate whether we can improve the performance of the tagger by training the classifiers on combinations of Levantine text with Egyptian and/or MSA tagged text. Furthermore, we will also apply this methodology to other Arabic dialects; we are currently preparing annotations in the same style for five additional dialects from across the Arab world (see (Al-Shargi et al., 2016) for Moroccan and Sanaani Yemeni), and we will follow exactly the same methodology as laid out in this paper.

Acknowledgment

This paper is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-12-C-0014. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA.

References

Hitham Abo Bakr, Khaled Shaalan, and Ibrahim Ziedan. 2008. A Hybrid Approach for Converting Written Egyptian Colloquial Dialect into Diacritized Arabic. In *The 6th International Conference on Informatics and Systems, INFOS2008*. Cairo University.

Rania Al-Sabbagh and Roxana Girju. 2012a. A supervised POS Tagger for Written Arabic Social Networking Corpora. In Jeremy Jancsary, editor, *Proceedings of KONVENS 2012*, pages 39–52. ÖGAI, September. Main track: oral presentations.

Rania Al-Sabbagh and Roxana Girju. 2012b. YADAC: Yet another Dialectal Arabic Corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 2882–2889.

Faisal Al-Shargi, Aidan Kaplan, Ramy Eskander, Nizar Habash, and Owen Rambow. 2016. Morphologically Annotated Corpora and Morphological Analyzers for Moroccan and Sanaani Yemeni Arabic. In *10th Language Resources and Evaluation Conference (LREC 2016)*.

Houda Bouamor, Nizar Habash, and Kemal Oflazer. 2014. A Multidialectal Parallel Corpus of Arabic. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Eric Brill. 1995. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics*, 21(4):543–565.

Tim Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. LDC catalog number LDC2004L02, ISBN 1-58563-324-0.

David Chiang, Mona Diab, Nizar Habash, Owen Rambow, and Safiullah Shareef. 2006. Parsing Arabic Dialects. In *Proceedings of EACL*, Trento, Italy.

Walter Daelemans and Antal van den Bosch. 2005. *Memory-Based Language Processing*. Studies in Natural Language Processing. Cambridge University Press, Cambridge, UK.

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2007. Automated Methods for Processing Arabic Text: From Tokenization to Base Phrase Chunking. In Antal van den Bosch and Abdelhadi Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Kluwer/Springer.

Mona Diab, Nizar Habash, Owen Rambow, Mohamed Altantawy, and Yassine Benajiba. 2010. COLABA: Arabic Dialect Annotation and Processing. In *LREC Workshop on Semitic Language Processing*, pages 66–74.

Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic Treebanks and Associated Corpora: Data Divisions Manual. *arXiv preprint arXiv:1309.5652*.

Kevin Duh and Katrin Kirchhoff. 2005. POS Tagging of Dialectal Arabic: a Minimally Supervised Approach. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, Semitic '05, pages 55–62, Ann Arbor, Michigan.

Ramy Eskander, Nizar Habash, Ann Bies, Seth Kulick, and Mohamed Maamouri. 2013a. Automatic Correction and Extension of Morphological Annotations. In *Proceedings of the 7th Linguistic Annotation Workshop & Interoperability with Discourse*, pages 1–10.

Ramy Eskander, Nizar Habash, and Owen Rambow. 2013b. Automatic Extraction of Morphological Lexicons from Morphologically Annotated Corpora. In *Proceedings of tenth Conference on Empirical Methods in Natural Language Processing*.

Hassan Gadalla, Hanaa Kilany, Howaida Arram, Ashraf Yacoub, Alaa El-Habashi, Amr Shalaby, Krisjanis Karins, Everett Rowson, Robert MacIntyre, Paul Kingsbury, David Graff, and Cynthia McLemore. 1997. CALLHOME Egyptian Arabic Transcripts. Linguistic Data Consortium, Philadelphia.

David Graff, Mohamed Maamouri, Basma Bouziri, Sondos Krouna, Seth Kulick, and Tim Buckwalter. 2009. Standard Arabic Morphological Analyzer (SAMA) Version 3.1. Linguistic Data Consortium LDC2009E73.

Nizar Habash and Owen Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 573–580, Ann Arbor, Michigan.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Habash, Reem Faraj, and Ryan Roth. 2009. Syntactic Annotation in the Columbia Arabic Treebank. In *Proceedings of MEDAR International Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Nizar Habash, Mona Diab, and Owen Rabmow. 2012a. Conventional Orthography for Dialectal Arabic. In *Proceedings of LREC*, Istanbul, Turkey.

Nizar Habash, Ramy Eskander, and Abdelati Hawwari. 2012b. A Morphological Analyzer for Egyptian Arabic. In *NAACL-HLT 2012 Workshop on Computational Morphology and Phonology (SIGMORPHON2012)*, pages 1–9.

Nizar Habash, Ryan Roth, Owen Rambow, Ramy Eskander, and Nadi Tomeh. 2013. Morphological Analysis and Disambiguation for Dialectal Arabic. In *Proceedings of the 2013 Conference of NAACL-HLT*, Atlanta, GA.

Nizar Habash. 2007. Arabic Morphological Representations for Machine Translation. In Antal van den Bosch and Abdelhadi Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Kluwer/Springer.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Jan Hajič. 2000. Morphological Tagging: Data vs. Dictionaries. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL'00)*, Seattle, WA.

Mustafa Jarrar, Nizar Habash, Diyam Akra, and Nasser Zalmout. 2014. Building a Corpus for Palestinian Arabic: a Preliminary Study. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 18–27, Doha, Qatar, October. Association for Computational Linguistics.

Salam Khalifa, Nizar Habash, Dana Abdulrahim, and Sara Hassan. 2016. A Large Scale Corpus of Gulf Arabic. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Portorož, Slovenia.

H. Kilany, H. Gadalla, H. Arram, A. Yacoub, A. El-Habashi, and C. McLemore. 2002. Egyptian Colloquial Arabic Lexicon. LDC catalog number LDC99L22.

Mohamed Maamouri, Ann Bies, Seth Kulick, Dalila Tabessi, and Sondos Krouna. 2012. Egyptian Arabic Treebank DF Parts 1-8 V2.0 - LDC catalog numbers LDC2012E93, LDC2012E98, LDC2012E89, LDC2012E99, LDC2012E107, LDC2012E125, LDC2013E12, LDC2013E21.

Mohamed Maamouri, Ann Bies, Seth Kulick, Michael Ciul, Nizar Habash, and Ramy Eskander. 2014. Developing an Egyptian Arabic Treebank: Impact of Dialectal Morphology on Annotation and Tool Development. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Abir Masmoudi, Mariem Ellouze Khmekhem, Yannick Esteve, Lamia Hadrich Belguith, and Nizar Habash. 2014. A Corpus and Phonetic Dictionary for Tunisian Arabic Speech Recognition. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Emad Mohamed, Behrang Mohit, and Kemal Oflazer. 2012. Annotating and Learning Morphological Segmentation of Egyptian Colloquial Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Istanbul.

Arfath Pasha, Mohamed Al-Badrashiny, Ahmed El Kholy, Ramy Eskander, Mona Diab, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *In Proceedings of LREC*, Reykjavik, Iceland.

Wael Salloum and Nizar Habash. 2011. Dialectal to Standard Arabic Paraphrasing to Improve Arabic-English Statistical Machine Translation. In *Proceedings of the First Workshop on Algorithms and Resources for Modelling of Dialects and Language Varieties*, pages 10–21, Edinburgh, Scotland.

Wael Salloum and Nizar Habash. 2014. ADAM: Analyzer for Dialectal Arabic Morphology. *Journal of King Saud University-Computer and Information Sciences*, 26(4):372–378.

Kamel Smaïli, Mourad Abbas, Karima Meftouh, and Salima Harrat. 2014. Building Resources for Algerian Arabic Dialects. In *15th Annual Conference of the International Communication Association Interspeech*.

Clare Voss, Stephen Tratz, Jamal Laoudi, and Douglas Briesch. 2014. Finding Romanized Arabic Dialect in Code-Mixed Tweets. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2249–2253, Reykjavik, Iceland, May. European Language Resources Association (ELRA). ACL Anthology Identifier: L14-1086.

Imed Zitouni, Jeffrey S. Sorensen, and Ruhi Sarikaya. 2006. Maximum Entropy Based Restoration of Arabic Diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 577–584, Sydney, Australia.

Multilingual Aliasing for Auto-Generating Proposition Banks

Alan Akbik
IBM Research Almaden
650 Harry Road, San Jose
CA 95120, USA
akbika@us.ibm.com

Xinyu Guan
Yale University
82-90 Wall Street, New Haven
CT 06520, USA
xinyu.guan@yale.edu

Yunyao Li
IBM Research Almaden
650 Harry Road, San Jose
CA 95120, USA
yunyaoli@us.ibm.com

Abstract

Semantic Role Labeling (SRL) is the task of identifying the predicate-argument structure in sentences with semantic frame and role labels. For the English language, the Proposition Bank provides both a lexicon of all possible semantic frames and large amounts of labeled training data. In order to expand SRL beyond English, previous work investigated automatic approaches based on parallel corpora to automatically generate Proposition Banks for new target languages (TLs). However, this approach heuristically produces the frame lexicon from word alignments, leading to a range of lexicon-level errors and inconsistencies. To address these issues, we propose to manually *alias* TL verbs to existing English frames. For instance, the German verb *drehen* may evoke several meanings, including "turn something" and "film something". Accordingly, we alias the former to the frame TURN.01 and the latter to a group of frames that includes FILM.01 and SHOOT.03. We execute a large-scale manual aliasing effort for three target languages and apply the new lexicons to automatically generate large Proposition Banks for Chinese, French and German with manually curated frames. We present a detailed evaluation in which we find that our proposed approach significantly increases the quality and consistency of the generated Proposition Banks. We release these resources to the research community.

1 Introduction

Semantic role labeling (SRL) is the task of labeling predicate-argument structure in sentences with shallow semantic information. The prominent labeling scheme for the English language is the Proposition Bank (Palmer et al., 2005), which provides a lexicon of possible *frames* for English verbs. Each frame corresponds to one semantic interpretation and comes with frame-specific *role* labels and descriptions. Over the past decade, large amounts of text data have been annotated based on these guidelines (Palmer et al., 2005; Hovy et al., 2006; Bonial et al., 2014). They enable the training of statistical SRL systems, which have proven useful for downstream applications such as information extraction (IE) (Fader et al., 2011), question answering (QA) (Shen and Lapata, 2007; Maqsud et al., 2014) and machine translation (Lo et al., 2013).

However, such manual efforts are known to be highly costly. Possible frames need to be manually determined, their roles individually described, and large volumes of text data annotated accordingly. For this reason, Proposition Banks do not exist for most languages.

Annotation projection. Recent research has explored the possibility of using annotation projection to automatically generate Proposition Banks from parallel corpora for new target languages (TL) (Padó and Lapata, 2009; Van der Plas et al., 2011; Akbik et al., 2015). This approach requires a large word-aligned corpus of English sentences and their TL translations. An English SRL system predicts semantic labels for the English sentences. These labels are then transferred along word alignments to automatically annotate the TL corpus. Recent work has shown that such auto-generated Proposition Banks can be used to train semantic role labelers for a wide range of languages (Akbik and Li, 2016).

Heuristically generated frame lexicon. However, a major drawback of such approaches is that the frame lexicon is heuristically produced from available alignments, which leads to a range of errors and inconsistencies. Consider the German verb *drehen*. In the English-German portion of the OPENSUB-

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3466–3474, Osaka, Japan, December 11-17 2016.

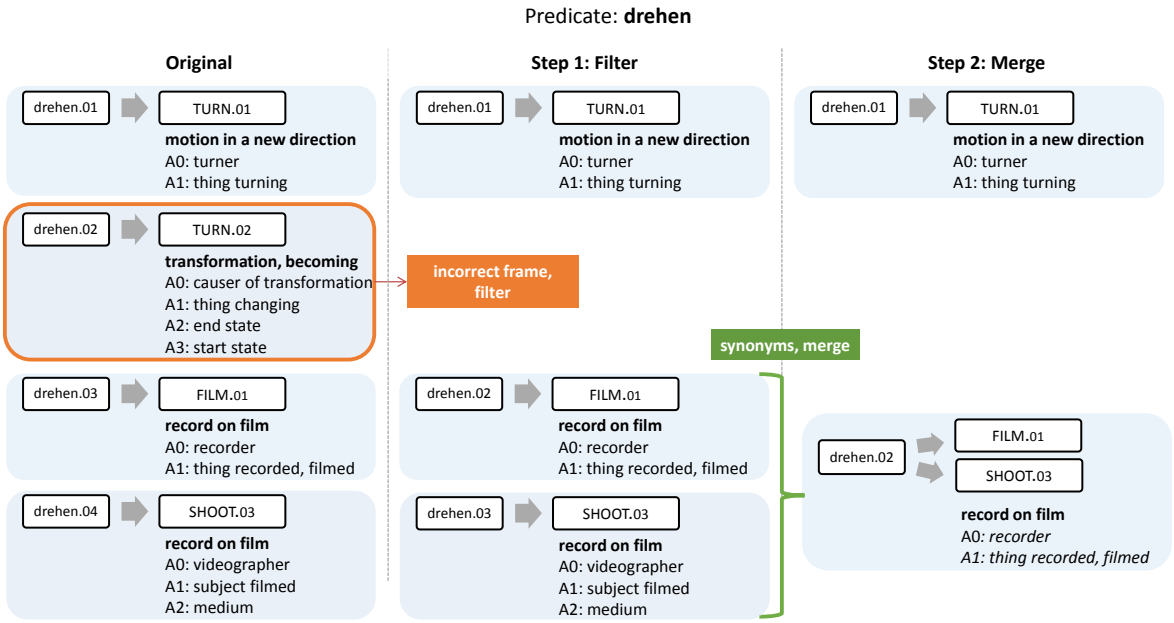


Figure 1: Illustration of merging and filtering steps over heuristically produced frame lexicon. This process reduces the number of distinct frames for the verb *drehen* from 4 to 2.

TITLES2016 parallel corpus (Lison and Tiedemann, 2016), this verb is aligned to many different English frames, four of which are illustrated in Figure 1. Previous annotation projection approaches treat each alignment as a separate and valid frame of the TL verb. In this example, it is therefore heuristically determined that *drehen* may evoke four separate frames, namely TURN.01, TURN.02, FILM.01 and SHOOT.03. See Figure 1 (left pane) for an illustration and explanation of the four frames. The example illustrates the two main problems in heuristically produced lexicons:

Incorrect frames The first problem is posed by errors in the frame lexicon. For instance, the German verb *drehen* cannot evoke the frame TURN.02 (transformation). The corresponding entry in the lexicon is therefore incorrect. This lexicon-level error has a significant impact on the generated Proposition Bank since every TL sentence with this annotation is incorrectly labeled. Therefore, as the middle pane in Figure 1 illustrates, we wish to filter out such lexicon-level errors.

Redundant frames The second problem is posed by redundancy that occurs if multiple entries for a TL verb are in fact synonyms. For instance, *drehen* is heuristically determined to evoke SHOOT.03 (record on film) and FILM.01 (record on film) as two separate meanings. However, the TL usages of *drehen* in these contexts are clearly identical. This lexicon-level error causes inconsistent annotation of the same semantics throughout the generated Proposition Bank. Therefore, as the right pane in Figure 1 illustrates, we wish to merge these two entries into a single entry comprising both frames.

In this paper, we propose to address these issues by manually curating incorrect alignments and grouping synonymous English frames with a process of filtering and merging as illustrated in Figure 1. With this approach, we effectively follow a process of *aliasing* TL verbs to English frames (Bonial et al., 2014; Jagfeld and van der Plas, 2015). Our goal is to remove lexicon-level errors and redundancies in order to generate higher quality TL Proposition Banks with consistent annotation and salient verb senses.

Contributions Our contributions are: 1) We propose a method for manually curating a heuristically determined frame lexicon and discuss our curation guidelines. 2) We execute our method over large-scale parallel data for three target languages (Chinese, French and German) to automatically generate Proposition Banks with curated frame lexicons. 3) We present an experimental evaluation in which we find that our proposed approach significantly increases the quality of automatically generated Proposition Banks and greatly reduces redundancy. 4) We analyze the verb coverage of the generated lexicons and

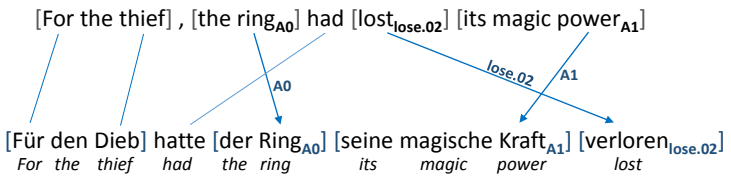


Figure 2: Example of annotation projection for an English-German sentence pair. English frame (LOSE.02) and role labels (A0, A1) are projected onto aligned German words.

conduct a comparison of our work against manual efforts to create Proposition Banks. 5) Finally, we release all resources to the research community for the training of multilingual SRL and the study of crosslingual semantics[1].

2 Related Work

Annotation projection Annotation projection takes as input a word-aligned parallel corpus of English sentences and their target language (TL) translations. An English semantic role labeler is used to predict semantic labels for the English sentences. These labels are then projected onto aligned TL words, automatically producing a TL corpus annotated with English frame and role labels. Refer to Figure 2 for illustration.

The use of annotation projection to train parsers for new languages was first introduced in the context of learning a PoS tagger (Yarowsky et al., 2001). Initial work on projecting semantic labels used FrameNet (Padó and Lapata, 2009), but subsequent work has focused on PropBank annotation due to its broader coverage and the availability of high quality semantic role labelers for English (Van der Plas et al., 2011; van der Plas et al., 2014). Recent work has focused on increasing the accuracy of projected labels by scaling up projection to larger corpora and retraining SRL models (Van der Plas et al., 2011; van der Plas et al., 2014) as well as using filtering techniques to block labels most likely affected by translation shift (Akbik et al., 2015). The latter found that the largest portion of errors in generated PropBanks results from incorrectly predicted labels for English sentences, which are then projected onto TL sentences, thereby propagating this error.

Consistency of the frame lexicon. However, so far, previous work has not investigated the overall correctness and consistency of the frame lexicon. All previous annotation projection efforts treat each distinct global alignment as a different sense of each verb, which, as argued in section 1, is not the case. In this paper, we specifically address this issue.

Aliasing Our work is similar to recent efforts in *aliasing*, in which existing English verb frames are reused for new types of frame evoking elements (Bonial et al., 2014). One advantage of this approach is that it reduces the effort required to define new frames. More importantly, this ensures consistent annotation for different syntactic elements that evoke the same semantics. An example is the verb frame FEAR.01 that is reused for the noun *fear* (as in *I have a fear of spiders*) and the adjective *afraid* (as in *I am afraid of spiders*). Recent work has proposed a method to automatically identify aliases for verbal complex predicates using a distributional model over parallel corpora (Jagfeld and van der Plas, 2015).

Unlike previous works that exclusively consider English, we consider a multilingual setting in which we alias English frames to verbs in other languages. We also allow multiple aliases for each TL verb. We pursue this approach not only to define a frame lexicon, but also to increase the quality and consistency of Proposition Banks generated with annotation projection.

3 Method

Our approach curates a frame lexicon of a Proposition Bank generated with annotation projection. The approach has two curation steps: filtering (section 3.1) and merging (section 3.2). We then make a final pass to add human readable explanations to the curated frame lexicon (section 3.3).

[1]Please contact the first author of this paper for access to the data.

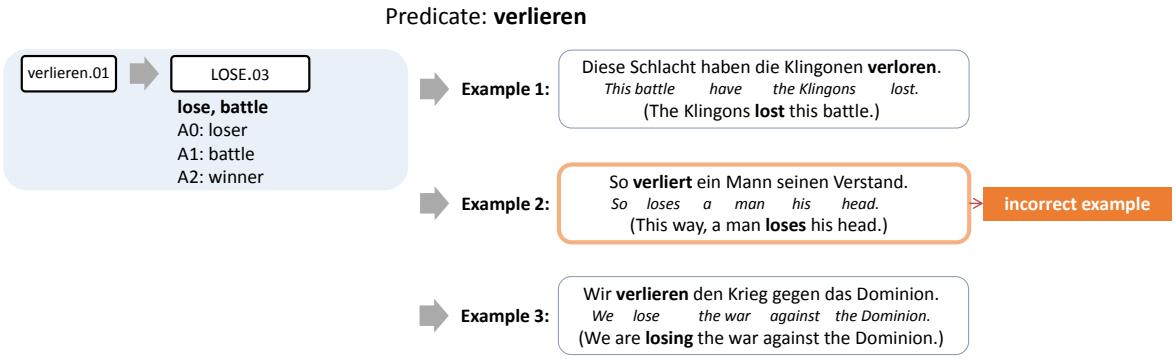


Figure 3: Filtering task example. A curator is shown one lexicon entry, consisting of a TL verb (*verlieren*) and an English frame (LOSE.03, as in *lose a battle*). In addition, the curator is shown five example sentences (only three displayed in this image). While examples 1 and 3 are correct, example 2 does not evoke the *lose a battle* sense.

3.1 First Curation Task: Filtering

The first task is to identify all incorrect frames for TL verbs. For each entry in the lexicon, curators must make a binary decision on whether the entry is correct or not. In order to make this decision, curators are presented with the following information: 1) The TL verb. 2) A description of the English frame and its roles. 3) A sample of TL sentences annotated with this frame. Refer to Figure 3 for illustration.

Given this information, curators must answer two questions (detailed below). If the answers to both questions are *yes*, the entry is considered valid. If one of the questions is answered with *no*, this entry must be removed from the lexicon.

Q1: Is the English frame a valid sense for the TL verb? The first question concerns the semantic validity of the English frame for the TL verb. To answer this question, curators only consider the English frame description. If the description refers to semantics that the TL verb clearly cannot evoke, the answer to this question is *no*. We encountered such a case in section 1 with the verb *drehen* that cannot evoke frame TURN.02. In all other cases, the answer is *yes*. Notably, we do not ask if an English frame is a perfect fit in semantics. At this point in the process, we are only interested in filtering out clear errors.

Q2: Does the TL verb accurately reflect the English frame description in the sample sentences? Even if an entry is valid in principle, it may still be subject to errors in practice. We find that some entries are correct judging from their description, but are never correctly detected in the corpus due to errors made by the English SRL. This problem disproportionally affects frames for which only limited English training data is available. For this reason, we require the curator to inspect a sample of 5 TL sentences per entry and determine whether they are correctly labeled. Refer to Figure 3 examples for both cases: Sentence 1 and 3 correctly invoke LOSE.03 (lose a battle), whereas Sentence 2 evokes LOSE.02 (lose an item). If a majority of example sentence is incorrectly labeled, this question must be answered with *no*.

3.2 Second Curation Task: Merging

The second task addresses the issue of redundancy caused by multiple entries for TL verbs that evoke the same semantics. For each pair of entries for the same TL verb, a curator must decide whether they are synonymous and need to be merged into a single entry. This task therefore effectively decides the semantic granularity of the lexicon entries for each TL verb.

We base merging decisions on the annotation guidelines of the English Proposition Bank, which specify that new frames need to be created to reflect different syntactic usages of a verb. In addition, new frames are created for broadly different meanings of a verb even if the syntactic subcategorization is the same (Palmer et al., 2005).

For each merging decision, we present curators with the following information: 1) The TL verb. 2) The two frames and their descriptions. 3) A set of TL sample sentences for each frame. The latter is the most important since sample sentences illustrate how the TL verb is used in related contexts when labeled with a specific frame. Refer to Figure 4 for example.

Given this information, curators must answer two questions (explained below). If the answer to any

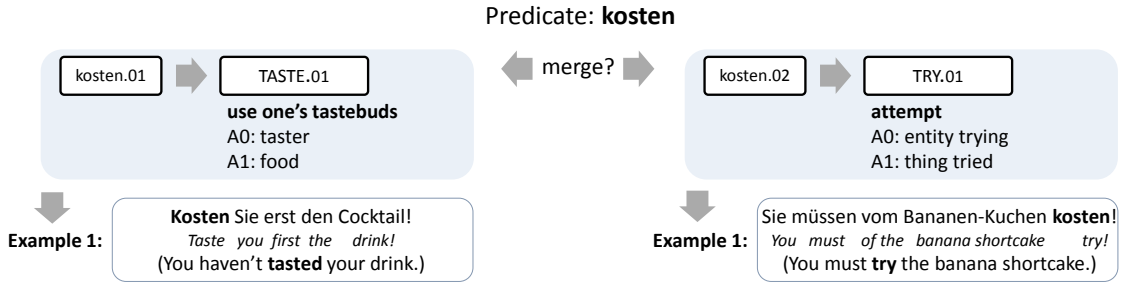


Figure 4: Information presented to curator for merge decisions: Two frames, their descriptions and example sentences.

of these questions is *no*, the two entries should not be merged.

Q1: Are the two entries usage-synonyms? We define *usage-synonyms* as target language usage synonyms. To illustrate the difference to regular synonyms, consider the example in Figure 4 in which curators must decide whether TASTE.01 and TRY.01 should be merged. While the two English frames are clearly not synonymous, their target language usages are. As lexicon entries for the German verb *kosten*, they are both solely used in the context of tasting food and are therefore usage-synonyms, illustrated by the sample sentences for each frame in Figure 4. If two entries are clearly not usage-synonyms, the answer to this question is *no*. In all other cases, the answer is *yes*.

Q2: Do the two entries represent syntactically different usages? We found a number of cases in which curators disagreed on whether two entries are usage-synonyms or not. An example of this were entries which partially overlapping semantics, such as the frame pair WRAP.01 (*enclose*) and PACK.01 (*fill, load*). To address this, we created a guideline to compare syntactic usage of TL verbs. We ask curators to build the *dictionary expansion* for both entries, which we define as the default syntactic expansion that one might find in a dictionary. An English example for the verb *turn* is *to turn something* for TURN.01 and *to turn into something* for TURN.02. However, we ask curators to create this form for TL verbs[2]. If the TL dictionary expansion is different, the answer to this question is *no*.

3.3 Final Pass: Dictionary Forms and Comments

After curators complete both tasks, we rerun annotation projection using the created lexicon to filter out incorrect entries and merge redundant entries. This produces a Proposition Bank with manually curated TL frames. To complete the curation process, we ask curators to inspect each entry in the dictionary and add comments or explanations, as well as dictionary expansions. This information is intended for human consumption. The purpose of this annotation is to make apparent the distinctions between multiple entries for the same TL verb and explain our aliasing decisions. The entire curation process thus produces an annotated Proposition Bank with salient, manually curated frames for each TL verb.

4 Evaluation

We present a set of large-scale experiments over three languages to evaluate our proposed approach. We first evaluate the curation process itself in terms of curator agreement scores, the required effort and the impact on the frame lexicons. We then present a detailed analysis of the auto-generated Proposition Banks in which we evaluate their quality in terms of precision, recall and F1 score both before and after curation. We further evaluate the curated Proposition Banks with regards to verb coverage and conduct a qualitative comparison against the manually created, official Chinese Proposition Bank (Xue and Palmer, 2005). Based on this analysis, we discuss the challenges and potential of combining large-scale annotation projection and manual aliasing to generate Proposition Banks for new target languages.

Experimental setup. We pre-generate Proposition Banks with the approach described in Akbik et. al (2015) for Chinese, French and German using parallel text from the OPENSUBTITLES2016

[2]In the example discussed in Figure 4, the dictionary expansion of *kosten* the same for both entries: *etwas kosten* ("*to taste/try something*"). This indicates that both entries take a direct object and are syntactically similar. They must therefore be merged only if the entries are also usage-synonyms.

		Proposition Bank				Evaluation		
Language	Parallel Corpus	Type	#Verbs	#Frames	#Sentences	P	R	F_1
Chinese	OpenSubtitles	projected	1,094	1,472	87,953	0.87	0.94	0.91
	(9 million sentences)	curated	942	1,003	68,829	0.93	0.96	**0.94**
French	OpenSubtitles	projected	1,323	2,249	175,636	0.82	0.94	0.87
	(15 million sentences)	curated	1,208	1,370	130,579	0.91	0.94	**0.93**
German	OpenSubtitles	projected	1,552	2,441	191,816	0.83	0.92	0.87
	(13 million sentences)	curated	1,532	1,717	150,949	0.90	0.93	**0.91**

Table 1: Annotation projection statistics for all three target languages: Number of parallel sentences available for each language, total number of covered verbs in auto-generated PropBanks as well as the total number of frames. The number of frames is higher because many verbs evoke more than one frame.

project (Lison and Tiedemann, 2016). This data is automatically mined from movie subtitles and thus covers a large array of topics (dramas, documentaries, science fiction etc.), reflecting verb usage in common speech. We execute annotation projection over 9-15 million parallel sentences for each target language, generating lexicons that cover over 1,000 verbs, respectively, as well as labeled corpora spanning over 100,000 sentences. For a full breakdown of our annotation projection numbers, refer to Table 1 (the uncurated PropBanks are marked as "PROJECTED").

These generated PropBanks are the starting point for our curation process. We had two persons each curate the French lexicon, while Chinese and German were curated by one person each. On average, for each language and each person about 60 person hours were required for the full curation process for all lexicon entries. All curators in our experiment have expert knowledge in semantic role labeling.

Using the curated lexicons, we generate the final generated Proposition Banks, marked as "CURATED" in Table 1. Following earlier evaluation practice, we randomly selected 100 sentences from each Proposition Bank before and after curation. We manually evaluated these sentences in order to get an understanding of precision, recall and F_1-score for the generated resources.

4.1 Evaluation Results

Refer to Table 1 for an overview of all three generated Proposition Banks before and after curation with our proposed process. We make a number of observations:

Curation significantly improves Proposition Bank quality. We find that incorporating the curated frame lexicons into annotation projection significantly boosts overall quality. For German, we estimate an F_1 of 0.91 (↑ 4pp), for French 0.93 (↑ 6pp), and for Chinese 0.91 (↑ 4pp). This indicates that a large number of errors are caused by incorrect entries in the frame lexicon, which can be handled globally at moderate effort using our proposed approach.

Curation reduces the amount of labeled data and covered verbs. We also note that filtering incorrect entries reduces the number of annotated sentences in the generated resource, since all affected projections are removed. For Chinese, French and German, a total of 18k, 20k and 36k sentences are affected by incorrect annotations and are therefore filtered out. Our approach therefore generates slightly smaller Proposition Banks with a higher overall quality. We also note that for some TL verbs all entries in the lexicon were deemed incorrect. These verbs are therefore no longer covered in the curated PropBanks. This reduces the amount of verbs included in the TL lexicons by 20 for German, 115 for French and 152 for Chinese, which seems to correspond to their linguistic distance to English: German, the closest relative to English, has the largest number of covered verbs while Chinese has the lowest.

Curation significantly reduces redundancy. We note that our approach significantly reduces the number of distinct frame entries for each language. Whereas in uncurated versions, every alignment is interpreted as a distinct verb frame, the filtering and merging process removes hundreds of incorrect and redundant entries. The curated Chinese, French and German PropBanks evoke 1,003 (↓ 32%), 1,370 (↓ 39%), and 1,717 (↓ 30%) frames respectively. We note that the highest difference is observed for French, for which the largest number of parallel sentences was available. It seems that greater amounts of parallel data lead to greater redundancies and more incorrect alignments.

FRENCH	SOURCE	#AL.	ERROR CLASS	GERMAN	SOURCE	#AL.	ERROR CLASS
rentrer	go	9,070	CP: *go home*	möchten	want	26,996	DI
ressembler	look	6,160	CP: *look like*	sollen	suppose	12,849	CP: *be supposed to*
naître	bear	5,937	LE: *be born*	sollen	want	12,619	CP: *want sb. to do sth.*
pouvoir	be	4,541	CP: *be able to*	herausfinden	find	7,756	CP: *find out*
asseoir	sit	4,300	LE: *s'assoir*	beschützen	protect	6,416	DI
adorer	love	3,391	DI	fällen	like	5,846	LE
pouvoir	get	3,325	CP: *get to do something*	mitnehmen	take	5,604	CP: *take along*
rentrer	come	2,813	CP: *come home*	kennenlernen	meet	5,254	CP: *get to know sb.*.
appartenir	belong	2,745	DI	übernehmen	take	4,029	CP: *take over*
enfermer	lock	2,143	DI	erledigen	do	3,914	CP: *get sth. done*

Table 2: Top 10 unlabeled verbs for French and German, with English source verbs and alignment counts in the parallel corpus. Error classes are lemmatiation error (LE), dictionary incomplete (DI) and complex predicates (CP).

4.2 Curation Guidelines

In order to assess our curation guidelines, we asked two curators to independently execute the curation process for the French lexicon. We produce two independently curated French Proposition Banks using the two lexicons. We compared both versions and found that curators had agreed on 1,317 out of 1,370 (96%) of all curated entries. This speaks to the deterministic nature of our guidelines.

One contributing factor to our high agreement score is that we discussed and determined representative cases of disagreement in early iterations of the curation process. In the filtering task, for instance, we encountered initial disagreements on theoretically correct entries that were incorrectly recognized in practice (see section 3.1). In the merging task, we also had initial difficulties concerning semantically related frames that were neither clear usage-synonyms nor clearly distinct (see section 3.2). As previously illustrated, we defined deterministic rules for these cases.

Disagreements. Some disagreement remained in the merging task: Some entries have syntactically similar usage and partially overlapping semantics. This often affects merging questions in which one entry is a highly specialized usage of the other. This frequently involves slang language. For example, the French verb *secher* (to dry) can also be used in spoken language to indicate "cutting class" as in not attending class in school. Due to the lack of a more appropriate English frame, this sense is aligned to the frame CUT.01 (slice or injure). Such cases required further discussion by curators.

4.3 Qualitative Evaluation of Curated Proposition Banks

The curated Proposition Banks give us the opportunity to gain insights on verb coverage and to compare annotation projection against manual annotation efforts. We present results of a qualitative inspection of common TL verbs that are absent in generated frame lexicons and a qualitative comparison of our generated resource for Chinese against the official Chinese Proposition Bank (Xue and Palmer, 2005).

Verb coverage. From the parallel corpora, we retrieve the most common TL verbs that are not contained in our frame lexicon. We manually inspect the 100 most common unlabeled verbs to determine reasons for lack of coverage. We find that lack of coverage can be traced back to one of three reasons: 1) The translation dictionary we use to filter out translation shift errors is incomplete (DI). 2) The lemmatizer is unable to correctly lemmatize certain verbs (LE). 3) The TL verb meaning can only be rendered in English as a complex predicate (CP). We list the top 10 most common unlabeled verbs for French and German in Table 2.

Complex phrasal constructions. A crucial error class are TL verbs that cannot be expressed with a single verb in English. Consider the French verb *rentrer*, which frequently expresses the meaning of "returning home", rendered in English with the LVCs *go home* or *come home*. Another example is *pouvoir*, which in English needs to be rendered with the adverb *able* as in *be able to*. A German example for this phenomenon is *sollen*, rendered as *to be supposed to* or *to want someone to do something*. This represents a limitation of our current verb-based projection approach. However, there are ongoing efforts to expand the English Proposition Bank with frames for complex predicates. We believe that this will allow us to address this source of verb coverage loss in future work.

Comparison to Chinese PropBank We randomly sample 100 Chinese verbs from our lexicon and compare all entries against the official Chinese Proposition Bank. We find an encouragingly high agreement, which 94.6% of our lexicon entries corresponding to senses in the Chinese PropBank. We also find 7 lexicon entries that do not exist in the Chinese PropBank, indicating that annotation projection with manual frame curation may be used to increase coverage of existing Proposition Banks. We also find 4 instances in which valid entries for a verb in the two PropBanks are complimentary. However, we point out that we conduct this study only in one direction. The official Chinese Proposition Bank contains frames for all types of frame evoking elements, including verbs, nouns and complex predicates (Xue, 2006; Xue and Palmer, 2009). Their coverage is therefore significantly larger than our current approach. Nevertheless, we find the significant overlap in both frame lexicons encouraging.

5 Discussion and Conclusion

We presented an approach for addressing lexicon-level inconsistencies in automatically generating Proposition Banks using annotation projection. Our approach manually curates the heuristically determined frame lexicon in two steps: A filtering and a merging step. We executed the approach on large-scale parallel data to generate Proposition Banks with curated frames. Our evaluation shows that our approach significantly increases the quality of the generated resources, while reducing redundancy and inconsistency in the frame lexicon.

Our evaluation also revealed TL verbs that require complex predicates in English as a natural limitation of our current verb-based approach. Accordingly, future work will investigate this issue by expanding the range of projections from verbs to other types of frame-evoking elements. We aim to expand our frame lexicon to include not only TL verbs, but also nouns, adjectives and eventually complex TL predicates.

Another avenue for future work is to investigate the use of SRL trained over projected Proposition Banks in applications. As previous work has shown information extraction and question answering to benefit from SRL, we aim to investigate multilingual applications in these tasks.

We release the Proposition Banks created with our approach in order to encourage discussion with the research community. We believe this resource to potentially be valuable for investigating crosslingual semantics and for training statistical SRL systems for Chinese, French and German.

References

Alan Akbik and Yunyao Li. 2016. Polyglot: Multilingual semantic role labeling with unified labels. In *ACL 2016, 54th Annual Meeting of the Association for Computational Linguistics: Demostration Session*, page to appear.

Alan Akbik, Laura Chiticariu, Marina Danilevsky, Yunyao Li, Shivakumar Vaithyanathan, and Huaiyu Zhu. 2015. Generating high quality proposition banks for multilingual semantic role labeling. In *ACL 2015, 53rd Annual Meeting of the Association for Computational Linguistics*, pages 397–407.

Claire Bonial, Julia Bonn, Kathryn Conger, Jena D Hwang, and Martha Palmer. 2014. Propbank: Semantics of new predicate types. In *LREC*, pages 3013–3019.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics.

Glorianna Jagfeld and Lonneke van der Plas. 2015. Towards a better semantic role labeling of complex predicates. In *NAACL-HLT 2015 Student Research Workshop (SRW)*, page 33.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *In Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*.

Chi-kiu Lo, Meriem Beloucif, and Dekai Wu. 2013. Improving machine translation into chinese by tuning against chinese meant. In *Proceedings of the Tenth International Workshop on Spoken Language Translation (IWSLT 2013)*.

Umar Maqsud, Sebastian Arnold, Michael Hülfenhaus, and Alan Akbik. 2014. Nerdle: Topic-specific question answering using wikia seeds. In *COLING (Demos)*, pages 81–85.

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36(1):307–340.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *EMNLP-CoNLL*, pages 12–21.

Lonneke Van der Plas, Paola Merlo, and James Henderson. 2011. Scaling up automatic cross-lingual semantic role annotation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 299–304. Association for Computational Linguistics.

Lonneke van der Plas, Marianna Apidianaki, and Chenhua Chen. 2014. Global methods for cross-lingual semantic role and predicate labelling. In *COLING*, pages 1279–1290. ACL.

Nianwen Xue and Martha Palmer. 2005. Automatic semantic role labeling for chinese verbs. In *IJCAI*, volume 5, pages 1160–1165. Citeseer.

Nianwen Xue and Martha Palmer. 2009. Adding semantic roles to the chinese treebank. *Natural Language Engineering*, 15(01):143–172.

Nianwen Xue. 2006. Semantic role labeling of nominalized predicates in chinese. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 431–438. Association for Computational Linguistics.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*, pages 1–8. Association for Computational Linguistics.

PanPhon: A Resource for Mapping IPA Segments to Articulatory Feature Vectors

David R. Mortensen, Patrick Littell, Akash Bharadwaj,
Kartik Goyal, Chris Dyer, Lori Levin
Carnegie Mellon University
Language Technologies Institute
5000 Forbes Ave., Pittsburgh PA 15213
United State of America
dmortens@cs.cmu.edu, plittell@cs.cmu.edu, akashb@cs.cmu.edu,
kartikgo@cs.cmu.edu, cdyer@cs.cmu.edu, lsl@cs.cmu.edu

Abstract

This paper contributes to a growing body of evidence that—when coupled with appropriate machine-learning techniques—linguistically motivated, information-rich representations can outperform one-hot encodings of linguistic data. In particular, we show that phonological features outperform character-based models using the PanPhon resource. PanPhon is a database relating over 5,000 IPA segments to 21 subsegmental articulatory features. We show that this database boosts performance in various NER-related tasks. Phonologically aware, neural CRF models built on PanPhon features are able to perform comparably to character-based models on monolingual Spanish and Turkish NER tasks. On transfer models (as between Uzbek and Turkish) they have been shown to perform better. Furthermore, PanPhon features also contribute measurably to Orthography-to-IPA conversion tasks.

1 Introduction

This paper introduces PanPhon[1], a resource consisting of a database that relates over 5,000 IPA segments (simple and complex) to their definitions in terms of 21 articulatory features (see Tab. 1) as well as a Python package for interacting with this database and manipulating the representations that it provides. While our previous publications (summarized in §4) have described experiments using it, this is the first full description of PanPhon. Combined with a sister package, Epitran[2], it allows the conversion of

	syl	son	cons	cont	delrel	lat	nas	strid	voi	sg	cg	ant	cor	distr	lab	hi	lo	back	round	tense	long
/p/	-	-	+	-	-	-	-	0	-	-	-	+	-	0	+	-	-	-	-	0	-
/p^{h}/	-	-	+	-	-	-	-	0	-	+	-	+	-	0	+	-	-	-	-	0	-
/p^{j}/	-	-	+	-	-	-	-	0	-	-	-	+	-	0	+	+	-	-	-	0	-
/p^{hj}/	-	-	+	-	-	-	-	0	-	+	-	+	-	0	+	+	-	-	-	0	-

Table 1: Illustration of IPA segments and feature vectors from PanPhon

orthographic texts to sequences of articulatory feature vectors. The Epitran-Panphon pipeline is illustrated in Fig. 1. The input to Epitran consists of word tokens in orthographic representation. Take, for example, the Spanish word *Madrid*. Epitran converts this string to a phonemic (not phonetic) representation in IPA, in this case /madɾid/. Epitran then calls a PanPhon function to convert this IPA string into a sequence of feature vectors. It then returns this sequence, aligned with the orthographic representation, capitalization, and Unicode character category features.

[1] https://github.com/dmort27/panphon

[2] https://github.com/dmort27/epitran

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3475–3484, Osaka, Japan, December 11-17 2016.

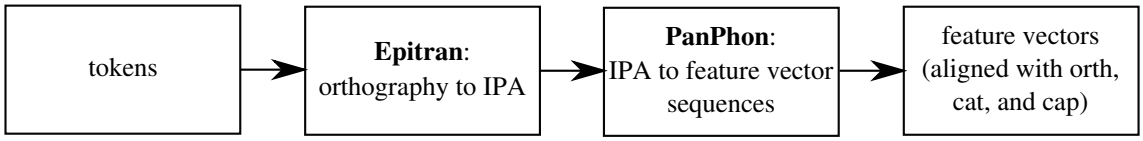


Figure 1: Feature vector pipline

This paper also shows that subsegmental features, as encoded in PanPhon, are useful in NLP tasks. The specific tasks for which PanPhon has been shown to improve performance are named entity recognition and conversion of lossy orthographies[3] to IPA.

Phonologists have long held that articulatory features play a role in three central aspects of phonology: **contrast**, **distribution**, and **alternation**. These constitute the distinctiveness of speech sounds, the restrictions on where they can occur in the speech-stream, and the phonologically-derived differences in the various realizations of the same morpheme. These may be illustrated through examples from Turkish (Lees, 1963) as illustrated in Tab. 2. The difference between /i/ and /e/ is sufficient to distinguish two

NOM.SG	GEN.SG	NOM.PL	GEN.PL	gloss
ip	ip-**in**	ip-**ler**	ip-**ler-in**	'rope'
el	el-**in**	el-**ler**	el-**ler-in**	'hand'
kɪz	kɪz-**ɪn**	kɪz-**lar**	kɪz-**lar-ɪn**	'girl'

Table 2: Turkish vowel harmony (NOM = nominative, GEN = genitive, SG = singular, and PL = plural)

words, evidence that the feature [±high][4] is contrastive (sufficient to make lexical contrasts) in Turkish. The vowels in the suffixes (*-ler*/*-lar* and *-in*/*-ın*) alternate to assimilate in quality to the preceding vowel. This can be easily expressed in terms of the phonological feature [±back]: The vowels ⟨i⟩ and ⟨e⟩ are [−back] while ⟨a⟩ and ⟨ɪ⟩ are [+back][5]. The alternation consists of the smallest change that allows agreement in the specification of this feature between the vowels in the root at the vowel in the suffix. In this way, phonological features allow a degree of generalization, even over orthographic patterns, that purely character-based (and also phoneme-based) models do not. Finally, the static observation that, in all but a minority of "disharmonic" words, each vowel in a word shares the same specification for [±high] and [±back] is a matter of distribution. These linguistic insights behind articulatory features translate into practical benefits for NLP systems.

The usefulness of articulatory features in natural language processing, this paper will show, is most valuable for replacing character-level models where characters bear some predictable relationship to phonemes or phones. In such a scenario, treating characters as atomic entities is rather like treating the chords of a musical score as unanalyzable units. While this approach may be adequate for many purposes, we will argue that it ignores aspects of phonological structure that have demonstrable utility. Articulatory features represent the parts in a layered articulatory score. Operations on this score target these individual features, not the segment as a whole ("the chord"), contrary to the assumption made in strictly character-based approaches. We have found, from a variety of perspectives, that exploiting these components can improve the performance of relevant NLP systems.

2 Past Use of Phonological Features in NLP

Within both formal and empirical linguistics, it is rare to encounter discussion of structural patterns in sounds without some mention of phonological features. Even radically empirical phonologists, rather than denying the existence of such phonological features, tend to simply deny that there is a universal

[3]"Lossy orthographies" are ambiguous writing systems that lose segmental information present in the speech stream.

[4]In the linguistic subfield of phonology, features are often represented in square brackets with the name on the right and the value presented as +, −, or ± (indicating that the feature is binary but that the value is not known).

[5]Note that alternate feature systems use the feature [±front] to account for this contrast.

and innate feature inventory (Mielke, 2008). Likewise, in the speech sciences and speech technology, phonological features have been the subject of widespread inquiry (Bromberg et al., 2007; Metze, 2007). This contrasts with NLP, where phonological features have been subject to less experimentation, perhaps because of the perceived lower relevance of phonology than morphology, syntax, and semantics to NLP tasks. However, some promising NLP results have been achieved using phonological features.

In one of the more widely-cited cases of this kind, Gildea and Jurafsky (1996) added phonological biases—stated in terms of phonological features—to aid the OSTIA (the Onward Subsequential Transducer Inferance Algorithm) in learning phonological rules. Subsequently , Tao et al. (2006) used hand-weighted articulatory feature edit distance (augmented with "pseudofeatures") to facilitate the transliteration of named entities. This feature-based system outperformed a temporal algorithm on a English/Hindi language pair and contributed to the performance of the best model the authors tested. In a successor study, Yoon et al. (2007) employed the model of Tao et al. (2006) but with features weighted by the winnow algorithm rather than by hand; they achieved comparable results without engineered weights. In a related strain of research, Kondrak and Sherif (2006) explored phonological similarity measures based, in some cases, on a kind of phonological feature system (but with a multivalued place feature unlike the binary and ternary features integral to PanPhon).

3 PanPhon and Its Functionality

PanPhon facilitates further experimentation with phonological (specifically, articulatory) features. Our goal in implementing PanPhon was not to implement a state-of-the-art feature system (from the standpoint of linguistic theory) but to develop a methodologically solid resource that would be useful for NLP researchers. Contemporary feature theories posit hierarchical and non-linear feature structures; they cannot be easily expressed in terms of vectors of binary or ternary values like the earlier-vintage system represented in PanPhon. Linear phonological models like that instantiated in PanPhon are arguably less explanatory than non-linear and feature-geometric models, but they can also be said to hew closer to the empirical ground. One limitation of PanPhon involves the representation of tone, one area in which non-linear representations are almost universally conceded to hold the upper hand. PanPhon is segmental by design and tone is suprasegmental by nature.

In constructing PanPhon, our approach was to start with a subset of the International Phonetic Alphabet (or IPA) where every segment represents a sound that is distinct from the others. Contrary to its design principles, the IPA allows multiple transcriptions for the same sound in a variety of cases. An attempt was made, using consensus definitions, to classify each of these segments according to binary (and occasionally, ternary) features. This consensus feature set and the accompanying definitions were based on a survey of the phonological literature.

PanPhon's contribution lies in the following attributes:

Universal. PanPhon will, when queried with a legal, segmental Unicode IPA string, return a sequence of valid vectors of articulatory feature values. Currently, it defines 5,395 simple and complex IPA characters in terms of 21 articulatory features.

Empirically verified. The general feature system used in PanPhon has been widely tested by linguists across a great range of phenomena and languages and found to be effective at modeling the phonological and morphophonological patterns of the world's languages.

Unicode compliant. PanPhon uses the Unicode encoding of the International Phonetic Alphabet both internally and externally. This provides human readability (or, at least, readability to human linguists) in a way that ASCII mappings of all or part of the IPA, like ARPAbet, WorldBet, SAMPA, and X-SAMPA do not. Compare German *müde* 'tired' in ARPAbet /M <no_symbol> D AH/, WordBet /m y: d &/, SAMPA /m y: d @/, X-SAMPA /m y: d @/ with IPA /myːdə/. The development of convenient input methods and appropriate rendering technologies for IPA mitigate much of the past difficulty involved in using it in computing applications.

Open source. Unlike many existing phonological feature resources[6], PanPhon is freely available under a liberal license (MIT).

PanPhon consists of a collection of components:

1. A database relating unmodified IPA segments to vectors of 21 features. These form the core of the database. The features are listed here in the canonical order in which they appear in the database:

   syl [±syllabic]. Is the segment the nucleus of a syllable?
   son [±sonorant]. Is the segment produced with a relatively unobstructed vocal tract?
   cons [±consonantal]. Is the segment consonantal (not a vowel or glide, or laryngeal consonant)?
   cont [±continuant]. Is the segment produced with continuous oral airflow?
   delrel [±delayed release]. Is the segment an affricate?
   lat [±lateral]. Is the segment produced with a lateral constriction?
   nas [±nasal]. Is the segment produced with nasal airflow?
   strid [±strident]. Is the segment produced with noisy friction?
   voi [±voice]. Are the vocal folds vibrating during the production of the segment?
   sg [±spread glottis]. Are the vocal folds abducted during the production of the segment?
   cg [±constricted glottis]. Are the vocal folds adducted during the production of the segment?
   ant [±anterior]. Is a constriction made in the front of the vocal tract?
   cor [±coronal]. Is the tip or blade of the tongue used to make a constriction?
   distr [±distributed]. Is a coronal constriction distributed laterally?
   lab [±labial]. Does the segment involve constrictions with or of the lips?
   hi [±high]. Is the segment produced with the tongue body raised?
   lo [±low]. Is the segment produced with the tongue body lowered?
   back [±back]. Is the segment produced with the tongue body in a posterior position?
   round [±round]. Is the segment produced with the lips rounded?
   tense [±tense]. Is the segment produced with an advanced tongue root.

   Feature vectors from PanPhon for a few example segments (both simple and complex) are shown in Tab. 1.

2. A collection of rules, written in user-editable YAML, that describe diacritics and modifiers—the Unicode codepoint of the modifier, the feature specifications that provide the necessary context for adding the diacritic or modifier, and the feature specifications changes that take place if the diacritic or modifier is added to a segment. An example of one of these YAML rules is shown below:

Listing 1: Diacritic rule

```
- marker: ʷ
  name: Labialized
  position: post
  conditions:
    - syl:  "-"
  exclude:
    - w
    - ʍ
    - ɥ
  content:
    round: "+"
    back:  "+"
    hi:  "+"
```

[6]But see also PHOIBLE's phonological feature set (Moran et al., 2014).

This example shows a rule (named "Labialized") that adds the modifier (marker) "ʷ" after a segment—post(fix), if the segment has the feature [−syllabic] and is not /w/, /ʍ/, or /ɥ/. The new segment has the same features as the input segment except that is it [+round], [+back], and [+hi].

3. A script for applying diacritics and modifiers, as defined in 2, to segments, as defined in 1 and the comprehensive segment database produced by this script.

4. A set of Python convenience functions and classes for accessing, manipulating, and employing the phonological feature vectors associated with any segment, whether simple or complex:

   (a) Greedily parsing IPA strings into segments defined in the database.
   (b) Converting such a sequence of segments into a sequence of articulatory feature vectors.
   (c) Querying sets of segments based upon their features.
   (d) Pattern matching against strings using feature specifications to define character classes.
   (e) Computing phonological distance: weighted and unweighted feature edit distance.
   (f) Computing phonological distance: Levenshtein distance between strings with collapsed, phonologically-based segment equivalence classes.

In some applications, PanPhon is used with a sister library, Epitran, or another—more specialized—package for converting orthography to IPA. Epitran provides a simple interface for quickly implementing grapheme to phoneme mappings for languages with phonemically adequate orthographies. It includes mappings for a variety of languages including Spanish, Dutch, Turkish, and Uyghur.

4 Empirical Evaluation of PanPhon

It is not simply the case that the articulatory features available through PanPhon are well founded in terms of linguistic theory. It is also true that they have been demonstrated to improve the performance of certain machine learning models at certain NLP tasks. This section summarizes the contribution of PanPhon to two classes of tasks: orthography-to-IPA character transduction and named entity recognition (NER). While the second set of experiments (on NER) were prompted by a need to test a particular class of model—phonologically aware LSTM-CRFs—the first set were motivated by the need to solve a particular problem: how best to convert Sorani Kurdish (a Northwestern Iranian language of Iraq and Iran) from orthographic to IPA representation.

4.1 Orthography-IPA Character Transduction

As part of a NER system for the low-resource Sorani Kurdish language, we developed a Sorani-orthography-to-IPA converter Littell et al. (2016). This was challenging because the Sorani orthography, like many Perso-Arabic scripts, badly underdetermines the equivalent phonetic representation. The following steps summarize the workflow behind building the Sorani-to-IPA converter:

1. Human linguists identify the orthographic units (i.e., characters and multigraphs) in the script.

2. Human linguists identify the possible IPA representations of each orthographic unit, using knowledge of the language and the writing system.

3. The system generates all possible hypotheses for a subset of tokens of the language using the mapping developed in the previous step.

4. Human linguists generate training data using a grammar and lexicon (Thackston, 2006) by picking one or more valid pronunciations (or, if unknown, one or more likely hypotheses) for a selection of tokens.

5. Character-level chain conditional random field (CRF) is trained (Lafferty et al., 2001; Dyer et al., 2010) on resulting data.

The hypothesis space within which the CRF operates was determined by the symbol-level IPA map developed in the first step of our work-flow. It is important to note that we allowed many-to-many mapping between orthographic input character sequences and IPA output character sequences in the sense that a single input character can be mapped to multiple IPA symbols and an input multigraph (consisting of multiple orthographic characters) can be mapped to a single IPA symbol.

PanPhon feature vectors were used to create one set of features that were consumed by the CRF. This move was motivated by the insight that phonotactic patterns—patterns in the sequences of speech sounds—tend to be based around the very sorts of classes that phonological features are intended to describe. For the purposes of these experiments, we divided these features into six classes which correspond largely to classes widely used by phonologists:

- **Major Class** features—represent the major classes of sounds: [±syllable], [±sonorant], [±consonantal], [±continuant].
- **Laryngeal** features—specify the glottal states of sounds: [±voice], [±spread glottis], [±constricted glottis].
- **Major place** features—focus on the place of articulation: [±anterior], [±coronal], [±labial] and [±distributed][7].
- **Minor Place** features—related to the position of the dorsum in the tongue: [±high], [±low] and [±back].
- **Manner** features—categorize IPA symbols according to their manner of articulation: [±nasal], [±lateral], [±delayed release] and [±strident].
- **Minor Manner** features—the attributes in this group were [±round] and [±tense].

We then generated the training data with 244 instances (types)[8] and report performance of their IPA predictor on 402 instances. In Table 3 (adapted from Littell et al. (2016)), we compare the accuracy and character error rate (CER) of the predictor, when using PanPhon features compared to, and along with:

- **Basic** features pertaining to the usage of orthography-to-IPA symbol translation rules, and whether the IPA symbols were identified as consonants and vowels[9].
- **Phon** features derived from the PanPhon IPA-to-feature-vector package.
- **Kurmanji** features derived from a 4-gram language model, built using SRILM (Stolcke, 2002), from the IPA-ized Kurmanji corpus. Kurmanji is a "sister" language of Sorani having a phonologically unambiguous orthography.
- **Tajik** features, derived as above, from the IPA-ized Tajik corpus. Tajik is a close cousin of Sorani having a phonologically unambiguous orthography.

While the single most important class of features, all told, were those derived from the Kurmanji language model, the inclusion of PanPhon features gave similar gains to the inclusion of the Tajik features, and the best results overall came from the inclusion of both Kurmanji and PanPhon features. This is a welcome result—while not all languages have a close sister variety with an unambiguous orthography, PanPhon-based features are universally available.

Among the PanPhon features, the most valuable features for this task were the *major class features*, *laryngeal features*, and *place features*. Unsurprisingly, having a language model from a sister language (in this case, Kurmanji) is the most useful single source of features for IPA conversion. However, it is surprising to find that just having the universal features derived from PanPhon vectors were as useful as having a language model from a cousin language (in this case, Tajik).

[7]Phonologists do not generally consider the feature [±distributed] to be a major place feature. This appears to have been an error in our implementation of the feature classes.

[8]It is worth noting that when reducing the training set to a quarter of this, we observed only slightly worse results, suggesting that very little manual effort would be necessary to generate training data for a task like this.

[9]The consonant/vowel feature should have been largely equivalent to the feature [±syllabic].

Features	Accuracy	CER
Basic	0.635	0.237
Basic+phon.	0.669	0.234
Basic+Kurmanji	0.701	0.223
Basic+Kurmanji+phon.	0.721	0.221
Basic+Tajik	0.661	0.231
Basic+Tajik+phon.	0.664	0.228
All features	0.721	0.221

Table 3: IPA prediction for Sorani, trained on 244 tokens, tested on 402 tokens

4.2 NER with Phonologically-Aware Neural Models

We subsequently experimented with PanPhon—and its sister package, Epitran—in performing NER with a character-based LSTM-CRF architecture (Bharadwaj et al., 2016). We made this architecture phonologically-aware by substituting phonological feature vectors from PanPhon for characters. We used the resulting features in a series of NER experiments in both monolingual and transfer scenarios. As a baseline, we employed a character based LSTM-CRF NER system with features from pre-trained word vectors.

In a series of monolingual experiment using CoNLL 2002 data from Spanish (see Tab. 4, adapted from (Bharadwaj et al., 2016)) it was found that substituting PanPhon and phonological attention features for orthographic and orthographic attention features (in a model also incorporating word vector, capitalization, and Unicode character category features) raised F1 from 85.25 to 85.81 and yielded the best-performing model in the series. In a second series of monolingual experiments using the LDC2014E115 BOLT Turk-

Features	F1
WVec	83.61
WVec+Phon	84.08
WVec+Orth	84.52
WVec+Phon+Orth+PhonAttn	84.53
WVec+Orth+OrthAttn	84.64
WVec+Phon+PhonAttn	84.88
WVec+Phon+Cap+Cat	84.89
WVec+Orth+Cap+Cat	84.91
WVec+Phon+Orth+Cap+Cat	84.92
WVec+Orth+OrthAttn+Cap+Cat	85.25
WVec+Phon+PhonAttn+Cap+Cat	**85.81**
WVec+Phon+Orth+OrthAttn+Cap+Cat	85.32
WVec+Phon+PhonAttn+Orth+Cap+Cat	84.84
All features	84.75

Table 4: Ablation tests on Spanish NER; bold indicates best model; factors are pre-trained word vectors (WVec), PanPhon features (Phon), phonological attention features (PhonAttn), orthographic features (Orth), orthographic attention features (OrthAttn), capitalization features (Cap), and Unicode category features (Cat)

ish Language Pack, Bharadwaj et al. (2016) found that the best model incorporated PanPhon as well as orthographic features (see Tab. 5. The second-best scoring model uses only PanPhon features and phonological attention features (in addition to pretrained word vectors) and because it is character-independent, it is more suitable for cross-lingual transfer. Finally, Bharadwaj et al. (2016) conducted a series of Uzbek-to-Turkish cross-lingual transfer experiments using the LDC2015E89 BOLT data pack for Uzbek and the

Features	F1
WVec	49.2
WVec+Orth	65.41
WVec+Orth+OrthAttn	64.76
WVec+Orth+Cap+Cat	60.57
WVec+Orth+Cap+Cat+OrthAttn	60.87
WVec+Phon	63.04
WVec+Phon+PhonAttn	**66.07**
WVec+Phon+Cap+Cat	59.08
WVec+Phon+Cap+Cat+PhonAttn	62.46
WVec+Phon+Orth+PhonAttn	63.43
WVec+Phon+Orth+Cap+Cat+PhonAttn	63.46
All features	**66.47**

Table 5: Ablation tests on Turkish NER; Boldface indicates the best model (66.47); factors are pre-trained word vectors (WVec), PanPhon features (Phon), phonological attention features (PhonAttn), orthographic features (Orth), orthographic attention features (OrthAttn), capitalization features (Cap), and Unicode category features (Cat)

LDC2014E115 BOLT data pack for Turkish. Unsurprisingly, monolingual models with no training data achieve an F1 of zero. In a transfer situation, with no Turkish training data, word vectors alone give an F-score of 2.09. However, using phonological features and phonological attention features with zero training data in the target language (but training in the transfer language) yields an F-score of 11.9. Adding capitalization and Unicode category features allowed the resulting model to achieve an F1 of 26.92 in a zero-shot scenario.

Features	**Source**	**Target 0-shot**	**5% data**	**20% data**	**40% data**	**60% data**	**80% data**	**All data**
WVec	41.87	2.09	23.44	35	42.75	46.32	48.81	50.34
WVec+Phon+PhonAttn	61.24	11.9	34.06	47.84	56.1	53.5	64.72	65.2
WVec+Phon+Cap+Cat	60.92	15.55	39.42	**60.14**	**63.23**	**62.54**	**65.24**	**65.63**
All features	**61.85**	**26.92**	**47.21**	58.58	60.32	60.7	62.84	63.58
Monolingual Models		**Target 0-shot**	**5% data**	**20% data**	**40% data**	**60% data**	**80% data**	**All data**
LSTM-CRF (Lample et al., 2016)		0	33.44	50.61	53.25	57.41	60	61.11
S-LSTM (Lample et al., 2016)		0	15.41	39.33	42.99	51.92	51.55	56.58

Table 6: Model transfer from Uzbek to Turkish at different target data availability thresholds compared to monolingual Turkish baseline, also at different data availability thresholds; factors are pre-trained word vectors (WVec), PanPhon features (Phon), phonological attention features (PhonAttn), capitalization features (Cap), and Unicode category features (Cat)

What do these experiments reveal about the value of PanPhon for NER? The monolingual experiments may not seem compelling, but they drive home an important point: even though conversion from orthography to PanPhon vectors entails a significant loss of information, it does not seem to entail a loss of performance. It is even possible that phonological features facilitate NER in the monolingual case by helping a neural NER system to identify phonologically aberrant tokens which are more likely to reflect borrowed lexical items which, in turn, are more likely to be named entities. The real benefit of using phonological feature representations in NER, however, is manifest in transfer scenarios like the Uzbek-Turkish transfer scenario we explored in Bharadwaj et al. (2016). By projecting both the source and target language into a common phonological space, similarities that would be masked in an orthographic space become accessible to a neural NER system. The value of such an approach is most obvious when it is applied to relatively closely-related languages that are written in different orthographies like Uzbek and

Turkish[10].

5 Conclusion

This paper has reported the creation of a new resource, PanPhon, a database of IPA segment-phonological feature correspondences with a collection of code for exploiting these relationships. Additionally, we briefly documented the PanPhon database and modules. Most significantly, we showed that PanPhon features can improve performance in two NLP tasks: orthography-to-IPA conversion and NER. At this point, the PanPhon feature mapping has still not be tested against other phonological feature implementations (which have typically not been made widely available) in these tasks, so it cannot be claimed that PanPhon boosts performance more than these other databases. However, it can be said conclusively that systems *like* PanPhon are a useful component in some NLP systems.

Acknowledgements

Supported by the U.S. Army Research Office under grant number W911NF-10-1-0533. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the U.S. Army Research Office or the United States Government.

Sponsored by Defense Advanced Research Projects Agency (DARPA) Information Innovation Office (I2O) Program: Low Resource Languages for Emergent Incidents (LORELEI) Issued by DARPA/I2O under Contract No. HR0011-15-C-0114.

References

Akash Bharadwaj, David Mortensen, Chris Dyer, and Jaime G. Carbonell. 2016. Phonologically aware neural model for named entity recognition in low resource transfer settings. Accepted for EMNLP 2016.

Ilana Bromberg, Jeremy Morris, and Eric Fosler-Lussier. 2007. Joint versus indepedent phonological feature models within CRF phone recognition. In *Proceedings of NAACL HLT 2007*, pages 13–16. Association for Computational Linguistics.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Procedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*.

Daniel Gildea and Daniel Jurafsky. 1996. Learning bias and phonological-rule induction. *Computational Linguistics*, 22(4):497–530.

Grzegorz Kondrak and Tarek Sherif. 2006. Evaluation of several phonetic similarity algorithms on the task of cognate identification. In *Proceedings of the Workshop on Linguistic Distances*, pages 43–50. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360.

Robert B. Lees. 1963. *The Phonology of Modern Standard Turkish*. Indiana University Press, Bloomington.

Patrick Littell, David Mortensen, Kartik Goyal, Chris Dyer, and Lori Levin. 2016. Bridge-language capitalization inference in Western Iranian: Sorani, Kurmanji, Zazaki, and Tajik. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 16)*.

Florian Metze. 2007. On using articulatory features for discriminative speaker adaptation. In *Proceedings of NAACL HLT 2007, Companion Volume*. Association for Computational Linguistics.

[10]Note that while Turkish and Uzbek are both written with the Latin alphabet, they use considerably different orthographic conventions that hide similarities in pronunciation behind differences in symbols. The utility of PanPhon is illustrated even more dramatically in the Turkish/Uzbek-Uyghur transfer scenario that is mentioned briefly by Bharadwaj et al. (2016), since Uyghur uses a Perso-Arabic script that differs entirely from the Latin scripts that are now used for Turkish and Uzbek.

Jeff Mielke. 2008. *The Emergence of Distinctive Features*. Oxford University Press, Oxford.

Steven Moran, Daniel McCloy, and Richard Wright, editors. 2014. *PHOIBLE Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Andreas Stolcke. 2002. SRILM — an extensible language modeling toolkit. In J. H. L. Hansen and B. Pellom, editors, *ICSLP*, volume 2, pages 901–904.

Tao Tao, Su-Youn Yoon, Andrew Fister, Richard Sproat, and ChengXiang Zhai. 2006. Unsupervised named entity transliteration using temporal and phonetic correlation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 250–257. Association for Computational Linguistics, July.

W. M. Thackston. 2006. Sorani Kurdish reference grammar with selected readings. Book manuscript.

Su-Youn Yoon, Kyoung-Young Kim, and Richard Sproat. 2007. Multilingual transliteration using feature based phonetic method. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, pages 112–119. Association for Computational Linguistics.

Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling

Peng Zhou[1], **Zhenyu Qi**[1]*, **Suncong Zheng**[1], **Jiaming Xu**[1], **Hongyun Bao**[1], **Bo Xu**[1,2]
(1) Institute of Automation, Chinese Academy of Sciences
(2) Center for Excellence in Brain Science and Intelligence Technology
{zhoupeng2013, zhenyu.qi, zhengsuncong,
jiaming.xu, hongyun.bao, xubo}@ia.ac.cn

Abstract

Recurrent Neural Network (RNN) is one of the most popular architectures used in Natural Language Processsing (NLP) tasks because its recurrent structure is very suitable to process variable-length text. RNN can utilize distributed representations of words by first converting the tokens comprising each text into vectors, which form a matrix. And this matrix includes two dimensions: the time-step dimension and the feature vector dimension. Then most existing models usually utilize one-dimensional (1D) max pooling operation or attention-based operation only on the time-step dimension to obtain a fixed-length vector. However, the features on the feature vector dimension are not mutually independent, and simply applying 1D pooling operation over the time-step dimension independently may destroy the structure of the feature representation. On the other hand, applying two-dimensional (2D) pooling operation over the two dimensions may sample more meaningful features for sequence modeling tasks. To integrate the features on both dimensions of the matrix, this paper explores applying 2D max pooling operation to obtain a fixed-length representation of the text. This paper also utilizes 2D convolution to sample more meaningful information of the matrix. Experiments are conducted on six text classification tasks, including sentiment analysis, question classification, subjectivity classification and newsgroup classification. Compared with the state-of-the-art models, the proposed models achieve excellent performance on 4 out of 6 tasks. Specifically, one of the proposed models achieves highest accuracy on Stanford Sentiment Treebank binary classification and fine-grained classification tasks.

1 Introduction

Text classification is an essential component in many NLP applications, such as sentiment analysis (Socher et al., 2013), relation extraction (Zeng et al., 2014) and spam detection (Wang, 2010). Therefore, it has attracted considerable attention from many researchers, and various types of models have been proposed. As a traditional method, the bag-of-words (BoW) model treats texts as unordered sets of words (Wang and Manning, 2012). In this way, however, it fails to encode word order and syntactic feature.

Recently, order-sensitive models based on neural networks have achieved tremendous success for text classification, and shown more significant progress compared with BoW models. The challenge for textual modeling is how to capture features for different text units, such as phrases, sentences and documents. Benefiting from its recurrent structure, RNN, as an alternative type of neural networks, is very suitable to process the variable-length text.

RNN can capitalize on distributed representations of words by first converting the tokens comprising each text into vectors, which form a matrix. This matrix includes two dimensions: the time-step dimension and the feature vector dimension, and it will be updated in the process of learning feature representation. Then RNN utilizes 1D max pooling operation (Lai et al., 2015) or attention-based operation (Zhou et al., 2016), which extracts maximum values or generates a weighted representation over

*Correspondence author: zhenyu.qi@ia.ac.cn

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3485–3495, Osaka, Japan, December 11-17 2016.

the time-step dimension of the matrix, to obtain a fixed-length vector. Both of the two operators ignore features on the feature vector dimension, which maybe important for sentence representation, therefore the use of 1D max pooling and attention-based operators may pose a serious limitation.

Convolutional Neural Networks (CNN) (Kalchbrenner et al., 2014; Kim, 2014) utilizes 1D convolution to perform the feature mapping, and then applies 1D max pooling operation over the time-step dimension to obtain a fixed-length output. However the elements in the matrix learned by RNN are not independent, as RNN reads a sentence word by word, one can effectively treat the matrix as an 'image'. Unlike in NLP, CNN in image processing tasks (LeCun et al., 1998; Krizhevsky et al., 2012) applies 2D convolution and 2D pooling operation to get a representation of the input. It is a good choice to utilize 2D convolution and 2D pooling to sample more meaningful features on both the time-step dimension and the feature vector dimension for text classification.

Above all, this paper proposes Bidirectional Long Short-Term Memory Networks with Two-Dimensional Max Pooling (BLSTM-2DPooling) to capture features on both the time-step dimension and the feature vector dimension. It first utilizes Bidirectional Long Short-Term Memory Networks (BLSTM) to transform the text into vectors. And then 2D max pooling operation is utilized to obtain a fixed-length vector. This paper also applies 2D convolution (BLSTM-2DCNN) to capture more meaningful features to represent the input text.

The contributions of this paper can be summarized as follows:

- This paper proposes a combined framework, which utilizes BLSTM to capture long-term sentence dependencies, and extracts features by 2D convolution and 2D max pooling operation for sequence modeling tasks. To the best of our knowledge, this work is the first example of using 2D convolution and 2D max pooling operation in NLP tasks.

- This work introduces two combined models BLSTM-2DPooling and BLSTM-2DCNN, and verifies them on six text classification tasks, including sentiment analysis, question classification, subjectivity classification, and newsgroups classification. Compared with the state-of-the-art models, BLSTM-2DCNN achieves excellent performance on 4 out of 6 tasks. Specifically, it achieves highest accuracy on Stanford Sentiment Treebank binary classification and fine-grained classification tasks.

- To better understand the effect of 2D convolution and 2D max pooling operation, this paper conducts experiments on Stanford Sentiment Treebank fine-grained task. It first depicts the performance of the proposed models on different length of sentences, and then conducts a sensitivity analysis of 2D filter and max pooling size.

The remainder of the paper is organized as follows. In Section 2, the related work about text classification is reviewed. Section 3 presents the BLSTM-2DCNN architectures for text classification in detail. Section 4 describes details about the setup of the experiments. Section 5 presents the experimental results. The conclusion is drawn in the section 6.

2 Related Work

Deep learning based neural network models have achieved great improvement on text classification tasks. These models generally consist of a projection layer that maps words of text to vectors. And then combine the vectors with different neural networks to make a fixed-length representation. According to the structure, they may divide into four categories: Recursive Neural Networks (RecNN[1]), RNN, CNN and other neural networks.

Recursive Neural Networks: RecNN is defined over recursive tree structures. In the type of recursive models, information from the leaf nodes of a tree and its internal nodes are combined in a bottom-up manner. Socher et al. (2013) introduced recursive neural tensor network to build representations of phrases and sentences by combining neighbour constituents based on the parsing tree. Irsoy and Cardie

[1]To avoid confusion with RNN, we named Recursive Neural Networks as RecNN.

(2014) proposed deep recursive neural network, which is constructed by stacking multiple recursive layers on top of each other, to modeling sentence.

Recurrent Neural Networks: RNN has obtained much attention because of their superior ability to preserve sequence information over time. Tang et al. (2015) developed target dependent Long Short-Term Memory Networks (LSTM (Hochreiter and Schmidhuber, 1997)), where target information is automatically taken into account. Tai et al. (2015) generalized LSTM to Tree-LSTM where each LSTM unit gains information from its children units. Zhou et al. (2016) introduced BLSTM with attention mechanism to automatically select features that have a decisive effect on classification. Yang et al. (2016) introduced a hierarchical network with two levels of attention mechanisms, which are word attention and sentence attention, for document classification. This paper also implements an attention-based model BLSTM-Att like the model in Zhou et al. (2016).

Convolution Neural Networks: CNN (LeCun et al., 1998) is a feedforward neural network with 2D convolution layers and 2D pooling layers, originally developed for image processing. Then CNN is applied to NLP tasks, such as sentence classification (Kalchbrenner et al., 2014; Kim, 2014), and relation classification (Zeng et al., 2014). The difference is that the common CNN in NLP tasks is made up of 1D convolution layers and 1D pooling layers. Kim (2014) defined a CNN architecture with two channels. Kalchbrenner et al. (2014) proposed a dynamic k-max pooling mechanism for sentence modeling. (Zhang and Wallace, 2015) conducted a sensitivity analysis of one-layer CNN to explore the effect of architecture components on model performance. Yin and Schütze (2016) introduced multichannel embeddings and unsupervised pretraining to improve classification accuracy. (Zhang and Wallace, 2015) conducted a sensitivity analysis of one-layer CNN to explore the effect of architecture components on model performance.

Usually there is a misunderstanding that 1D convolutional filter in NLP tasks has one dimension. Actually it has two dimensions (k, d), where k, $d \in \mathbb{R}$. As d is equal to the word embeddings size d^w, the window slides only on the time-step dimension, so the convolution is usually called 1D convolution. While d in this paper varies from 2 to d^w, to avoid confusion with common CNN, the convolution in this work is named as 2D convolution. The details will be described in Section 3.2.

Other Neural Networks: In addition to the models described above, lots of other neural networks have been proposed for text classification. Iyyer et al. (2015) introduced a deep averaging network, which fed an unweighted average of word embeddings through multiple hidden layers before classification. Zhou et al. (2015) used CNN to extract a sequence of higher-level phrase representations, then the representations were fed into a LSTM to obtain the sentence representation.

The proposed model BLSTM-2DCNN is most relevant to DSCNN (Zhang et al., 2016) and RCNN (Wen et al., 2016). The difference is that the former two utilize LSTM, bidirectional RNN respectively, while this work applies BLSTM, to capture long-term sentence dependencies. After that the former two both apply 1D convolution and 1D max pooling operation, while this paper uses 2D convolution and 2D max pooling operation, to obtain the whole sentence representation.

3 Model

As shown in Figure 1, the overall model consists of four parts: BLSTM Layer, Two-dimensional Convolution Layer, Two dimensional max pooling Layer, and Output Layer. The details of different components are described in the following sections.

3.1 BLSTM Layer

LSTM was firstly proposed by Hochreiter and Schmidhuber (1997) to overcome the gradient vanishing problem of RNN. The main idea is to introduce an adaptive gating mechanism, which decides the degree to keep the previous state and memorize the extracted features of the current data input. Given a sequence $S = \{x_1, x_2, \ldots, x_l\}$, where l is the length of input text, LSTM processes it word by word. At time-step t, the memory c_t and the hidden state h_t are updated with the following equations:

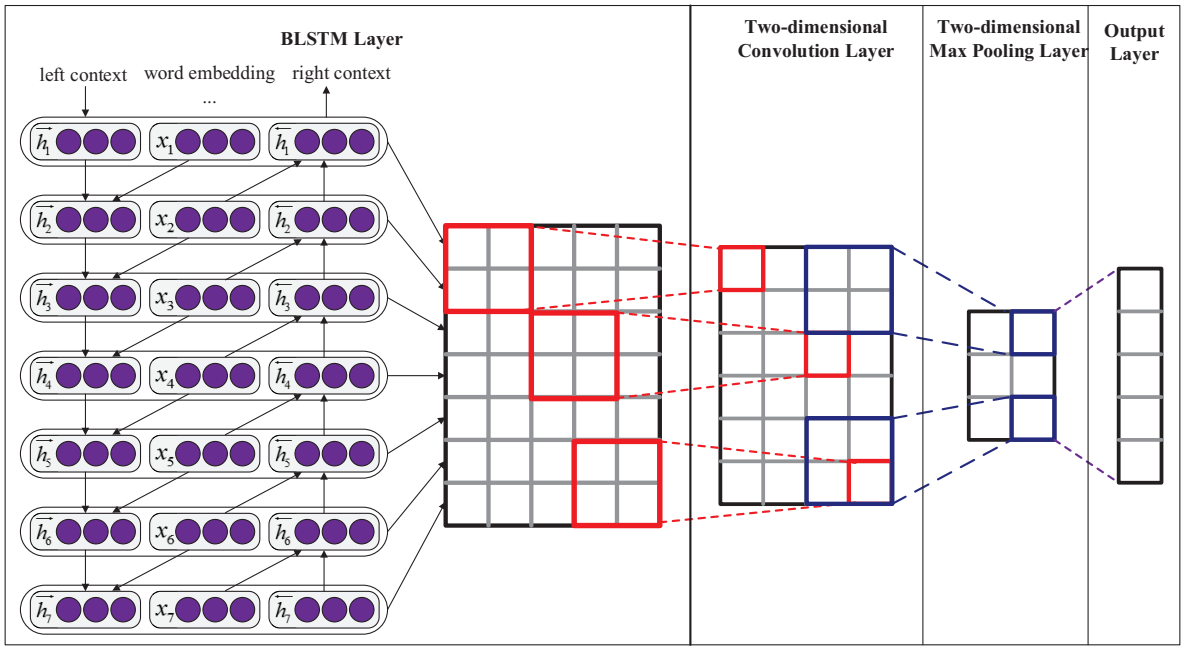


Figure 1: A BLSTM-2DCNN for the seven word input sentence. Word embeddings have size 3, and BLSTM has 5 hidden units. The height and width of convolution filters and max pooling operations are 2, 2 respectively.

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [h_{t-1}, x_t] \tag{1}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t \tag{2}$$

$$h_t = o_t \odot \tanh(c_t) \tag{3}$$

where x_t is the input at the current time-step, i, f and o is the input gate activation, forget gate activation and output gate activation respectively, $\hat{c}$ is the current cell state, σ denotes the logistic sigmoid function and $\odot$ denotes element-wise multiplication.

For the sequence modeling tasks, it is beneficial to have access to the past context as well as the future context. Schuster and Paliwal (1997) proposed BLSTM to extend the unidirectional LSTM by introducing a second hidden layer, where the hidden to hidden connections flow in opposite temporal order. Therefore, the model is able to exploit information from both the past and the future.

In this paper, BLSTM is utilized to capture the past and the future information. As shown in Figure 1, the network contains two sub-networks for the forward and backward sequence context respectively. The output of the i^{th} word is shown in the following equation:

$$h_i = [\overrightarrow{h_i} \oplus \overleftarrow{h_i}] \tag{4}$$

Here, element-wise sum is used to combine the forward and backward pass outputs.

3.2 Convolutional Neural Networks

Since BLSTM has access to the future context as well as the past context, h_i is related to all the other words in the text. One can effectively treat the matrix, which consists of feature vectors, as an 'image', so 2D convolution and 2D max pooling operation can be utilized to capture more meaningful information.

3.2.1 Two-dimensional Convolution Layer

A matrix $H = \{h_1, h_2, \ldots, h_l\}$, $H \in \mathbb{R}^{l \times d^w}$, is obtained from BLSTM Layer, where d^w is the size of word embeddings. Then narrow convolution is utilized (Kalchbrenner et al., 2014) to extract local features over H. A convolution operation involves a 2D filter $\mathbf{m} \in \mathbb{R}^{k \times d}$, which is applied to a window of k words and d feature vectors. For example, a feature $o_{i,j}$ is generated from a window of vectors $H_{i:i+k-1,\, j:j+d-1}$ by

$$o_{i,j} = f(\mathbf{m} \cdot H_{i:i+k-1,\, j:j+d-1} + b) \tag{5}$$

where i ranges from 1 to $(l - k + 1)$, j ranges from 1 to $(d^w - d + 1)$, $\cdot$ represents dot product, $b \in \mathbb{R}$ is a bias and an f is a non-linear function such as the hyperbolic tangent. This filter is applied to each possible window of the matrix H to produce a feature map O:

$$O = [o_{1,1}, o_{1,2}, \cdots, o_{l-k+1, d^w-d+1}] \tag{6}$$

with $O \in \mathbb{R}^{(l-k+1)\times(d^w-d+1)}$. It has described the process of one convolution filter. The convolution layer may have multiple filters for the same size filter to learn complementary features, or multiple kinds of filter with different size.

3.2.2 Two-dimensional Max Pooling Layer

Then 2D max pooling operation is utilized to obtain a fixed length vector. For a 2D max pooling $p \in \mathbb{R}^{p_1 \times p_2}$, it is applied to each possible window of matrix O to extract the maximum value:

$$p_{i,j} = down(O_{i:i+p_1,\, j:j+p_2}) \tag{7}$$

where $down(\cdot)$ represents the 2D max pooling function, $i = (1, 1 + p_1, \cdots, 1 + (l - k + 1/p_1 - 1) \cdot p_1)$, and $j = (1,\ 1 + p_2, \cdots, 1 + (d^w - d + 1/p_2 - 1) \cdot p_2)$. Then the pooling results are combined as follows:

$$h^* = [p_{1,1}, p_{1,1+p_2}, \cdots, p_{1+(l-k+1/p_1-1)\cdot p_1, 1+(d^w-d+1/p_2-1)\cdot p_2}] \tag{8}$$

where $h^* \in \mathbb{R}$, and the length of h^* is $\lfloor l - k + 1/p_1 \rfloor \times \lfloor d^w - d + 1/p_2 \rfloor$.

3.3 Output Layer

For text classification, the output h^* of 2D Max Pooling Layer is the whole representation of the input text S. And then it is passed to a softmax classifier layer to predict the semantic relation label $\hat{y}$ from a discrete set of classes Y. The classifier takes the hidden state h^* as input:

$$\hat{p}(y|s) = softmax\left(W^{(s)} h^* + b^{(s)}\right) \tag{9}$$

$$\hat{y} = \arg\max_y \hat{p}(y|s) \tag{10}$$

A reasonable training objective to be minimized is the categorical cross-entropy loss. The loss is calculated as a regularized sum:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} t_i \log(y_i) + \lambda \|\theta\|_F^2 \tag{11}$$

where $\boldsymbol{t} \in \mathbb{R}^m$ is the one-hot represented ground truth, $\boldsymbol{y} \in \mathbb{R}^m$ is the estimated probability for each class by softmax, m is the number of target classes, and λ is an L2 regularization hyper-parameter. Training is done through stochastic gradient descent over shuffled mini-batches with the AdaDelta (Zeiler, 2012) update rule. Training details are further introduced in Section 4.3.

Data	c	l	m	train	dev	test	$\|V\|$	$\|V_{pre}\|$
SST-1	5	18	51	8544	1101	2210	17836	12745
SST-2	2	19	51	6920	872	1821	16185	11490
Subj	2	23	65	10000	-	**CV**	21057	17671
TREC	6	10	33	5452	-	500	9137	5990
MR	2	21	59	10662	-	**CV**	20191	16746
20Ng	4	276	11468	7520	836	5563	51379	30575

Table 1: Summary statistics for the datasets. c: number of target classes, l: average sentence length, m: maximum sentence length, train/dev/test: train/development/test set size, $|V|$: vocabulary size, $|V_{pre}|$: number of words present in the set of pre-trained word embeddings, **CV**: 10-fold cross validation.

4 Experimental Setup

4.1 Datasets

The proposed models are tested on six datasets. Summary statistics of the datasets are in Table 1.

- **MR**[2]: Sentence polarity dataset from Pang and Lee (2005). The task is to detect positive/negative reviews.

- **SST-1**[3]: Stanford Sentiment Treebank is an extension of MR from Socher et al. (2013). The aim is to classify a review as fine-grained labels (very negative, negative, neutral, positive, very positive).

- **SST-2**: Same as SST-1 but with neutral reviews removed and binary labels (negative, positive). For both experiments, phrases and sentences are used to train the model, but only sentences are scored at test time (Socher et al., 2013; Le and Mikolov, 2014). Thus the training set is an order of magnitude larger than listed in table 1.

- **Subj**[4]: Subjectivity dataset (Pang and Lee, 2004). The task is to classify a sentence as being subjective or objective.

- **TREC**[5]: Question classification dataset (Li and Roth, 2002). The task involves classifying a question into 6 question types (abbreviation, description, entity, human, location, numeric value).

- **20Newsgroups**[6]: The 20Ng dataset contains messages from twenty newsgroups. We use the bydate version preprocessed by Cachopo (2007). We select four major categories (comp, politics, rec and religion) followed by Hingmire et al. (2013).

4.2 Word Embeddings

The word embeddings are pre-trained on much larger unannotated corpora to achieve better generalization given limited amount of training data (Turian et al., 2010). In particular, our experiments utilize the GloVe embeddings[7] trained by Pennington et al. (2014) on 6 billion tokens of Wikipedia 2014 and Gigaword 5. Words not present in the set of pre-trained words are initialized by randomly sampling from uniform distribution in $[-0.1, 0.1]$. The word embeddings are fine-tuned during training to improve the performance of classification.

[2]https://www.cs.cornell.edu/people/pabo/movie-review-data/
[3]http://nlp.stanford.edu/sentiment/
[4]http://www.cs.cornell.edu/people/pabo/movie-review-data/
[5]http://cogcomp.cs.illinois.edu/Data/QA/QC/
[6]http://web.ist.utl.pt/acardoso/datasets/
[7]http://nlp.stanford.edu/projects/glove/

4.3 Hyper-parameter Settings

For datasets without a standard development set we randomly select 10% of the training data as the development set. The evaluation metric of the 20Ng is the Macro-F1 measure followed by the state-of-the-art work and the other five datasets use accuracy as the metric. The final hyper-parameters are as follows.

The dimension of word embeddings is 300, the hidden units of LSTM is 300. We use 100 convolutional filters each for window sizes of (3,3), 2D pooling size of (2,2). We set the mini-batch size as 10 and the learning rate of AdaDelta as the default value 1.0. For regularization, we employ Dropout operation (Hinton et al., 2012) with dropout rate of 0.5 for the word embeddings, 0.2 for the BLSTM layer and 0.4 for the penultimate layer, we also use l2 penalty with coefficient 10^{-5} over the parameters.

These values are chosen via a grid search on the SST-1 development set. We only tune these hyper-parameters, and more finer tuning, such as using different numbers of hidden units of LSTM layer, or using wide convolution (Kalchbrenner et al., 2014), may further improve the performance.

5 Results

5.1 Overall Performance

This work implements four models, BLSTM, BLSTM-Att, BLSTM-2DPooling, and BLSTM-2DCNN. Table 2 presents the performance of the four models and other state-of-the-art models on six classification tasks. The BLSTM-2DCNN model achieves excellent performance on 4 out of 6 tasks. Especially, it achieves 52.4% and 89.5% test accuracies on SST-1 and SST-2 respectively.

BLSTM-2DPooling performs worse than the state-of-the-art models. While we expect performance gains through the use of 2D convolution, we are surprised at the magnitude of the gains. BLSTM-CNN beats all baselines on SST-1, SST-2, and TREC datasets. As for Subj and MR datasets, BLSTM-2DCNN gets a second higher accuracies. Some of the previous techniques only work on sentences, but not paragraphs/documents with several sentences. Our question becomes whether it is possible to use our models for datasets that have a substantial number of words, such as 20Ng and where the content consists of many different topics. For that purpose, this paper tests the four models on document-level dataset 20Ng, by treating the document as a long sentence. Compared with RCNN (Lai et al., 2015), BLSTM-2DCNN achieves a comparable result.

Besides, this paper also compares with ReNN, RNN, CNN and other neural networks:

- Compared with ReNN, the proposed two models do not depend on external language-specific features such as dependency parse trees.
- CNN extracts features from word embeddings of the input text, while BLSTM-2DPooling and BLSTM-2DCNN captures features from the output of BLSTM layer, which has already extracted features from the original input text.
- BLSTM-2DCNN is an extension of BLSTM-2DPooling, and the results show that BLSTM-2DCNN can capture more dependencies in text.
- AdaSent utilizes a more complicated model to form a hierarchy of representations, and it outperforms BLSTM-2DCNN on Subj and MR datasets. Compared with DSCNN (Zhang et al., 2016), BLSTM-2DCNN outperforms it on five datasets.

Compared with these results, 2D convolution and 2D max pooling operation are more effective for modeling sentence, even document. To better understand the effect of 2D operations, this work conducts a sensitivity analysis on SST-1 dataset.

5.2 Effect of Sentence Length

Figure 2 depicts the performance of the four models on different length of sentences. In the figure, the x-axis represents sentence lengths and y-axis is accuracy. The sentences collected in test set are no

NN	Model	SST-1	SST-2	Subj	TREC	MR	20Ng
ReNN	RNTN (Socher et al., 2013)	45.7	85.4	-	-	-	-
	DRNN (Irsoy and Cardie, 2014)	49.8	86.6	-	-	-	-
CNN	DCNN (Kalchbrenner et al., 2014)	48.5	86.8	-	93.0	-	-
	CNN-non-static (Kim, 2014)	48.0	87.2	93.4	93.6	-	-
	CNN-MC (Kim, 2014)	47.4	88.1	93.2	92	-	-
	TBCNN(Mou et al., 2015)	51.4	87.9	-	96.0	-	-
	Molding-CNN (Lei et al., 2015)	51.2	88.6	-	-	-	-
	CNN-Ana (Zhang and Wallace, 2015)	45.98	85.45	93.66	91.37	81.02	-
	MVCNN (Yin and Schütze, 2016)	49.6	89.4	93.9	-	-	-
RNN	RCNN (Lai et al., 2015)	47.21	-	-	-	-	96.49
	S-LSTM (Zhu et al., 2015)	-	81.9	-	-	-	-
	LSTM (Tai et al., 2015)	46.4	84.9	-	-	-	-
	BLSTM (Tai et al., 2015)	49.1	87.5	-	-	-	-
	Tree-LSTM (Tai et al., 2015)	51.0	88.0	-	-	-	-
	LSTMN (Cheng et al., 2016)	49.3	87.3	-	-	-	-
	Multi-Task (Liu et al., 2016)	49.6	87.9	94.1	-	-	-
Other	PV (Le and Mikolov, 2014)	48.7	87.8	-	-	-	-
	DAN (Iyyer et al., 2015)	48.2	86.8	-	-	-	-
	combine-skip (Kiros et al., 2015)	-	-	93.6	92.2	76.5	-
	AdaSent (Zhao et al., 2015)	-	-	**95.5**	92.4	**83.1**	-
	LSTM-RNN (Le and Zuidema, 2015)	49.9	88.0	-	-	-	-
	C-LSTM (Zhou et al., 2015)	49.2	87.8	-	94.6	-	-
	DSCNN (Zhang et al., 2016)	49.7	89.1	93.2	95.4	81.5	-
ours	BLSTM	49.1	87.6	92.1	93.0	80.0	94.0
	BLSTM-Att	49.8	88.2	93.5	93.8	81.0	94.6
	BLSTM-2DPooling	50.5	88.3	93.7	94.8	81.5	95.5
	BLSTM-2DCNN	**52.4**	**89.5**	94.0	**96.1**	82.3	**96.5**

Table 2: Classification results on several standard benchmarks. **RNTN**: Recursive deep models for semantic compositionality over a sentiment treebank (Socher et al., 2013). **DRNN**: Deep recursive neural networks for compositionality in language (Irsoy and Cardie, 2014). **DCNN**: A convolutional neural network for modeling sentences (Kalchbrenner et al., 2014). **CNN-nonstatic/MC**: Convolutional neural networks for sentence classification (Kim, 2014). **TBCNN**: Discriminative neural sentence modeling by tree-based convolution (Mou et al., 2015). **Molding-CNN**: Molding CNNs for text: non-linear, non-consecutive convolutions (Lei et al., 2015). **CNN-Ana**: A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification (Zhang and Wallace, 2015). **MVCNN**: Multichannel variable-size convolution for sentence classification (Yin and Schütze, 2016). **RCNN**: Recurrent Convolutional Neural Networks for Text Classification (Lai et al., 2015). **S-LSTM**: Long short-term memory over recursive structures (Zhu et al., 2015). **LSTM/BLSTM/Tree-LSTM**: Improved semantic representations from tree-structured long short-term memory networks (Tai et al., 2015). **LSTMN**: Long short-term memory-networks for machine reading (Cheng et al., 2016). **Multi-Task**: Recurrent Neural Network for Text Classification with Multi-Task Learning (Liu et al., 2016). **PV**: Distributed representations of sentences and documents (Le and Mikolov, 2014). **DAN**: Deep unordered composition rivals syntactic methods for text classification (Iyyer et al., 2015). **combine-skip**: skip-thought vectors (Kiros et al., 2015). **AdaSent**: Self-adaptive hierarchical sentence model (Zhao et al., 2015). **LSTM-RNN**: Compositional distributional semantics with long short term memory (Le and Zuidema, 2015). **C-LSTM**: A C-LSTM Neural Network for Text Classification (Zhou et al., 2015). **DSCNN**: Dependency Sensitive Convolutional Neural Networks for Modeling Sentences and Documents (Zhang et al., 2016).

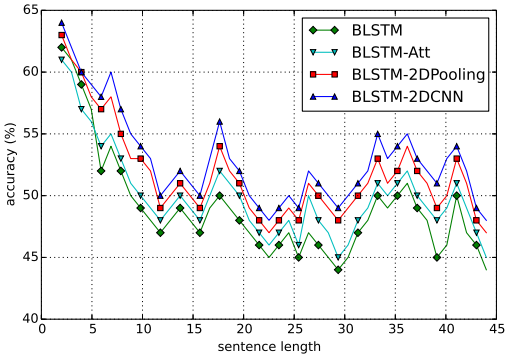


Figure 2: Fine-grained sentiment classification accuracy vs. sentence length.

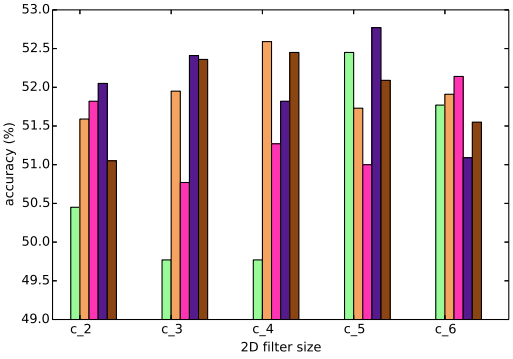


Figure 3: Prediction accuracy with different size of 2D filter and 2D max pooling.

longer than 45 words. The accuracy here is the average value of the sentences with length in the window $[l-2, l+2]$. Each data point is a mean score over 5 runs, and error bars have been omitted for clarity.

It is found that both BLSTM-2DPooling and BLSTM-2DCNN outperform the other two models. This suggests that both 2D convolution and 2D max pooling operation are able to encode semantically-useful structural information. At the same time, it shows that the accuracies decline with the length of sentences increasing. In future work, we would like to investigate neural mechanisms to preserve long-term dependencies of text.

5.3 Effect of 2D Convolutional Filter and 2D Max Pooling Size

We are interested in what is the best 2D filter and max pooling size to get better performance. We conduct experiments on SST-1 dataset with BLSTM-2DCNN and set the number of feature maps to 100.

To make it simple, we set these two dimensions to the same values, thus both the filter and the pooling are square matrices. For the horizontal axis, c means 2D convolutional filter size, and the five different color bar charts on each c represent different 2D max pooling size from 2 to 6. Figure 3 shows that different size of filter and pooling can get different accuracies. The best accuracy is 52.6 with 2D filter size (5,5) and 2D max pooling size (5,5), this shows that finer tuning can further improve the performance reported here. And if a larger filter is used, the convolution can detector more features, and the performance may be improved, too. However, the networks will take up more storage space, and consume more time.

6 Conclusion

This paper introduces two combination models, one is BLSTM-2DPooling, the other is BLSTM-2DCNN, which can be seen as an extension of BLSTM-2DPooling. Both models can hold not only the time-step dimension but also the feature vector dimension information. The experiments are conducted on six text classificaion tasks. The experiments results demonstrate that BLSTM-2DCNN not only outperforms RecNN, RNN and CNN models, but also works better than the BLSTM-2DPooling and DSCNN (Zhang et al., 2016). Especially, BLSTM-2DCNN achieves highest accuracy on SST-1 and SST-2 datasets. To better understand the effective of the proposed two models, this work also conducts a sensitivity analysis on SST-1 dataset. It is found that large filter can detector more features, and this may lead to performance improvement.

Acknowledgements

We thank anonymous reviewers for their constructive comments. This research was funded by the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDB02070005), the National High Technology Research and Development Program of China (No.2015AA015402), and the National Natural Science Foundation of China (No. 61602479).

References

Ana Margarida de Jesus Cardoso Cachopo. 2007. *Improving methods for single-label text categorization*. Ph.D. thesis, Universidade Técnica de Lisboa.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Swapnil Hingmire, Sandeep Chougule, Girish K Palshikar, and Sutanu Chakraborti. 2013. Document classification by topic labeling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 877–880. ACM.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

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

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the Association for Computational Linguistics*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, pages 2267–2273.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. *arXiv preprint arXiv:1503.02510*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. *arXiv preprint arXiv:1508.04112*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. *arXiv preprint arXiv:1504.01106*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

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

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Target-dependent sentiment classification with long short term memory. *arXiv preprint arXiv:1512.01100*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Alex Hai Wang. 2010. Don't follow me: Spam detection in twitter. In *Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on*, pages 1–10. IEEE.

Ying Wen, Weinan Zhang, Rui Luo, and Jun Wang. 2016. Learning text representation using recurrent convolutional neural network with highway layers. *arXiv preprint arXiv:1606.06905*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Wenpeng Yin and Hinrich Schütze. 2016. Multichannel variable-size convolution for sentence classification. *arXiv preprint arXiv:1603.04513*.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Rui Zhang, Honglak Lee, and Dragomir Radev. 2016. Dependency sensitive convolutional neural networks for modeling sentences and documents. In *Proceedings of NAACL-HLT*, pages 1512–1521.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. *arXiv preprint arXiv:1504.05070*.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 207.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1604–1612.

More is not always better: balancing sense distributions for all-words Word Sense Disambiguation

Marten Postma, Ruben Izquierdo Bevia, Piek Vossen
Vrije Universiteit Amsterdam
De Boelelaan 1105, 1081 HV Amsterdam
The Netherlands
m.c.postma@vu.nl, rubensanvi@gmail.com, piek.vossen@vu.nl

Abstract

Current Word Sense Disambiguation systems show an extremely poor performance on low frequent senses, which is mainly caused by the difference in sense distributions between training and test data. The main focus in tackling this problem has been on acquiring more data or selecting a single predominant sense and not necessarily on the meta properties of the data itself. We demonstrate that these properties, such as the volume, provenance, and balancing, play an important role with respect to system performance. In this paper, we describe a set of experiments to analyze these meta properties in the framework of a state-of-the-art WSD system when evaluated on the SemEval-2013 English all-words dataset. We show that volume and provenance are indeed important, but that approximating the perfect balancing of the selected training data leads to an improvement of 21 points and exceeds state-of-the-art systems by 14 points while using only simple features. We therefore conclude that unsupervised acquisition of training data should be guided by strategies aimed at matching meta properties.

1 Introduction

The task of automatically selecting the meaning of a word in a linguistic context, known as Word Sense Disambiguation (WSD), has been and is one of the main challenges for NLP. Despite the huge amount of research that has been carried out to analyze and tackle this problem, it is nevertheless still considered unsolved. The best performances on the SemEval-2013 task 12: "Multilingual Word Sense Disambiguation" English all-words subtask (from now on *sem2013-aw*) (Navigli et al., 2013) was 72.28 (Weissenborn et al., 2015)) out of competition and 64.7 in competition (Gutiérrez et al., 2013), exceeding the naive Most Frequent Sense (MFS) baseline by only 10 points at most.

Supervised machine learning systems have been successful for WSD and it is commonly believed that the problem of the low performance can be solved by providing more (manually annotated) training data. However, in addition to the sparseness of training data, another aspect of the problem is the Zipfian distribution of word senses (McCarthy et al., 2007). In both training data and test data, the same single sense of a word tends to heavily dominate, making the MFS not only a baseline that is difficult to beat (Kilgarriff, 2004), but also being used as a fall-back option by many systems.

Given the Zipfian distribution of word senses, it comes as no surprise that WSD systems have a bias towards assigning the MFS (Preiss, 2006). Building upon this observation, Postma et al. (2016) analyzed systems participating in previous Senseval and SemEval competitions with respect to their performance on the MFS and on all other senses, which are called the less frequent senses (LFS). This study convincingly showed that systems excel at identifying MFS instances, but that all systems underperform on LFS instances. For instance on the sem2013-aw task, systems performed on average around 80% in accuracy on the MFS instances, but they hardly achieved on average 20% accuracy for the LFS instances.

The main challenge for WSD is therefore not only to acquire more training data, but to also address the overfitting towards the majority case in order to boost the performance for the LFS cases. Assuming that

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3496–3506, Osaka, Japan, December 11-17 2016.

their low performance is due to both the sparseness of training data and differences in sense distributions, the challenge is how to acquire more training data and obtaining the right distribution. In this paper, we therefore investigate the following research questions:

1. **Volume**: What is the influence of using more training data without distinguishing between MFS and LFS cases?
2. **LFS**: What is the influence of only adding more LFS training examples?
3. **Provenance**: What is the effect of training using manually annotated data versus automatically annotated data?
4. **Balancing**: What is the effect of mimicking the perfect target distribution in the training data?

By providing evidence for each research question, we firstly demonstrate that more data has a small impact on the performance of a state-of-the-art supervised system (It Makes Sense (IMS), (Zhong and Ng, 2010)). Interestingly enough, the type of data has a bigger impact, more specifically, adding silver data with a better fit to the test set with respect to time and genre appears to be better than adding more manually annotated data. Our biggest contribution is however that a perfect balance of data has a major impact on the performance. Balancing the training data according to the sense distribution of the test data boosts the results for the LFS cases while maintaining the high performance for the MFS instances. Following this assumption, our experiments presented in this paper reach an overall accuracy of 86.8 as an upper ceiling. This points towards the conclusion that the evaluation data sets have so-called "long-tail details" that need to be modeled to obtain a high-performance in addition to the properties of the head of the distribution. We therefore conclude that the distributional effect is more complex than suggested in (McCarthy et al., 2004a), who focus only on the predominant sense, but also has a larger potential if acquisition is guided by strategies to match meta properties.

The paper is structured as follows: in Section 2 we discuss related work, following by a description of the resources and the evaluation framework (Section 3). The experiments and the results are presented in Section 4. Finally, we discuss the results in Section 5 and conclude the paper in Section 6. All the training data, system output files and scripts required to fully reproduce the experiments presented in this paper have been made publicly available at: `https://github.com/cltl/MoreIsNotAlwaysBetter`.

2 Related Work

Many natural phenomena can be described by power laws (Newman, 2005), ranging from city populations to name frequencies. Word sense distributions are no exception to this interesting phenomenon and also show Zipfian characteristics (McCarthy et al., 2007; Kilgarriff, 2004). This means that, given any document or collection of documents, one sense of a lemma is usually overrepresented, while the other senses are barely used. The MFS consequently plays a critical role in the task of WSD and establishes a challenge for systems. In evaluation, the MFS is usually used as a hard to beat baseline and it is therefore also used as a fallback strategy by systems.

In general, WSD systems perform well on the MFS instances, whereas their performance drops dramatically for the LFS instances (Postma et al., 2016). This is not surprising considering the Zipfian distribution of senses which has a big impact on supervised approaches. Given the dominance of MFS examples in the training data, this results in better sense representations for these cases. This unbalance contributes to the system bias towards the MFS. For unsupervised approaches, in particular graph-based approaches, the MFS of a lemma also appears to have a higher connectivity in the graph (Calvo and Gelbukh, 2015), hence also favoring it in the sense assignment phase of a WSD system. Overall, less attention has been paid to the less represented and less frequent senses, despite the fact that these provide the biggest room for improvement (Postma et al., 2016).

Although the MFS in the training data often coincides with the MFS of the task, this is not always the case, specially in cross-domain or genre scenarios. A system able to detect the predominant sense of a lemma within a target document would obtain a vast increase in accuracy. McCarthy et al. (2004b) and Koeling et al. (2005) provided the first proof of concepts in order to demonstrate this. They collected for

each target word k nearest neighbors using distributional similarity, where the similarity between each sense of the target word and the nearest neighbors is determined by WordNet similarity measures. The sense with the highest similarity is chosen as the predominant sense. Building upon the same idea, Chan and Ng (2005) apply two algorithms, a confusion matrix algorithm and an Expectation-maximization-based algorithm, to determine the sense distribution of the test data. This information is fed into the systems to improve the sense assignment.

Similar approaches have been used to tackle the task of Word Sense Induction. The work presented in Lau et al. (2012) introduces a Topic Modeling approach based on LDA (Blei et al., 2003) and HDP (Teh et al., 2012) for deriving clusters that can be identified for different senses of a word. As an intermediate outcome, the authors also provide the expected predominant senses in the target corpus. Similarly, Boyd-Graber and Blei (2007) apply a model that considers the words of a document generated coherently to the topic distribution of that document. The most likely sense for each instance of a word is predicted from this topic distribution considering the words in the context. Finally, Lau et al. (2014; 2012) focused on the detection of novel senses, which might be considered as an extreme case of unbalanced acquisition with respect to training and evaluation distributions.

Mismatches between the training and test data have been of central interest to research on domain adaptation (Daume III, 2007; Carpuat et al., 2013; Jiang and Zhai, 2007). The 2010 edition of the SemEval series (SemEval-2010) proposed a task called "All-words Word Sense Disambiguation on a Specific Domain" (Agirre et al., 2010). The aim was to analyze to what extent WSD systems are sensitive to specific domains when there is no annotated data available for that domain. In the same direction, another goal was to investigate how a general domain WSD system should be adapted to perform properly when the sense distribution is unknown and different to the sense distribution of traditional corpora used for training machine learning models. This evaluation showed that the most successful approaches included knowledge from the specific domain in different forms. For example, Kouno et al. (2015) present a framework for WSD where an unsupervised approach is applied to abstract the features across different domains and then feeds these features into a Support Vector Machine. A semi-supervised framework is introduced by Agirre and Lopez de Lacalle (2008). Several domains are used to establish cross-domain experiments, where two supervised machine learning WSD systems (k-NN and SVM) are applied on one domain and evaluated on a different one. Singular Value Decomposition (SVD) is employed to find correlations between terms, alleviate the scarcity of data, and extract examples from unlabeled data. The authors show an improvement on the cross-domain setting when including the SVD technique to add training data of the target domain.

The importance of the MFS bias in training data was already highlighted in previous work. For instance, Agirre and Martinez (2004) try to overcome the problem of this skewness by automatically acquiring examples from the Internet using an heuristic based on monosemous words. Improvement is achieved on the Senseval-2 task for nouns with less than 10 examples in SemCor (Miller et al., 1993).

Finally, researchers combined the task of WSD with Entity Linking to improve the performance on the disambiguation part. For example, Weissenborn et al. (2015) jointly disambiguate nouns and entities by exploiting the links between them in BabelNet (Navigli and Ponzetto, 2012). In addition, the Babelfy system (Moro et al., 2014) goes one step further by also making use of interlingual relations.

Despite these efforts to either acquire more data or adapt the distributions, none of these systems gained an improvement big enough to consider the problem as solved. Our work differs from these approaches mainly in that we analyze more precisely the contribution of the volume, nature, and distribution of the training data in relation to the properties of the test data. The experiments are designed to isolate each phenomenon and be able to extract meaningful conclusions. For example, whereas Agirre and Martinez (2004) acquire more data for all senses of low frequent words, we focus on adding more examples for the less frequent senses instead. Just by obtaining more examples for a word does not imply that we are able to create better sense representations for all the senses of this word. Similarly, whereas McCarthy et al. (2004b) and Koeling et al. (2005) restrict the system to find a deterministic predominant sense, we propose that the probabilities of the target data need to be used to obtain a more fine-grained behavior. We show that properly balancing the probabilities of the acquired data gives a

major improvement even using a straight-forward machine learning system with a basic set of features.

3 Methodology

In this section we describe our training and evaluation framework: which WSD system has been selected, what corpora have been used to create our models and how these models have been evaluated. We have set up the framework in such a way that we can systematically vary the type of training data used, the volume of training data and the sense distribution in this data. The impact of the variables is then tested using the same state-of-the-art supervised WSD system. We measure the overall performance in addition to the performance on the MFS and LFS cases and also generate some statistics to highlight the characteristics of the training data.

3.1 Training data sets

In our experiments, we exploit three sources of English sense annotated data: SemCor, Princeton WordNet Gloss Corpus, and what we have called "Wordnet2Wikipedia". The last corpus was created as part of the experiments and hence its creation will be described in more detail below.

Semcor (SC) The Semantic Concordance (Miller et al., 1993), or SemCor (SC), is a corpus containing approximately 240,000 sense annotated words. The tagged documents originate from the Brown corpus (Francis and Kucera, 1979) and cover various genres. The corpus contains annotations for more than 20,000 lemmas. The creators note that the main focus when creating the corpus was on word frequencies, and not on sense frequencies. This might explain why the proportion of the MFS on the total amount of annotated senses in this corpus is (unnaturally) high: more than 70%.

Princeton WordNet Gloss Corpus (GC) The Princeton WordNet Gloss Corpus, or GC, is a special sense annotated corpus. It does not contain annotations of natural text in documents but of the WordNet glosses (Fellbaum, 1998). The corpus contains more than 310,000 annotated words, which is significantly more than in SemCor. Annotations exist for approximately 15,000 lemmas, which is less than in SemCor. The MFS proportion is around 55%, which makes sense given that glosses are tagged and not natural text.

Wordnet2Wikipedia (WW) For the last source of sense annotated data, WordNet2Wikipedia, or WW, we exploit the existing relation between WordNet and Wikipedia as present in BabelNet 2.5 (Navigli and Ponzetto, 2012). BabelNet 2.5 contains the relations *direct* and *redirect*, which link a WordNet sense to a Wikipedia entry. This relation only exists for nouns in the resource. For each target sense of a target lemma: 1. we check whether a link to Wikipedia exists for this target sense. 2. if so, we extract all sentences from the Wikipedia article with the target lemma and tag the target lemmas with the target sense. By exploiting this relation in BabelNet, we are able to extract 43,000 training examples for the 751 lemmas of the sem2013-aw competition. In addition, on average, 63% of the examples we extract are examples of LFS instances.

3.2 The WSD System

We used the "It makes sense" (IMS) WSD system in our experiments (Zhong and Ng, 2010). There are several reasons for choosing IMS. First of all, IMS has shown to achieve a very high performance in the Senseval and SemEval all-words tasks. Secondly, it is an open source system that provides a flexible framework, allowing us to train and evaluate it with different kinds and amounts of data. Finally, IMS is based on a Support Vector Machine (SVM) learning engine, which is a linear classifier known to suffer not as much from unbalanced training as other learning paradigms (e.g. probabilistic learning paradigms like Naive Bayes).[1]

IMS uses three features in its default setting: surrounding words, part-of-speech tags, and collocations in a certain window around the target word. We used the system with the default setting in terms of features and learning parameters. IMS creates word experts, which means that one single classifier is

[1] http://www.win-vector.com/blog/2015/02/does-balancing-classes-improve-classifier-performance

built for a specific lemma (with the corresponding part-of-speech tag). We could easily adapt the specific word experts by manipulating the input training data while keeping all the other settings the same.[2]

3.3 Evaluation framework

For evaluation, we selected the test set from SemEval-2013, task 12: Multilingual Word Sense Disambiguation (**sem2013-aw**, (Navigli et al., 2013)). The task consisted of two disambiguation tasks: Entity Linking and Word Sense Disambiguation for English, German, French, Italian, and Spanish. We focused on the WSD part using WordNet as a sense repository. The test set contains 13 articles obtained from previous editions of the workshop on Statistical Machine Translation.[3] The articles cover different domains, ranging from sports to financial news. With respect to the English WSD part, there are 1,644 test instances in total, all nouns. The sense repository used to tag these instances is WordNet version 3.0. Based on this sense repository, we computed the number of instances annotated with the MFS, and the number of instances annotated with one of the LFS. Out of the 1,644 test instances, the MFS applies in 1,035 cases, while one of the LFS applies in the rest of 609 cases. This gives a baseline of 62.96% for the MFS heuristic.

3.4 Experiments

In order to gain insight into system performance with respect to the quality, quantity, and distribution of the training data, four main research questions have been formulated, which we will repeat here for the sake of clarity:

1. **Volume**: What is the influence of using more training data without distinguishing between MFS and LFS cases?
2. **LFS**: What is the influence of only adding more LFS training examples?
3. **Provenance**: What is the effect of training using manually annotated data versus automatically annotated data?
4. **Balancing**: What is the effect of mimicking the perfect target distribution in the training data?

We use several selection techniques to test each research question. We distinguish between a *Base* corpus from which we add all training instances available, and an *Expansion* corpus, from where subsets of instances are extracted by applying different selection techniques. SemCor (**SC**) is used as a Base and the WordNet gloss-corpus (**GC**) and the WordNet-Wikipedia corpus (**WW**) as Expansions, representing both more Volume and different types of annotation (Provenance). In addition, we defined the following selection techniques when expanding the training data:

All: all instances, both MFS and LFS, are added. This technique will allow us to provide evidence for the **Volume** and **Provenance** research questions.

LFS: only the LFS instances are added, which provides insight into the **MFS** versus **LFS** research question.

Top-down: to provide evidence for this question, we use the sense distribution of the test data starting from the MFS. For every lemma in the test set, we attempt to fit the training distribution as much as possible to the test sense distribution. In this *top-down* approach, we start with the sense with the highest relative frequency in the test set and determine the number of examples in our experiment for this sense. We then calculate the number of examples for the other senses relative to the number of examples for the sense with the highest relative frequency. When a sense does not occur in the test set, we assign to this sense a default number of 1 (top-down-1) or 5 (top-down-5) training examples, which makes sure that we perform balancing instead of filtering. Without the assignment of the default number of senses for missing senses in the test data, we would on average reduce the average polysemy to almost one, which would make the task almost solved.

[2]During our experiments, we noticed that IMS produces exceptions when using larger training sets than SemCor. We solved this by including a number of checks in the source code and by modifying two files with respect to the original IMS distribution. These files are included under the folder *ims_amended_files* in the data and scripts package delivered together with the paper. The installation script will take care of copying them to the proper place in the IMS project structure and recompile the full library. The changes have been documented in the README file.

[3]`http://www.statmt.org`

Bottom-up: to provide evidence for this question, we use the sense distribution of the test data starting from the LFS. This approach is almost identical to the *Top-down* approach but now we use the number of examples for the sense of a lemma with the lowest relative frequency in the test set as a starting point. When a sense does not occur in the test set, we assign a default number of 1 (Bottom-up-1) or 5 (Bottom-up-5) examples to the sense. Both the **Bottom-up** technique and the **Top-down** technique provide evidence for the **Balancing** research question.

To allow replication of our results and stimulate further research, all the data and scripts for training and evaluating the models have been made publicly available. There is an installation script to take care of the downloading and installation of the required libraries and data sets. There is also one main script that runs all the experiments described in Section 4. The package can be found here: `https://github.com/cltl/MoreIsNotAlwaysBetter`.

4 Results

Table 1 presents the results for all our experiments. Each experiment has been defined to analyze one of our research questions and is hence based on a certain training dataset, which is the result of applying various selection techniques, which include the type of data, the volume, and the sense distribution. For each run of an experiment (**ID**), we present the Base corpora (**Base**), the Expansion Corpora (**Expansion**), the selection technique used on the Expansion corpora (**Technique**), the overall accuracy (**Acc**), the accuracy on the MFS instances (Acc_{mfs}), the accuracy on the LFS instances (Acc_{mfs}), the micro average between the accuracy on the MFS and LFS instances (**MicroAV**), the number of training instances (**#**), the average number of training instances per lemma (**Avg#**), and the average percentage of MFS instances per lemma (Dom_{mfs}). The last three values provide information about the quantitative characteristics of the training data.

				WSD results				**Training stats**		
ID	Base	Expansion	Technique	Acc	Acc_{mfs}	Acc_{lfs}	MicroAV	#	Avg#	Dom_{mfs}
1	SC	-	-	65.60	**95.60**	14.80	55.20	22k	43	70
2	SC	GC	All	66.80	89.10	29.10	59.10	60k	110	59
3	SC	GC+WW	All	68.90	90.30	**32.50**	**61.40**	102k	187	52
4	SC	WW	All	**69.30**	**92.80**	29.40	61.10	65k	120	54
5	SC	GC	LFS	63.20	75.70	41.90	58.80	38k	71	43
6	SC	GC+WW	LFS	62.00	70.50	**47.60**	59.05	65k	120	31
7	SC	WW	LFS	**67.50**	**87.50**	33.30	**60.40**	49k	91	44
8	-	SC+GC+WW	Bottom-up1	85.40	95.90	67.50	81.70	46k	87	56
9	-	SC+GC+WW	Bottom-up5	80.40	93.30	58.50	75.90	50k	96	53
10	-	SC+GC+WW	Top-down1	**86.80**	**96.50**	**70.30**	**83.40**	45k	85	55
11	-	SC+GC+WW	Top-down5	82.00	94.40	60.90	77.65	65k	120	54

Table 1: Results of our experiments on SemEval-2013 dataset

Volume and Provenance Experiments *1*, *2*, *3*, and *4* provide insight into the Volume and Provenance research questions. We observe a clear trend that more training data indeed improves the performance. By only adding the GC corpus, the accuracy improves by 1.3 points, and if we add both GC and WW by 3.4 points. These experiments are not solely increasing the number of training examples, they also lower the MFS dominance per lemma as shown in the last column (SC scoring 70, and the expansions scoring lower 59, 42, and 54, respectively) and thus change the balancing or distribution of the senses in the training data. Both factors seem to contribute to the improved overall accuracy. Surprisingly, the best result is not found by using all of the available sense annotated data, as shown by experiment 4. By only adding the WW corpus to SC, we improve the baseline by 3.7 points despite the fact that the examples are extracted automatically (silver) compared to 1.2 points gain when adding an equal amount of data from the manual annotation of GC to SC. We suspect that the WW corpus is more similar to the test data with respect to creation time and genre since they come from Wikipedia whereas the SemCor texts go back to 1961. The GC corpus contains usage examples and definitions, which are older than the test data and timeless while also being more formal in style, written for a different purpose. Finally,

the experiments 2, 3, and 4 all dramatically improve the performance on the LFS instances by more than 15(!) points compared to the baseline in experiment 1. The price for this can be found in a slight drop on the MFS instances. In order to get a better understanding of the improvement of our best run (experiment 4), compared to the baseline, we computed the accuracy per sense rank class, as shown in Figure 1. We can observe a clear improvement of the sense rank classes 2, 3, 4, and 10+ among the LFS cases.

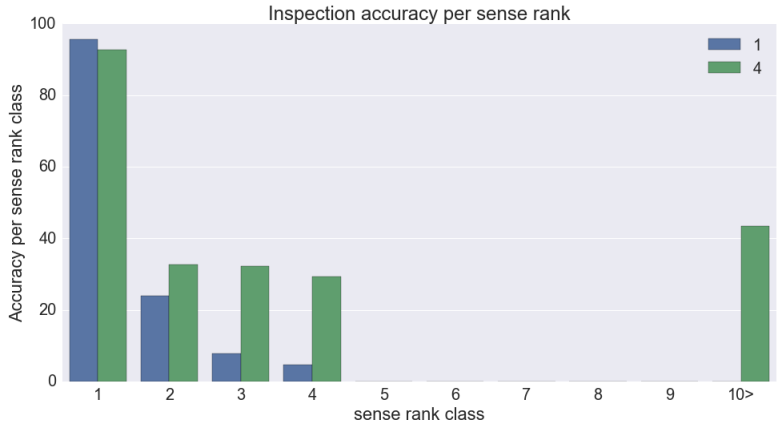


Figure 1: Performance of run 1 (left column) and 4 (right column) with respect to the performance per sense rank class.

LFS What is the effect of only adding LFS instances? We know that some senses of WordNet annotated in the test set are not annotated in SemCor, so they are not available for the training of the IMS system. What will happen if senses with few or no training data get boosted? Experiments 5 to 7 provide evidence to answer these questions. We note that the dominance of the MFS in the training instances now drops to 30-40%. On the other hand, this also results in very impressive performances on the LFS instances, between 33.33% and 47.62%. What we gain on one side, we lose on the other and the only experiment that is able to beat the baseline is run 7, which beats the baseline by 1.9 points.

Balancing Apparently the trick is to find the proper balancing between the MFS and LFS to maintain the high performance of the former and still gain in the performance of the latter. To demonstrate the potential of proper balancing, we apply the Top-down and Bottom-up balancing strategies on the training data using the test data as an approximation of the perfect balance. This can be seen as approximating the upper bound for systems being aware of the sense distribution of the test data. We see in experiments 8, 9, 10, and 11 that properly balancing the distribution gives us the highest gain (21 points) up to an overall accuracy of even 86.8. We see that it enormously boosts the accuracy of the LFS to levels normally achieved for the MFS and it also improves the accuracy of the MFS to 96.52. Note that the system still needs to make a choice between senses in this setting since all senses are represented in the model, including senses that do not occur in the test data. We also see that defining a lower bound for senses not occurring in the test set makes a difference. If we define the lower bound to 5, performance drops by 4 to 5 points, confirming the earlier experiment in which we just boost the LFS cases.

5 Discussion

In general, we note that adding more training data, obtained through manual annotation or unsupervised learning, will lead to results that exceed the standard system trained on SemCor, but will not transcend the performance to a level where we would consider the task solved. Interestingly, balancing the distribution to the task turns out to be very effective to boost the accuracy to an upper bound of around 85%, which is close to what other NLP tasks, such as entity detection and linking, achieve. In addition, the amount of training data to achieve this can be kept relatively low and can be acquired automatically. The main conclusion we draw from this is that test sets appear to contain very specific idiosyncratic details (long-tail details) when it comes to semantic tasks (as opposed to for instance syntactic tasks). These details are difficult to capture using the available training data in general. Just providing more does not help.

What helps is determining the semantic specifics of the test data. We believe that this is not just a matter of finding the predominant sense as argued by McCarthy et al. (2004a), since the distribution of both predominant and nondominant senses play a role. We have seen that also MFS distributions continue to play a role as they show shifts that need to be captured as well. We thus expect that acquisition, and likewise modeling, should be considered as problem-driven tasks rather than applying bulk acquisition. This is exactly what Figure 2 indicates. The improvements are for very specific low frequent senses and not for others. We believe that each test set has a unique long tail profile that needs to be captured by the system. In future work, we hope to perform well on this task without any prior knowledge on the sense distribution of the task, but by analyzing the properties of the texts.

The results from our experiment show that a lot of recent silver data results in better performance than a small amount of old manually-annotated data. It would be interesting to perform experiments with old silver data and manually annotated recent data. The practical obstacle towards achieving this is the availability of the data. For manual annotation, this would require a big annotation effort. For silver data, this would require an innovative approach that is different from recent approaches, which are mostly based on Wikipedia, e.g. Babelfied Wikipedia (Scozzafava et al., 2015).

One could argue that our system just uses the prior probabilities and that the machine learning is not adding anything on top of this. This is undeniable a factor but we also see that the Provenance of the training data has some impact, as the WW data help more than the GC data. Furthermore, it is not the case that the lemmas in the test set only occur in a single sense. The system does need to make a choice between different instances of a lemma in the test set for 41.4% of the lemmas.

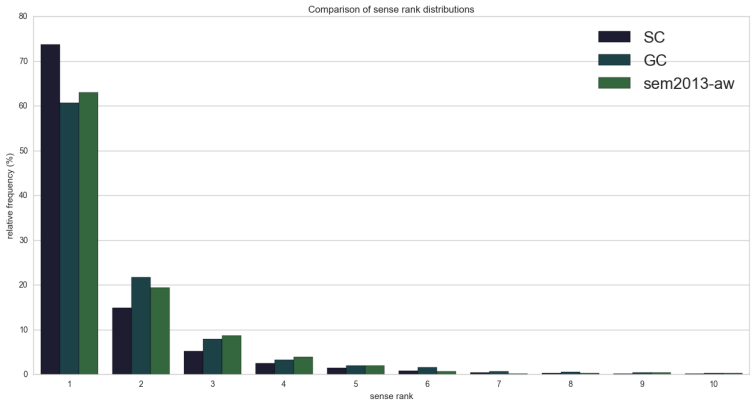


Figure 2: Comparison of sense rank distributions from the corpora: SC (left column), GC (middle column), and sem2013-aw (right column) for the sense ranks 1 till 10.

To our knowledge, only two systems (out-of-competition) achieved a better accuracy score on the **sem2013-aw** competition compared to our best run, experiment 4, from the unbalanced systems (experiments 1 to 7). The Game-Theoretic approach to WSD in Tripodi and Pelillo (2016) leads to an accuracy of 70.8, which is achieved by not only relying on the sentence as the context, but on the full document. Weissenborn et al. (2015) achieve the best result to date with an accuracy of 72.28%. The improvement is mainly due to the joint disambiguation of nouns and entities. Besides the fact that our system is more basic and uses a poorer context, we also see that even a simple system as ours can achieve far better results if we would know the distribution of the test set. We can expect that more advanced systems such as Tripodi and Pelillo (2016) and Weissenborn et al. (2015) may even exceed our best results of 86.8 with perfect balancing.

For future work, we hence intend to develop models that are problem-driven and attempt to obtain meta properties of the target data to estimate better sense distributions. An interesting starting point is a Word Sense Induction system called HCA-WSI (Bennett et al., 2016). This system could be used to determine the sense distributions of the time period in which a document in the test data was written, which would remove the dependence on sense distributions from manually annotated corpora.

6 Conclusions

We addressed the problem that most WSD systems perform well on the MFS and extremely poorly on the LFS, due to the skewness of the training data to the MFS. We analyzed the impact of adapting the training data with regard to this skewed performance: more data, better data, and balanced data. In general, we observe that more training data does indeed improve the results. However, the provenance of the data proved to be more important than the volume. We observed that automatically-acquired training data that was closer to the test data with respect to time and genre yielded better results than manually-created training data from the WordNet glosses. Finally, the real improvement was found by mimicking the sense distribution of the test data in the training data, which lead to results close to where we would consider the task solved. Hence, we argue that the solution to the task can be found in problem-driven approaches that attempt to adapt their models with respect to properties of the test set. In future work, we will focus on unsupervised methods to identify the meta properties of a test set and to capture its idiosyncratic long-tail details.

References

Eneko Agirre and Oier Lopez de Lacalle. 2008. On robustness and domain adaptation using svd for word sense disambiguation. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 17–24. Coling 2008 Organizing Committee.

Eneko Agirre and David Martinez, 2004. *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, chapter Unsupervised WSD based on Automatically Retrieved Examples: The Importance of Bias.

Eneko Agirre, Oier López de Lacalle, Christiane Fellbaum, Shu-Kai Hsieh, Maurizio Tesconi, Monica Monachini, Piek Vossen, and Roxanne Segers. 2010. SemEval-2010 Task 17: All-Words Word Sense Disambiguation on a Specific Domain. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 75–80. Association for Computational Linguistics.

Andrew Bennett, Timothy Baldwin, Han Jey Lau, Diana McCarthy, and Francis Bond. 2016. Lexsemtm: A semantic dataset based on all-words unsupervised sense distribution learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1513–1524. Association for Computational Linguistics.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Jordan Boyd-Graber and David Blei. 2007. Putop: Turning predominant senses into a topic model for word sense disambiguation. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 277–281. Association for Computational Linguistics.

Hiram Calvo and Alexander Gelbukh. 2015. Is the Most Frequent Sense of a Word Better Connected in a Semantic Network? In *Advanced Intelligent Computing Theories and Applications*, pages 491–499. Springer.

Marine Carpuat, Hal Daume III, Katharine Henry, Ann Irvine, Jagadeesh Jagarlamudi, and Rachel Rudinger. 2013. Sensespotting: Never let your parallel data tie you to an old domain. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1435–1445. Association for Computational Linguistics.

Yee Seng Chan and Hwee Tou Ng. 2005. Word Sense Disambiguation with Distribution Estimation. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 1010–1015, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, pages 256–263, Prague, Czech Republic. Association for Computational Linguistics.

Christiane Fellbaum, editor. 1998. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London, May.

Winthrop Nelson Francis and Henry Kucera. 1979. Brown corpus manual. *Brown University*.

Yoan Gutiérrez, Yenier Castañeda, Andy González, Rainel Estrada, D. Dennys Piug, I. Jose Abreu, Roger Pérez, Antonio Fernández Orquín, Andrés Montoyo, Rafael Muñoz, and Franc Camara. 2013. Umcc_dlsi: Reinforcing a ranking algorithm with sense frequencies and multidimensional semantic resources to solve multilingual word sense disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 241–249. Association for Computational Linguistics.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271. Association for Computational Linguistics.

Adam Kilgarriff, 2004. *How Dominant Is the Commonest Sense of a Word?*, pages 103–111. Springer Berlin Heidelberg, Berlin, Heidelberg.

Rob Koeling, Diana McCarthy, and John Carroll. 2005. Domain-specific sense distributions and predominant sense acquisition. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Kazuhei Kouno, Hiroyuki Shinnou, Minoru Sasaki, and Kanako Komiya. 2015. Unsupervised domain adaptation for word sense disambiguation using stacked denoising autoencoder. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation: Posters*, pages 224–231.

Han Jey Lau, Paul Cook, Diana McCarthy, David Newman, and Timothy Baldwin. 2012. Word sense induction for novel sense detection. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 591–601. Association for Computational Linguistics.

Han Jey Lau, Paul Cook, Diana McCarthy, Spandana Gella, and Timothy Baldwin. 2014. Learning word sense distributions, detecting unattested senses and identifying novel senses using topic models. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 259–270. Association for Computational Linguistics.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004a. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004b. Proceedings of senseval-3, the third international workshop on the evaluation of systems for the semantic analysis of text.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2007. Unsupervised acquisition of predominant word senses. *Computational Linguistics, Volume 33, Number 4, December 2007*.

George Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*.

Andrea Moro, Francesco Cecconi, and Roberto Navigli. 2014. Multilingual word sense disambiguation and entity linking for everybody. In *Proceedings of the 2014 International Conference on Posters & Demonstrations Track-Volume 1272*, pages 25–28. CEUR-WS.org.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. SemEval-2013 task 12: Multilingual Word Sense Disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 222–231, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Mark EJ Newman. 2005. Power laws, Pareto distributions and Zipf's law. *Contemporary physics*, 46(5):323–351.

Marten Postma, Ruben Izquierdo, Eneko Agirre, German Rigau, and Piek Vossen. 2016. Addressing the MFS Bias in WSD systems. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Judita Preiss. 2006. A detailed comparison of WSD systems: an analysis of the system answers for the Senseval-2 English all words task. *Natural Language Engineering*, 12(03):209–228.

Federico Scozzafava, Alessandro Raganato, Andrea Moro, and Roberto Navigli. 2015. Automatic Identification and Disambiguation of Concepts and Named Entities in the Multilingual Wikipedia. In *Congress of the Italian Association for Artificial Intelligence*, pages 357–366. Springer.

Yee Whye Teh, Michael Jordan, Matthew Beal, and David Blei. 2012. Hierarchical dirichlet processes. *Journal of the American Statistical Association*.

Rocco Tripodi and Marcello Pelillo. 2016. A game-theoretic approach to word sense disambiguation. *arXiv preprint arXiv:1606.07711*.

Dirk Weissenborn, Leonhard Hennig, Feiyu Xu, and Hans Uszkoreit. 2015. Multi-objective optimization for the joint disambiguation of nouns and named entities. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 596–605. Association for Computational Linguistics.

Zhi Zhong and Tou Hwee Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83. Association for Computational Linguistics.

Language classification from bilingual word embedding graphs

Steffen Eger
UKP Lab
TU Darmstadt
64289 Darmstadt

Armin Hoenen
Text Technology Lab
Goethe University
60325 Frankfurt am Main

Alexander Mehler
Text Technology Lab
Goethe University
60325 Frankfurt am Main

Abstract

We study the role of the second language in bilingual word embeddings in monolingual semantic evaluation tasks. We find strongly and weakly positive correlations between down-stream task performance and second language similarity to the target language. Additionally, we show how bilingual word embeddings can be employed for the task of semantic language classification and that joint semantic spaces vary in meaningful ways across second languages. Our results support the hypothesis that semantic language similarity is influenced by both structural similarity as well as geography/contact.

1 Introduction

Word embeddings derived from context-predicting neural network architectures have become the state-of-the-art in distributional semantics modeling (Baroni et al., 2014). Given the success of these models and the ensuing hype, several extensions over the standard paradigm (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013; Pennington et al., 2014) have been suggested, such as retro-fitting word vectors to semantic knowledge-bases (Faruqui et al., 2015), multi-sense (Huang et al., 2012; Neelakantan et al., 2014), and multi-lingual word vectors (Klementiev et al., 2012; Faruqui and Dyer, 2014; Hermann and Blunsom, 2014; Chandar et al., 2014; Lu et al., 2015; Gouws et al., 2015; Gouws and Søgaard, 2015; Huang et al., 2015; Šuster et al., 2016).

The models underlying the latter paradigm, which we focus on in the current work, project word vectors of two (or multiple) languages into a joint semantic space, thereby allowing to evaluate semantic similarity of words from different languages; see Figure 1 for an illustration. Moreover, the resulting word vectors have been shown to produce on-par or better performance even in a *monolingual* setting, e.g., when using them for measuring semantic similarity in one of the two languages involved (Faruqui and Dyer, 2014).

While multilingual word vectors have been evaluated with respect to intrinsic parameters such as embedding dimensionality, empirical work on another aspect appears to be lacking: the second language involved. For example, it might be the case that projecting two languages with very different lexical semantic associations in a joint embedding space inherently deteriorates *monolingual* embeddings as measured by performance on an intrinsic monolingual semantic evaluation task, relative to a setting in which the two languages have very similar lexical semantic associations. To illustrate, the classical Latin word *vir* is sometimes translated in English as both 'man' and 'warrior', suggesting a semantic connotation, in Latin, that is putatively lacking in English. Hence, projecting English and Latin in a joint semantic space may invoke semantic relations that are misleading for an English evaluation task. Alternatively, it may be argued that heterogeneity in semantics between the two languages involved is beneficial for monolingual evaluation tasks in the same way that uncorrelatedness in classifiers helps in combining them.

Here, we study two questions (**main contributions**). On the one hand, we are interested in the effect of language similarity on bilingual word embeddings in a (Q1) **monolingual (intrinsic) semantic evaluation task**. Thus, our first question is: how does the performance of bilingual word embeddings

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3507–3518, Osaka, Japan, December 11-17 2016.

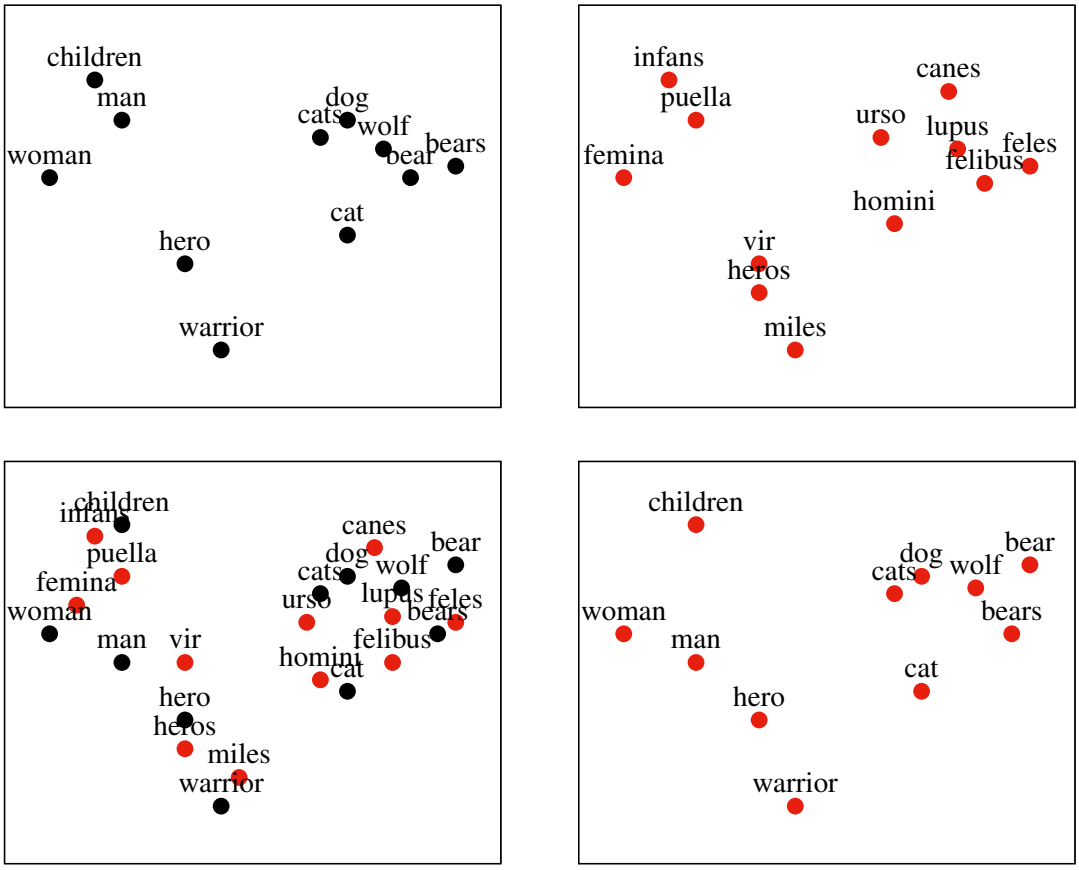


Figure 1: Monolingual embeddings (top left and top right) have been shown to capture semantic (as well as syntactic) properties of languages, here exemplarily: p = English and ℓ = Latin. Bottom left: The (idealized) goal of crosslingual embeddings is to capture these relationships across two or more languages. Bottom right: After projection in a joint semantic space, semantic (as well as syntactic) properties of words in language p have adapted to those of language ℓ. Note, in particular, the movement of *man* in these idealized plots, i.e., the different positions of *man* in top left vs. bottom right.

in monolingual semantic evaluation tasks depend on the second language involved?[1] Secondly, we ask how bilingual word embeddings can be employed for the task of semantic (Q2) **language classification**. Our approach here is simple: we project languages onto a common pivot language p so as to make them comparable. We directly use bilingual word embeddings for this. More precisely, we first project languages ℓ in a common semantic space with the pivot p by means of bilingual word embedding methods. Subsequently, we ignore language ℓ words in the joint space. Semantic distance measurement between languages then amounts to comparison of graphs that have the same nodes — pivot language words — and different edge weights — semantic similarity scores between pivot language words based on bilingual embeddings that vary as a function of the second language ℓ involved. This core idea is illustrated in Figures 1 and 2.

We show that joint semantic spaces induced by bilingual word embeddings vary in meaningful ways across second languages. Moreover, our results support the hypothesis that semantic language similarity is influenced by both genealogical language similarity and by aspects of language contact.

This work is structured as follows. Section 2 introduces our approach of constructing graphs from bilingual word embeddings and its relation to the two questions outlined. Section 3 describes our data,

[1]Our initial expectation was that bilingual word embeddings lead to better results in monolingual settings, at least for some second languages. However, this was not confirmed in any of our experiments. This may be related to our (small) data set sizes (see Section 3) or to other factors, but has no concern for the question (Q1) we are investigating.

which is based on the Europarl corpus (Koehn, 2005). Section 4 details our experiments, which we discuss in Section 5. We relate to previous work in Section 6 and conclude in Section 7.

2 Model

In this section, we formally outline our approach.

Given $N + 1$ languages, choose one of them, p, as *pivot language*. Construct N weighted networks $G_\ell^{(p)} = (V^{(p)}, E^{(p)}, \mathsf{w}_\ell^{(p)})$ as follows: nodes $V^{(p)}$ are the words of language p, graphs are fully connected, i.e., $E^{(p)} = V^{(p)} \times V^{(p)}$, and edge weights are $\mathsf{w}_\ell^{(p)}(u, v) = \mathsf{sim}(\mathbf{u}_{p,\ell}, \mathbf{v}_{p,\ell})$. The similarity function sim is, e.g., cosine similarity, and $\mathbf{u}_{p,\ell}, \mathbf{v}_{p,\ell} \in \mathbb{R}^d$ are *bilingual* word embeddings of words u and v, respectively, derived from any suitable method (see below). Here, ℓ ranges over the N *second languages*.

For (Q1) **monolingual semantic evaluation** in language p, choose p as pivot and consider $G_\ell^{(p)}$ for varying second languages ℓ. We can then evaluate semantic similarity between two language p words u and v by querying the edge weight $\mathsf{w}_\ell^{(p)}(u, v)$. This is the classical situation of (intrinsic) monolingual evaluation of bilingual word embeddings.

For (Q2) **language classification**, we compare the graphs $G_\ell^{(p)}$ across all second languages ℓ, and a fixed pivot p. Here, we have many choices how to realize distance measures between graphs, such as which metric we use and at which level we compare graphs (Bunke and Shearer, 1998; Rothe and Schütze, 2014). We choose the following: we first represent each node (pivot language word) in a graph as the vector of distances to all other nodes. That is, to $u \in V[G_\ell^{(p)}] = V^{(p)}$ we assign the vector $\mathbf{r}_u^{(\ell)} = \left(\mathsf{w}_\ell^{(p)}(u, v)\right)_{v \in V^{(p)}}$. The distance $d(G_\ell^{(p)}, G_{\ell'}^{(p)})$ between two graphs $G_\ell^{(p)}$ and $G_{\ell'}^{(p)}$ is then defined as the average distance (Euclidean norm) of the so represented nodes in the graphs: $d(G_\ell^{(p)}, G_{\ell'}^{(p)}) = \frac{1}{|V^{(p)}|} \sum_{u \in V^{(p)}} ||\mathbf{r}_u^{(\ell)} - \mathbf{r}_u^{(\ell')}||$. Finally, we define the *(syntacto-)semantic distance* $D(\ell, \ell')$ of two languages ℓ and ℓ' as the average graph distance over all $N - 1$ pivots (l and l' excluded):

$$D(\ell, \ell') = \frac{1}{N-1} \sum_{\tilde{p}} d(G_\ell^{(\tilde{p})}, G_{\ell'}^{(\tilde{p})}). \tag{1}$$

By summing over pivots, we effectively 'integrate out' the influence of the pivot language, leading to a 'pivot independent' language distance calculation. In addition, this ensures that the distance matrix D encompasses all languages, including all possible pivots.

Figure 2 illustrates our idea of projecting semantic spaces of different languages onto a common pivot.

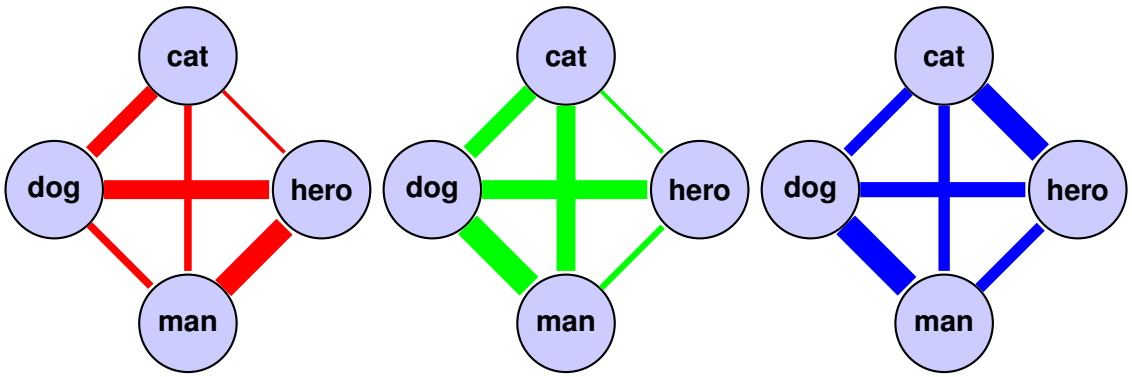


Figure 2: Schematic illustration of our approach. Repeating the "four-stage" process illustrated in Figure 1 for three different languages ℓ (marked by different colors) and the same pivot p. Edge strengths between pivot language words indicate their semantic similarity as measured by cosine distances in semantic spaces as in Figure 1 bottom right.

Bilingual embedding models: We consider two approaches to constructing bilingual word embeddings. The first is the canonical correlation analysis (**CCA**) approach suggested in Faruqui and Dyer

(2014). This takes *independently* constructed word vectors from two different languages and projects them onto a common vector space such that (one-best) translation pairs, as determined by automatic word alignments, are maximally linearly correlated. CCA relies on word level alignments and we use cdec for this (Dyer et al., 2010).

The second approach we employ is called BilBOWA (**BBA**) (Gouws et al., 2015). Rather than separately training word vectors for two languages and subsequently enforcing cross-lingual constraints, this model *jointly* optimizes monolingual and cross-lingual objectives similarly as in Klementiev et al. (2012):

$$\mathcal{L} = \sum_{\ell \in \{e,f\}} \sum_{w,h \in \mathfrak{D}^\ell} \mathcal{L}^\ell(w, h; \theta^\ell) + \lambda \Omega(\theta^e, \theta^f)$$

is minimized, where w and h are target words and their contexts, respectively, and θ^e, θ^f are embedding parameters for two languages. The terms $\mathcal{L}^\ell$ encode the monolingual constraints and the term $\Omega(\theta^e, \theta^f)$ encodes the cross-lingual constraints, enforcing similar words across languages (obtained from *sentence aligned* data) to have similar embeddings.

3 Data

For our experiments, we use the Wikipedia extracts available from Al-Rfou et al. (2013)[2] as monolingual data and Europarl (Koehn, 2005) as bilingual database. We consider two settings, one in which we take all 21 (`All21`) languages available in Europarl and one in which we focus on the 10 (`Big10`) largest languages. These languages are bg, cs, **da**, **de** , el, **en**, **es**, et, **fi**, **fr**, hu, **it**, lt, lv, **nl**, pl, **pt**, ro, sk, sl, **sv** (`Big10` languages highlighted). To induce a comparable setting, we extract in the `All21` setup: 195,842 parallel sentences from Europarl and roughly 835K (randomly extracted) sentences from Wikipedia for each of the 21 languages. In the `Big10` setup, we extract 1,098,897 parallel sentences from Europarl and 2,540K sentences from Wikipedia for each of the 10 languages involved. We note that the above numbers are determined by the minimum available for the respective two sets of languages in the Europarl and Wikipedia data, respectively. As preprocessing, we tokenize all sentences in all datasets and we lower-case all words.

4 Experiments

We first train $d = 200$ dimensional skip-gram word2vec vectors (Mikolov et al., 2013)[3] on the union of the Europarl and Wikipedia data for each language in the respective `All21` and `Big10` setting. For CCA, we then obtain bilingual embeddings for each possible combination (ℓ, ℓ') of languages in each of the two setups, by projecting these vectors in a joint space via word alignments obtained on the respective Europarl data pair. For BBA, we use the monolingual Wikipedias of ℓ and ℓ' for the monolinugal constraints, and the Europarl sentence alignments of ℓ and ℓ' for the bilingual constraints. We only consider words that occur at least 100 times in the respective data sets.

4.1 Monolingual semantic task (Q1)

We first evaluate the obtained BBA and CCA embedding vectors on monolingual $p =$ English evaluation tasks, for varying second language ℓ. The tasks we consider are WS353 (Finkelstein et al., 2002), MTurk287 (Radinsky et al., 2011), MTurk771,[4] SimLex999 (Hill et al., 2015), and MEN (Bruni et al., 2014), which are standard semantic similarity datasets for English, documented in an array of previous research. In addition, we include the SimLex999-de and SimLex999-it (Leviant and Reichart, 2015) for $p =$ German and $p =$ Italian, respectively. In each task, the goal is to determine the semantic similarity between two language p words, such as *dog* and *cat* (when $p =$ English). For the tasks, we indicate average Spearman correlation coefficients $\delta = \delta_{p,\ell}$ between

[2] https://sites.google.com/site/rmyeid/projects/polyglot.

[3] All other parameters set to default values.

[4] http://www2.mta.ac.il/ gideon/mturk771.html

- the predicted semantic similarity — measured in cosine similarity — between respective language p word pair vectors obtained when projecting p and ℓ in a joint embedding space, and

- the human gold standard (i.e., human judges have assigned semantic similarity scores to word pairs such as *dog,cat*).

Table 1 below exemplarily lists results for MTurk-287, for which p = English. We notice two trends. First, for BBA, results can roughly be partitioned into three classes. The languages pt, es, fr, it have best performances as second languages with δ values between 54% and close to 60%; the next group consists of da, nl, ro, de, el, bg, sv, sl, cs with values of around 50%; finally, fi, pl, hu, lv, lt, sk, et perform worst as second languages with δ values of around 47%. So, for BBA, the choice of second language has evidently a considerable effect in that there is ~26% difference in performance between best second language, ℓ = it, and worst second languages, ℓ = pl/sk/et. Moreover, it is apparently better to choose (semantically) similar languages — with reference to the target language p = English — as second language in this case. Secondly, for CCA, variation in results is much less pronounced. For example, the best second languages, et/lv, are just roughly 5.5% better than the worst second language, lt. Moreover, it is not evident, on first view, that performance results depend on language similarity in this case.[5]

	BBA	CCA		BBA	CCA
pt	56.54	57.48	sv	50.02	56.06
es	54.87	56.76	fi	47.41	56.76
fr	54.48	56.76	pl	47.12	56.47
it	59.70	57.12	cs	49.94	56.74
da	50.49	56.49	sl	52.96	57.05
nl	49.94	57.49	hu	48.84	56.46
ro	51.44	58.10	lv	47.55	58.81
de	50.08	58.24	lt	47.49	55.66
el	51.23	56.66	sk	47.22	57.17
bg	49.90	57.04	et	47.26	58.75

Table 1: Correlation coefficients $\delta = \delta_{p,\ell}$ in % on MTurk-287 for BBA and CCA methods, respectively, for various second languages ℓ. Second languages ordered by semantic similarity to p = English, as determined by Eq. (1); see §4.2 for specifics.

To quantify this, we systematically compute correlation coefficients τ between the correlation coefficients $\delta = \delta_{p,\ell}$ and the language distance values $D(p, \ell)$ from Eq. (1) (see §4.2 for specifics on $D(p, \ell)$). Table 2 shows that, indeed, monolingual semantic evaluation performance is consistently positively correlated with (semantic) language similarity for BBA. In contrast, for CCA, correlation is positive in eight cases and negative in six cases; moreover, coefficients are significant in only two cases. Overall, there is a strongly positive average correlation for BBA (75.75%) and a (very) weakly positive one for CCA (10.04%).

[5] As further results, we note *en passant*: CCA performed typically better than BBA, particularly in three — MEN, WS353, SimLex999 — out of our five English datasets as well as the non-English datasets. This could be due to the fact that we trained the vectors for the skip-gram model — the monolingual vectors that form the basis for CCA — on the union of Europarl and Wikipedia, while BBA used only Wikipedia as a monolingual basis. Other explanations could be the particular default hyperparameters chosen, which may have coincidentally favored CCA, or the fact that CCA uses only 1-best word alignments for projection; see Section 5 for further discussions. Moreover, in no case did we find that either BBA or CCA outperformed the purely monolingually constructed skip-gram vectors on the English evaluation task. On the one hand, this may be due to our rather small bilingual databases — containing just roughly 200K and 1,000K parallel sentences. On the other hand, while this finding is partly at odds with Faruqui and Dyer (2014), who report large improvements for bilingual word vectors over monolingual ones in some settings, it is (more) in congruence with Lu et al. (2015) and Huang et al. (2015).

$p =$		BBA	CCA
en	WS353-All21	63.75**	-6.16
	WS353-Big10	93.33**	58.33†
	MTurk287-All21	80.75***	5.11
	MTurk287-Big10	88.33**	-21.66
	MTurk771-All21	74.28***	-19.24
	MTurk771-Big10	93.33***	11.66
	SimLex999-All21	83.60***	11.57
	SimLex999-Big10	73.33*	-20.00
	MEN-All21	70.82***	-11.27
	MEN-Big10	94.99***	41.66
de	SimLex999-de-All21	60.45**	10.07
	SimLex999-de-Big10	73.33*	-31.66
it	SimLex999-it-All21	48.57*	73.83***
	SimLex999-it-Big10	61.66†	38.33
	Avg.	75.75	10.04

Table 2: Correlation τ, in %, between language similarity and monolingual semantic evaluation performance. For example, on WS353 in the Big10 setup, the more a language, say ℓ = French, is (semantically) similar to p = English, the more is it likely that correlations $\delta_{p,\ell}$ are large, when word pair similarity of p = English words is measured from embedding vectors that have been projected in a joint French-English semantic embedding space. More precisely, the exact correlation values are 93.33% and 58.33%, respectively, depending on whether vectors have been projected via BBA or CCA. '***' means significant at the 0.1% level; '**' at the 1% level, '*' at the 5% level, '†' at the 10% level.

	Geo	WALS	Sem
Geo		5%/45%**	40%***/65%***
WALS			23%**/62%***
Sem			

Table 3: Correlation between dist. matrices, Mantel test, All21/Big10.

4.2 Language classification (Q2)

Finally, we perform language classification on the graphs $G_\ell^{(p)}$ as indicated in §2. Since we use two different methods for inducing bilingual word embeddings, we obtain two distance matrices.[6] Figure 3 below shows a two-dimensional representation of all 21 languages obtained from *averaging* the BBA and CCA distance matrices in the All21 setup, together with a k-means cluster assignment for $k = 6$. We note a grouping together of es, pt, fr, en, it; nl, da, de, sv; fi, et; ro, bg, el; hu, pl, cs, sk, sl; and lt, lv. In particular, {es, pt, fr, it, en} appear to form a homogeneous group with, consequently, similar semantic associations, as captured by word embeddings. Observing that fi is relatively similar to sv, which is at odds with genealogical/structural language classifications, we test another question, namely, whether the resulting semantic distance matrix is more similar to a distance matrix based on genealogical/structural relationships or to a distance matrix based on geographic relations. To this end, we determine the degree of structural similarity between two languages as the number of agreeing features (a feature is, e.g., *Number of Cases*) in the WALS[7] database of structural properties of languages divided by the number of total features available for the language pair (Cysouw, 2013a). For geographic distance, we use the dataset from Mayer and Zignago (2011) which lists distances between countries. We make the simplifying as-

[6] For All21, these two distance matrices have a correlation of close to 70% (Mantel test), and of 73% for Big10. Hence, overall, semantic language classification results produced by either of the two methods alone — BBA or CCA — are expected to be very similar.

[7] http://wals.info/

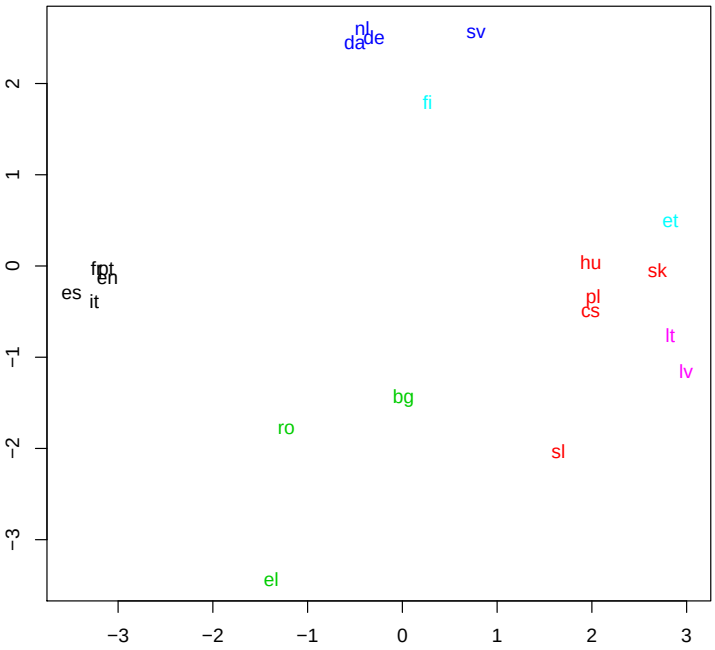


Figure 3: Two-dimensional PCA (principal component analysis) projection of average distance matrices as described in the text.

sumption that, e.g., language it and country Italy agree, i.e., it is spoken in Italy (exclusively). Table 3 shows that geographic distance correlates better with our semantic distance calculation than does WALS structural similarity under the Mantel test measure. This may hint at an interesting result: since semantics is changing fast, it may be more directly influenced by contact phenomena than by genealogical processes that operate on a much slower time-scale.

Note that our results are in accordance with the assumption that the probability of borrowing and geographical distance are inversely correlated (Cysouw, 2013b). In our case, this may relate to semantic loans (adopting the semantic neighborhoods of loaned words within the target language) rather than to structural or grammatical borrowings. That is, geographically related languages exhibit a higher probability to borrow words from each other together with the same range of semantic associations. At least, this hypothesis is not falsified by our experiment.

5 Discussion

Our initial expectation was that 'distant' second languages ℓ — in terms of language similarity — would greatly deteriorate monolingual semantic evaluations in a target language p, as we believed they would invoke 'unusual' semantic associations from the perspective of p. Such a finding would have been a word of caution regarding with which language to embed a target language p in a joint semantic space, if this happens for the sake of improving monolingual semantic similarity in p. We were surprised to find that only BBA was sensitive to language similarity in our experiments in this respect, whereas CCA seems quite robust against choice of second language. An explanation for this finding may be the different manners in which both methods induce joint embedding spaces: While CCA takes independently constructed vector spaces of two languages, BBA jointly optimizes mono- and bilingual constraints and may thus be more sensitive to the interplay, and relation, between both languages. Another plausible explanation is that CCA uses only *1-best* alignments for projecting two languages in a joint semantic space. Thus, it may be less sensitive to varying polysemous associations across different languages (cf. our *vir* example in Section 1), and hence less adequate for capturing cross-lingual polysemy.[8]

In terms of language similarity, we mention that our approach is formally similar to approaches as in

[8]Thus, we would also expect CCA to perform better in monolingual intrinsic evaluations (as our experiments have partly confirmed) and BBA to perform better in multilingual intrinsic evaluations. We thank one reviewer for pointing this out.

(Eger et al., 2015; Asgari and Mofrad, 2016) and others. Namely, we construct graphs, one for each language, and compare them to determine language distance. Compared to Eger et al. (2015), our approach differs in that they use translations in a second language ℓ to measure similarity between pivot language p words. This idea also underlies very well-known lexical semantic resources such as the paraphrase database (PPDB) (Bannard and Callison-Burch, 2005; Ganitkevitch et al., 2013); see also Eger and Sejane (2010). In contrast, we directly use bilingual embeddings for this similarity measurement by jointly embedding p and ℓ, which are arguably best suited for this task. Our approach also differs from Eger et al. (2015) in that we do not apply a random-surfer process to our semantic graphs.

We finally note that the linguistic problem of (semantic) language classification, as we consider, involves some vagueness as there is de facto no gold standard that we can compare to. Reasonably, however, languages should be semantically similar to a degree that reflects structural, genealogical, and contact relationships. One approach may then be to disentangle or, as we pursued here, (relatively) weigh each of these effects.

From an application perspective, our approach allows for enriching (automatically generated) lexica. This relates, for example, to the generation of sentiment lexica listing prior polarities for selected lexemes (Sonntag and Stede, 2015). Since the acquisition of such specialized information (e.g., by annotation) is cost-intensive, approaches are needed that allow for automatically generating or extending such resources especially in the case of historical languages (e.g., Latin). Here our idea is to start from pairs of semantically (most) similar languages in order to induce polarity cues for words in the target language as a function of their distances to selected seed words in the pivot language, for which polarities are already known. By averaging over groups of semantically related pivot languages, for which sentiment lexica already exist, the priority listings for the desired target language may stabilize. Obviously, this procedure can be applied to whatever lexical information to be annotated automatically (e.g., grammatical or semantic categories like agency, animacy etc. as needed, for example, for semantic role labeling (Palmer et al., 2010)).

A second application scenario relates to measuring (dis-)similarities of translations and their source texts (Baker, 1993): starting from our model of bilingual semantic spaces, we may ask, for example, whether words for which several alternatives exist within the target language tend to be translated by candidates that retain most of their associations within the source language – possibly in contradiction to frequency effects. Such a finding would be in line with Toury's notion of interference (Toury, 1995) according to which translations reflect characteristics of the source language – the latter leaves, so to speak, fingerprints within the former. Such a finding would bridge between the notion of interference in translation studies and distributional semantics based on deep learning.

6 Related work

Besides the mono- and multilingual word vector representation research that forms the basis of our work and which has already been referred to, we mention the following three related approaches to language classification. Koehn (2005) compares down-stream task performance in SMT to language family relationship, finding positive correlation. Cooper (2008) measures semantic language distance via bilingual dictionaries, finding that French appears to be semantically closer to Basque than to German, supporting our arguments on contact as co-determining semantic language similarity. Bamman et al. (2014) and Kulkarni et al. (2015b) study semantic distance between dialects of English by comparing region specific word embeddings.

Studying geographic variation of (different) languages is also closely related to studying temporal variation within one and the same language (Kulkarni et al., 2015a), with one crucial difference being the need to find a common representation in the former case. Word embeddings — in particular, monolingual ones — can also be used to address the latter scenario (Eger and Mehler, 2016; Hamilton et al., 2016).

In terms of classifying languages, the work that is closest to ours is that of Asgari and Mofrad (2016). A key difference between their approach and ours is that, in order to achieve a common representation between languages, they translate words. This has the disadvantage that translation pairs need to be known, which typically requires large amounts of parallel text. In contrast, bilingual word embeddings, which

form the basis of our experiments, can be generated from as few as ten translation pairs, as demonstrated in Zhang et al. (2016).

There is by now a long-standing tradition that compares languages via analysis of complex networks that encode their words and the (semantic) relationships between them (Cancho and Solé, 2001; Gao et al., 2014). These studies often only look at very abstract statistics of networks such as average path lengths and clustering coefficients, rather than analyzing them on a level of content of their nodes and edges. In addition, they often substitute co-occurrence as a proxy for semantic similarity. However, as Asgari and Mofrad (2016) point out, co-occurrence is a naive estimate of similarity; e.g., synonyms rarely co-occur.

7 Conclusion

Using English, German and Italian as pivot languages, we show that the choice of the second language may significantly matter when the resulting space is used for monolingual semantic evaluation tasks. More specifically, we show that the goodness of this choice is influenced by genealogical similarity and by (geographical) language contact. This finding may be important for the question which languages to integrate in multilingual embedding spaces (Huang et al., 2015). Moreover, we show that semantic language similarity — estimated on the basis of bilingual embedding spaces as suggested in this work — may be better predicted by contact than by genealogical relatedness. The validation of this hypothesis by means of bigger data sets will be the object of future work.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ehsaneddin Asgari and Mohammad R.K. Mofrad. 2016. Comparing fifty natural languages and twelve genetic languages using word embedding language divergence (weld) as a quantitative measure of language distance. In *Proceedings of the Workshop on Multilingual and Cross-lingual Methods in NLP*, pages 65–74, San Diego, California, June. Association for Computational Linguistics.

Mona Baker. 1993. Corpus linguistics and translation studies: Implications and applications. In Mona Baker, Francis Gill, and Elena Tognini-Bonelli, editors, *Text and Technology: In Honour of John Sinclair*, pages 233–250, Amsterdam/Philadelphia. John Benjamins.

David Bamman, Chris Dyer, and Noah A. Smith. 2014. Distributed representations of geographically situated language. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 828–834, Baltimore, Maryland, June. Association for Computational Linguistics.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 597–604, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland, June. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Int. Res.*, 49(1):1–47, January.

Horst Bunke and Kim Shearer. 1998. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3-4):255–259, March.

Ramon F. Cancho and Richard V. Solé. 2001. The small world of human language. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 268(1482):2261–2265, November.

A. P. Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M. Khapra, Balaraman Ravindran, Vikas C. Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 1853–1861.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Martin C. Cooper. 2008. Measuring the semantic distance between languages from a statistical analysis of bilingual dictionaries. *Journal of Quantitative Linguistics*, 15(1):1–33.

Michael Cysouw, 2013a. *Approaches to Measuring Linguistic Differences*, chapter Predicting language learning difficulty, pages 57–82. De Gruyter Mouton.

Michael Cysouw. 2013b. Disentangling geography from genealogy. In Peter Auer, Martin Hilpert, Anja Stukenbrock, and Benedikt Szmrecsanyi, editors, *Space in Language and Linguistics: Geographical, Interactional, and Cognitive Perspectives*, pages 21–37, Berlin. De Gruyter Mouton.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Steffen Eger and Alexander Mehler. 2016. On the linearity of semantic change: Investigating meaning variation via dynamic graph models. In *Proceedings of ACL 2016*. Association for Computational Linguistics.

Steffen Eger and Ineta Sejane. 2010. Computing Semantic Similarity from Bilingual Dictionaries. In *Proceedings of the 10th International Conference on the Statistical Analysis of Textual Data (JADT-2010)*, pages 1217–1225, Rome, Italy, June. JADT-2010.

Steffen Eger, Niko Schenk, and Alexander Mehler. 2015. Towards semantic language classification: Inducing and clustering semantic association networks from Europarl. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 127–136, Denver, Colorado, June. Association for Computational Linguistics.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131, January.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia, June. Association for Computational Linguistics.

Yuyang Gao, Wei liang, Yuming Shi, and Qiuling Huang. 2014. Comparison of directed and weighted co-occurrence networks of six languages. *Physica A: Statistical Mechanics and its Applications*, (393):579–589.

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1386–1390, Denver, Colorado, May–June. Association for Computational Linguistics.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 748–756.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 58–68.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 873–882, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kejun Huang, Matt Gardner, Evangelos E. Papalexakis, Christos Faloutsos, Nikos D. Sidiropoulos, Tom M. Mitchell, Partha Pratim Talukdar, and Xiao Fu. 2015. Translation invariant word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1084–1088.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*, pages 1459–1474, Mumbai, India, December. The COLING 2012 Organizing Committee.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015a. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 625–635, New York, NY, USA. ACM.

Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. 2015b. Freshman or fresher? quantifying the geographic variation of internet language. *CoRR*, abs/1510.06786.

Ira Leviant and Roi Reichart. 2015. Separated by an un-common language: Towards judgment language informed vector space modeling. *CoRR*, abs/1508.00106.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 250–256, Denver, Colorado, May–June. Association for Computational Linguistics.

Thierry Mayer and Soledad Zignago. 2011. Notes on cepiis distances measures: The geodist database. Working Papers 2011-25, CEPII.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069, Doha, Qatar, October. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Nianwen Xue. 2010. *Semantic Role Labeling*. Morgan & Claypool Publishers, San Rafael.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 337–346, New York, NY, USA. ACM.

Sascha Rothe and Hinrich Schütze. 2014. Cosimrank: A flexible & efficient graph-theoretic similarity measure. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1392–1402, Baltimore, Maryland, June. Association for Computational Linguistics.

Jonathan Sonntag and Manfred Stede. 2015. Sentiment analysis: Whats your opinion? In Chris Biemann and Alexander Mehler, editors, *Text Mining: From Ontology Learning to Automated Text Processing Applications. Festschrift in Honor of Gerhard Heyer*, Theory and Applications of Natural Language Processing. Springer, Heidelberg.

Gideon Toury. 1995. *Descriptive Translation Studies and Beyond*. John Benjamins, Amsterdam/Philadelphia.

Simon Šuster, Ivan Titov, and Gertjan van Noord. 2016. Bilingual learning of multi-sense embeddings with discrete autoencoders. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1346–1356, San Diego, California, June. Association for Computational Linguistics.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016. Ten pairs to tag – multilingual pos tagging via coarse mapping between embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1307–1317, San Diego, California, June. Association for Computational Linguistics.

Word Embeddings, Analogies, and Machine Learning: Beyond $King$ - Man + $Woman$ = $Queen$

Aleksandr Drozd[†], **Anna Gladkova**[‡], **Satoshi Matsuoka**[†]
† Tokyo Institute of Technology, Meguro-ku, Tokyo 152-8550, Japan
alex@smg.is.titech.ac.jp, matsu@is.titech.ac.jp
‡ The University of Tokyo, Meguro-ku, Tokyo 153-8902 Japan
gladkova@phiz.c.u-tokyo.ac.jp

Abstract

Solving word analogies became one of the most popular benchmarks for word embeddings on the assumption that linear relations between word pairs (such as $king$:man :: $woman$:$queen$) are indicative of the quality of the embedding. We question this assumption by showing that the information not detected by linear offset may still be recoverable by a more sophisticated search method, and thus is actually encoded in the embedding.

The general problem with linear offset is its sensitivity to the idiosyncrasies of individual words. We show that simple averaging over multiple word pairs improves over the state-of-the-art. A further improvement in accuracy (up to 30% for some embeddings and relations) is achieved by combining cosine similarity with an estimation of the extent to which a candidate answer belongs to the correct word class. In addition to this practical contribution, this work highlights the problem of the interaction between word embeddings and analogy retrieval algorithms, and its implications for the evaluation of word embeddings and the use of analogies in extrinsic tasks.

1 Introduction

Discovering analogical relations is currently one of the most popular benchmarks for word embeddings. This trend started after (Mikolov et al., 2013b) showed that proportional analogies (a is to b as c is to d) can be solved by finding the vector closest to the hypothetical vector calculated as c - a + b (e.g. $king$ - man + $woman$ = $queen$). Many subsequent studies used this approach to evaluate the performance of word embeddings with the Google test set (Mikolov et al., 2013a); the top current result is over 80% accuracy (Pennington et al., 2014). The assumption is that a "good" word embedding encodes linguistic relations in such a way that they are identifiable via linear vector offset (see section 2).

Analogies are interesting not only as a benchmark, but also potentially as a method for discovering linguistic relations (Turney, 2008). They are already used for morphological analysis (Lavallée and Langlais, 2010), word sense disambiguation (Federici et al., 1997), semantic search (Cohen et al., 2015), and even for broad-range detection of both morphological and semantic features (Lepage and Goh, 2009). However, Mikolov's study was a demonstration of how word embeddings capture linguistic relations, rather than a proposal of linear vector offset as a method for their discovery. It was later shown to not work as well for a wider range of relations (Köper et al., 2015; Gladkova et al., 2016).

This study questions the underlying assumption that linguistic relations should translate to linear relations between vectors rather than a more complex correspondence pattern. We show that relations not detected by vector offset may be recoverable by other methods, and thus are actually encoded in the embedding. The method we propose is based on learning the target relation from multiple word pairs, since reliance on single word pair makes linear vector offset sensitive to word idiosyncrasies. A naive average-based baseline outperforms the state-of-the-art. A more sophisticated machine-learning algorithm achieves further improvement (up to 30% for some embeddings and linguistic relations) by combining similarity to a source word vector ($king$) with the estimate of whether a candidate answer ($queen$) belongs to the correct class of words ("woman").

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3519–3530, Osaka, Japan, December 11-17 2016.

2 State of the Art: Analogical Reasoning Based on the Offset of Word Vectors

The starting point for this study is (Mikolov et al., 2013a), the first work to demonstrate the possibility of capturing relations between words as the offset of their vectors. The answer to the question "a is to b as c is to ?" is represented by hidden vector d, calculated as $argmax_{d \in V}(sim(d, c - a + b))$. Here V is the vocabulary (excluding word vectors a, b and c), and sim is a similarity measure, for which Mikolov and most other researchers use angular distance between vectors u and v: $sim(u, v) = cos(u, v) = \frac{u \cdot v}{||u||||v||}$ We will refer to this method as **3CosAdd**. The intuition behind it is that the position of, e.g., vector man relative to $king$ should be roughly the same as the position of $woman$ relative to $queen$. Vylomova et al. (2016) use this method as a basis for learning lexical relations with spectral clustering and Support Vector Machines (SVM).

An alternative method was introduced by Levy and Goldberg (2014) who propose to calculate the hidden vector as $argmax_{d \in V}(cos(d - c, b - a))$. They report that this method produces more accurate results for some categories. Its essence is that it accounts for $d - c$ and $b - a$ to share the same direction and discards lengths of these vectors. We will refer to this method as **PairDistance**.

Linzen (2016) reports results of experiments with 6 more functions, including reversing the relation, returning simply the nearest neighbour of the c word, and the word most similar to both b and c. None of these functions outperformed 3CosAdd and PairDistance consistently. Reversal was beneficial for some relations, but it is only applicable to symmetrical one-on-one relations. Crucially, when the words a, b and c are not excluded from the set of possible candidates, the performance drops to zeroes, and for the singular-plural noun category the correct answers are obtained with 70% accuracy as simply the nearest neighbours of the c word.

3 The Alternative: Learning From Multiple Examples

3.1 Naive Approach

The vector offset approach relies on a single pair of words, which makes it sensitive to noise and word idiosyncrasies, such as differences in polysemy networks. Consider the above $king$:$queen$ example: depending on the corpus, there may be more differences in their vectors than just masculinity/femininity. $Queen$ is also a musical group, and therefore appears in many contexts in which $king$ does not appear.

The alternative is to learn the relation from a set of example pairs. The "naive" baseline would be a simple average of the offset between every pair of vectors in the training set: $argmax_{d \in V}(sim(d, c + avg\_offset))$, where $avg\_offset = \frac{\sum_{i=0}^{m} a_i}{m} - \frac{\sum_{i=0}^{n} b_i}{n}$ and a_i and b_i represents words from source and target classes. We refer to this method as **3CosAvg**. To the best of our knowledge, this has not been explored before - surprising as it is.

3.2 LRCos Method

We propose an alternative approach to discovering linguistic relations with analogies based on a set of word pairs that have the same relation, such as the country:capital relation shown in Table 1:

Source	Target
France	Paris
Japan	Tokyo
China	Beijing

Table 1: Example analogy pairs set: capitals

In this set the right-hand-side and left-hand-side elements represent coherent groups of words - in this example, "countries" and "capitals". We shall refer to the left-hand-side of such analogies as the "source class", and to the right-hand-side - as "target class". Given a set of such word pairs, the question "what is related to $France$ as $Tokyo$ is related to $Japan$?" can be reformulated as "what word belongs to the same class as $Tokyo$ and is the closest to $France$?"

We detect words belonging to the target class (e.g. "capitals") with logistic regression[1]. Given a set of word pairs (e.g. *Japan:Tokyo*), the available target words are used as positive samples, and source words, along with random words from the dictionary, as negative samples. The number of random words and other parameters of logistic regression such as regularization strength can affect the performance of the classifier, but in our pilot tests no set of parameters yielded significant gains over the default choices. In experiments reported below the number of random words was equal to the number of positive samples. We used logistic regression implementation from Python `linear_model.LogisticRegression` module from Python's sklearn module version 0.17.1 with default parameters[2].

The probability of a word being the correct answer for a given analogy is calculated by combining (in this study, multiplying) the probability of this word belonging to the target class, and its similarity with the vector a measured using angular distance. Theoretically this enables further optimization through different weighting schemes, although our test did not show significant gains over simple multiplication.

Both set-based methods (3CosAvg and LRCos) were evaluated in exclude-n scheme. Given a set of 50 word pairs, n of them are excluded, and remaining are used for obtaining the "rule" of transfer (this part differs by the method). Then each of the n pairs become the question, and the learned "rule" is used to try to derive the answer. Larger n speeds up the computation and can be used for larger datasets, while n=1 will maximize the number of training elements to obtain the "rule". In this study we used n=2.

3.3 Filtering Vector Dimensions

Performance of LRCos could theoretically benefit from forcing the similarity metric to ignore irrelevant features. Consider the task of identifying plural forms of nouns (e.g. *phone:phones*, *banana:bananas*). The two linguistic classes (in this case singular and plural nouns) necessarily introduce some dissimilarity between $phone$ and $phones$.

Assuming that this dissimilarity is shared by all word pairs, we can learn which features are responsible for it, and exclude them in the similarity estimation step. This should give an advantage to words from the target class. Ideally, when the "plurality" features are excluded, the $phones$ vector should be the most similar to the $phone$ vector. To implement this method we have additionally trained C-Support Vector Classifier (sklearn.svm.SVC) with a linear kernel to discriminate between "left" and "right" words and used complimentary values of the weights assigned to the features it learned to scale individual dimensions. We will refer to this "filtered" variant of LRCos method as LRCosF.

4 Corpora and Word Embeddings

Word embeddings represent words in the vocabulary as vectors that can be derived directly from co-occurrence counts ("explicit models") or learned implicitly by neural nets (see (Erk, 2012) for general overview of the field). It is currently debated whether explicit and implicit models are conceptually different (Levy and Goldberg, 2014), or whether the latter have an advantage over the former (Baroni et al., 2014). To contribute to the ongoing debate, this work explores both types of models.

The source corpus combines an English Wikipedia snapshot from July 2015 (1.8B tokens), Araneum Anglicum Maius (1.2B) (Benko, 2014) and ukWaC (2B) (Baroni et al., 2009) (uncased, words occurring less than 100 times were discarded). The resulting vocabulary size is 301,949 words.

The **SVD**-based explicit model is built upon co-occurrence matrix weighted by Positive Pointwise Mutual Information (PPMI, Church and Hanks (1990)). The co-occurrence matrix was computed using the co-occurrence extraction kernel by (Drozd et al., 2015) with a window size of 8. Singular Value Decomposition (SVD) transformation was used to obtain low-rank approximation of the sparse co-occurrence matrix. SVD factorizes $m \times n$ real or complex matrix M in a form $M = U\Sigma V^*$ (Golub and Van Loan, 1996), and embeddings can be obtained as $U\Sigma$. Σ is a diagonal matrix the elements of which reflect how much of a variance of original data is captured in a given dimension. We used the technique by (Caron, 2001) of rising Σ matrix element-wise to the power of a where $0 < a \leq 1$ to give a boost to dimensions

[1]We tried several other classification algorithms such as SVM with linear, polynomial and radial basis function kernels, but neither of them yielded higher classification accuracy, and they also were more computationally expensive. Building a classifier directly from the set of vector offsets of all word pairs was also not successful.

[2]Source code and additional materials are available at `http://vsm.blackbird.pw`

with smaller variance, with a =0.1 for 300-dimensional embeddings and a=0.6 for the rest. We have used embeddings of size 300 and 1000 for comparison with GloVe and Skip-Gram models, and sizes 100-1200 for studying the dimensionality effect. Finally, we have normalized each embedding vector individually, as we have found that it increases the performance of SVD-based embeddings.

As representatives of implicit models we used GloVe and Skip-Gram. The **GloVe** model was trained with the original software by (Pennington et al., 2014) with 300 dimensions, window size 8, 20 iterations, parameters x_{max}= 100, a = 3/4. **Skip-Gram** embeddings were trained with original software by Mikolov (Mikolov et al., 2013a) in skip-gram mode, with windows size 8, 25 negative samples, 5 iterations, "sample" parameter (for down-sampling of frequent words) equal to 1e-4. It is also worth noting that co-occurrences for the SVD model were collected with respect to sentence boundaries, while GloVe and Skip-Gram models disregard them.

The performance of word embeddings can be drastically affected by their parameters (Levy et al., 2015; Lai et al., 2015), which prompts parameter searching for different tasks. However, accuracy of solving word analogies also varies immensely for different linguistic relations (Gladkova et al., 2016). Optimizing for "average accuracy" on a diverse set of relations may not be meaningful, as it does not necessarily guarantee better performance on a particular relation. Therefore we did not attempt such parameter search for our models. However, in section 5.1 we will test our embeddings on the widely used Google analogy test to show that they are generally on the par with the previously reported results (Levy et al., 2015; Pennington et al., 2014), and not victims of some particularly unfortunate configuration.

5 Evaluation

5.1 The Google Test Set

3CosAdd is widely used for benchmarking word embeddings on the test known as the Google test set (Mikolov et al., 2013a). It contains 14 categories with 20-70 unique example pairs per category, which are combined in all possible ways to yield 8,869 semantic and 10,675 syntactic questions. The state-of-the-art on this test has over 65% average accuracy: 67.8% for DRRS (Garten et al., 2015), 70.64% for GCeWE (Zhou et al., 2015), and 75.6% for GloVe (Pennington et al., 2014).

The average accuracy for 3 models with 4 analogy detection methods is presented in table 2. We used logistic regression as a classification algorithm in the "exclude one" scheme, where the classifier is re-trained each time on all target class words excluding the one from the pair in question. Table 2 shows that LRCos clearly outperforms 3CosAdd and 3CosAvg, although for all methods accuracy varies between relations and models.

We compute both Mean_{all} (the number of correct answers divided by the total number of questions in the whole dataset) and Mean_{rel} (the average accuracy scores for all categories), and, for the latter, also SD (standard deviation) between categories. It is Mean_{all} that is typically reported (Mikolov et al., 2013a; Pennington et al., 2014), but table 2 suggests that Mean_{all} tends to be higher than Mean_{rel}. We attribute this to the fact that the Google test set is not balanced (20-70 unique pairs per category), and the more successful country:capital relations constitute the bulk of the semantic questions. Mean_{all} also can not represent the variation between categories, which in our experiments is between 17-28%.

Method	3CosAdd			PairDistance			3CosAvg			LRCos		
	Mean_{all}	Mean_{rel}	SD	Mean_{all}	Mean_{rel}	SD	Mean_{all}	Mean_{rel}	SD	Mean_{all}	Mean_{rel}	SD
SVD300	50.6%	45.1%	24%	22.7%	16.1%	17%	54.8%	51.2%	26%	68.2%	68.1%	23%
SVD1000	58.1%	49.4%	25%	23.6%	22.3%	17%	59.7%	54.0%	27%	74.6%	72.6%	21%
GloVe	**79.6%**	67.8%	26%	**33.5%**	**26.9%**	22%	79.1%	74.2%	28%	73.7%	70.9%	24%
Skip-Gram	75.1%	**66.6%**	23%	28.6%	24.2%	21.6%	**80.3%**	**78.3%**	17%	**79.8%**	**78.0%**	17%

Table 2: Average accuracy in the total dataset (Mean_{all}), between 14 categories (Mean_{rel}), and the standard deviation (SD) between categories in the Google test set.

Table 2 highlights the interaction between the method of discovering analogies and the word embedding itself. The results of GloVe and Skip-Gram improve with LRCos as compared to 3CosAdd, but the simple average 3CosAvg works even slightly better for them. However, SVD gets an over 15% boost

from LRCos, but not from 3CosAvg. This suggests that (a) the information not detected by 3CosAdd was actually contained in the SVD model, and (b) evaluating different embeddings with the same method might not be as meaningful as is commonly believed. Perhaps a better question to ask is why these embeddings behave so differently, and what does it tell us about them and the methods used.

5.2 The Bigger Analogy Test Set

The above results suggest that performance on the Google test set varies significantly between categories, and it was shown that some relations are in principle not detected successfully with 3CosAdd (Köper et al., 2015). Gladkova et al. (2016) developed the BATS analogy test set that is balanced across 4 types of linguistic relations (grammatical inflections, word-formation, lexicographical and world-knowledge relations), with 50 word pairs per category and 10 categories of each type (overall 2000 unique word pairs)[3]. This test presents a bigger challenge: the performance of GloVe is reported to drop from 80.4% on the Google test set to 28.5% on BATS due to difficulties with derivational and lexicographic categories.

Table 3 and figure 1 show that LRCos follows this overall trend, achieving only 47.7% average accuracy on BATS with Skip-Gram, the best-performing embedding. But it still outperforms the others: the best average for 3CosAdd is 28.1% (Glove), 7.5% for PairDistance (GloVe), 34.4% for 3CosAvg (GloVe). LRCosF is only slightly behind (47.2% for LRCosF on Skip-Gram).

Compared to 3CosAdd LRCos achieves up to 25% boost on encyclopedic relations (SVD model with 1000 dimensions), up to 8% boost on lexicographic relations (SVD), and, most significantly, up to 34% boost on the difficult derivational relations (for Skip-Gram). For inflectional morphology LRCosF yielded even better results (up to 28% for SVD).

Method	Encyclopedia			Lexicography			Inflectional Morphology			Derivational Morphology		
	SVD	GloVe	Skip-Gram	SVD	GloVe	Skip-Gram	SVD	GloVe	Skip-Gram	SVD	GloVe	Skip-Gram
PairDistance	11.8%	13.6%	12.4%	1.1%	1.0%	0.8%	12.8%	14.5%	14.9%	1.9%	0.8%	0.8%
3CosAdd	18.5%	31.5%	26.5%	10.1%	10.9%	9.1%	44.0%	59.9%	61.0%	9.8%	10.2%	11.2%
3CosAvg	30.0%	**44.8%**	34.6%	12.2%	13.0%	9.6%	51.2%	68.8%	69.8	13.0%	11.2%	15.2%
LRCos	39.3%	40.6%	**43.6%**	**18.0%**	16.8%	**15.4%**	65.2%	74.6%	87.2%	**30.4%**	17.0%	**45.6%**
LRCosF	**43.7%**	40.8%	42.6%	16.4%	**17.6%**	14.4%	**72.2%**	**75.0%**	**87.4%**	30.0%	**17.1%**	44.2%

Table 3: Average accuracy per relation type in BATS per method for SVD1000, GloVe and w2v models.

Figure 1 demonstrates variation in performance of 3CosAdd, 3CosAvg and LRCos on GloVe and SVD models by individual categories of BATS. LRCos almost always performs the best, but the pattern of results for GloVe and SVD is a little different. SVD did worse on inflectional morphology on SVD than on GloVe, so it benefitted more from LRCos - but it is interesting that (a) the benefit for the overall better-performing GloVe was overall smaller, and (b) LRCos almost never improves the results for categories where 3CosAdd already achieved near 80% accuracy. This suggests that there might be a certain limit on how accurate we can get on the test, at least for a given corpus.

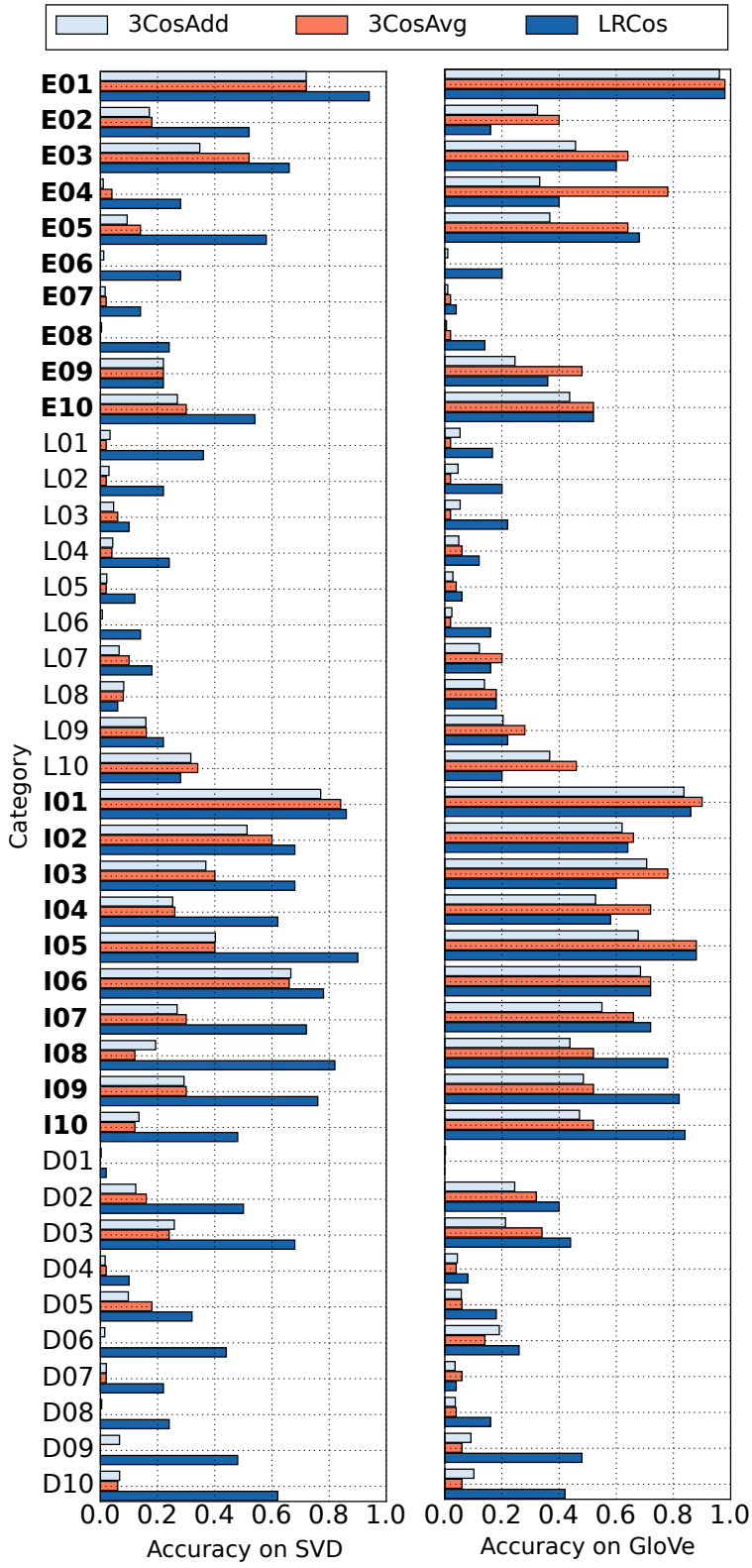
Table 3 shows that different methods for discovering analogies do not perform uniformly across the whole set or different embeddings. LRCos and LRCosF are "preferred" by different types of relations (although the gap between them is not so large), and in one case the baseline actually performs better.

One of the possible explanations for why LRCos yields significant improvement for derivational morphology, but not for lexicographic relations, is that LRCos relies on the notion of "target word class". In case of suffixes and prefixes such a target class is relatively coherent ("all words with the suffix *-ness*"), but for, e.g., synonyms, BATS includes different parts of speech (e.g. *scream:cry*, *car:automobile*, *loyal:faithful*). In this case there is no clear target class, and LRCos should actually be at a disadvantage compared to 3CosAdd (although it still improves results for some of the more coherent categories).

5.3 Russian Morphological Categories

As an additional task we compiled a small set consisting of 6 Russian noun forms: nominative case paired with instrumental, dative and prepositional cases in singular and plural form, such as *yaponets : yapontsem* ("a Japanese": "by a Japanese"). As in BATS, each category contains 50 unique word pairs.

[3]BATS dataset can be downloaded from `http://vsm.blackbird.pw/bats`



Encyclopedic relations

E01: geography: capitals (*Athens:Greece*)
E02: geography: languages (Peru:*Spanish*)
E03: geography: UK counties (*York:Yorkshire*)
E04: people: nationality (*Lincoln:American*)
E05: people: occupation (*Lincoln:president*)
E06: animals: the young (*cat:kitten*)
E07: animals: sounds (*dog:bark*)
E08: animals: shelter (*fox:den*)
E09: thing:color (*blood:red*)
E10: male:female (*actor:actress*)

Lexicographic relations

L01: hypernyms: animals (*turtle:reptile*)
L02: hypernyms: miscellaneous (peach:*fruit*)
L03: hyponyms: miscellaneous (*color:white*)
L04: meronyms: substance (*sea:water*)
L05: meronyms: member (*player:team*)
L06: meronyms: part-whole (*car:engine*)
L07: synonyms: intensity (*cry:scream*)
L08: synonyms: exact (*sofa:couch*)
L09: antonyms: gradable (*clean:dirty*)
L10: antonyms: opposites (*up:down*)

Inflectional Morphology

I01: noun sg:pl (regular) (*student:students*)
I02: noun sg:pl (irregular) (*wife:wives*)
I03: adjective: comparative (*strong:stronger*)
I04: adjective: superlative (*strong:strongest*)
I05: infinitive: 3Ps.Sg (*follow:follows*)
I06: infinitive: participle (*follow:following*)
I07: infinitive: past (*follow:followed*)
I08: participle: 3Ps.Sg (*following:follows*)
I09: participle: past (*following:followed*)
I10: 3Ps.Sg : past (*follows:followed*)

Derivational Morphology

D01: noun+ness (home:*homeness*)
D02: un+adjective (*able:unable*)
D03: adjective+ly (*usual:usually*)
D04: over+adjective (*used:overused*)
D05: adjective+ness (*mad:madness*)
D06: re+verb (*create:recreate*)
D07: verb+able (*edit:editable*)
D08: verb+er (*bake:baker*)
D09: verb+tion (*continue:continuation*)
D10: verb+ment (*argue:argument*)

Figure 1: Performance of 3CosAdd, 3CosAvg and LRCos methods on BATS categories.

The overall accuracy on the SVD embedding with 1000 dimensions is **18.0%** with 3CosAdd method, **19.2%** with 3CosAvg and **43.1%** with LRCos. For GloVe the results are **28.1%**, **34.4%** and **39.4%**, respectively.

While this is not a comprehensive test like BATS, it is sufficient to see if the different methods of discovering analogical relations are equally successful on morphologically complex and simple languages. The problem with the former is that there are many more word forms per lemma (e.g., the English text of *Pride and Prejudice* contains 7266 tokens for 6576 lemmas, and its Russian translation – 17903 tokens for 7715 lemmas, i.e. almost three times more). For word-level word embeddings this means that there are more vectors from the same volume of text, that they are built with less information, and that there are more vectors that are very similar (which complicates the task for a method relying on cosine similarity).

For this test we built an SVD-based model from Araneum Russicum Maximum (Benko and Zakharov, 2016) - a web-corpus containing 13.4B tokens. The parameters are as follows: 1000 dimensions, window size 3, with PMI-weighted co-occurrence matrix and Σ raised to the power $a = 1$.

Fig. 2 shows that morphological complexity does increase difficulty for analogical reasoning with word embeddings. In English 3CosAdd scores over 50% on many morphological categories, but in Russian cases its performance is in the 20% range. LRCos performs significantly better, although not always on the par with English. Further research is needed to tell why, e.g., Russian prepositional case appears to be more difficult than instrumental.

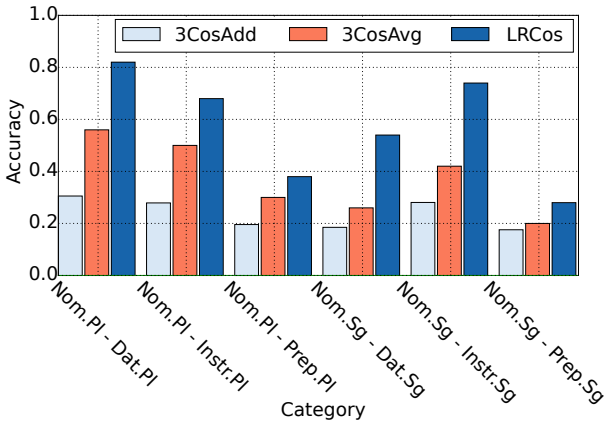


Figure 2: Accuracy of LRCos, 3CosAdd and 3CosAvg on Russian noun case forms.

6 Exploring LRCos

6.1 Effect of Training Set Size

Any method relying on supervised learning is only useful when there is sufficient data for learning. To use a method such as LRCos for practical analogical reasoning tasks we need to know how much data we would have to provide for each relation.

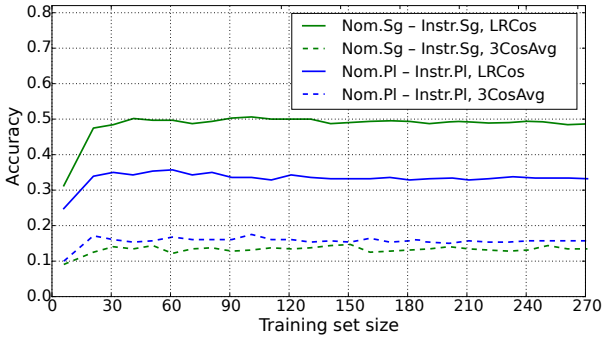


Figure 3: Effect of training set size.

We performed such an evaluation with our Russian morphology data, creating sets that contained up to a thousand word pairs. To estimate the optimal number of training samples we repeated the experiment multiple times, each time randomly selecting a subset of word pairs and observing how its size affects performance.

Two sample categories are shown in Figure 3 (LRCos and 3CosAvg methods). Our experiments suggest that accuracy for Russian morphological categories for both methods saturates at 50 pairs on average. While more tests are needed to determine if this number may be different for other types of relations or for other languages, such a range is feasible for LRCos to be used in practical tasks such as morphological parsing.

6.2 Effect of Vector Size

Our experiments suggest that although higher dimensionality implies more information about words being captured, it does not necessarily leads to better accuracy with the 3CosAdd method (a similar effect was observed by Cai et al. (2015) for similarity task). Some categories benefit slightly from higher dimensions, but for many there is no improvement, or there is even a slight degradation, while for

3CosAdd performance continues to rise. Data for four example categories are shown in Figure 4. One possible explanation for this phenomenon is that once the dimensions corresponding to the core aspects of a particular analogical relation are included in the vectors, adding more dimensions increases noise.

LRCos is not completely free from this negative effect, but it suffers less as the effect is mitigated by the fact that regression can assign near-zero weights to the dimensions which are not responsible for the target analogical relation. Thus algorithm performance continues to grow with larger vector sizes.

This result suggests that there is potential for LRCos to achieve even better results with combined models (Garten et al., 2015; Fried and Duh, 2014). For example, different window sizes are believed to be more beneficial for different tasks: larger windows for topical relations, smaller windows for morphological relations, as shown in (Lebret and Collobert, 2015). This would prevent any one model from achieving top performance on all tasks. However, we can have, e.g., a model that combines window 10 and window 2, and the extra dimensions will not become noise for LRCos method.

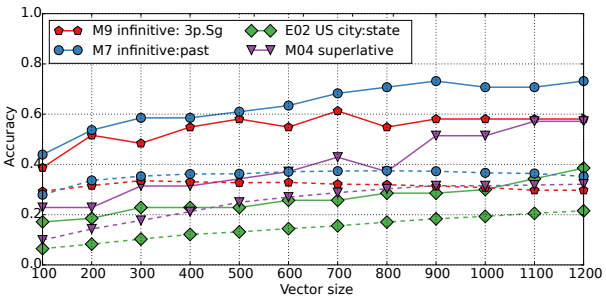


Figure 4: Effect of vector dimensionality on 3CosAdd (dashed lines) and LRCos (solid lines) methods

6.3 Effect of a Parameter

As described in section 4, we raise the elements of Σ matrix of factorization to the power of a to control the contribution of different singular vectors. If a is equal to 1, then each column in the transformed matrix is scaled proportionally to the variance in the original data it represents. Smaller values of a essentially boost the features which were less pronounced in the corpus in terms of co-occurrence patterns.

Figure 5 illustrates the impact of changing a for several example relations. Similarly to other model parameters, there is no value that provides the best results for all cases. The average accuracy is affected slightly, but certain relations may experience up to a two times decrease or improvement. This suggests that treating individual dimensions of embeddings independently could yield better results. While experimenting with the size of embeddings suggests that the LRCos method is better at selectively choosing useful dimensions for the embeddings, there is still a lot of room for improvement.

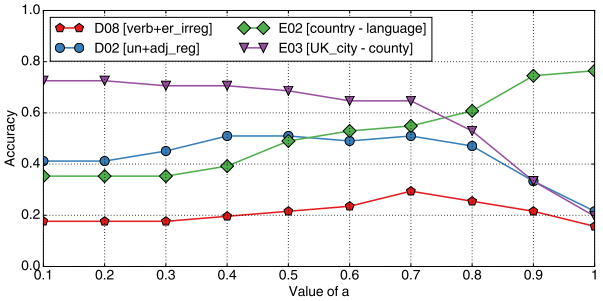


Figure 5: Effect of changing the value of α

7 Further Ways to Improve: Mistakes of 3CosAdd and LRCos

Since LRCos relies on both cosine similarity and degree to which the hypothetical answer belongs to the target class of words, it could be expected to yield a different pattern of mispredictions than 3CosAdd. To investigate the differences in the output of the two methods we manually annotated the incorrect answers by 3CosAdd and LRCos on 4 BATS categories: E05 (*Lincoln:president*), L05 (*parishioner:parish*), M05 (*follow:follows*) and WF05 (*create:recreate*). The evaluation was done with the SVD model at 1000 dimensions, window size 3. The results of this evaluation are summarized in table 4.

First of all, for both methods the ratio of "random" answers is insignificant; most of the wrong answers are morphologically, derivationally, collocationally, or semantically related to one or more of the source words – as can be expected for methods relying on cosine similarity. The problem, traditionally, is distinguishing between different types of relations.

When proposing the 3CosAdd method, Mikolov et al. (2013b) exclude the three source words from

Type of answer	Example	E05 name:occupation		L05 member: group		I05 infinitive: 3p.Sg		D05 re+verb	
		3CosAdd	LRCos	3CosAdd	LRCos	3CosAdd	LRCos	3CosAdd	LRCos
Correct answer	*hear: hears :: seem: seems*	10.62	52.00	0.97	4.08	35.1	78.00	26.24	48.00
Acceptable answer	*plato:philosopher :: hegel:?theorist*	1.42	6.00	1.75	6.12	-	-	-	-
Morphological relation	*define:redefine :: imagine: *imagining*	3.84	-	44.28	56.00	43.39	-	23.19	-
Misspelling of a source word	*confirm:reconfirm :: acquire:*aquire*	-	-	-	2.04	0.08	-	2.20	2.00
Derivational relation	*hear: hears :: seem: *seemingly*	0.94	-	1.25	-	0.82	-	1.55	2.00
Lexicographical relation	*include: includes:: appear: *seems*	61.6	22.00	27.69	4.08	4.53	4.00	14.04	8.00
Frame-semantic relation	*sit: resit :: learn: *coursework*	15.03	14.00	20.13	24.49	9.63	10.00	13.11	36.00
Collocate of a source word	*protect: protects:: learn: *basics*	-	-	1.99	4.08	3.35	2.00	0.49	-
Mistake due to polysemy	*parishioner: parish :: relative: *density*	1.67	-	8.49	4.08	0.94	-	-	-
Partially correct answer	*protect: protects :: maintain: *ensures*	-	-	0.97	10.20	6.82	18.00	13.71	28.00
Unrelated word	*send: resend :: engage: *categorise*	2.37	4.00	2.63	-	1.39	4.00	8.28	2.00

* Several relations may be applicable to each case, so the sum for each column does not necessarily add up to 100%.

Table 4: Types of mistakes for 3CosAdd and LRCos methods in different linguistic relations.

the set of possible answers, because otherwise one of them is too likely to turn up to be the closest to the hypothetical vector. But even if they are excluded, these source vectors can still "misdirect" the method. The fact that in L05, I05, and D05 the most frequent type of mistakes are the wrong morphological forms of a source word is consistent with the finding of Linzen (2016) that in the noun plural category of the Google test set the nearest neighbor of the c word provides the correct answer in 70% of cases. The plurals-as-nearest-neighbors were the pitfall for our L05 *member:group* category, where the analogy *student:class :: bird:?(flock)* would yield the answer *birds*. Likewise, with the verbs in I05 and D05 we were getting many participles with *-ing* ending: *arrange:rearrange::grow:*growing* (expected: *regrow*), *create:creates::accept:*accepting* (expected: *accepts*).

Consider now the E05 category that seems to break the pattern of morphologically-related nearest-neighbor: here the most mistakes are "lexicographic'. E05 category has analogies such as *aristotle:philosopher::bach:?composer*. The typical mistake is a co-hyponym of the a or (usually) c word, i.e. another composer in this case. This is also explained by the fact that for the names of famous people their nearest neighbors frequently happen to be co-hyponyms: in our SVD model the nearest neighbors of *Bach* are *Haydn* and *Schubert*, and in GloVe - *Handel* and *Beethoven*. This means that in all the categories the basic source of mistakes is the same indiscriminateness of cosine similarity.

Unfortunately, this means that word analogies fail to provide sufficient "context" to words: ideally, *king:queen :: man:woman* and *king:kings :: queen:queens* should profile sufficiently different aspects of the $king$ vector to avoid the nearest-neighbor trap. However, it does not seem to work this way. This is particularly clear in mistakes resulting from polysemy of one of the source words. For example, in L05 we had: *crow:murder::fish:*killing* (expected: *school*), *lion:pride::bird:*ambition* (expected: *flock*).

In E05, D05 and I05 LRCos significantly improves over 3CosAdd by reducing this nearest-neighbor kind of mistake, but it is telling that this improvement comes with the increase of partially-correct answers: the model comes up with the correct target feature in an incorrect word, e.g. ask $asks$ + $happen$ = $realizes$ (expected: *$happens$). Such mistakes suggest that the contribution of classifier and cosine similarity could differ for different words, although it is not clear how to determine them dynamically.

Another observation from our data is that for both methods the margin of error is very thin. For example, in the E05 category LRCos gives *Depp:screenwriter* a total score of 0.36, and this incorrect answer beats the correct answer *Depp:actor* that is ranked 0.35. Average accuracy would be much higher if we allowed the answers to be in the top five nearest neighbors, although this, of course, brings up the problem of where analogies could be used in practice, and what level of precision that would require.

8 Discussion: Embeddings vs Methods

LRCos offers a significant boost in accuracy for detecting analogical relations over the most widely-used 3CosAdd method, including derivational relations where the latter does not perform well. However, LRCos is by no means perfect, and there is room for further improvement, especially with respect to lexicographic relations. This includes algorithms aimed at searching for complex patterns of correspondences between vectors rather than simple similarity. A different (and potentially more fruitful) approach is to investigate whether the target relations are at all reflected in the distributional properties of words.

Aside from the practical result for non-lexicographic relations, this work also brings up a theoretical question. We have shown that different methods of detecting analogies provide different results on different embeddings, and this means that low performance of a word embedding with, e.g., 3CosAdd method, does not prove that the embedding does not capture certain linguistic relations - only that they are not detectable with this particular method. This brings into question the validity of analogy detection with 3CosAdd as a benchmark for word embeddings, as it is frequently used (Pennington et al., 2014; Garten et al., 2015; Cui et al., 2014).

It could be argued that embeddings could be judged "good" as in "easy to work with"; in this sense a "good" embedding is an embedding that yields correct answers with simple rather than complex extraction methods. However, what is good for practical applications is not necessarily the same as what is good for a benchmark. In this case with analogies, 3CosAdd is at disadvantage with embeddings that encode a lot of extra (but useful) information in dimensions that are irrelevant to a particular relation, and thus misleads 3CosAdd. On the other hand, it could be argued that machine-learning-based methods should not be used for benchmarking because they could learn to ignore noise too well.

9 Conclusion

We presented LRCos, a method of analogical reasoning that is based on supervised learning from a group of examples. LRCos significantly outperforms the popular 3CosAdd method (based on offset for individual word pairs) on both the Google and BATS test sets, although the gain varies between embeddings and relation types. Importantly, LRCos achieves high accuracy in two areas where 3CosAdd mostly failed: word-formation in English and grammar in Russian, a morphologically rich language. Unlike 3CosAdd, LRCos is less sensitive to idiosyncrasies of individual word pairs, and does not suffer from higher vector dimensionality.

We compared 5 analogical reasoning methods on 40 types of linguistic relations with two word embeddings: explicit SVD model and neural-net-based GloVe. Both models yielded overall similar patterns of performance with different methods, offering further evidence for conceptual similarity of explicit and implicit word embeddings.

This work also makes a theoretical contribution in demonstrating the interaction between word embeddings, types of analogies, and different types of search algorithms: with LRCos our SVD-based model approaches the state-of-the-art performance for GloVe and Skip-Gram. This suggests that the information about linguistic relations from the test set was actually encoded in the SVD-based embedding, possibly in a different way. In that case we need to decide whether failure to detect a relation with 3CosAdd method actually indicates inferiority of a word embedding, and whether a "good" embedding should encode different kinds of relations in the same way - as the Google test set in conjunction with 3CosAdd is still one of the most popular benchmarks.

Acknowledgements

This paper was partially supported by JST, CREST (Research Area: Advanced Core Technologies for Big Data Integration).

References

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

Marco Baroni, Georgiana Dinu, and Germn Kruszewski. 2014. Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 238–247.

Vladimr Benko and V.P. Zakharov. 2016. Very large Russian corpora: New opportunities and new challenges. In *Kompjuternaja Lingvistika I Intellektuanyje Technologii: Po Materialam Medunarodnoj konferencii "Dialog" (2016)*, volume 15(22), pages 79–93. Moskva: Rossijskij gosudarstvennyj gumanitarnyj universitet.

Vladimír Benko. 2014. Aranea: Yet another family of (comparable) web corpora. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, speech, and dialogue: 17th international conference, TSD 2014, Brno, Czech Republic, September 8-12, 2014. Proceedings*, LNCS 8655, pages 257–264. Springer.

Yuanyuan Cai, Wei Lu, Xiaoping Che, and Kailun Shi. 2015. Differential Evolutionary Algorithm Based on Multiple Vector Metrics for Semantic Similarity Assessment in Continuous Vector Space. In *Proceedings of DMS 2015*, pages 241–249. [doi:10.18293/DMS2015-001].

John Caron. 2001. Computational information retrieval. chapter Experiments with LSA Scoring: Optimal Rank and Basis, pages 157–169. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

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

Trevor Cohen, Dominic Widdows, and Thomas Rindflesch. 2015. Expansion-by-analogy: A vector symbolic approach to semantic search. In *Quantum Interaction*, pages 54–66. Springer.

Qing Cui, Bin Gao, Jiang Bian, Siyu Qiu, and Tie-Yan Liu. 2014. Learning effective word embedding using morphological word similarity. *arXiv preprint arXiv:1407.1687*.

Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2015. Python, performance, and natural language processing. In *Proceedings of the 5th Workshop on Python for High-Performance and Scientific Computing*, PyHPC '15, pages 1:1–1:10, New York, NY, USA. ACM.

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

Stefano Federici, Simonetta Montemagni, and Vito Pirrelli. 1997. Inferring semantic similarity from distributional evidence: an analogy-based approach to word sense disambiguation. In *Proceedings of the ACL/EACL Workshop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 90–97.

Daniel Fried and Kevin Duh. 2014. Incorporating both distributional and relational semantics in word representations. *arXiv preprint arXiv:1412.4369*.

Justin Garten, Kenji Sagae, Volkan Ustun, and Morteza Dehghani. 2015. Combining distributed vector representations for words. In *Proceedings of NAACL-HLT 2015*, pages 95–101.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of NAACL-HLT 2016*, pages 47–54. Association for Computational Linguistics.

Gene H. Golub and Charles F. Van Loan. 1996. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA.

Maximilian Köper, Christian Scheible, and Sabine Schulte im Walde. 2015. Multilingual reliability and semantic structure of continuous word spaces. In *Proceedings of the 11th International Conference on Computational Semantics 2015*, pages 40–45. Association for Computational Linguistics.

Siwei Lai, Kang Liu, Liheng Xu, and Jun Zhao. 2015. How to generate a good word embedding? *arXiv preprint arXiv:1507.05523*.

Jean-Franois Lavallée and Philippe Langlais. 2010. Unsupervised morphological analysis by formal analogy. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments*, pages 617–624. Springer.

Rmi Lebret and Ronan Collobert. 2015. Rehabilitation of count-based models for word vector representations. In *Computational Linguistics and Intelligent Text Processing*, pages 417–429. Springer.

Yves Lepage and Chooi-ling Goh. 2009. Towards automatic acquisition of linguistic features. In *Proceedings of the 17th Nordic Conference on Computational Linguistics (NODALIDA 2009), eds., Kristiina Jokinen and Eckard Bick*, pages 118–125.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. In *Transactions of the Association for Computational Linguistics*, volume 3, pages 211–225.

Tal Linzen. 2016. Issues in evaluating semantic spaces using word analogies. In *Proceedings of the 1st Workshop on Evaluating Vector Space Representations for NLP*, pages 13–18. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *Proceedings of International Conference on Learning Representations (ICLR)*.

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: global vectors for word representation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, volume 12, pages 1532–1543.

Peter D. Turney. 2008. A uniform approach to analogies, synonyms, antonyms, and associations. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 905–912.

Ekaterina Vylomova, Laura Rimmel, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1671–1682. Association for Computational Linguistics.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. Category enhanced word embedding. *arXiv preprint arXiv:1511.08629*.

Semantic Tagging with Deep Residual Networks

Johannes Bjerva
University of Groningen
The Netherlands
j.bjerva@rug.nl

Barbara Plank
University of Groningen
The Netherlands
b.plank@rug.nl

Johan Bos
University of Groningen
The Netherlands
johan.bos@rug.nl

Abstract

We propose a novel semantic tagging task, *sem-tagging*, tailored for the purpose of multilingual semantic parsing, and present the first tagger using deep residual networks (ResNets). Our tagger uses both word and character representations, and includes a novel residual bypass architecture. We evaluate the tagset both intrinsically on the new task of semantic tagging, as well as on Part-of-Speech (POS) tagging. Our system, consisting of a ResNet and an auxiliary loss function predicting our semantic tags, significantly outperforms prior results on English Universal Dependencies POS tagging (95.71% accuracy on UD v1.2 and 95.67% accuracy on UD v1.3).

1 Introduction

A key issue in computational semantics is the transferability of semantic information across languages. Many semantic parsing systems depend on sources of information such as POS tags (Pradhan et al., 2004; Copestake et al., 2005; Bos, 2008; Butler, 2010; Berant and Liang, 2014). However, these tags are often customised for the language at hand (Marcus et al., 1993) or massively abstracted, such as the Universal Dependencies tagset (Nivre et al., 2016). Furthermore, POS tags are syntactically oriented, and therefore often contain both irrelevant and insufficient information for semantic analysis and deeper semantic processing. This means that, although POS tags are highly useful for many downstream tasks, they are unsuitable both for semantic parsing in general, and for tasks such as recognising textual entailment.

We present a novel set of semantic labels tailored for the purpose of multilingual semantic parsing. This tagset (i) abstracts over POS and named entity types; (ii) fills gaps in semantic modelling by adding new categories (for instance for phenomena like negation, modality, and quantification); and (iii) generalises over specific languages (see Section 2). We introduce and motivate this new task in this paper, and refer to it as *semantic tagging*. Our experiments aim to answer the following two research questions:

1. Given an annotated corpus of semantic tags, it is straightforward to apply off-the-shelf sequence taggers. Can we significantly outperform these with recent neural network architectures?

2. Semantic tagging is essential for deep semantic parsing. Can we find evidence that semtags are effective also for other NLP tasks?

To address the first question, we will look at convolutional neural networks (CNNs) and recurrent neural networks (RNNs), which are both highly prominent approaches in the recent natural language processing (NLP) literature. A recent development is the emergence of deep residual networks (ResNets), a building block for CNNs. ResNets consist of several stacked residual units, which can be thought of as a collection of convolutional layers coupled with a 'shortcut' which aids the propagation of the signal in a neural network. This allows for the construction of much deeper networks, since keeping a 'clean' information path in the network facilitates optimisation (He et al., 2016). ResNets have recently shown state-of-the-art performance for image classification tasks (He et al., 2015; He et al., 2016), and have

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3531–3541, Osaka, Japan, December 11-17 2016.

also seen some recent use in NLP (Östling, 2016; Conneau et al., 2016; Bjerva, 2016; Wu et al., 2016). However, no previous work has attempted to apply ResNets to NLP tagging tasks.

To answer our second question, we carry out an extrinsic evaluation exercise. We investigate the effect of using semantic tags as an auxiliary loss for POS tagging. Since POS tags are useful for many NLP tasks, it follows that semantic tags must be useful if they can improve POS tagging.

2 Semantic Tagging

2.1 Background

We refer to *semantic tagging*, or *sem-tagging*, as the task of assigning semantic class categories to the smallest meaningful units in a sentence. In the context of this paper these units can be morphemes, words, punctuation, or multi-word expressions. The linguistic information traditionally obtained for deep processing is insufficient for fine-grained lexical semantic analysis. The widely used Penn Treebank (PTB) Part-of-Speech tagset (Marcus et al., 1993) does not make the necessary semantic distinctions, in addition to containing redundant information for semantic processing. Let us consider a couple of examples.

There are significant differences in meaning between the determiners *every* (universal quantification), *no* (negation), and *some* (existential quantification), but they all receive the `DT` (determiner) POS label in PTB. Since determiners form a closed class, one could enumerate all word forms for each class. Indeed some recent implementations of semantic parsing follow this strategy (Bos, 2008; Butler, 2010). This might work for a single language, but it falls short when considering a multilingual setting. Furthermore, determiners like *any* can have several interpretations and need to be disambiguated in context.

Semantic tagging does not only apply to determiners, but reaches all parts of speech. Other examples where semantic classes disambiguate are reflexive versus emphasising pronouns (both POS-tagged as `PRP`, personal pronoun); the comma, that could be a conjunction, disjunction, or apposition; intersective vs. subsective and privative adjectives (all POS-tagged as `JJ`, adjective); proximal vs. medial and distal demonstratives (see Example 1); subordinate vs. coordinate discourse relations; role nouns vs. entity nouns. The set of semantic tags that we use in this paper is established in a data-driven manner, considering four languages in a parallel corpus (English, German, Dutch and Italian). This first inventory of classes comprises 13 coarse-grained tags and 75 fine-grained tags (see Table 1). As can be seen from this table and the examples given below, the tagset also includes named entity classes (see also Example 2).

(1) *These cats live in that house .*
`PRX CON ENS REL DST CON NIL`

(2) *Ukraine 's glory has not yet perished , neither her freedom .*
`GPE HAS CON ENT NOT IST EXT NIL NOT HAS CON NIL`

In Example 1, both *these* and *that* would be tagged as `DT`. However, with our semantic tagset, they are disambiguated as `PRX` (proximal) and `DST` (distal). In Example 2, *Ukraine* is tagged as `GPE` rather than `NNP`.

2.2 Annotated data

We use two semtag datasets. The Groningen Meaning Bank (GMB) corpus of English texts (1.4 million words) containing silver standard semantic tags obtained by running a simple rule-based semantic tagger (Bos et al., Forthcoming). This tagger uses POS and named entity tags available in the GMB (automatically obtained with the C&C tools (Curran et al., 2007) and then manually corrected), as well as a set of manually crafted rules to output semantic tags. Some tags related to specific phenomena were hand-corrected in a second stage.

Our second dataset is smaller but equipped with gold standard semantic tags and used for testing (PMB, the Parallel Meaning Bank). It comprises a selection of 400 sentences of the English part of a parallel corpus. It has no overlap with the GMB corpus. For this dataset, we used the Elephant tokeniser, which performs word, multi-word and sentence segmentation (Evang et al., 2013). We then used the

ANA PRO pronoun	COM EQA equative	EVE EXS untensed simple
DEF definite	MOR comparative pos.	ENS present simple
HAS possessive	LES comparative neg.	EPS past simple
REF reflexive	TOP pos. superlative	EFS future simple
EMP emphasizing	BOT neg. superlative	EXG untensed prog.
ACT GRE greeting	ORD ordinal	ENG present prog.
ITJ interjection	DEM PRX proximal	EPG past prog.
HES hesitation	MED medial	EFG future prog.
QUE interrogative	DST distal	EXT untensed perfect
ATT QUA quantity	DIS SUB subordinate	ENT present perfect
UOM measurement	COO coordinate	EPT past perfect
IST intersective	APP appositional	EFT future perfect
REL relation	MOD NOT negation	ETG perfect prog.
RLI rel. inv. scope	NEC necessity	ETV perfect passive
SST subsective	POS possibility	EXV passive
PRI privative	ENT CON concept	TNS NOW present tense
INT intensifier	ROL role	PST past tense
SCO score	NAM GPE geo-political ent.	FUT future tense
LOG ALT alternative	PER person	TIM DOM day of month
EXC exclusive	LOC location	YOC year of century
NIL empty	ORG organisation	DOW day of week
DIS disjunct./exist.	ART artifact	MOY month of year
IMP implication	NAT natural obj./phen.	DEC decade
AND conjunct./univ.	HAP happening	CLO clocktime
BUT contrast	URL url	

Table 1: Semantic tags used in this paper.

simple rule-based semantic tagger described above to get an initial set of tags. These tags were then corrected by a human annotator (one of the authors of this paper).

For the extrinsic evaluation, we use the POS annotation in the English portion of the Universal Dependencies dataset, version 1.2 and 1.3 (Nivre et al., 2016). An overview of the data used is shown in Table 2.

CORPUS	TRAIN (SENTS/TOKS)	DEV (SENTS/TOKS)	TEST (SENTS/TOKS)	N TAGS
ST Silver (GMB)	42,599 / 930,201	6,084 / 131,337	12,168 / 263,516	66
ST Gold (PMB)	n/a	n/a	356 / 1,718	66
UD v1.2 / v1.3	12,543 / 204,586	2,002 / 25,148	2,077 / 25,096	17

Table 2: Overview of the semantic tagging data (ST) and universal dependencies (UD) data.

3 Method

Our tagger is a hierarchical deep neural network consisting of a bidirectional Gated Recurrent Unit (GRU) network at the upper level, and a Convolutional Neural Network (CNN) and/or Deep Residual Network (ResNet) at the lower level, including an optional novel residual bypass function (cf. Figure 1).

3.1 Gated Recurrent Unit networks

GRUs (Cho et al., 2014) are a recently introduced variant of RNNs, and are designed to prevent vanishing gradients, thus being able to cope with longer input sequences than vanilla RNNs. GRUs are similar to the more commonly-used Long Short-Term Memory networks (LSTMs), both in purpose and implementation (Chung et al., 2014). A bi-directional GRU is a GRU which makes both forward and backward passes over sequences, and can therefore use both preceding and succeeding contexts to predict a tag (Graves and Schmidhuber, 2005; Goldberg, 2015). Bi-directional GRUs and LSTMs have been shown to yield high performance on several NLP tasks, such as POS tagging, named entity tagging, and chunking (Wang et al., 2015; Yang et al., 2016; Plank et al., 2016). We build on previous approaches by combining bi-GRUs with character representations from a basic CNN and ResNets.

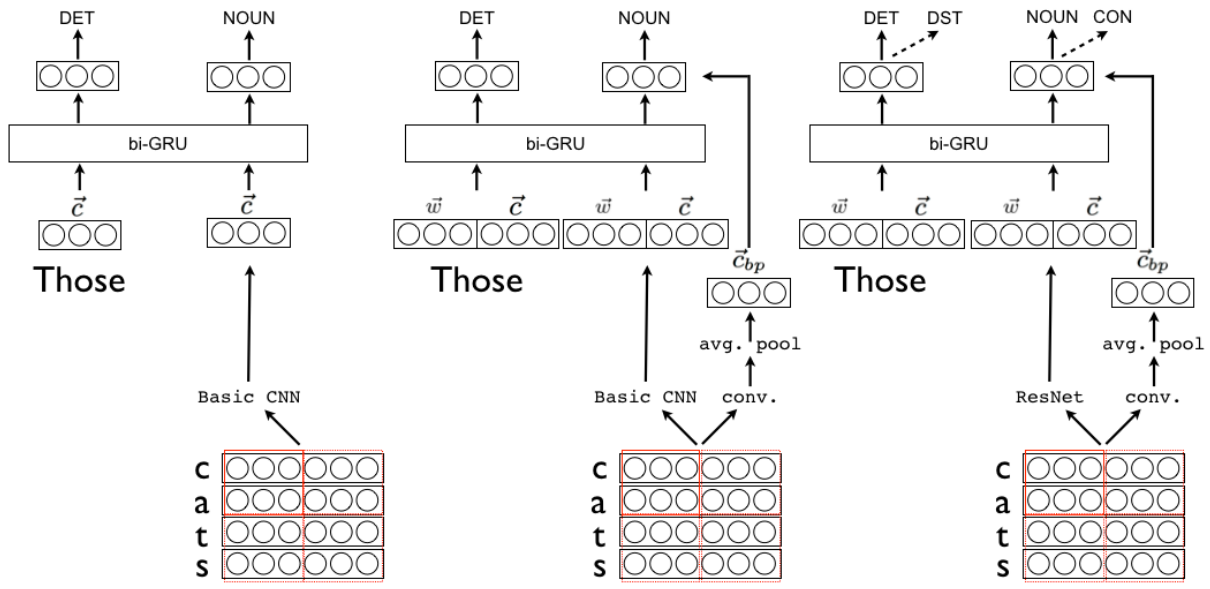


Figure 1: Model architecture. Left: Architecture with basic CNN char representations ($\vec{c}$), Middle: basic CNN with char and word representations and bypass ($\vec{c}_{bp} \wedge \vec{w}$), Right: ResNet with auxiliary loss and residual bypass ($+\text{AUX}_{bp}$).

3.2 Deep Residual Networks

Deep Residual Networks (ResNets) are built up by stacking residual units. A residual unit can be expressed as:

$$\begin{aligned} y_l &= h(x_l) + \mathcal{F}(x_l, \mathcal{W}_l), \\ x_{l+1} &= f(y_l), \end{aligned} \tag{3}$$

where x_l and x_{l+1} are the input and output of the l-th layer, $\mathcal{W}_l$ is the weights for the l-th layer, and $\mathcal{F}$ is a residual function (He et al., 2016), e.g., the identity function (He et al., 2015), which we also use in our experiments. ResNets can be intuitively understood by thinking of residual functions as paths through which information can propagate easily. This means that, in every layer, a ResNet learns more complex feature combinations, which it combines with the shallower representation from the previous layer. This architecture allows for the construction of much deeper networks. ResNets have recently been found to yield impressive performance in image recognition tasks, with networks as deep as 1001 layers (He et al., 2015; He et al., 2016), and are thus an interesting and effective alternative to simply stacking layers. In this paper we use the *assymetric* variant of ResNets as described in Equation 9 in He et al. (2016):

$$x_{l+1} = x_l + \mathcal{F}(\hat{f}(x_l), \mathcal{W}_l). \tag{4}$$

ResNets have been very recently applied in NLP to morphological reinflection (Östling, 2016), language identification (Bjerva, 2016), sentiment analysis and text categorisation (Conneau et al., 2016), as well as machine translation (Wu et al., 2016). Our work is the first to apply ResNets to NLP sequence tagging tasks. We further contribute to the literature on ResNets by introducing a residual bypass function. The intuition is to combine both deep and shallow processing, which opens a path of easy signal propagation between lower and higher layers in the network.

3.3 Modelling character information and residual bypass

Using sub-token representations instead of, or in combination with, word-level representations has recently obtained a lot of attention due to their effectiveness (Sutskever et al., 2011; Chrupała, 2013;

Zhang et al., 2015; Chung et al., 2016; Gillick et al., 2015). The use of sub-token representations can be approached in several ways. Plank et al. (2016) and Yang et al. (2016) use a hierarchical bi-directional RNN, first passing over characters in order to create word-level representations. Gillick et al. (2015) similarly apply an LSTM-based model using byte-level information directly. Dos Santos and Zadrozny (2014) construct character-based word-level representations by running a convolutional network over the character representations of each word. All of these approaches have in common that the character-based representation is passed through the entire remainder of the network. Our work is the first to combine the use of character-level representations with both deep processing (i.e., passing this representation through the network) and shallow processing (i.e., bypassing the network in our residual bypass function). We achieve this by applying our novel residual bypass function to our character representations, inspired by the success of ResNets. In particular, we first apply the bypass to a CNN-based model achieving large gains over a plain CNN, and later evaluate its effectiveness in a ResNet.

A core intuition behind CNNs is the processing of an input signal in a hierarchical manner (LeCun et al., 1998; Goodfellow et al., 2016). Taking, e.g., a 3-dimensional image ($width \times height \times depth$), the approach is typically to reduce spatial dimensions of the image while increasing depth. This hierarchical processing allows a CNN to learn high-level features of an input, essential to image recognition tasks. A drawback of this method, however, is that lower-level features are potentially lost in the abstraction to higher-level features. This issue is partially countered by ResNets, as information is allowed to flow more easily between residual blocks. However, this approach does not allow for simple and direct use of information in the network input in final layers. To alleviate this issue, we present a residual bypass function, which can be seen as a global residual function (depicted in Figure 1). This function allows both lower-level and higher-level features to be taken directly into account in the final layers of the network. The intuition behind using such a global residual function in NLP is that character information primarily ought to be of importance for the prediction of the current word. Hence, allowing these representations to bypass our bi-GRU might be beneficial. This residual bypass function is not dependent on the usage of ResNets, and can be combined with other NN architectures as in our experiments. We define the penultimate layer of a network with n layers, using a residual bypass, as follows:

$$y_{n-1} = h(x_{n-1}) + \mathcal{F}(x_i, \mathcal{W}_i), \tag{5}$$

where x_i and $\mathcal{W}_i$ are the input and weights of the i_{th} layer, $\mathcal{F}$ is a residual function (in our case the identity function), and $h(x_{n-1})$ is the output of the penultimate layer. In our experiments, we apply a residual bypass function to our convolutional character representations.

3.4 System description

The core of our architecture consists of a bi-GRU taking an input based on words and/or characters, with an optional residual bypass as defined in subsection 3.3. We experiment with a basic CNN, ResNets and our novel residual bypass function. We also implemented a variant of the *Inception* model (Szegedy et al., 2015), but found this to be outperformed by ResNets. Our system is implemented in Keras using the Tensorflow backend (Chollet, 2015; Abadi et al., 2016), and the code is available at `https://github.com/bjerva/semantic-tagging`.

We represent each sentence using both a character-based representation (S_c) and a word-based representation (S_w). The character-based representation is a 3-dimensional matrix $S_c \in \mathbb{R}^{s \times w \times d_c}$, where s is the zero-padded sentence length, w is the zero-padded word length, and d_c is the dimensionality of the character embeddings. The word-based representation is a 2-dimensional matrix $S_w \in \mathbb{R}^{s \times d_w}$, where s is the zero-padded sentence length and d_w is the dimensionality of the word embeddings. We use the English Polyglot embeddings (Al-Rfou et al., 2013) in order to initialise the word embedding layer, but also experiment with randomly initialised word embeddings.

Word embeddings are passed directly into a two-layer bi-GRU (Chung et al., 2014). We also experimented using a bi-LSTM. However, we found GRUs to yield comparatively better validation data performance on semtags. We also observe better validation data performance when running two consecutive forward and backward passes before concatenating the GRU layers, rather than concatenating after each forward/backward pass as is commonplace in NLP literature.

We use CNNs for character-level modelling. Our basic CNN is inspired by dos Santos and Zadrozny (2014), who use character-representations to produce local features around each character of a word, and combine these with a maximum pooling operation in order to create fixed-size character-level word embeddings. The convolutions used in this manner cover a few neighbouring letters at a time, as well as the entire character vector dimension (d_c). In contrast to dos Santos and Zadrozny (2014), we treat a word analogously to an image. That is to say, we see a word of n characters embedded in a space with dimensionality d_c as an image of dimensionality $n \times d_c$. This view gives us additional freedom in terms of sizes of convolutional patches used, which offers more computational flexibility than using only, e.g., $4 \times d_c$ convolutions. This view is applied to all CNN variations explored in this work.

A neural network is trained with respect to some loss function, such as the cross-entropy between the predicted tag probability distribution and the gold probability distribution. Recent work has shown that the addition of an auxiliary loss function can be beneficial to several tasks. Cheng et al. (2015) use a language modelling task as an auxiliary loss, as they attempt to predict the next token while performing named entity recognition. Plank et al. (2016) use the log frequency of the current token as an auxiliary loss function, and find this to improve POS tagging accuracy. Since our semantic tagging task is based on predicting fine semtags, which can be mapped to coarse semtags, we add the prediction of these coarse semtags as an auxiliary loss for the sem-tagging experiments. Similarly, we also experiment with POS tagging, where we use the fine semtags as an auxiliary information.

3.4.1 Hyperparameters

All hyperparameters are tuned with respect to loss on the semtag validation set. We use rectified linear units (ReLUs) for all activation functions (Nair and Hinton, 2010), and apply dropout with $p = 0.1$ to both input weights and recurrent weights in the bi-GRU (Srivastava et al., 2014). In the CNNs, we apply batch normalisation (Ioffe and Szegedy, 2015) followed by dropout with $p = 0.5$ after each layer. In our basic CNN, we apply a 4×8 convolution, followed by 2×2 maximum pooling, followed by 4×4 convolution and another 2×2 maximum pooling. Our ResNet has the same setup, with the addition of a residual connection. We also experimented with using average pooling instead of maximum pooling, but this yielded lower validation data performance on the semantic tagging task. We set both d_c and d_w to 64. All GRU layers have 100 hidden units. All experiments were run with early stopping monitoring validation set loss, using a maximum of 50 epochs. We use a batch size of 500. Optimisation is done using the ADAM algorithm (Kingma and Ba, 2014), with the categorical cross-entropy loss function as training objective. The main and auxiliary loss functions have a weighting parameter, λ. In our experiments, we weight the auxiliary loss with $\lambda = 0.1$, as set on the semtag auxiliary task.

Multi-word expressions (MWEs) are especially prominent in the semtag data, where they are annotated as single tokens. Pre-trained word embeddings are unlikely to include entries such as 'International Organization for Migration', so we apply a simple heuristic in order to avoid treating most MWEs as unknown words. In particular, the representation of a MWE is set to the sum of the individual embeddings of each constituent word.

4 Evaluation

We evaluate our tagger on two tasks: semantic tagging and POS tagging. Note that the tagger is developed solely on the semantic tagging task, using the GMB silver training and validation data. Hence, no further fine-tuning of hyperparameters for the POS tagging task is performed. We calculate significance using bootstrap resampling (Efron and Tibshirani, 1994). We manipulate the following independent variables in our experiments:

1. character and word representations ($\vec{w}, \vec{c}$);
2. residual bypass for character representations ($\vec{c}_{bp}$);
3. convolutional representations (Basic CNN and ResNets);
4. auxiliary loss (using coarse semtags on ST and fine semtags on UD).

We compare our results to four baselines:

1. the most frequent baseline per word (MFC), where we assign the most frequent tag for a word in the training data to that word in the test data, and unseen words get the global majority tag;
2. the trigram statistic based TNT tagger offers a slightly tougher baseline (Brants, 2000);
3. the BI-LSTM baseline, running the off-the-shelf state-of-the-art POS tagger for the UD dataset (Plank et al., 2016) (using default parameters with pre-trained Polyglot embeddings);
4. we use a baseline consisting of running our own system with only a BI-GRU using word representations ($\vec{w}$), with pre-trained Polyglot embeddings.

4.1 Experiments on semantic tagging

We evaluate our system on two semantic tagging (ST) datasets: our silver semtag dataset and our gold semtag dataset. For the +AUX condition we use coarse semtags as an auxiliary loss. Results from these experiments are shown in Table 3.

	BASELINES				BASIC CNN				RESNET					
	MFC	TNT	BI-LSTM	BI-GRU	$\vec{c}$	$\vec{c}_{bp}$	$\vec{c}_{bp} \wedge \vec{w}$	+AUX$_{bp}$	$\vec{c}$	$\vec{c} \wedge \vec{w}$	+AUX	$\vec{c}_{bp}$	$\vec{c}_{bp} \wedge \vec{w}$	+AUX$_{bp}$
ST Silver	84.64	92.09	94.98	94.26	91.39	90.18	94.63	94.53	94.39	95.14	94.23	94.23	**95.15**	94.58
ST Gold	77.39	80.73	82.96	80.26	69.21	65.77	76.83	80.73	76.89	**83.64**	74.84	75.84	82.18	73.73

Table 3: Experiment results on semtag (ST) test sets (% accuracy). MFC indicates the per-word most frequent class baseline, TNT indicates the TNT tagger, and BI-LSTM indicates the system by Plank et al. (2016). BI-GRU indicates the $\vec{w}$ only baseline. $\vec{w}$ indicates usage of word representations, $\vec{c}$ indicates usage of character representations, and $\vec{c}_{bp}$ indicates usage of residual bypass of character representations. The +AUX column indicates the usage of an auxiliary loss.

4.2 Experiments on Part-of-Speech tagging

We evaluate our system on v1.2 and v1.3 of the English part of the Universal Dependencies (UD) data. We report results for POS tagging alone, comparing to commonly used baselines and prior work using LSTMs, as well as using the fine-grained semantic tags as auxiliary information. For the +AUX condition, we train a single joint model using a multi-task objective, with POS and ST as our two tasks. This model is trained on the concatenation of the ST silver data with the UD data, updating the loss of the respective task of an instance in each iteration. Hence, the weights leading to the UD softmax layer are not updated on the ST silver portion of the data, and vice-versa for the ST softmax layer on the UD portion of the data. Results from these experiments are shown in Table 4.

	BASELINES				BASIC CNN				RESNET					
	MFC	TNT	BI-LSTM	BI-GRU	$\vec{c}$	$\vec{c}_{bp}$	$\vec{c}_{bp} \wedge \vec{w}$	+AUX$_{bp}$	$\vec{c}$	$\vec{c} \wedge \vec{w}$	+AUX	$\vec{c}_{bp}$	$\vec{c}_{bp} \wedge \vec{w}$	+AUX$_{bp}$
UD v1.2	85.06	92.66	95.17	94.39	77.63	83.53	94.68	95.19	92.65	94.92	**95.71**	92.45	94.73	95.51
UD v1.3	85.07	92.69	95.04	94.32	77.51	82.89	94.89	95.34	92.63	94.88	**95.67**	92.86	94.69	95.57

Table 4: Experiment results on Universal Dependencies (UD) test sets (% accuracy). Adding semtags as auxiliary tags results in the best results obtained so far on English UD datasets.

5 Discussion

5.1 Performance on semantic tagging

The overall best system is the ResNet combining both word and character representations $\vec{c} \wedge \vec{w}$. It outperforms all baselines, including the recently proposed RNN-based bi-LSTM. On the ST silver data, a significant difference ($p < 0.01$) is found when comparing our best system to the strongest baseline

(BI-LSTM). On the ST gold data, we observe significant differences at the alpha values recommended by Søgaard et al. (2014), with $p < 0.0025$. The residual bypass effectively helps improve the performance of the basic CNN. However, the tagging accuracy of the CNN falls below baselines. In addition, the large gap between gold and silver data for the CNN shows that the CNN model is more prone to overfitting, thus favouring the use of the ResNet. Adding the coarse-grained semtags as auxiliary task only helps for the weaker CNN model. The ResNet does not benefit from this additional information, which is already captured in the fine-grained labels.

It is especially noteworthy that the ResNet character-only system performs remarkably well, as it outperforms the BI-GRU and TNT baselines, and is considerably better than the basic CNN. Since performance increases further when adding in $\vec{w}$, it is clear that the character and word representations are complimentary in nature. The high results for characters only are particularly promising for multilingual language processing, as this allows for much more compact models (see, e.g., Gillick et al. (2015)), which is a direction we want to explore next.

5.2 Performance on Part-of-Speech tagging

Our system was tuned solely on semtag data. This is reflected in, e.g., the fact that even though our $\vec{c} \wedge \vec{w}$ ResNet system outperforms the Plank et al. (2016) system on semtags, we are substantially outperformed on UD 1.2 and 1.3 in this setup. However, adding an auxiliary loss based on our semtags markedly increases performance on POS tagging. In this setting, our tagger outperforms the BI-LSTM system, and results in new state-of-the-art results on both UD 1.2 (95.71% accuracy) and 1.3 (95.67% accuracy). The difference between the BI-LSTM system and our best system is significant at $p < 0.0025$.

The fact that the semantic tags improve POS tagging performance reflects two properties of semantic tags. Firstly, it indicates that the semantic tags carry important information which aids the prediction of POS tags. This should come as no surprise, considering the fact that the semtags abstract over and carry more information than POS tags. Secondly, it indicates that the new semantic tagset and released dataset are useful for downstream NLP tasks. In this paper we show this by using semtags as an auxiliary loss. In future work we aim to investigate the effect of introducing the semtags directly as features into the embedded input representation.

5.3 ResNets for sequence tagging

This work is the first to apply ResNets to NLP tagging tasks. Our experiments show that ResNets significantly outperform standard convolutional networks on both POS tagging and sem-tagging. ResNets allow better signal propagation and carry lower risk of overfitting, allowing for the model to capture more elaborate feature representations than in a standard CNN.

5.4 Pre-trained embeddings

In our main experiments, we initialised the word embedding layer with pre-trained polyglot embeddings. We also compared this with initialising this layer from a uniform distribution over the interval $[-0.05, 0.05)$. For semantic tagging, the difference with random initialisation is negligible, with pre-trained embeddings yielding an increase in about 0.04% accuracy. For POS tagging, however, using pre-trained embeddings increased accuracy by almost 3 percentage points for the ResNet.

6 Conclusions

We introduce a semantic tagset tailored for multilingual semantic parsing. We evaluate tagging performance using standard CNNs and the recently emerged ResNets. ResNets are more robust and result in our best model. Combining word and ResNet-based character representations helps to outperform state-of-the-art taggers on semantic tagging. Coupling this with an auxiliary loss from our semantic tagset yields state-of-the-art performance on the English UD 1.2 and 1.3 POS datasets.

Acknowledgements

The authors would like to thank Robert Östling for tips on ResNets, and Calle Börstell, Johan Sjons, Lasha Abzianidze, and the anonymous reviewers for feedback on earlier versions of this manuscript. We would like to thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high performance computing cluster. This work was partially funded by the NWO–VICI grant "Lost in Translation – Found in Meaning" (288-89-003).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. *CoNLL-2013*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *ACL*, pages 1415–1425.

Johannes Bjerva. 2016. Byte-based language identification with deep convolutional networks. *arXiv preprint arXiv:1609.09004*.

Johan Bos, Valerio Basile, Kilian Evang, Noortje Venhuizen, and Johannes Bjerva. Forthcoming. The Groningen Meaning Bank. In Nancy Ide and James Pustejovsky, editors, *The Handbook of Linguistic Annotation*. Springer, Berlin.

Johan Bos. 2008. Wide-Coverage Semantic Analysis with Boxer. In J. Bos and R. Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, volume 1 of *Research in Computational Semantics*, pages 277–286. College Publications.

Thorsten Brants. 2000. Tnt: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231. Association for Computational Linguistics.

Alastair Butler. 2010. *The Semantics of Grammatical Dependencies*, volume 23. Emerald Group Publishing Limited.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2015. Open-domain name error detection using a multi-task rnn. In *EMNLP*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Grzegorz Chrupała. 2013. Text segmentation with character-level text embeddings. In *Workshop on Deep Learning for Audio, Speech and Language Processing, ICML*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *Procedings of ACL 2016, arXiv preprint arXiv:1603.06147*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781*.

Ann Copestake, Dan Flickinger, Ivan Sag, and Carl Pollard. 2005. Minimal recursion semantics: An introduction. *Journal of Research on Language and Computation*, 3(2–3):281–332.

James Curran, Stephen Clark, and Johan Bos. 2007. Linguistically Motivated Large-Scale NLP with C&C and Boxer. In *Procedings of ACL 2007*, pages 33–36, Prague, Czech Republic.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*, pages 1818–1826.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Kilian Evang, Valerio Basile, Grzegorz Chrupała, and Johan Bos. 2013. Elephant: Sequence labeling for word and sentence segmentation. In *EMNLP*, pages 1422–1426.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2015. Multilingual language processing from bytes. *arXiv preprint arXiv:1512.00103*.

Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *arXiv preprint arXiv:1510.00726*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep learning. `http://www.deeplearningbook.org`. Book in preparation for MIT Press.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*.

Robert Östling. 2016. Morphological reinflection with convolutional neural networks. In *Proceedings of the 2016 Meeting of SIGMORPHON*, Berlin, Germany. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of ACL 2016, arXiv preprint arXiv:1604.05529*.

Sameer S Pradhan, Wayne Ward, Kadri Hacioglu, James H Martin, and Daniel Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *HLT-NAACL*, pages 233–240.

Anders Søgaard, Anders Johannsen, Barbara Plank, Dirk Hovy, and Hector Martinez. 2014. Whats in a p-value in nlp? In *CoNLL-2014*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. A unified tagging solution: Bidirectional lstm recurrent neural network with word embedding. *arXiv preprint arXiv:1511.00215*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

A Supervised Approach for Enriching the Relational Structure of Frame Semantics in FrameNet

Shafqat Mumtaz Virk and **Philippe Muller** and **Juliette Conrath**
IRIT, Toulouse University
118 Route de Narbonne
31062 Toulouse, France

Abstract

Frame semantics is a theory of linguistic meanings, and is considered to be a useful framework for shallow semantic analysis of natural language. FrameNet, which is based on frame semantics, is a popular lexical semantic resource. In addition to providing a set of core semantic frames and their frame elements, FrameNet also provides relations between those frames (hence providing a network of frames i.e. FrameNet). We address here the limited coverage of the network of conceptual relations between frames in FrameNet, which has previously been pointed out by others. We present a supervised model using rich features from three different sources: structural features from the existing FrameNet network, information from the WordNet relations between synsets projected into semantic frames, and corpus-collected lexical associations. We show large improvements over baselines consisting of each of the three groups of features in isolation. We then use this model to select frame pairs as candidate relations, and perform evaluation on a sample with good precision.

1 Introduction

In the area of formal linguistics, frame semantics is a theory of meanings, which was introduced by Charles J. Fillmore and his colleagues back in the early 1980's. Frame semantic analysis (Das et al., 2014), which is based on frame semantics, itself is a type of shallow semantic analysis in which the focus is on predicates and their arguments. Frame semantic analysis is abstracting away from single verbal predicates, used in other semantic analysis approaches such as Propbank-based work (Kingsbury and Palmer, 2012), to "semantic frames" (Fillmore, 1982). The idea in frame semantics is to group predicates referring to similar events, processes, and/or object types under the umbrella term 'a semantic frame', which can be expressed with different parts of speech and with arguments that can have various syntactic realizations. For instance, the sentence in example (1-a) could be seen as introducing a commerce frame, indicated by the lexical unit *buy*. This frame has a set of necessary arguments, a *buyer*, some *goods*, and a *seller*, which are realized by role fillers *Lester*, *a car* and *Jimmy* in the example. Frames can be realized with different lexical units and syntactic constructions, so that sentences (1-b)-(1-c) would have a similar meaning with respect to the frame in question.

(1) a. $[\text{Lester}]_{\text{buyer}}$ $[\text{bought}]_{\text{commerce}}$ $[\text{a car}]_{\text{goods}}$ $[\text{from Jimmy}]_{\text{seller}}$.
b. $[\text{Jimmy}]_{\text{seller}}$ $[\text{sold}]_{\text{commerce}}$ $[\text{a car}]_{\text{goods}}$ $[\text{to Lester}]_{\text{buyer}}$.
c. $[\text{Lester}]_{\text{buyer}}$'s $[\text{purchase}]_{\text{commerce}}$ of $[\text{Jimmy}]_{\text{seller}}$'s $[\text{car}]_{\text{goods}}$ (...)

Most recent NLP work on such semantic frames are based on FrameNet (henceforth FN) (Baker et al., 1998), a resource listing frames, how they can be evoked, and their argument types. FrameNet also lists how the frames are organized with respect to each other, with a set of frame-to-frame relations. If FrameNet based analysis has proven useful for certain tasks such as information extraction (Surdeanu et al., 2003), question-answering (Shen and Lapata, 2007), or coreference resolution (Ponzetto and Strube, 2006), its frame structure is assumed to be useful on its own for semantic analysis (Burchardt et al.,

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3542–3552, Osaka, Japan, December 11-17 2016.

2005), or paraphrase extraction (Hasegawa et al., 2011). It also provides additional information for semantic analysis in contextual role linking, as demonstrated in (Li et al., 2015), or to improve prediction of roles, as in the work of Kshirsagar et al. (2015). The relations are also useful to build thesauri and to retrieve semantically related words (Ruppenhofer et al., 2006). However, one big issue with FrameNet is its partial coverage of the lexicon and the intended set of frames, which translates also into a partial coverage of frame relations. Numerous studies address FrameNet's lack of lexical coverage (Pennacchiotti et al., 2008; Das and Smith, 2012; Pavlick et al., 2015). In contrast, little work has been done on extending frame relations except by Ovchinnikova et al. (2010), in which cluster of frames are proposed based on collocation information, along with principles to be respected when adding new relations, or Pennacchiotti and Wirth (2009), which defines a generic notion of "relatedness" between frames, with no semantics to compare this to FrameNet relations. Both approaches are unsupervised, and provide little evaluation of the relevance to the intended FN structure.

We present a supervised approach to enrich FrameNet's relational structure by training a model on the existing set of frame relations and a rich set of features combining linguistic and structural information. Further, we also leverage external resources such as WordNet. The resulting model is used to predict new frame-to-frame relations whose validity is then manually evaluated. The rest of the paper is organized as follows: Section 2 presents in more detail FN frame relations, Section 3 presents our methodology and the three group of features we use for training a supervised model. Section 4 presents our experimental results based (1) on a separate test set taken from already existing FrameNet relations, and (2) a human evaluation of newly proposed relations.

2 FrameNet Relations

FrameNet is a lexical semantic resource which is based on the theory of frame semantics (Fillmore, 1982). A set of script-like descriptions, known as semantic frames, of real world situations is collected and maintained along with the participants of those situations. Each of the semantic frames has a set of associated words (known as frame evoking elements, FEE) which can evoke a particular predicate. The participants of a situation (called frame elements, FE instead of the more classical term 'semantic role') are also identified for each frame. In addition, each semantic frame is coupled with example sentences taken from naturally occurring natural language text. FrameNet-1.5 has 1230 semantic frames, 11829 lexical units, and 173018 example sentences.

As mentioned previously, FrameNet has defined a set of frame-to-frame relations, and has proposed connections of certain frames to certain other frames, thus providing a network. The backbone of this network is built on a hierarchy of predicate types, and typical sub-events of specific situations. Event participants, also called frame elements, of the connected frames may also have one-to-one connections. Some of these frames are introduced purely for the coherence of the structure and may not have associated lexical items ("unlexicalized frames"). The following is a list of relations defined between two frames X and Y:

- Subframe(X,Y): a complex scenario X, e.g. CRIMINAL_PROCESS, is composed of typical sub-events (e.g. Y can be TRIAL, SENTENCING...).
- Precedes(X,Y): X is before Y in a typical scenario, e.g. TRIAL precedes SENTENCING.
- Inheritance(X,Y): A relation between frames where one frame is a more specific version of another frame e.g. ANIMALS inherits from BIOLOGICAL_ENTITY.
- Causative_of(X,Y): X is a potential cause of Y, e.g AIMING is causative_of HIT_OR_MISS.
- Inchoative_of(X,Y): X is an inchoative of Y, e.g. ROTTING is inchoative of BEING_ROTTEN.
- Perspectivized_in(X,Y): X represents a specific perspective of Y, e.g. COMMERCE_BUY and COMMERCE_SELL are two different perspectives of COMMERCE_TRANSFER_GOODS.
- Using(X,Y): X somehow involves Y, e.g. PEOPLE_BY_AGE uses the AGE frame.

- See_also(X,Y): Frames that have some similarities, but need to be differentiated carefully (e.g. MEASURABLE_ATTRIBUTE and DIMENSION).

Figure (1) gives an example of a subset of relations around the CRIMINAL PROCESS Frame, that will be used for illustration in the rest of the paper.

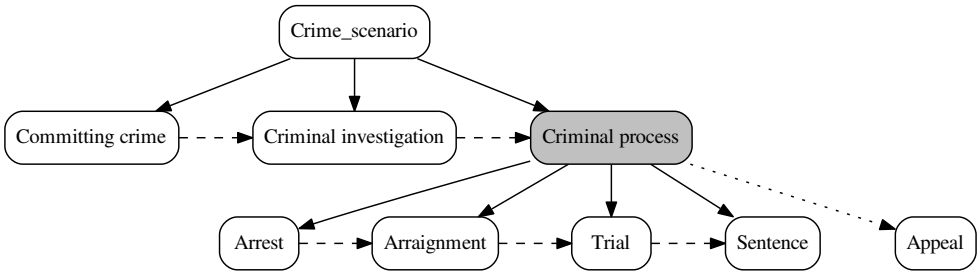


Figure 1: Example sub-graph around the CRIMINAL PROCESS frame; Solid, dashed and dotted lines respectively indicate the "SubFrame", "Precede" and "Using" relations. Only subframes of CRIMINAL PROCESS are shown.

3 Experimental Design and Features

Our goal is to provide a method to enrich FrameNet structures with new relations where supervision is given by the existing frame relations. Our experimental methodology is thus two-fold: 1) we trained a discriminative model to predict frame-to-frame relations in a supervised setting using the existing relations listed in the original FrameNet and selected counter-examples; and 2) we applied this model to predict likely frame-to-frame relations that are not already listed. We used rich features from three different sources, taking inspiration from the unsupervised relatedness measures of (Pennacchiotti and Wirth, 2009), which are based on lexical collocations, but adding more features: structural features from the existing FrameNet network, information from WordNet relations between synsets projected into semantic frames, and different additional corpus-collected lexical associations. Since we want to focus on event-denoting predicates, which have arguably richer and potentially more interesting argument structures, and are more likely to be related in common scenarios, we restricted ourselves to frames with at least one verb trigger. This restricts the existing relations to 824 frame pairs. Next we describe the different set of features used in our experiments.

3.1 WordNet Based Features

WordNet provides useful lexical and semantic relations between synonym sets. The most important among those relations are hypernymy/hyponymy and meronymy/honolymy. Hypernym and hyponym (and troponymy for verbs) are super-subordinate relations and link more general synsets to specific ones and could be relevant for Frame inheritance, and thus make the first source of information for predicting frame relations. As we are interested in relationships between pairs of frames, instead of pair of synsets, we need to transfer knowledge about relations between synsets to useful features between frames whose verbs appear in WordNet synsets. There have been attempts at matching WordNet sense inventory to FrameNet frames (Shi and Mihalcea, 2005; Tonelli and Pighin, 2009) but since both resources are incomplete, we decided to compute frame relatedness using an existing WordNet relation between synsets which include verbs from the given frames, irrespective of their senses. This certainly introduces some noise, that we hope to control with redundancies in frame-to-frame associations. For our purposes, we have divided WN-based features into two groups: (1) occurrence-based features (2) similarity-based features. Occurrence-based features are simple counts of existence of a particular relation between a pair

from all possible pairs of all senses of all lexical units of the two candidate frames. By taking the existence of a particular relation between senses and summing them up, we are projecting the knowledge about WordNet's synset relations to FrameNet's frames.

For a given pair of frames (F_i, F_j), and a given WordNet relation R (i.e. hypernym, hyponym, all meronym relations, all holonyms relations, attributes, entailments, causes, also_sees) the number of occurrence-based feature values were computed using the following formula:

$$number\_rel(F_i, F_j, R) = \sum_{(lu_i, lu_j) \in LU_{F_i} \times LU_{F_j}} \left[\sum_{(s_{lu_i}, s_{lu_j}) \in S_{lu_i} \times S_{lu_j}} R(s_{lu_i}, s_{lu_j}) = True \right]$$

LU_{F_i} and LU_{F_j} are sets of lexical units of frames F_i and F_j respectively, while S_{lu_i} and S_{lu_j} are sets of various senses of the corresponding lexical units in WordNet. The values of the semantic similarity based features were computed using a similar formula (but averaged) for a given similarity function f and two frames F_1 and F_2:

$$similarity(F_i, F_j, f) = \frac{1}{|LU_{F_i} \times LU_{F_j}|} \times \sum_{(lu_i, lu_j) \in LU_{F_i} \times LU_{F_j}} \left[\frac{\sum_{(s_{lu_i}, s_{lu_j}) \in S_{lu_i} \times S_{lu_j}} f(s_{lu_i}, s_{lu_j})}{|S_{lu_i} \times S_{lu_j}|} \right]$$

The semantic similarity measures are classical wordnet similarity measures: Path, Wu-Palmer (Wu and Palmer, 1994) and Leacock-Chodorow (Leacock and Chodorow, 1998), computed using NLTK's (Bird, 2006) Python interface. They all take into consideration the path between synsets in WordNet's hierarchy with different normalization factors. For instance Wu-Palmer is based on the depths of two synsets in the WordNet hierarchy along with the depth of the least common subsumer. The path similarity measure is a semantic association measure which is based on the shortest path that connects two synsets in the taxonomy in which the senses occurred. LCH similarity takes into consideration the depth of the taxonomy in addition to the shortest path, and is computed as $-log(p/2d)$, where p denotes the shortest path and d is the depth. For the purpose of projecting WordNet's synset knowledge to FrameNet's frames, we are summing up and averaging the corresponding semantic similarity values of all possible combinations of various senses of all lexical units of the two candidate frames.

3.2 FrameNet Based Features

FrameNet network structure can also be a useful resource to compute frame-relatedness between non explicitely related frames. (Pennacchiotti and Wirth, 2009) proposed to apply equivalents of Wu-Palmer similarity and Hirst-St.Onge's (Hirst and St-Onge, 1998) measure to FrameNet's hierarchy. They also suggested to take into account frame element overlap (the proportion of predicate roles having the same name). Here, the frame element overlap similarly is based on the number of frame-elements (role names) shared by the two candidate frames. Hirst-St.Onge measure is a semantic similarity measure which is based on the path connecting two WordNet synsets and how often one needs to change direction to reach one from the other in the network. A "change of direction" would be for instance a path along an hypernym relation then an hyponym relation (which yields a likely co-hyponym). The intuition is that two synsets are semantically closer if they are connected through a "not too long path which does not change direction too often". This can be applied similarly over FrameNet's relations, even though some of these are vaguer than WordNet relations. An example path in figure 1 would be the path (SubFrame^{-1}+Using) between SENTENCE and APPEAL.

In addition to the hierarchy-based features, frame verbal definitions can potentially provide useful information for frame relatedness, and we decided to also use a definition overlap as a feature, with a Jaccard between the sets of open-class words of each definition (open(F) being the set of open-class words in F definition):

$$def\_overlap(F_1, F_2) = \frac{\|open(F_1)\| \cup \|open(F_2)\|}{\|open(F_1)\| \cap \|open(F_2)\|}$$

3.3 Corpus-Based Features

In addition to the previous sets of features, which make use of existing resources or of the existing structure we want to expand, we derived clues from large corpora where frame lexical units can be observed in the same contexts, indicating a potentially regular relation between their corresponding frames. We collected both general ococcurences, in the spirit of (Pennacchiotti and Wirth, 2009), and targetted associations of verbs in explicit discursive contexts, inspired by (Conrath et al., 2014).

Frame Coocurrences We used statistics of frame co-occurrences from a large-scale corpus to predict frame relatedness, relying on the intuition that related frames tend to co-occur more often in the same context within a given corpus. The context could be either a document, a sentence, or a specific number of sentences. We computed the co-occurrence of frames as point-wise mutual information (PMI) within the GigaWord corpus (Graff et al., 2003). Similar information has already been used in (Pennacchiotti and Wirth, 2009), but we deal with the frame ambiguity problem differently.

Ideally, one would like to compute frame co-occurrences statistics from a frame annotated corpus which is big enough for machine learning tasks. FrameNet does provide a frame annotated corpus, but it is too small to be truly useful for making generalizations. A workaround is to use an unannotated corpus together with FN's frame-evoking lexicon to decide which frame is being triggered by a particular lexical unit. The problem is now to resolve the ambiguity in cases where a lexical unit can potentially trigger more than one frame. (Pennacchiotti and Wirth, 2009) suggested a weighted co-occurrence measure, which gave lower weights to the co-occurrence of ambiguous words. The probabilities of sense occurrences of lexical units were learned from the WordNet sense tagged corpus SemCor. Our approach is slightly different in the sense that instead of learning word-sense probabilities first and then mapping it to frames, we directly learn the probabilities of lexical units triggering particular frames from FrameNet's annotated corpus using the ratio of the number of frames triggered by a lexical unit lu in the FrameNet corpus and of the total number of occurrences of lu in the corpus. This is arguably more direct and only use FrameNet information, although both approaches need annotated data. The probabilities are then used to compute a weighted PMI between frames (the weighting function simply sums up the probabilities of a lexical unit triggering a particular frame F over the entire GigaWord).

Lexical Coocurrences We also used as features measures of semantic similarity derived from corpus-based specific associations between lexical items. If we can find valid lexical associations to match the targeted frame relations, we can obtain relevant association measures. We adapted the method of (Conrath et al., 2014), in which semantic associations between verbs are derived from cooccurrences between two verbs and certain classes of discourse markers, using mutual information measures. This shallow discourse analysis can be seen as extraction of typical semantic relations between verbs. Using the Penn Discourse Treebank (Prasad et al., 2008) list of markers, grouped into semantic classes, we computed six types of associations between verbs: (1) causal, (2) temporal, (3) continuation, (4) contrast, (5) disjunction, (6) elaboration. Each corresponds to a coherent set of markers and can provide clues to corresponding FrameNet relations: causative-of for (1), precede for (2) and (3), subframe for (6), and (4) and (5) for some form of looser relatedness that is sometimes encoded as "see-also" or "using" in FrameNet. For each class we implemented the three best measures according to (Conrath et al., 2014), which correspond to different normalizations or combinations of the verbs and discourse relation cooccurrences : normalized PMI (Evert, 2005), a specificity measure taken from (Mirroshandel et al., 2013), and a combined association measure they call w_combined. They are supposed to capture different aspects of the targeted lexical associations. Let P(V_1,V_2,R) be the probability of the association of the two verbs V_1,V_2 with the given semantic relation R:

$$\mathrm{NPMI}(V_1, V_2, R) = \mathrm{PMI}(V_1, V_2, R)/(-2\log(P(V_1, V_2, R)))$$

$$\textit{specificity}(V_1, V_2, R) = \frac{1}{3} \times \left(\frac{P(V_1, V_2, R)}{\sum_i P(V_1, V_i, R)} + \frac{P(V_1, V_2, R)}{\sum_i P(V_i, V_2, R)} + \frac{P(V_1, V_2, R)}{\sum_i P(V_1, V_2, R_i)}\right)$$

$$W_{combined}(V_1, V_2, R) = \frac{1}{3}(w_{V_1} + w_{V_2} + w_R)$$

with: $w_{V_1} = \frac{P(V_1,V_2,R)}{\max_i(P(V_i,V_2,R))}$, $w_{V_2} = \frac{P(V_1,V_2,R)}{\max_i(P(V_1,V_i,R))}$, and $w_R = \frac{P(V_1,V_2,R)}{\max_i(P(V_1,V_2,R_i))}$.

We computed the corresponding association values for frame-pairs using the same averaging scheme as for the previous features over groups of lexical units, see 3.1

4 Experiments and Results

The previous features were computed for two sets of frame-pairs: (1) All those frames-pairs which have any of the frame-relations mentioned in section 2. As mentioned previously, we restricted ourselves to frames denoting events, i.e. which have at least one verb trigger. There are 824 such instances. (2) The set of all unrelated frame-pairs. From these two sets, a balanced set of 1648 frame-pairs (824 positive, and 824 negative) was collected, 10% of which was set aside for testing the chosen model. The remaining 90% of the balanced set was used to train and evaluate different models with different feature combinations. The best combination, evaluated by cross-validation, was evaluated on the test set, and then used to train a new model on the full balanced set, which we applied to all "unrelated" frame pairs (i.e. to predict likely frame relations between frame pairs that are not already listed in the FrameNet), with a fixed *a priori* confidence threshold in order to focus first on precision of the candidate pairs[1].

4.1 Training a Supervised Model

For the first part of our method, we trained a binary classifier to decide whether a given pair of frames is related or not. To provide negative examples, we randomly sampled the set of unrelated frame pairs. Of course we don't know the real proportion of frames that should be (ideally) related, but we suspect that such a relational structure is likely to be sparse, and thus taking random pairs should yield mostly truly unrelated pairs.

Assuming the actual relational structure is sparse, we probably face an imbalanced classification problem, in which we want to identify primarily the minority class. That is why we chose to balance the number of positive and negative instances during training, since it means we are doing majority class undersampling, a classic simple way of addressing class imbalance, and it is also a simple way to evaluate the relevance of our features. This is likely to generate too many candidate pairs on the test, degrading precision while helping recall of the positive class, something we can control *a posteriori* on new instances by imposing a confidence threshold on the classifier output (see below).

We used a Random Forest classifier with 1000 estimators and a minimum of 10 instances for splitting, and the implementation provided in the scikit-learn package (Pedregosa et al., 2011). Random Forest is a robust classifier used in a variety of tasks and perform best among the classifiers evaluated on cross-validation on the training set. As baselines, we tested each separate group of features: WordNet transfer, corpus cooccurences, and FrameNet features. Note that a majority/random baseline would give a score of 50% with the balanced setting we chose. We observed that the combination of features proves very useful, as cross-validation accuracies for the three groups are much lower than the full model. Interestingly, all groups of features seem necessary, as even removing the group which performs worse by itself lowers the training accuracy by 5% (ablations of each group of features were tried by cross-validation on the train only). Final results are presented in Table 1. Not surprisingly, the most informative features come from FrameNet itself, since positively related frames are more likely to be in "denser" parts of the network, at least those parts that have been the focus of the lexicographic work, and this can be seen as a bias of the model. Note however that (1) these denser areas of the network might still be incomplete and (2) the other sources of information definitely contribute to the overall result, adding 10 points over

[1] An archive can be found at `http://www.irit.fr/~Philippe.Muller/fn_structure.zip` which contains instances and their extracted features, the manually annotated sample of frame relation instances, and various scripts to reproduce the experimental results.

FrameNet based features. We provide confidence intervals on the scores (which are proportion estimations), and we can see the different models' intervals don't overlap much, even on the small test sample.

Model	Cross-Validation / training	Test	Confidence Interval
WordNet-based	60.3	63.4	2.5/7.4
Frame information	78.1	79.3	2.1/6.2
Corpus-based	64.8	60.4	2.4/7.5
Full Model	87.3	88.4	1.7/4.9

Table 1: Accuracy of predictions with balanced dataset (in %); confidence intervals $\pm$ are given at 95% for the train/test accuracy.

4.2 Finding New Relations

We used the previous supervised model on the set of all possible pairs of frames to predict potential new relations. Since the model has been trained on a balanced set, we probably over-generate positive labels, so as a first basic step to improve the precision of the extraction, we set an arbitrary cutoff for the confidence level according to the model, at the value 0.8. This yielded over 1500 pairs, out of which we randomly selected 100 pairs for validation. We mixed the pairs with an equal number of 100 "distractors" (random pairs absent from FrameNet), to prevent annotation bias. Note that, again, we have no *a priori* way of ensuring these distractors are not actual unknown relations, but we expect true relations to be rare in this set. Two of the authors then judged if one of the FN relations could hold, based on the summary of the relations from the FrameNet book, and a description of the frames as defined in FN, i.e. a short definition, sometimes a few example sentences, a list of frame element names (arguments of the frame), and a list of lexical units that can evoke the frame. They had to label each pair as Yes/No, and were asked to indicate a tentative label from the set of FrameNet relations (there could be 0, 1, or 2 relations according to their level of confidence in their decision). We did the same procedure on the top 100 frame pairs according to the classifier, also mixed with 100 random pairs. We computed inter-annotator agreement with Cohen's Kappa which was 0.65, generally considered as an acceptable agreement above 0.6, at least for semantic tasks. Raw agreement was 0.83. To estimate agreement on the labels, we consider judgments for common Yes decisions, and we considered the intersection of proposed labels (if two labels were proposed and one was common we counted that as a positive). Keeping only pairs for which each annotator gave an answer (only half of the pairs), the Kappa was at 0.5, mainly because of agreement on inheritance pairs. Given that a Kappa of 0.5 indicates moderate agreement, even with the lenient setting we provided, the relation types need some clarification. We can still tentatively conclude that the inheritance relation can be annotated, but in the remaining, we only discuss relatedness.

Frame 1	Frame 2
Building	Manufacturing
Evidence	Reasoning
Motion_directional	Path_shape
Cause_motion	Motion
Cause_impact	Cause_motion
Intentional_traversing	Path_shape
Discussion	Statement
Make_cognitive_connection	Relating_concepts
Destroying	Killing
Cause_harm	Cause_to_fragment

Table 2: Top ten true new relations predicted, according to the probability given by the model.

Frame 1	Frame 2
Filling	Removing
Change_position_on_a_scale	Motion
Bringing	Ride_vehicle
Communication	Evidence
Change_position_on_a_scale	Path_shape
Getting_vehicle_underway	Removing
Change_direction	Path_shape
Bringing	Change_direction
Change_direction	Removing
Change_direction	Departing

Table 3: Top ten false positive new relations predicted, according to the probability given by the model.

Both authors adjudicated the differences in the relatedness question, and only pairs with both positive judgments were finally considered positive. Out of the top 100 positive candidates, 63 were labeled as positive by human annotators giving a 63% precision, and 52 out of the 100 selected above the 0.8 threshold, with respectively ± 9.3 and $\pm 9.8\%$ confidence interval. This means a large proportion of proposed frame pairs could be considered for addition, and that proportion does not decrease too quickly when the confidence of the classifier decreases. We cannot estimate recall, as we don't know how many relations there should be overall, but we observed that only 4 pairs out of the distractors were judged positive, which seems to indicate relations are indeed rare with respect to all possible pairs. Note that estimating recall among a subsample of n frames would require a number of judgments equal to $n^2 \times$ the number of relation types. Table 2 and 3 shows a list of top 10 true and false positive relations predicted by our model.

We did an error analysis on the 37 false positive in the top 100 test set, and came to the conclusion that most frame pairs were picked up as related because they involve related lexical units (and often with different senses), rather obviously, and that they involve either (1) co-hyponyms at different levels of granularity; or (2) frame describing opposite events, such as Removing and Filling, which do not correspond to any FrameNet relation. Case (2) is arguably a FrameNet problem, since a lot of frames involve antonyms, and could indicate here missing frames at a higher level up the hierarchy of inheritance, an inconsistency noted in previous work (Hasegawa et al., 2011). As an example the two adjectives *easy* and *difficult* are part of the same Frame DIFFICULTY, while the verbs *empty* and *fill* are in separate frames EMPTYING and FILLING. Case (1) is an interesting perspective for improvement: relations should not be considered between frame pairs in isolation, but with respect to all existing or predicted relations, in a global way. If a frame F_1 is already related to a frame F_2, there should not be a relation between F_1 and, e.g., subframes of F_2. Note also that most Frames that are siblings in a subframe relation also have explicit relations, usually a "Precede" relation (see figure 1), but also vaguer links such as "Using".

5 Conclusion and Future Directions

Our main contribution is to present the first supervised method to provide candidate relations between FrameNet frames, using a rich set of structural and linguistic features inspired by previous unsupervised work which did not provide evaluation with respect to FrameNet's set of relations. A secondary contribution is in showing that Frame relation annotation is possible with good agreement, based only on Frame descriptions, although labeling proves to be more difficult. This also raised the question of the completeness of relation types in FrameNet, or at least the consistency of certain decisions that are made in grouping lexical predicates. As a perspective, we plan to evaluate our model predictions at various levels of confidence, to determine the best compromise between precision and recall when providing relation instance candidates for human validation. It would also be interesting to combine this approach with work on lexical expansion of FN, such as (Pavlick et al., 2015), or use an unsupervised method to

pre-filter candidate pairs, using looser lexical association resources, such as the Moby thesaurus (Ward, 1996).

An interesting perspective given the conclusions of the error analysis would be to try to predict new relations in a global way, using principles on the well-formedness of such ontological/lexical relationships, following the ontological principles given in (Ovchinnikova et al., 2010), or work on lexical taxonomy induction, see e.g. (Navigli et al., 2011).

As mentioned previously, FrameNet does provide a list of FE-to-FE relations for each of the connected frame pairs. In this paper, we have proposed ways to predict new frame-to-frame relations, and we plan to explore the possibilities of automatically linking frame-elements of the newly proposed frame-to-frame relations.

Acknowledgments

This work was funded by the French National Research Agency (ASFALDA project ANR-12-CORD-023). Juliette Conrath was supported by the Polymnie ANR project. We thanks the reviewers for helpful comments.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, Montreal, Quebec, Canada, August. Association for Computational Linguistics.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Aljoscha Burchardt, Anette Frank, and Manfred Pinkal. 2005. Building text meaning representations from contextually related frames - a case study. In *Proceedings of the Sixth International Workshop on Computational Semantics*, pages 12–. to appear.

Juliette Conrath, Stergos Afantenos, Nicholas Asher, and Philippe Muller. 2014. Unsupervised extraction of semantic relations using discourse cues. In *International Conference on Computational Linguistics - COLING 2014*, pages pp. 2184–2194. Association for Computational Linguistics.

Dipanjan Das and Noah A. Smith. 2012. Graph-based lexicon expansion with sparsity-inducing penalties. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 677–687, Montréal, Canada, June. Association for Computational Linguistics.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Comput. Linguist.*, 40(1):9–56, March.

Stefan Evert. 2005. *The statistics of word cooccurrences*. Ph.D. thesis, Stuttgart University.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press.

Charles J. Fillmore, 1982. *Frame semantics*, pages 111–137. Hanshin Publishing Co., Seoul, South Korea.

Graff, David, and Christopher Cieri. 2003. English gigaword ldc2003t05. Web Download. Philadelphia: Linguistic Data Consortium.

Yoko Hasegawa, Russell Lee-Goldman, Albert Kong, and Kimi Akita. 2011. Framenet as a resource for paraphrase research. *Constructions and Frames*, 3(1):104–127.

Graeme Hirst and David St-Onge. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. In Fellbaum (Fellbaum, 1998).

Paul Kingsbury and Martha Palmer. 2012. From treebank to propbank. In *Proceedings of LREC 2012*.

Meghana Kshirsagar, Sam Thomson, Nathan Schneider, Jaime Carbonell, Noah A. Smith, and Chris Dyer. 2015. Frame-semantic role labeling with heterogeneous annotations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 218–224, Beijing, China, July. Association for Computational Linguistics.

Claudia Leacock and Martin Chodorow. 1998. Combining local context and wordnet similarity for word sense identification. In Fellbaum (Fellbaum, 1998).

Ru Li, Juan Wu, Zhiqiang Wang, and Qinghua Chai. 2015. Implicit role linking on chinese discourse: Exploiting explicit roles and frame-to-frame relations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1263–1271, Beijing, China, July. Association for Computational Linguistics.

Seyed Abolghasem Mirroshandel, Alexis Nasr, and Benoît Sagot. 2013. Enforcing subcategorization constraints in a parser using sub-parses recombining. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 239–247, Atlanta, Georgia, June. Association for Computational Linguistics.

Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *Proceedings of the* 22^{nd} *International Joint Conference on Artificial Intelligence*, pages 1872–1877, Barcelona, Spain.

Ekaterina Ovchinnikova, Laure Vieu, Ro Oltramari, Stefano Borgo, and Theodore Alex. 2010. Data-driven and ontological analysis of framenet for natural language reasoning. In *In Proc. of LREC10*.

Ellie Pavlick, Travis Wolfe, Pushpendre Rastogi, Chris Callison-Burch, Mark Dredze, and Benjamin Van Durme. 2015. Framenet+: Fast paraphrastic tripling of framenet. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 408–413, Beijing, China, July. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Marco Pennacchiotti and Michael Wirth. 2009. Measuring frame relatedness. In Alex Lascarides, Claire Gardent, and Joakim Nivre, editors, *EACL*, pages 657–665. The Association for Computer Linguistics.

Marco Pennacchiotti, Diego De Cao, Roberto Basili, Danilo Croce, and Michael Roth. 2008. Automatic induction of FrameNet lexical units. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 457–465, Honolulu, Hawaii, October. Association for Computational Linguistics.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 192–199, New York City, USA, June. Association for Computational Linguistics.

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

Josef Ruppenhofer, Michael Ellsworth, Miriam R.L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. 2006. *FrameNet II: Extended Theory and Practice*. International Computer Science Institute, Berkeley, California. Distributed with the FrameNet data.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 12–21, Prague, Czech Republic, June. Association for Computational Linguistics.

Lei Shi and Rada Mihalcea. 2005. Putting the pieces together: Combining FrameNet, VerbNet, and WordNet for robust semantic parsing. In *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics*, Mexico.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 8–15, Sapporo, Japan, July. Association for Computational Linguistics.

Sara Tonelli and Daniele Pighin. 2009. New features for framenet - wordnet mapping. In *CoNLL'09: Thirteenth Conference on Computational Natural Language Learning*, Boulder, CO, USA, 06/2009. ACL, ACL.

G. Ward. 1996. Moby thesaurus. Moby project.

Zhibiao Wu and Martha Palmer. 1994. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138, Las Cruces, New Mexico, USA, June. Association for Computational Linguistics.

Reddit Temporal N-gram Corpus and its Applications on Paraphrase and Semantic Similarity in Social Media using a Topic-based Latent Semantic Analysis

Anh Dang[1], Abidalrahman Moh'd[1], Aminul Islam[2], Rosane Minghim[3], Michael Smit[4], and Evangelos Milios[1]

[1]{anh,amohd,eem}@cs.dal.ca, Dalhousie University, 6050 University Avenue, Halifax, NS, Canada B3H 4R2

[2]aminul@louisiana.edu, School of Computing and Informatics, University of Louisiana at Lafayette Lafayette, LA, USA 70503

[3]rminghim@icmc.usp.br, University of São Paulo-USP, ICMC, São Carlos, Brazil

[4]mike.smit@dal.ca, School of Information Management, Dalhousie University, 6100 University, Halifax, NS, Canada B3H 4R2

Abstract

This paper introduces a new large-scale n-gram corpus that is created specifically from social media text. Two distinguishing characteristics of this corpus are its monthly temporal attribute and that it is created from 1.65 billion comments of user-generated text in Reddit. The usefulness of this corpus is exemplified and evaluated by a novel Topic-based Latent Semantic Analysis (TLSA) algorithm. The experimental results show that unsupervised TLSA outperforms all the state-of-the-art unsupervised and semi-supervised methods in SEMEVAL 2015: paraphrase and semantic similarity in Twitter tasks.

1 Introduction

A word n-gram is a continuous sequence of n words from a corpus of texts or speech. Word n-gram language models are widely used in Natural Language Processing (NLP), such as speech recognition, machine translation, and information retrieval. The effectiveness of a word n-gram language model is highly dependent on the size and coverage of its training corpus (Clarke et al., 2002). A simple algorithm can outperform a more complicated algorithm if it uses a larger corpus (Norvig, 2008). Many large-scale corpora (Brants and Franz, 2006; Baroni et al., 2009; Wang et al., 2010b) based on web contents have been created for this purpose. As the use of social media is increasing, Online Social Networks (OSNs) have become a norm to spreading news, rumours, and social events (Kwak et al., 2010). This growing usage of social media has created both challenges and opportunities. One major challenge is that social media data is intrinsically short and noisy. A study by (Wang et al., 2010a) revealed that different text corpora have significantly different properties and lead to varying performance in many NLP applications. More importantly, we observe that there is no existing large-scale n-gram corpus that is created specifically from social media text. This has motivated us to create an n-gram corpus that is derived from 1.65 billion comments in the Reddit corpus (Baumgartner, July 2015) and make it available to the research community. There are two main features of this corpus that do not exist in the available large-scale corpora in the literature: monthly time-varying (temporal) and purely social media text. This corpus will allow researchers to analyze and make sense of massive social network text, such as finding corresponding terms across time (Zhang et al., 2015) and improving named entity recognition in tweets (Li and Liu, 2015). Moreover, a cloud-based visualization interface is implemented to allow end users to query any n-gram from the corpus.

Although there are many applications that can be derived from this corpus, in this paper, we use the Paraphrase Identification (PI) and Semantic Similarity (SS) tasks of SEMEVAL 2015 (Xu et al., 2015) to exemplify the usefulness of this corpus. Paraphrases are words, phrases or sentences that have the same meaning, but their vocabulary may be different (Xu et al., 2015). PI and SS tasks have a strong correlation, as both focus on the underlying structural and semantic similarity between two texts (e.g., "selfie" is a paraphrase of "picture of myself"). Improving the results of PI and SS helps to increase the performance of NLP systems, such as statistical machine translation (Madnani et al., 2007) and plagiarism detection (Barrón-Cedeño et al., 2013). PI and SS have been studied intensively for formal

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3553–3564, Osaka, Japan, December 11-17 2016.

text with important results as shown in (Par, 2016). As social media text is usually very short (e.g., 140-character limit for Twitter) and noisy (flexible nature of personal communication), many NLP systems suffer from the large degree of spelling, syntactic and semantic variants, for example, "ICYMI"= "In case you missed it" or "b/c I love u" = "Because I love you". Traditional approaches have been studied intensively and proved not to work well for social media text (Zanzotto et al., 2011). A few preliminary results have shown that the shortness and noisiness of social media text have significantly decreased the performance of PI (Zanzotto et al., 2011; Xu et al., 2013) and SS tasks (Guo and Diab, 2012; Dang et al., 2015a). In this paper, we proposed a Topic-based Latent Semantic Analysis (TLSA) approach for the SS task, which assigns a semantic similarity score between two social media texts. Next, we use this similarity score to determine if two texts are a paraphrase of each other.

Latent Semantic Analysis (LSA) has been widely used for semantic text similarity tasks because of its simplicity and efficiency (Landauer et al., 1998). LSA has been used as a strong benchmark in the Microsoft Research sentence completion challenge (Zweig and Burges, 2011) and its baseline has outperformed a few state-of-the-art neural network models (Mikolov et al., 2013). However, LSA has its own drawbacks. Its models are trained on a large corpus where words in the same document have a stronger relationship. This does not consider how close two words are in a text ("apple" and "fruit" are closer in the 5-gram "apple is a fruit" instead of a whole document) (Hofmann, 1999). Another example is two topics "Barack Obama" and "Hillary Clinton" have a different meaning in two contexts "2012 US presidential race" and "2016 US presidential race". In the first one, they are opponents, while in the second one, "Barack Obama" endorsed "Hillary Clinton". In addition, LSA is usually trained on a whole corpus. This makes it not scalable with an intrinsic, dynamic, and large-scale nature of social network data. To address this issue, we proposed an approach to train an LSA model that considers the topic being discussed. This proposed LSA model is trained on word 5-grams instead of whole documents. The proposed TLSA method achieved the best result for the SS task and is more scalable compared to other LSA models. Combining TLSA with sentiment analysis, the proposed approach also achieved the best result for PI task in SEMEVAL 2015. These are the contributions of our paper:

1. We create a new word n-gram (1-5) social network corpus from 1.65 billion comments of Reddit[1]. This corpus has two distinctive characteristics that are useful for social media applications: temporal and large-scale social media text.

2. We implement a cloud-based visualization interface so that end users can query and analyze the social media n-grams in real time.

3. We propose TLSA[2], a Topic-based Latent Semantic Analysis model that is trained on word 5-grams from social media text. To the best of our knowledge, there is no similar work that employs a topic-based approach using LSA for PI and SS tasks for social media text.

4. We combine TLSA with sentiment analysis, which outperforms the state-of-the-art unsupervised and semi-supervised methods in SEMEVAL 2015: Paraphrase and Semantic Similarity in Twitter tasks.

2 Related Work

2.1 Corpus-Based algorithms

Corpus-based machine learning algorithms have an advantage over knowledge-based ones as they do not involve in human which can be expensive. The Google web 1T n-gram corpus (Brants and Franz, 2006) included all words appearing on the web in January, 2006 and is available in English and 10 European Languages (Brants and Franz, 2009). This corpus has been used for text relatedness (Islam et al., 2012) and linguistic steganography (Chang and Clark, 2010). The WaCky corpus of more than one billion

[1]Reddit n-gram temporal corpus - https://web.cs.dal.ca/~anh/?page_id=1699

[2]Topic-based Latent Semantic Analysis - http://cgm6.research.cs.dal.ca:8080/RedditFileDownload/tlsa.html

words from three languages, English, German, and Italian was introduced in 2009 by (Baroni et al., 2009). It has been used in bilingual lexicography (Ferraresi et al., 2010) and translators (Pecina et al., 2012). In 2010, Microsoft Web n-gram corpus provided all the word n-grams that are indexed by Bing search engine and provided through an XML web service (Wang et al., 2010b). Some notable usage includes textbox enriching (Agrawal et al., 2010) and social media language study (Liu et al., 2012). Google Book n-gram corpus (Michel et al., 2011), introduced in 2012, includes all word n-grams found in Google book corpus from 1505 to 2008. Due to its yearly temporal characteristics, it has been used to study the changing psychology of culture (Greenfield, 2013), concepts of happiness (Oishi et al., 2013), and mapping book to time (Islam et al., 2015). Twitter n-gram corpus (Herdağdelen and Baroni, 2011) only provides a small subset of social media n-grams in Twitter. As Twitter does not allow researchers to share full text of Tweets as a large corpus, it is not possible to collect, create and share terabyte-scale n-gram corpus for Tweets. Unlike Twitter, Reddit implements an open data policy and users can query any posted data on the website. Although OSNs have been studied intensively in recent years, there is no existing corpus that could be shared and provide insights from massive social network text. To the best of our knowledge, this new corpus is the first large-scale n-gram corpus that provides n-grams with a temporal feature (monthly) that is designed specifically for massive user-generated social media text.

2.2 Paraphrase Identification and Semantic Similarity

A summary of all the existing state-of-the-art paraphrase identification algorithms for traditional texts (e.g., newswire) using the Microsoft Research Paraphrase Corpus (MRPC) is in (Par, 2016). Although supervised approaches, such as typical machine learning classifiers using various feature sets (Das and Smith, 2009; Ji and Eisenstein, 2013) and semantic text similarity (Blacoe and Lapata, 2012; Madnani et al., 2012), achieved the best results, unsupervised methods using explicit semantic space (Hassan and Mihalcea, 2011), vector-based similarity (Milajevs et al., 2014), and WordNet similarity with matrix (Fernando and Stevenson, 2008) also attained comparable results. With the increasing popularity of OSNs, researchers started to focus on the importance of developing paraphrase identification for social media text (Zanzotto et al., 2011; Xu et al., 2013; Guo and Diab, 2012). The results and findings support the hypothesis that informal language in social media with a high degree of lexical variations has posed serious challenges to both tasks. In this paper, our focus is not the general PI or SS tasks but concentrates on the domain of social media.

The SemEval-2015 task 1 is the first competition that focuses on Paraphrase Identification and Semantic Similarity for social media text. There were 19 and 14 teams that participated in the PI and SS tasks, respectively. Most teams used supervised approach, for example, typical machine learning classifiers (Eyecioglu and Keller, 2015), neural networks (Xu et al., 2015), align and penalize architecture, semantic relatedness (van der Goot and van Noord, 2015). Two teams used unsupervised approaches (Orthogonal Matrix Factorization (Guo et al., 2014) and pre-trained word and phrase vectors on Google News dataset (Xu et al., 2015)) and one team uses semi-supervised approach that combines several word measures built from Rovereto Twitter n-gram corpus (Herdağdelen and Baroni, 2011). Our proposed approach will be compared and evaluated against these unsupervised and semi-supervised approaches.

Lately, large corpora are being used for the machine learning tasks. LSA has been widely used for paraphrase identification and semantic text analysis (Hassan and Mihalcea, 2011). (Guo and Diab, 2012) proposed Weighted Textual Matrix Factorization (WTMF), which is a novel latent model that captures the contextual meanings of words in sentences based on internal term-sentence matrix. This model uses both knowledge-based and large-scale corpus-based techniques to learn word representation. Our work uses the new corpus and introduces a novel approach to learn word representation that is dependent on the topic being discussed.

3 The New Reddit Temporal N-gram Corpus

We have created a word n-gram (1-5) corpus of 1.65 billion Reddit comments from October, 2007 until August, 2016 (Baumgartner, July 2015) using high performance distributed processing models on a cluster of 256 nodes with 16TB of shared memory. Most of the comments are in the English language.

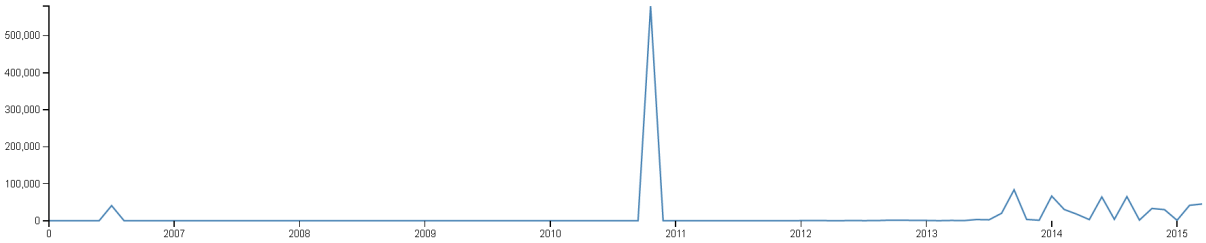


Figure 1: The frequency count of unigram "ISIS" in Reddit from 2007 to 2016. The x-axis represents the year while the y-axis shows the frequency count per month. The highest peak in the graph represents the rise of "ISIS" in October, 2010 following the outbreak of the Syrian Civil War in August, 2010.

Each comment is separated into sentences, and each sentence is tokenized using Lucence (Apache). All the comments are lowercased. The Reddit comments are close to 2TB of text containing 135 billion sentences.

Each entry in the Reddit temporal n-gram corpus is an n-gram (1-5) and its frequency, month, and year from October, 2007 to August, 2016. The size of the corpus is 2.6 TB uncompressed. We show an example of each n-gram (1-5) in Table 1. This corpus can be accessed through downloadable files and a JSON web service which will return the frequency of an n-gram for each month. In addition, we implement a cloud-based visualization interface so that end users can query and analyze the social media n-grams in real time as shown in Figure 1. As our n-gram corpus is time-dependent, we also count the total number of occurrences of each n-gram for comparison with other corpora. A detailed statistical comparison between the new corpus and some other existing corpora is shown in Table 2. Although Microsoft Web n-gram is the largest n-gram corpus, they do not provide all the data to end users. Analyzing the whole Microsoft n-grams corpus is not practical through an XML web service. The Reddit temporal n-gram corpus is much larger than the Google Web 1T n-gram corpus. One of the reasons is that Google Web 1T n-gram corpus only keeps unigrams with more than 200 frequency counts and other n-grams with more than 40 frequency counts. After analyzing the Google Web corpus, we found that although our corpus has a larger vocabulary than the Google one, the frequency count for each n-gram is lower. This confirms the noise and shortness hypothesis of social media text. We decide to keep all the raw n-grams of the new corpus to preserve these characteristics of social media text. We illustrate in Figure 2 how the Reddit temporal n-gram corpus shows the evolution of the word "ISIS".

(a) (b)

Figure 2: An example of word cloud showing the context words of "ISIS" in the 5-grams of the corpus before and after August, 2010. a) "ISIS" is mainly discussed as an Egyptian god before August, 2010, b) "ISIS" means the Islamic State in Iraq and Syria after August, 2010.

Table 1: Examples of n-grams (1-5) of the newly created corpus about the topic "Donald Trump". Each entry includes the word (n-gram), its frequency, its month, and its year from October 2007 to August 2016.

Reddit temporal n-gram corpus	word	frequency	year	month
1-gram	trump	981	2015	01
2-gram	trump apprentice	31	2015	01
3-gram	donald trump battle	16	2015	01
4-gram	donald trump ignorant tweet	8	2015	01
5-gram	take donald trump advice in	2	2015	01

Table 2: Statistical comparison between Reddit temporal n-gram corpus and its counterparts.

Corpus	1-gram	2-gram	3-gram	4-gram	5-gram
Google web 1T n-gram corpus	13.5M	314M	977M	1.3B	1.12B
Microsoft web n-gram corpus	1.2B	11.7B	60.1B	148.5B	237B
Reddit temporal n-gram corpus	170.2M	1.2B	6.7B	18.4B	30.1B

4 Topic-based Latent Semantic Analysis

We first formulate the approach for TLSA. Consider a list of topics $T = \{T_1, T_2, ..., T_l\}$ and each topic T_i has a list of pairs of Tweets $P = \{(t_{11}, t_{12}), (t_{21}, t_{22}), ..., (t_{m1}, t_{m2})\}$ where each pair is evaluated for PI and SS tasks. For each topic T_i, we construct a list of unigrams $O = \{o_1, o_2, ..., o_p\}$ from P and a list of 5-grams $F = \{f_1, f_2, ..., f_q\}$ from Reddit temporal n-gram corpus where f_i contains the topic T_i. Next, we construct the unigram/5-gram matrix X from O and F.

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1q} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2q} \\ \dots & \dots & \dots & \dots & \dots \\ x_{p1} & x_{p2} & x_{p3} & \dots & x_{pq} \end{bmatrix}$$

where each row r_i represents the occurrence of a unigram term u_i to all 5-grams in F and x_{ij} describes the occurrence of unigram o_i in a 5-gram f_j plus the frequency of the 5-gram f_j in the Reddit temporal n-gram corpus. This matrix considers both the relation between a word with other words in a 5-gram and with the frequency of this 5-gram in the corpus. Next, we decompose matrix X using a Singular Value Decomposition (SVD):

$$X = U\Sigma V^T$$

$$= \begin{pmatrix} u_1 & \cdots & u_r & u_{r+1} & \cdots & u_p \\ | & & | & | & & | \\ | & & | & | & & | \\ \underbrace{\qquad\qquad}_{\text{col}(A)} & & & \underbrace{\qquad\qquad}_{\text{null}(A)} & & \end{pmatrix} \begin{pmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix} \begin{pmatrix} \text{---} & v_1^T & \text{---} \\ & \cdots & \\ \text{---} & v_r^T & \text{---} \\ \text{---} & v_{r+1}^T & \text{---} \\ & \cdots & \\ \text{---} & v_q^T & \text{---} \end{pmatrix} \begin{matrix} \\ \\ \} \text{row}(A) \\ \\ \\ \} \text{null}(A) \\ \\ \end{matrix}$$

where Σ is a diagonal matrix that contains the singular values in descending values. U and V are orthogonal matrices that contain the left and right singular vectors respectively.

Next, for each sentence s_i in topic T_i, we construct a vector $\vec{v}$ which represents the occurrence of s_i in the list of unigrams of topic T_i. This vector is translated into a sentence vector representation by the following formula:

$$\vec{v} = \vec{v} * U_k * S_k$$

where k is the chosen k singular values which show the dimensions with the greatest variance between words and documents (the value of k is explained in Section 6.3). Finally, the semantic similarity between two sentences is calculated using the cosine similarity between their vectors.

Due to the enormous size of the Reddit temporal n-gram corpus, selecting the related 5-grams for each topic is not feasible using a traditional relational database system. We tried to load our data into IBM Netezza data warehouse but the query time was not reasonable for a real-time system. We load all the corpus data to Google Bigquery. For each topic T_i, we query all the related 5-grams f_i using Google Bigquery regular expression "word like $(\%T_i\%)$" where $\%$ represents the wild card search. After constructing matrix X, we use Microsoft Azure Apache Spark for SVD decomposition. A summary of the proposed approach is shown in Figure 3.

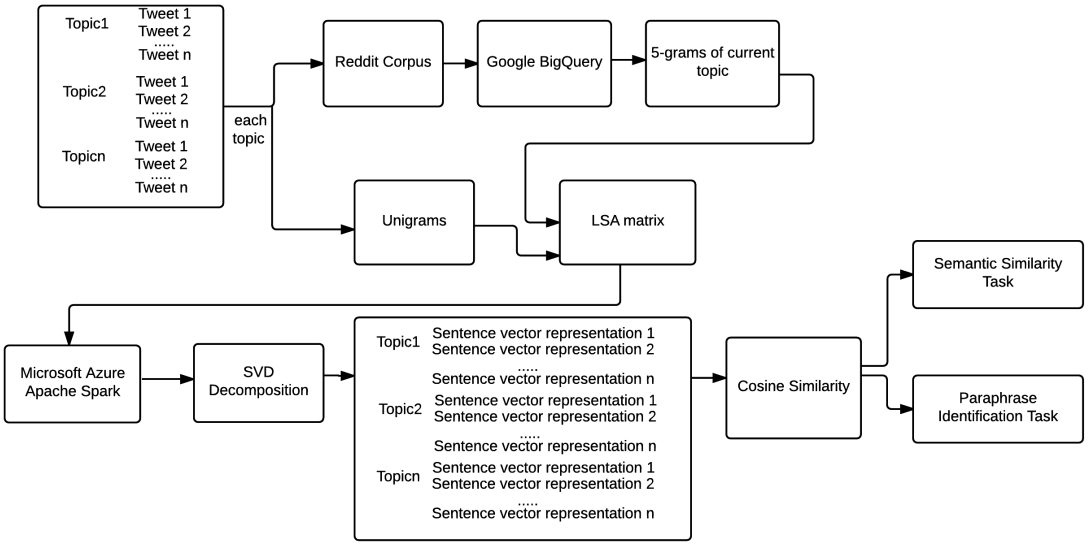


Figure 3: The proposed Topic-based Latent Semantic Analysis using distributed parallel computing, Google BigQuery, and Microsoft Azure Apache Spark. The semantic similarity between two sentences is computed with regard to a specific topic being discussed in two sentences.

5 Evaluations of Paraphrase Identification and Semantic Similarity for Social Media Text

To evaluate the performance of TLSA algorithm, we use the PIT-2015 Twitter dataset (Xu et al., 2014). Although this approach uses PIT-2015 dataset for evaluation, it can be extended to any general topic-based datasets. The PIT-2015 dataset includes 17,790 sentence pairs for training and 972 test sentence pairs which were annotated and developed by (Xu et al., 2014). The dataset was constructed from Twitter data and has intrinsic characteristics from social network data: (i) opinionated and colloquial sentences from realistic social media text; (ii) lexically diverse pairs of sentences for paraphrases; and (iii) sentences that seem lexically similar but semantically dissimilar (Xu et al., 2015). Example pairs of sentences for paraphrase, non-paraphrase, and debatable cases are shown in Table 3. The detailed statistics of this ground-truth dataset is shown in Table 4. Each sentence is processed with tokenization, part-of-speech and named entity tags and each sentence pair is annotated by experts. In the test set, there

Table 3: Examples of Paraphrase Identification and Semantic Similarity sentence pairs. All three sentence pairs are about the movie "8 Mile" which is a topic for TLSA. A sentence pair is a paraphrase if its Pearson Correlation score is above 0.6. A sentence pair is a non-paraphrase if its Pearson Correlation score is below 0.6. A sentence pair is debatable if its Pearson Correlation score is equal to 0.6.

Topic	Paraphase	Sentence 1	Sentence 2
8 mile	True	The Ending to 8 Mile is my fav part of the whole movie	Those last 3 battles in 8 Mile are THE shit
8 mile	False	All the home alones watching 8 mile	The last rap battle in 8 Mile nevr gets old ahah
8 mile	Debatable	8 mile is just a classic	After watching 8 mile I feel like such a thug

are 972 sentence pairs collected from Twitter in 20 trending topics between May 13th and June 10th, 2013. As mentioned in (Das and Smith, 2009), some algorithms may work well specifically for MRPC because of its imbalanced nature (lack of non-paraphrases). PIT-2015 Twitter dataset is more balanced as it contains 70% non-paraphrases and the 34% paraphrases.

Table 4: PIT-2015 Twitter dataset. The test data is more balanced than MRPC as it has a higher percentage of non-paraphrase sentence pairs. The unsupervised TLSA only uses the test data for evaluation.

	Sent Pairs	Paraphase	Non-paraphrase	Debatable
Train	13063	3996 (30.6%)	7534 (57.7%)	1533 (11.7%)
Test	972	175 (18.0%)	663 (68.2%)	134 (13.8%)

5.1 Task 1 - Paraphrase Identification and Evaluation Metrics

For a specific topic, given two sentences, the system has to determine if two sentences have the same or similar meaning and discuss the same topic. For two non-paraphrase sentence pairs, the sentence pair discussing the same topic has a higher score than the sentence pair discussing an unrelated topic. Precision, recall, and F1 (harmonic mean of precision and recall) are used as evaluation metrics.

5.2 Task 2 - Semantic Similarity and Evaluation Metrics

For a specific topic, given two sentences, the system has to give a score between 0 (no relation) and 1 (semantic equivalence) to represent their semantic equivalence. For two sentence pairs, the sentence pair discussing the same topic has a higher semantic similarity score than the sentence pair discussing an unrelated topic. Pearson correlation is used as an evaluation metric.

6 Evaluation

6.1 Baselines

We used first two baselines from (Xu et al., 2015) and introduced two new baselines that are more related to the proposed corpus-based and topic-based LSA.

Random: Each sentence pair is assigned a random real semantic similarity score between [0, 1]. For PI task, this baseline applies 0.5 as a cutoff (paraphrase if semantic similarity score is above 0.5).

Weighted Matrix Factorization (WTMF): This baseline uses the state-of-the-art unsupervised method of (Guo and Diab, 2012). It not only considers the semantic space of words presenting in the data but also missing words from the sentences. This feature is designed specifically for short texts in social media. Finally, the value 0.5 is used as a cutoff for the PI task.

Random 5-gram: This baseline determines whether introducing the use of topics in LSA improves the accuracy of both PI and SS tasks for SEMEVAL 2015. To construct matrix X, we select random

5-grams from the Reddit temporal n-gram corpus with the same size of the 5-grams that contain the topic.

Google Tri-gram Method (GTM): Google Tri-gram Method (Islam et al., 2012) assigns a semantic similarity score between two sentences using the unigrams and trigrams of the Google Web 1T corpus. We also use 0.5 as a cutoff for the PI task.

6.2 SEMEVAL 2015 Unsupervised and Semi-supervised Methods

Columbia: This method used Orthogonal Matrix Factorization to compute a representation vector for each sentence (Guo et al., 2014) and then computes a similarity score based on these vectors (Xu et al., 2015).

Yamraj: This method learned sentence vectors from Google News dataset (about 100 billion words) and Wikipedia articles. Cosine distance is used to compute the vector similarity scores.

MathLingBp: This method exploits the use of the align-and-penalize architecture of (Han et al., 2013) and adopts the use of several word similarity metrics using a semi-supervised approach (Xu et al., 2015).

6.3 Experimental Results

First, we compare the performance of TLSA with various parameters, such as the number of singular values and the dimensionality of the 5-grams. For SS task, we achieved the best result for SS task when the singular value k is equal to 80 with an increasing 5-gram dimensionality size as shown in Figure 4. In addition, for SS task, the Pearson correlation is not improving when the number of 5-grams is above 1M.

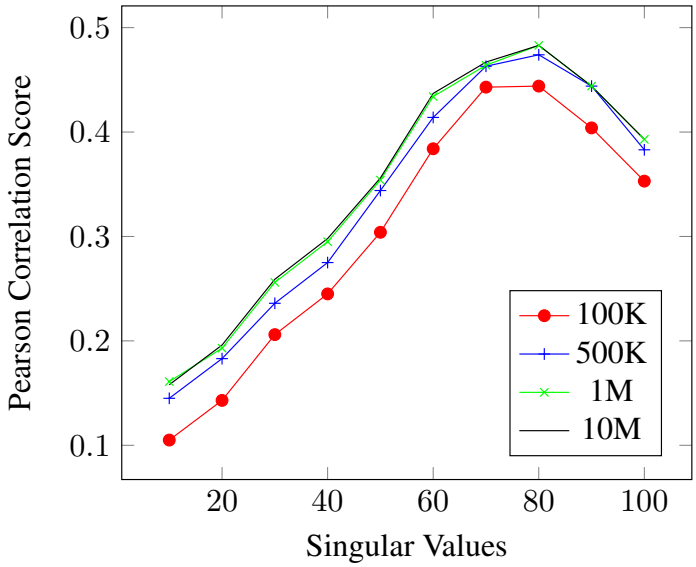


Figure 4: For SS task, Pearson correlation score with an increasing singular values and 5-gram dimensionality. TLSA achieves the best Pearson correlation score for k = 80 and the dimensionality of 5-grams = 1M.

6.3.1 Topic-based LSA versus Baselines and other Methods

This section compares the proposed approach with the baselines and SEMEVAL 2015 unsupervised and semi-supervised methods. As shown in Table 5, TLSA achieved the best result for the SS task (Pearson correlation) compared with all the baselines and compared methods. This means that training an LSA model using topic-based 5-gram helps increase the result of PI and SS tasks. For the PI task, observing that the semantic similarity scores for sentence pairs are either very high or very low, we tried two cutoffs 0.25 and 0.5 (SEMEVAL 2015 allows two runs per team) and TLSA outperforms all the baselines. With a low cutoff value, TLSA achieves a high precision and a low recall. To improve the PI results, we assumed that two sentences are paraphrases only if they have the same sentiment scores

(e.g., both are positives or negatives). Based on this assumption, each sentence is assigned a sentiment score using OpenNLP. Adding sentiment analysis to TLSA (i.e., TLSA & Sentiment) outperforms all the baselines and compared methods. Another important observation is that although our unsupervised approach achieves the best results against the baselines and compared methods, its results are still not comparable with human upperbound. This means that improving the results of PI and SS tasks for social media text using an unsupervised approach is still a challenge for researchers.

Table 5: TLSA results with other baselines and compared methods. Combining TLSA with sentiment analysis achieves the best result for both PI and SS tasks.

Methods / *Baselines*	Paraphrase Identification			Semantic Similarity			
	F1	Precision	Recall	Pearson	maxF1	maxPrec	maxRecall
Human Upperbound	0.823	0.752	0.0908	0.735	–	–	–
TLSA & Sentiment	**0.591**	0.764	0.480	**0.483**	0.582	0.761	0.472
COLUMBIA	0.588	0.593	0.583	0.425	0.599	0.623	0.577
TLSA	**0.585**	0.761	0.474	**0.483**	0.585	0.761	0.474
YAMRAJ	0.496	0.725	0.377	0.360	0.542	0.502	0.589
WTMF	0.536	0.450	0.663	0.350	0.587	0.570	0.606
Random 5-gram	0.504	0.716	0.389	0.466	0.564	0.824	0.429
GTM	0.495	0.391	0.674	0.371	0.582	0.761	0.472
Random	0.266	0.192	0.434	0.017	0.350	0.215	0.949

7 Conclusions

In this paper, we introduced Reddit temporal n-gram corpus, which is designed specifically for social media text. We create the corpus using distributed parallel computing and implement a cloud-based visualization interface so that end users can query any n-grams from the corpus. Both the corpus and the interface are publicly available in this URL - Reddit n-gram temporal corpus. This large-scale terabyte corpus includes all the word unigram to 5-gram, and their frequency per month from October, 2007 to August, 2016.

To show the usefulness of this corpus, we propose a novel Topic-based Latent Semantic Analysis approach which exploits the 5-grams of the corpus. The proposed TLSA outperforms all the state-of-the-art unsupervised and semi-supervised methods in SEMEVAL 2015 Task 1 - Semantic Similarity for the PIT-2015 dataset. Combining with sentiment analysis, the proposed approach also achieves the best result for the Paraphrase Identification of SEMEVAL 2015 Task 1. In addition, TLSA is language-independent and scalable for the large-scale nature of social media text.

For future work, we aim to use this corpus to study the linguistic patterns of social media text, for example, finding the meaning of new words in social media. In addition, we plan to integrate this proposed semantic similarity score into our existing work to improve the results of meme clustering tasks (Dang et al., 2015b) and rumour detection and visualization framework (Dang et al., 2016).

Acknowledgment

The research was funded in part by CNPq 487186/2013-3, FAPESP 2014/09599-5 (Brazil), Natural Sciences and Engineering Research Council of Canada, and International Development Research Centre, Ottawa, Canada.

References

Rakesh Agrawal, Sreenivas Gollapudi, Krishnaram Kenthapadi, Nitish Srivastava, and Raja Velu. 2010. Enriching textbooks through data mining. In *Proc. of the First ACM Symposium on Computing for Development*, page 19. ACM.

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

Alberto Barrón-Cedeño, Marta Vila, M Antònia Martí, and Paolo Rosso. 2013. Plagiarism meets paraphrasing: Insights for the next generation in automatic plagiarism detection. *Computational Linguistics*, 39(4):917–947.

Jason Baumgartner. July, 2015. Complete public reddit comments corpus. Available: https://archive.org/details/2015_reddit_comments_corpus [Accessed: April 13, 2016].

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proc. of the 2012 Joint Conference on EMNLP and Computational Natural Language Learning*, pages 546–556. ACL.

Thorsten Brants and Alex Franz. 2006. {Web 1T 5-gram Version 1}.

Thorsten Brants and Alex Franz. 2009. Web 1t 5-gram, 10 european languages version 1. *Linguistic Data Consortium, Philadelphia*.

Ching-Yun Chang and Stephen Clark. 2010. Linguistic steganography using automatically generated paraphrases. In *HLT- NAACL*, pages 591–599.

Charles LA Clarke, Gordon V Cormack, M Laszlo, Thomas R Lynam, and Egidio L Terra. 2002. The impact of corpus size on question answering performance. In *ACM SIGIR*, pages 369–370. ACM.

Anh Dang, Raheleh Makki, Abidalrahman Moh'd, Aminul Islam, Vlado Keselj, and Evangelos E Milios. 2015a. Real time filtering of tweets using wikipedia concepts and google tri-gram semantic relatedness. In *TREC*.

Anh Dang, Abidalrahman Moh'd, Anatoliy Gruzd, Evangelos Milios, and Rosane Minghim. 2015b. A visual framework for clustering memes in social media. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, ASONAM '15, pages 713–720, New York, NY, USA. ACM.

Anh Dang, Abidalrahman Moh'd, Evangelos Milios, and Rosane Minghim. 2016. What is in a rumour: Combined visual analysis of rumour flow and user activity. In *Proceedings of the 33rd Computer Graphics International*, pages 17–20. ACM.

Dipanjan Das and Noah A Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proc. of the Joint Conference of the 47th Annual Meeting of the ACL and AFNLP: Volume 1-Volume 1*, pages 468–476. ACL.

Asli Eyecioglu and Bill Keller. 2015. Asobek: Twitter paraphrase identification with simple overlap features and svms. *SemEval*.

Samuel Fernando and Mark Stevenson. 2008. A semantic similarity approach to paraphrase detection. In *Proc. of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*, pages 45–52. Citeseer.

Adriano Ferraresi, Silvia Bernardini, Giovanni Picci, and Marco Baroni. 2010. Web corpora for bilingual lexicography: a pilot study of english/french collocation extraction and translation. *Using Corpora in Contrastive and Translation Studies. Newcastle: Cambridge Scholars Publishing*, pages 337–362.

Patricia M Greenfield. 2013. The changing psychology of culture from 1800 through 2000. *Psychological Science*, 24(9):1722–1731.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proc. of the 50th Annual Meeting of the ACL: Long Papers-Volume 1*, pages 864–872. ACL.

Weiwei Guo, Wei Liu, and Mona T Diab. 2014. Fast tweet retrieval with compact binary codes. In *COLING*, pages 486–496. Citeseer.

Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. Umbc ebiquity-core: Semantic textual similarity systems. In *Proc. of the Second Joint Conference on Lexical and Computational Semantics*, volume 1, pages 44–52.

Samer Hassan and Rada Mihalcea. 2011. Semantic relatedness using salient semantic analysis. In *AAAI*.

Amaç Herdağdelen and Marco Baroni. 2011. Stereotypical gender actions can be extracted from web text. *Journal of the American Society for Information Science and Technology*, 62(9):1741–1749.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Proc. of the Fifteenth conference on Uncertainty in Artificial Intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc.

Aminul Islam, Evangelos Milios, and Vlado Kešelj. 2012. Text similarity using google tri-grams. In *Advances in Artificial Intelligence*, pages 312–317. Springer.

Aminul Islam, Jie Mei, Evangelos E Milios, and Vlado Kešelj. 2015. When was macbeth written? mapping book to time. In *Computational Linguistics and Intelligent Text Processing*, pages 73–84. Springer.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *EMNLP*, pages 891–896.

Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is twitter, a social network or a news media? In *WWW*, pages 591–600. ACM.

Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse Processes*, 25(2-3):259–284.

Chen Li and Yang Liu. 2015. Improving named entity recognition in tweets via detecting non-standard words. In *ACL*.

Fei Liu, Fuliang Weng, and Xiao Jiang. 2012. A broad-coverage normalization system for social media language. In *ACL*, pages 1035–1044.

Nitin Madnani, Necip Fazil Ayan, Philip Resnik, and Bonnie J Dorr. 2007. Using paraphrases for parameter tuning in statistical machine translation. In *Proc. of the Second Workshop on Statistical Machine Translation*, pages 120–127. ACL.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proc. of the 2012 Conference of the NAACL: Human Language Technologies*, pages 182–190. ACL.

Jean-Baptiste Michel, Yuan Kui Shen, et al. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.

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

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. *ArXiv Preprint ArXiv:1408.6179*.

Peter Norvig. 2008. Statistical learning as the ultimate agile development tool. In *CIKM*.

Shigehiro Oishi, Jesse Graham, Selin Kesebir, and Iolanda Costa Galinha. 2013. Concepts of happiness across time and cultures. *Personality and Social Psychology Bulletin*, 39(5):559–577.

Par. 2016. Paraphrase identification (state of the art) @ONLINE. Available: http://aclweb.org/aclwiki/index.php?title=Paraphrase_Identification_(State_of_the_art) [Accessed: July 15, 2016].

Pavel Pecina, Antonio Toral, et al. 2012. Domain adaptation of statistical machine translation using web-crawled resources: a case study. In *EAMT*, pages 145–152.

Rob van der Goot and Gertjan van Noord. 2015. Rob: Using semantic meaning to recognize paraphrases. *SemEval*.

Kuansan Wang, Xiaolong Li, and Jianfeng Gao. 2010a. Multi-style language model for web scale information retrieval. In *ACM SIGIR*, pages 467–474. ACM.

Kuansan Wang, Christopher Thrasher, Evelyne Viegas, Xiaolong Li, and Bo-june Paul Hsu. 2010b. An overview of microsoft web n-gram corpus and applications. In *NAACL HLT 2010 Demonstration Session*, pages 45–48.

Wei Xu, Alan Ritter, and Ralph Grishman. 2013. Gathering and generating paraphrases from twitter with application to normalization. In *Proc. of the Sixth Workshop on Building and Using Comparable Corpora*, pages 121–128. Citeseer.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from twitter. *Transactions of the ACL*, 2:435–448.

Wei Xu, Chris Callison-Burch, and William B Dolan. 2015. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). *SemEval*.

Fabio Massimo Zanzotto, Marco Pennacchiotti, and Kostas Tsioutsiouliklis. 2011. Linguistic redundancy in twitter. In *EMNLP*, pages 659–669. ACL.

Yating Zhang, Adam Jatowt, Sourav S Bhowmick, and Katsumi Tanaka. 2015. Omnia mutantur, nihil interit: Connecting past with present by find-ing corresponding terms across time. In *ACL*, pages 645–655.

Geoffrey Zweig and Christopher JC Burges. 2011. The microsoft research sentence completion challenge. Technical report, Technical Report MSR-TR-2011-129, Microsoft.

Dictionaries as Networks: Identifying the graph structure of Ogden's Basic English

Camilo Garrido and **Claudio Gutierrez**
Center for Semantic Web Research & Department of Computer Science,
Universidad de Chile
{cgarrido, cgutierr}@dcc.uchile.cl

Abstract

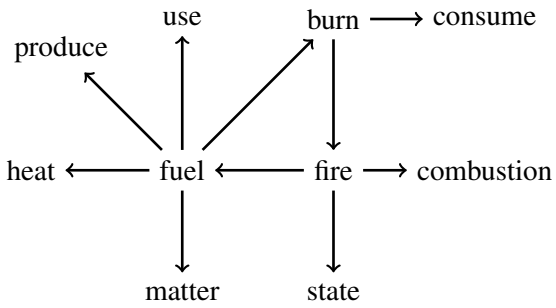
We study the network structure underlying dictionaries. We systematize the properties of such networks and show their relevance for linguistics. As case of study, we apply this technique to identify the graph structure of Ogden's Basic English. We show that it constitutes a strong core of the English language network and that classic centrality measures fail to capture this set of words.

1 Introduction

Dictionaries are rich sources to investigate the semantic structure of natural language. The purpose of dictionaries, writes Wilks et al. (1993), "is to provide definitions of senses of words and, in so doing, supply knowledge about not just language, but the world." The definition of a word involves recursively new words, and thus, new senses and meanings. In this way, a dictionary can naturally be viewed as a network where each word w is related to the set of words $w_1, \ldots, w_n$ that define it: for each word w so defined, consider the relationship $w \to w_i$ for each $i = 1, \ldots, n$, and proceed recursively with all the entries of the dictionary. The idea is not new and was already proposed by K. C. Litkowski (1978).

Clearly this basic idea must be refined. There are words that are in inflected form (e.g. verbs); that are the same but have different meanings (e.g. singer: the machine, the musician, etc.); that are in plural or singular; that are the same adjective with different gender, etc. In order to make the network conceptually coherent, one should define classes of words; for example, all the inflected forms of the verb "play" define one class whose representative is the word "play". There are several other simple processing decisions to be made. This naive version can be further refined by incorporating more elaborated linguistic features, like labeling the edges according to parts of speech to which they point, *e.g.* nouns, verbs, adjectives, adverbs, etc., or giving different nodes and weights to different meanings of a word, and so on. The surprising fact is that even using a naive approach, the network obtained gives highly relevant and interesting information about the language. See Figure 1 for a small example.

Although the idea of using mathematical and computational tools to capture the semantics information in dictionaries has been broadly explored (Amsler, 1980; Calzolari, 1984; Wilks et al., 1988), the idea



FIRE: Fuel in a state of combustion.

FUEL: Any matter used to produce heat by burning.

BURN: To consume with fire.

Figure 1: Entries in the dictionary for the words *burn*, *fire*, and *fuel*, and their corresponding subgraph built from them.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3565–3576, Osaka, Japan, December 11-17 2016.

of exploiting the inherent network structure of dictionaries has not been pursued systematically. As we mentioned, the idea was proposed several decades ago (Litkowski, 1978), but only recently, with the explosion of network studies and hardware availability, there have been some works in this direction (we discuss them in the Related Work section).

The aim of this paper is to provide evidence of the fruitfulness of studying dictionaries as networks. We show that dictionaries (in general) have similar structure from the point of view of networks, and as expected, their structural properties differ from networks obtained from other areas. We claim that dictionary networks have particular properties (strong connectivity, resilience, component analysis, etc.) that shed light on the structure of the languages and deserve to be studied in depth. We found out that classical tools for studying and analyzing networks –particularly those popularized by Social Network Analysis (Wasserman and Faust, 1994)– like centrality measures, subgroups, affiliation, etc. are not always meaningful in this new realm, and that their successful application to this linguistic setting requires to be reworked. For example, it is not evident that they help capturing notions such as "most relevant" or "non-relevant" words in a dictionary that are important, for example, for building small dictionaries, basic sets of words for beginners, etc.

In order to test these and other ideas in practice we chose as a study case *Ogden's Basic English*, a set of 850 words selected by the linguist C. K. Ogden to serve as a basic language (Ogden, 1930) . In order to study it from a network point of view, we built a network out of an English dictionary. We chose the *Online Plain Text English* (OPTED) because it is reliable, contains 94.5% of Ogden's words, and is open data, thus, allowing anyone to replicate our experiments. We then applied different graph-theoretical notions and techniques to this network, aiming to capture Ogden's set of words.

Our study shows, using *only* graph-theoretical tools, that Ogden's set of words is part of a strongly connected core of the English dictionary, a subset of words that directly use each other in their definitions. We then show that it is not formed by the "most central" words (according to classic ranking measures), but by a combination of high ranked words plus others that play the role of "covering" the rest of the network, that is, being "close" to most words in the dictionary.

The main contribution of our work is to add evidence about the value of using dictionary networks to study linguistic properties of languages.

2 Related Work

The community agrees that dictionaries are a source of lexical knowledge (Calzolari, 1984; Dolan et al., 1993). This knowledge can be used for the development of NLP techniques, establishing usage relation between words or hierarchy relations like hypernyms or "part_of". They also can be used for the creation of pocket dictionaries and many other applications.

One of the first uses of dictionaries was to develop Machine-Readable Dictionaries (Zingarelli, 1970) (MRDs). With MRDs and the concept of lexical databases, the importance of the information and knowledge that dictionaries contain began to gain attention. Amsler(1980) presents some efforts to exploit dictionaries and extract information for applications in computational linguistics. He investigates the possibility of building of taxonomies based on the structure of the definition of words. He also offers some insight on the frequency of the vocabulary and semantic ambiguity. Calzolari (1984) detects some patterns among lexical entries: hyponyms and restriction relations. Later, Calzolari et al. (1988) focused their efforts on extracting semantic information from dictionaries. They state "The dictionary is now considered as a primary source not only of lexical knowledge but also of basic general knowledge". They parse the entries and try to organize the knowledge with functions like Hypernym, Relation, Qualifier, etc. Wilks et al.(1988) discuss the importance of dictionaries for NLP tasks, in particular, the value of transforming machine readable dictionaries (MRDs) into machine tractable dictionaries (MTDs). They show three approaches for this: Obtaining and using co-occurrence statistics, producing a lexicon and extracting a Key Defining vocabulary.

At the same time MRDs began to get attention, so did modeling the dictionaries as networks. K. C. Litkowski (1978) was one of the first to state the importance of studying and exploiting dictionary networks, both as sources of material for natural language and to unravel the complexities of meaning.

He presented three models to represent a dictionary. The first model uses nodes to represent words and edges to represent the relation w_a *"is used to define"* w_b. The second model extends the first one, adding relations between words and senses. In the third and final model, nodes represent concepts and edges represent different relations between them (senses, part of sentence, etc). Definitions are broken down into subphrases. For example, "Broadcast: the act of spreading abroad" may be broken into "the act", "of", "spreading abroad", where these subphrases may be broken into smaller pieces.

After the seminal this work of Litkowski (1978), several researchers have used dictionary networks to study or extract information about the language.

Dolan el al.(1993) developed an automatic strategy to exploit dictionaries to construct a source of lexical an common sense information based on *hypernyms*, *locations*, *part_of*, and other relations. Although a network can be formed with those relations, the methods of the system to extract such relations between words is not clear. Picard et al.(2009) address the following question: "How many words do you need to know in order to be able to learn all the rest from definitions?" They approach this question representing dictionaries as networks. Levary et al.(2012) show that if we follow the definition of a word over and over, one typically finds that definitions loop back upon themselves. They also show that the loop is an essential element of the growth process of networks. They showed that words within these loops tend to be introduced into the English language at similar times. And, the evolution of these networks follow the "rich-get-richer" growth. Mihalcea(2004a) used networks derived from WordNet to test disambiguation.

There are other forms of building networks of words and using graph ideas in word analysis, *e.g.* co-occurrence of words in certain windows (bigrams, etc.) (Dorogovtsev et al., 2001; Mihalcea, 2004b).

Finally, there is a line of research that investigates the relationship between semantic networks and graph measures. Abbott et al. (2012) compare the functioning of the human mind when searching for memories with a random walk in a semantic network. They conclude results that can help clarify the possible mechanisms that could account for PageRank predicting the prominence of words in semantic memory. Yeh et al. (2009) used random walks to determine the semantic relatedness between two elements. They conclude that random walks performed with personalized PageRank is a feasible and potentially fruitful means of computing semantic relatedness for words and texts. Hughes et al. (2007) introduce a new measure of lexical relatedness based on the divergence of the stationary distributions computed from random walks over graphs extracted from WordNet. Steyvers and Tenenbaum (2005) conjecture about semantic networks stating that "these structures reflect the mechanisms by which semantic networks grow." All of these works served as sources of inspiration for our research.

3 Dictionaries as Networks

> "Ordinary dictionaries have not been given their due, either as sources of material for natural language understanding systems or as corpora that can be used to unravel the complexities of meaning and how it is represented. If either of these goals are ever to be achieved, I believe that investigators must develop methods for extracting the semantic content of dictionaries (or at least for transforming it into a more useful form). [...] A suitable framework appears to be provided by the theory of labeled directed graphs (digraphs)." (Litkowski, 1978).

If words are viewed as basic building blocks of more complex meaning structures, the network of their relationships can be considered as the skeleton that holds them together. Dictionaries are one of the primary sources to obtain such skeletons of meaning.

A network (or graph: both used synonymously) is defined by the nature of its nodes and the of relationships that connect its nodes. A dictionary viewed as a network on the lines we explained above, gives rise to different types of nodes and edges. Nodes have types of n.; n.pl.; a.; v.; v.t.;v.i.; adv.; etc. Edges also can be of different types, according the role or the place of the word in the definition. For example, consider the following three entries of the word "act", each with a different type:

Act (n.) A formal solemn writing, expressing that something has been done.

Act (v. i.) To exert power; to produce an effect; as, the stomach acts upon food.

Act (v. t.) To perform; to execute; to do.

Also, the words occurring in these definitions play different grammatical roles, can occur more than once, etc. All of these features should be included in a faithful network of a dictionary, ideally one from which one can reconstruct the dictionary (see some insights in (Litkowski, 1978)).

On the other extreme, one can build a simple (naive) network without any typing on nodes and edges, that is, just words pointing to words represented in some standard form (e.g. lemmatized). There is a compromise between these two extreme approaches: as usual, the simpler the better (for network analysis, more tools available; for comparison with other fields, particular features do not help) at the cost of losing some subtle linguistic properties. In what follows we develop the simplest possible approach, with the idea of showing the potentialities of the method, and hoping to keep enhancing this baseline with further linguistic annotations.

3.1 Building the Basic Network

For this work we implemented the following procedure to build the networks:

1. Model or Design. Consider all types of words as a single type: forget if they were nouns, verbs, adverbs, etc. Merge the entries that correspond to the same word into one definition, e.g. *Singer (n.) A machine for sewing cloth.* and *Singer (n.) One who, or that which, singes*. Forget the role and place of occurrence of a word, as well as its number of occurrences, inside a sentence (i.e. transform the defining text of a word in a set of words).

2. Clean. Remove the terms that are inflected forms, e.g *singing: from Sing*. Remove prepositions and articles. They appear too often in any text, so they would add noise to the graph. Lemmatize each word occurring in the definitions (transform nouns into singular; verbs into the infinitive; adjectives into their male singular form). Remove any word that does not appear in the dictionary, *e.g.* prefixes and suffixes like *Ex-* and *-able*.

3. Mathematical model of the dictionary. Build the graph over the previous data. At this point, the dictionary D has become a universe of words W and a set of pairs $(w, \mathrm{def}(w))$, where $w \in W$ is an entry in D and $\mathrm{def}(w) \subseteq W$ is the set of words occurring in the definition of w.

4. Build the Network. From the data in (3), construct a directed graph $G = (V, E)$, where the nodes are $V = \{w|(w, S) \in D\}$ and the edges $E = \{(w, w')|(w, S) \in D \text{ and } w' \in \mathrm{def}(w)\}$. For example, from the entry "*Eaglet (n.) A young eagle, or a diminutive eagle.*" we get the edges *(Eaglet,young), (Eaglet, eagle)* and *(Eaglet, diminutive)*.

The OPTED dictionary. We applied the above methodology to the *The Online Plain Text English Dictionary*[1] (OPTED) and the *Diccionario de la Real Academia Española*[2] (DRAE, Royal Spanish Academy Dictionary). We chose OPTED because is a public and free-access dictionary, based on Webster's Unabridged Dictionary, and an important and recognized dictionary. On the other hand, we chose DRAE because it is the most authoritative dictionary of the Spanish language. The first edition of the DRAE was published in 1780, and the current, twenty-third edition, was published in 2014.

The OPTED network has 95,095 nodes and 979,523 edges. The nodes are composed of 58,750 nouns and 12,261 verbs. The remaining 24,084 nodes correspond to adjectives and adverbs. The RAE network has 89,767 nodes and 1,152,301 edges. The nodes are composed of 54,767 nouns and 12,046 verbs. The remaining 22,954 nodes correspond to adjectives and adverbs.

To make a good description of the dictionary network, we analyzed its different features. First, we present a set of basic properties and compare them to other kinds of networks (social, information, etc.). Second, we show a component analysis. And third, we present other characteristics obtained with graph machinery.

3.2 Dictionary Networks compared to other networks

We do the comparison with other types of networks based on classic parameters used to describe networks (Newman, 2003). Table 1 shows basic parameters for three different dictionary networks, and another three networks built by humans.[3]

[1] `http://www.mso.anu.edu.au/~ralph/OPTED/`

[2] `http://www.rae.es/`

[3] We use *igraph* `http://igraph.org/` as network analysis package and Stanford CoreNLP(2014) for lemmatizing the words in the dictionary.

	n	m	z	l	α	$c1$	$c2$	r
OPTED	95 095	979 523	20.601	4.64	2.63 / 3.13	0.009	0.217	-0.0081
DRAE network	89 767	1 152 301	25.673	3.26	2.39 / 2.74	0.044	0.201	-0.0092
WordNet	84 967	1 134 957	26.715	2.99	2.84 / 2.99	0.029	0.203	-0.0157
ca-HepPh	11 204	235 268	41.997	4.67	1.76 / 1.76	0.659	0.690	0.630
cit-HepTh	27 400	352 542	25.733	4.28	2.72 /4.14	0.120	0.329	0.002
p2p-Gnutella04	10 876	39 994	7.355	4.64	- /3.55	0.005	0.008	-0.0083

Table 1: Basic measures for networks. OPTED is an English dictionary network. DRAE is a Spanish dictionary network. WordNet is a dictionary network built from WordNet. ca-HepPh is a collaboration network from the e-print arXiv. cit-HepTh is the Citation graph from the e-print arXiv. p2p-Gnutella04 is a sequence of snapshots of the Gnutella peer-to-peer file sharing network. Details for the last three networks are in (Leskovec, 2014).

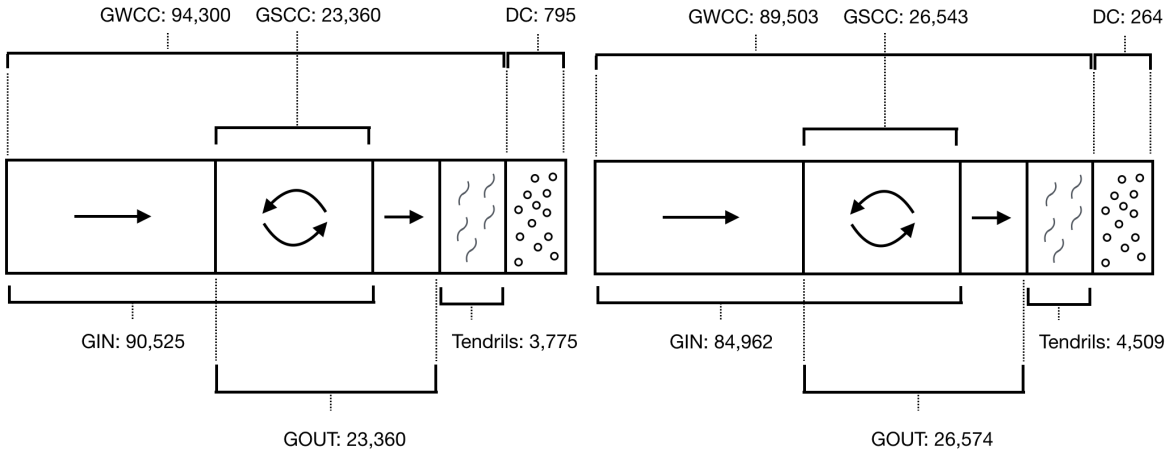
The number of nodes n tells the "size" of the network; m is the number of edges that allows for an estimation of its density, the fraction $0 \leq \frac{m}{n(n-1)} \leq 1$. Our three dictionary networks have m about 10 times n. The mean degree z gives an idea of the distribution of the edges on vertices. The mean vertex-vertex distance l tells how related/close the pairs of nodes are. The numbers in the table indicate that dictionaries have the small-world property. The parameter α refers to the exponent of the degree distribution function ($p_k \sim k^{-\alpha}$, where p_k is the fraction of the nodes that have degree k, in/out-degree) when the network (as in this case) follows this type of distribution ("power law"). It means that there are few nodes with a high degree and a large tail of low-degree nodes. The clustering coefficients $c1(= \frac{6 \times \text{number of triangles}}{\text{number of paths of length 2}})$ and $c2$ $(= \frac{1}{n}\sum_i c_i$ where $c_i = \frac{\text{number of triangles connected to vertex } i}{\text{number of triples centered on vertex } i})$ refer to the degree to which vertices tend to cluster together. In terms of network topology, the clustering coefficient refers to the presence of triangles in the network, being $c1$ a global coefficient and $c2$ a local one. In the language of social networks, the friend of your friend is likely to also be your friend. In our setting, two words having a common (non frequent) word in their definitions are likely to be related. The r coefficient indicates whether the high-degree vertices in the network associate (have links) preferentially with other high-degree vertices or not. $r = 1$ means high connectivity among them; $r = -1$ means low connectivity.

It is interesting to observe that the three dictionaries have similar parameters (as compared to other types of networks), and their properties are similar to semantic networks. Steyvers and Tenenbaum (2005) observed for the latter: "they have a small-world structure, characterized by sparse connectivity, short average path lengths between words, and strong local clustering."

Another measure is network resilience, which correlates with high connectivity. The standard measure is vertex attack tolerance VAT (Matta et al., 2014), *i.e.* behaviour of the network after removal of some nodes, defined as $\min_{S \subset V}\{\frac{|S|}{|V-S-C|+1}\}$, where C is the largest connected component in $V - S$. We determine that VAT is 0.245 for OPTED and 0.3 for DRAE. Comparing to other scale-free networks (Matta et al., 2014) (HOTNet 0.06, big barbell 0.08, star 0.11, C3 0.15, barbell 0.2, PLOD 0.25, wheel 1.0), dictionary networks are placed among the most resilient, meaning that removing some words will cause little disruption, since with high probability there will be other good relations to supply the loss.

3.3 Component Analysis

Components are classic features when describing the topology of networks. The graph is divided in two main parts: the *Giant Weakly Connected Component* (GWCC), the biggest weakly connected component present in the graph (v is connected to w if there is a undirected path from v to w), and the rest, the *Disconnected Components* (DC), that consist of separate small connected components. GWCC consists of three parts: the *Giant Strongly Connected Component* (GSCC) (strongly connected means that for each pair of nodes v, w, there is a directed path from v to w and vice versa), usually the most relevant part of the network; the *Giant in-component* (GIN), the set of nodes that have paths to GSCC (in our setting, words that in their definitions recursively use words in GSCC and are not used to define those in GSCC); and the *Giant out-component* (GOUT), the set of words that are used to define those in GSCC. Finally, the *Tendrils* are nodes which have no access to GSCC and are not reachable from it.



(a) OPTED (English, 95,095 words)

(b) DRAE (Spanish, 89,767 words)

Figure 2: Component Analysis showing similar structures for English and Spanish dictionary networks. The core part of the network (GSCC) is composed of words that are entangled –recursively use themselves in their definitions–, and amounts to approx. 25-30% of all entries in the dictionary.

3.4 Other characteristics obtained with graph machinery

One of the most basic measures to study words in text is consider their frequency of occurrence. The dictionary as network allows the use of other measures, in particular, classical centrality measures in the literature: Degree, PageRank, betweenness centrality, and closeness centrality. As shown in Figure 3, each of them captures different features as they have little correlation. To give a taste of the results, we list in Table 2 "top" words for different measures previously mentioned.

Another productive topic of application is the search for similarities among words. To illustrate it we show that big (bidirectional) cliques, which are rare in a dictionary, are formed by words with similar meanings. In OPTED there is no K_6, seven K_5 (shown in Figure 4), 174 K_4 and 2,641 K_3. In DRAE no K_5, four K_4 and 243 K_3.

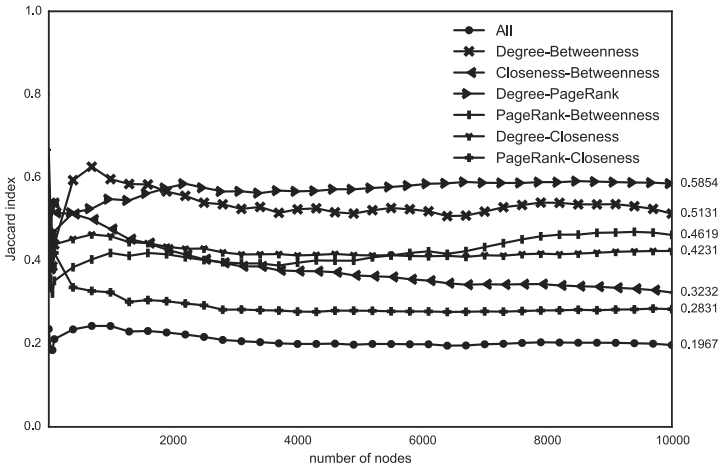


Figure 3: Common words of top rankings under different centralities, measured by Jaccard index ($\frac{|S_1 \cap S_2|}{|S_1| \cup |S_2|}$) for different number of nodes (0 to 10,000). For example the top ranked words for Degree and PageRank have 58.54% of their universe in common. All together they have 19.67% in common.

3.5 Core/Periphery Structure

Another feature that would help us understand the structure of dictionaries is the core/periphery characterization. This concept refers to the categorization of the nodes of the network. The nodes corresponding

Top Words OPTED

#	Deg	Pag	Clo	Bet
1	be	be	be	see
2	have	have	have	make
3	see	see	make	part
4	make	not	see	alt
5	use	make	part	form
6	pertain	manner	use	state
7	act	act	form	be
8	also	use	act	call
9	state	part	person	use
10	not	state	set	set
11	form	alt	call	take
12	part	person	state	act
13	call	thing	also	scale
14	alt	pertain	give	have
15	quality	place	take	manner
16	manner	form	point	point
17	person	word	run	body
18	place	certain	out	place
19	same	quality	place	line
20	body	time	right	give

Top Words DRAE

#	Deg	Pag	Clo	Bet
1	decir	algo	decir	hacer
2	persona	decir	ser	dar
3	otro	ser	persona	decir
4	ser	otro	otro	acción
5	tener	no	algo	estar
6	hacer	persona	tener	tener
7	algo	hacer	hacer	efecto
8	acción	tener	estar	persona
9	estar	cosa	cosa	medio
10	perteneciente	acción	dar	agua
11	relativo	estar	no	parte
12	no	dar	como	punto
13	cosa	como	más	cuerpo
14	efecto	efecto	parte	ser
15	como	relativo	acción	tiempo
16	parte	perteneciente	alguno	cosa
17	dar	pertenecer	medio	relativo
18	muy	parte	poder	derecho
19	más	poder	muy	mano
20	alguno	alguno	poner	estado

Table 2: Top words in OPTED and DRAE under diverse centrality measures: Degree (Deg), PageRank (Pag), Closeness (Clo), and Betweenness (Bet) Centrality. Note that there is a high degree of common notions among the top ranked words in the English and Spanish dictionaries.

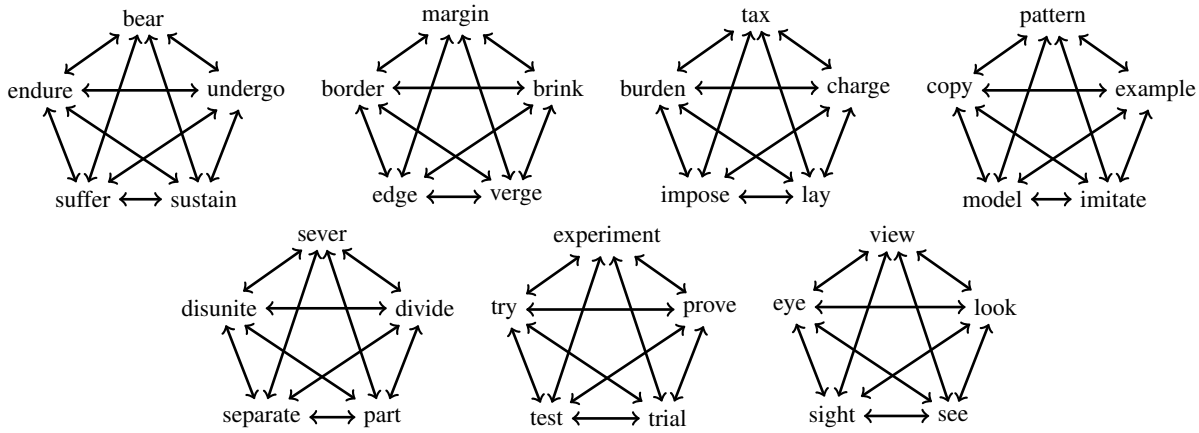


Figure 4: The only seven cliques of size 5 in OPTED (there are no bigger cliques). These words use each other in their definitions. Note their semantic closeness.

to the network core refers to a central and densely connected set. In the other hand, the periphery denotes a sparsely connected and non-central set of nodes that are linked to the core.

There are several types of core structures(Csermely et al., 2013): "traditional" core-periphery networks, rich-club networks, nested networks, bow-tie networks and onion networks. Intuitively, a dictionary network should follow one of these structures. The production of learner's dictionaries that uses a defining vocabulary to write all the definitions, or the simplification of languages through the definition of a small set of words(Ogden, 1930) supports this intuition. Unfortunately, to the best of our knowledge, there is no categorization of the core structure of dictionary networks.

4 Ogden's Basic English

Ogden's Basic English is an English-based controlled language created by Charles Kay Ogden in 1930. It is a simplified subset of the English language. According to Ogden, it is "a system in which everything may be said for all the purposes of everyday existence" (Ogden, 1930). This subset consists of 850

words[4]. The rationale of the choice of words is explained as follows (Ogden, 1930):

> The greater part of the words in use are shorthand for other words. Most common words are colored by our feelings, the words express judgment of our feelings in addition to their straight forward sense. It is generally possible to get to the factual level without much trouble.
> By putting the word to be tested in relationship with other possible words, questions can be framed in the form, "What word takes the place of the word in the middle in this connection?" Puppy is a Dog and time, young. Bitch is a Dog and sex, female. There are thirty lines for thirty sorts of questions.
> Questions of what a word will do for us has little relation to the number of times it is used in newspapers or letters.
> The number of 850 was found with 600 names of things, 150 are names of qualities, and the last 100 are the words which put the others into operation and make them do their work in statements.

Clearly the main arguments for the choice of the words are linguistic. In what follows, we will attempt to capture these words by purely graph-theoretical methods, thus shedding some light on the essential structure of Ogden's basic vocabulary in the network of the language. For our experiments we use the OPTED dictionary, that contains 803 words of the 850 of the Ogden's vocabulary.

4.1 Centrality Measures

The first naive hypothesis is that Ogden's set has good correlation with "central" nodes in the dictionary network. We investigated this with four classic centrality measures: *Degree* (most central nodes are those with higher number of adjacent nodes), *Closeness* (most central nodes are those that minimize the sum of the "distance" to other nodes in the graph), *Betweenness* (counts the number of shortest paths between all pairs of nodes passing through a given node), *PageRank* (essentially tells the number of steps taken to reach the node by a random walk starting from an arbitrary node).

We took the best k nodes for each centrality measure and every $0 < k \leq 803$, and checked how many of Ogden's words are in each of these sets. From the results (Figure 5) it follows that none of the centrality measures do a good job capturing Ogden's Basic English.

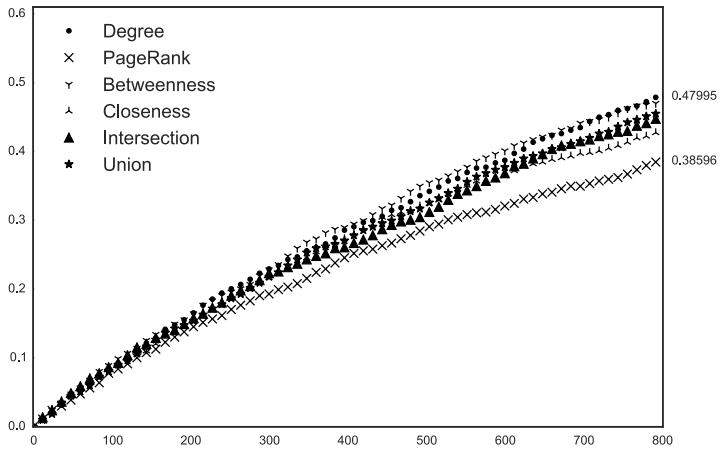


Figure 5: Ogden's Basic English words in top 800 words using different centrality measures. X-axis indicates k top-words and Y-axis, the percentage of Ogden's words in that set. Centralities by themselves are not a good method to capture the notion of importance that Ogden's Basic English represents.

The best measure in this task is degree centrality that captures almost 48% of Ogden. On the other hand, PageRank has the worst performance, capturing only 38.6%. In some sense we knew that degree centrality (which captures frequency) should perform poorly because Ogden stated explicitly that "what

[4]The list of the words can be seen in `http://ogden.basic-english.org/wordalph.html`

a word will do for us has little relation to the number of times it is used in newspapers and business letters". More surprising is the performance of PageRank, one of the most popular centrality measures today, used in multiple areas like ranking webpages, sense disambiguation (Mihalcea, 2004a), keywords and sentences from text (Mihalcea, 2004b), among others.

Group Centrality. Refining the idea, one could hypothesize that the problem is with *individual* centrality. The meaning of words is essentially a network property and not an individual one. There is an extended notion of centrality, called *group centrality* (Everett and Borgatti, 1999), that captures "centrality" of groups, not individuals. Unfortunately it is still not well developed, algorithmically.

We performed some experiments in this direction with groups of Ogden's words. We ranked Ogden's set using PageRank (seems the most promising to capture word senses (Abbott et al., 2012; Yeh et al., 2009)) and formed two groups, one with the top third and the other one with the bottom third of Ogden. As comparison and baseline, we extracted two sets of the same size from the set of words in the OPTED dictionary, one using the top nodes based on frequency, and the other one using a random selection. Results can be seen in Table 3.

	Degree	PageRank	Closeness	Betweenness
Ogden's	10 568	0.0310	0.5547	$4.06 \cdot 10^8$
Frequency	11 460	0.0314	0.5522	$4.40 \cdot 10^8$
Random	5 670	0.0129	0.5277	$2.10 \cdot 10^8$

(a) Top third set from 803 nodes (268 nodes).

	Degree	PageRank	Closeness	Betweenness
Ogden's	8 486	0.0157	0.5464	$2.95 \cdot 10^8$
Frequency	10 314	0.0199	0.5589	$3.41 \cdot 10^8$
Random	3 394	0.0097	0.5122	$1.34 \cdot 10^8$

(b) Bottom set from 803 nodes (268 nodes).

Table 3: Group Centrality for subsets of 803 words (nodes) chosen from three different sources: Ogden's set of words; selected from the OPTED dictionary by best frequency; chosen from OPTED at random.

For each of them we tested the four group centrality measures. Table 3 sheds some light on the existence of different types of roles in Ogden's set of words. The top third Ogden is rather aligned with classic centrality in the network (PageRank, many connections, in the middle of paths, etc.). On the contrary, the bottom third of Ogden behaves very much like random selection regarding PageRank and strongly diminishes its degree. This points to a role of covering an ample part of the network or being "spread" around the network. Though only slightly, this is further supported by the numbers of closeness. The closeness value of Ogden's bottom third is smaller than Ogden's top third (contrary to frequency that increases). The numbers are far from being conclusive due to the limits of the experimentation. As a baseline to compare to Ogden's top and bottom third, we had to use individual rankings, because we could not compute the actual (and ideal) group centralities due to lack of good algorithms and libraries (the problem is known to be NP complete (Garrido, 2016)).

In conclusion we state (although cannot explain well its rationale at this stage) that centrality measures inspired basically on social networks cannot be directly applied in this area. This points to the need for more sophisticated types of centrality measures for semantic networks (if the notion makes sense at all in the area), and in particular for dictionary networks.

4.2 Strong components of graphs

There are graph-theoretical notions about what the "core" (kernel) of a graph is, mainly using connectivity notions. For our dictionary network they seem promising under the hypothesis that connectivity (relationship) between and among groups of words is at the base of language.

We already saw in the component analysis (which holds for any network) that for our purposes one easily can get rid of more than 2/3 of the words in the OPTED dictionary by eliminating those words that are not used to define others (i.e. are "terminal" in some sense).

One can conduct a finer analysis as shown in Figure 6. From the whole OPTED network (which contains 803 words of Ogden) one can get the *strongly connected component* (SCC), those words that, by means of a cycle, are "used" in some sense to define themselves recursively. It has $23,360$ nodes and 802 of Ogden (99.87%). The discarded words (approx. 3/4 of the total) are those that either are terminal (not used to define other words) or n-th level terminals (and terminals after eliminating the terminals and so on).

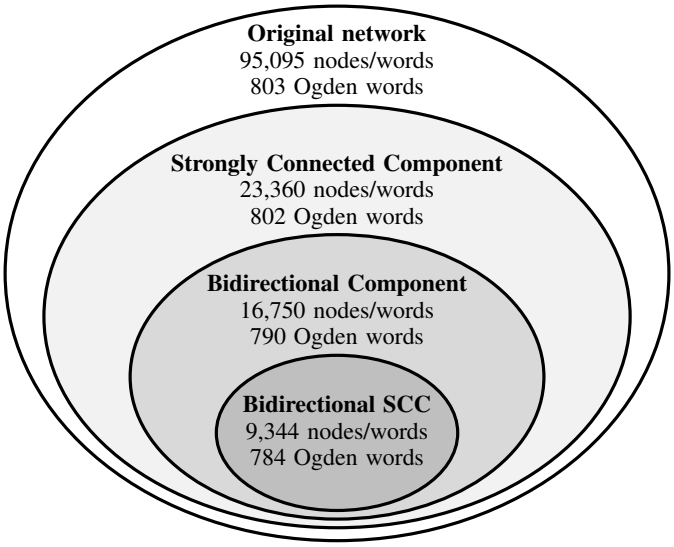


Figure 6: Connectivity analysis of components of OPTED network: The complete graph, Strongly Connected Component (words that recursively define themselves), Bidirectional Component (words that mutually need each other in order to be defined), and Bidirectional SCC. In the latter component only 3% of Ogden's words are lost), showing that Ogden's words strongly need each other.

Next, we consider a strong notion of connectivity: two words are connected if they are mutually used in the definition of the other (*e.g. fire* and *light*). Considering the subgraph induced by this relation, the Bidirectional Component (BC), one gets $16,750$ words, which contain 790 of Ogden (96.89%).

From here one can consider the biggest strongly connected component of BC (there are many small islands in BC), called BSCC in the figure, that has $9,344$ nodes and 784 words of Ogden (97.63%). This shows that Ogden is strongly correlated with these graph theoretical notions.

Picard et al.(2009) explored a notion of core (grounding kernel, which essentially recursively eliminates terminal words) and got a graph of 10% of the original graph. In size it matches our BSCC. Levary et al.(2012) used this notion in eXtended WordNet ($79,689$ nodes) and additionally collapsed synsets in one word, getting a core of $1,595$ nodes. In this core there are 314 Ogden words (52% of the part of Ogden they considered and 36.9% of total Ogden).

From these data, it seems that our BSCC is reaching the limit of the reduction of the English Dictionary (like OPTED) that can be obtained using only connectivity notions in order to capture most of Ogden's words (we are losing only 3% of all Ogden words). The challenge now is how to continue shrinking this graph while keeping most of Ogden's Basic English inside.

5 Conclusion

We provided evidence that dictionary networks share a common network structure and have a great potential to help understanding some properties of languages. We showed weaknesses and strengths of classical network notions in studying properties of dictionary/semantics networks. The results of this study highlight the need to devise more elaborated (than the classical ones) notions of centrality to understand and rank words and sets of words.

Acknowledgement The authors are funded by the Millennium Nucleus Center for Semantic Web Research under Grant NC120004. Garrido is also funded by CONICYT under grant CONICYT-PCHA/Doctorado Nacional/2015-21150149.

References

Joshua T. Abbott, Joseph T. Austerweil, Thomas L. Griffiths. 2012. Human memory search as a random walk in a semantic Network. Advances in Neural Information Processing Systems. 25, 3050-58, 2012.

Robert Alfred Amsler. 1980. The Structure of the Merriam-Webster Pocket Dictionary. PhD. Thesis. Department of Computer Sciences. University of Texas. Austin, Texas.

Vladimir Bagatelj, Andrej Mrvar and Matjaž Zaveršnik. 2002. Network analysis of dictionaries. University of Ljubljana, Institute of Mathematics, Physics and Mechanics. Dept. of Theoretical Computer Science. Preprint series, Vol. 40 (2002), 834.

Nicoletta Calzolari. 1984. Detecting Patterns in a Lexical Data Base. *Proceedings of the 10th International Conference on Computational Linguistics and 22nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1984.

Nicoletta Calzolari and Eugenio Picchi. 1988. Acquisition of Semantic Information From an On-Line Dictionary. *Proceedings of the 12th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 1988.

Elizabeth Costenbader and Thomas W. Valente. 2003. The stability of centrality measures when networks are sampled. *Social Networks*. 25 (2003) 283-307

Peter Csermely, András London, Ling-Yun Wu, and Brian Uzzi 2013. Structure and dynamics of core/periphery networks. *Journal of Complex Networks 1.2 (2013): 93-123*.

William Dolan, Lucy Vanderwende, and Stephen D. Richardson. Automatically deriving structured knowledge bases from on-line dictionaries. *Proceedings of the First Conference of the Pacific Association for Computational Linguistics*. 1993.

S. N. Dorogovtsev, J. F. F. Mendes. 2001. Language as an Evolving Word Web Proceedings of the Royal Society of London B: Biological Sciences. 2001, 268 2603-2606.

M. G. Everett and S. P. Borgatti. 1999. The Centrality of Groups and Classes. *Journal of Mathematical Sociology*, 23(3): 181-201.

Linton C. Freeman. 1978. Centrality in Social Networks. Conceptual Clarification. *Social Networks*. 1 (1978/79) 215-239.

Camilo Garrido, Ricardo Mora and Claudio Gutierrez. 2016. Group Centrality for Semantic Networks: a SWOT analysis featuring Random Walks. *ArXiv Preprints*, `https://arxiv.org/abs/1601.00139`

Thad Hughes, Daniel Ramage. 2007. Lexical semantic relatedness with random graph walks. EMNLP-CoNLL, 581-589.

Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. `http://snap.stanford.edu/data`

David Levary, Jean-Pierre Eckmann, Elisha Moses, and Tsvi Tlusty. 2012. Loops and Self-Reference in the Construction of Dictionaries. *Physical Review X*, 2, 031018 (2012).

Ken C. Litkowski. 1978. Models of the Semantic Structure of Dictionaries. *American Journal of Computational Linguistics*, Microfiche 81, Frames 25-74.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.

John Matta, Jeffrey Borwey and Gunes Ercal. 2014. Comparative Resilience Notions and Vertex Attack Tolerance of Scale-Free Networks. *ArXiv Preprints*, `http://arxiv.org/abs/1404.0103`

Rada Mihalcea, Paul Tarau and Elizabeth Figa. 2004. PageRank on Semantic Networks, with Application to Word Sense Disambiguation. *Proc. COLING '04*.

Rada Mihalcea and Paul Tarau. 2004. TextRank Bringing Order into Texts *Proc. Association for Computational Linguistics 2004*.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39-41.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. *Artificial Intelligence*, 193:217-250.

Mark Newman. 2003. The structure and function of complex networks. *SIAM review*, 45(2):167-256.

Charles Kay Ogden. 1930. *Basic English: A General Introduction with Rules and Grammar*, London, K. Paul, Trench, Trubner & Co.

Olivier Picard, Alexandre Blondin Massé, Stevan Harnard, Odile Marcotte, Guillaume Chicoisne, Yassine Gargouri. 2009. Hierarchies in Dictionary Definition Space. *23rd. Annual Conf. on Neural Information Proc. Systems: Workshop on Analyzing Networks and Learning With Graphs*, Vancouver BC, Canada, 11-12 Dec. 2009. Preprint *arXiv:0911.5703*.

Alain Polguère. 2014. From Writing Dictionaries to Weaving Lexical Networks. *Int J Lexicography*, (2014) 27 (4): 396-418.

Mark Steyvers and Joshua B. Tenenbaum. 2005. The Large-scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive Science*, 29 (1): 41-78.

Stanley Wasserman and Katherine Faust. 1994. *Social Network Analysis. Methods and Applications*, Cambridge Univ. Press, Cambridge, UK.

Yorick Wilks 1993. Providing machine tractable dictionary tools. *Semantics and the Lexicon*. Springer Netherlands, 1993. 341-401.

Yorick Wilks, Dan Fass, Cheng-ming Guo, James E. McDonald, Tony Plate, and Brian M. Slator. 1988. Machine Tractable Dictionaries as Tools and Resources for Natural Language Processing. *Proceedings of the 12th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics.

Eric Yeh, Daniel Ramage, Christopher D. Manning, Eneko Agirre, Aitor Soroa. 2009. Wikiwalk: random walks on wikipedia for semantic relatedness. Workshop on Graph-based Methods for NLP, 4149, Stroudsburg, PA, USA.

Nicola Zingarelli. 1970 Vocabolario della lingua italiana, a cura di Miro Dogliotti, Luigi Rosiello & Paolo Valesio. *Bologna: Zanichelli* (1970).

Structured Generative Models of Continuous Features for Word Sense Induction

Alexandros Komninos
Department of Computer Science
University of York
York, YO10 5GH
United Kingdom
ak1153@york.ac.uk

Suresh Manandhar
Department of Computer Science
University of York2
York, YO10 5GH
United Kingdom
suresh@cs.york.ac.uk

Abstract

We propose a structured generative latent variable model that integrates information from multiple contextual representations for Word Sense Induction. Our approach jointly models global lexical, local lexical and dependency syntactic context. Each context type is associated with a latent variable and the three types of variables share a hierarchical structure. We use skip-gram based word and dependency context embeddings to construct all three types of representations, reducing the total number of parameters to be estimated and enabling better generalization. We describe an EM algorithm to efficiently estimate model parameters and use the Integrated Complete Likelihood criterion to automatically estimate the number of senses. Our model achieves state-of-the-art results on the SemEval-2010 and SemEval-2013 Word Sense Induction datasets.

1 Introduction

Word Sense Induction (WSI) aims to automatically discover the different senses of polysemous words by unsupervised processing of text corpora. The related task of Word Sense Disambiguation (WSD) seeks to map the senses of word instances in a specific context to a predefined sense inventory. WSI overcomes the problem of having to define sense inventories, which may not have the appropriate granularity for all applications, and the effort of updating them for new domains or novel senses (Klapaftis and Manandhar, 2013). WSI is a challenging task that remains largely unsolved, but has important applications to a large number of tasks that require semantic processing of natural language (Navigli, 2009).

WSI is typically modelled as a clustering task, where the aim is to cluster samples of context representations of ambiguous words. Since context is the only available information to a WSI model, the choice of informative representations is a very important modelling aspect. Broad context related to topic or domain can restrict the possible senses that are applicable to an ambiguous word, but in order to make fine grained distinctions, context on the phrasal or syntactic level is usually needed. Ideally, a WSI system should incorporate different types of contexts to increase the confidence of its decisions. Combining the information present in different context representations can pose many difficulties in an unsupervised setting. Previous work has combined lexical with syntactic context (Brody and Lapata, 2009; Lau et al., 2012), and topical with local lexical context (Wang et al., 2015).

Another challenge for WSI systems is the need to apply clustering methods in high dimensional spaces of sparse features. Probabilistic latent variable models have been very successful in WSI by inducing latent representations of features that help improve generalization. While the latent variable approach has been very successful for word features, it has not provided considerable advantages when used with syntactic features (Brody and Lapata, 2009; Lau et al., 2012). A possible reason for this is that syntactic features, such as dependency contexts, exhibit much more sparsity than words.

A promising method to overcome the sparsity problem inherent in high dimensional discrete feature spaces is learning a low dimensional representation of the features. Word embeddings are an example of low dimensional vector representations for words learned in an unsupervised manner. The skip-gram

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3577–3587, Osaka, Japan, December 11-17 2016.

model (Mikolov et al., 2013) is a very popular technique for learning embeddings that scales to huge corpora and can capture important semantic and syntactic properties of words. Skip-gram embeddings exhibit compositional properties under addition, making them useful for constructing representations of phrases and larger units of text. Recently, the skip-gram model has been extended to learn embeddings of dependency context features (Levy and Goldberg, 2014; Komninos and Manandhar, 2016) that capture additional syntactic information. While word embeddings have been successfully used in many supervised NLP problems to overcome the problem of sparsity and improve generalization (Turian et al., 2010; Collobert et al., 2011), their application in WSI has been so far very limited.

In this paper, we propose a WSI model to address both the issue of multiple context representations and feature sparsity. Our model is a structured generative model that jointly models topical, phrasal and syntactic context in a hierarchical way. The probabilistic framework allows us to integrate different types of information in a principled way and also allows the application of model selection criteria to automatically determine the optimal number of senses. We address the issue of high dimensional feature spaces when dealing with syntactic features, by using word and dependency feature embeddings. In particular, we use the same skip-gram based model to create representations of all three context types.

We evaluate our model in two competitive benchmarks: SemEval-2010 Task 14: Word Sense Induction and Disambiguation (Manandhar et al., 2010), and SemEval-2013 Task 11: Word Sense Induction for graded and non-graded senses (Jurgens and Klapaftis, 2013). The two tasks provide different WSI evaluation frameworks and metrics. The proposed model achieves the state-of-the-art results in both datasets. Code is available at `https://cs.york.ac.uk/nlp/extvec`

2 Related Work

Among the most successful WSI systems are probabilistic latent variable models. Brody and Lapata (2009) extend the Latent Dirichlet Allocation (LDA) model (Blei et al., 2003) to combine evidence from different types of contexts. A limitation of this model is that the number of senses needs to be determined manually. Lau et al. (2012) propose using the non-parametric extension of LDA with a hierarchical dirichlet process (Teh et al., 2012) prior (HDP-LDA) to automatically estimate the appropriate number of senses from the data. They showed that HDP-LDA performs significantly better than LDA even when the number of senses is set to the same value. They also experimented with combined syntactic and word features. Syntactic features did not provide any advantage to either of these two LDA type models, but this could be attributed to sparsity. In addition, none of these models associate different context types with different stages of the generating process.

LDA type models assume that the topic distribution of context words correspond to different senses of the target word. Chang et al. (2014) proposed a model similar to LDA, but specifically tailored to WSI by estimating different latent variable distributions for context words and senses. In their setting, the latent topics for context words provide a method to overcome the sparsity problem related with the high dimensionality of the discrete feature space.

The sense-topic model (Wang et al., 2015) is another type of structured latent variable model related to our work. It makes a distinction between local and global context as we do, and jointly infers latent representations for both. The authors also make use of word embeddings as a method of feature weighting and for extracting additional context for ambiguous word instances. Contrary to our work, their features are discrete and syntax is ignored.

While several word embeddings models apply WSI in their training stage to create sense embeddings (Neelakantan et al., 2015; Iacobacci et al., 2015), there has been limited application of word embeddings as input representations to WSI models. The model of Huang et al. (2016) uses a recursive autoencoder to compose word embeddings to a context representation according to the structure provided by a syntactic parser. The final context representation captures both semantic and syntactic information and is used as the input to a rival penalization competitive learning clustering algorithm. While this approach uses both continuous word embeddings and syntactic information, it is fundamentally different from our work since their framework is not probabilistic. This makes difficult to incorporate additional contextual information like global context, and to define structural dependencies between context types.

A large variety of other clustering methods have been applied to WSI. A notable class of approaches is those using co-occurrence graphs and applying graph-based algorithms to identify hubs which are indicative of word senses (Klapaftis and Manandhar, 2010; Di Marco and Navigli, 2013).

3 Model Description

Target Word: operate
Global Context: ... while profits are volatile , many industries with volatile profits ranging from oil exploration to computer software *operate* without substantial government regulation . moreover , free markets generally work well for industries with large fluctuations , because ...
Local (win5) Context: from oil exploration to computer software without substantial government regulation
Syntactic Dependencies: advcl_volatile punct_, nsubj_industries nmod:without_regulation punct_.

Table 1: Global, local and syntactic context extracted for a target word.

We propose a generative model of continuous feature vectors that captures interactions between different types of contexts for a target ambiguous word. We separate context into three distinct types: global lexical, local lexical and syntactic context.

Global lexical context is indicative of the text's topic or domain, which can restrict coarse grained senses of a word. It can range between a few sentences around the target word or consider the whole document. In this work, we define global context as the words observed in the same sentence as the target word and one sentence before and after, as this is the typical context size provided by WSI evaluation datasets.

Local lexical context captures the semantics of phrases and is the most typically used context used by WSI systems. We define it as the words within a five word window before and after the target word.

Finally, we define the syntactic context of a target word as the typed dependencies with its neighbours in a dependency graph. This allows extraction of dependency context features such as $compound\_programming$, $compound^{-1}\_language$, where directionality is encoded by using an inverse dependency relation. Syntactic dependencies are a traditional word sense disambiguation feature as they capture the selectional preferences of a word.

An example of context selection can be seen in Table 1.

3.1 Continuous Context Feature Vectors

While our probabilistic model allows using the output of different models for each context type representation, e.g. a topic model for global context and a neural network for local, we use the publicly available Extended Dependency Skip-gram embeddings (Komninos and Manandhar, 2016) to construct representations for all three types. This skip-gram variant provides embeddings for both words and dependency context features trained jointly on Wikipedia text. Embeddings of dependency contexts have been used to incorporate syntactic information for sentence classification, and can be composed additively to model longer dependency relations.

Given the discrete features of the three context types, we create three continuous feature vectors by aggregating their corresponding embeddings. The operation to construct continuous context vectors is a weighted addition. We use the self information of discrete features to weight the contribution of each embedding to the overall feature vector:

$$I(x) = -log(P(x)) \tag{1}$$

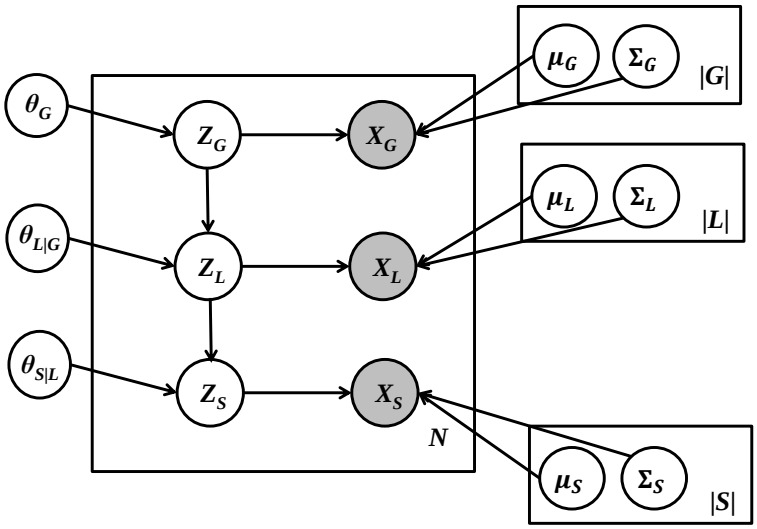


Figure 1: Graphical Representation of the model.

The three types of context feature vectors are formally defined by:

$$\boldsymbol{x_g} = \sum_{w \in G} I(w)\boldsymbol{v_w} \tag{2}$$

$$\boldsymbol{x_l} = \sum_{w \in L} I(w)\boldsymbol{v_w} \tag{3}$$

$$\boldsymbol{x_s} = \sum_{d \in S} I(d)\boldsymbol{v_d} \tag{4}$$

G,L and S are bags of discrete features corresponding to global, local and syntactic context as defined above, w and d are discrete word and dependency context features, $\boldsymbol{v_w}$ and $\boldsymbol{v_d}$ are embeddings of words and dependency contexts. We also apply L2-normalization to all feature vectors. The weighting by self-information reflects the intuition that rare words are more important for distinguishing senses than common words, and provides an automatic way of filtering the contribution of words without semantic content such as stop-words.

3.2 Probabilistic Generative Model of Context Feature Vectors

We infer the senses of an ambiguous word by combining information provided by the three context feature vectors within a structured generative model. The model assumes that there are three discrete latent variables for each target word z_g, z_l and z_s, each one generating one of the context feature vectors. The three latent variables form a hierarchical structure, where variables responsible for broader context generate a more context specific latent variable and their corresponding observed feature vector. The density of the context vectors is modelled by Mixtures of Gaussians. The full model is a structured generalization of Gaussian Mixture Models. Sampling context feature vectors follows the generative story:

For each target ambiguous word n*:*

select $z_{ng} \backsim Discrete(\boldsymbol{\theta_g})$
sample $\boldsymbol{x_{ng}} \backsim \mathcal{N}(\boldsymbol{\mu_{g=z_{ng}}}, \boldsymbol{\Sigma_{g=z_{ng}}})$
select $z_{nl} \backsim Discrete(\boldsymbol{\theta_{l|g=z_{ng}}})$
sample $\boldsymbol{x_{nl}} \backsim \mathcal{N}(\boldsymbol{\mu_{l=z_{nl}}}, \boldsymbol{\Sigma_{l=z_{nl}}})$
select $z_{ns} \backsim Discrete(\boldsymbol{\theta_{s|l=z_{nl}}})$
sample $\boldsymbol{x_{ns}} \backsim \mathcal{N}(\boldsymbol{\mu_{s=z_{ns}}}, \boldsymbol{\Sigma_{s=z_{ns}}})$

The corresponding graphical model can be seen in Figure 1. The parameters of the model are:

$$\boldsymbol{\Theta} = \{\boldsymbol{\theta_g}, \boldsymbol{\theta_{l|g}}, \boldsymbol{\theta_{s|l}}, \boldsymbol{\mu_g}, \boldsymbol{\Sigma_g}, \boldsymbol{\mu_l}, \boldsymbol{\Sigma_l}, \boldsymbol{\mu_s}, \boldsymbol{\Sigma_s}\} \tag{5}$$

We constrain the covariance matrices to be diagonal, hence having a smaller number of parameters compared to discrete mixture models for WSI.

3.3 Parameter Estimation

We estimate the parameters of the model by maximum likelihood estimation using the EM algorithm. The equations of E-step and M-step are the following:

E-step

For each sample $n \in \{1, .., N\}$ we compute:

$$\gamma(\boldsymbol{Z}) = p(\boldsymbol{Z}|\boldsymbol{X}) = \frac{p(\boldsymbol{Z}, \boldsymbol{X})}{\sum_{\boldsymbol{Z}} p(\boldsymbol{Z}, \boldsymbol{X})} \tag{6}$$

Since there are only three dependant latent variables per sample, we can easily compute this step by exact inference.

M-step

We estimate parameters that maximise:

$$\boldsymbol{\Theta^{new}} = E_{z|x}[\arg\max_{\boldsymbol{\Theta^{new}}} \log p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\Theta^{old}})] \tag{7}$$

Given the factorization implied by the graphical model, each parameter can be estimated independently. The update equations are:

$$\boldsymbol{\theta_g^{new}} = \frac{1}{N} \sum_{n,l,s} \gamma(z_{ngls}) \tag{8}$$

$$\boldsymbol{\theta_{l|g}^{new}} = \frac{\sum_{n,s} \gamma(z_{ngls})}{\sum_{n,l,s} \gamma(z_{ngls})} \tag{9}$$

$$\boldsymbol{\theta_{s|l}^{new}} = \frac{\sum_{n,l} \gamma(z_{ngls})}{\sum_{n,g,s} \gamma(z_{ngls})} \tag{10}$$

Updates for means and covariances for the g context type are given by:

$$\boldsymbol{\mu_g^{new}} = \frac{\sum_{n,l,s} \gamma(z_{ngls}) \boldsymbol{x_{ng}}}{\sum_{n,l,s} \gamma(z_{ngls})} \tag{11}$$

$$\boldsymbol{\Sigma_g^{new}} = \frac{\sum_{n,l,s} \gamma(z_{ngls})(\boldsymbol{x_{ng}} - \boldsymbol{\mu_g})(\boldsymbol{x_{ng}} - \boldsymbol{\mu_g})^T}{\sum_{n,l,s} \gamma(z_{ngls})} \tag{12}$$

and similarly for the l and s types.

We initialize means with the centroids of k-means run independently for each context type, and covariance matrices with the sample covariance.

3.4 Model Selection

One of the most challenging parts of WSI is estimating the number of senses for each ambiguous word type. Since we are working with a probabilistic model we can apply model selection criteria. We use the Integrated Complete Likelihood (ICL) criterion (Biernacki et al., 2000). ICL is a model selection criterion similar to the Bayesian Information Criterion (BIC) (Schwarz, 1978) that seeks a model that provides large evidence for the observed data with a small number of parameters. Following (McLachlan and Peel, 2004) we use the approximation:

$$ICL(m, \boldsymbol{X}) = \log p(\boldsymbol{X}|\boldsymbol{\Theta}) - \frac{m_k}{2} \log(N) + \sum_{n,g,l,s} \gamma(z_{ngls}) \log \gamma(z_{ngls}) \tag{13}$$

where m_k is the total number of parameters to be estimated.

This ICL approximation is exactly equal to BIC additionally penalized by the last term, which is the mean entropy of the distribution of latent variables. The extra penalty term favours models that result in more confident assignments since the entropy of the latent variable distribution will become lower in such cases. This behaviour favours well separated clusters and avoids strongly overlapping components which may be favoured in model selection by BIC. In our experiments, we set $|z_g| = |z_l| = |z_s| = K$ and train models with K in the range of [2, 50]. We observe that given enough training instances, ICL picks models with a large numbers of components corresponding to more fine-grained senses. This is reasonable since with more data the model becomes more confident into making such fine-grained distinctions, and is also likely to encounter unusual word usages.

4 Evaluation

We evaluate our model in two SemEval WSI datasets. For both datasets we parse the data using the Stanford Neural Network dependency parser (Chen and Manning, 2014) using Universal Dependencies (De Marneffe et al., 2014), which is the same format used by the dependency based embeddings. We train a different model for every word type.

4.1 SemEval-2010 Task 14: Word Sense Induction and Disambiguation

The SemEval-2010 WSI dataset consists of 50 verbs and 50 nouns. The task organizers provide a fixed training set with 879,807 instances of the target words. The distribution of instances for each word is highly imbalanced. The test set consists of 8,915 instances. Two types of evaluation are performed: supervised and unsupervised.

The supervised evaluation is performed in two steps. In the first step, a part of the data is used to map the induced sense clusters to a fixed inventory of word senses. In the second step, the fixed sense inventory is used to evaluate the clustering of the rest of the data as a Word Sense Disambiguation system. The reported metric is F-score. Following the task procedures we report two results, one using an 80-20 split of the data for mapping and scoring, and one using a 60-40 split. For both cases, reported results are an average over a 5-fold split.

Unsupervised evaluation for the SemEval task was performed by two clustering quality metrics, V-measure and paired F-score. A problem with clustering evaluation metrics is that they are sensitive to the number of senses (Klapaftis and Manandhar, 2013), with V-measure favouring a high number of senses and F-score the opposite. This behaviour results into ranking two uninformative baselines, 1-cluster-per-instance and most-frequent-sense (or all-in-one), as the best solutions. Li and Titov (2014) argue that for the V-measure, this behaviour can be explained by biases in the estimation of entropy when there is a large number of clusters compared to the number of samples, as is the case in WSI evaluation. They propose the usage of the Best-Upper-Bound (BUB) Entropy Estimator (Paninski, 2003) instead of maximum likelihood estimation that was used by the organizers. They show that the V-measure with the BUB estimator successfully evaluates both uninformative baselines as worse solutions than actual WSI systems. Following this recommendation, we report the V-measure estimated with BUB as the unsupervised evaluation metric.

Our default approach for assigning a sense to a word instance is taking the value of $p(z_s|\boldsymbol{x})$ as the probability of a sense being applicable, since it is the variable most directly associated with the ambiguous word. It is possible however, to assign senses to joint configurations of two or all three of the latent variables, in order to better use the information provided by our model. Since the number of possible configurations grows exponentially with the number of latent variables, this approach can lead to very fine-grained partitions of the data. In practice, we observe that only a small number of all these configurations are given a high probability mass. Evaluation results for our model MultiContextContinuous (MCC) and state-of-the-art systems are reported in Table 2. We report results with both the single variable approach (MCC-S) and joint variable assignments (MCC-L,S and MCC-G,L,S).

We see that our model achieves the highest score in both metrics, however, by utilizing different information. The combined evidence provided by the three latent variables helps induce an accurate mapping

Model	80-20	60-40	VM (BUB)
MCC-G,L,S	**70.6**	**69.0**	8.1
MCC-L,S	69.3	68.1	12.4
MCC-S	68.9	67.5	**17.1**
Hidden Concept (Chang et al., 2014)	69.7	68.9	-
HDP-LDA (Lau et al., 2012)	68.0	-	-
UoY (Korkontzelos and Manandhar, 2010)	62.4	62.0	11.4

Table 2: Results on the Semeval-2010 WSI dataset. Dashes indicate that this result was not reported by the authors of the corresponding model. UoY is the best performing model participating in the SemEval-2010 WSI evaluation.

of the sense clusters to the fixed sense inventory and results in the best F-score for the supervised evaluation. The supervised evaluation benefits from this fine-grained sense distribution since splitting the senses into smaller clusters does not affect the mapping operation, as long as the induced senses are consistent subsets of the gold standard senses. The 60-40 split results into a more difficult mapping problem and can indicate how reliable is the mapping between the sense distribution and the gold standard (Klapaftis and Manandhar, 2013). We see that using the joint configuration of the latent variables again results in the highest F-score, providing evidence of the consistency of the mapping. Contrary to the supervised evaluation, the V-measure favours the clustering provided by the z_s variable alone, since it penalizes the mismatch between the more fine-grained clustering and the gold standard.

4.2 SemEval-2013 Task 13: WSI for graded and non-graded senses.

The SemEval-2013 WSI evaluation dataset consists of 50 word types: 20 verbs, 20 nouns and 10 adjectives. There are several differences compared to the SemEval-2010 setting. There are less restriction on the training set, which can be any part or all of the UkWac corpus (Ferraresi et al., 2008). In addition, the test data include instances with multiple applicable senses. There are 4664 instances in the test set, 88.5% of which are labelled with a single sense, 11% labelled with two senses and 0.5% with three. Systems are asked to provide an estimate of the applicability of each sense.

The evaluation metrics also differ from those of SemEval-2010, in order to cope with the multiple sense labelling. Clustering evaluation is performed with the Fuzzy B-cubed and Fuzzy normalized Mutual Information (NMI) criteria. Fuzzy NMI favours solutions with many clusters giving a high score to the 1-cluster-per-instance baseline, while Fuzzy B-cubed favours few clusters and favours the all-in-one baseline. Following (Wang et al., 2015), we use those two metrics but also report the geometric mean of the two, which provides a more balanced metric and also assigns a score of zero to both the uninformative baselines. We use the top 3 most probable assignments of the z_s variable with the corresponding probability as the applicability weight. Results can be seen in Table 3.

We distinguish results for the standard test data provided by the SemEval task organizers and the augmented test data used in the evaluation of the sense-topic model. The first data augmentation result, indicated as "add-actual-context", uses an extra two sentences before and after the provided test data sentences, extracted by finding the test instances in their original corpus. The second data augmentation result "add-UkWac-context", uses context extracted from UkWac by finding word instances in a similar context as the test instances. Similarities are calculated by averaging word embeddings in the test instance and calculating cosine similarities. These context augmentation techniques improve the performance of the sense-topic model since the test data provided usually consist of a single sentence as the context of the target word. Since our model also considers global context, we expect that such data augmentation techniques would also increase its performance. However, we did not apply this approach in our evaluation setting and only used the provided context found in the SemEval test data. When using the actual SemEval test data, our model achieves the highest score in all three metrics.

Model	Fuzzy-NMI	Fuzzy B-cubed	Geom. Mean
MCC-S	**7.62**	**55.6**	**20.58**
Sense-Topic (Wang et al., 2015)	6.96	53.5	19.30
Sense-Topic (sim. weighted)	7.14	55.4	19.89
AI-KU (Baskaya et al., 2013)	6.5	39.0	15.92
unimelb (Lau et al., 2013)	6.0	48.3	17.02
Sense-Topic (add-actual-context)	9.39	**59.1**	**23.56**
Sense-Topic (add-UkWac-context)	**9.74**	54.5	23.04
1cl-per-inst	7.09	0	0
all-in-one	0	62.3	0

Table 3: Results on the SemEval-2013: Task 13 dataset. All Sense-Topic model variants are reported from (Wang et al., 2015). Results with extra context (add-actual-context, add-UkWac-context) do not use the same evaluation setting and are not directly comparable to the rest. AI-KU and unimelb are systems participating in the SemEval-2013 evaluation.

5 Discussion

We attribute the good performance of the proposed model to its capacity to handle data sparsity. Both the multiple context representations and the usage of low dimensional feature embeddings contribute towards that goal. The importance of dealing with sparse inputs can be seen from the generally good performance of Bayesian latent variable models for WSI (Wang et al., 2015; Chang et al., 2014) . While these models manage to deal with the sparsity of words, they still do not manage to effectively utilize syntactic features as we do with dependency feature embeddings. By using pretrained embeddings our model has access to a very large feature set (about 220k words and 1.3m dependency context features), while having to estimate a relatively small number of parameters. In Table 4, we show some examples where clusters are formed by different but semantically related words, and the importance of syntactic features.

A weakness of the proposed model is that context representations corresponding to senses are assumed to follow Gaussian distributions. We cannot expect this assumption to hold in general, but our evaluation suggests that it is a reasonable approximation. It is possible to extend the model by using a separate Mixture of Gaussians to model each individual sense. While this extension would provide additional capacity to the model for modelling sense specific contextual representations with complex probability densities, it can lead to parametric explosion and severe overfitting.

In Figure 2, we use t-SNE (Maaten and Hinton, 2008) to visualize the syntactic context vectors of the word "operate" from the SemEval-2010 training data and their sense assignments. We see that clustered points generally form compact groups, though they are not clearly separated. In order to model senses with contexts that do not follow a Gaussian distribution, the model favours additional clusters. In practice, we observe that this behaviour does not pose a significant problem as long as these finer-grained clusters are consistent with the underlying sense distribution, which was shown by the supervised evaluation of SemEval-2010. If a coarse grained distribution is desired, methods that merge Gaussian components as a postprocessing step could be considered (Hennig, 2010).

6 Conclusion

We propose a probabilistic latent variable model for Word Sense Induction. Our model integrates information from three different context types: global lexical, local lexical and dependency syntactic context. A different latent variable is inferred for each context type and dependencies are modelled in a structured top-to-bottom way, where broader context representations directly influence only more specific representations. Our context representations are constructed by weighted addition of word and dependency context embeddings that provide a way to overcome sparsity and reduce the number of parameters needed to be estimated. The number of senses is automatically determined by applying the Integrated Likelihood

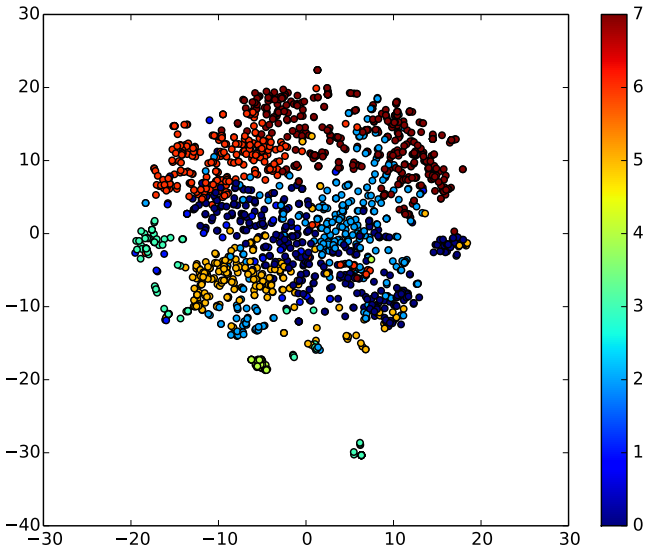


Figure 2: 2-d t-SNE of the syntactic context vectors of the word “operate”. Different colours correspond to different cluster assignments. Best viewed in colour.

Cluster 6
... energy-efficient appliances and how to *operate* them efficiently ...
... temperature rises enough for the heat pump to *operate* more efficiently than your old furnace ...
... software to enable users to *operate* their computers remotely ...
Cluster 5
... 44 of these stores *operate* as monro muffler brake & service ...
... many industries with volatile profits ranging from oil exploration to computer software *operate* without substantial government regulation ...
... bfx hospitality group , inc. owns and *operates* food services ...

Table 4: Instances of “operate” belonging into two different clusters. The contexts do not share many common words, but they are semantically related. The second instance of cluster 5 shares words with the third instance of cluster 6 (“software”, “computer”), but is assigned the correct sense because the syntactic features indicate the long range subject dependency with “industries”.

Criterion. We evaluate our model in two competitive WSI benchmarks, achieving state-of-the-art results.

Acknowledgements

Alexandros Komninos was supported by EPSRC via an Engineering Doctorate in LSCITS. Suresh Manandhar was supported by EPSRC grant EP/I037512/1, A Unified Model of Compositional & Distributional Semantics: Theory and Application.

References

Osman Baskaya, Enis Sert, Volkan Cirik, and Deniz Yuret. 2013. Ai-ku: Using substitute vectors and co-occurrence modeling for word sense induction and disambiguation. pages 300–306.

Christophe Biernacki, Gilles Celeux, and Gérard Govaert. 2000. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE transactions on pattern analysis and machine intelligence*, 22(7):719–725.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Samuel Brody and Mirella Lapata. 2009. Bayesian word sense induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 103–111. Association for Computational Linguistics.

Baobao Chang, Wenzhe Pei, and Miaohong Chen. 2014. Inducing word sense with automatically learned hidden concepts. In *COLING*, pages 355–364.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 1, pages 740–750.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. 2014. Universal stanford dependencies: A cross-linguistic typology. In *Proceedings of LREC*, pages 4585–4592.

Antonio Di Marco and Roberto Navigli. 2013. Clustering and diversifying web search results with graph-based word sense induction. *Computational Linguistics*, 39(3):709–754.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukwac, a very large web-derived corpus of english. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, pages 47–54.

Christian Hennig. 2010. Methods for merging gaussian mixture components. *Advances in data analysis and classification*, 4(1):3–34.

Yanzhou Huang, Deyi Xiong, Xiaodong Shi, Yidong Chen, ChangXing Wu, and Guimin Huang. 2016. Adapted competitive learning on continuous semantic space for word sense induction. *Neurocomputing*, 171:1475–1485.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: learning sense embeddings for word and relational similarity. In *Proceedings of ACL*, pages 95–105.

David Jurgens and Ioannis Klapaftis. 2013. Semeval-2013 task 13: Word sense induction for graded and non-graded senses. In *Second joint conference on lexical and computational semantics (* SEM)*, volume 2, pages 290–299.

Ioannis P Klapaftis and Suresh Manandhar. 2010. Word sense induction & disambiguation using hierarchical random graphs. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 745–755. Association for Computational Linguistics.

Ioannis P Klapaftis and Suresh Manandhar. 2013. Evaluating word sense induction and disambiguation methods. *Language resources and evaluation*, 47(3):579–605.

Alexandros Komninos and Suresh Manandhar. 2016. Dependency based embeddings for sentence classification tasks. In *Proceedings of NAACL:HLT*, pages 1490–1500. Association for Computational Linguistics.

Ioannis Korkontzelos and Suresh Manandhar. 2010. Uoy: Graphs of unambiguous vertices for word sense induction and disambiguation. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 355–358. Association for Computational Linguistics.

Jey Han Lau, Paul Cook, Diana McCarthy, David Newman, and Timothy Baldwin. 2012. Word sense induction for novel sense detection. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 591–601. Association for Computational Linguistics.

Jey Han Lau, Paul Cook, and Timothy Baldwin. 2013. unimelb: Topic modelling-based word sense induction for web snippet clustering. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*, pages 217–221.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL*, pages 302–308.

Linlin Li, Ivan Titov, and Caroline Sporleder. 2014. Improved estimation of entropy for evaluation of word sense induction. *Computational Linguistics*, 40(3):671–685.

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

Suresh Manandhar, Ioannis P Klapaftis, Dmitriy Dligach, and Sameer S Pradhan. 2010. Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 63–68. Association for Computational Linguistics.

Geoffrey McLachlan and David Peel. 2004. *Finite mixture models*. John Wiley & Sons.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*.

Liam Paninski. 2003. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253.

Gideon Schwarz. 1978. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2012. Hierarchical dirichlet processes. *Journal of the american statistical association*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Jing Wang, Mohit Bansal, Kevin Gimpel, Brian D Ziebart, and T Yu Clement. 2015. A sense-topic model for word sense induction with unsupervised data enrichment. *Transactions of the Association for Computational Linguistics*, 3:59–71.

Author Index